Université de Montréal

Méthodes et outils pour une affectation optimale des juges lors des compétitions : une application au concours John Molson

par
Amina Lamghari

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Octobre, 2008

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée:

Méthodes et outils pour une affectation optimale des juges lors des compétitions : une application au concours John Molson

présentée par:

Amina Lamghari

a été évaluée par un jury composé des personnes suivantes:

Jean-Yves Potvin,	président-rapporteur
Jacques A. Ferland,	directeur de recherche
Bernard Gendron,	membre du jury
Philippe Michelon,	examinateur externe
Jean-Yves Potvin,	représentant du doyen de la FES

Thèse acceptée le:

ABSTRACT

This thesis addresses the judge assignment problem during competitions. This problem is at the heart of the organization process. In fact, whenever competitions take place, judges have to be selected to evaluate the performance of the competitors, and to identify a winner. According to the competition context, specific rules must be followed in assigning the judges. In general, it makes sense to have an odd number of judges having complementary expertises to cover as exhaustively as possible all the expertises required to evaluate the performances of the competitors. Furthermore, conflicts of interest should be avoided. The problem to solve consists of determining which judges should be assigned to the various matches in such way to satisfy as far as possible the competition organizers objectives according to the restrictions. The number of constraints and their complexity make the problem difficult to solve. It is then useful to develop efficient solution methods capable of solving practical applications.

We illustrate the judge assignment problem in the particular case of the *John Molson International case competition* that takes place every year at Concordia University. More specifically, we consider two problems: *the problem for a round* and *the problem for several rounds*. The constraints taken into account, their nature, their complexity, as well as the horizon of the assignment are as many elements which distinguish these two problems. We propose mathematical models to formulate them, we develop methods to solve them, we analyze different variants of each method, and we compare their relative efficiency. Computational experiments indicate that these strongly combinatorial problems can be solved efficiently with the proposed heuristic methods. Although they apply first to the context of the *John Molson* competition, we think that the reach of these methods is beyond this specific context.

The solution approach summarized in this thesis was also tested using real data. Indeed, during the last three editions of the *John Molson* competition, we collaborated with the organizing committee to generate the judge schedules. Furthermore, one of the developed methods was embedded into an interactive software.

Keywords: Assignment, heuristics, metaheuristics, tabu search, variable neighborhood search, diversification.

RÉSUMÉ

Cette thèse aborde le problème d'affectation des juges lors des compétitions. Ce problème est au cœur du processus d'organisation de celles-ci. En fait, une compétition est une épreuve mettant en concurrence des joueurs ou des équipes qui se confrontent dans le cadre de rencontres ou de matchs en présence d'arbitres ou de juges. Ces juges sont chargés d'évaluer les performances des concurrents, d'établir un classement, et de désigner les vainqueurs. L'affectation des juges aux matchs doit tenir compte d'un ensemble de règles spécifiques au contexte de la compétition. En général, dans chaque match, il importe d'avoir un nombre impair de juges ayant des expertises complémentaires pour couvrir aussi exhaustivement que possible toutes les expertises requises pour évaluer les performances des concurrents. De plus, les conflits d'intérêt doivent être évités. Le problème qu'il faut alors résoudre consiste à déterminer quels juges assigner aux différents matchs de façon à satisfaire au mieux les objectifs fixés par les organisateurs de la compétition compte tenu des restrictions sur comment le faire. La multitude et la complexité des contraintes qu'il faut prendre en considération rendent ce problème difficile. Il est dès lors utile de concevoir des méthodes permettant de le résoudre efficacement.

Nous illustrons le problème d'affectation des juges dans le cas particulier du *concours international d'étude de cas MBA John Molson* organisé annuellement par l'Université Concordia. Plus précisément, nous considérons deux problèmes : *le problème sur une seule ronde* et *le problème sur plusieurs rondes*. Les contraintes prises en compte, leur nature, leur complexité ainsi que l'horizon de l'affectation sont autant d'éléments qui distinguent ces deux problèmes. Nous proposons des modèles mathématiques pour les formuler, nous développons des méthodes pour les résoudre, nous analysons différentes variantes de ces méthodes, et nous comparons leur efficacité relative. Les résultats des expérimentations numériques menées indiquent que ces problèmes fortement combinatoires peuvent avantageusement être résolus avec les méthodes heuristiques mises au point. Bien qu'elles s'appliquent en premier lieu au contexte du concours *John Molson*, nous pensons que la portée de ces méthodes dépasse largement le cadre spécifique dans lequel elles ont été développées.

L'approche de résolution résumée dans cette thèse a été également testée avec des données réelles. En effet, lors des trois dernières éditions du concours *John Molson*, nous avons collaboré avec le comité d'organisation pour établir les affectations des juges.

De plus, l'une des méthodes de résolution développées a été intégrée dans un logiciel interactif.

Mots clés: Affectation, heuristiques, métaheuristiques, recherche avec tabous, recherche à voisinage variable, diversification.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

REMERCIEMENTS

Je tiens d'abord à remercier profondément mon directeur de recherche, monsieur Jacques A. Ferland, pour son soutien constant, sa disponibilité, et la confiance qu'il m'a toujours témoigné. Je lui suis également très reconnaissante de m'avoir offert l'opportunité d'élargir mes connaissances et de travailler sur des problèmes pratiques. Merci, monsieur Ferland, de m'avoir guidée, motivée et encouragée tout au long de ces années. Merci pour vos conseils avisés et votre aide précieuse lors de la rédaction de ce manuscrit ; sans vous, je n'aurais jamais pu y arriver. Votre patience, votre bienveillance, votre grande compréhension, et votre gentillesse ont été et restent pour moi un exemple. Merci de m'avoir tant appris.

J'adresse également mes vifs remerciements aux membres du jury qui m'ont fait l'honneur d'accepter d'évaluer cette thèse : monsieur Jean-Yves Potvin qui a bien voulu présider ce jury, et messieurs Bernard Gendron et Philippe Michelon qui ont accepté de faire partie de mon jury de thèse.

Merci aussi à toutes les personnes avec qui j'ai collaboré et qui m'ont aidée à réaliser ce travail. Je remercie plus particulièrement messieurs Serge Bisaillon et Richard Simard pour leurs conseils judicieux lors des phases de programmation, leur gentillesse, et leur disponibilité. Un petit mot aussi pour Anne Drolet que j'ai côtoyée durant les premières années de ma thèse et dont l'aide et le soutien me furent très précieux.

Je tiens également à exprimer ma profonde reconnaissance à Gérard Parienti qui m'a régulièrement encouragée et poussée pour aller de l'avant. Merci Gérard pour ton amitié généreuse, ton soutien sans limite, et ces longues discussions qui m'aidaient à mettre de l'ordre dans mes idées. Merci de croire en moi. Tu as largement contribué à l'aboutissement de ce travail. Je remercie aussi chaleureusement Catherine Pettigrew et Marcela pour leur gentillesse, leurs encouragements, et leur soutien sans faille.

Enfin, je réserve une pensée toute particulière à ma mère, ma sœur Nawal, et mon beau-frère Ali qui m'ont encouragée et soutenue dans toutes les épreuves de la vie. Je te dois beaucoup maman et je ne pourrai jamais assez te remercier ; sans toi, rien de cela n'aurait pu se réaliser. Merci Nawal de m'avoir apporté joie et bonne humeur, surtout dans les moments difficiles. Merci pour ta grande fierté à mon égard qui m'a toujours poussée à persévérer. Merci Ali pour ta gentillesse, ton attention, et d'avoir transformé mes séjours au Maroc en vraie détente.

INTRODUCTION

L'affectation, qui peut être décrite comme un problème d'allocation de ressources à des activités sujet à des contraintes, occupe une place de choix dans la recherche opérationnelle. Les nombreux travaux consacrés à ce sujet depuis de nombreuses décennies témoignent de l'intérêt que lui accorde la communauté scientifique. Les problèmes d'affectation puisent tout leur intérêt en tant que modélisation mathématique d'un grand nombre de problèmes réels. En effet, leur champ d'application s'étend à des domaines aussi variés que la gestion, la production, les transports, la logistique, l'informatique ... Une liste d'exemples concrets ne pourrait en aucun cas être exhaustive.

Cette thèse aborde le problème d'affectation des juges lors des compétitions. Ce problème est au cœur du processus d'organisation de celles-ci. En fait, une compétition est une épreuve mettant en concurrence des joueurs ou des équipes qui se confrontent dans le cadre de rencontres ou de matchs en présence d'arbitres ou de juges. Ces juges sont chargés d'évaluer les performances des concurrents, d'établir un classement, et de désigner les vainqueurs. L'affectation des juges aux matchs doit tenir compte d'un ensemble de règles spécifiques au contexte de la compétition. En général, dans chaque match, il importe d'avoir un nombre impair de juges ayant des expertises complémentaires pour couvrir aussi exhaustivement que possible toutes les expertises requises pour évaluer les performances des concurrents. De plus, les conflits d'intérêt doivent être évités. Le problème qu'il faut alors résoudre consiste à déterminer quels juges assigner aux différents matchs de façon à satisfaire au mieux les objectifs fixés par les organisateurs de la compétition compte tenu des restrictions sur comment le faire.

Nous illustrons le problème d'affectation des juges dans le cas particulier du *concours international d'étude de cas MBA John Molson* organisé annuellement par l'Université Concordia. Plus précisément, nous considérons deux problèmes : *le problème sur une seule ronde* et *le problème sur plusieurs rondes*. Les contraintes prises en compte, leur nature, leur complexité ainsi que l'horizon de l'affectation sont autant d'éléments qui distinguent ces deux problèmes.

Voici la démarche qui a conduit aux deux problèmes auxquels nous nous intéressons.

Notre collaboration avec les organisateurs du concours *John Molson* a commencé suite à la demande d'un membre du comité d'organisation de ce concours en charge de générer les affectations des juges aux matchs. Celui-ci nous a décrit globalement le problème. Le format du concours est un tournoi à la ronde. Au cours de chaque ronde, plusieurs matchs ont lieu et, à chaque match, il faut affecter des juges pour évaluer les performances des équipes concurrentes. Cette affectation doit nécessairement prendre en compte de nombreuses contraintes telles que les disponibilités des juges, leurs compétences spécifiques ... Il est aussi souhaitable que l'affectation satisfasse à d'autres contraintes telles que la couverture des expertises dans chaque match et des contraintes inter rondes qui imposent des restrictions sur les affectations permises au fur et à mesure du déroulement des rondes. Habituellement, les affectations sont construites manuellement. La multitude des contraintes à considérer rend cette tâche difficile. De plus, cette difficulté se trouve accrue par un autre facteur intrinsèque au mode d'organisation du concours : les affectations doivent être générées dans des délais très courts. Il devenait donc indispensable de se doter d'outils pour faciliter cette tâche.

Nos travaux ont débuté par une analyse du problème tel qu'il nous a été décrit suite à ce premier échange. Dans un premier temps, l'élaboration des méthodes de résolution se fit sans tenir compte des interactions entre les rondes ; l'objectif étant de simplifier le problème. En effet, vu la structure du problème, il est facile de le décomposer en sous-problèmes, un sous-problème étant défini comme le problème d'affectation des juges aux matchs d'une seule ronde. Également, les contraintes inter rondes étant des contraintes liantes, en leur absence, les sous-problèmes deviennent indépendants et plus simples à résoudre. Ces considérations sont à l'origine du premier problème étudié dans cette thèse : *le problème sur une seule ronde.*

Dans un second temps, nous avons cherché à réintroduire les contraintes inter rondes que l'on avait éludées au début. Mais avant cela, il nous a semblé judicieux de recontacter les organisateurs du concours afin d'obtenir des informations plus précises sur le problème et de savoir s'il avait évolué depuis lors. Nous voulions mettre en œuvre des méthodes de résolution pratiques et utiles qui répondent à leurs besoins réels. Stimulés par les excellents résultats que nous avons obtenus, les organisateurs du concours nous

ont fait part d'autres besoins. Ainsi, outre les contraintes inter rondes, le problème s'est enrichi d'autres contraintes. Ceci a donné lieu au second problème abordé dans cette thèse : *le problème sur plusieurs rondes.*

Comme nous venons de le mentionner, les deux problèmes d'affectation étudiés dans cette thèse ont été dictés par des situations réelles. Les contraintes prises en compte correspondent aux règles d'affectation des juges aux matchs dans le contexte du concours *John Molson.* À notre connaissance, ces problèmes n'ont jamais été traités dans la littérature. Aussi, les objectifs de cette thèse sont-ils de proposer des modèles mathématiques pour les formuler, développer des méthodes pour les résoudre, analyser différentes variantes de ces méthodes, et comparer leur efficacité relative. Dans le cas des deux problèmes, nous nous intéressons exclusivement à des méthodes de résolution approchées. Bien qu'elles s'appliquent en premier lieu au contexte du concours *John Molson*, nous pensons que la portée des méthodes mises au point dépasse largement le cadre spécifique dans lequel elles ont été développées.

Modélisation. En raison des spécifités des problèmes auxquels nous nous intéressons et des exigences particulières qu'ils présentent, on peut noter d'emblée que la conception de modèles mathématiques pour les formuler n'a pas fait l'objet d'autres études. Un des objectifs de cette thèse consiste alors à modéliser ces deux problèmes. Les deux problèmes consistent à affecter un ensemble de juges à un ensemble de matchs tout en considérant un ensemble de règles reflétant les requêtes des organisateurs du concours. La quantification de ces différents aspects se modélise de manière générale par un problème d'optimisation combinatoire dont la solution donne des indications sur les juges à affecter à chaque match afin de satisfaire un ensemble de contraintes et d'optimiser une fonction objectif. Dans les deux cas, les modalités d'affectation déterminent les contraintes sur les décisions à prendre et les différents critères qui mesurent la qualité d'une solution.

Ainsi, *le problème sur une seule ronde* est d'abord formulé à l'aide d'un modèle de programmation linéaire en nombres entiers, puis résolu avec CPLEX. D'autres modèles sont également considérés et diverses méthodes sont conçues pour les résoudre.

En ce qui concerne le second problème, la multitude des contraintes à considérer augmente la complexité des modèles. Nous proposons d'abord un modèle de programmation à buts multiples (*goal programming*) qui sert de base à la résolution avec CPLEX. Nous déterminons de nouvelles linéarisations pour obtenir une formulation de programmation linéaire et nous proposons un nouvel algorithme pour calculer les poids associés aux différents objectifs. Nous présentons également d'autres modèles permettant d'étendre les méthodes de résolution proposées pour le premier problème.

Méthodes de résolution. Le second objectif de cette thèse concerne la conception de méthodes permettant de résoudre efficacement les deux problèmes auxquels nous nous intéressons. Nous avons privilégié les méthodes approchées aux dépens des méthodes exactes. Les raisons de cette préférence tiennent essentiellement à des impératifs pratiques. Les méthodes exactes garantissent certes l'optimalité des solutions, mais elles sont généralement limitées à la résolution d'instances de petite taille. Elles peuvent également s'avérer fortement pénalisantes en termes de temps de résolution. Pour les méthodes approchées, la perte du caractère optimal des solutions est compensée par la diminution des temps de résolution. Dans notre cas, le temps de résolution a une importance capitale puisqu'il est nécessaire de générer les affectations dans un délai relativement court, voire en temps réel. Nous avons donc opté pour les méthodes approchées. Les méthodes approchées offrent également d'autres avantages significatifs tels que la capacité de s'adapter facilement à des modifications du problème tels que l'ajout de contraintes, ce qui les rend parfaitement adaptées aux exigences de notre contexte.

Pour résoudre *le problème sur une seule ronde*, nous suivons deux voies. La première voie consiste à proposer des méthodes basées sur les heuristiques. La deuxième voie consiste à utiliser l'approche métaheuristique. Dans cette optique, nous proposons une adaptation de la méthode de recherche avec tabous (RT), une adaptation de la méthode de recherche à voisinage variable (RVV), et une approche de résolution hybride associant des techniques de la recherche avec tabous, de la recherche à voisinage variable, et des algorithmes génétiques (RTVS). Les

trois méthodes de résolution proposées (RT, RVV et RTVS) reposent principalement sur la métaheuristique de recherche avec tabous. Son efficacité à résoudre de nombreux problèmes difficiles ainsi que sa facilité d'adaptation nous a encouragé à l'utiliser pour la résolution de nos problèmes.

Pour résoudre *le problème sur plusieurs rondes*, nous introduisons une approche de résolution basée sur le principe de décomposition où les sous-problèmes associés aux différentes rondes sont considérés séquentiellement. Cette approche peut être assimilée à une recherche à voisinage variable. Succintement, le fait de changer de voisinage permet de passer d'un sous-problème à un autre. Cette approche permet premièrement, de considérer successivement des problèmes de petite taille, deuxièmement, de mieux gérer l'exploration du voisinage, et enfin et surtout, d'exploiter les résultats de notre première étude. Ainsi, les méthodes de résolution dédiées initialement au premier problème sont reprises en leur apportant toutefois certaines modifications pour intégrer les nouvelles contraintes.

Obtenir de bonnes performances passe généralement par une phase d'ajustement des diverses composantes de la méthode de résolution considérée. Nous mettons en évidence les différents changements et impacts induits par certaines modifications que nous avons apportées aux méthodes de résolution proposées dans le but d'accroître davantage leur performance. Ces modifications concernent essentiellement l'utilisation d'une mémoire à long terme permettant de mieux guider la recherche, la sélection de la solution voisine en cas d'égalité d'évaluation, et l'ordre dans lequel les solutions voisines sont considérées.

Variantes. Pour une même méthode de résolution, on peut envisager plusieurs variantes. Il est dès lors intéressant de vérifier si ces variantes obtiennent des résultats relativement similaires ou, au contraire, très différents. De plus, il est parfois possible que les choix effectués au niveau de certaines étapes de la méthode, notamment le choix de la solution initiale, ait plus d'influence sur les performances que la méthode elle-même.

Pour *le problème sur une seule ronde*, diverses variantes sont considérées pour

chacune des méthodes de résolution basées sur les métaheuristiques. Elles sont spécifiées en termes de la solution initiale à partir de laquelle ces méthodes débutent leurs recherches, et de la stratégie utilisée pour sélectionner la solution dans le voisinage de la solution courante.

En ce qui concerne *le problème sur plusieurs rondes*, l'approche de résolution constitue un cadre général qui, moyennant quelques restrictions sur les sous-problèmes à considérer et les solutions initiales à utiliser, donne lieu à différentes variantes.

Nous pensons que l'analyse effectuée a servi à faire ressortir les atouts et les faiblesses des différentes méthodes considérées, et a permis d'identifier, pour chaque méthode, quelle variante il vaut mieux utiliser pour obtenir les meilleurs résultats.

Comparaison. Comparer les méthodes de résolution et analyser leur comportement empirique est pertinent dans la mesure où cela permet de déterminer la méthode la mieux appropriée pour résoudre un problème donné. Nos méthodes de résolution ainsi que leurs différentes variantes sont testées sur des instances générées aléatoirement. Elles sont évaluées sur la base de la qualité des solutions obtenues et du temps de résolution nécessaire pour générer ces solutions, puis comparées d'abord avec CPLEX et ensuite entre elles. Nous comparons aussi les méthodes selon le degré de complexité des instances utilisées et effectuons une analyse statistique des résultats. En effet, il est généralement admis qu'aucune méthode n'est réellement meilleure qu'une autre pour toutes les instances bien que, pour certaines instances, certaines méthodes peuvent être plus adaptées que d'autres.

La comparaison avec CPLEX révèle l'intérêt des méthodes de résolution développées et confirme notre propos sur le fait que les méthodes exactes ne sont pas appropriées pour la résolution de nos problèmes. La comparaison des méthodes entre elles a permis d'identifier la méthode la plus performante et la plus efficace pour résoudre chacun des deux problèmes considérés.

Logiciel. Enfin, rappelons que ces travaux de recherche ont été motivés par le fait que les organisateurs du concours *John Molson* soient en quête d'un outil informatique

opérationnel leur permettant une aide lors de l'affectation des juges. Ainsi, lors des trois dernières éditions du concours, nous avons collaboré avec les organisateurs à établir les affectations des juges avec l'approche de résolution résumée dans cette thèse. De plus, l'une des méthodes de résolution développées a été intégrée dans un logiciel interactif.

Cette thèse est divisée en quatre chapitres. Le premier chapitre décrit, d'une manière plus détaillée, les deux problèmes sujets de notre étude. Nous y dévoilons leurs particularités, les objectifs visés ainsi que les différentes contraintes à respecter. Le second chapitre contient une revue de littérature sur quelques problèmes d'affectation. Nous faisons d'abord un tour d'horizon de certains problèmes classiques. Nous passons en revue brièvement les aspects théoriques afférents et les différentes approches envisagées pour leur résolution. Nous présentons ensuite quelques problèmes pratiques similaires aux nôtres. Nous exposons les principales modélisations et méthodes proposées dans la littérature pour leur résolution.

Nous abordons dans les deux chapitres suivants la résolution des deux problèmes auxquels nous nous intéressons. Le chapitre 3 présente nos travaux concernant *le problème sur une seule ronde* et le chapitre 4 décrit nos travaux concernant *le problème sur plusieurs rondes*. Nous donnons d'abord une description complète des modèles formalisant les problèmes considérés et les résultats obtenus avec CPLEX. Nous présentons ensuite les méthodes de résolution approchées que nous proposons. Pour chaque méthode de résolution, nous présentons d'abord le modèle mathématique qui sert de base à la méthode. Nous donnons ensuite tous les détails d'implémentation nécessaires à sa mise en œuvre, puis présentons des résultats numériques détaillés ainsi que des analyses du comportement de ses différentes variantes. La dernière partie de ces deux chapitres est consacrée à la comparaison des meilleures variantes des méthodes de résolution développées.

Enfin, la conclusion souligne les principales contributions de cette thèse. Nous donnons également des perspectives de recherche pour des travaux futurs.

CHAPITRE 1

DESCRIPTION DES PROBLÈMES ÉTUDIÉS

Dans cette thèse, nous abordons deux problèmes pratiques d'affectation des juges lors des compétitions. Ces problèmes ont des spécificités issues des besoins réels exprimés par les organisateurs du *concours international d'étude de cas MBA John Molson*. L'objectif de ce chapitre est d'**introduire et de décrire ces problèmes**.

Nous commençons par introduire le contexte général dans lequel se situe la problématique et présentons les traits communs aux deux problèmes étudiés (section 1.1). Nous décrivons par la suite les particularités de chacun de ces deux problèmes (sections 1.2 et 1.3). Ils sont présentés dans l'ordre chronologique où ils nous ont été soumis par les organisateurs du concours. L'étude du premier problème fera ensuite l'objet du chapitre 3 alors que le second sera traité dans le chapitre 4.

1.1 Le contexte

Le *concours international d'étude de cas MBA John Molson* est un événement organisé annuellement par l'Université Concordia. Il est ouvert aux meilleures écoles de gestion du monde entier et est reconnu comme le plus grand concours de ce type. Chaque école participante est représentée par une *équipe* formée de quatre étudiants. Le format du concours est un tournoi à la *ronde* constitué de cinq études de cas de gestion. En effet, les équipes sont partitionnées en groupes de 6 équipes et, à chaque ronde, chaque équipe rencontre une des autres équipes de son groupe. On parle alors de *match*. Suite au tournoi à la ronde, les neuf meilleures équipes sont sélectionnées pour les finales.

Au cours d'une ronde, les représentants d'une compagnie présentent simultanément aux équipes un cas réel rencontré. Les équipes proposent alors des solutions et des plans pour la compagnie. Dans chaque match, des professeurs universitaires et/ou des cadres dirigeants de compagnies sont présents en tant que *juges* pour évaluer la présentation des participants et désigner l'équipe gagnante.

L'affectation des juges aux matchs est régie par un ensemble de règles limitant les matchs auxquels un juge peut être affecté. Parmi ces règles, certaines sont à respecter strictement alors que d'autres sont à respecter «autant que possible». Les premières, que nous appelons *contraintes incontournables*, régissent la construction des affectations dans chaque match. Les secondes, que nous appelons *contraintes souples*, sont introduites dans un souci d'homogénéité et d'équité des affectations en considérant l'ensemble des matchs et/ou l'ensemble des rondes. Globalement, un nombre impair de juges, ayant des expertises complémentaires pour couvrir aussi exhaustivement que possible toutes les expertises requises pour évaluer les performances des concurrents, doit être affecté à chaque match. De plus, les conflits d'intérêt doivent être évités.

Les affectations des juges aux matchs étaient construites manuellement. Ceci représentait une tâche ardue pour les organisateurs du concours d'autant plus qu'elles doivent être générées dans un délai relativement court. En effet, si les juges et les équipes sont connus bien avant la tenue des compétitions, le tirage au sort déterminant les groupes et les matchs n'a lieu que dans la soirée précédant les compétitions. S'ajoute à cela la taille du problème : typiquement, 15 matchs ont lieu dans chaque ronde et le pool des juges disponibles dépasse la centaine. Aussi, certains aspects du problème n'étaient-ils pas pris en considération et la qualité des affectations construites laissait souvent à désirer. Nous avons cherché à développer des méthodes et des techniques d'optimisation qui permettent de générer, en temps raisonnable, des affectations respectant les contraintes incontournables du problème et garantissant une bonne couverture des autres besoins exprimés par les organisateurs du concours.

On distingue deux classes de problèmes selon l'horizon d'affectation considéré :

1. *le problème sur une seule ronde* consiste à générer les affectations des juges aux matchs d'une ronde ;
2. *le problème sur plusieurs rondes* vise à déterminer les affectations pour les matchs de l'ensemble des rondes.

La spécification des problèmes est liée non seulement à l'horizon d'affectation, mais aussi aux contraintes prises en compte pour générer ces affectations. En particulier, le

second problème intègre des contraintes additionnelles qui relient les rondes entre elles. Nous décrivons plus en détail les deux problèmes dans les sections qui suivent.

1.2 Le problème sur une seule ronde

Le problème introduit dans cette section est celui qui nous a été initialement présenté par les organisateurs du concours *John Molson*, en faisant toutefois abstraction des interactions entre les rondes comme cela a déjà été mentionné dans l'introduction.

Dans chaque match, 3 ou 5 juges sont affectés dépendamment du nombre de juges disponibles pour la ronde. Les organisateurs du concours ont recours à deux ensembles de juges : l'ensemble des *juges en chef* et l'ensemble des juges *autres*. Les *juges en chef* sont ceux qui possèdent les qualifications nécessaires pour présider le jury dans un match. Le reste des juges constitue l'ensemble des juges *autres*. Par ailleurs, six (6) différents domaines d'expertise des juges sont considérés. Chaque juge possède l'un de ces domaines d'expertise.

Les règles suivantes doivent être considérées lors de l'affectation des juges aux matchs :

Règles incontournables ou contraintes qui doivent être satisfaites :

1. un juge ne peut être affecté à un match impliquant une équipe représentant l'Université dans laquelle il a étudié ;
2. un juge ne peut être affecté à un match impliquant une équipe représentant l'Université à laquelle il est attaché ;
3. 3 ou 5 juges doivent être affectés à chaque match ;
4. au moins l'un de ces juges appartient à l'ensemble des *juges en chef.*

Règles souples ou contraintes (ou objectifs) à satisfaire autant que possible :

1. *contrainte de diversité* : les expertises des juges affectés à un même match doivent être aussi différentes que possible pour couvrir le plus grand nombre de domaines d'expertise ;
2. *contrainte du nombre* : le nombre de matchs avec 5 juges affectés doit être maximisé.

1.3 Le problème sur plusieurs rondes

Nous présentons maintenant le second problème traité dans cette thèse. Contrairement au problème précédent qui est limité au contexte d'une ronde, le problème consiste ici à déterminer l'affectation des juges aux matchs de toutes les rondes. Cette affectation devra prendre en considération deux types de contraintes : des contraintes locales régissant les affectations dans chaque ronde indépendamment des autres rondes, et des contraintes globales reliant les affectations dans les matchs des différentes rondes. De plus, les organisateurs du concours nous ont présenté d'autres facettes du problème qu'ils ne nous avaient pas spécifiées initialement. Le problème s'en trouve donc enrichi de nouvelles contraintes.

Comme nous l'avons déjà souligné à la section 1.1, au cours d'une ronde, chaque équipe affronte l'une des autres équipes de son groupe. Un cas est présenté aux concurrents, et les équipes disposent d'une période de trois heures pour analyser le cas et préparer leur présentation orale devant les juges. Chaque équipe spécifie *a priori* la langue (française ou anglaise) dans laquelle elle soumettra sa présentation. Cette information conduit à distinguer entre deux catégories disjointes de matchs : les matchs *francophones* et ceux *anglophones*. On dira qu'un match est *francophone* si au moins l'une des équipes en compétition dans ce match présente en français. Les autres matchs sont appelés matchs *anglophones*.

D'un autre côté, les organisateurs du concours classent les juges *autres* (ceux qui ne sont pas des *juges en chef*) en deux catégories disjointes : les juges *experts* et les *nouveaux* juges. Les juges *experts* sont ceux qui ont participé à l'une des éditions précédentes du concours alors que les *nouveaux* juges sont, comme leur nom l'indique, des juges nouvellement recrutés. Notons que tous les *juges en chef* sont *experts*, mais l'inverse est faux.

Finalement, chaque juge indique les rondes pour lesquelles il est disponible, les Universités dans lesquelles il a complété ses études (chaque juge peut avoir étudié dans plusieurs universités et non seulement une seule comme c'est le cas dans le problème présenté à la section 1.2), les Universités et/ou les compagnies auxquelles il est atta-

ché (là encore, chaque juge peut être attaché à plusieurs universités et non seulement une seule. Cependant, les organisateurs du concours considèrent qu'un juge est attaché à une compagnie au plus), s'il est *bilingue* (tous les juges sont *anglophones*, mais certains seulement sont également *francophones*), ses domaines d'expertise (chaque juge peut posséder plusieurs domaines d'expertise et non seulement un seul comme c'est le cas dans le problème présenté à la section 1.2), et s'il y a lieu, les équipes qu'il ne souhaite pas évaluer.

Les différentes contraintes qui régissent l'affectation des juges aux matchs sont décrites succinctement ci-après.

Règles incontournables ou contraintes qui doivent être satisfaites :

1. un juge ne peut être affecté à un match impliquant une équipe représentant une Université dans laquelle il a étudié ou à laquelle il est attaché (s'il est professeur). De plus, il ne peut être affecté à un match impliquant une équipe qu'il ne souhaite pas évaluer. Dans les deux cas, on dira que le juge est en *conflit* avec cette équipe. Notons que les organisateurs du concours prennent en compte au plus quatre (4) équipes avec lesquelles un juge peut être en *conflit* ;
2. un juge affecté à un match *francophone* doit être *bilingue* ;
3. 3 ou 5 juges doivent être affectés à chaque match ;
4. au moins un *juge en chef* doit être affecté à chaque match ;
5. au moins un juge *expert*, autre que le *juge en chef*, doit être affecté à chaque match.

Règles souples ou contraintes (ou objectifs) à satisfaire autant que possible :

1. *contrainte d'équilibre* : hormis le *juge en chef*, 2 ou 4 juges additionnels sont affectés à chaque match. Pour ces juges, le nombre des *experts* doit être égal au nombre des *nouveaux* ;
2. *contrainte d'affiliation* : si plusieurs juges représentant des compagnies sont affectés à un match, alors ils doivent être affiliés à des compagnies différentes ;

3. *contrainte de diversité* : les expertises des juges affectés à un même match doivent couvrir le plus grand nombre de domaines d'expertise ;
4. *contrainte de couplage* : une paire spécifique de juges ne peut être affectée plus d'une fois au cours de l'ensemble des rondes ;
5. *contrainte de séquence* : durant les différentes rondes, un juge ne doit pas être affecté à différents matchs impliquant la même équipe ;
6. *contrainte du nombre* : le nombre de matchs avec 5 juges affectés doit être maximisé.

1.4 Les problèmes tests

Les procédures de résolution développées dans cette thèse l'ont été dans le cadre de notre collaboration avec les organisateurs du *concours international d'étude de cas MBA John Molson* pour déterminer les affectations des juges au cours des trois dernières éditions de ce concours (2006, 2007 et 2008). Ceci nous a permis de bien identifier la structure du problème qui a évolué en fonction des possibilités offertes par les méthodes informatisées, et de considérer de plus en plus de contraintes réalistes pour mieux modéliser le problème. Du même coup, cette évolution a exigé d'adapter nos méthodes originales basées sur des heuristiques et de se tourner vers des métaheuristiques plus performantes.

Le concours *John Molson* comporte 5 rondes. Généralement, 30 équipes représentant 30 universités différentes, dont 3 *francophones*, participent au concours. Au total, 75 matchs sont organisés (15 matchs ont lieu dans chaque ronde), et le comité d'organisation a recours à environ 240 juges pour arbitrer ces matchs. Le nombre de juges disponibles diffère d'une ronde à l'autre. Comme il a déjà été souligné à la section 1.3, chaque juge est en *conflit* avec au plus 4 équipes (i.e., ne peut pas être affecté aux matchs impliquant ces équipes) et possède au moins l'un des 6 domaines d'expertise requis. Certains juges sont des professeurs universitaires tandis que d'autres sont des cadres dirigeants de compagnies.

Lors de nos collaborations avec le comité d'organisation du concours, nous avons été

en mesure de constater que les problèmes spécifiques dans le cadre du concours étaient faciles à résoudre par n'importe laquelle des méthodes que nous avons développées. Ceci est dû au fait qu'actuellement, le nombre de juges disponibles est élevé. De plus, la majorité de ces juges sont admissibles pour être affectés à la plupart des matchs sans aucun conflit d'intérêt. Néanmoins, le comité d'organisation a été entièrement satisfait puisque les affectations doivent être générées dans un délai très court, entre la soirée où le tirage au sort a lieu pour déterminer les groupes, et le matin suivant où les compétitions débutent. Afin d'analyser les performances de nos procédures de résolution et d'étudier leur capacité à trouver de bonnes solutions, nous avons décidé d'utiliser pour nos tests numériques des problèmes générés aléatoirement ayant la même structure que ceux du concours *John Molson*, mais dont la génération est biaisée pour obtenir des problèmes plus difficiles à résoudre. La méthodologie employée pour générer ces problèmes sera présentée dans les chapitres 3 et 4.

1.5 Conclusion

Dans ce chapitre, nous avons présenté les particularités des deux problèmes qui sont les sujets principaux de cette thèse. Pour les deux problèmes, il s'agit de déterminer un ensemble d'affectations respectant un ensemble de contraintes incontournables et satisfaisant le plus possible à un certain nombre de contraintes souples. Le second problème est plus complexe que le premier en raison de la nature et du nombre de contraintes à considérer.

Le chapitre suivant dresse un état de l'art de quelques problèmes classiques d'affectation largement étudiés dans la littérature, ainsi que certains problèmes issus de la pratique qui sont plus ou moins similaires aux problèmes auxquels nous nous intéressons.

CHAPITRE 2

ÉTAT DE L'ART

L'affectation est au cœur de nombreuses applications pratiques. Les situations concrètes qui y font appel sont très variées. Il n'est donc pas étonnant qu'on recense dans la littérature un nombre important de travaux de recherche se rattachant à ce sujet.

Faire un état de l'art exhaustif de tous ces travaux est une tâche colossale. Aussi, l'objectif de ce chapitre est-il de **brosser un tableau général des principaux modèles et méthodes de résolution développés pour certains problèmes ayant trait à l'affectation**, avant d'entamer la modélisation et la résolution des problèmes auxquels nous nous intéressons. Dans cette perspective, nous évoquons d'abord certains problèmes d'affectation classiques qui ont reçu une attention particulière de la part de la communauté scientifique, car ils recouvrent plusieurs applications. Pour chacun de ces problèmes, nous présentons le modèle général le formulant, puis nous passons brièvement en revue les méthodes les plus utilisées pour le résoudre ainsi que ses principaux champs d'application. Nous nous intéressons ensuite à des problèmes issus de contextes particuliers présentant certaines similarités avec le nôtre, plus spécifiquement, le problème d'affectation du personnel, le problème d'affectation des articles aux évaluateurs (*reviewers*), celui de la formation de groupes, et finalement le problème d'affectation des arbitres aux tournois sportifs. Nous exposons certaines modélisations et méthodes proposées pour la résolution de ces quatre problèmes.

Même en limitant l'état de l'art à une liste non exhaustive de problèmes, la littérature portant sur les problèmes que nous allons aborder dans ce chapitre est très riche. Il existe une panoplie de modélisations, de reformulations, et de méthodes de résolution développées pour ces problèmes. La présentation qui suit est donc très sommaire, et on pourra se référer à des travaux qui les traitent plus spécifiquement pour un approfondissement.

2.1 Exemples de problèmes d'affectation classiques

Mathématiquement parlant, un problème d'affectation peut être décrit comme suit : étant donné un ensemble d'activités et de ressources, le problème consiste à affecter les ressources aux activités tout en satisfaisant un ensemble de contraintes et en minimisant une fonction objectif. Une variable binaire indique si l'affectation d'une ressource à une activité a lieu ou pas. Il s'agit donc d'une catégorie de problèmes d'optimisation combinatoire renfermant plusieurs variantes. La nature de la fonction objectif et le type de contraintes permettent de caractériser ces variantes.

Certaines variantes ont suscité un intérêt particulier en raison de leur simplicité et du fait qu'elles constituent une formulation généralisée de plusieurs applications. Nous citons dans ce qui suit celles qui sont le plus étudiées.

2.1.1 Le problème d'affectation linéaire

C'est la variante de base du problème d'affectation. La fonction objectif est linéaire et les contraintes sont assez simples. Plus spécifiquement, ce problème est caractérisé par :

- le nombre de ressources est égal au nombre d'activités (n) ;
- chaque ressource est affectée à une activité unique ;
- chaque activité utilise une ressource unique ;
- un coût c_{ij} est associé à l'affectation de la ressource i à l'activité j.

Le modèle s'écrit comme suit :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Sujet à

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n$$

$$x_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,n,\ j = 1,\ldots,n$$

où,

$$x_{ij} = \begin{cases} 1 & \text{si l'activité } i \text{ est affectée à la ressource } j \\ 0 & \text{sinon.} \end{cases}$$

Ce problème peut être formulé comme un problème de couplage parfait de poids minimum dans un graphe biparti complet qui, à son tour, peut être ramené à un problème de flots. Il en résulte donc que le problème d'affectation linéaire peut être résolu en temps polynomial.

On recense dans la littérature un bon nombre d'algorithmes exacts développés pour résoudre ce problème classique. Ils peuvent être groupés en trois classes : algorithme primal-dual, purement primal et purement dual. En utilisant l'approche primale-duale, KUHN [63] obtint en 1955 la première méthode polynomiale (en $O(n^4)$). Il l'appela *méthode hongroise* en l'honneur au mathématicien hongrois DÉRES KONIG. La complexité de cette méthode a été réduite à $O(n^3)$ par LAWLER [69].

Plusieurs revues de littérature traitant du problème ont été également publiées, parmi lesquelles on retrouve celles de DELL'AMICO ET MARTELLO [22] et de DELL'AMICO ET TOTH [23].

2.1.2 Le bottleneck assignment problem

La différence entre ce problème (qui n'a pas d'appellation claire en français) et le problème précédent réside dans le fait que la fonction objectif est ici exprimée comme la minimisation du coût maximal. Plusieurs situations pratiques, comme celles présentées dans [37, 93], requièrent cette formulation plutôt que la minimisation de la somme des coûts. Le modèle s'écrit alors comme suit :

$$\min \max_{i,j} c_{ij} x_{ij}$$

$$\begin{aligned} \textbf{Sujet à} \quad & \sum_{i=1}^{n} x_{ij} = 1 && j = 1,\ldots,n \\ & \sum_{j=1}^{n} x_{ij} = 1 && i = 1,\ldots,n \\ & x_{ij} = 0 \text{ ou } 1 && i = 1,\ldots,n,\ j = 1,\ldots,n. \end{aligned}$$

Les premiers travaux sur le problème remontent à la fin des années 50 [52]. GARFINKEL ET RAO [41] démontrèrent que si le problème admet une solution réalisable, alors il admet une solution de base optimale. Ils développèrent ainsi un algorithme exact glouton basé sur la méthode du simplexe. PFERSCHY [89] présente une synthèse de certains algorithmes proposés dans la littérature pour résoudre le problème, et propose de nouvelles méthodes de résolution. Il présente aussi une analyse et une comparaison empirique de ces algorithmes.

2.1.3 Le problème d'affectation généralisé

Le problème d'affectation généralisé examine l'affectation à coût minimum de n ressources à m activités telle que chaque activité utilise une ressource unique, mais sujet à des restrictions de capacité sur les ressources. Les applications pratiques de ce problème sont nombreuses. Il apparaît dans de nombreux problèmes concernant le groupement, la classification, la planification, la confection d'horaire, et l'ordonnancement. Il permet également de modéliser des problèmes classiques comme le problème de la k-coloration et celui de la p-médiane. Le modèle s'écrit comme suit :

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

Sujet à

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,m$$

$$\sum_{j=1}^{m} a_{ij} x_{ij} \leq b_i \qquad i = 1,\ldots,n$$

$$x_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,n,\ j = 1,\ldots,m$$

où,

- a_{ij} représente la quantité de la ressource i nécessaire pour réaliser l'activité j
- b_i est la quantité totale disponible de la ressource i
- c_{ij} et x_{ij} sont tels que définis précédemment.

Ce problème peut être vu comme une généralisation du problème d'affectation linéaire présenté au paragraphe 2.1.1. En effet, la fonction objectif est similaire ; seules

les contraintes diffèrent. Son intérêt est la prise en compte de la contrainte de capacité sur les ressources ce qui est plus expressif, mais constitue toutefois une arme à double tranchant : si cette contrainte supplémentaire permet de modéliser des situations réelles, elle complexifie sensiblement le problème en le rendant NP-complet comme cela fut démontré par FISHER ET AL. [35].

Durant les deux dernières décennies, un effort considérable a été consacré à la mise au point d'algorithmes efficaces pour résoudre les problèmes de grande taille. Ces algorithmes se résument essentiellement à des procédures de séparation et évaluation progressive (*branch-and-bound*). Pour améliorer l'efficacité de la recherche, des techniques variées sont utilisées pour calculer les bornes permettant d'élaguer le plus tôt possible les branches conduisant à un échec. Parmi ces techniques, mentionnons les différentes relaxations lagrangiennes [31, 35, 60], la relaxation agrégée (*surrogate relaxation*) [61], ou encore une combinaison des deux [83]. Des heuristiques sont également utilisées pour guider le choix des variables et des valeurs durant l'exploration de l'arborescence. LORENA ET NARCISO [73] proposent et comparent six heuristiques basées sur un algorithme de sous-gradient. Ils rapportent que la combinaison des deux relaxations lagrangienne et agrégée, hybridée avec une heuristique constructive permet d'obtenir les meilleurs résultats. CATTRYSE ET VAN WASSENHOVE [17] font un tour d'horizon des différents algorithmes proposés dans la littérature pour résoudre le problème et fournissent une évaluation comparative de ceux-ci. Un algorithme de *branch-and-price* est proposé dans [98]. Mentionnons finalement l'algorithme plus récent de NAUSS [84] qui résout de plus grandes instances et qui est plus performant selon l'auteur. Il s'agit d'un algorithme de séparation et évaluation progressive utilisant des méthodes de coupes, la relaxation lagrangienne, des générateurs de solutions réalisables, et des méthodes de sous-gradient.

La littérature portant sur les méthodes de résolution approchées est également très riche. Parmi les méthodes proposées, on retrouve des heuristiques gloutonnes et des méthodes de recherche locale [78], des heuristiques basées sur des techniques de génération de colonnes [18], des heuristiques utilisant la recherche à profondeur variable (*variable depth search*) [3, 117], la recherche avec tabous [64, 85, 115], le recuit simulé [85],

les algorithmes génétiques [21], la métaheuristique GRASP et les algorithmes de colonies de fourmis [74]. Récemment, YAGIURA ET AL. [116] ont proposé un algorithme s'appuyant sur la recomposition du chemin (*path-relinking*) et les chaînes d'éjection. Ils rapportent que celui-ci est plus efficace que les autres algorithmes rapportés dans la littérature.

2.1.4 Le bottleneck generalized assignment problem

Ce problème peut être considéré comme une combinaison des deux problèmes présentés aux paragraphes 2.1.2 et 2.1.3. En effet, dans plusieurs situations pratiques, il faut prendre en considération des contraintes de capacité sur les ressources, et il est souvent plus approprié de minimiser le coût maximal plutôt que de minimiser la somme des coûts. Le modèle s'écrit alors comme suit :

$$\min \max_{i,j} c_{ij}x_{ij}$$

$$\begin{aligned}
\textbf{Sujet à} \quad & \sum_{i=1}^{n} x_{ij} = 1 && j = 1,\ldots,m \\
& \sum_{j=1}^{m} a_{ij}x_{ij} \leq b_i && i = 1,\ldots,n \\
& x_{ij} = 0 \text{ ou } 1 && i = 1;\ldots,n,\ j = 1,\ldots,m.
\end{aligned}$$

Ce problème fut introduit par FRANCIS ET WHITE [38] en 1974 dans un contexte de localisation de sites. Depuis lors, plusieurs auteurs s'y sont intéressés. À l'instar du problème d'affectation généralisé, ce problème est NP-complet. MAZZOLA ET NEEBE [79] ont développé un algorithme efficace pour le résoudre.

2.1.5 Le problème d'affectation quadratique

Le problème d'affectation quadratique est un problème classique d'optimisation combinatoire dans lequel il convient de trouver le placement optimal de n unités sur n sites de façon à minimiser un coût quadratique dépendant à la fois des distances inter-

sites et des flux inter-unités. Ce problème a de très nombreuses applications pratiques tant en placement qu'en planification, en informatique, en électronique ou en architecture. Il permet également de modéliser des problèmes classiques comme le problème du voyageur de commerce, le problème de partitionnement, et le problème de la clique maximale.

KOOPMANS ET BECKMANN [62] furent les premiers à introduire ce problème en 1957. Ils ont utilisé la formulation suivante :

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ij}b_{kl}x_{ik}x_{jl}$$

$$\textbf{Sujet à} \quad \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,\ldots,n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,n$$

$$x_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,n,\ j = 1,\ldots,n$$

où,

- a_{ij} représente le flux entre les deux unités i et j
- b_{kl} est la distance entre les sites k et l.

SAHNI ET GONZALEZ [96] ont prouvé que le problème d'affectation quadratique est non seulement NP-complet, mais fait partie du noyau dur de cette classe de complexité, dans le sens où même trouver une $\varepsilon-$approximation de la solution demeure NP-complet. Par ailleurs, CHRISTOFIDES ET GERRARD [20] et RENDL [95] ont dérivé des conditions sous lesquelles il peut être résolu en temps polynomial.

De nombreuses méthodes, tant exactes qu'approchées, ont été développées pour résoudre ce problème. Les méthodes exactes utilisent la programmation dynamique, les méthodes de coupes, et les techniques de séparation et évaluation progressive. Ces dernières semblent être les plus prometteuses. Il demeure toutefois que, malgré plus de 40 années de recherches, la taille des applications pouvant être résolues exactement en des temps raisonnables demeure très petite (autour de $n = 20$). Une littérature abondante est consacrée aux méthodes approchées : les méthodes constructives [12, 45], les méthodes

d'énumération [9, 13], la recherche avec tabous [102, 108], le recuit simulé [15, 113], les algorithmes génétiques [109], et les algorithmes de colonies de fourmis [40]. PARDALOS ET AL. [86] ont comparé certaines de ces méthodes dans le contexte particulier d'un problème de partitionnement.

Le parallélisme permet d'accélérer la recherche et d'atteindre ainsi des tailles de problèmes jusqu'alors inaccessibles par l'algorithmique séquentielle. En particulier, LI ET AL. [71] font état d'accélérations presque linéaires lors de l'implémentation parallèle de la métaheuristique GRASP. D'autre part, l'hybridation d'algorithmes permet également d'améliorer la qualité des solutions obtenues. FLEURENT ET FERLAND [36] ont proposé une métaheuristique fondée sur un algorithme génétique hybridé avec d'autres heuristiques, en particulier la recherche avec tabous et la recherche locale.

D'autres approches de résolution se basent sur une reformulation linéaire du problème [1, 8]. Pour ce faire, on introduit de nouvelles variables binaires pour simuler la présence du terme quadratique dans la fonction objectif. On ajoute également de nouvelles contraintes pour s'assurer que le nouveau problème est conforme au problème original.

Pour teminer ce paragraphe, mentionnons une excellente référence, la monographie [19], qui brosse un tableau complet des différents aspects reliés au problème d'affectation quadratique.

2.1.6 Le problème d'affectation multidimensionnel

Les paragraphes précédents ont présenté des problèmes d'affectation où il s'agissait d'associer, d'une manière optimale, les éléments de deux ensembles. Dans ce paragraphe, nous présentons une extension de ces problèmes à trois dimensions ou plus, qu'on appelle communément par le problème d'affectation multidimensionnel. PIERSKALLA [90] fut le premier à introduire ce problème en 1968. Il en a suggéré plusieurs applications. Voici la formulation mathématique qu'il utilisa pour le problème à trois

dimensions, connu aussi sous le nom de *axial three-dimensional assignment problem* :

$$\min \sum_{i=1}^{p} \sum_{j=1}^{q} \sum_{k=1}^{r} c_{ijk} x_{ijk}$$

Sujet à

$$\sum_{i=1}^{p} \sum_{j=1}^{q} x_{ijk} \leq 1 \qquad k = 1,\ldots,r$$

$$\sum_{i=1}^{p} \sum_{k=1}^{r} x_{ijk} \leq 1 \qquad j = 1,\ldots,q$$

$$\sum_{j=1}^{q} \sum_{k=1}^{r} x_{ijk} = 1 \qquad i = 1,\ldots,p$$

$$x_{ijk} = 0 \text{ ou } 1 \qquad i = 1,\ldots,p,\ j = 1,\ldots,q,\ k = 1,\ldots,r$$

où,

- c_{ijk} représente le coût d'affectation de l'activité i à la ressource j pendant la période k
- $x_{ijk} = \begin{cases} 1 & \text{si l'activité } i \text{ est affectée à la ressource } j \text{ pendant la période } k \\ 0 & \text{sinon.} \end{cases}$

Les algorithmes exacts proposés pour résoudre le problème d'affectation multidimensionnel sont pour la plupart des techniques de séparation et évaluation progressive utilisant des relaxations lagrangiennes pour identifier les bornes [7, 39, 77]. Toutefois, ce problème étant un problème NP-complet, diverses méthodologies heuristiques ont aussi été développées pour le résoudre. BALAS ET SALTZMAN [7] ont testé un certain nombre d'heuristiques similaires à celles utilisées pour trouver une solution initiale aux problèmes de transport. MAGOS [76] a proposé un algorithme basé sur la recherche avec tabous. Notons que la plupart des travaux de recherche sont dédiés au problème à trois dimensions. Néanmoins, plusieurs des résultats obtenus peuvent être naturellement étendus à des problèmes de plus grande dimension.

Plusieurs revues de littérature traitant du problème ont aussi été publiées. L'état de l'art, références et applications du problème sont décrits par GILBERT ET HOFSTRA [44]. BURKARD ET CELA [14] ont recensé la bibliographie afférente au sujet parue jusqu'en 1997. D'autres revues de littérature plus récentes sont proposées dans [87, 104].

2.2 Exemples de problèmes pratiques similaires au problème d'affectation des juges

Nous avons recensé dans la littérature certains problèmes pratiques qui sont plus ou moins similaires aux problèmes auxquels nous nous intéressons. Bien entendu, nous ne pouvons être exhaustif tant le domaine est large. Nous nous en tiendrons à quatre d'entre eux : le problème d'affectation du personnel, le problème d'affectation des articles aux évaluateurs, celui de la formation de groupes, et le problème d'affectation des arbitres aux tournois sportifs. Nous résumons dans cette section certaines méthodologies adoptées pour aborder ces quatre problèmes.

2.2.1 Le problème d'affectation du personnel

Le problème d'affectation du personnel est un problème classique en gestion des ressources humaines. Il a été étudié dans des contextes variés : militaire, industriel, médical, transport, académique... pour n'en citer que quelque uns. Certes, chaque contexte a ses propres impératifs et contraintes, mais, d'une façon générale, le problème peut être caractérisé comme suit :

- le besoin d'affecter un nombre restreint de personnes à un ensemble restreint de tâches ;
- la nécessité de satisfaire plusieurs contraintes relatives au nombre de personnes affectées à chaque tâche, le nombre de tâches assignées à chaque personne, ainsi que les attributs requis pour chaque tâche (qualification, aptitude...) ;
- l'objectif est d'affecter les personnes aux tâches de façon à satisfaire le plus possible les préférences de tout un chacun.

Du fait de la grande diversité des situations d'affectation, plusieurs modèles distincts sont proposés. Nous présentons ici un modèle général proposé par HILL ET AL. [58] :

$$\max \sum_{i=1}^{N} \sum_{j=1}^{M} u_{ij} x_{ij}$$

Sujet à

$$t_i \leq \sum_{j=1}^{M} x_{ij} \leq t_i' \qquad i = 1,\ldots,N$$

$$p_j \leq \sum_{i=1}^{N} x_{ij} \leq p_j' \qquad j = 1,\ldots,M$$

$$\sum_{i=1}^{N} a_{ik} x_{ij} \geq r_{jk} \qquad j = 1,\ldots,M, k = 1,\ldots,L$$

$$x_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,N, j = 1,\ldots,M$$

où,

- u_{ij} est une mesure de l'utilité de la personne i pour la tâche j
- t_i (respectivement t_i') est le nombre minimum (respectivement maximum) de tâches auxquelles la personne i peut être affectée
- p_j (respectivement p_j') est le nombre minimum (respectivement maximum) de personnes qui peuvent être affectées à la tâche j
- $a_{ik} = \begin{cases} 1 & \text{si la personne } i \text{ possède l'attribut } k \\ 0 & \text{sinon} \end{cases}$
- r_{jk} est le nombre de personnes ayant l'attribut k requis pour la tâche j
- $x_{ij} = \begin{cases} 1 & \text{si la personne } i \text{ est affectée à la tâche } j \\ 0 & \text{sinon.} \end{cases}$

Plusieurs méthodes ont été développées pour résoudre le problème d'affectation du personnel. HILL ET AL. [58] les classent en trois catégories : l'affectation aléatoire, les méthodes exactes, et les procédures heuristiques.

Affectation aléatoire. Cette approche est souvent utilisée en l'absence de contraintes, lorsque celles-ci sont relativement souples, ou encore lorsque les données sont assez coûteuses à collecter.

Méthodes exactes. Le problème de programmation des cours fut résolu efficacement

par SHIH ET SULLIVAN [101] en utilisant des codes classiques de programmation 0-1, en l'occurence *RIP30C*. LEE ET SCHNIEDERJANS [70] ont utilisé un modèle de programmation à buts multiples pour affecter des enseignants à des écoles privées. Leur objectif consiste à minimiser les coûts de transport tout en satisfaisant aussi bien les préférences des enseignants que celles des administrateurs. La même approche fut adoptée par REEVES ET HICKMAN [94] pour affecter des étudiants à des projets tout en respectant le plus possible leurs préférences et en garantissant une certaine qualité des équipes formées.

Certains auteurs considèrent le problème d'affectation de personnel sous un autre angle. Ils le reformulent dans un cadre de flot. Le principe consiste à construire un réseau dans lequel un chemin du nœud source au nœud destination respecte les contraintes du problème. LIANG ET THOMPSON [72] ont développé un modèle de type *capacitated transshipment* pour affecter le personnel de la marine. Dans le même ordre d'idées, HILL ET AL. [58] ont utilisé l'algorithme *out-of-Kilter* pour affecter des étudiants à des projets en tenant compte de leurs préférences. Ce même problème fut étudié par PROLL [91] lequel suggéra de résoudre le problème en deux étapes : la première nécessite la résolution d'un problème de type *bottleneck assignment problem* alors que la seconde implique la résolution d'un problème d'affectation linéaire.

Méthodes heuristiques. On distingue deux heuristiques de base : *people sequential heuristic* et *utility sequential heuristic*. La première heuristique ordonne toutes les personnes selon une certaine règle (rang, expérience...), et par la suite, affecte la première personne sur la liste à la plus haute tâche réalisable qu'elle préfère. Ce processus se poursuit séquentiellement pour toutes les personnes. La seconde heuristique affecte d'abord toutes les personnes à leur premier choix sans se soucier de la contrainte relative au nombre de personnes affectées à chaque tâche. Par la suite, les tâches pour lesquelles le nombre de personnes affectées excède le nombre souhaité sont reprises. Pour chacune de ces tâches, une personne est choisie aléatoirement, puis est réaffectée à son second choix. Ce processus se poursuit jusqu'à ce qu'un ensemble d'affectations réalisables soit trouvé. Ces deux heuristiques

partagent la même carence fondamentale suivante : elles peuvent ne pas trouver de solutions réalisables en présence de plusieurs contraintes.

2.2.2 Le problème d'affectation des articles aux évaluateurs

L'affectation des articles aux évaluateurs (*reviewers*) est une tâche habituelle dans la communauté scientifique. Elle constitue une partie importante des responsabilités des éditeurs de journaux, des comités de programme des conférences, et des conseils de recherches. Il est parfois difficile de s'acquitter de cette tâche notamment dans le contexte des conférences où le nombre d'articles soumis est élevé. C'est pourquoi, plusieurs chercheurs se sont intéressés au problème [10, 30, 51, 57, 97, 99, 106]. Bien qu'il présente quelques similarités avec le problème d'affectation du personnel présenté au paragraphe précédent, certaines particularités le distinguent de ce dernier.

Chaque article doit être affecté à un nombre précis d'évaluateurs [51, 57, 97, 99, 106], ou à au moins un certain nombre de ceux-ci [10, 30]. Afin d'éviter les conflits d'intérêt, un article ne doit pas être évalué par l'un de ses co-auteurs, ou par des amis ou des ennemis de ceux-ci [10, 30, 51, 99, 106]. De même, dans certains contextes, le nombre d'articles affectés à un même évaluateur ne doit pas dépasser une limite maximale [10, 51, 57, 97, 106]. Il est également souhaitable que la répartition de la charge de travail soit équilibrée entre les évaluateurs [10, 30, 97, 99, 106], et que chacun d'entre eux reçoive les articles qui correspondent le mieux à ses préférences [10, 30, 51, 97]. Finalement, quoiqu'une conférence soit consacrée à un thème précis, les articles soumis peuvent couvrir de nombreux domaines. Les évaluateurs auxquels serait affecté un article donné doivent donc, autant que possible, être reconnus pour leur expertise dans les domaines traités dans cet article. Dans [30, 99], les auteurs proposent certaines mesures pour quantifier cet aspect.

GOLDSMITH ET SLOAN [51] présentent une variété de critères d'optimalité qui peuvent être considérés et analysent la complexité des problèmes qui en résultent. Ils montrent que si le seul critère à optimiser est la satisfaction des préférences des évaluateurs, alors le problème fait partie de la classe P. En effet, dans ce cas, il est possible de le formuler comme un problème de flot à coût minimum lequel peut être résolu en

temps polynomial. Néanmoins, lorsqu'il faut aussi tenir compte des expertises des évaluateurs, le problème devient plus complexe. Les auteurs montrent que, dans ce cas, le problème peut être vu comme une variante du *stable marriage problem.* Ils soulignent les différences qui existent entre le problème d'affectation des articles aux évaluateurs et le *stable marriage problem* classique, différences qui rendent parfois le premier problème NP-complet.

Plusieurs problèmes ont toutefois déjà été résolus exactement. La méthode à deux phases proposée par HARTVIGSEN ET AL. [57] constitue un cadre de résolution exact pour des problèmes où il s'agit d'affecter chaque article aux évaluateurs les plus experts possible pour cet article. Durant la première phase, le niveau d'expertise de chaque évaluateur pour chaque article est quantifié en résolvant une série de problèmes de transport. Les affectations sont déterminées au cours de la seconde phase en utilisant les niveaux d'expertise obtenus lors de la première phase. Pour ce faire, une variante du *capacited transshipment problem* est formulée, puis résolue avec des techniques d'optimisation exactes propres à ce problème.

Dans le problème étudié par SAMPSON [97], l'objectif est, d'une part, de satisfaire les préférences des évaluateurs et, d'autre part, d'équilibrer la charge de travail entre eux. Le problème est d'abord formulé comme un modèle de programmation à buts multiples, puis reformulé comme un problème de flot à coût minimum qui est résolu avec une méthode exacte classique appropriée.

DUMAIS ET NIELSEN [30] visent également à satisfaire les préférences des évaluateurs. Ils proposent une méthode appelée «*n of 2n*» qui consiste à affecter aux évaluateurs deux fois plus d'articles que ce qu'on leur demande d'évaluer en réalité leur permettant ainsi de choisir une partie de leur charge eux mêmes.

L'approche heuristique proposée par SCHIRRER ET AL. dans [99] repose sur un algorithme mémétique. Les solutions initiales sont générées en utilisant une heuristique constructive gloutonne. La recherche locale est basée sur le remplacement d'un évaluateur par un autre pour un article donné. Les auteurs rapportent que cette approche s'est avérée plus avantageuse que l'approche exacte basée sur la reformulation du problème en un problème de flot à coût minimum, dans le sens où elle produit des solutions de

bonne qualité très rapidement. Mentionnons que les objectifs considérés par les auteurs sont d'affecter chaque article aux évaluateurs les plus experts pour cet article et d'équilibrer la charge de travail entre les différents évaluateurs.
Finalement, pour répondre à ces mêmes objectifs, la solution proposée par SUN ET AL. [106] consiste à résoudre d'abord un problème de transport pour affecter chaque article aux évaluateurs les plus experts pour cet article, puis à ajuster par la suite les affectations afin d'équilibrer la charge de travail entre les évaluateurs. Pour ce dernier point, les auteurs proposent deux solutions : soit «embaucher» d'autres évaluateurs, soit réaffecter les articles en surplus aux évaluateurs auxquels sont assignés peu d'articles. Ils ne mentionnent toutefois pas les techniques utilisées pour y arriver.

2.2.3 Le problème de formation de groupes

Le problème de formation de groupes a été abondamment étudié dans la littérature [5, 6, 11, 80, 94, 110–112], notamment dans le contexte académique. Typiquement, on dispose d'un nombre de personnes dont chacune est considérée comme une ressource ayant ses propres caractéristiques. Suivant le contexte, ces caractéristiques peuvent refléter diverses données (démographique, tranche d'âge, expérience, cursus universitaire...). L'objectif est d'affecter ces personnes à des groupes de façon à bien répartir les caractéristiques entre les groupes et éviter la concentration d'une caractéristique uniquement dans certains groupes.

Plusieurs modèles sont proposés pour aborder ce problème. La principale différence entre ces modèles est liée à la nature de la fonction objectif. En effet, certains auteurs raisonnent au niveau local cherchant à ce que chaque groupe soit le plus hétérogène possible. D'autres optent pour une approche plus globale et mettent l'emphase sur l'homogénéité de l'ensemble des groupes. Ainsi, les premiers s'intéressent à maximiser la diversité au sein d'un même groupe tandis que les autres visent à minimiser les différences entre les divers groupes. Pour une étude et une évaluation comparative des différentes fonctions objectifs proposées dans la littérature dans le contexte particulier de la création de groupes d'étudiants, l'article de BAKER ET POWELL [6] est à consulter.

Voici quelques modélisations du problème de formation de groupes que l'on peut

retrouver dans la littérature :

Modèle quadratique. Ce modèle a été proposé par WEITZ ET LAKHSHMINARAYANAN [112] dans un contexte académique :

$$\max \sum_{j}\sum_{i}\sum_{l>i} d_{il}x_{ij}x_{lj}$$

$$\textbf{Sujet à} \quad \sum_{i=1}^{N} x_{ij} = S \qquad j = 1,\ldots,M$$

$$\sum_{j=1}^{M} x_{ij} = 1 \qquad i = 1,\ldots,N$$

$$x_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,N,\ j = 1,\ldots,M$$

où,

- d_{il} mesure la différence entre les étudiants i et l
- S est le nombre d'étudiants dans chaque groupe ($S = \frac{N}{M}$)
- $x_{ij} = \begin{cases} 1 & \text{si l'étudiant } i \text{ est affecté au groupe } j \\ 0 & \text{sinon.} \end{cases}$

Modèle de programmation à buts multiples. MINGERS ET O'BRIEN [80] ont proposé un modèle de programmation à buts multiples pour affecter des étudiants à des groupes. La même approche a été adoptée par BAKER ET BENN [5] pour affecter des élèves à des classes dans une école secondaire. Chaque étudiant présente un certain nombre de caractéristiques. L'objectif consiste à créer des groupes le plus similaires possible, toutes caractéristiques confondues. Un objectif additionnel est considéré dans [5] : ne pas séparer certains étudiants lors de l'affectation. Comme leur nom l'indique, les deux modèles visent à minimiser la somme des déviations des objectifs visés.

Modèle de flot. L'approche décrite ci-après a été proposée par BHADURY ET AL. [11] pour maximiser la diversité des membres d'un groupe de travail. Les auteurs se sont basés sur le *dining problem*, lequel est un problème peu connu de la théorie des flots. Ce problème appartient à la classe des problèmes de partitionnement [2, 11]. Il s'énonce comme suit : n familles veulent aller dîner. Chaque famille i est

composée de a_i membres. Le restaurant dans lequel ils veulent se rendre compte m tables. Chaque table j a une capacité c_j. Afin de maximiser les interactions sociales entre elles, les familles désirent développer un arrangement de telle sorte que le nombre maximum des membres de la même famille installés à la même table soit aussi petit que possible.

Pour modéliser ce problème, on construit un graphe contenant : un nœud source S, un nœud F_i pour chaque famille i, un nœud T_j pour chaque table j, et un nœud destination D. Pour les arcs, on considère un arc de capacité a_i de S vers F_i, un arc de capacité c_j de T_j vers D, et un arc de capacité d^* de F_i vers T_j où, d^* représente le nombre maximal des membres de la famille F_i assis à la table T_j. Le flot n'est rien d'autre que le nombre des membres des familles qu'il faut asseoir aux tables. Pour adapter le *dining problem* au problème de formation de groupes de diversité maximale, BHADURY ET AL. [11] suggèrent de répartir les personnes en familles selon leurs caractéristiques, et de considérer les tables comme étant les groupes.

Nous mentionnons maintenant, à titre indicatif, le principe global des méthodes proposées dans la littérature pour résoudre les modèles susmentionnés.

Le premier modèle est quadratique et par conséquent NP-complet. Il serait ambitieux d'opter pour une méthode exacte pour le résoudre. L'utilisation d'une approche heuristique pour le résoudre est donc largement justifiée [112]. La même approche a été adoptée par BAKER ET BENN [5] pour résoudre leur modèle de programmation à buts multiples. On distingue deux procédures heuristiques de base : les procédures de construction et celles d'amélioration.

Procédures de construction. La procédure est initialisée avec des ensembles vides de groupes. L'affectation est ensuite construite en ajoutant les étudiants aux groupes, un à la fois.

WEITZ ET JELASSI [110] choisissent aléatoirement le premier étudiant et l'affectent au premier groupe. Ensuite, ils identifient l'étudiant le plus similaire à celui-ci et le placent dans le second groupe. La procédure se poursuit ainsi jusqu'à ce que tous les étudiants soient affectés.

MINGERS ET O'BRIEN [80] proposent également une procédure de construction, mais qui est plus efficace et plus rapide que la précédente [6, 80]. Il s'agit d'une méthode gloutonne où les étudiants non encore affectés sont partitionnés en listes suivant leurs caractéristiques. Ces listes sont par la suite ordonnées selon la rareté des caractéristiques (ordre décroissant). La procédure considère successivement les différentes listes. À chaque étape, chaque étudiant non encore affecté de la liste est testé pour chaque groupe en tant que solution partielle. L'étudiant qui permet d'obtenir la meilleure affectation est retenu pour le groupe en question, et bien entendu, supprimé de la liste des étudiants non affectés. La procédure se poursuit ainsi jusqu'à ce que tous les étudiants soient affectés.

Procédures d'amélioration. À l'encontre des heuristiques de construction, la procédure est initialisée avec une affectation complète des étudiants générée soit aléatoirement, soit en utilisant l'une des procédures de construction décrites ci-dessus. Par la suite, une paire d'étudiants est choisie soit parmi tous les groupes [112], soit en restreignant le choix à un ensemble spécifique de groupes [5]. Les deux étudiants sélectionnés sont permutés et l'impact sur la fonction objectif est mesuré. L'échange n'est retenu que si une amélioration est enregistrée. Les tests d'échange se poursuivent jusqu'à ce qu'un critère d'arrêt soit vérifié. WEITZ ET LAKSHMINARAYANAN [112], qui brossent un tableau comparatif des différentes procédures rapportées dans la littérature, proposent différentes alternatives pour le critère d'arrêt. Ils rapportent que celui qui consiste à arrêter le processus lorsqu'aucune amélioration n'est possible quel que soit l'échange, permet d'obtenir les meilleurs résultats. Ce même critère d'arrêt a été retenu dans [6].

Finalement, le modèle de flot a été résolu efficacement en temps polynomial. Pour ce faire, BHADURY ET AL. [11] résolvent successivement des problèmes de flot maximum.

2.2.4 Le problème d'affectation des arbitres aux tournois sportifs

Le problème d'affectation des arbitres aux tournois sportifs n'est pas un problème formellement défini comme l'est le problème de formation de groupes. En effet, les ar-

ticles traitant du sujet se concentrent généralement sur des cas particuliers impliquant des disciplines et des ligues sportives bien précises (baseball [32, 33], basket-ball [33], cricket [114], football [43, 118]). Dès lors, les règles et les exigences particulières imposées par le contexte considéré rendent chaque problème d'affectation unique. Toutefois, d'une manière générale, nous considérons qu'un problème d'affectation des arbitres aux tournois sportifs consiste à déterminer une affectation d'un ensemble d'arbitres à un ensemble de matchs organisés dans le cadre d'un tournoi. L'horaire et le lieu des matchs sont prédéterminés. Les principes propres aux disciplines sportives ainsi que les règles qui régissent ces tournois sont autant de contraintes à prendre en compte pour déterminer les affectations des arbitres aux matchs. Voici une liste de certaines contraintes typiques considérées dans la littérature :

- nombre d'arbitres à affecter à chaque match. Ce nombre est fixe et dépend de la discipline sportive considérée ;
- indisponibilité de certains arbitres à certaines dates [27–29, 114] ;
- restrictions sur les matchs auxquels les arbitres peuvent être affectés [27–29, 33, 114] ;
- nombre minimal et/ou maximal de matchs auxquels les arbitres peuvent être affectés [27–29, 33, 114, 118], ou encore, le nombre maximal de matchs consécutifs qu'ils peuvent arbitrer [114] ;
- restrictions sur le nombre de fois qu'un arbitre peut être affecté aux matchs impliquant la même équipe [25, 32, 33, 114, 118] ;
- considérations concernant la réalisabilité des déplacements des arbitres (décalage horaire, horaire des vols programmés par les compagnies aériennes ...) [32, 33] ;
- pré-requis minimaux de compétences pour certains matchs ou certaines positions dans les matchs [27–29, 33, 114].

Le problème d'affectation des arbitres aux tournois sportifs peut être mono-critère ou multicritères. Voici quelques fonctions objectifs que l'on retrouve dans la littérature :

- minimiser les coûts liés au déplacement des arbitres [33] ;
- minimiser l'écart entre le nombre de matchs auxquels est affecté chaque arbitre et une valeur cible prédéterminée [27–29] ;

- minimiser les coûts liés au déplacement des arbitres et équilibrer le nombre de matchs auxquels ils sont affectés [32] ;
- minimiser la somme des violations des contraintes souples [118] ;
- outre la minimisation de la somme pondérée des violations des contraintes souples, d'autres objectifs additionnels sont considérés dans [114] dont notamment, la minimisation de la distance des déplacements.

Alors qu'il paraît trivial d'affecter les arbitres aux matchs d'un tournoi, en présence d'exigences additionnelles comme celles que nous avons citées plus haut, le problème devient un problème d'optimisation combinatoire très difficile à résoudre. Dans [29], les auteurs présentent une preuve formelle que la version de décision du problème auquel ils s'intéressent est NP-complet. Dans [118], les auteurs soulignent certaines similarités entre le problème qu'ils considèrent et des problèmes classiques NP-complets.

La quasi-totalité des travaux sur le problème d'affectation des arbitres aux tournois sportifs portent sur sa résolution par des méthodes approchées. Pour affecter les arbitres aux matchs de basket-ball de l'*Atlantic Coast Conference*, EVANS ET AL. [33] utilisent une approche basée sur une décomposition du problème permettant d'utiliser un algorithme de flot à coût minimum. Il ne s'agit toutefois pas d'une méthode exacte puisque les affectations sont générées séquentiellement, et la solution trouvée à une étape donnée dépend fortement de celles obtenues au cours des étapes précédentes.

Dans le but de planifier l'affectation des arbitres aux matchs de la ligue américaine de baseball, EVANS [32] présente un système interactif d'aide à la décision utilisant des techniques d'optimisation, des règles heuristiques, et des jugements humains. Toutefois, l'auteur n'explique pas clairement le fonctionnement de la méthode de résolution utilisée, si ce n'est qu'elle se résume à la résolution d'une séquence de problèmes d'affectation linéaire. Il fait plutôt une description détaillée de la structure du système informatique développé.

WRIGHT [114] a également conçu un sytème informatique pour affecter les arbitres aux matchs de cricket en Angleterre. Là encore, l'auteur ne fournit pas les détails des algorithmes utilisés. Il mentionne que la stratégie de résolution qu'il utilise combine deux phases. Durant la première phase, une solution initiale respectant les contraintes incon-

tournables du problème est identifiée sans toutefois tenir compte de sa qualité. Cette solution est améliorée au cours de la seconde phase en utilisant une méthode d'amélioration itérative fondée sur deux types de modifications : échanger l'affectation de deux arbitres et remplacer un arbitre par un autre. Le procesus itératif se poursuit jusqu'à ce qu'un optimum local soit atteint.

DINITZ ET STINSON [25] se sont quant à eux intéressés à la conception de tournois équilibrés (*balanced tournament design*). Ils montrent que ce problème peut être résolu en utilisant une variante du *Room squares*, en l'occurence le *maximum empty subarray Room squares*, et formulent des conditions suffisantes d'existence d'une solution.

DUARTE ET AL. [27–29] ont introduit une approche heuristique à trois phases pour résoudre un problème qui, selon les auteurs, est commun à plusieurs ligues et disciplines sportives. Dans [27], lors de la phase 1, une heuristique gloutonne construit une solution initiale. Dans le cas où la solution générée n'est pas réalisable, une deuxième heuristique est utilisée au cours de la phase 2 pour la réparer. Une heuristique de recherche locale, employée lors de la phase 3, sert à améliorer la qualité de la nouvelle solution. Dans une version étendue de cet article [29], les auteurs mentionnent que les phases 2 et 3 sont basées sur une recherche locale itérée (*iterated local search*). Elles utilisent d'abord une recherche locale, puis exécutent un nombre d'itérations qui débutent toutes par une procédure de peturbation pour générer une nouvelle solution initiale. Par la suite, une recherche locale est appliquée pour obtenir un optimum local, et la méthode itère entre une procédure de perturbation et une recherche locale jusqu'à ce qu'un critère de réinitialisation soit satisfait. Les voisinages considérés au cours de la recherche locale sont similaires à ceux utilisés dans [114]. Ils consistent à échanger l'affectation de deux arbitres et remplacer un arbitre par un autre. Enfin, dans leur approche présentée dans [28], DUARTE ET AL. utilisent les mêmes trois phases susmentionnées, mais plutôt que d'utiliser une recherche locale au cours des phases 2 et 3 (suite à la procédure de perturbation), un modèle de programmation en nombres entiers est formulé, puis résolu par des méthodes d'optimisation classiques. Les auteurs rapportent que cette hybridation entre les méthodes exactes et les heuristiques permet d'améliorer significativement les performances de l'algorithme de résolution. Ils soulignent que la nouvelle approche surclasse

l'approche utilisée précédemment dans [29].

Finalement, l'article publié en 2008 parYAVUZ ET AL. [118] aborde l'affectation des arbitres aux matchs de football de la ligue Turque. Bien que le problème considéré nécessite l'affectation de quatre arbitres à chaque match, le modèle et la méthode de résolution proposés par les auteurs ne concernent que l'affectation d'un seul arbitre. La procédure de résolution comprend deux phases. Une solution initiale est générée lors de la première phase par l'entremise d'une heuristique constructive. L'approche utilisée consiste à résoudre une série de problèmes d'affectation linéaire obtenus en limitant l'horizon de l'affectation. Au cours de la seconde phase, une procédure de recherche locale est utilisée dans le but d'améliorer progressivement la solution en main jusqu'à ce qu'un optimum local soit atteint. La méthode d'amélioration est basée sur l'échange de l'affectation de deux arbitres. Les auteurs suggèrent deux stratégies pour gérer l'évaluation du voisinage de la solution courante. La première consiste à évaluer le voisinage au complet tandis que la seconde restreint l'évaluation à une portion du voisinage : seuls les échanges tels que l'un des arbitres viole l'une des contraintes du problème sont considérés. Les deux stratégies peuvent être utilisées séparément ou conjointement. Les auteurs rapportent que la deuxième alternative permet d'obtenir de meilleurs résultats.

Pour conclure, notons que le problème d'affectation des arbitres aux tournois sportifs s'inscrit dans la catégorie des problèmes d'optimisation des sports, laquelle a suscité durant les dernières décennies un intérêt croissant de la part des chercheurs. Pour un aperçu d'ensemble et une classification de la littérature afférente au sujet, nous référons le lecteur à la monographie [16] et au site web :
`http://www.informatik.uni-osnabrueck.de/knust/sportlit_class/`.

2.3 Conclusion

Dans ce chapitre, nous avons présenté quelques problèmes d'affectation classiques et nous avons résumé les principales approches utilisées pour les résoudre. Ce tour d'horizon a été complété par un survol des modèles et méthodes de résolution adoptés pour aborder certains problèmes d'affectation pratiques.

Ces problèmes, *a priori* différents, ont tous un point en commun : il s'agit d'affecter un ensemble de ressources à un ensemble d'activités tout en satisfaisant un ensemble de contraintes et en minimisant une fonction objectif. Ils se modélisent donc par des problèmes d'optimisation combinatoire. La spécification de la fonction objectif et des contraintes dépend des exigences particulières du contexte d'affectation, et donne lieu à des modèles différents destinés majoritairement à des situations bien précises. Il existe toutefois des modèles génériques pouvant servir à la formulation de plusieurs problèmes.

Les méthodes développées pour résoudre ces problèmes sont multiples. Elles sont classifiées en méthodes exactes et méthodes approchées. Les méthodes exactes font un parcours exhaustif de l'ensemble des solutions et garantissent l'obtention d'une solution optimale, mais au prix d'un effort de calcul important et de délais conséquents. Les méthodes approchées ne considèrent qu'une partie de cet ensemble et fournissent ainsi, en un temps raisonnable, la meilleure solution rencontrée. Rien ne garantit toutefois que les régions de l'ensemble des solutions non considérées lors de cette recherche partielle ne contiennent pas la solution optimale. Le recours aux méthodes approchées est largement justifié. En effet, un grand nombre de problèmes étudiés dans la littérature sont NP-complets. Par conséquent, la résolution d'une instance de taille comparable à celles rencontrées dans la pratique par les méthodes exactes est inappropriée. De plus, souvent dans la pratique, l'objectif n'est pas d'obtenir systématiquement une solution optimale, mais plutôt d'obtenir une solution de bonne qualité en un temps minimal.

La résolution exacte passe le plus souvent par une reformulation mathématique du problème sous forme d'un programme linéaire ou d'un graphe, pour le solutionner ensuite avec des méthodes de résolution traditionnelles comme des procédures de séparation et évaluation progressive ... Le développement d'un algorithme exact nécessite donc un effort de modélisation considérable.

Quant à la résolution approchée, plusieurs méthodes sont utilisées allant des heuristiques les plus simples (heuristiques constructives intuitives, algorithmes gloutons, méthodes d'amélioration itérative ...) aux métaheuristiques les plus sophistiquées (recherche avec tabous, algorithmes génétiques, recuit simulé, colonies de fourmis, algorithmes hybrides ...). Le développement d'une méthode approchée efficace requiert une connaissance ap-

profondie du problème traité.

Certaines approches présentées dans ce chapitre seront reprises et adaptées pour résoudre le problème étudié dans le chapitre suivant : *le problème sur une seule ronde.*

CHAPITRE 3

ÉTUDE EXPÉRIMENTALE DU PROBLÈME SUR UNE SEULE RONDE

Ce chapitre s'intéresse au *problème sur une seule ronde* introduit à la section 1.2. Notre objectif est de **développer, d'évaluer, et de comparer des méthodes permettant de résoudre efficacement ce problème**. La plupart de ces méthodes sont présentées dans [66–68].

Après avoir introduit un modèle mathématique qui formalise le problème (section 3.1), nous tentons de le résoudre à l'aide d'un solveur de programmes linéaires (section 3.2). Nous nous intéressons par la suite aux méthodes de résolution approchées. Nous proposons d'abord deux procédures heuristiques constructives basées sur une approche séquentielle (section 3.3), ensuite des adaptations de deux métaheuristiques classiques : la méthode de recherche avec tabous (section 3.4) et la méthode de recherche à voisinage variable (section 3.5), et finalement une méthode que nous nommons méthode de recherche avec tabous à voisinages structurés (section 3.6). Cette dernière associe des techniques de la recherche avec tabous, de la recherche à voisinage variable, et des algorithmes génétiques. La section 3.7 présente une synthèse des expérimentations numériques menées. Une conclusion termine ce chapitre.

Les instances dont nous nous servons pour tester les différentes méthodes sont générées aléatoirement. Toutes les procédures de résolution sont implantées en utilisant le langage de programmation `Java`. Sauf mention contraire, les tests sont réalisés sur un ordinateur doté d'un processeur *AMD Opteron 246 1993 Mhz* avec *2 GB* de mémoire et fonctionnant sous le système d'opération *Linux*.

3.1 Modélisation du problème

Le *problème sur une seule ronde* modélise la construction d'un ensemble d'affectations des juges aux matchs respectant un ensemble de contraintes incontournables et satisfaisant le plus possible à un certain nombre de contraintes souples. Ces contraintes

sont :

Règles incontournables ou contraintes qui doivent être satisfaites :

1. un juge ne peut être affecté à un match impliquant une équipe représentant l'Université dans laquelle il a étudié ;
2. un juge ne peut être affecté à un match impliquant une équipe représentant l'Université à laquelle il est attaché ;
3. 3 ou 5 juges doivent être affectés à chaque match ;
4. au moins l'un de ces juges appartient à l'ensemble des *juges en chef.*

Les deux premières contraintes définissent les affectations permises ou admissibles. On parlera indifféremment de juge admissible à un match et affectation admissible pour dire que ces contraintes sont respectées.

Règles souples ou contraintes (ou objectifs) à satisfaire autant que possible :

1. *contrainte de diversité* : les expertises des juges affectés à un même match doivent être aussi différentes que possible pour couvrir le plus grand nombre de domaines d'expertise ;
2. *contrainte du nombre* : le nombre de matchs avec 5 juges affectés doit être maximisé.

Nous présentons d'abord un modèle mathématique général qui formalise ce problème, puis une linéarisation de ce modèle.

3.1.1 Modèle mathématique général

Nous formulons le problème comme un problème de programmation binaire. Les règles incontournables sont utilisées pour définir les contraintes du problème. La fonction objectif est spécifiée en termes des règles souples. Nous utilisons la notation suivante :

- N : le nombre total de juges disponibles
- M : le nombre de matchs
- K : le nombre de domaines d'expertise

- i : l'indice du juge, $i = 1,\ldots,N$
- j : l'indice du match, $j = 1,\ldots,M$
- $A = [a_{ij}]$ où, $a_{ij} = \begin{cases} 1 & \text{si le juge } i \text{ est admissible au match } j \\ 0 & \text{sinon} \end{cases}$

  Rappelons que le juge i est admissible au match j si i n'a pas étudié et n'est pas attaché aux universités des deux équipes en compétition dans ce match.
- $l_i = \begin{cases} 1 & \text{si le juge } i \text{ est un } \textit{juge en chef} \\ 0 & \text{sinon} \end{cases}$
- $e_{ik} = \begin{cases} 1 & \text{si le juge } i \text{ possède le domaine d'expertise } k \\ 0 & \text{sinon} \end{cases}$
- Pour chaque $i = 1,\ldots,N$ et chaque $j = 1,\ldots,M$, les variables x_{ij} représentent l'affectation du juge i au match j :

  $x_{ij} = \begin{cases} 1 & \text{si le juge } i \text{ est affecté au match } j \\ 0 & \text{sinon} \end{cases}$
- Pour chaque $j = 1,\ldots,M$, les variables y_j^3 et y_j^5 indiquent le nombre de juges affectés au match j :

  $y_j^3 = \begin{cases} 1 & \text{si 3 juges sont affectés au match } j \\ 0 & \text{sinon} \end{cases}$

  $y_j^5 = \begin{cases} 1 & \text{si 5 juges sont affectés au match } j \\ 0 & \text{sinon.} \end{cases}$

Le modèle mathématique que nous proposons est résumé dans la figure 3.1. Dans la fonction objectif (3.1), nous minimisons d'une part, le nombre de fois où plusieurs juges affectés à un match possèdent le même domaine d'expertise, et d'autre part, le nombre de matchs avec 3 juges affectés (ce qui est équivalent à maximiser le nombre de matchs avec 5 juges affectés). Le coefficient $5M$ du second terme de la fonction objectif garantit qu'une paire de juges additionnelle serait affectée à un match même si cela entraînait que, dans chaque match, les 5 juges possèdent le même domaine d'expertise. En effet, dans ce cas, la paire de juges additionnelle augmente le premier terme de 4 dans chaque match. Les contraintes (3.2) indiquent qu'un juge ne peut être affecté à plus d'un match, et les contraintes (3.3) ne permettent pas qu'un juge inadmissible soit affecté à un match.

Au moins un *juge en chef* est affecté à chaque match d'après les contraintes (3.4). Les contraintes (3.5) et (3.6) garantissent que 3 ou 5 juges sont affectés à chaque match.

$$\min \sum_{j=1}^{M} \sum_{k=1}^{K} \max \left\{ (\sum_{i=1}^{N} e_{ik} x_{ij}) - 1, 0 \right\} + 5M \sum_{j=1}^{M} y_j^3 \tag{3.1}$$

(M^1) **Sujet à**

$$\sum_{j=1}^{M} x_{ij} \leq 1 \qquad i = 1, \ldots, N \tag{3.2}$$

$$\sum_{j=1}^{M} (1 - a_{ij}) x_{ij} = 0 \qquad i = 1, \ldots, N \tag{3.3}$$

$$\sum_{i=1}^{N} l_i x_{ij} \geq 1 \qquad j = 1, \ldots, M \tag{3.4}$$

$$\sum_{i=1}^{N} x_{ij} = 3y_j^3 + 5y_j^5 \qquad j = 1, \ldots, M \tag{3.5}$$

$$y_j^3 + y_j^5 = 1 \qquad j = 1, \ldots, M \tag{3.6}$$

$$x_{ij} = 0 \text{ ou } 1 \qquad i = 1, \ldots, N,\ j = 1, \ldots, M \tag{3.7}$$

$$y_j^3, y_j^5 = 0 \text{ ou } 1 \qquad j = 1, \ldots, M. \tag{3.8}$$

Figure 3.1 – Modèle M^1 formalisant le problème sur une seule ronde

3.1.2 Linéarisation du modèle

Nous pouvons transformer le modèle M^1 en un nouveau modèle qui peut être directement résolu par des solveurs de programmes linéaires comme CPLEX. En effet, connaître, si possible, la solution optimale des instances testées nous permettra de juger de la qualité des solutions générées par les méthodes de résolution que nous avons développées.

La linéarisation est triviale. Il suffit de remplacer l'expression $\max \left\{ (\sum_{i=1}^{N} e_{ik} x_{ij}) - 1, 0 \right\}$

par des variables $z_{kj} \geq 0$, et d'ajouter des contraintes linéaires de la forme :

$$z_{kj} \geq \sum_{i=1}^{N} e_{ik} x_{ij} - 1 \qquad j = 1,\ldots,M, \quad k = 1,\ldots,K.$$

Nous obtenons alors le modèle linéaire mixte suivant :

$$\min \sum_{j=1}^{M} \sum_{k=1}^{K} z_{kj} + 5M \sum_{j=1}^{M} y_j^3 \tag{3.9}$$

(ML^1) **Sujet à**

$$\sum_{i=1}^{N} e_{ik} x_{ij} - z_{kj} \leq 1 \qquad j = 1,\ldots,M, k = 1,\ldots,K \tag{3.10}$$

$$z_{kj} \geq 0 \qquad j = 1,\ldots,M, k = 1,\ldots,K \tag{3.11}$$

et l'ensemble des contraintes (3.2) à (3.8).

Figure 3.2 – Modèle linéaire mixte ML^1

3.2 Résolution à l'aide d'un solveur de programmes linéaires

Tel que mentionné dans l'état de l'art, plusieurs auteurs se sont intéressés à des problèmes plus ou moins similaires au nôtre. Malheureusement, au meilleur de notre connaissance, il n'existe pas de librairie de problèmes standard sur lesquels ont été appliquées les méthodes de résolution proposées dans la littérature. Cette lacune est due notamment à la grande diversité des problèmes étudiés. Chaque problème a ses propres spécificités. Dès lors, les méthodes développées sont testées avec des données concernant les problèmes réels considérés. De ce fait, nous avons été amenés à générer des problèmes aléatoirement pour tester nos différentes méthodes de résolution. La méthodologie employée pour générer ces problèmes est résumée dans le paragraphe suivant. Nous présentons ensuite les résultats obtenus en utilisant le solveur de programmes linéaires CPLEX 9.13.

3.2.1 Problèmes tests

Les problèmes générés sont regroupés en fonction de la valeur (nulle ou positive) du premier et du second terme de la fonction objectif (3.1), et du nombre de matchs.

Nous considérons trois ensembles de problèmes : P_1, P_2 et P_3. Les problèmes dans P_1 sont générés de façon à ce qu'il existe une solution où les juges affectés à chaque match ont des domaines d'expertise différents (i.e., le premier terme de la fonction objectif (3.1) est égal à 0). Une telle solution n'existe pas pour les problèmes dans P_2 et P_3. Pour les problèmes dans P_2, il existe une solution où tous les matchs comptent 5 juges affectés (i.e., le second terme de la fonction objectif (3.1) est égal à 0), alors que pour les problèmes dans P_1 et P_3 certains matchs comptent 3 juges affectés seulement. D'autre part, chacun des ensembles P_1, P_2 et P_3 comporte 4 sous-ensembles avec 15, 50, 150 et 500 matchs. Finalement, chacun des 12 sous-ensembles contient 10 instances différentes.

La méthode de génération des problèmes se résume comme suit. Le nombre de *juges en chef* disponibles est généré aléatoirement, mais de sorte à garantir le respect de la contrainte stipulant qu'au moins un *juge en chef* doit être affecté à chaque match. Dans le même ordre d'idées, nous générons le nombre de juges *autres* disponibles de sorte à garantir que le nombre total de juges disponibles permet d'affecter au moins 3 juges à chaque match. Pour chaque juge disponible, ses universités d'attache et d'études correspondent à des nombres aléatoires générés à partir d'une distribution uniforme discrète sur un intervalle spécifié (voir tableau 3.1). L'université que représente chaque équipe de chaque match est choisie aléatoirement dans le même intervalle. Ceci nous permet de disposer d'instances *a priori* difficiles à résoudre. Finalement, le domaine d'expertise de chaque juge est choisi de manière aléatoire selon une loi uniforme dans l'intervalle $[1,6]$. Notons que pour les problèmes dans P_2 et P_3, le nombre de juges qui possèdent le même domaine d'expertise est fixé *a priori* afin de calculer une borne inférieure sur la valeur optimale.

Nous indiquons dans le tableau 3.1 les intervalles utilisés pour effectuer les différents choix aléatoires. Dans ce tableau, M désigne le nombre de matchs tandis que I_l, I_o, et I_u

représentent les intervalles pour choisir respectivement le nombre de *juges en chef* disponibles, le nombre de juges *autres* disponibles, et les indices des universités d'attache et d'études des juges ainsi que celles que représentent les équipes. Notons que le nombre de ces universités est fixe et qu'il dépend du nombre de matchs.

M	I_l	I_o	I_u
15	[15, 50]	[60, 300]	[1, 10]
50	[50, 60]	[100, 220]	[1, 10]
150	[150, 160]	[300, 620]	[1, 30]
500	[500, 510]	[1000, 2020]	[1, 100]

Tableau 3.1 – Caractéristiques des instances de test en fonction du nombre de matchs

3.2.2 Expérimentations numériques

Nous présentons dans ce paragraphe les résultats obtenus en utilisant le solveur de programmes linéaires CPLEX 9.13 sur le modèle linéaire mixte ML^1 présenté au paragraphe 3.1.2. Les tests sont réalisés sur l'ensemble des instances décrites au paragraphe précédent.

Les premières expérimentations numériques ont été effectuées sur une machine avec une mémoire de *2 GB*, mais le solveur ne pouvait pas résoudre la majorité des instances, la mémoire étant saturée. Dès lors, nous avons utilisé un ordinateur avec *4 GB* de mémoire. Il est doté d'un processeur *AMD Opteron 246 1993 Mhz* et fonctionne sous le système d'opération *Linux*. Là encore, nous avons constaté que CPLEX était incapable de trouver la solution optimale de certaines instances même après plusieurs jours de calcul (probablement à cause de la dégénérescence). Nous avons donc limité la durée totale de résolution à 10 heures. Nous avons conservé les valeurs par défaut des autres paramètres de CPLEX.

Les résultats obtenus sont résumés dans le tableau 3.2. Les colonnes désignent dans

l'ordre, l'ensemble de problèmes, le nombre de matchs (*Taille*), le pourcentage des instances résolues à l'optimalité (*%Opt*), et finalement le temps minimal (*Min CPU*), maximal (*Max CPU*) et moyen (*Ave CPU*) requis pour générer les solutions optimales de ces instances. Les temps de résolution sont indiqués en secondes. La figure 3.3 permet de visualiser l'évolution du pourcentage des instances résolues à l'optimalité et l'évolution du temps moyen de résolution en fonction du nombre de matchs.

	Taille	*%Opt*	*Min CPU*	*Max CPU*	*Ave CPU*
P_1	15	100	0,52	98,33	28,74
	50	60	1,44	3,21	2,31
	150	50	17,34	198,01	96,38
	500	50	8 374,73	21 054,26	12 159,96
P_2	15	100	0,53	0,81	0,69
	50	100	1,59	2,51	1,93
	150	100	22,70	43,79	32,41
	500	100	15 565,45	27 719,00	21 299,74
P_3	15	100	0,56	47,84	13,12
	50	60	2,08	34,28	8,85
	150	50	36,63	517,12	276,17
	500	40	18 183,28	27 907,13	23 824,47

Tableau 3.2 – Résultats obtenus en utilisant le solveur CPLEX

À la lecture des résultats du tableau 3.2, il ressort que CPLEX a été en mesure de résoudre toutes les instances de l'ensemble P_2 même celles de grande taille. En revanche, pour les instances des deux autres ensembles, il éprouve des difficultés dès que le nombre de matchs dépasse 15. De plus, le nombre d'instances résolues à l'optimalité décroît en fonction du nombre de matchs. Rappelons que pour les instances dans P_2, il existe une solution où tous les matchs comptent 5 juges affectés (i.e., le second terme de la fonction objectif (3.9) est égal à 0) alors que pour celles dans P_1 et P_3 une telle solution n'existe pas (i.e., le second terme de la fonction objectif (3.9) est positif). Les résultats semblent donc indiquer que la difficulté de résolution est davantage liée à la valeur du second terme de la fonction objectif qu'à la taille des problèmes.

On peut également constater que les temps de résolution évoluent différemment selon les ensembles. En particulier, si on considère les instances dans P_1 et P_3 avec 15 et 50

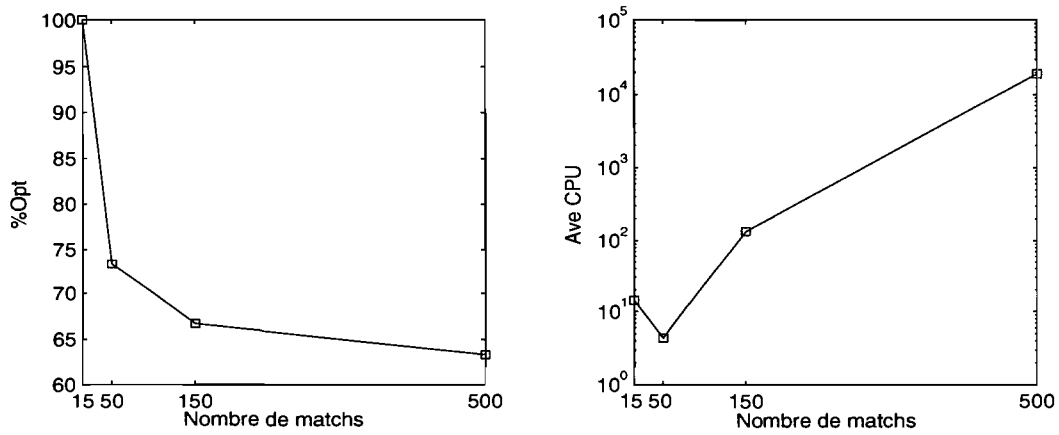

Figure 3.3 – Évolution du pourcentage des instances résolues avec CPLEX et des temps moyens de résolution en fonction de la taille des problèmes

matchs, on remarque que le temps moyen (*AveCPU*) diminue lorsque la taille augmente. Il est difficile d'avancer une explication satisfaisante à cet effet d'autant plus que 40% de ces instances n'ont pas pu être résolues dans une limite de temps de 10 heures. Toutefois, on note que, globalement, les temps de résolution augmentent en fonction du nombre de matchs, mais que l'accroissement n'est pas linéaire. Si on peut obtenir des résultats assez rapidement pour les instances avec 15, 50 et 150 matchs, les temps deviennent prohibitifs pour celles avec 500 matchs.

3.3 Techniques heuristiques séquentielles

Les techniques heuristiques introduites dans cette section reposent sur une approche séquentielle à deux étapes. Les *juges en chef* sont affectés lors de la première étape, et 2 ou 4 juges additionnels sont affectés lors de la seconde. Les *juges en chef* sont affectés en premier car ceux non affectés lors de la première étape sont disponibles pour être affectés en tant que juges *autres* au cours de la seconde.

Les méthodes pour affecter les *juges en chef* sont exposées dans le paragraphe 3.3.1. Nous présentons ensuite, dans le paragraphe 3.3.2, les méthodes pour affecter les juges additionnels. Ces méthodes sont ensuite combinées pour produire différentes techniques

heuristiques séquentielles. Des expérimentations numériques préliminaires nous ont permis d'identifier deux combinaisons différentes plus performantes. Ces deux techniques sont présentées au paragraphe 3.3.3 où nous comparons la qualité de leurs solutions ainsi que leur efficacité.

3.3.1 Affectation des *juges en chef*

Nous introduisons deux méthodes distinctes pour affecter un *juge en chef* à chaque match : une méthode exacte (**LAP**) et une procédure heuristique constructive (**HLA**).

Métode exacte (LAP). L'affectation est obtenue en résolvant un *problème d'affectation linéaire*. Pour formuler le problème, nous associons un nœud source à chaque *juge en chef* disponible et un nœud destination à chaque match.

Notons par N^l le nombre de *juges en chef* disponibles. Si N^l est supérieur à M, alors nous ajoutons $(N^l - M)$ nœuds destination fictifs. Les coûts linéaires c_{ij} pour affecter le *juge en chef* i au match j sont définis comme suit :

$$c_{ij} = \begin{cases} 0 & \text{si le juge } i \text{ est admissible au match } j \\ 1 & \text{sinon.} \end{cases}$$

Considérons une solution optimale y^* du *problème d'affectation linéaire* suivant :

$$\min \sum_{j=1}^{N^l} \sum_{i=1}^{N^l} c_{ij}\, y_{ij}$$

$$\textbf{Sujet à} \quad \sum_{j=1}^{N^l} y_{ij} = 1 \qquad i = 1,\ldots,N^l$$

$$\sum_{i=1}^{N^l} y_{ij} = 1 \qquad j = 1,\ldots,N^l$$

$$y_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,N^l,\ j = 1,\ldots,N^l.$$

Comme par hypothèse il existe une affectation où un *juge en chef* peut être affecté à chaque match, il existe une solution de ce problème de valeur nulle. Par conséquent, si

$y^*_{ij} = 1$, alors nous affectons le *juge en chef* i au match j.

Heuristique constructive (HLA). Nous proposons une procédure heuristique séquentielle où, à chaque itération, un *juge en chef* est affecté à un match. Elle intègre une composante gloutonne afin de biaiser la sélection pour faciliter les affectations futures. Analysons une itération typique de cette procédure.

Considérons l'ensemble des *juges en chef* possédant le domaine d'expertise le moins rare k' (i.e., parmi tous les domaines d'expertise, k' est celui que possède le plus grand nombre de juges non encore affectés). Parmi les juges de cet ensemble, nous sélectionnons le juge i' admissible pour le moins de matchs parmi les matchs auxquels aucun *juge en chef* n'a encore été affecté. Si aucun juge n'a pu être identifié dans cet ensemble, alors nous considérons successivement les domaines d'expertise les moins rares suivants jusqu'à ce qu'un *juge en chef* soit sélectionné.

Considérons maintenant l'ensemble des matchs pour lesquels le juge i' est admissible, et qui comptent le plus petit nombre de *juges en chef* admissibles encore disponibles. Parmi les matchs de cet ensemble, nous sélectionnons le match j' auquel le plus petit nombre de juges *autres* possédant le domaine d'expertise k' sont admissibles. Le juge i' est alors affecté au match j' (i.e., $x_{i'j'} = 1$).

Notons que pour chacune des sélections précédentes, en cas d'égalité, les choix se font aléatoirement. Les critères de sélection que nous utilisons reposent sur une vision globale du processus de résolution pour affecter un *juge en chef* admissible à chaque match en prenant des mesures supplémentaires pour faciliter l'affectation des juges additionnels. La procédure **HLA** est résumée dans la figure 3.4.

3.3.2 Affectation des juges additionnels

Nous cherchons maintenant à affecter 2 ou 4 juges additionnels à chaque match dépendamment du nombre total de juges disponibles N^d. Nous proposons deux méthodes différentes : une méthode hybride (**HYM**) et une procédure heuristique constructive (**HOA**). Les deux méthodes comportent deux phases. Une première paire de juges est affectée à chaque match lors de la première phase, et une seconde paire est affectée au

Initialisation

$L = \{i : l_i = 1\}$, l'ensemble des *juges en chef* disponibles

$O = \{i : l_i = 0\}$, l'ensemble des juges *autres* disponibles

$E^k = \{i \in L \cup O : e_{ik} = 1\}$, l'ensemble des juges disponibles possédant l'expertise k

$\mathtt{M} = \{1, \ldots, M\}$, l'ensemble des matchs auxquels aucun *juge en chef* n'a encore été affecté

Algorithme

Tant que $(\mathtt{M} \neq \emptyset)$ **faire**

SÉLECTION D'UN JUGE

$i' := \infty$

$\mathtt{K} := \{1, \ldots, K\}$

Tant que $(i' = \infty)$ **faire**

$k' := \arg\max_{k \in \mathtt{K}} |E^k|$

$L^{k'} := \{i \in L \cap E^{k'} : \sum_{j \in \mathtt{M}} a_{ij} \geq 1\}$

Si $(L^{k'} \neq \emptyset)$ **Alors**

$i' := \arg\min_{i \in L^{k'}} \sum_{j \in \mathtt{M}} a_{ij}$

Sinon

$\mathtt{K} := \mathtt{K} - \{k'\}$

Fin Si

Fait

SÉLECTION D'UN MATCH

$\overline{J} := \{\overline{j} \in \mathtt{M} : a_{i'\overline{j}} = 1 \text{ et } \overline{j} = \arg\min_{j \in \mathtt{M}} \sum_{i \in L} a_{ij}\}$

$j' := \arg\min_{\overline{j} \in \overline{J}} \sum_{i \in O \cap E^{k'}} a_{i\overline{j}}$

MISE À JOUR

$x_{i'j'} := 1$

$E^{k'} := E^{k'} - \{i'\}$

$L := L - \{i'\}$

$\mathtt{M} := \mathtt{M} - \{j'\}$

Fait

Figure 3.4 – Procédure HLA

maximum de matchs lors de la seconde. Notons que l'affectation des juges par paires permet de satisfaire la contrainte incontournable stipulant que 3 ou 5 juges doivent être affectés à chaque match.

Méthode hybride (HYM). Lors de la première phase, nous utilisons une procédure permettant d'affecter séquentiellement le premier et le second juge additionnels à tous les matchs. Cette procédure comprend deux étapes. D'abord, l'ensemble des juges les plus intéressants à affecter est déterminé en résolvant un *bottleneck assignment problem*. Par la suite, les affectations sont obtenues en résolvant un *problème d'affectation linéaire*. Une approche similaire a été utilisée par PROOL [91] dans un contexte académique où l'objectif consiste à affecter un projet à chaque étudiant de manière à satisfaire le plus possible ses préférences.

Pour formuler les deux problèmes d'affectation, nous associons un nœud source à chaque juge disponible et un nœud destination à chaque match. Puisque le nombre de juges disponibles N^a (N^a est égal respectivement à N^d et à $(N^d - M)$ lorsque nous affectons le premier et le second juge additionnels) est supérieur ou égal à M, nous ajoutons $(N^a - M)$ nœuds destination fictifs. Les coûts linéaires w_{ij} pour le *bottleneck assignment problem* sont définis comme suit :

$$w_{ij} = \begin{cases} C & \text{si le juge } i \text{ n'est pas admissible au match } j \\ \alpha_{ij} & \text{sinon} \end{cases}$$

où C est une constante assez grande et α_{ij} représente le nombre de juges déjà affectés au match j qui possèdent le même domaine d'expertise que le juge i.

Considérons une solution optimale y^* du *bottleneck assignment problem* suivant :

$$\min \max_{1 \le j \le N^a} \sum_{i=1}^{N^a} w_{ij} y_{ij}$$

$$\begin{aligned}
\textbf{Sujet à} \quad & \sum_{j=1}^{N^a} y_{ij} = 1 && i = 1,\ldots,N^a \\
& \sum_{i=1}^{N^a} y_{ij} = 1 && j = 1,\ldots,N^a \\
& y_{ij} = 0 \text{ ou } 1 && i = 1,\ldots,N^a,\ j = 1,\ldots,N^a.
\end{aligned}$$

Compte tenu de la définition des coûts linéaires w_{ij}, dans la solution optimale y^*, nous affectons à chaque match un juge admissible (si possible), possédant un domaine d'expertise différent de celui du plus grand nombre possible de juges déjà affectés à ce match. La valeur optimale z^* correspond au coût minimal des matchs présentant la pire affectation.

Pour améliorer l'ensemble des affectations, nous résolvons ensuite un *problème d'affectation linéaire* en tenant compte de la valeur optimale z^* obtenue à l'issue de la première étape. Nous définissons les coûts linéaires $\overline{w}_{ij}$ comme suit :

$$\overline{w}_{ij} = \begin{cases} w_{ij} & \text{si } w_{ij} \le z^* \\ C & \text{sinon.} \end{cases}$$

Considérons le *problème d'affectation linéaire* suivant :

$$\min \sum_{j=1}^{N^a} \sum_{i=1}^{N^a} \overline{w}_{ij} y_{ij}$$

$$\begin{aligned}
\textbf{Sujet à} \quad & \sum_{j=1}^{N^a} y_{ij} = 1 && i = 1,\ldots,N^a \\
& \sum_{i=1}^{N^a} y_{ij} = 1 && j = 1,\ldots,N^a \\
& y_{ij} = 0 \text{ ou } 1 && i = 1,\ldots,N^a,\ j = 1,\ldots,N^a.
\end{aligned}$$

La fonction objectif minimise le coût total de l'affectation. Cependant, la manière selon laquelle les coûts linéaires $\overline{w}_{ij}$ sont définis permet d'assurer que, pour chaque match, le coût d'affectation ne dépasse pas la valeur optimale z^*.
Considérons maintenant une solution optimale y^* de ce dernier problème. Si $y^*_{ij} = 1$, alors le juge i est affecté au match j (i.e., $x_{ij} = 1$).

Après avoir affecté un premier juge additionnel à chaque match, la même procédure est reprise pour tenter d'en affecter un deuxième.

Lors de la seconde phase, le nombre de juges disponibles est réduit à $N^a = N^d - 2M$. Si $N^a \geq 2M$, alors nous pouvons affecter une paire de juges additionnelle à chaque match. Sinon, nous affectons une seconde paire de juges au maximum de matchs. Dans les deux cas, un *problème d'affectation quadratique* est formulé puis résolu.

Supposons que le nombre de juges disponibles N^a est suffisant pour affecter une paire de juges additionnelle à chaque match. Pour formuler le *problème d'affectation quadratique*, nous associons deux nœuds destination j et $(M + j)$ à chaque match j. Si $N^a > 2M$, alors $N^a - 2M$ nœuds destination fictifs sont ajoutés. Chaque nœud source est associé à un juge i. Nous définissons les coûts quadratiques de façon à pénaliser la violation des contraintes. Aussi, les coûts quadratiques $c_{irj(M+j)} = c_{rij(M+j)}$, pour affecter la paire de juges (i, r) au match j, sont-ils spécifiés comme suit :

$$c_{irj(M+j)} = \begin{cases} 2C & \text{si les deux juges } i \text{ et } r \text{ ne sont pas admissibles au match } j \\ C & \text{si le juge } i \text{ ou le juge } r \text{ n'est pas admissible au match } j \\ \alpha_{ij} + 1 & \text{si les deux juges } i \text{ et } r \text{ sont admissibles au match } j \text{ et possèdent} \\ & \text{le même domaine d'expertise} \\ \alpha_{ij} + \alpha_{rj} & \text{sinon} \end{cases}$$

où, C est une constante assez grande et α_{ij} représente le nombre de juges déjà affectés au match j possédant le même domaine d'expertise que le juge i.

Considérons une solution optimale y^* du *problème d'affectation quadratique* suivant :

$$\min \sum_{j=1}^{M} \sum_{i=1}^{N^a-1} \sum_{r=i+1}^{N^a} c_{irj(M+j)} \left(y_{ij} y_{r(M+j)} + y_{rj} y_{i(M+j)}\right)$$

$$\textbf{Sujet à} \quad \sum_{j=1}^{N^a} y_{ij} = 1 \qquad i = 1,\ldots,N^a$$

$$\sum_{i=1}^{N^a} y_{ij} = 1 \qquad j = 1,\ldots,N^a$$

$$y_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,N^a,\ j = 1,\ldots,N^a.$$

Pour chaque match j, la paire de juges (i,r) telle que $y^*_{ij} = y^*_{r(M+j)} = 1$ lui est affectée.

Supposons maintenant que le nombre de juges disponibles $N^a < 2M$. Nous affectons alors une seconde paire de juges à $\lfloor \frac{N^a}{2} \rfloor$ matchs en résolvant un *problème d'affectation quadratique* où, $\kappa = N^a \bmod 2$ nœud destination est associé à un match fictif, et $(2M - N^a + \kappa)$ nœuds source sont associés à des juges fictifs. Les coûts quadratiques sont définis comme dans le premier cas ($N^a \geq 2M$) de façon à pénaliser la violation des contraintes. De plus, pour chaque match j, nous devons éviter de coupler un juge i disponible avec un juge fictif r car ceci serait équivalent à n'affecter qu'un seul juge additionnel à j. Les coûts quadratiques sont spécifiés comme suit :

$$c_{irj(M+j)} = \begin{cases} 3MC & \text{si l'un des deux juges } i \text{ ou } r \text{ est fictif} \\ 2C & \text{si les deux juges } i \text{ et } r \text{ ne sont pas admissibles au match } j \\ C & \text{si le juge } i \text{ ou le juge } r \text{ n'est pas admissible au match } j \\ \alpha_{ij} + 1 & \text{si les deux juges } i \text{ et } r \text{ sont admissibles au match } j \text{ et possèdent} \\ & \text{le même domaine d'expertise} \\ \alpha_{ij} + \alpha_{rj} & \text{sinon.} \end{cases}$$

Notons que si i est un juge fictif, alors il est admissible au match j et $\alpha_{ij} = 0$.
Étant donnée une solution optimale y^* du *problème d'affectation quadratique* correspondant, pour chaque match j, si $y^*_{ij} = y^*_{r(M+j)} = 1$, et si les deux juges i et r ne sont pas fictifs, alors cette paire de juges (i,r) lui est affectée.

Notons qu'au cours des deux phases de la méthode **HYM**, certains juges non admissibles peuvent être affectés à certains matchs. Dans ce cas, la solution générée n'est pas réalisable.

Heuristique constructive (HOA). Cette méthode s'inspire de la méthode **HLA** introduite au paragraphe 3.3.1 pour affecter les *juges en chef*. À chaque phase, une procédure heuristique séquentielle est utilisée pour affecter les paires de juges aux matchs. Pour introduire cette procédure, nous avons besoin de la notion additionnelle d'*éligibilité*. Un juge i est dit *éligible* à un match s'il y est admissible et possède un domaine d'expertise (que nous notons par d_i) différent de celui de tous les autres juges affectés à ce match. Analysons une itération typique de la première phase.

Considérons l'ensemble des juges possédant le domaine d'expertise le moins rare k' (i.e., parmi tous les domaines d'expertise, k' est celui que possède le plus grand nombre de juges non encore affectés), et tel qu'il existe au moins un match pour lequel ce juge et un autre juge disponible possédant un domaine d'expertise différent de k' sont *éligibles*. Nous dirons qu'un tel match est *acceptable* pour le premier juge. Parmi les juges de cet ensemble, nous sélectionnons le juge i' *éligible* pour le moins de matchs auxquels aucune paire de juges n'a encore été affectée.

Si aucun juge n'a pu être identifié dans cet ensemble, alors nous considérons successivement les domaines d'expertise les moins rares suivants jusqu'à ce qu'un juge soit sélectionné ou jusqu'à ce que tous les domaines d'expertise soient considérés. Dans ce dernier cas, nous répétons le processus en relaxant successivement un à la fois les différents critères de sélection dans l'ordre suivant. D'abord, nous éliminons la condition que le second juge doit posséder un domaine d'expertise différent de celui du premier, ensuite nous relaxons la condition d'*éligibilité* en une condition d'admissibilité pour le second juge, et finalement pour le premier juge.

Considérons maintenant l'ensemble des matchs *acceptables* pour le juge sélectionné i', et qui comptent le plus petit nombre de juges *éligibles* ayant un domaine d'expertise différent de $d_{i'}$. Parmi les matchs de cet ensemble, nous sélectionnons le match j' auquel le plus petit nombre de juges disponibles possédant le domaine d'expertise $d_{i'}$ sont

éligibles.

Finalement, pour sélectionner le second juge r', nous considérons l'ensemble des juges possédant le domaine d'expertise le moins rare différent de $d_{i'}$, et qui sont *éligibles* pour le match j'. Nous sélectionnons le juge r' *éligible* pour le plus petit nombre de matchs. Les juges i' et r' sont alors affectés au match j' (i.e., $x_{i'j'} = x_{r'j'} = 1$).

Notons que les critères de sélection du match et du second juge sont ajustés en fonction des conditions qui doivent être relaxées pour sélectionner le premier juge. De plus, en cas d'égalité, les choix se font aléatoirement.

Lors de la première phase, ce processus est répété jusqu'à ce qu'une paire de juges additionnelle soit affectée à chaque match ou jusqu'à ce qu'aucune paire de juges ne puisse être affectée aux matchs restants. Au cours de la seconde phase, nous utilisons le même processus pour affecter une paire de juges additionnelle au maximum de matchs. Les règles de sélection du match sont légèrement modifiées. Nous le sélectionnons parmi les matchs *acceptables* pour le juge i', et qui comptent le plus petit nombre de juges disponibles *éligibles* ayant un domaine d'expertise différent de $d_{i'}$. En effet, le nombre de juges disponibles *éligibles* possédant le domaine d'expertise $d_{i'}$ n'a plus aucune importance car aucune autre paire de juges ne sera affectée au match.

Finalement, notons qu'à l'instar de la méthode **HLA**, la méthode **HOA** repose sur une stratégie à long terme. Les différentes sélections sont faites de façon à fournir une vision globale au processus de résolution dans le but de faciliter les affectations ultérieures.

La procédure **HOA** pour la première phase est résumée dans la figure 3.5. Il convient de souligner que cette heuristique peut s'arrêter avant de générer une solution réalisable.

Initialisation

$I = \{1,\ldots,N^d\}$, l'ensemble des juges disponibles

$E^k = \{i \in I : e_{ik} = 1\}$, l'ensemble des juges disponibles possédant l'expertise k

$\varepsilon_{ij} = \begin{cases} 1 & \text{si le juge } i \text{ est } \textit{éligible} \text{ pour le match } j \\ 0 & \text{sinon} \end{cases}$

$\mathtt{M} = \{1,\ldots,M\}$, l'ensemble des matchs auxquels aucune paire de juges n'a encore été affectée

fin-algo := **faux**

Algorithme

Tant que (*fin-algo* = **faux ET** $\mathtt{M} \neq \emptyset$) **faire**

SÉLECTION D'UN PREMIER JUGE

ÉTAPE 1-a

$i' := \infty$, $\mathtt{K} := \{1,\ldots,K\}$, *fin-étape* := **faux**

Tant que (*fin-étape* = **faux**) **faire**

$k' := \arg\max_{k \in \mathtt{K}} |E^k|$

$I^{k'} := \{i \in E^{k'} : \exists j \in \mathtt{M}, \exists r \in I - E^{k'} \text{ tel que } \varepsilon_{ij} = \varepsilon_{rj} = 1\}$

Si ($I^{k'} \neq \emptyset$) **Alors**

$i' := \arg\min_{i \in I^{k'}} \sum_{j \in \mathtt{M}} \varepsilon_{ij}$

fin-étape := **vrai**

Aller à ÉTAPE 2-a

Sinon

$\mathtt{K} := \mathtt{K} - \{k'\}$

Fin Si

Si ($\mathtt{K} = \emptyset$ **ET** $i' = \infty$) **Alors**

fin-étape := **vrai**

Aller à ÉTAPE 1-b

Fin Si

Fait

ÉTAPE 1-b

$i' := \infty$, $\mathtt{K} := \{1,\ldots,K\}$, *fin-étape* := **faux**

Tant que (*fin-étape* = **faux**) **faire**

$k' := \arg\max_{k \in \mathtt{K}} |E^k|$

$I^{k'} := \{i \in E^{k'} : \exists j \in \mathtt{M}, \exists r \in E^{k'} \text{ tel que } \varepsilon_{ij} = \varepsilon_{rj} = 1\}$

Si ($I^{k'} \neq \emptyset$) **Alors**

$i' := \arg\min_{i \in I^{k'}} \sum_{j \in \mathtt{M}} \varepsilon_{ij}$

fin-étape := **vrai**

Aller à ÉTAPE 2-b

Sinon

$\mathtt{K} := \mathtt{K} - \{k'\}$

Fin Si

Si ($\mathtt{K} = \emptyset$ **ET** $i' = \infty$) **Alors**

fin-étape := **vrai**

Aller à ÉTAPE 1-c

Fin Si

Fait

ÉTAPE 1-c

$i' := \infty$, $\mathtt{K} := \{1, \ldots, K\}$, *fin-étape* := **faux**

Tant que (*fin-étape* = **faux**) **faire**

  $k' := \arg\max_{k \in \mathtt{K}} |E^k|$

  $I^{k'} := \{i \in E^{k'} : \exists j \in \mathtt{M}, \exists r \in I \text{ tel que } \varepsilon_{ij} = a_{rj} = 1\}$

  Si ($I^{k'} \neq \emptyset$) **Alors**

    $i' := \arg\min_{i \in I^{k'}} \sum_{j \in \mathtt{M}} \varepsilon_{ij}$

    fin-étape := **vrai**

    Aller à ÉTAPE 2-c

  Sinon

    $\mathtt{K} := \mathtt{K} - \{k'\}$

  Fin Si

  Si ($\mathtt{K} = \emptyset$ **ET** $i' = \infty$) **Alors**

    fin-étape := **vrai**

    Aller à ÉTAPE 1-d

  Fin Si

Fait

ÉTAPE 1-d

$i' := \infty$

$\bar{I} := \{i \in I : \exists j \in \mathtt{M}, \exists r \in I \text{ tel que } a_{ij} = a_{rj} = 1\}$

Si ($\bar{I} \neq \emptyset$) **Alors**

  $i' := \arg\min_{i \in \bar{I}} \sum_{j \in \mathtt{M}} a_{ij}$

  Aller à ÉTAPE 2-d

Sinon

  fin-algo := **vrai**

Fin Si

SÉLECTION D'UN MATCH

ÉTAPE 2-a

$\bar{J} := \{\bar{j} \in \mathtt{M} : \exists r \in I - E^{d_{i'}} \text{ tel que } \varepsilon_{i'\bar{j}} = \varepsilon_{r\bar{j}} = 1 \text{ et } \bar{j} = \arg\min_{j \in \mathtt{M}} \sum_{i \in I - E^{d_{i'}}} \varepsilon_{ij}\}$

$j' := \arg\min_{\bar{j} \in \bar{J}} \sum_{i \in E^{d_{i'}}} \varepsilon_{i\bar{j}}$

Aller à ÉTAPE 3-a

ÉTAPE 2-b

$\bar{J} := \{\bar{j} \in \mathtt{M} : \exists r \in E^{d_{i'}} \text{ tel que } \varepsilon_{i'\bar{j}} = \varepsilon_{r\bar{j}} = 1\}$

$j' := \arg\min_{\bar{j} \in \bar{J}} \sum_{i \in I} \varepsilon_{i\bar{j}}$

Aller à ÉTAPE 3-b

ÉTAPE 2-c

$\overline{J} := \{\overline{j} \in \mathtt{M} : \exists r \in I \quad \text{tel que} \quad \varepsilon_{i'\overline{j}} = a_{r\overline{j}} = 1\}$

$j' := \arg\min_{\overline{j}\in\overline{J}} \sum_{i\in I} a_{i\overline{j}}$

Aller à ÉTAPE 3-c

ÉTAPE 2-d

$\overline{J} := \{\overline{j} \in \mathtt{M} : \exists r \in I \quad \text{tel que} \quad a_{i'\overline{j}} = a_{r\overline{j}} = 1\}$

$j' := \arg\min_{\overline{j}\in\overline{J}} \sum_{i\in I} a_{i\overline{j}}$

Aller à ÉTAPE 3-d

SÉLECTION D'UN SECOND JUGE

ÉTAPE 3-a

$r' := \infty, \quad \mathtt{K} := \{1, \ldots, K\} - \{d_{i'}\}$

Tant que $(r' = \infty)$ **faire**

- $k' := \arg\max_{k\in\mathtt{K}} |E^k|$
- $R^{k'} := \{r \in E^{k'} : \varepsilon_{rj'} = 1\}$
- **Si** $(R^{k'} \neq \emptyset)$ **Alors**
  - $r' := \arg\min_{r\in R^{k'}} \sum_{j\in\mathtt{M}} \varepsilon_{rj}$
  - **Aller à** MISE À JOUR
- **Sinon**
  - $\mathtt{K} := \mathtt{K} - \{k'\}$
- **Fin Si**

Fait

ÉTAPE 3-b

$R := \{r \in E^{d_{i'}} : r \neq i' \quad \text{tel que} \quad \varepsilon_{rj'} = 1\}$

$r' := \arg\min_{r\in R} \sum_{j\in\mathtt{M}} \varepsilon_{rj}$

Aller à MISE À JOUR

ÉTAPE 3-c

$r' := \infty, \quad \mathtt{K} := \{1, \ldots, K\}$

Tant que $(r' = \infty)$ **faire**

- $k' := \arg\max_{k\in\mathtt{K}} |E^k|$
- $R^{k'} := \{r \in E^{k'} : r \neq i' \quad \text{tel que} \quad a_{rj'} = 1\}$
- **Si** $(R^{k'} \neq \emptyset)$ **Alors**
  - $r' := \arg\min_{r\in R^{k'}} \sum_{j\in\mathtt{M}} \varepsilon_{rj}$
  - **Aller à** MISE À JOUR
- **Sinon**
  - $\mathtt{K} := \mathtt{K} - \{k'\}$
- **Fin Si**

Fait

ÉTAPE 3-d
$R := \{r \in I : r \neq i' \quad \text{tel que} \quad a_{rj'} = 1\}$
$r' := \arg\min_{r \in R} \sum_{j \in \mathtt{M}} a_{rj}$
Aller à MISE À JOUR
MISE À JOUR
$x_{i'j'} := 1, x_{r'j'} := 1$
$E^{d_{i'}} := E^{d_{i'}} - \{i'\}, E^{d_{r'}} := E^{d_{r'}} - \{r'\}$
$I := I - \{i', r'\}, \mathtt{M} := \mathtt{M} - \{j'\}$
Fait

Figure 3.5 – Procédure HOA pour la phase 1

3.3.3 Expérimentations numériques

Des expérimentations numériques préliminaires nous ont permis de retenir deux techniques différentes en combinant les méthodes pour affecter les *juges en chef* et celles pour affecter les juges additionnels. La première **LAP-HYM** est obtenue en combinant les méthodes utilisant des modèles mathématiques alors que la deuxième **HLA-HOA** repose sur des heuristiques. Nous utilisons les instances de test décrites au paragraphe 3.2.1 pour les évaluer et comparer leur efficacité.

La technique **LAP-HYM** nécessite la résolution de trois problèmes largement étudiés dans la littérature : le *problème d'affectation linéaire*, le *bottleneck assignment problem* et le *problème d'affectation quadratique*. Nous utilisons les méthodes exactes proposées par JONKER et VOLGENANT [59] pour résoudre le *problème d'affectation linéaire* et celui du *bottleneck assignment* (les codes sont disponibles à l'adresse : `http://www.magiclogic.com/assignment.html`), et la méthode de recherche avec tabous proposée par TAILLARD [108] pour résoudre le *problème d'affectation quadratique* (le code est disponible à l'adresse : `http://www.opt.math.tugraz.ac.at/qaplib/codes.html`).

Les paramètres de la recherche avec tabous sont fixés comme dans [108], et le nombre maximal d'itérations est limité à 10 000. De plus, afin de pénaliser considérablement la violation des contraintes d'admissibilité, le paramètre C est fixé à $50M^2$ (M étant le nombre de matchs). Finalement, notons que les deux techniques de résolution opèrent des choix aléatoires à certaines étapes. Nous avons donc résolu chaque instance 10 fois avec une valeur d'initialisation différente pour le générateur des nombres aléatoires.

Nous comparons les deux techniques **LAP-HYM** et **HLA-HOA** selon les cinq critères suivants :

1. *Ave dev* représente l'écart moyen entre les valeurs des solutions générées et la valeur optimale (lorsque nous disposons de cette information) ou la borne inférieure sur cette valeur ;
2. *%Feas* représente le pourcentage des solutions réalisables trouvées ;
3. *%Opt* correspond au pourcentage des solutions optimales trouvées ;
4. σ_{dev} représente l'écart type de la déviation. Cette mesure nous permet d'avoir une indication sur la stabilité de chaque technique ;
5. *Ave CPU* mesure le temps moyen de résolution. Il est indiqué en secondes.

Le tableau 3.3 récapitule les résultats obtenus pour chaque sous-ensemble de problèmes. Notons que toutes les solutions générées par les deux techniques sont toujours réalisables. Par conséquent, le critère *%Feas* ne figure pas dans ce tableau. La figure 3.6 représente l'évolution de l'écart moyen (*Ave dev*) et du temps moyen de résolution (*Ave CPU*) en fonction du nombre de matchs.

En premier lieu, si l'on effectue une comparaison globale entre les deux techniques (cf. tableau 3.3), il ressort que **HLA-HOA** se démarque clairement au niveau du temps de résolution (réduit en moyenne d'un facteur de 135). En ce qui concerne le pourcentage des solutions optimales obtenues, globalement, il est meilleur de 10% en utilisant **HLA-HOA**, mais pour les plus gros problèmes de type P_1 et P_2, le pourcentage est meilleur en utilisant **LAP-HYM**. Notons que **LAP-HYM** est également coûteuse en termes de mémoire. En effet, les tests pour les problèmes avec 500 matchs ont été réalisés sur une machine munie de *16GB* de mémoire. Ceci est indiqué par le symbole (†) dans

	Taille	LAP-HYM				HLA-HOA			
		Ave dev	*%Opt*	σ_{dev}	*Ave CPU*	*Ave dev*	*%Opt*	σ_{dev}	*Ave CPU*
P_1	15	0,27	75	0,49	0,31	0,16	85	0,39	0,03
	50	1,69	37	1,85	4,16	7,83	71	42,81	0,11
	150	0,23	86	0,72	16,71	0,83	73	1,71	0,38
	500	0,00	100	0,00	†102,37	3,12	43	4,60	9,56
P_2	15	0,37	64	0,51	1,23	0,03	97	0,17	0,06
	50	3,07	18	2,18	8,31	47,94	18	138,68	0,14
	150	8,15	7	7,17	110,05	29,18	0	104,72	0,44
	500	18,01	3	8,65	†1 992,08	66,40	0	21,85	11,12
P_3	15	0,27	83	0,63	0,29	0,06	94	0,24	0,02
	50	7,43	6	5,72	5,72	25,69	47	83,16	0,13
	150	28,85	0	13,25	105,46	3,17	45	6,30	0,40
	500	79,49	0	45,26	†1 993,76	16,44	32	23,58	9,70

Tableau 3.3 – Comparaison de l'efficacité de LAP-HYM et HLA-HOA en fonction des sous-ensembles de problèmes

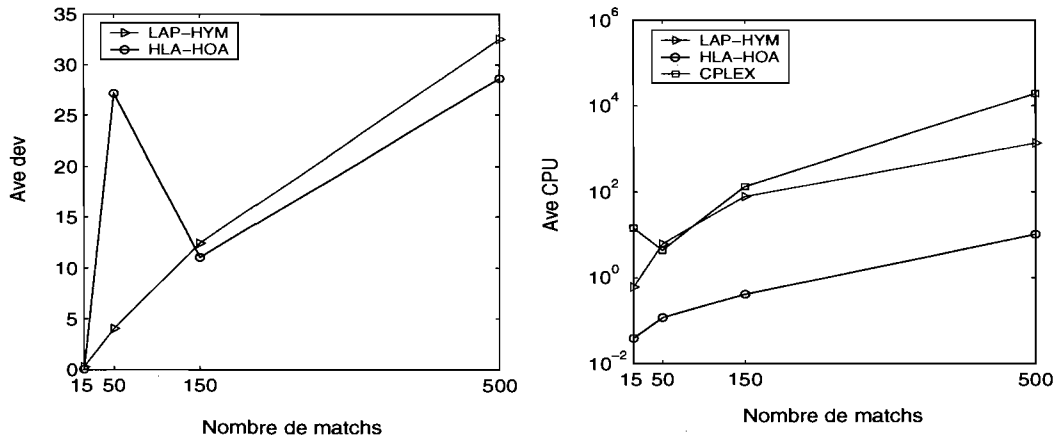

Figure 3.6 – Comparaison de l'efficacité de LAP-HYM et HLA-HOA en fonction de la taille des problèmes

la colonne *AveCPU*. Si l'on s'intéresse à l'écart moyen, à première vue, les résultats sont légèrement en faveur de **LAP-HYM** (*Avedev* est réduit en moyenne d'un facteur de 1,36). De plus, cette technique présente en général une meilleure stabilité (meilleure valeur de σ_{dev}).

Analysons maintenant les résultats relativement à la taille des problèmes. Si l'on s'intéresse à l'écart moyen (cf. premier graphique de la figure 3.6), on note que les résultats sont partagés : si **HLA-HOA** semble dominer pour les problèmes avec 15, 150 et 500 matchs, ceci n'est pas le cas pour ceux avec 50 matchs. De plus, l'écart entre les deux techniques se creuse davantage pour cette catégorie de problèmes.
Afin d'affiner l'analyse, nous avons regroupé dans la figure 3.7 l'écart obtenu par chaque technique sur la totalité des 1 200 résolutions.

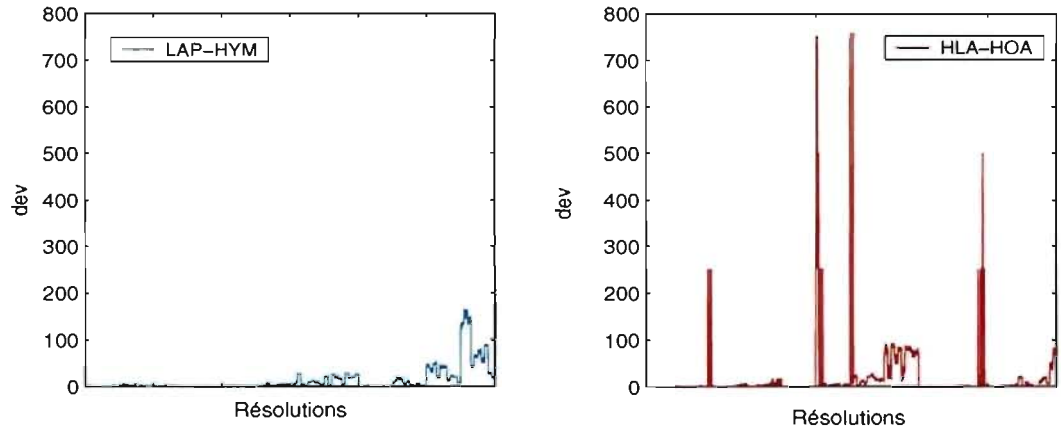

Figure 3.7 – Écarts pour l'ensemble des solutions produites par LAP-HYM et HLA-HOA

On constate que **LAP-HYM** produit des solutions avec une plus grande homogénéité. Les écarts entre les solutions produites et les solutions optimales tendent à se regrouper autour des valeurs inférieures à $5M$, M étant le nombre de matchs. Dans le cas de **HLA-HOA**, la dispersion est plus importante. De plus, pour certaines instances, la valeur de l'écart est supérieure à $5M$. Cela signifie que 3 juges sont affectés à certains matchs

quoiqu'il reste des paires de juges non affectées. Cette faiblesse est sans doute due à la stratégie myope de la technique **HLA-HOA** visant à choisir les juges en premier lieu sur la base de leur domaine d'expertise. Par conséquent, lors des dernières itérations, les paires de juges disponibles (juges non affectés) ne sont admissibles à aucun match restant.

Pour approfondir la comparaison, les résultats de l'ensemble des résolutions ont fait l'objet d'une analyse statistique basée sur le test des rangs pour échantillons appariés de WILCOXON [92]. Ce test nous renseigne sur la différence qui peut exister entre deux échantillons avec des moyennes différentes. Étant données une hypothèse nulle H0 affirmant l'absence de différence et une hypothèse alternative H1, on choisit un seuil de confiance α qui représente le risque d'erreur que l'on accepte de commettre lors de la conclusion. Le test renvoie une *p_valeur* qui représente le seuil pour lequel on rejetterait l'hypothèse nulle H0. Ainsi, si $p\_valeur > \alpha$, on décide que H0 est vraie (i.e., il n'y a pas de différence), sinon, H0 est rejetée et l'hypothèse alternative H1 est acceptée.
Nous avons testé l'hypothèse nulle selon laquelle les performances des deux techniques **LAP-HYM** et **HLA-HOA** (en termes de la qualité des solutions) sont identiques versus l'hypothèse alternative qu'il existe une différence significative entre les deux. En utilisant un seuil de confiance de 0,05, l'hypothèse nulle a été acceptée ($p\_valeur = \mathbf{0,95}$). Nous concluons donc, qu'au niveau de la qualité des solutions, il n'y a pas de différence statistiquement significative entre **LAP-HYM** et **HLA-HOA**.

Considérons maintenant les temps moyens de résolution (cf. deuxième graphique de la figure 3.6). Pour des fins de comparaison, nous avons aussi représenté sur ce graphique les résultats obtenus avec CPLEX. Notons que les résultats de CPLEX concernent uniquement les instances qui ont pu être résolues à l'optimalité avec le solveur dans une limite de temps de 10 heures.
La différence entre les deux techniques est cette fois-ci nette : **HLA-HOA** domine indéniablement **LAP-HYM**. On constate que les temps augmentent en fonction du nombre des matchs, ce qui est logique. Cependant, les taux de croissance sont plus accentués pour **LAP-HYM**. Ceci est sans doute la conséquence du fait que les deux techniques ont des critères d'arrêt différents.

Si l'on effectue une comparaison entre CPLEX et les deux techniques, celles-ci se révèlent meilleures, sauf pour les problèmes avec 50 matchs pour lesquels le temps moyen requis par CPLEX est légèrement inférieur à celui de **LAP-HYM** (4,36 secondes vs 6,06 secondes). Notons toutefois que 26,67% de ces problèmes n'ont pas été résolus avec CPLEX (dans une limite de temps de 10 heures). On constate également que la différence entre les deux techniques et CPLEX est plus prononcée pour les problèmes de grande taille : résoudre les problèmes avec 500 matchs de manière exacte prend presque 6 heures, **LAP-HYM** requiert environ 22 minutes alors que le temps requis par **HLA-HOA** ne dépasse jamais 12 secondes.

3.4 Adaptation de la méthode de recherche avec tabous

Nous proposons dans cette section une adaptation de la métaheuristique de recherche avec tabous. Nous rappelons d'abord les principes de cette méthode (paragraphe 3.4.1). Par la suite, dans le paragraphe 3.4.2, nous introduisons une version pénalisée du modèle mathématique original M^1 décrit au paragraphe 3.1.1, puis présentons la procédure que nous proposons pour résoudre cette nouvelle formulation du problème. Nous détaillons les différentes composantes de cette procédure dans les paragraphes 3.4.3, 3.4.4 et 3.4.5. Nous terminons cette section par les résultats des expérimentations numériques.

3.4.1 Présentation de la méthode

La méthode de recherche avec tabous (RT) a été développée indépendamment par GLOVER [46] et HANSEN [53] dans les années 80. Elle peut être vue comme une généralisation des méthodes de recherche locale dans la mesure où elle explore itérativement l'ensemble des solutions réalisables X d'un problème en se déplaçant d'une solution courante x vers une nouvelle solution x' située dans le voisinage de x ($x' \in V(x)$).
Contrairement aux méthodes itératives classiques, la RT autorise les déplacements dégradant la qualité de la solution courante. Si cette modification du processus d'exploration permet de continuer l'exploration au-delà des optima locaux, elle introduit, en revanche, le risque de cycler. Pour contrer cet effet, la méthode utilise un mécanisme

de mémoire, dit liste tabou (LT), interdisant temporairement le retour vers des solutions récemment visitées. Le schéma de base de la méthode est résumé dans la figure 3.8.

Initialisation
Générer une solution initiale x^0
Initialiser la solution courante $x := x^0$, et la meilleure solution $x^* := x^0$
Initialiser la liste tabou $LT := \emptyset$
Algorithme
Tant que (le critère d'arrêt n'est pas satisfait) **faire**
  Générer le voisinage de la solution courante $V(x)$
  Sélectionner la meilleure solution voisine $x' \in V(x)$ telle que $x' \notin LT$
  Si (x' est meilleure que x^*) **Alors**
    $x^* := x'$
  Fin Si
  $x := x'$
  Mettre à jour la liste tabou
Fait

Figure 3.8 – Schéma de base de la méthode de recherche avec tabous

Dans la description que nous venons de faire, nous avons introduit les éléments de base de la méthode de recherche avec tabous. Nous allons maintenant les définir d'une manière plus formelle, et indiquer brièvement les approches les plus utilisées dans la littérature pour les mettre en place. Cette revue ne se veut pas exhaustive. Nous renvoyons le lecteur à des présentations générales consacrées à la méthode comme le livre de GLOVER [48].

Ensemble des solutions : c'est le domaine dans lequel s'effectue la recherche. La manière la plus simple pour le définir consiste à le restreindre à l'ensemble des solutions réalisables, soit celles qui satisfont toutes les contraintes du problème traité. Or, cette approche soulève parfois quelques difficultés notamment si le problème est fortement contraint. On élargit alors l'ensemble des solutions pour inclure des solutions non réalisables. Bien entendu, la fonction objectif est modifiée pour tenir

compte de la violation éventuelle des contraintes.

Solution initiale : comme son nom l'indique, la solution initiale est utilisée pour amorcer la recherche. Deux approches sont envisageables pour générer cette solution de départ. La première et la plus simple consiste à générer une solution aléatoirement. La seconde approche fait appel à une heuristique constructive pour générer une solution de bonne qualité. Notons toutefois qu'une bonne solution initiale ne garantit pas l'obtention d'une meilleure solution finale.

Voisinage : il contrôle le déplacement à l'intérieur de l'ensemble des solutions. Formellement, le voisinage d'une solution x, qu'on appelle aussi l'ensemble des solutions voisines, est défini comme suit [34] : $V(x) = \{x' \in X \quad : \quad x' = x \oplus m\}$ est l'ensemble des solutions obtenues en appliquant des modifications $m \in \mathtt{M}$ à x. La modification apportée à la solution est appelée communément mouvement. L'ensemble des mouvements $\mathtt{M}$ est spécifique au problème considéré.

Liste tabou : elle agit lors de l'évaluation du voisinage pour interdire le choix des solutions voisines récemment visitées. La manière la plus simple pour mettre en place cette composante de la méthode consiste à conserver une liste des dernières solutions rencontrées et à interdire à la procédure d'y retourner. Cependant, cette approche peut se révéler très coûteuse. On se limite plutôt à garder une liste des mouvements récemment utilisés pour générer les solutions courantes, ou une liste de certaines de leurs caractéristiques. On peut interdire le choix d'une solution voisine si toutes les caractéristiques du mouvement utilisé pour la générer sont dans la liste tabou. On peut également être plus restrictif et l'interdire si au moins l'une de ces caractéristiques figure dans la liste. Habituellement, la durée des interdictions (longueur de la liste tabou) est fixe et dépend de la taille de l'instance traitée [47]. L'ajustement de cette valeur est cependant délicat : une liste trop longue restreint l'exploration du voisinage (plusieurs solutions sont tabou) alors qu'une liste trop courte ne prévient pas suffisamment le cyclage. Pour éviter cette difficulté, on modifie la longueur de la liste au cours de la recherche [26, 108].

Critère d'arrêt : plusieurs critères d'arrêt peuvent être utilisés. Le plus souvent on

arrête la procédure après un certain nombre d'itérations, après un certain temps CPU, ou après qu'un certain nombre d'itérations successives aient été complétées sans qu'il y ait eu amélioration de la meilleure solution trouvée.

On recense dans la littérature un grand nombre de concepts et de raffinements qui peuvent s'ajouter au schéma de base de la méthode de recherche avec tabous. Nous allons brièvement aborder ceux que nous utiliserons dans la suite de ce chapitre :

Critère d'aspiration : le statut tabou permet certes de contrer l'effet de cyclage à court terme, mais il peut s'avérer parfois préjudiciable. En effet, il se pourrait que des solutions qui n'ont jamais été visitées soient interdites. Le mécanisme d'aspiration permet alors d'outrepasser cette interdiction sous certaines conditions. Le plus souvent, on accepte une solution tabou si sa valeur est meilleure que celle de la meilleure solution rencontrée au cours de la recherche.

Stratégie de sélection dans le voisinage : tel qu'illustré dans la figure 3.8, pour sélectionner la solution dans le voisinage de la solution courante, on génère le voisinage au complet et on retient la meilleure solution voisine non tabou ou l'une des meilleures solutions tabou vérifiant le critère d'aspiration (stratégie *meilleure amélioration*). Or, un parcours exhaustif du voisinage se révèle souvent coûteux. Une autre manière de procéder consiste à interrompre la génération du voisinage dès qu'une première solution voisine non tabou améliorant la valeur de la solution courante est rencontrée, ou dès qu'une première solution tabou vérifiant le critère d'aspiration est atteinte. Si une telle solution n'a pas été identifiée, alors la meilleure solution voisine non tabou est sélectionnée (stratégie *première amélioration*). Dans certains contextes, la stratégie *première amélioration* s'est révélée meilleure que la stratégie *meilleure amélioration*. Elle a permis d'obtenir de meilleures solutions tout en nécessitant des temps de résolution nettement plus faibles [65].

Diversification : l'objectif de la diversification est de diriger la recherche vers des régions faiblement explorées de l'ensemble des solutions. On distingue deux formes de diversification. La première est utilisée continuellement au cours de l'explora-

tion du voisinage. Elle consiste à enregistrer la fréquence d'utilisation des mouvements, puis à réduire les chances de choisir une solution voisine en réduisant sa valeur proportionnellement à la fréquence d'utilisation du mouvement requis pour la générer [49]. Dans le même ordre d'idées, une autre technique consiste à interdire d'effectuer les mouvements dont la fréquence d'occurrence dans la recherche dépasse un certain seuil [26]. La seconde forme de diversification consiste à interrompre périodiquement la procédure et à la relancer à partir d'une nouvelle solution initiale. Les techniques utilisées pour générer cette nouvelle solution sont variées et résultent le plus souvent d'un compromis : la solution générée est de bonne qualité et se trouve dans une région peu ou pas explorée. Une description plus détaillée de ces techniques est fournie dans [34].

Intensification : à l'opposé de la diversification, l'objectif de l'intensification est d'approfondir la recherche dans certaines régions de l'ensemble des solutions jugées prometteuses. La technique la plus simple consiste à reprendre l'exploration à partir de la meilleure ou de l'une des meilleures solutions obtenues au cours de la recherche [103]. D'autres techniques consistent à dégager les propriétés communes des meilleures solutions rencontrées et à modifier les règles de sélection pour favoriser leur présence [49].

3.4.2 Version pénalisée du modèle original et approche de résolution

Dans le paragraphe 3.1.1, nous avons présenté un modèle mathématique (M^1) pour formuler *le problème sur une seule ronde*. Dans ce modèle, les règles incontournables d'affectation sont utilisées pour définir les contraintes du problème. La fonction objectif est spécifiée en termes des règles souples d'affectation en vue de réduire le nombre de fois où plusieurs juges affectés à un match possèdent le même domaine d'expertise, et de maximiser le nombre de matchs avec 5 juges affectés. La complexité de ce modèle rend difficile la définition d'un voisinage qui conserve la réalisabilité des solutions. Il nous a alors paru plus judicieux de relaxer les contraintes du problème afin d'obtenir un voisinage plus maniable.

Nous cherchons à ramener le problème à un problème qui comporte uniquement les contraintes classiques d'affectation. La modélisation que nous proposons s'inspire de celle que nous avons adoptée lors de la seconde phase de la méthode hybride **HYM** (cf. paragraphe 3.3.2). Rappelons que lors de cette seconde phase, notre objectif était d'affecter deux juges au maximum de matchs. Dans le cas présent, nous cherchons à en affecter 5. Nous associons donc 5 nœuds destination j, $(M+j)$, $(2M+j)$, $(3M+j)$, et $(4M+j)$ à chaque match j. Si $N > 5M$, alors $(N-5M)$ nœuds destination fictifs sont ajoutés. Ces nœuds sont associés à un match fictif $(M+1)$. Chaque nœud source est associé à un juge i. Supposons maintenant que le nombre de juges $N < 5M$. Nous devons alors affecter 3 juges à chaque match, et une paire de juges additionnelle à $\lfloor \frac{N-3M}{2} \rfloor$ matchs. Nous ajoutons donc $\kappa = (N-3M) \bmod 2$ nœud destination fictif que nous associons au match fictif $(M+1)$, et $(5M-N+\kappa)$ nœuds source que nous associons à des juges fictifs.

L'extension que nous proposons requiert également certaines modifications au niveau de la fonction objectif. Dans le nouveau modèle, nous visons à minimiser la somme pondérée des violations de toutes les contraintes. Rappelons brièvement les contraintes qui régissent *le problème sur une seule ronde* :

Les règles incontournables :

1. un juge ne peut être affecté à un match impliquant une équipe représentant l'Université dans laquelle il a étudié ou à laquelle il est attaché (contrainte d'admissibilité) ;
2. 3 ou 5 juges doivent être affectés à chaque match ;
3. au moins un *juge en chef* doit être affecté à chaque match.

Les règles souples :

4. les expertises des juges affectés à un même match doivent être aussi différentes que possible pour couvrir le plus grand nombre de domaines d'expertise (contrainte de *diversité*) ;
5. le nombre de matchs avec 5 juges affectés doit être maximisé (contrainte du *nombre*).

Dénotons par $\sum_{j=1}^{M} ad_j(x)$, $\sum_{j=1}^{M} nj_j(x)$, $\sum_{j=1}^{M} lj_j(x)$, et $\sum_{j=1}^{M} dv_j(x)$ la somme des violations des contraintes 1, 2, 3, et 4 respectivement. Le nouveau modèle mathématique se résume comme suit :

$$\min C \sum_{j=1}^{M} \left(ad_j(x) + nj_j(x) + lj_j(x)\right) + \sum_{j=1}^{M} dv_j(x) \tag{3.12}$$

(M^2) **Sujet à**

$$\sum_{\mu=1}^{\overline{N}} x_{i\mu} = 1 \qquad i = 1,\ldots,\overline{N} \tag{3.13}$$

$$\sum_{i=1}^{\overline{N}} x_{i\mu} = 1 \qquad \mu = 1,\ldots,\overline{N} \tag{3.14}$$

$$x_{i\mu} = 0 \text{ ou } 1 \qquad i = 1,\ldots,\overline{N},\ \mu = 1,\ldots,\overline{N} \tag{3.15}$$

Figure 3.9 – Modèle M^2 (version pénalisée du modèle original M^1)

où,

$$\overline{N} = \begin{cases} 5M + \kappa & \text{si } N < 5M \\ N & \text{sinon.} \end{cases}$$

Notons que les nœuds destination du modèle sont dénotés par $\mu = 1,\ldots,\overline{N}$ alors que les matchs sont dénotés par $j = 1,\ldots,M$, et qu'exactement 5 nœuds destination sont associés à chaque match $j \neq M+1$. C est une constante assez grande servant à accorder un facteur de pénalité supérieur à la violation des règles incontournables d'affectation. Notons également qu'aucun terme correspondant à la violation de la contrainte du *nombre* (contrainte 5 ci-dessus) n'apparaît dans la fonction objectif (3.12), car la modélisation que nous proposons veille implicitement au respect de cette contrainte.

Explicitons maintenant les termes $ad_j(x)$, $nj_j(x)$, $lj_j(x)$ et $dv_j(x)$ en fonction des variables de décision $x_{i\mu}$:

1. Le terme $ad_j(x)$ doit refléter le nombre de juges non admissibles affectés au match

j. Il s'écrit naturellement :

$$ad_j(x) = \sum_{\beta=0}^{4} \sum_{i=1}^{N} (1 - a_{ij}) x_{i(\beta M+j)} \quad j = 1, \ldots, M.$$

2. Le terme $nj_j(x)$ doit refléter le respect de la contrainte 2 dans le match j. Dénotons par $F_j(x) = \sum_{\beta=0}^{4} \sum_{i=N+1}^{\overline{N}} x_{i(\beta M+j)}$ le nombre de juges fictifs affectés au match j. Alors, $nj_j(x)$ doit être égal à 0 si et seulement si $F(j) = 0$ ou 2. L'expression :

$$nj_j(x) = \max\left(\left\lfloor \frac{F_j(x)}{2} \right\rfloor - 1, F_j(x) - 2\left\lfloor \frac{F_j(x)}{2} \right\rfloor\right) \quad j = 1, \ldots, M$$

nous permet de formaliser la contrainte d'avoir 3 ou 5 juges affectés au match j.

3. $L_j(x) = \sum_{\beta=0}^{4} \sum_{i=1}^{N} l_i x_{i(\beta M+j)}$ correspond au nombre de *juges en chef* affectés au match j. Nous voulons que le terme $lj_j(x)$ prenne une valeur nulle si et seulement si $L_j(x) \geq 1$. Nous le définissons donc :

$$lj_j(x) = \max\left(1 - L_j(x), 0\right) \quad j = 1, \ldots, M.$$

4. $d_{kj}(x) = \sum_{\beta=0}^{4} \sum_{i=1}^{N} e_{ik} x_{i(\beta M+j)}$ représente le nombre de juges possédant le domaine d'expertise k affectés au match j. Nous cherchons à ce que chaque domaine d'expertise soit représenté au plus une fois dans le match j. Le dernier terme a donc pour expression :

$$dv_j(x) = \sum_{k=1}^{K} \max\left(d_{kj}(x) - 1, 0\right) \quad j = 1, \ldots, M.$$

La procédure de résolution comprend deux étapes. Elle est initialisée avec une solution réalisable du modèle M^2. Dans un premier temps, nous utilisons une recherche avec tabous pour réduire le nombre de violations des différentes règles d'affectation (i.e., pour optimiser la fonction objectif (3.12)). Par la suite, nous utilisons une stratégie de diver-

sification pour générer une nouvelle solution initiale et réinitialiser la procédure.
Dans le but de pouvoir comparer plus facilement les différentes méthodes de résolution que nous avons développées, le critère d'arrêt de ces procédures est spécifié en termes d'un temps maximal de résolution *tempsmax*. Bien entendu, la procédure s'arrête également dès qu'une solution optimale est générée. Notons qu'une solution est optimale si sa valeur est égale à 0 ou à une borne inférieure b_{inf} connue *a priori* pour le problème.

Soit *xbest* la meilleure solution générée avec la procédure présentée ci-dessus. Si $xbest_{i(\beta M+j)} = 1$ où $\beta \in [0,4]$, alors le juge i est affecté au match j.

3.4.3 Solution initiale

Nous considérons deux processus différents pour générer la solution initiale. Dans le processus aléatoire, les $\overline{N}$ juges (nœuds source) sont affectés aléatoirement aux $\overline{N}$ nœuds destination associés aux matchs.

Le second processus est la technique heuristique **HLA-HOA** introduite à la section 3.3. Nous distinguons quatre cas :

Cas 1. Si $N \geq 5M$ et dans la solution produite par **HLA-HOA** 5 juges sont affectés à chaque match, alors ces $5M$ juges (nœuds source) sont affectés aux $5M$ nœuds destination associés aux matchs respectifs auxquels ils sont assignés. Les autres juges (s'il en reste) sont affectés aux $(\overline{N} - 5M)$ nœuds destination associés au match fictif $(M+1)$.

Cas 2. Si $N \geq 5M$ et dans la solution produite par **HLA-HOA** certains matchs comptent un seul juge (respectivement 3 juges) affecté(s), alors nous complétons d'abord les affectations dans ces matchs en leur assignant 4 juges (respectivement 2 juges) additionnels choisis aléatoirement parmi ceux qui sont disponibles. Nous obtenons ainsi une solution où 5 juges sont affectés à chaque match, ce qui nous ramène au cas précédent.

Cas 3. Si $N < 5M$ et dans la solution produite par **HLA-HOA** 3 ou 5 juges sont affectés à chaque match et il ne reste pas de juges non affectés ou il en reste un seul, alors, afin d'obtenir une solution avec 5 juges dans chaque match, nous affectons des

paires de juges fictifs à chacun des matchs avec 3 juges. Nous nous retrouvons ainsi dans le cas 1.

Cas 4. Si $N < 5M$ et il reste au moins une paire de juges non affectée, alors nous procédons comme dans le cas 2 en vue de compléter les affectations dans les matchs avec 1 ou 3 juges seulement, mais en considérant cette fois-ci l'ensemble des juges fictifs en plus de ceux non affectés.

La solution initiale est améliorée en utilisant une recherche avec tabous. Les composantes de cette méthode sont présentées dans le paragraphe qui suit.

3.4.4 Recherche avec tabous

Le voisinage d'une solution réalisable x du modèle M^2 est généré en échangeant l'affectation de deux juges i et r affectés présentement à deux matchs différents j et l respectivement. Nous dénotons la nouvelle solution voisine générée par $x \oplus (i,j,r,l)$. Notons que si les deux juges i et r sont fictifs, échanger leurs affectations n'a aucune influence sur la valeur de la solution courante. Aussi, afin de mieux guider la recherche, considérons-nous uniquement les solutions voisines $x \oplus (i,j,r,l)$ telles que $i \leq N$ ou $r \leq N$. Nous dirons, dans ce cas, que le mouvement (i,j,r,l) est *intéressant*.

Afin de minimiser le risque de cycler, les réaffectations récemment utilisées sont déclarées tabou. Pour simplifier la mise à jour de la liste tabou et la vérification du statut d'une solution, nous utilisons une matrice tabou $LT = [LT_{ij}]$ où, LT_{ij} identifie l'itération après laquelle le juge i peut être affecté au match j. Lorsque nous nous déplaçons de x vers $x \oplus (i,j,r,l)$, deux éléments de la matrice tabou sont modifiés comme suit :

$$LT_{ij} = curiter + t_1 \quad \text{et} \quad LT_{rl} = curiter + t_2$$

où, *curiter* représente l'indice de l'itération courante, et t_1 et t_2 sont des nombres entiers choisis aléatoirement dans l'intervalle $[t_{min}, t_{max}]$.

D'autre part, étant donnée la matrice tabou courante LT, une solution voisine $x \oplus (i,j,r,l)$

est tabou à l'itération *iter* si :

$$LT_{il} \geq iter \quad \text{ou} \quad LT_{rj} \geq iter.$$

Le critère tabou que nous avons choisi est sévère et peut restreindre l'exploration. En effet, nous interdisons toute solution qui viserait à réaffecter l'un des deux juges au match auquel il était affecté durant les dernières itérations. Pour ne pas omettre de bonnes solutions, nous utilisons le critère d'aspiration classique pour révoquer le statut tabou d'une solution lorsque sa valeur est meilleure que celle de la meilleure solution rencontrée jusqu'à présent.

Pour choisir le mouvement (i,j,r,l) à utiliser pour générer la prochaine solution courante, nous avons considéré et comparé numériquement les deux stratégies *meilleure amélioration* et *première amélioration*. De plus, nous considérons la fréquence d'utilisation des mouvements.

Pour spécifier la fréquence d'utilisation d'un mouvement, nous enregistrons la fréquence à laquelle nous avons utilisé les caractéristiques du mouvement plutôt que la fréquence du mouvement lui-même. Plus explicitement, ce n'est pas le nombre de fois que le mouvement (i,j,r,l) a été sélectionné que nous considérons, mais le nombre de fois que le juge i a été affecté au match l et que le juge r a été affecté au match j. Nous utilisons donc une matrice $Freq = [Freq_{ij}]$ que nous mettons à jour à chaque itération : si $x \oplus (i,j,r,l)$ devient solution courante, alors nous incrémentons d'une unité les valeurs des deux éléments $Freq_{il}$ et $Freq_{rj}$. Si plusieurs solutions voisines ont la même qualité (en termes de la valeur de la fonction objectif), alors nous nous basons sur la mesure suivante pour déterminer celle que nous choisirons :

$$P(i,j,r,l) = \sqrt{Freq_{il}} + \sqrt{Freq_{rj}}.$$

Notons que ce critère de sélection peut également être utilisé dans le cadre de la stratégie *première amélioration* lorsque tout le voisinage est énuméré sans pouvoir identifier une solution voisine non tabou améliorant la solution courante, ou une solution tabou satis-

faisant le critère d'aspiration. En effet, dans ce cas, cette stratégie est équivalente à celle de la *meilleure amélioration.*

La mesure P offre trois avantages principaux :

- elle constitue *ipso facto* une stratégie de diversification mise en œuvre à chaque itération. Notons que la racine carrée est utilisée pour diriger la recherche vers des régions peu ou pas explorées. Pour illustrer ce propos, considérons deux solutions voisines candidates $x \oplus (i,j,r,l)$ et $x \oplus (i',j',r',l')$ (i.e., qui induisent la même modification dans la fonction objectif), et telles que $(Freq_{il} + Freq_{rj}) = (Freq_{i'l'} + Freq_{r'j'})$. Alors, le critère basé sur l'expression $P(i,j,r,l)$ ci-dessus, indique de sélectionner $x \oplus (i,j,r,l)$ si $min(Freq_{il}, Freq_{rj}) \leq min(Freq_{i'l'}, Freq_{r'j'})$;
- son utilisation en tant que second critère d'évaluation permet de mieux guider la recherche en départageant les solutions voisines de même qualité sans altérer les coûts originaux (si on ajoute une mesure de fréquence pondérée à la fonction objectif par exemple) ;
- elle permet de remédier aux problèmes de cyclage et de diriger plus efficacement la recherche à long terme. En effet, supposons que la mesure P n'est pas utilisée et que la solution initiale utilisée pour réinitialiser la procédure (solution obtenue suite à la diversification) est similaire à une solution générée précédemment. Il y a alors de très fortes chances de retourner vers des solutions antérieures au cours de la recherche. La mesure de fréquence P permet de contrer cet inconvénient puisqu'elle tient compte du processus de résolution dans sa globalité.

Il est clair que le temps de calcul nécessaire à l'évaluation des solutions voisines influe considérablement sur les performances de la méthode puisque cette phase est effectuée à chaque itération. Souvent, on préfère calculer la modification induite dans la fonction objectif $(f(x \oplus m) - f(x))$ plutôt que le coût de la solution voisine $f(x \oplus m)$. Les simplifications algébriques permettent alors d'accélérer l'évaluation du voisinage. Dans notre cas, outre cette simplification, nous utilisons une matrice $\Delta(x) = [\Delta_{ir}(x)]$ où, $\Delta_{ir}(x)$ représente la valeur de la modification induite dans la fonction objectif lorsqu'on applique le mouvement (i,j,r,l) à la solution courante x (i.e., $\Delta_{ir}(x) = f(x \oplus (i,j,r,l)) - f(x)$). À chaque itération, nous mettons à jour certaines des valeurs contenues dans $\Delta(x)$.

En effet, si à l'itération courante $x \oplus (i,j,r,l)$ est choisie comme solution courante pour la prochaine itération, alors seules les valeurs des mouvements impliquant les juges affectés aux matchs j et l nécessitent d'être modifiées.

Finalement, le critère d'arrêt de la méthode est spécifié en termes d'un nombre maximal *nitermax* d'itérations successives complétées sans qu'il y ait eu amélioration de la fonction objectif. La méthode s'arrête également dès qu'une solution optimale est générée.

Nous résumons dans la figure 3.10 la méthode lorsque nous utilisons la stratégie *meilleure amélioration*.

3.4.5 Stratégie de diversification

Une fois que nous avons complété la première étape (recherche avec tabous), si la meilleure solution générée n'est pas optimale, et si le temps est inférieur au temps maximal alloué *tempsmax*, nous générons une nouvelle solution initiale pour réinitialiser la procédure de résolution.

La nouvelle solution initiale x^0 est générée à partir de *xbest*, la meilleure solution rencontrée jusqu'à présent, en appliquant une série d'échanges. Nous procédons en deux phases. Dans un premier temps, nous mettons en place une liste de sélection L qui contient les mouvements susceptibles d'apporter une modification significative à *xbest*. Le choix des mouvements utilisés pour générer x^0 s'effectue par la suite dans cette liste.

Liste de sélection. D'abord, nous partitionnons l'ensemble des juges (disponibles et fictifs) en deux ensembles disjoints W et G. Dénotons par $I(j,xbest)$ l'ensemble des juges affectés au match j dans la solution *xbest*. Les juges i faisant partie de l'ensemble W sont choisis comme suit :

- si $ad_j(xbest) > 0$ (i.e., la contrainte d'admissibilité est violée dans le match j) et $i \in I(j,xbest)$ n'est pas admissible au match j ;
- si $nj_j(xbest) > 0$ (i.e., la contrainte stipulant qu'exactement 3 ou 5 juges doivent être affectés à chaque match est violée dans le match j) et $i \in I(j,xbest)$ est un juge fictif ;

Initialisation
x^0, une solution initiale
$x := x^0$, la solution courante
$x^* := x^0$, la meilleure solution générée par la méthode
$\Delta(x)$, la matrice contenant la valeur des mouvements
LT, la liste tabou
$Freq$, la matrice des fréquences
$niter := 0$, le nombre d'itérations successives sans améliorer la fonction objectif
$fin :=$ **faux**
Algorithme
Tant que ($fin =$ **faux**) **faire**
  $niter := niter + 1$, $E := \emptyset$, $\Delta_{min} := \infty$
  EXPLORATION DES VOISINS
  Pour chaque mouvement *intéressant* (i,j,r,l) **faire**
    Si ($x \oplus (i,j,r,l)$ est non tabou **OU** vérifie le critère d'aspiration) **Alors**
      Si ($\Delta_{ir}(x) < \Delta_{min}$) **Alors**
        $\Delta_{min} := \Delta_{ir}(x)$, $E := \{(i,j,r,l)\}$
      Sinon
        Si ($\Delta_{ir}(x) = \Delta_{min}$) **Alors** $E := E \cup \{(i,j,r,l)\}$ **Fin Si**
      Fin Si
    Fin Si
  Fin Pour
  REMPLACEMENT DE LA SOLUTION COURANTE
  Si ($E \neq \emptyset$) **Alors**
    $(i',j',r',l') := \arg\min_{e \in E} \{P(e)\}$, $x := x \oplus (i',j',r',l')$
  Fin Si
  MISE À JOUR
  Si ($E \neq \emptyset$) **Alors**
    Mettre à jour $\Delta(x)$, LT et $Freq$
    Si ($f(x) < f(x^*)$) **Alors**
      $x^* := x$, $niter := 0$
    Fin Si
  Fin Si
  ÉVALUATION DES CRITÈRES D'ARRÊT
  Si ($f(x) = 0$ ou b_{inf} **OU** $niter = nitermax$) **Alors**
    $fin :=$ **vrai**
  Fin Si
Fait
x^* est la meilleure solution générée

Figure 3.10 – Recherche avec tabous utilisant la stratégie meilleure amélioration pour l'étape 1 de la RT

- si $lj_j(xbest) > 0$ (i.e., aucun *juge en chef* n'est affecté au match j), alors un juge $i \in I(j,xbest)$ est choisi aléatoirement parmi ceux affectés au match j ;
- si $dv_j(xbest) > 0$ (i.e., la contrainte de *diversité* est violée dans le match j), pour chaque paire de juges possédant le même domaine d'expertise, nous choisissons aléatoirement un juge $i \in I(j,xbest)$ pour l'inclure dans W.

L'ensemble G est formé par les juges qui ne font pas partie de W.

Par la suite, la liste de sélection est construite séquentiellement en y ajoutant les mouvements qui vérifient les trois critères suivants :

1. $i \in W$ et $r \in G$;
2. i et r sont affectés à des matchs différents (i.e., $i \in I(j,xbest)$, $r \in I(l,xbest)$ et $j \neq l$) ;
3. i et r ne sont pas tous les deux des juges fictifs (i.e., (i,j,r,l) est un mouvement *intéressant*).

Génération de la nouvelle solution initiale. Nous utilisons le processus séquentiel suivant, résumé dans la figure 3.11, pour générer une nouvelle solution initiale x^0.

x^0 est initialisée avec *xbest*. Considérons l'ensemble des mouvements $(i,j,r,l) \in L$. À chaque itération, nous sélectionnons parmi les mouvements dans L, celui qui a été le moins fréquemment utilisé selon la mesure de fréquence P introduite dans le paragraphe précédent. Si plusieurs mouvements dans L ont la même valeur de fréquence, alors nous sélectionnons celui qui conduit à la solution ayant la meilleure valeur de la fonction objectif. En cas d'égalité, le choix se fait aléatoirement. Le mouvement sélectionné (i',j',r',l') est alors appliqué à x^0 (i.e., $x^0 := x^0 \oplus (i',j',r',l')$), et la matrice des fréquences est mise à jour. De plus, afin de ne pas retourner à la solution *xbest* à partir de laquelle la nouvelle solution x^0 a été générée, la liste tabou est également mise à jour. Par la suite, tous les mouvements impliquant les juges i' ou r' sont éliminés de la liste de sélection L. Le processus s'arrête lorsque la liste L est vide.

Notons que nous avons déjà utilisé une première stratégie de diversification dans la phase d'exploration du voisinage de la solution courante (cf. paragraphe 3.4.4). Cependant, son impact reste limité. Favoriser les mouvements les moins fréquemment utilisés

rapproche certes la recherche vers les régions les moins explorées, mais le déplacement peut nécessiter plusieurs itérations. La stratégie considérée dans ce paragraphe permet d'effectuer des déplacements rapides vers les régions cibles. De plus, elle tient compte du contexte (liste de sélection), et de l'état courant de l'exploration (critères de sélection des mouvements utilisés pour générer x^0). Elle confère à la méthode la possibilité de mieux s'adapter aux spécificités du problème permettant de guider la recherche de manière plus efficace, et de la diriger vers des régions à la fois inexplorées et prometteuses.

3.4.6 Expérimentations numériques

Nous comparons quatre variantes de la procédure de résolution. Ces variantes sont spécifiées en termes du processus utilisé pour générer la solution initiale (**HLA-HOA** et aléatoirement que nous dénotons par **H** et **R** respectivement), et de la stratégie utilisée pour sélectionner la solution dans le voisinage de la solution courante (*meilleure amélioration* et *première amélioration* que nous dénotons par **Best** et **First** respectivement) : **RT-H-Best**, **RT-H-First**, **RT-R-Best**, et **RT-R-First**.

La première phase des expérimentations a consisté à ajuster les valeurs des différents paramètres de la procédure de résolution. Nous avons fixé la valeur de la constante C (utilisée dans le modèle M^2 pour accorder un facteur de pénalité supérieur à la violation des règles incontournables d'affectation) à $50M^2$ (M étant le nombre de matchs), et nous avons testé les 18 différentes combinaisons générées à partir de :

- deux valeurs pour l'intervalle $[t_{min}, t_{max}]$ (dans lequel est choisie la durée tabou) : $[\lfloor 0,8N \rfloor, \lceil 1,2N \rceil]$ et $[\lfloor 0,9N \rfloor, \lceil 1,1N \rceil]$ (N étant le nombre de juges) ;
- trois valeurs de *nitermax* (nombre maximal d'itérations successives de la recherche avec tabous sans qu'il y ait eu amélioration de la fonction objectif) : N, $5N$ et $10N$;
- trois valeurs de *tempsmax* (temps maximal alloué à la procédure de résolution) : $2N$, $3N$ et $4N$ secondes.

Les quatre variantes ont servi comme support pour ces tests. Les instances que nous avons utilisées sont celles avec 500 matchs (qui semblent être les plus difficiles à résoudre). Les meilleurs résultats ont été obtenus avec la combinaison suivante :

Initialisation
xbest, la meilleure solution rencontrée jusqu'à présent
$x^0 := xbest$, la solution initiale à générer
L, la liste de sélection. Les mouvements qui y sont contenus sont classés par ordre croissant au moyen de la mesure P
$LT := \emptyset$, la liste tabou
Freq, la matrice des fréquences
Algorithme
Tant que $(L \neq \emptyset)$ **faire**
  SÉLECTION D'UN MOUVEMENT
  $e_1 :=$ tête de L, $E := \{e_1\}$, $s := 2$
  Tant que $(P(e_s) = P(e_1)$ **ET** $s \leq |L|)$ **faire**
    $E := E \cup \{e_s\}$, $s := s+1$
  Fait
  $Z := \{x^0 \oplus (i,j,r,l) \quad : \quad (i,j,r,l) \in E\}$
  $E' := \left\{(i,j,r,l) \quad : \quad f(x^0 \oplus (i,j,r,l)) = \min_{z \in Z} f(z)\right\}$
  Choisir aléatoirement $(i',j',r',l') \in E'$
  MISE À JOUR
  $x^0 := x^0 \oplus (i',j',r',l')$
  Mettre à jour *LT* et *Freq*
  $L := L - \{(i,j,r,l) \quad : \quad i = i' \text{ ou } r = r'\}$
Fait
x^0 est la nouvelle solution initiale générée pour réinitialiser la procédure de résolution

Figure 3.11 – Stratégie de diversification pour l'étape 2 de la RT

- $[t_{min}, t_{max}] = [\lfloor 0,8N \rfloor, \lceil 1,2N \rceil]$
- $nitermax = N$
- $tempsmax = 3N$.

C'est elle donc que nous utilisons dans tous les tests subséquents.

Nous reprenons l'ensemble des instances de test décrites au paragraphe 3.2.1 pour évaluer et comparer les quatre variantes. Dans un premier temps, nous analysons l'impact de la qualité de la solution initiale sur les performances de la procédure de résolution. Par la suite, nous analysons l'impact de la stratégie de sélection. Ensuite, nous comparons les quatre variantes entre elles. Finalement, nous analysons l'impact du mécanisme de mémoire à long terme (mesure P) sur les performances de chaque variante.

Chaque instance a été résolue 10 fois avec une valeur d'initialisation différente pour le générateur des nombres aléatoires. De plus, afin de disposer d'une plateforme de comparaison objective, les mêmes valeurs d'initialisation sont utilisées pour chaque variante.

Nous présentons dans le tableau 3.4 les résultats obtenus. Pour chaque sous-ensemble de problèmes, nous indiquons l'écart moyen entre les valeurs des solutions générées et la valeur optimale (lorsque nous disposons de cette information) ou la borne inférieure sur cette valeur (*Ave dev*), le pourcentage des solutions optimales trouvées (*%Opt*), le temps moyen de résolution en secondes (*Ave CPU*), et finalement l'écart-type des temps de résolution (σ_{CPU}). Notons que toutes les solutions générées par les quatre variantes sont toujours réalisables. Par conséquent, le critère *%Feas* ne figure pas dans ce tableau. Pour des fins de comparaison, nous donnons également dans la dernière colonne du tableau les résultats obtenus avec CPLEX. Rappelons que seulement certaines instances ont pu être résolues avec CPLEX dans une limite de temps de 10 heures. Pour cette colonne, *%Opt* indique donc le pourcentage de ces instances. Aussi, *Ave CPU* et σ_{CPU} concernent les temps de résolution de ces instances. *Ave dev* ne s'applique pas à CPLEX comme indiqué par NA dans les cellules correspondantes à ce critère. La figure 3.12 représente l'évolution de l'écart moyen (*Ave dev*) et du temps moyen de résolution (*Ave CPU*) en fonction de la taille des problèmes.

Si l'on effectue une comparaison entre CPLEX et les quatre variantes, celles-ci le surclassent indéniablement. En effet, les résultats du tableau 3.4 indiquent que les quatre

		Taille	RT-H-Best	RT-R-Best	RT-H-First	RT-R-First	CPLEX
Ave dev	P_1	15	0	0	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0	0	0	0	NA
	P_2	15	0	0	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0,12	0	0,13	0	NA
	P_3	15	0	0	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0,03	0	0,04	0	NA
%Opt	P_1	15	100	100	100	100	100
		50	100	100	100	100	60
		150	100	100	100	100	50
		500	100	100	100	100	50
	P_2	15	100	100	100	100	100
		50	100	100	100	100	100
		150	100	100	100	100	100
		500	91	100	90	100	100
	P_3	15	100	100	100	100	100
		50	100	100	100	100	60
		150	100	100	100	100	50
		500	97	100	96	100	40
Ave CPU (sec)	P_1	15	0,04	0,07	0,05	0,09	28,74
		50	0,17	0,47	0,17	0,37	2,31
		150	0,46	9,11	0,45	3,54	96,38
		500	12,34	411,99	11,40	85,98	12 159,96
	P_2	15	0,08	0,13	0,10	0,15	0,69
		50	1,05	1,05	1,17	0,97	1,93
		150	35,76	15,42	34,41	16,38	32,41
		500	3 189,12	969,25	3 327,33	667,98	21 299,74
	P_3	15	0,04	0,07	0,05	0,08	13,12
		50	0,19	0,48	0,17	0,44	8,85
		150	2,37	20,02	2,32	17,77	276,17
		500	994,93	899,32	940,77	740,20	23 824,47
σ_{CPU}	P_1	15	0,05	0,10	0,06	0,11	33,59
		50	0,21	0,40	0,18	0,31	0,58
		150	0,33	8,03	0,30	2,94	71,55
		500	3,53	419,26	2,13	124,89	5 408,41
	P_2	15	0,10	0,16	0,19	0,17	0,09
		50	1,39	1,07	1,53	0,84	0,25
		150	37,57	15,47	34,53	17,68	8,14
		500	3 214,04	1 312,49	3 406,61	1 210,45	4 130,39
	P_3	15	0,06	0,10	0,07	0,10	16,75
		50	0,23	0,34	0,23	0,33	12,51
		150	10,87	17,03	10,44	17,45	201,12
		500	2 311,41	687,82	2 266,87	723,50	4 216,50

Tableau 3.4 – Comparaison de l'efficacité des variantes de la RT en fonction des sous-ensembles de problèmes

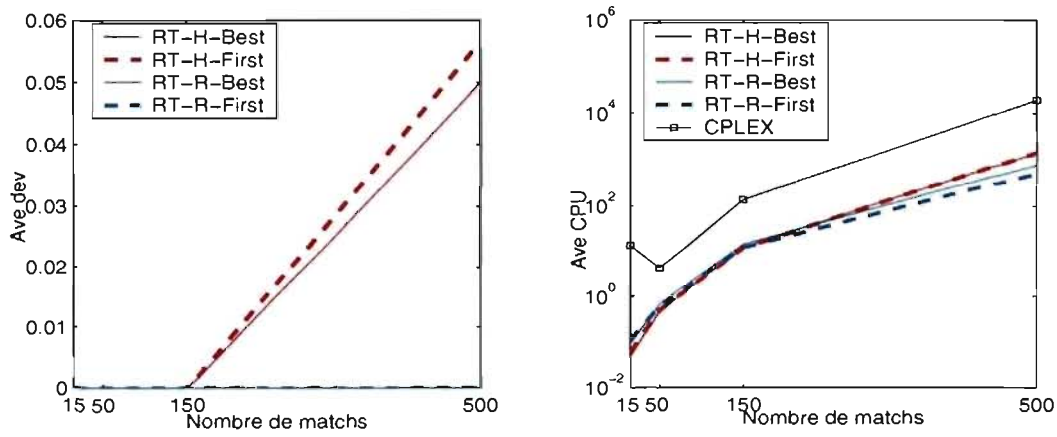

Figure 3.12 – Comparaison de l'efficacité des variantes de la RT en fonction de la taille des problèmes

variantes génèrent des solutions d'excellente qualité pour l'ensemble des instances alors que CPLEX n'est pas en mesure, dans une limite de temps de 10 heures, de trouver la solution optimale de plusieurs instances dans P_1 et P_3. Si l'on considère les temps de résolution (cf. deuxième graphique de la figure 3.12), on constate que les performances de CPLEX se situent nettement en retrait par rapport à celles des quatre variantes. En particulier, pour les instances avec 15 matchs, *Ave CPU* est réduit d'un facteur de 261, 222, 156, et 130 respectivement pour les variantes **RT-H-Best**, **RT-H-First**, **RT-R-Best**, et **RT-R-First**. Finalement, si l'on s'intéresse à l'écart-type (cf. tableau 3.4), on note que, globalement, les quatre variantes sont plus stables que CPLEX. Par conséquent, il semble avantageux d'utiliser l'approche métaheuristique puisqu'elle permet de générer des solutions d'excellente qualité tout en nécessitant des temps de résolution de loin meilleurs que CPLEX.

Solution initiale générée aléatoirement versus solution initiale de bonne qualité. Les résultats des problèmes avec 15, 50 et 150 matchs ainsi que ceux de l'ensemble P_1 avec 500 matchs n'indiquent pas d'impact de la qualité de la solution initiale sur la qualité des solutions générées. Pour les deux autres sous-ensembles (P_2 et P_3 avec

500 matchs), les deux variantes **RT-R-Best** et **RT-R-First** (où les solutions initiales sont générées aléatoirement) dominent légèrement les deux autres variantes, puisqu'elles génèrent toujours des solutions optimales (amélioration moyenne de 6,50%).

Si l'on s'intéresse au temps de résolution, les résultats sont partagés : pour les problèmes de l'ensemble P_1, on perçoit clairement la nette supériorité des deux variantes **RT-H-Best** et **RT-H-First** (temps de résolution réduit en moyenne d'un facteur de 20,41) tandis que pour ceux de l'ensemble P_2, les deux variantes **RT-R-Best** et **RT-R-First** semblent dominer. Quant aux problèmes de l'ensemble P_3, **RT-H-Best** et **RT-H-First** sont meilleures, sauf pour les gros problèmes avec 500 matchs.
Une explication plausible de la supériorité des deux variantes **RT-R-Best** et **RT-R-First** (en termes du temps de résolution) pour les problèmes dans P_2 et P_3 avec 500 matchs réside dans le fait que toutes les solutions produites par ces deux variantes sont optimales ce qui n'est pas le cas pour les deux autres variantes **RT-H-Best** et **RT-H-First**. En effet, rappelons que le critère d'arrêt des différentes variantes est spécifié en termes d'un temps de résolution maximal *tempsmax*. Elles s'arrêtent également dès qu'une solution optimale est générée. Ainsi, parfois, **RT-H-Best** et **RT-H-First** ne s'arrêtent qu'au bout de *tempsmax* secondes alors que **RT-R-Best** et **RT-R-First** s'arrêtent plus tôt puisqu'elles vérifient le deuxième critère d'arrêt.

Stratégie *meilleure amélioration* versus stratégie *première amélioration*. Les résultats obtenus n'indiquent pas un impact clair de la stratégie de sélection sur la qualité des solutions générées. En effet, si l'on se réfère au premier graphique de la figure 3.12, on peut observer que, excepté les problèmes avec 500 matchs pour lesquels la stratégie *meilleure amélioration* domine très légèrement lorsque la solution initiale est générée avec la technique **HLA-HOA**, les deux stratégies obtiennent les mêmes résultats.

En revanche, si l'on s'intéresse au temps de résolution, on constate que c'est la stratégie *première amélioration* qui se démarque, en particulier lorsque la solution initiale est générée aléatoirement. De plus, la différence entre les deux stratégies est plus prononcée pour les gros problèmes avec 500 matchs (cf. deuxième graphique de la figure 3.12).

Comparaison globale. En premier lieu, il convient de souligner l'excellente performance de la procédure de résolution indépendamment de la qualité de la solution initiale et de la stratégie de sélection dans le voisinage. L'écart moyen n'a jamais dépassé 1 ce qui signifie qu'en moyenne, pour chaque instance, la solution générée comprend au plus un match auquel sont affectés deux juges possédant le même domaine d'expertise. De plus, les quatre variantes sont très robustes dans le sens où, pour chacune des 120 instances testées, au moins l'une des 10 solutions générées est optimale. Mieux encore, des solutions optimales sont systématiquement obtenues par les deux variantes **RT-R-Best** et **RT-R-First**. On peut également observer que les temps de résolution de cette dernière variante sont en moyenne meilleurs.

Pour approfondir la comparaison, les résultats de l'ensemble des résolutions ont fait l'objet d'une analyse statistique basée sur le test des rangs pour échantillons appariés de WILCOXON. Pour chaque paire de variantes (A,B), nous avons testé l'hypothèse nulle selon laquelle les performances des deux variantes (qualité des solutions générées ou temps de résolution) sont identiques versus l'hypothèse alternative que la variante A est meilleure que la variante B. Les résultats de ce test sont résumés dans le tableau 3.5. Pour chaque entrée (A,B), nous indiquons la *p_valeur* obtenue. Le seuil de confiance utilisé étant de 0,05 ; une *p_valeur* inférieure à 0,05 indique que la variante A (dont le nom figure à la colonne 1 du tableau) domine la variante B (dont le nom figure à la ligne 2 du tableau). À l'opposé, une *p_valeur* supérieure à 0,05 indique que l'hypothèse nulle n'est pas rejetée, et donc qu'il n'y a pas de différence statistiquement significative entre les deux variantes A et B. Nous indiquons en caractère gras de telles valeurs.

	Qualité de la solution			Temps de résolution		
	RT-R-Best	**RT-H-First**	**RT-H-Best**	**RT-R-Best**	**RT-H-First**	**RT-H-Best**
RT-R-First	**1,00**	4,29E-04	1,23E-03	<2,2E-16	< 2,2E-16	< 2,2E-16
RT-R-Best	-	4,29E-04	1,23E-03	-	<2,2E-16	<2,2E-16
RT-H-First	-	-	**0,42**	-	-	9,42E-03

Tableau 3.5 – Résultats du test de WILCOXON pour les variantes de la RT

Les résultats obtenus nous permettent de conclure, qu'au niveau de la qualité des solutions, il n'y a pas de différence statistiquement significative, ni entre **RT-R-Best** et **RT-**

R-First, ni entre **RT-H-Best** et **RT-H-First**. Cependant, les deux premières variantes sont statistiquement meilleures que les deux dernières. Au niveau du temps de résolution, la variante **RT-R-First** se classe sans conteste en premier, **RT-R-Best** en second, **RT-H-First** en troisième, et **RT-H-Best** en dernier.

La figure 3.13 illustre le comportement de *Ave dev* durant la résolution. Chaque courbe est associée à une variante et montre les valeurs moyennes de *Ave dev* durant les 10 résolutions des problèmes avec 500 matchs, valeurs calculées à différents moments de la résolution. Notons que les problèmes avec 500 matchs sont ceux qui semblent être les plus difficiles à résoudre.

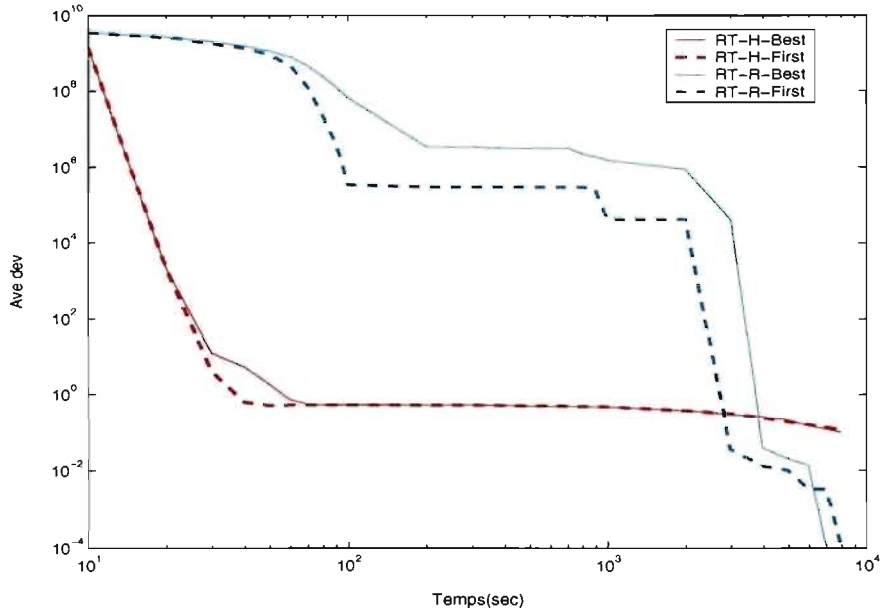

Figure 3.13 – Évolution de *Ave dev* des variantes de la RT pour les problèmes avec 500 matchs

Les courbes associées aux variantes **RT-H-Best** et **RT-H-First** ont à peu près la même allure. Au début, elles se confondent, ce qui est logique. En effet, c'est la période nécessaire pour générer la solution initiale et les deux variantes utilisent le même processus (**HLA-HOA**). Durant la période entre 30 et 60 secondes, **RT-H-First** affiche de meilleures performances que **RT-H-Best** indiquant qu'il n'est pas avantageux d'effec-

tuer une exploration exhaustive du voisinage. En effet, à ce moment, seule la diversité peut être améliorée car les solutions générées avec **HLA-HOA** vérifient toutes les règles incontournables d'affectation et comptent 5 juges affectés au maximum de matchs. Par conséquent, la meilleure amélioration qui peut être identifiée avec la stratégie *meilleure amélioration* est égale à 2 alors que l'utilisation la stratégie *première amélioration* peut identifier plus rapidement une modification induisant le même gain ou un gain égal à 1. Le temps supplémentaire investi dans l'énumération du voisinage au complet n'entraîne donc pas une amélioration significative de la valeur de la solution. Bien au contraire, ceci retarde la convergence vers des solutions de bonne qualité. Durant le reste du temps de résolution, les deux courbes stagnent. Elles se confondent car aucune amélioration n'est obtenue, et par conséquent, les deux stratégies *meilleure amélioration* et *première amélioration* sont équivalentes.

L'allure des courbes associées aux variantes **RT-R-Best** et **RT-R-First** indique que plusieurs applications de la stratégie de diversification sont nécessaires avant d'atteindre l'état où seule la diversité peut être améliorée. Elles surpassent les deux autres variantes puisqu'elles sont capables de réduire la valeur de *Ave dev* à 0.

La mauvaise performance de **RT-H-Best** et **RT-H-First** (comparativement aux deux autres variantes) peut se justifier par le fait que la technique **HLA-HOA** génère à la base des solutions de très bonne qualité qu'il est difficile d'améliorer. En effet, si l'on observe de plus près l'évolution de la valeur de la solution courante au cours du processus de résolution, le plus souvent, on constate une série de périodes de stagnation interrompues par de faibles perturbations enregistrées suite à l'application de la stratégie de diversification. De plus, l'efficacité de cette stratégie est réduite dans la mesure où elle ne permet pas de réaliser des déplacements importants dans l'ensemble des solutions, et la recherche reste confinée dans les régions proches de l'optimum local trouvé. En effet, comme il a déjà été expliqué au paragraphe 3.4.5, la nouvelle solution initiale x^0 est obtenue à partir de *xbest*, la meilleure solution trouvée jusqu'à présent, en lui appliquant une série d'échanges. Le nombre de ces échanges est fonction du nombre et de la nature des contraintes violées. Par conséquent, plus *xbest* est de bonne qualité (i.e., moins il y a de contraintes violées), plus x^0 est similaire à *xbest* (i.e., nous reprenons la recherche

à partir d'une solution qui ne diffère pas beaucoup de *xbest*).

Impact du mécanisme de mémoire à long terme sur les performances des quatre variantes. Les derniers tests sont destinés à évaluer l'impact de l'utilisation de la mémoire à long terme (mesure de fréquence P) sur la qualité des solutions générées ainsi que sur les temps de résolution. Pour ce faire, nous avons procédé comme suit :

- nous avons testé les quatre variantes dans les mêmes conditions que précédemment (mêmes instances, mêmes paramètres, mêmes valeurs d'initialisation du générateur des nombres aléatoires, 10 résolutions pour chaque instance), sauf que nous n'utilisons pas la mesure de fréquence P, ni dans la phase d'exploration du voisinage pour départager les solutions voisines candidates induisant la même modification dans la fonction objectif, ni dans la phase de diversification pour sélectionner les mouvements à appliquer pour générer la nouvelle solution initiale. Nous dénotons par $(Ave\,dev)'$ et $(Ave\,CPU)'$ respectivement l'écart moyen entre les valeurs des solutions générées et la valeur optimale (ou la borne inférieure sur cette valeur), et le temps moyen de résolution (en secondes) ;
- pour chaque sous-ensemble de problèmes et chaque variante, nous avons calculé le gain moyen en termes de la qualité des solutions ($Gdev = (Ave\,dev)' - Ave\,dev$), et en termes du temps de résolution ($GCPU = \frac{(Ave\,CPU)'}{Ave\,CPU}$). Ainsi, des valeurs de $Gdev$ et de $GCPU$ supérieures respectivement à 0 et à 1, indiquent que l'utilisation de la mesure de fréquence P a été bénéfique.

Nous reportons dans la figure 3.14 les résultats obtenus. Les deux graphiques représentent respectivement, en abscisse le nombre de matchs, et en ordonnée les valeurs de $Gdev$ et de $GCPU$.

Globalement, les résultats sont en faveur de l'utilisation de la mémoire à long terme. Les améliorations les plus importantes ont lieu pour les variantes **RT-H-First** et **RT-R-First** (à l'exception des instances avec 15 matchs pour lesquels on note une légère dégradation des temps de résolution due sans doute à l'effort de calcul supplémentaire). Quant à **RT-R-Best**, on constate également une amélioration de la qualité des solutions accompagnée d'une réduction, certes moins importante, des temps de résolution. Le gain est moins

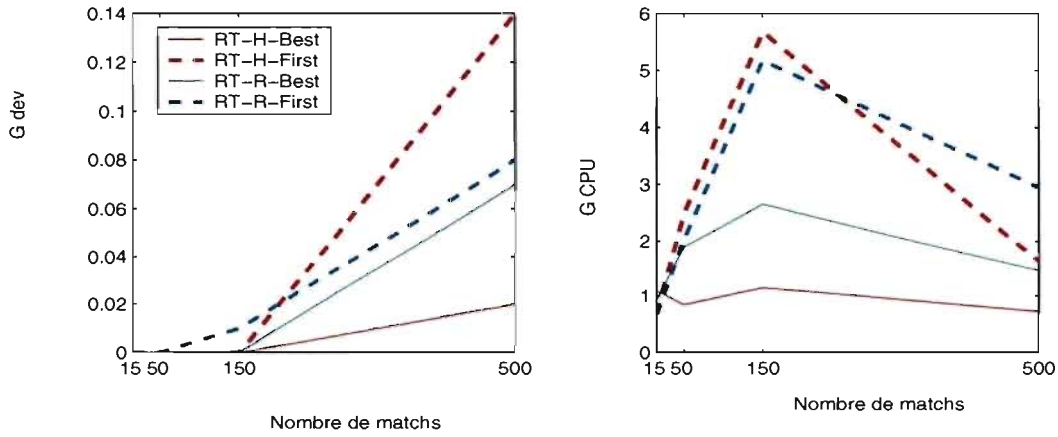

Figure 3.14 – Impact du mécanisme de mémoire à long terme sur les performances des variantes de la RT

probant pour la variante **RT-H-Best**. En particulier, pour les instances avec 500 matchs, l'amélioration de la qualité des solutions se fait au détriment du temps de résolution.

3.5 Adaptation de la méthode de recherche à voisinage variable

Nous proposons maintenant une adaptation de la métaheuristique de recherche à voisinage variable. Nous commençons par un bref rappel des principes de cette méthode. Nous présentons ensuite la procédure que nous proposons pour résoudre notre problème et détaillons ses différentes composantes. Nous terminons cette section par les résultats des expérimentations numériques. La terminologie et la notation adoptées sont similaires à celles employées dans la section 3.4.

3.5.1 Présentation de la méthode

La méthode de recherche à voisinage variable (RVV) a été développée par MLADENOVIC et HANSEN [81] dans les années 90. L'idée de base de cette méthode consiste à changer systématiquement de voisinage durant la recherche afin de s'échapper des vallées qui contiennent les optima locaux. Ainsi, tant qu'il n'y a pas eu amélioration de la valeur de la fonction objectif, la méthode explore séquentiellement les voisinages

proches, puis de plus en plus éloignés de la meilleure solution connue. Aussitôt qu'une meilleure solution est obtenue, la recherche est relancée dans le premier voisinage (le plus proche de la meilleure solution connue). Le schéma de base de la méthode RVV est résumé dans la figure 3.15.

Initialisation
Choisir une solution initiale et initialiser la meilleure solution *xbest* avec cette solution
$V_k\ (k = 1, \ldots, k_{max})$ un ensemble fini de structures de voisinages
Algorithme
Tant que (le critère d'arrêt n'est pas satisfait) **faire**
  $k := 1$
  Tant que $(k \leq k_{max})$ **faire**
    PERTURBATION
    Choisir aléatoirement une solution $x^0 \in V_k(xbest)$
    RECHERCHE LOCALE
    Appliquer une méthode de recherche locale à x^0. Soit x^* la meilleure solution obtenue
    MISE À JOUR
    Si (x^* est meilleure que *xbest*) **Alors**
      $xbest := x^*$
      $k := 1$
    Sinon
      $k := k + 1$
    Fin Si
  Fait
Fait

Figure 3.15 – Schéma de base de la méthode de recherche à voisinage variable

Dans la description que nous venons de faire, nous avons introduit les principes directeurs de la version de base de la méthode de recherche à voisinage variable. Nous allons maintenant les expliciter un peu plus :

Structures de voisinage : la plupart du temps, les structures de voisinage sont imbri-

quées ($V_k(x) \subseteq V_{k+1}(x)$) [81], ou encore induits par une certaine métrique sur l'ensemble des solutions [24, 82].

Perturbation : la solution x^0, qui sert de solution initiale pour la recherche locale, est choisie aléatoirement dans $V_k(xbest)$ afin d'éviter le cyclage pouvant découler de l'utilisation d'une règle déterministe [81]. Toutefois, il est tout à fait envisageable de la choisir de façon à ce qu'elle soit de bonne qualité. Ainsi, dans [81], les auteurs proposent de générer aléatoirement b (paramètre de la méthode) solutions dans $V_k(xbest)$ et de retenir la meilleure. Les stratégies utilisées dans [4, 24] tentent également d'obtenir des solutions de bonne qualité, mais elles sont spécifiques aux problèmes traités.

Recherche locale : la méthode a recours à une recherche locale pour améliorer la solution x^0. N'importe quelle méthode de recherche locale peut être utilisée [54]. Dans [82], les auteurs utilisent une technique de descente tandis que dans [4], une méthode de recherche avec tabous est employée. En général, la recherche locale utilise toujours la même structure de voisinage [81]. Le voisinage utilisé dans [4] n'appartient pas à l'ensemble V_k $(k = 1, \ldots, k_{max})$ alors que dans [24], c'est le voisinage V_1 qui est utilisé.

Critère d'arrêt : les critères d'arrêt classiques sont utilisés comme par exemple, un temps CPU maximal alloué [82], un nombre maximal d'itérations [24], ou encore un nombre maximal d'itérations sans amélioration [4].

Différentes extensions de la méthode sont proposées dans la littérature. Nous allons brièvement aborder quelques unes ; l'état de l'art, les références, et les applications sont inclus dans [55, 56] :

Descente à voisinage variable : à chaque itération, la solution x^* est identifiée à l'aide d'une technique de descente utilisant la structure de voisinage V_k ($x^* \in V_k(xbest)$), ce qui permet d'accélérer la résolution.

Recherche à voisinage variable biaisée : elle consiste à accepter une solution non améliorante x^* si elle est suffisamment distante de $xbest$; le but étant d'atteindre des vallées éloignées de la meilleure solution.

Recherche et décomposition à voisinage variable : elle consiste à fixer le sous-espace dans lequel la procédure de recherche locale va évoluer. Elle est particulièrement utilisée pour résoudre les instances de grande taille pour lesquels les temps de résolution peuvent devenir prohibitifs.

3.5.2 Approche de résolution

Nous avons choisi d'utiliser la RVV de base étant donné qu'elle s'est montrée efficace et robuste pour la résolution d'un grand nombre de problèmes d'optimisation combinatoire tel que rapporté dans [54]. De plus, elle est plus simple à mettre en œuvre et possède peu de paramètres.

L'espace des solutions est défini par l'ensemble des contraintes du modèle M^2 présenté au paragraphe 3.4.2. Pour construire les différentes structures de voisinage, nous considérons la métrique d suivante où, la distance entre deux solutions x et $\overline{x}$ est définie en termes du nombre d'affectations distinctes :

$$d(x,\overline{x}) = \frac{1}{4}\sum_{i}\sum_{\mu}|x_{i\mu} - \overline{x}_{i\mu}|.$$

Ainsi, si $d(x,\overline{x}) = k$, alors ceci signifie que $\overline{x}$ est obtenue à partir de x en lui appliquant k échanges. Pour caractériser les éléments de $V_k(xbest)$ à l'aide de la distance d, nous dirons que :

$$x \in V_k(xbest) \text{ si } d(x,xbest) = k$$

c'est à dire que x est obtenue à partir de *xbest* en lui appliquant k échanges.

Tel qu'illustré à la figure 3.15, la procédure de résolution comprend deux étapes. Elle est initialisée avec une solution réalisable (i.e., qui vérifie les contraintes du modèle M^2) générée soit aléatoirement, soit par l'entremise de la technique heuristique **HLA-HOA** présentée à la section 3.3. k est initialisé avec la valeur 1. Une solution $x^0 \in V_k(xbest)$ est d'abord générée en utilisant l'information collectée lors des étapes de perturbation précédentes par le biais d'une mesure de fréquence d'utilisation des matchs tel qu'il sera décrit dans le paragraphe 3.5.3. Nous lui appliquons ensuite une méthode de recherche

avec tabous similaire à celle décrite au paragraphe 3.4.4 (i.e., utilisant la structure de voisinage V_1, et la mesure de fréquence d'utilisation des mouvements P pour départager les solutions voisines candidates induisant la même modification dans la fonction objectif). Si une meilleure solution est obtenue, alors nous nous déplaçons dans son voisinage en lui appliquant un seul échange (i.e., $k := 1$). Sinon, nous augmentons la distance entre *xbest* et la nouvelle solution initiale x^0 (i.e., $k := k + 1$). Lorsque k atteint k_{max}, nous retournons au premier voisinage (i.e., $k := 1$). Le processus est réitéré aussi longtemps que le temps CPU est inférieur au temps maximal alloué *tempsmax*. Bien entendu, la procédure s'arrête également dès qu'une solution optimale est générée.

3.5.3 Stratégie de perturbation

Rappelons que l'objectif à cette étape est de générer une solution $x^0 \in V_k(xbest)$ pour amorcer la recherche locale. À cette fin, nous utilisons la procédure séquentielle résumée dans la figure 3.16.

La procédure comprend k itérations. À chaque itération, un échange *intéressant* (i', j', r', l') (*intéressant* dans le sens où i' et r' ne sont pas tous les deux des juges fictifs) est sélectionné, puis appliqué à x^0 (i.e., $x^0 := x^0 \oplus (i', j', r', l')$) ; x^0 étant initialisée avec *xbest*. Afin de ne pas retourner (lors de la recherche locale) à la solution *xbest* à partir de laquelle la solution x^0 a été générée, la liste tabou est mise à jour après la sélection de chaque échange. La matrice des fréquences *Freq* (cf. paragraphe 3.4.4) est également mise à jour.

Pour sélectionner les échanges (i', j', r', l'), nous considérons la fréquence d'utilisation des matchs au cours des étapes de perturbation précédentes. Pour ce faire, nous utilisons un vecteur $Freq_1(.)$ et une matrice $Freq_2(.,.)$ pour enregistrer respectivement le nombre de fois qu'un match et une paire de matchs ont été sélectionnés dans le passé pour générer la nouvelle solution initiale. Ainsi, chaque fois que (i, j, r, l) est utilisé pour générer x^0, nous incrémentons d'une unité les valeurs de $Freq_1(j)$, $Freq_1(l)$ et $Freq_2(j, l)$. Pour définir l'échange (i', j', r', l') qui sera utilisé à l'étape courante, nous

Initialisation
$xbest$, la meilleure solution rencontrée jusqu'à présent
$x^0 := xbest$, la solution initiale à générer
k, l'indice du voisinage
α, la probabilité de sélection
$LT := \emptyset$, la liste tabou
$Freq$, la matrice des fréquences
$Freq_1$ et $Freq_2$, les compteurs d'utilisation des matchs
Algorithme
Choisir aléatoirement $p \in [0,1]$
Pour s **de** 1 **à** k **faire**
  SÉLECTION D'UNE PAIRE DE MATCHS
  $\overline{J} := \{(\bar{j},\bar{l}) \quad : \quad U(\bar{j},\bar{l}) = \min_{j \neq l} U(j,l)\}$
  Choisir aléatoirement $(j',l') \in \overline{J}$
  SÉLECTION D'UNE PAIRE DE JUGES
  $E := \emptyset$
  Pour chaque échange *intéressant* (i,j',r,l') **faire**
    $E := E \cup (i,j',r,l')$
  Fin Pour
  Si $(p \leq \alpha)$ **Alors**
    Choisir aléatoirement $(i',j',r',l') \in E$
  Sinon
    $Z := \{x^0 \oplus (i,j',r,l') \quad : \quad (i,j',r,l') \in E\}$
    $E' := \{(i,j',r,l') \quad : \quad f\left(x^0 \oplus (i,j',r,l')\right) = \min_{z \in Z} f(z)\}$
    Choisir aléatoirement $(i',j',r',l') \in E'$
  Fin Si
  MISE À JOUR
  $x^0 := x^0 \oplus (i',j',r',l')$
  Mettre à jour LT, $Freq$, $Freq_1$ et $Freq_2$
Fin Pour
x^0 est la nouvelle solution initiale générée pour amorcer la recherche locale

Figure 3.16 – Stratégie de perturbation pour la RVV

choisissons d'abord la paire de matchs (j',l') qui minimise la mesure suivante :

$$U(j,l) = \sqrt{Freq_1(j)} + \sqrt{Freq_1(l)} + Freq_2(j,l).$$

Notons que le critère basé sur l'expression $U(j,l)$ ci-dessus permet de sélectionner les paires de matchs selon deux critères considérés de façon hiérarchique. Pour illustrer ce propos, considérons deux paires de matchs candidates (j,l) et (j',l') telles que $Freq_1(j) + Freq_1(l) = Freq_1(j') + Freq_1(l')$. Alors, le critère utilisé indique de sélectionner (j,l) si $Freq_2(j,l) < Freq_2(j',l')$. Maintenant, si $Freq_2(j,l) = Freq_2(j',l')$, alors ce critère indique de sélectionner (j,l) si $\min(Freq_1(j), Freq_1(l)) \leq \min(Freq_1(j'), Freq_1(l'))$.

Par la suite, pour choisir la paire de juges (i',r'), nous considérons l'ensemble des paires de juges (i,r) telles que i et r sont affectés à j' et l' respectivement, et $i \leq N$ ou $r \leq N$ (i.e., l'échange (i,j',r,l') est *intéressant*). Parmi les paires de juges de cet ensemble, nous sélectionnons la paire (i',r') soit aléatoirement, soit parmi celles qui conduisent à la solution ayant la meilleure valeur de la fonction objectif. Nous associons à la première stratégie une probabilité de sélection α. La seconde stratégie est sélectionnée avec une probabilité $(1-\alpha)$.

Notons que pour chacune des sélections précédentes, en cas d'égalité, les choix se font aléatoirement. D'autre part, la première stratégie utilisée pour choisir les juges (choix aléatoire) correspond à une diversification de la recherche. La seconde stratégie (choix glouton) correspond davantage à une intensification de la recherche. De plus, les critères de sélection des matchs constituent également une stratégie de diversification : en favorisant les matchs de faible fréquence (qui minimisent la mesure U), nous favorisons par extension les mouvements rarement choisis, et nous augmentons les chances de générer des solutions n'ayant que peu de caractéristiques en commun avec les solutions générées antérieurement.

3.5.4 Expérimentations numériques

Nous dénotons par **RVV-H-Best**, **RVV-H-First**, **RVV-R-Best**, et **RVV-R-First** les quatre variantes implémentées. Rappelons que **H** et **R** désignent le processus utilisé pour générer la solution initiale (**HLA-HOA** et aléatoirement respectivement) alors que **Best** et **First** représentent la stratégie utilisée, lors de la phase de recherche locale, pour sélectionner la solution dans le voisinage de la solution courante (*meilleure amélioration* et *première amélioration* respectivement).

Le paramètre k_{max} est fixé à $\lfloor \frac{M+1}{2} \rfloor$ (M étant le nombre de matchs) dans le but de compléter une exploration de l'ensemble des solutions qui soit proportionnelle à la taille de l'instance. Le paramètre α (utilisé lors de la phase de perturbation pour choisir la stratégie de sélection des juges) est fixé à $\frac{2}{3}$ afin de favoriser les choix aléatoires. En effet, nous avons noté au cours des tests préliminaires que le fait de choisir une paire de juges quelconque, non nécessairement la meilleure, permettait souvent d'éviter de rester piégé dans un optimum local. Les autres paramètres de la procédure sont fixés comme pour la procédure de recherche avec tabous : la valeur de la constante C (utilisée dans le modèle M^2 pour pénaliser la violation des règles incontournables d'affectation) est fixée à $50M^2$. La durée tabou est choisie aléatoirement dans l'intervalle $[t_{min}, t_{max}] = [\lfloor 0,8N \rfloor, \lceil 1,2N \rceil]$ (N étant le nombre de juges). Le nombre maximal d'itérations successives sans amélioration *nitermax* est fixé à N, et un temps maximal $tempsmax = 3N$ secondes est alloué à chaque variante. Le choix de ces valeurs est motivé par deux raisons. D'une part, elles ont permis d'obtenir de bons résultats (si l'on se réfère aux résultats numériques présentés au paragraphe 3.4.6), et d'autre part, elles vont nous permettre de comparer objectivement les deux procédures **RT** et **RVV**.

Nous reprenons l'ensemble des instances de test décrites au paragraphe 3.2.1 pour évaluer et comparer les quatre variantes. Chaque instance est résolue 10 fois avec une valeur d'initialisation différente pour le générateur des nombres aléatoires. Notons que les mêmes valeurs sont utilisées pour chaque variante. De plus, ces valeurs sont identiques à celles utilisées pour la procédure **RT**.

La comparaison se fait à l'aide des quatre critères utilisés dans le paragraphe 3.4.6

à savoir, l'écart moyen entre les valeurs des solutions générées et la valeur optimale (lorsque nous disposons de cette information) ou la borne inférieure sur cette valeur (*Avedev*), le pourcentage des solutions optimales trouvées (*%Opt*), le temps moyen de résolution en secondes (*AveCPU*), et l'écart-type des temps de résolution (σ_{CPU}). De plus, étant donné que certaines solutions générées ne sont pas réalisables, dans le sens où elles ne vérifient pas les règles incontournables d'affectation (contraintes du modèle M^1), nous considérons également le pourcentage des solutions réalisables obtenues (*%Feas*). Les résultats obtenus pour chaque sous-ensemble de problèmes sont présentés dans le tableau 3.6. Nous indiquons également, dans la dernière colonne de ce tableau, les résultats obtenus avec CPLEX pour les problèmes qui ont pu être résolus dans une limite de temps de 10 heures. La figure 3.17 illustre l'évolution de l'écart moyen (*Avedev*) et du temps moyen de résolution (*AveCPU*) en fonction de la taille des problèmes.

Si l'on effectue une comparaison entre CPLEX et les quatre variantes, le temps de résolution de celles-ci est uniformément inférieur à celui de CPLEX (cf. deuxième graphique de la figure 3.17), indiquant encore une fois qu'il est plus avantageux d'utiliser l'approche métaheuristique. En particulier, si l'on considère les instances avec 15 matchs qui ont toutes été résolues à l'optimalité avec CPLEX (cf. tableau 3.6), le temps de résolution est réduit en moyenne d'un facteur de 277, 247, 146, et 134 respectivement pour les variantes **RVV-H-First**, **RVV-H-Best**, **RVV-R-Best**, et **RVV-R-First**. De plus, toutes les solutions générées par ces quatre variantes sont optimales.

Solution initiale générée aléatoirement versus solution initiale de bonne qualité. Les résultats des problèmes avec 15, 50 et 150 matchs n'indiquent pas d'impact de la qualité de la solution initiale sur la qualité des solutions générées. Pour ceux avec 500 matchs, les résultats diffèrent selon les ensembles de problèmes. En particulier, pour les instances de l'ensemble P_2, les deux variantes **RVV-R-Best** et **RVV-R-First** dominent **RVV-H-Best** et **RVV-H-First** tant au niveau du pourcentage des solutions optimales obtenues (amélioré en moyenne de 15,50%) qu'au niveau de l'écart moyen (réduit en moyenne d'un facteur de 22). Le contraire se produit pour les instances des ensembles P_1 et P_3 : toutes les solutions générées par les deux variantes **RVV-H-Best** et **RVV-H-First**

		Taille	RVV-H-Best	RVV-R-Best	RVV-H-First	RVV-R-First	CPLEX
Ave dev	P_1	15	0	0	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0	4 623 150	0	0	NA
	P_2	15	0	0	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0,24	0,02	0,2	0	NA
	P_3	15	0	0	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0,02	1 874 250,10	0,01	0	NA
%Feas	P_1	15	100	100	100	100	100
		50	100	100	100	100	100
		150	100	100	100	100	100
		500	100	83	100	100	100
	P_2	15	100	100	100	100	100
		50	100	100	100	100	100
		150	100	100	100	100	100
		500	100	100	100	100	100
	P_3	15	100	100	100	100	100
		50	100	100	100	100	100
		150	100	100	100	100	100
		500	100	89	100	100	100
%Opt	P_1	15	100	100	100	100	100
		50	100	100	100	100	60
		150	100	100	100	100	50
		500	100	83	100	100	50
	P_2	15	100	100	100	100	100
		50	100	100	100	100	100
		150	100	100	100	100	100
		500	81	99	87	100	100
	P_3	15	100	100	100	100	100
		50	100	100	100	100	60
		150	100	100	100	100	50
		500	98	89	99	100	40
Ave CPU (sec)	P_1	15	0,05	0,07	0,04	0,08	28,74
		50	0,17	1,61	0,17	0,43	2,31
		150	0,47	21,83	0,45	4,71	96,38
		500	12,22	1 250,24	11,51	106,57	12 159,96
	P_2	15	0,09	0,15	0,08	0,15	0,69
		50	1,11	1,10	1,12	1,01	1,93
		150	42,62	16,41	38,94	16,19	32,41
		500	3 409,06	992,85	2 999,83	672,84	21 299,74
	P_3	15	0,04	0,08	0,03	0,08	13,12
		50	0,20	0,86	0,18	0,48	8,85
		150	1,86	36,29	2,48	19,03	276,17
		500	1 013,25	1 633,40	658,60	836,32	23 824,47
σ_{CPU}	P_1	15	0,06	0,11	0,05	0,10	33,59
		50	0,20	5,57	0,21	0,45	0,58
		150	0,34	66,92	0,32	8,73	71,55
		500	3,29	2 416,11	2,15	392,53	5 408,41
	P_2	15	0,08	0,20	0,09	0,18	0,09
		50	1,59	1,17	1,62	0,99	0,25
		150	51,19	17,00	40,14	15,57	8,14
		500	3 231,52	1 437,54	2 977,96	1 177,93	4 130,39
	P_3	15	0,07	0,11	0,05	0,10	16,75
		50	0,27	1,61	0,21	0,37	12,51
		150	6,05	71,70	13,21	19,47	201,12
		500	2 052,83	2 008,09	1 532,68	894,04	4 216,50

Tableau 3.6 – Comparaison de l'efficacité des variantes de la RVV en fonction des sous-ensembles de problèmes

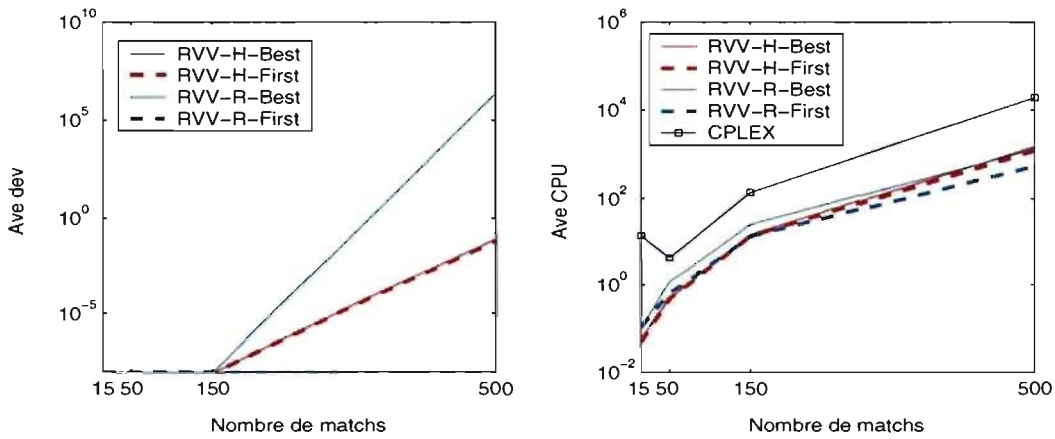

Figure 3.17 – Comparaison de l'efficacité des variantes de la RVV en fonction de la taille des problèmes

sont réalisables et pratiquement optimales (99,50%) alors que la variante **RVV-R-Best** n'a pas été en mesure d'obtenir des solutions réalisables dans 14% des cas.

Rappelons que la différence entre les instances des ensembles P_2 d'une part, et P_1 et P_3 d'autre part, se situe au niveau du nombre de juges pouvant être affectés à chaque match : pour les instances dans P_2, il existe une solution où tous les matchs comptent 5 juges affectés ($N \geq 5M$) alors que pour celles dans P_1 et P_3, certains matchs comptent 3 juges affectés seulement ($N < 5M$). Il semble donc que, pour la variante **RVV-R-Best**, la difficulté de résolution est liée à la satisfaction de la contrainte stipulant qu'exactement 3 ou 5 juges doivent être affectés à chaque match (lorsque le nombre de juges est insuffisant pour affecter 5 juges à chaque match). Les bonnes performances de **RVV-H-Best** s'expliqueraient alors par le fait que toutes les solutions générées par la technique **HLA-HOA** satisfont cette contrainte (cf. résultats présentés au paragraphe 3.3.3).

Pour confirmer le bien fondé de notre propos, nous nous sommes intéressés aux solutions non réalisables obtenues pour identifier les contraintes incontournables violées. Nous avons constaté qu'en effet, c'était la seule contrainte non satisfaite (certains matchs comptaient un seul juge affecté). Le problème ne se pose pas dans le cas des instances dans P_2, car le nombre de juges disponibles est suffisant pour affecter 5 juges à chaque

match. Ceci explique les bonnes performances de la variante **RVV-R-Best** pour cette catégorie de problèmes.

Nous retrouvons les mêmes constatations si l'on s'intéresse au temps de résolution. Les deux variantes **RVV-H-Best** et **RVV-H-First** dominent pour les instances des ensembles P_1 et P_3 alors que le contraire se produit pour celles de l'ensemble P_2. Comme nous l'avons souligné précédemment au paragraphe 3.4.6, il existe une corrélation entre la qualité des solutions générées et le temps de résolution. Plus le pourcentage des solutions optimales est élevé, plus le temps moyen de résolution est réduit.

Stratégie *meilleure amélioration* versus stratégie *première amélioration*. Il semble que la différence entre les deux stratégies est liée à la qualité de la solution initiale. On constate que la stratégie *première amélioration* se démarque davantage lorsque la solution initiale est générée aléatoirement. En particulier, pour les instances des sous-ensembles P_1 et P_3 avec 500 matchs, toutes les solutions générées sont réalisables. Le pourcentage des solutions optimales obtenues est amélioré de 17% et de 11% respectivement. Le temps de résolution est, pour sa part, réduit en moyenne d'un facteur de 12 et 2 respectivement. Lorsque la solution initiale est générée avec la technique **HLA-HOA**, la stratégie *première amélioration* donne également de meilleurs résultats que la stratégie *meilleure amélioration*, mais sa supériorité est beaucoup moins significative comme on peut le constater sur la figure 3.17. Cela peut s'expliquer par le fait que la technique **HLA-HOA** génère à la base des solutions de très bonne qualité qu'il est difficile d'améliorer. Souvent, au cours de la recherche locale, bien que nous utilisions la stratégie *première amélioration*, tout le voisinage est énuméré sans pouvoir identifier une solution voisine améliorant la solution courante. Dès lors, les deux stratégies *première amélioration* et *meilleure amélioration* sont équivalentes et obtiennent les mêmes résultats.

Finalement, si l'on s'intéresse à l'écart-type, on note que la stratégie *première amélioration* est plus stable que la stratégie *meilleure amélioration*.

Comparaison globale. Considérons la qualité des solutions générées par les différentes variantes. La variante **RVV-R-First** affiche d'excellentes performances puisque toutes les solutions générées par cette variante sont optimales alors que la variante **RVV-R-Best** est celle qui produit les plus mauvaises solutions. Si l'on considère les temps de résolution, la variante **RVV-H-Best** se révèle la plus lente alors que **RVV-R-First** est la plus rapide.

Ces conclusions sont basées sur les valeurs moyennes (*Ave dev* et *Ave CPU*). Pour poursuivre la comparaison entre les quatre variantes, nous avons effectué le test de WILCOXON pour échantillons appariés sur les résultats de l'ensemble des résolutions. Le seuil de confiance utilisé est égal à 0,05. Les résultats de ce test sont résumés dans le tableau 3.7. Celui-ci possède la même structure que le tableau 3.5 présenté au paragraphe 3.4.6 (cf. page 86). Pour chaque couple de variantes (A,B), nous indiquons la *p_valeur* obtenue. Nous soulignons cette valeur en caractère gras lorsque la différence entre A et B n'est pas statistiquement significative (i.e., si la *p_valeur* obtenue est supérieure à 0,05). Une *p_valeur* inférieure à 0,05 révèle que la variante A (dont le nom figure à la colonne 1 du tableau) est statistiquement meilleure que la variante B (dont le nom figure à la ligne 2 du tableau).

	Qualité de la solution			Temps de résolution		
	RVV-H-First	**RVV-H-Best**	**RVV-R-Best**	**RVV-H-First**	**RVV-H-Best**	**RVV-R-Best**
RVV-R-First	6,50E-04	1,52E-05	2,44E-06	<2,2E-16	< 2,2E-16	< 2,2E-16
RVV-H-First	-	**0,38**	1,05E-05	-	<2,2E-16	<2,2E-16
RVV-H-Best	-	-	8,65E-05	-	-	<2,2E-16

Tableau 3.7 – Résultats du test de WILCOXON pour les variantes de la RVV

Les résultats du test indiquent qu'au niveau de la qualité des solutions, il n'y a pas de différence statistiquement significative entre les deux variantes **RVV-H-Best** et **RVV-H-First**. Cependant, si l'on tient compte du temps de résolution, la comparaison globale indique que **RVV-H-First** domine. La variante **RVV-R-First** se classe en premier et **RVV-R-Best** en dernier, et ce aussi bien au niveau de la qualité des solutions générées qu'au niveau du temps de résolution.

Comme nous l'avons déjà mentionné, les instances avec 15, 50 et 150 matchs ne

semblent pas poser des problèmes à aucune des quatre variantes. Pour illuster le comportement des différentes variantes lors de la résolution des problèmes avec 500 matchs, nous présentons dans la figure 3.18 l'évolution de *Ave dev* en fonction du temps.

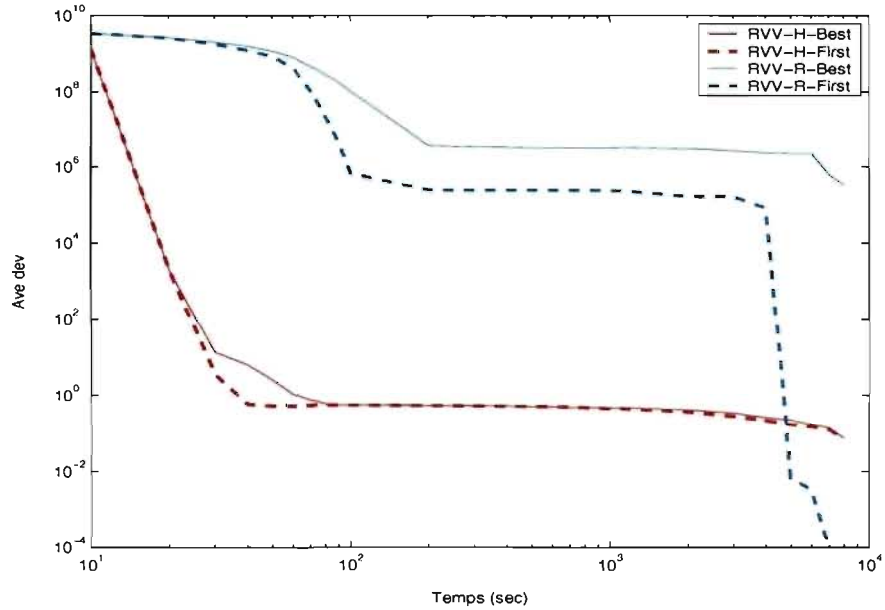

Figure 3.18 – Évolution de *Ave dev* des variantes de la RVV pour les problèmes avec 500 matchs

Les courbes associées aux variantes **RVV-H-Best** et **RVV-H-First** ont la même allure que celles associées aux variantes **RT-H-Best** et **RT-H-First** présentées à la figure 3.13. Il convient de souligner que les deux procédures **RT** et **RVV** sont très similaires. En effet, si l'on assimile la phase de perturbation de la **RVV** à une diversification, on peut considérer que les deux procédures ne diffèrent que sur la façon de générer la nouvelle solution initiale pour réinitialiser la recherche locale. Par conséquent, pour justifier les différences entre les deux variantes **RVV-H-Best** et **RVV-H-First**, les mêmes raisons évoquées au paragraphe 3.4.6 (cf. page 87) sont valables ici. D'autre part, il semble que lorsque la solution initiale est générée avec **HLA-HOA**, ni la stratégie de diversification de la **RT** (présentée au paragraphe 3.4.5) ni celle de perturbation de la **RVV** (décrite au paragraphe 3.5.3) ne permet de s'échapper des vallées qui contiennent les optima lo-

caux. Notons toutefois, qu'en termes de *Ave dev*, les résultats obtenus avec la stratégie de diversification de la **RT** sont légèrement meilleurs que ceux obtenus avec la stratégie de perturbation de la **RVV**.

La courbe associée à la variante **RVV-R-Best** se situe nettement en retrait par rapport aux courbes associées aux trois autres variantes. Durant les 200 premières secondes, la solution est améliorée, mais, par la suite, on note une longue période de stagnation. Ce n'est que vers la fin du processus de résolution qu'une légère amélioration est enregistrée. Malgré de multiples applications de la stratégie de perturbation, la variante ne parvient pas à obtenir des solutions réalisables pour certaines instances.

Finalement, la courbe associée à la variante **RVV-R-First** stagne aussi à partir de la $200^{\text{ème}}$ seconde à une valeur de *Ave dev*, certes moins élevée que **RVV-R-Best**, mais plus élevée que les deux autres variantes. Toutefois, vers la fin du processus de résolution, **RVV-R-First** surpasse toutes les autres variantes puisqu'elle est capable de réduire la valeur de *Ave dev* à 0.

Impact de la stratégie de perturbation sur les performances des quatre variantes. Comme nous l'avons mentionné au début de cette section, dans la version de base de la **RVV**, la solution x^0 utilisée pour initialiser la recherche locale est choisie aléatoirement dans $V_k(xbest)$. Nous avons procédé différemment en introduisant une mesure de fréquence d'utilisation des matchs (U) pour favoriser le déplacement vers les régions les moins explorées de l'ensemble des solutions.

Pour évaluer l'impact de la stratégie que nous avons adoptée, nous avons effectué un comparatif des performances des deux versions (la version que nous proposons versus la version de base). La démarche utilisée est similaire à celle décrite au paragraphe 3.4.6 (cf. page 89). Nous avons implémenté quatre variantes de la version de base (en combinant les deux processus **H** et **R**, et les deux stratégies **Best** et **First**). Nous avons testé ces quatre variantes avec l'ensemble des instances, et chaque instance a été résolue 10 fois. Les deux versions utilisent la même méthode de recherche locale, les mêmes valeurs des paramètres, et les mêmes valeurs d'initialisation du générateur des nombres aléatoires. Finalement, pour chaque sous-ensemble de problèmes et

chaque variante, nous avons calculé le gain moyen en termes de la qualité des solutions ($Gdev = (Avedev)' - Avedev$), et en termes du temps de résolution ($GCPU = \frac{(AveCPU)'}{AveCPU}$). $(Avedev)'$ et $(AveCPU)'$ désignent respectivement l'écart moyen entre les valeurs des solutions générées et la valeur optimale (ou la borne inférieure sur cette valeur), et le temps moyen de résolution (en secondes) lorsque la version de base de la **RVV** est utilisée. Nous reportons dans la figure 3.19 les résultats obtenus en fonction de la taille des problèmes.

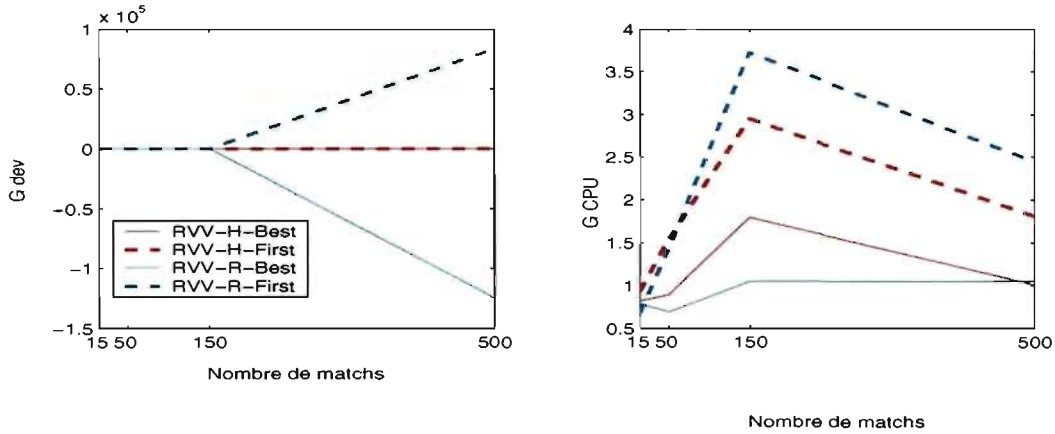

Figure 3.19 – Impact du mécanisme de mémoire à long terme sur les performances des variantes de la RVV

Il semble que la stratégie de perturbation que nous proposons est plus bénéfique lorsque la stratégie de sélection utilisée est la stratégie *première amélioration*. Ainsi, pour la variante **RVV-R-First**, et en particulier pour les gros problèmes avec 500 matchs, on constate une nette amélioration de la qualité des solutions générées. En effet, avec la version de base, la variante ne parvient pas toujours à trouver des solutions réalisables alors qu'avec la stratégie proposée toutes les solutions générées sont optimales. Cette amélioration est accompagnée d'une réduction du temps de résolution. Pour les problèmes avec 50 et 150 matchs, les solutions générées ne sont certes pas meilleures, mais elles sont obtenues plus rapidement. L'augmentation du temps de résolution, dans le cas des problèmes avec 15 matchs, peut s'expliquer par l'effort de calcul supplémen-

taire pour choisir la nouvelle solution x^0 alors que ces problèmes sont faciles à résoudre. Pour la variante **RVV-H-First**, les résultats indiquent que la stratégie de perturbation n'a aucun impact sur la qualité des solutions, mais que le gain au niveau du temps de résolution est probant. À l'opposé, la stratégie proposée semble influencer négativement la variante **RVV-R-Best** laquelle voit ses performances se dégrader en général. Quant à **RVV-H-Best**, on observe une légère amélioration des performances de cette variante (en termes du temps de résolution). Cette observation n'est cependant pas générale comme l'illustrent les résultats obtenus pour les problèmes avec 15 et 50 matchs, pour lesquels la version de base de la **RVV** prend l'avantage.

3.6 Recherche avec tabous à voisinages structurés

Les deux procédures de résolution présentées dans les sections 3.4 et 3.5 reposent sur le modèle M^2 dans lequel toutes les contraintes du modèle M^1 (règles incontournables d'affectation) sont pénalisées, et la somme pondérée de leur violation est minimisée. Dès lors, les solutions voisines générées ne sont pas nécessairement réalisables. La procédure présentée dans cette section (RTVS) est sensiblement différente dans la mesure où l'exploration de l'ensemble des solutions tient davantage compte des règles incontournables d'affectation.

Dans le paragraphe 3.6.1, nous introduisons une version modifiée du modèle mathématique original M^1 décrit au paragraphe 3.1.1, puis présentons la procédure que nous proposons pour résoudre cette nouvelle formulation du problème. Les paragraphes qui suivent décrivent les différentes étapes de cette procédure. Le paragraphe 3.6.6 est consacré aux résultats des expérimentations numériques.

3.6.1 Modèle modifié et approche de résolution

Dans le modèle modifié, la contrainte stipulant qu'exactement 3 ou 5 juges doivent être affectés à chaque match est relaxée. Ainsi, les matchs avec un seul juge affecté sont réalisables, mais engendrent un coût dont la pondération est très élevée ($C \gg 5M$). De plus, le surplus de juges (s'il y en a) est affecté à un match fictif ($M+1$) auquel tous les

juges sont admissibles et qui ne nécessite pas qu'un *juge en chef* lui soit affecté. Pour formuler ce nouveau modèle, nous utilisons la notation introduite au paragraphe 3.1.1. De plus, pour tout $j = 1,\ldots,M$, une variable binaire y_j^1 est introduite pour indiquer si un seul juge est affecté au match j :

$$y_j^1 = \begin{cases} 1 & \text{si un seul juge est affecté au match } j \\ 0 & \text{sinon.} \end{cases}$$

Le nouveau modèle mathématique M^3 est résumé dans la figure 3.20.

$$\min f(x) = \sum_{j=1}^{M} \sum_{k=1}^{K} \max \left\{ \left(\sum_{i=1}^{N} e_{ik} x_{ij} \right) - 1, 0 \right\} + 5M \sum_{j=1}^{M} y_j^3 + C \sum_{j=1}^{M} y_j^1 \tag{3.16}$$

(M^3) **Sujet à**

$$\sum_{j=1}^{M+1} x_{ij} = 1 \qquad i = 1,\ldots,N \tag{3.17}$$

$$\sum_{j=1}^{M} (1 - a_{ij}) x_{ij} = 0 \qquad i = 1,\ldots,N \tag{3.18}$$

$$\sum_{i=1}^{N} l_i x_{ij} \geq 1 \qquad j = 1,\ldots,M \tag{3.19}$$

$$\sum_{i=1}^{N} x_{ij} = y_j^1 + 3y_j^3 + 5y_j^5 \qquad j = 1,\ldots,M \tag{3.20}$$

$$y_j^1 + y_j^3 + y_j^5 = 1 \qquad j = 1,\ldots,M \tag{3.21}$$

$$x_{ij} = 0 \text{ ou } 1 \qquad i = 1,\ldots,N,\ j = 1,\ldots,M \tag{3.22}$$

$$y_j^1, y_j^3, y_j^5 = 0 \text{ ou } 1 \qquad j = 1,\ldots,M. \tag{3.23}$$

Figure 3.20 – Modèle M^3 (version modifiée du modèle original M^1)

Avant de présenter la procédure que nous proposons pour résoudre cette nouvelle formulation du problème, introduisons d'abord quelques notations qui nous seront utiles dans la suite de cette section.

Pour chaque solution réalisable x du modèle M^3, nous dénotons par :

- $I(j,x)$, l'ensemble des juges i affectés au match j dans la solution x ;
- $M_1(x)$, l'ensemble des matchs $j \neq M+1$ avec un seul juge affecté (*juge en chef*) ;
- $M_3(x)$, l'ensemble des matchs $j \neq M+1$ avec 3 juges affectés ;
- $M_5(x)$, l'ensemble des matchs $j \neq M+1$ avec 5 juges affectés.

Par ailleurs, pour simplifier la notation, nous dénotons par $d_{kj}(x) = \sum_{i=1}^{N} e_{ik}x_{ij}$ le nombre de juges possédant le domaine d'expertise k, et affectés au match j dans la solution x.

La procédure de résolution est une métaheuristique comprenant trois principales étapes. Elle est initialisée avec une solution réalisable du modèle M^3. Dans un premier temps, nous utilisons une recherche avec tabous à voisinages structurés pour améliorer la qualité de la solution en réduisant le nombre de matchs avec 1 et 3 juges affectés (i.e., pour optimiser les deux derniers termes de la fonction objectif (3.16)). Par la suite, une seconde recherche avec tabous est utilisée pour améliorer la diversité des domaines d'expertise des juges affectés à un même match (i.e., pour optimiser le premier terme de la fonction objectif (3.16)). Finalement, une stratégie de diversification est utilisée pour générer une nouvelle solution initiale afin de réinitialiser la procédure. La procédure s'arrête lorsqu'une solution optimale est générée ou lorsque le temps de résolution maximal *tempsmax* s'est écoulé. Notons qu'une solution est optimale si 5 juges sont affectés à chaque match, ou si 3 ou 5 juges sont affectés à chaque match et $I(M+1,x) = 0$ ou 1 (i.e., il ne reste pas de juges non affectés ou il en reste un seul), et si la valeur de $\sum_{j=1}^{M}\sum_{k=1}^{K} \max\{d_{kj}(x)-1,0\}$ est égale à 0 ou à une borne inférieure b_{inf} connue *a priori* pour le problème.

3.6.2 Solution initiale

Nous considérons deux processus différents pour générer la solution initiale. Dans le premier que nous dénotons par **RS**, une solution initiale x^0 où chaque match compte un seul *juge en chef* affecté (i.e., $|M_1(x^0)| = M$ et $|M_3(x^0)| = |M_5(x^0)| = 0$) est générée aléatoirement. Le second processus est la technique heuristique **HLA-HOA** introduite à la section 3.3.

Affectation aléatoire (RS). Nous procédons en deux étapes. D'abord, nous identifions les *juges en chef* admissibles à chaque match j. Par la suite, nous considérons les matchs séquentiellement selon un processus probabiliste biaisé en faveur des matchs comptant le moins de *juges en chef* admissibles. Pour chaque match j, nous choisissons aléatoirement un *juge en chef* admissible disponible (i.e., non encore affecté), puis nous l'affectons à j.

Cependant, cette méthode d'affectation aléatoire peut nous amener vers des situations où il ne reste aucun *juge en chef* admissible pour être affecté à un match donné. Dans ce cas, nous reprenons l'affectation depuis le début en spécifiant un nouvel ordre des matchs où ce match est le premier traité. Le processus se poursuit ainsi jusqu'à ce que nous ayons un *juge en chef* i affecté à chaque match j. La procédure complète est résumée dans la figure 3.21.

Technique heuristique (HLA-HOA). Rappelons que cette technique combine deux procédures heuristiques constructives. La procédure **HLA** est utilisée pour affecter un *juge en chef* à chaque match. La procédure **HOA** comprend deux phases. Lors de la première phase, nous tentons d'affecter une paire de juges additionnelle à chaque match, et au cours de la seconde phase, une autre paire de juges est affectée au maximum de matchs. Ainsi, dans la solution produite par **HLA-HOA**, tous les matchs $j = 1,\ldots,M$ comptent 1, 3 ou 5 juges affectés.

Par ailleurs, il se peut que certains juges ne soient affectés à aucun match soit parce que la procédure **HOA** a échoué à affecter des paires additionnelles à certains matchs (car, à une itération donnée, les paires de juges encore disponibles ne sont admissibles à aucun match restant), ou bien parce qu'au cours de la seconde phase, il ne reste qu'un seul juge non affecté, ou encore parce qu'il y a un surplus de juges ($N > 5M$). Dans les trois cas, ces juges (non affectés) sont assignés au match fictif $(M+1)$, et la solution ainsi obtenue est utilisée comme solution initiale pour amorcer la procédure **RTVS**.

ÉTAPE 1
$To := 0$
Pour j **de** 1 **à** M **faire**
  $AL_j := \{i \quad : \quad l_i = 1 \text{ et } a_{ij} = 1\}$
  $To := To + |AL_j|$
Fin Pour
Pour j **de** 1 **à** M **faire**
  $\rho_j := To - |AL_j|$
Fin Pour
Permutation aléatoire proportionnelle des M matchs
Pour s **de** 1 **à** M **faire**
  Choisir un nombre aléatoire $\alpha \in \left]0, \sum_{1 \leq p \leq M} \rho_p\right]$
  Déterminer le plus petit indice j tel que $\sum_{1 \leq p \leq j} \rho_p \geq \alpha$
  $m_s := j, \rho_j := 0$
Fin Pour
ÉTAPE 2
$j := 0$
Tant que $(j \leq M)$ **faire**
  $j := j + 1$
  Si $(AL_{m_j} \neq \emptyset)$ **Alors**
    Choisir aléatoirement un juge $i \in AL_{m_j}$ et l'affecter au match m_j
    Pour s **de** $j+1$ **à** M **faire**
      Si $(i \in AL_{m_s})$ **Alors**
        $AL_{m_s} := AL_{m_s} - \{i\}$
      Fin Si
    Fin Pour
  Sinon
    $dummy := m_j$
    $m_1 := dummy$
    Pour s **de** 2 **à** j **faire**
      $m_s := m_{s-1}$
    Fin Pour
    Pour s **de** 1 **à** M **faire**
      Réinitialiser les affectations et les ensembles AL_s
    Fin Pour
    $j := 0$
  Fin Si
Fait

Figure 3.21 – Procédure RS

3.6.3 Recherche avec tabous à voisinages structurés

Rappelons que durant la première étape, l'objectif est de réduire le nombre de matchs avec 1 ou 3 juges affectés. Il est atteint en réaffectant des paires de juges. Aussi, le voisinage d'une solution réalisable x est-il généré en réaffectant une paire de juges (i,r) affectée présentement à un certain match j à un autre match $l \neq j$. Nous dénotons la nouvelle solution générée par $x \oplus (i,r,j,l)$.
La réaffectation est réalisable et la solution générée appartient au voisinage de x si et seulement si :

- les juges i et r sont admissibles au match l :

$$a_{il} = a_{rl} = 1$$

- il reste au moins un *juge en chef* affecté au match j :

$$\exists \bar{i} \in I(j,x) - \{i,r\} \quad \text{tel que} \quad l_{\bar{i}} = 1$$

- au plus 5 juges seront affectés au match l :

$$|I(l,x) \cup \{i,r\}| \leq 5.$$

Notons que la deuxième condition (respectivement la première et la dernière) ne sont pas prises en compte lorsque $j = M+1$ (respectivement $l = M+1$), car il s'agit d'un match fictif.

Afin de minimiser le risque de cycler, les réaffectations récemment utilisées sont déclarées tabou. Nous utilisons une matrice tabou $LT = [LT_{ij}]$ où, LT_{ij} identifie l'itération après laquelle le juge i peut être affecté au match j. Lorsque nous nous déplaçons de x vers $x \oplus (i,r,j,l)$, deux éléments de la matrice tabou sont modifiés comme suit :

$$LT_{ij} = curiter + t_1 \quad \text{et} \quad LT_{rj} = curiter + t_2$$

où, *curiter* représente l'indice de l'itération courante, et t_1 et t_2 sont des nombres entiers choisis aléatoirement dans l'intervalle $[t_{min}, t_{max}]$. D'autre part, étant donnée la matrice tabou courante LT, une solution voisine $x \oplus (i,r,j,l)$ est tabou à l'itération *iter* si :

$$LT_{il} \geq iter \quad \text{et} \quad LT_{rl} \geq iter.$$

Dans notre implémentation, nous utilisons le critère d'aspiration classique pour révoquer le statut tabou d'une solution lorsque sa valeur est meilleure que celle de la meilleure solution rencontrée jusqu'à présent.

Nous tirons parti de la structure du problème afin de partitionner l'ensemble des réaffectations, ce qui nous amène à définir différentes structures de voisinage. D'une manière plus formelle, soit $V(x)$ l'ensemble des réaffectations réalisables pour la solution x. Étant donnée une paire de sous-ensembles de matchs M_{out} et M_{in}, dénotons par $\{M_{out}, M_{in}\}$ le sous-ensemble de réaffectations réalisables de $j \in M_{out}$ vers $l \in M_{in}$, $j \neq l$.
Considérons la partition suivante :

$$\bigcup_{\tau=1}^{8} V_\tau(x) \subseteq V(x)$$

où, les sous-ensembles $V_\tau(x)$ sont spécifiés dans le tableau 3.8.
Nous utilisons différentes structures de voisinage $V_\tau(x)$, $\tau = 1, \ldots, 8$, à l'instar d'une recherche à voisinage variable, mais avec une stratégie différente pour se déplacer d'une structure de voisinage à une autre. Celle-ci dépend fortement de la partition du voisinage, et de l'amélioration potentielle associée aux réaffectations dans les différents sous-ensembles.
Pour chaque réaffectation menant à une solution voisine $x \oplus (i,r,j,l)$, dénotons par $\Delta(i,r,j,l)$ la modification induite dans la fonction objectif :

$$\Delta(i,r,j,l) = f(x \oplus (i,r,j,l)) - f(x).$$

τ	$V_\tau(x)$	$\Delta_\tau(x)$
1	$\{M+1, M_1(x)\}$	$-C+5M$
2	$\{M_5(x), M_1(x)\}$	$-C+10M-2$
3	$\{M+1, M_3(x)\}$	$-5M$
4	$\{M_5(x), M_3(x)\}$	-2
5	$\{M_3(x), M_1(x)\}$	-1
6	$\{M_5(x), M+1\}$	$5M-2$
7	$\{M_3(x), M_3(x)\}$	$C-10M-1$
8	$\{M_3(x), M+1\}$	$C-5M-2$

Tableau 3.8 – Partition de $V(x)$

De même, soit $\Delta_\tau(x) = \min_{(i,r,j,l)\in V_\tau(x)} \Delta(i,r,j,l)$ la meilleure modification pouvant être induite par toute réaffectation $(i,r,j,l) \in V_\tau(x)$.

Compte tenu des valeurs $\Delta_\tau(x)$ données dans la troisième colonne du tableau 3.8, il s'ensuit que les sous-ensembles $V_\tau(x)$ sont ordonnés par ordre décroissant de ces valeurs. De plus, les réaffectations dans les trois premiers sous-ensembles sont des *réaffectations améliorantes* ($\Delta_\tau(x) < 0$, $\tau = 1, 2, 3$), celles dans $V_4(x)$ et $V_5(x)$ peuvent être *légèrement améliorantes* ou *détériorantes*, et celles dans les trois derniers sous-ensembles sont des *réaffectations détériorantes* ($\Delta_\tau(x) > 0$, $\tau = 6, 7, 8$). Cette partition conduit à la stratégie de recherche suivante dans le voisinage. La recherche avec tabous est initialisée en utilisant séquentiellement les sous-ensembles $V_\tau(x)$ où $\tau = 1, 2, 3$. De plus, la recherche dans chacun de ces sous-ensembles est *exhaustive* (dans le sens où il n'y a plus de réaffectations réalisables non tabou disponibles) avant de se déplacer vers le sous-ensemble suivant. Une fois que nous avons complété la recherche *exhaustive* de $V_3(x)$, nous utilisons séquentiellement les autres sous-ensembles. Or, puisque les réaffectations comprises dans chacun de ces sous-ensembles peuvent être *détério-*

rantes, nous ne complétons pas une recherche *exhaustive* avant de se déplacer vers le sous-ensemble suivant. Après avoir complété chaque réaffectation dans $V_4(x)$, nous retournons au second sous-ensemble $V_2(x)$ initialisant une nouvelle recherche *exhaustive* utilisant les sous-ensembles $V_2(x)$ et $V_3(x)$. En effet, chaque réaffectation dans $V_4(x)$ implique que de nouveaux matchs ont été créés dans $M_3(x)$ et $M_5(x)$, et par conséquent, de nouvelles *réaffectations améliorantes* peuvent être disponibles dans $V_2(x)$ et $V_3(x)$. D'une manière analogue, après avoir complété toute réaffectation dans $\bigcup_{\tau=5}^{8} V_\tau(x)$, nous retournons au premier sous-ensemble $V_1(x)$ puisqu'une telle réaffectation implique qu'un nouveau match a été créé dans $M_1(x)$ ou de nouveaux juges ont été replacés dans le match fictif $(M+1)$.

Pour sélectionner la solution dans le voisinage de la solution courante x (généré en utilisant l'un des sous-ensembles de réaffectations $V_\tau(x)$), nous considérons les deux stratégies introduites au paragraphe 3.4.1 : la stratégie *meilleure amélioration* et la stratégie *première amélioration*. Quoiqu'en général la première stratégie nécessite la génération de toutes les solutions voisines, nous pouvons interrompre la génération du voisinage dès qu'une solution induisant une modification $\Delta_\tau(x)$ de la fonction objectif est atteinte. Rappelons que vu la structure du problème, les valeurs de $\Delta_\tau(x)$ sont connues au préalable (cf. tableau 3.8).

Finalement, le critère d'arrêt de la méthode est spécifié en termes d'un nombre maximal *nitermax* d'itérations successives complétées sans qu'il y ait eu amélioration de la fonction objectif. La méthode s'arrête également lorsque 5 juges sont affectés à tous les matchs $j \neq M+1$, ou lorsque tous les matchs $j \neq M+1$ comptent 3 ou 5 juges affectés et aucune paire de juges non affectée n'est disponible (i.e., $|M_1(x)| = 0$ et $I(M+1,x) = 0$ ou 1).

La méthode de recherche avec tabous à voisinages structurés pour l'étape 1, utilisant la stratégie *meilleure amélioration* pour sélectionner la prochaine solution courante, est résumée dans la figure 3.22.

Initialisation
x^0, une solution initiale, et $x := x^0$, la solution courante
$x^* := x^0$, la meilleure solution générée au cours de l'étape 1
Γ, l'ensemble des ρ meilleures solutions générées jusqu'à présent
$LT := \emptyset$, la liste tabou
$niter := 0$, le nombre d'itérations successives sans améliorer la fonction objectif
$fin :=$ **faux**
$\tau := 1$, l'indice du sous-ensemble de réaffectations utilisé présentement
Algorithme
Tant que ($fin =$ **faux**) **faire**
  $niter := niter + 1$, $Z := \emptyset$, $s := 1$
  EXPLORATION DES VOISINS
  Tant que ($s \leq |V_\tau(x)|$) **faire**
    $x' \in V_\tau(x)$ (considérées séquentiellement)
    Si (x' n'est pas tabou **OU** vérifie le critère d'aspiration) **Alors**
      Si ($f(x') - f(x) = \Delta_\tau(x)$) **Alors**
        $Z := \{x'\}$, $s := |V_\tau(x)| + 1$
      Sinon
        $Z := Z \cup \{x'\}$, $s := s + 1$
      Fin Si
    Sinon
      $s := s + 1$
    Fin Si
  Fait
  REMPLACEMENT DE LA SOLUTION COURANTE
  Si ($Z \neq \emptyset$) **Alors** $x := \arg\min_{z \in Z} \{f(z)\}$
  SÉLECTION DE LA PROCHAINE STRUCTURE DE VOISINAGE
  Si ($Z \neq \emptyset$) **Alors**
    Si ($\tau = 4$) **Alors** $\tau := 2$
    Si ($\tau \geq 5$) **Alors** $\tau := 1$
  Sinon
    Si ($\tau < 8$) **Alors** $\tau := \tau + 1$ **Sinon** $\tau := 1$
  Fin Si
  MISE À JOUR
  Si ($Z \neq \emptyset$) **Alors**
    Mettre à jour LT et Γ
    Si ($f(x) < f(x^*)$) **Alors** $x^* := x$, $niter := 0$
  Fin Si
  ÉVALUATION DES CRITÈRES D'ARRÊT
  Si ($|M_5(x)| = M$ **OU** ($|M_1(x)| = 0$ **ET** $I(M+1,x) = 0$ ou 1) **OU** $niter = nitermax$) **Alors** $fin :=$ **vrai**
Fait
x^* est la meilleure solution générée au cours de l'étape 1

Figure 3.22 – Recherche avec tabous à voisinages structurés pour l'étape 1 de la RTVS

3.6.4 Recherche avec tabous

La solution générée à la première étape est utilisée pour initialiser la recherche avec tabous de la seconde étape. L'objectif consiste maintenant à améliorer la diversité des domaines d'expertise des juges affectés à chaque match (bien entendu si la solution obtenue à l'étape 1 n'est pas optimale et si le temps est inférieur au temps maximal alloué *tempsmax*).

Le voisinage d'une solution réalisable x est généré en échangeant l'affectation de deux juges i et r affectés présentement à deux matchs différents j et l respectivement. Nous dénotons la nouvelle solution générée par $x \oplus (i,j,r,l)$.
L'échange est réalisable et la solution générée appartient au voisinage de x si et seulement si :

- le juge i est admissible au match l et le juge r est admissible au match j :

$$a_{il} = 1 \quad \text{et} \quad a_{rj} = 1$$

- il existe au moins un *juge en chef* dans chacun des matchs j et l :

$$\exists\, \bar{i} \in I(j,x) - \{i\} \cup \{r\} \quad \text{tel que} \quad l_{\bar{i}} = 1$$

$$\exists\, \bar{\bar{i}} \in I(l,x) - \{r\} \cup \{i\} \quad \text{tel que} \quad l_{\bar{\bar{i}}} = 1.$$

Comme à l'étape 1, nous utilisons une matrice tabou $LT = [LT_{ij}]$ où, LT_{ij} identifie l'itération après laquelle le juge i peut être affecté au match j. Lorsque nous nous déplaçons de x vers $x \oplus (i,j,r,l)$, alors deux éléments de la matrice tabou sont modifiés comme suit :

$$LT_{ij} = curiter + t_1 \quad \text{et} \quad LT_{rl} = curiter + t_2$$

où, *curiter* représente l'indice de l'itération courante, et t_1 et t_2 sont des nombres entiers choisis aléatoirement dans l'intervalle $[t_{min}, t_{max}]$. D'autre part, étant donnée la matrice

tabou courante LT, une solution voisine $x \oplus (i,j,r,l)$ est tabou à l'itération *iter* si :

$$LT_{il} \geq iter \quad \text{et} \quad LT_{rj} \geq iter.$$

De plus, nous utilisons le même critère d'aspiration et nous considérons les mêmes stratégies pour sélectionner la solution dans le voisinage de la solution courante x. Afin d'accélérer l'évaluation du voisinage, nous mémorisons la valeur des mouvements dans une matrice $\Delta(x) = [\Delta_{ir}(x)]$ où, $\Delta_{ir}(x) = f(x \oplus (i,j,r,l)) - f(x)$, et à chaque itération, nous mettons à jour seulement certaines valeurs de cette matrice. Également, lorsque nous utilisons la stratégie *meilleure amélioration*, nous pouvons interrompre la génération du voisinage dès qu'une solution voisine $x \oplus (i,j,r,l)$ induisant une modification $\Delta_{ir}(x) = -2$ de la fonction objectif est atteinte.

Finalement, le critère d'arrêt de la méthode est spécifié en termes d'un nombre maximal *nitermax* d'itérations successives complétées sans amélioration de la fonction objectif. La méthode s'arrête également lorsque la valeur de $\sum_{j=1}^{M} \sum_{k=1}^{K} \max\{d_{kj}(x) - 1, 0\}$ est égale à 0 ou à une borne inférieure b_{inf} connue *a priori* pour le problème.

Le pseudo-code de cette méthode, lorsque la stratégie *meilleure amélioration* est utilisée pour sélectionner la prochaine solution courante, est donné dans la figure 3.23.

3.6.5 Stratégie de diversification

Une fois que nous avons complété les deux premières étapes de la procédure de résolution, si la meilleure solution générée n'est pas optimale, et si le temps est inférieur au temps maximal alloué *tempsmax*, nous utilisons la stratégie de diversification suivante pour faire une recherche plus exhaustive de l'ensemble des solutions. Une nouvelle solution initiale est générée pour réinitialiser la première étape de la procédure de résolution.

Dénotons par Γ l'ensemble des ρ meilleures solutions générées jusqu'à présent au cours de la procédure, et soit $xbest \in \Gamma$ la meilleure solution dans Γ. D'abord, une nouvelle affectation x^0 des juges (non nécessairement réalisable) est générée en utilisant une variante de l'opérateur de croisement uniforme [107], lequel est communément utilisé

Initialisation
x^*, la meilleure solution générée à l'étape 1, et $x := x^*$, la solution courante
$x^{**} := x^*$, la meilleure solution générée au cours de l'étape 2
$\Delta(x)$, la matrice contenant la valeur des mouvements
Γ, l'ensemble des ρ meilleures solutions générées jusqu'à présent
$LT := \emptyset$, la liste tabou
$niter := 0$, le nombre d'itérations successives sans améliorer la fonction objectif
$fin :=$ **faux**
Algorithme
Tant que ($fin =$ **faux**) **faire**
  $niter := niter + 1$, $Z := \emptyset$, $s := 1$
  EXPLORATION DES VOISINS
  Tant que ($s \leq |V(x)|$) **faire**
    $x' := x \oplus (i, j, r, l)$ (considérées séquentiellement)
    Si (x' n'est pas tabou **OU** vérifie le critère d'aspiration) **Alors**
      Si ($\Delta_{ir}(x) = -2$) **Alors**
        $Z := \{x'\}$, $s := |V(x)| + 1$
      Sinon
        $Z := Z \cup \{x'\}$, $s := s + 1$
      Fin Si
    Sinon
      $s := s + 1$
    Fin Si
  Fait
  REMPLACEMENT DE LA SOLUTION COURANTE
  Si ($Z \neq \emptyset$) **Alors** $x := \arg\min_{z \in Z} \{f(z)\}$
  MISE À JOUR
  Si ($Z \neq \emptyset$) **Alors**
    Mettre à jour LT, $\Delta(x)$ et Γ
    Si ($f(x) < f(x^{**})$) **Alors**
      $x^{**} := x$, $niter := 0$
    Fin Si
  Fin Si
  ÉVALUATION DES CRITÈRES D'ARRÊT
  Si ($\sum_{j=1}^{M} \sum_{k=1}^{K} \max\{d_{kj}(x) - 1, 0\} = 0$ ou b_{inf} **OU** $niter = nitermax$) **Alors**
    $fin :=$ **vrai**
  Fin Si
Fait
x^{**} est la meilleure solution générée au cours de l'étape 2

Figure 3.23 – Recherche avec tabous utilisant la stratégie meilleure amélioration pour l'étape 2 de la RTVS

dans les algorithmes génétiques [50]. Ainsi, la paire de solutions *xbest* et $\bar{x} \neq xbest$, choisie aléatoirement dans Γ, sont combinées selon le processus suivant. Dans un premier temps, nous sélectionnons aléatoirement un sous-ensemble $M_{best} \subset \{1,\ldots,M+1\}$ de $\lceil \frac{\beta M}{countiter+1} \rceil$ matchs où, *countiter* représente le nombre d'itérations successives récentes pendant lesquelles la valeur de la meilleure solution générée par la procédure n'a pas été améliorée. Pour ces matchs dans M_{best}, nous retenons les affectations de *xbest*. Nous complétons ensuite la solution x^0 en retenant pour les matchs restants ($j \notin M_{best}$) les affectations de $\bar{x}$:

$$I(j,x^0) = \begin{cases} I(j,xbest) & \text{si } j \in M_{best} \\ I(j,\bar{x}) & \text{sinon.} \end{cases}$$

Notons qu'en général, le nombre d'éléments choisis de chaque solution n'est pas spécifié *a priori* lorsque l'opérateur de croisement uniforme classique est utilisé.

Maintenant x^0 peut être non réalisable car le même juge i peut être affecté à deux matchs différents $j_1 \in M_{best}$ et $j_2 \notin M_{best}$ (i.e., $i \in I(j_1,x^0) \cap I(j_2,x^0)$). Un tel juge i est éliminé de j_1 ou de j_2 comme suit. Afin de favoriser la présence des affectations de *xbest* dans la nouvelle solution x^0, nous éliminons i de j_2 sauf si $j_1 = M+1$, ou si i est le seul *juge en chef* affecté à j_2 et j_1 compte plusieurs *juges en chef* affectés ; auquel cas nous éliminons i de j_1.

Mais x^0 peut demeurer non réalisable parce que certains matchs n'ont aucun *juge en chef* affecté ou comptent seulement 2 ou 4 juges affectés. Dans ce cas, nous appliquons le *processus de réparation* suivant comprenant deux phases.
Dans un premier temps, un *juge en chef* est affecté à chaque match. Dénotons par :

$$A = \{j \neq M+1 \quad : \quad \text{aucun } \textit{juge en chef} \text{ n'est affecté à } j\}.$$

À chaque itération de cette première phase, nous sélectionnons aléatoirement un match $l \in A$. Nous permutons aléatoirement l'ensemble des *juges en chef* i admissibles à l, et nous les considérons séquentiellement (dans l'ordre de la permutation) :

– si i est affecté au match fictif $(M+1)$ ou à un match $j \neq M+1$ auquel plusieurs

juges en chef sont affectés, alors nous le retirons de j et nous le réaffectons à l ;

- si i est le seul *juge en chef* affecté à $j \neq M+1$, et si r est un autre *juge en chef* admissible à j et affecté au match fictif $(M+1)$ ou à un match $\overline{j} \neq M+1$ comptant plusieurs *juges en chef*, alors nous réaffectons r à j et i à l.

(Notons qu'il est toujours possible d'affecter un *juge en chef* à chaque match par hypothèse.)

Une fois que chaque match compte au moins un *juge en chef* affecté, nous procédons à la seconde phase pour traiter les matchs avec 2 ou 4 juges affectés. Dénotons par :

$$B = \{j \neq M+1 \quad : \quad \text{2 ou 4 juges sont affectés à } j\}.$$

À chaque itération de cette seconde phase, nous sélectionnons aléatoirement $l \in B$. S'il existe un juge $i \in I(M+1, x^0)$ admissible à l, alors nous le retirons de $(M+1)$ et le réaffectons à l. Sinon, nous choisissons aléatoirement un juge $r \in I(l, x^0)$ que nous retirons de l pour le réaffecter à $(M+1)$. Ce faisant, pour préserver la réalisabilité, nous veillons à ce que r ne soit pas le seul *juge en chef* affecté à l.

À l'issue de la phase 2, x^0 est réalisable et peut être utilisée pour réinitialiser l'étape 1 de la procédure de résolution.

Notons que l'ensemble Γ (dans lequel sont choisies les deux solutions utilisées pour générer x^0) est géré comme une liste de taille fixe (ρ). La stratégie de sélection des solutions qui feront partie de Γ se résume comme suit. Soit x la solution obtenue à l'issue d'une itération de l'étape 1 ou de l'étape 2 de la procédure de résolution. Si x est strictement meilleure que la pire solution dans Γ, alors x est systématiquement retenue. Elle remplace la pire solution dans Γ lorsque la liste est pleine. Si $f(x) = f(x')$ où, $x' \in \Gamma$ et $x' \neq x$, alors x est également retenue. Elle remplace x' lorsque la liste est pleine. Ceci permet d'inclure régulièrement des nouvelles solutions dans l'ensemble Γ tout en minimisant le risque de se retrouver avec des solutions presque identiques.

Finalement, en retenant les affectations de $\lceil \frac{\beta M}{countiter+1} \rceil$ matchs de la meilleure solution rencontrée jusqu'à présent *xbest* (pour construire la nouvelle solution initiale), où *countiter* est le nombre d'itérations successives récentes pendant lesquelles la valeur de

la meilleure solution n'a pas été améliorée, nous tentons d'explorer les régions proches, puis de plus en plus éloignées de *xbest* s'il n'y a pas eu amélioration de la valeur de la fonction objectif.

La stratégie de diversification utilisée à l'étape 3 est résumée dans la figure 3.24.

3.6.6 Expérimentations numériques

Nous comparons numériquement quatre variantes différentes de la procédure de résolution **RTVS**. Comme dans les sections précédentes, ces variantes sont spécifiées en termes du processus utilisé pour générer la solution initiale (**HLA-HOA** et la méthode d'affectation aléatoire **RS** introduite au paragraphe 3.6.2), et la stratégie utilisée pour sélectionner la solution dans le voisinage de la solution courante (*meilleure amélioration* et *première amélioration*) : **RTVS-H-Best**, **RTVS-H-First**, **RTVS-RS-Best**, et **RTVS-RS-First**.

Les paramètres C , $[t_{min}, t_{max}]$, *nitermax*, et *tempsmax* sont fixés comme pour les deux procédures **RT** et **RVV** ($50M^2$, $[\lfloor 0,8N \rfloor, \lceil 1,2N \rceil]$, N, et $3N$ respectivement où M représente le nombre de matchs et N le nombre de juges). Le paramètre ρ (nombre des meilleures solutions conservées) est fixé à la valeur $(M+1)$. Finalement, nous avons considéré trois valeurs du paramètre β (0,55, 0,65 et 0,75). Des expérimentations numériques préliminaires ont indiqué que les meilleurs résultats sont obtenus avec la valeur 0,65. C'est cette valeur donc que nous utilisons dans tous les tests subséquents.

Les tests sont complétés en utilisant l'ensemble des instances décrites au paragraphe 3.2.1. Chaque instance est résolue 10 fois avec une valeur d'initialisation différente pour le générateur des nombres aléatoires. Ces valeurs sont identiques à celles utilisées pour les deux procédures **RT** et **RVV**.

Nous utilisons également les mêmes critères de comparaison : l'écart moyen entre les valeurs des solutions générées et la valeur optimale (lorsque nous disposons de cette information) ou la borne inférieure sur cette valeur (*Ave dev*), le pourcentage des solutions optimales trouvées (*%Opt*), le temps moyen de résolution en secondes (*Ave CPU*), et l'écart-type des temps de résolution (σ_{CPU}). Toutes les solutions générées par les quatre variantes sont toujours réalisables. Par conséquent, le critère *%Feas* n'est pas considéré.

Initialisation

Γ, l'ensemble des ρ meilleures solutions générées jusqu'à présent
$xbest \in \Gamma$, la meilleure solution rencontrée jusqu'à présent
$x^0 := \emptyset$, la solution initiale à générer
$M_{best} := \emptyset$, l'ensemble des matchs où les juges affectés sont ceux dans $xbest$
$L(j,x)$, l'ensemble des *juges en chef* affectés au match j dans la solution x

Algorithme

CONSTRUCTION DE LA NOUVELLE SOLUTION

Choisir aléatoirement dans Γ une solution $\bar{x} \neq xbest$
Pour j **de** 1 **à** $\lceil \frac{\beta M}{countiter+1} \rceil$ **faire**
  Choisir aléatoirement un match $l \notin M_{best}$
  $M_{best} := M_{best} \cup \{l\}$
Fin Pour
Pour j **de** 1 **à** $M+1$ **faire**
  Si $(j \in M_{best})$ **Alors** $I(j,x^0) := I(j,xbest)$ **Sinon** $I(j,x^0) := I(j,\bar{x})$
Fin Pour
Pour chaque $i \in I(j_1,xbest) \cap I(j_2,\bar{x})$ **faire**
  Si $((j_1 = M+1)$ **OU** $(j_1 \neq M+1$ **ET** $|L(j_1,x^0)| \geq 2$ **ET** $|L(j_2,x^0)| = 1))$
  Alors
    $I(j_1,x^0) := I(j_1,x^0) - \{i\}$
  Sinon
    $I(j_2,x^0) := I(j_2,x^0) - \{i\}$
  Fin Si
Fin Pour

PROCESSUS DE RÉPARATION

PHASE 1

$A = \{j \neq M+1 \quad : \quad |L(j,x^0)| = 0\}$
Tant que $(A \neq \emptyset)$ **faire**
  Choisir aléatoirement $l \in A$
  $fin :=$ **faux**
  Tant que $(fin =$ **faux**$)$ **faire**
    Pour chaque $i \in L(j,x^0)$ tel que $a_{il} = 1$ (considérant séquentiellement les j selon un ordre aléatoire) **faire**
      Si $((j = M+1)$ **OU** $(j \neq M+1$ **ET** $|L(j,x^0)| \geq 2))$ **Alors**
        $I(j,x^0) := I(j,x^0) - \{i\}, I(l,x^0) := I(l,x^0) \cup \{i\}, fin :=$ **vrai**
      Fin Si
      Si $(j \neq M+1$ **ET** $|L(j,x^0)| = 1)$ **Alors**
        Si $((\exists r \in L(\bar{j},x^0))$ **ET** $(a_{rj} = 1)$ **ET** $(\bar{j} = M+1$ **OU** $L(\bar{j},x^0) \geq 2))$ **Alors**
          $I(\bar{j},x^0) := I(\bar{j},x^0) - \{r\}, I(j,x^0) := I(j,x^0) - \{i\} \cup \{r\}$
          $I(l,x^0) := I(l,x^0) \cup \{i\}, fin :=$ **vrai**
        Fin Si
      Fin Si
    Fin Pour
  Fait
  $A := A - \{l\}$

PHASE 2
$B = \{j \neq M+1 \quad : \quad |I(j,x^0)| = 2 \text{ ou } 4\}$
Tant que $(B \neq \emptyset)$ **faire**
  Choisir aléatoirement un match $l \in B$
  $fin :=$ **faux**
  $s := 1$
  Tant que $(fin =$ **faux**$)$ **faire**
    $i \in I(M+1,x^0)$ un juge non affecté (considérés séquentiellement)
    Si $(a_{il} = 1)$ **Alors**
      $I(M+1,x^0) := I(M+1,x^0) - \{i\}$
      $I(l,x^0) := I(l,x^0) \cup \{i\}$
      $fin :=$ **vrai**
    Sinon
      $s := s+1$
    Fin Si
    Si $(fin =$ **faux** **ET** $s > |I(M+1,x^0)|)$ **Alors**
      Choisir aléatoirement $r \in I(l,x^0)$ tel que $|L(l,x^0) - \{r\}| \geq 1$
      $I(l,x^0) := I(l,x^0) - \{r\}$
      $I(M+1,x^0) := I(M+1,x^0) \cup \{r\}$
      $fin :=$ **vrai**
    Fin Si
  Fait
  $B := B - \{l\}$
Fait
x^0 est la nouvelle solution initiale générée pour réinitialiser l'étape 1

Figure 3.24 – Stratégie de diversification pour l'étape 3 de la RTVS

Les résultats obtenus pour chaque sous-ensemble de problèmes sont résumés dans le tableau 3.9. Nous indiquons également, dans la dernière colonne de ce tableau, les résultats obtenus avec CPLEX pour les problèmes qui ont pu être résolus dans une limite de temps de 10 heures.

Solution initiale générée aléatoirement versus solution initiale de bonne qualité. À la lecture des résultats du tableau 3.9, il ressort que les deux variantes **RTVS-H-Best** et **RTVS-H-First** dominent **RTVS-RS-Best** et **RTVS-RS-First**, indiquant qu'il est plus avantageux d'initialiser la procédure de résolution avec une solution de bonne qualité. En particulier, pour les problèmes du sous-ensemble P_3 avec 500 matchs, le pourcentage des solutions optimales est amélioré de 20 %. L'écart moyen et le temps moyen de résolution sont, pour leur part, réduits d'un facteur de 137 et 4 respectivement. Ces deux variantes sont également plus stables en général.

Stratégie *meilleure amélioration* versus stratégie *première amélioration*. Il semble que la différence entre les deux stratégies est liée à la qualité de la solution initiale. On constate que la stratégie *meilleure amélioration* est légèrement meilleure lorsque la solution initiale est générée avec la technique heuristique **HLA-HOA**. Par contre, lorsque la procédure est initialisée avec une solution générée aléatoirement, la stratégie *première amélioration* s'avère meilleure, ce qui rejoint les résultats obtenus avec les deux autres procédures **RT** et **RVV**. En particulier, la variante **RTVS-RS-Best** affiche de mauvaises performances pour les problèmes avec 15 matchs alors que leur taille paraît loin de constituer une difficulté. Nous y reviendrons un peu plus loin.

Comparaison globale. Analysons maintenant les résultats relativement à la taille des problèmes. Nous présentons dans la figure 3.25 l'évolution de l'écart moyen (*Avedev*) et du temps moyen de résolution (*AveCPU*) en fonction du nombre de matchs.
Considérons les résultats des problèmes avec 50, 150 et 500 matchs. On note que la variante **RTVS-H-Best** domine suivie de près par **RTVS-H-First**, puis **RTVS-RS-First**, et finalement **RTVS-RS-Best**. Si l'on s'intéresse à l'écart moyen, on note que la diffé-

		Taille	**RTVS-H-Best**	**RTVS-RS-Best**	**RTVS-H-First**	**RTVS-RS-First**	**CPLEX**
Ave dev	P_1	15	0	0,03	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0	0	0	0	NA
	P_2	15	0	0,17	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0,03	0,01	0,03	0,04	NA
	P_3	15	0	0,06	0	0	NA
		50	0	0	0	0	NA
		150	0	0,04	0	0	NA
		500	0	1,12	0,01	0,25	NA
%Opt	P_1	15	100	97	100	100	100
		50	100	100	100	100	60
		150	100	100	100	100	50
		500	100	100	100	100	50
	P_2	15	100	85	100	100	100
		50	100	100	100	100	100
		150	100	100	100	100	100
		500	97	99	97	96	100
	P_3	15	100	94	100	100	100
		50	100	100	100	100	60
		150	100	96	100	100	50
		500	100	70	99	89	40
Ave CPU (sec)	P_1	15	0,04	6,11	0,03	0,06	28,74
		50	0,15	0,17	0,18	0,23	2,31
		150	0,46	0,32	0,48	1,07	96,38
		500	11,26	1,30	12,23	26,98	12 159,96
	P_2	15	0,08	59,63	0,10	0,15	0,69
		50	1,25	1,82	1,01	1,13	1,93
		150	60,08	145,71	29,90	31,55	32,41
		500	1 036,40	1 477,29	1 354,39	1 246,47	21 299,74
	P_3	15	0,04	12,42	0,04	0,06	13,12
		50	0,21	0,30	0,21	0,33	8,85
		150	3,34	217,94	13,25	132,70	276,17
		500	436,64	3 291,74	770,37	1 655,56	23 824,47
σ_{CPU}	P_1	15	0,05	34,69	0,04	0,05	33,59
		50	0,15	0,20	0,20	0,30	0,58
		150	0,30	0,31	0,32	0,39	71,55
		500	2,16	0,58	3,14	6,46	5 408,41
	P_2	15	0,09	146,32	0,26	0,18	0,09
		50	2,34	2,40	1,77	1,30	0,25
		150	106,64	192,36	48,21	39,83	8,14
		500	1 570,29	1 667,90	1 974,28	1 941,67	4 130,39
	P_3	15	0,09	49,16	0,03	0,05	16,75
		50	0,31	0,36	0,28	0,39	12,51
		150	10,20	482,73	42,23	286,23	201,12
		500	1 024,36	2 868,76	1 454,71	2 240,83	4 216,50

Tableau 3.9 – Comparaison de l'efficacité des variantes de la RTVS en fonction des sous-ensembles de problèmes

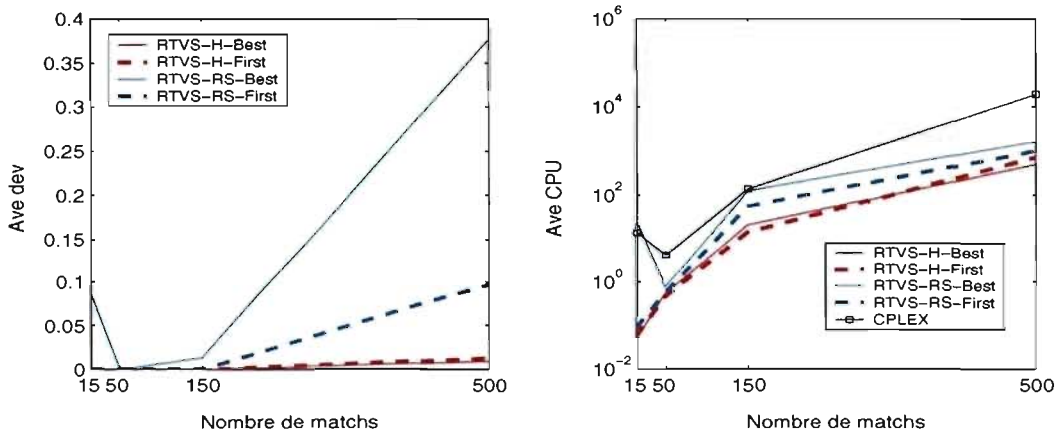

Figure 3.25 – Comparaison de l'efficacité des variantes de la RTVS en fonction de la taille des problèmes

rence entre les quatre variantes se creuse avec la taille des problèmes. Si l'on s'attache au temps moyen de résolution, on constate que la différence entre les quatre variantes est plus accentuée pour les problèmes avec 150 matchs. En particulier, les temps de résolution de la variante **RTVS-RS-Best** sont presque similaires à ceux de CPLEX. Notons toutefois que seulement 66,67% de ces instances ont pu être résolues à l'optimalité avec CPLEX dans une limite de temps de 10 heures.

Considérons maintenant les résultats des problèmes avec 15 matchs. Les trois variantes **RTVS-H-Best**, **RTVS-H-First** et **RTVS-RS-First** affichent d'excellents résultats. Toutes les solutions générées sont optimales, et les temps de résolution sont sans commune mesure avec ceux de CPLEX (réduits en moyenne d'un facteur de 263, 250 et 159 respectivement). Les performances de la variante **RTVS-RS-Best** sont radicalement opposées. Les solutions générées ne sont pas toutes optimales, et les temps de résolution dépassent ceux de CPLEX (en moyenne d'un facteur de 2). Malgré ces bémols, nous avons pu constater que pour chaque instance, au moins l'une des 10 solutions générées par la variante est optimale, et qu'au plus, dans un seul match, un seul juge a le même domaine d'expertise qu'un autre juge.

Pour analyser davantage la relation entre les quatre variantes, nous les avons comparé

deux à deux en utilisant le test des rangs pour échantillons appariés de WILCOXON. Le tableau 3.10 synthétise les résultats obtenus avec un seuil de confiance égal à 0,05. Comme dans les sections précédentes, nous reportons dans les cellules de ce tableau les *p_valeurs* obtenues pour les différents couples de variantes. Celles-ci sont identifiées par leurs noms (colonne 1 et ligne 2 du tableau). Rappelons qu'une *p_valeur* inférieure à 0,05 indique que la variante dont le nom figure à la colonne 1 est statistiquement meilleure que celle dont le nom figure à la ligne 2. Autrement (i.e., si $p\_valeur > 0,05$), ceci signifie qu'il n'y a pas de différence statistiquement significative entre les deux variantes considérées, auquel cas le résultat du test est souligné en caractère gras.

	Qualité de la solution			Temps de résolution		
	RTVS-H-First	**RTVS-RS-First**	**RTVS-RS-Best**	**RTVS-H-First**	**RTVS-RS-First**	**RTVS-RS-Best**
RTVS-H-Best	**0,78**	2,56E-03	3,27E-11	5,57E-08	< 2,2E-16	< 2,2E-16
RTVS-H-First	-	4,47E-03	5,85E-11	-	<2,2E-16	2,69E-15
RTVS-RS-First	-	-	1,03E-10	-	-	**0,50**

Tableau 3.10 – Résultats du test de WILCOXON pour les variantes de la RTVS

Les résultats du test indiquent qu'au niveau de la qualité des solutions, il n'y a pas de différence statistiquement significative entre les variantes **RTVS-H-Best** et **RTVS-H-First**, mais que celles-ci surpassent les deux autres variantes **RTVS-RS-Best** et **RTVS-RS-First**. Quant au temps de résolution, les résultats du test confirment la nette supériorité de la variante **RTVS-H-Best**. **RTVS-H-First** se classe en second alors qu'il n'y a pas de différence statistiquement significative entre les variantes **RTVS-RS-Best** et **RTVS-RS-First**, lesquelles se classent en dernier.

La figure 3.26 illustre le comportement de *Ave dev* durant la résolution des problèmes avec 500 matchs. Chaque courbe est associée à une variante différente.

Durant les 30 premières secondes, l'allure des courbes indique que les deux variantes **RTVS-RS-Best** et **RTVS-RS-First** affichent de meilleures performances que **RTVS-H-Best** et **RTVS-H-First**. Ceci est dû au fait que la technique **HLA-HOA** consomme plus de temps pour générer la solution initiale de **RTVS-H-Best** et **RTVS-H-First** que l'étape 1 de la procédure de résolution utilisée pour améliorer la solution initiale générée presque instantanément avec la procédure **RS**. Toutefois, le temps CPU requis par

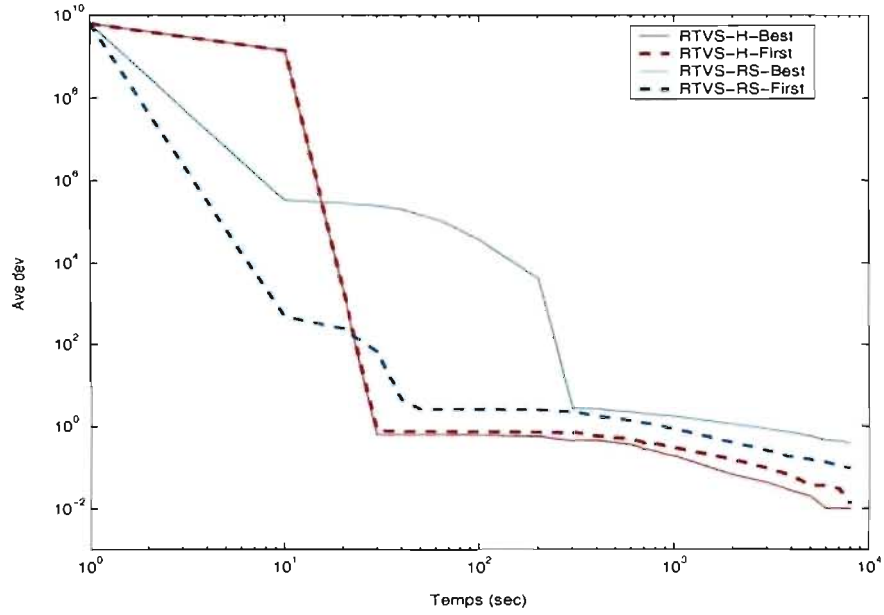

Figure 3.26 – Évolution de *Ave dev* des variantes de la RTVS pour les problèmes avec 500 matchs

HLA-HOA est bénéfique puisque, lorsque complété (i.e., après 30 secondes), les solutions de **RTVS-H-Best** et **RTVS-H-First** sont meilleures que celles de **RTVS-RS-Best** et **RTVS-RS-First**. De plus, les deux variantes utilisant **HLA-HOA** maintiennent leur avance sur les deux autres variantes durant le reste du temps de résolution.

Considérons maintenant les deux courbes associées aux variantes **RTVS-RS-Best** et **RTVS-RS-First** (toujours durant les 30 premières secondes). On constate que **RTVS-RS-First** affiche de meilleures performances que **RTVS-RS-Best**. Les valeurs élevées de *Ave dev* de cette dernière variante traduisent le fait que 3 juges sont affectés à certains matchs quoiqu'il reste des paires de juges non affectés (disponibles dans $(M+1)$). Une explication plausible réside dans le fait que le caractère «glouton» de **RTVS-RS-Best** lors de l'utilisation de la structure de voisinage $\{M+1, M_1(x)\}$ réduit l'impact de la structure de voisinage $\{M+1, M_3(x)\}$, dans le sens où, parfois, les paires de juges disponibles dans $(M+1)$ ne sont admissibles à aucun match dans $M_3(x)$. *Ave dev* de **RTVS-RS-Best** ne rejoint celle de **RTVS-RS-First** que vers la $300^{\text{ème}}$ seconde après plusieurs

applications de la stratégie de diversification. À partir de ce moment, les améliorations obtenues par **RTVS-RS-Best** ne sont pas très importantes et sa courbe a tendance à stagner. Les courbes associées aux trois autres variantes stagnent aussi, mais à des valeurs moins élevées de *Ave dev*.

En conclusion, les résultats numériques indiquent que **RTVS-H-Best** domine les trois autres variantes qui peuvent être classées comme suit : **RTVS-H-First**, **RTVS-RS-First**, et **RTVS-RS-Best**.

3.7 Comparaison des meilleures variantes

Dans les trois dernières sections, nous avons présenté trois procédures de résolution basées sur une approche métaheuristique (**RT**, **RVV** et **RTVS**), et nous avons comparé quatre variantes de chaque procédure pour évaluer l'impact de la qualité de la solution initiale et de la stratégie de sélection dans le voisinage sur les performances de chacune des procédures développées.

Il ressort de l'ensemble des tests réalisés qu'on ne peut pas tirer de conclusion générale quant à l'influence de la qualité de la solution initiale. En effet, les performances de la **RVV** semblent dépendre davantage de la stratégie de sélection dans le voisinage. Pour la **RT**, il semble que l'utilisation d'une solution initiale de bonne qualité a un impact négatif sur les performances de la procédure. Le contraire se produit pour la **RTVS**.
En revanche, l'impact de la stratégie de sélection dans le voisinage est plus clair : il semble qu'il est toujours plus avantageux d'utiliser la stratégie *première amélioration*, sauf pour les deux variantes **RTVS-H-Best** et **RTVS-H-First** où la première variante s'est avérée meilleure eu égard au temps de résolution.

Nous comparons maintenant les trois meilleures variantes **RT-R-First**, **RVV-R-First** et **RTVS-H-Best**. Le tableau 3.11 récapitule les résultats obtenus : pour chaque taille de poblèmes, nous indiquons le pourcentage des solutions optimales obtenues (*%Opt*) et le temps moyen de résolution en secondes (*Ave CPU*). Ce tableau contient également les résultats obtenus avec CPLEX.
Ces résultats indiquent clairement l'avantage d'utiliser les méthodes basées sur une ap-

Taille	CPLEX		RT-R-First		RVV-R-First		RTVS-H-Best	
	%Opt	*Ave CPU*	*%Opt*	*Ave CPU*	*%Opt*	*Ave CPU*	*%Opt*	*Ave CPU*
15	100	14,18	100	0,11	100	0,11	100	0,05
50	73,33	4,36	100	0,60	100	0,64	100	0,54
150	66,67	134,99	100	12,56	100	13,31	100	21,29
500	63,33	19 094,72	100	498,06	100	538,58	99	494,76

Tableau 3.11 – Récapitulatif des résultats des meilleures variantes

proche métaheuristique plutôt que CPLEX, mais il est plus difficile de décider quelle méthode choisir.

Pour poursuivre la comparaison des trois variantes, nous avons effectué le test de WILCOXON pour échantillons appariés sur les résultats de l'ensemble des résolutions. Le seuil de confiance utilisé est égal à 0,05. Les résultats de ce test, présentés dans le tableau 3.12, indiquent qu'au niveau de la qualité des solutions, les trois variantes ne sont pas statistiquement différentes ($p\_valeur > 0,05$). Au niveau du temps de résolution, **RTVS-H-Best** domine nettement les deux autres variantes (*p_valeur* très petite). Ces dernières sont comparables.

	Qualité de la solution		Temps de résolution	
	RT-R-First	**RVV-R-First**	**RT-R-First**	**RVV-R-First**
RTVS-H-Best	**0,15**	**0,15**	< 2,2E-16	< 2,2E-16
RT-R-First	-	1	-	**0,30**

Tableau 3.12 – Résultats du test de WILCOXON pour les meilleures variantes

Enfin, pour avoir un aperçu plus complet sur le comportement des trois variantes **RT-R-First**, **RVV-R-First** et **RTVS-H-Best**, nous présentons dans la figure 3.27, l'évolution de *Ave dev* en fonction du temps pour les problèmes avec 500 matchs.
On remarque que de bonnes solutions sont trouvées par **RTVS-H-Best** dès la 30ème seconde. Les améliorations obtenues par la suite sont très modérées, et la courbe a tendance à stagner. Ce comportement (stagnation) n'est pas spécifique à cette variante. On peut également l'observer pour les deux autres variantes, mais il faut remarquer que leurs courbes stagnent à des valeurs nettement plus élevées de *Ave dev*. De plus, ce n'est qu'après 1200 secondes (repectivement 1400 secondes) que **RT-R-First** (respectivement **RVV-R-First**) parvient à atteindre un niveau égal à celui de **RTVS-H-Best**. Par contre,

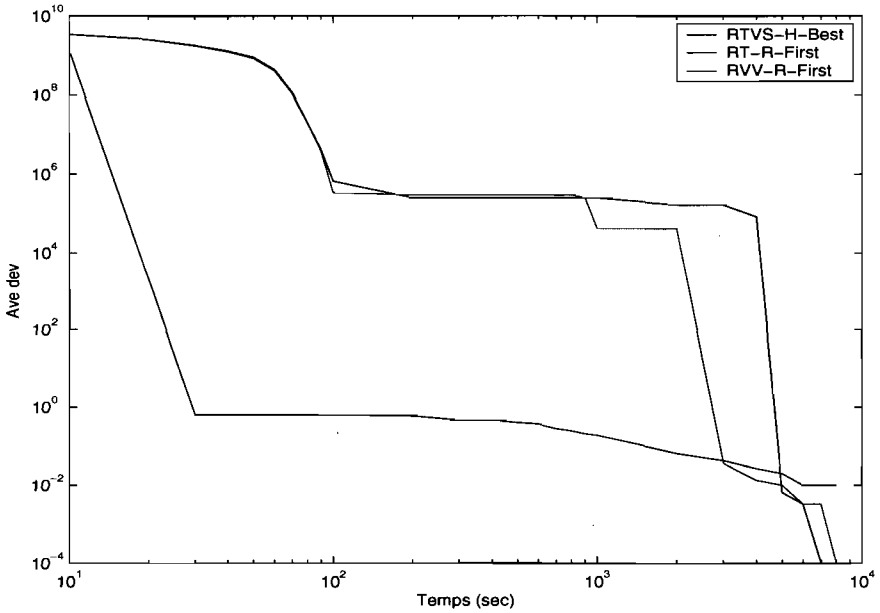

Figure 3.27 – Évolution de *Avedev* des meilleures variantes pour les problèmes avec 500 matchs

ces deux variantes réussissent vers la fin du processus de résolution à réduire l'écart à 0, ce qui n'est pas le cas pour **RTVS-H-Best**. L'écart demeure toutefois très marginal.
À quelques exceptions près, les deux variantes **RT-R-First** et **RVV-R-First** ont le même comportement. Durant les 1 000 premières secondes, leurs courbes sont presque confondues. Sur l'intervalle [1 000, 1 400], on constate un net avantage pour **RT-R-First**, ce qui laisserait supposer que la stratégie de diversification de la **RT** est plus efficace que la stratégie de perturbation de la **RVV**, dans la mesure où elle accélère la convergence. Par contre, vers la fin du processus de résolution, **RVV-R-First** prend l'avantage.

En somme, les trois variantes surclassent CPLEX et génèrent des solutions d'excellente qualité, mais, compte tenu du temps de résolution requis, **RTVS-H-Best** domine **RT-R-First** et **RVV-R-First**.

3.8 Conclusion

L'objectif de ce chapitre était de développer, d'évaluer, et de comparer des méthodes permettant de résoudre efficacement *le problème sur une seule ronde* introduit à la section 1.2.

Nous avons d'abord formulé ce problème à l'aide d'un modèle de programmation binaire (modèle M^1). Une linéarisation de ce modèle nous a permis de le résoudre avec le logiciel CPLEX. Nous avons pu constater que trouver une solution exacte du problème peut s'avérer extrêmement laborieux notamment lorsque le nombre de juges disponibles est insuffisant pour affecter 5 juges à chaque match. Nous nous sommes donc tournés vers la résolution approchée.

En premier lieu, nous avons proposé deux techniques de résolution heuristiques basées sur une approche séquentielle. La première est une hybridation de méthodes exactes et de métaheuristiques. La seconde utilise des heuristiques constructives gloutonnes et intègre une vision à long terme du processus de résolution. Les expérimentations menées ont indiqué que les deux techniques produisent de bons résultats. Cependant, elles reposent sur une approche séquentielle qui présente une lacune de taille : les affectations sont faites sans retour en arrière. Par conséquent, les deux techniques proposées peuvent ne pas obtenir des solutions réalisables. Nous avons donc considéré l'approche métaheuristique. Cette approche est plus puissante et offre une plus grande souplesse. Nous avons donc modifié nos formulations du problème et considéré deux nouvelles versions pénalisées du modèle mathématique original M^1.

Pour résoudre les nouvelles formulations du problème, nous avons d'abord proposé une adaptation de la métaheuristique de recherche avec tabous qui exploite à bon escient les informations collectées durant le processus de recherche. Cela se traduit par :

- le recours à une mesure de fréquence pour favoriser le déplacement vers les régions les moins explorées de l'ensemble des solutions et ce, aussi bien lors de la phase d'exploration du voisinage que lors de la phase de diversification ;
- l'adaptation de la stratégie de diversification en fonction des contraintes violées.

Les expérimentations numériques menées ont montré l'efficacité de la procédure dé-

veloppée. Toutefois, une analyse plus approfondie des résultats obtenus nous a conduit à déceler quelques faiblesses de l'approche de résolution : la stratégie de diversification adoptée ne permet pas de s'échapper des vallées contenant les optima locaux. Plus explicitement, le processus de recherche reste localisé dans les régions proches de la meilleure solution lorsque le nombre de violations des contraintes est réduit. Nous avons donc considéré la métaheuristique de recherche à voisinage variable étant donné que son principe permet d'emblée de remédier à cette faiblesse.

L'étape perturbation de la procédure que nous avons développée est caractérisée par la mise en place d'une mémoire à long terme pour favoriser le déplacement vers les régions les moins explorées de l'ensemble des solutions. Au vu des résultats obtenus, il ressort que la RVV affiche des performances en deçà de nos espérances. En effet, nous avons constaté que la procédure développée a réellement avantage à être appliquée à partir d'une solution de bonne qualité. Le cas échéant, la recherche peut rester confinée dans des régions ne contenant que des solutions non réalisables (en particulier lorsque le nombre de juges est insuffisant pour affecter 5 juges à chaque match).

La troisième procédure que nous avons développée (recherche avec tabous à voisinages structurés) se propose de combler cette lacune. Elle exploite à la fois des principes de la recherche à voisinage variable, de la recherche avec tabous, et des algorithmes génétiques. En effet, la première étape de cette procédure est caractérisée par l'utilisation de différentes structures de voisinage. La seconde étape utilise une recherche avec tabous. La dernière étape (stratégie de diversification) se distingue par les éléments suivants :

- une mémoire adaptative servant à conserver les solutions qui seront utilisées pour générer la nouvelle solution initiale. La stratégie utilisée pour choisir ces solutions favorise la sélection de solutions à la fois diverses et de bonne qualité ;
- un mécanisme de sélection de matchs qui vise, d'une part, à produire une nouvelle solution initiale de bonne qualité et, d'autre part, à explorer les régions proches, puis de plus en plus éloignées de la meilleure solution s'il n'y a pas eu amélioration de la valeur de la fonction objectif ;
- un mécanisme de croisement qui vise à produire une nouvelle solution de bonne

qualité ;

- un mécanisme de réparation qui permet de ramener la recherche dans le domaine réalisable.

Bien que cette dernière procédure a permis d'obtenir de très bons résultats, il n'en demeure pas moins que l'approche utilisée dépend fortement de la structure du problème (pour déterminer les différentes structures de voisinage utilisées lors de la première étape).

En somme, nous avons introduit et comparé plusieurs procédures de résolution. Nous avons identifié les lacunes de chaque procédure et les avons modifié en conséquence. Les résultats sont très satisfaisants. Pour la majorité des instances testées, nous sommes en mesure de retrouver les résultats obtenus avec CPLEX bien plus rapidement.

Les perspectives de développement de cette étude sont multiples. En particulier, pour la procédure RVV, augmenter graduellement les valeurs des pénalités associées à la violation des contraintes incontournables (paramètre C du modèle M^2) pour se ramener dans le domaine réalisable est une voie prometteuse pour améliorer l'efficacité de l'approche (cette technique a été proposée dans [42]). Par ailleurs, le principe des différentes procédures proposées peut être utilisé et adapté pour résoudre d'autres problèmes. Dans le chapitre suivant, nous proposons des extensions pour résoudre *le problème sur plusieurs rondes* introduit à la section 1.3. Quelques modifications seront apportées à ces procédures pour tenir compte des nouvelles contraintes que nous ne retrouvons pas dans le problème étudié dans ce chapitre, et aussi, pour accroître davantage leurs performances.

CHAPITRE 4

ÉTUDE EXPÉRIMENTALE DU PROBLÈME SUR PLUSIEURS RONDES

Ce chapitre aborde *le problème sur plusieurs rondes* introduit à la section 1.3. Plutôt que de déterminer l'affectation des juges aux matchs d'une seule ronde comme nous l'avons fait au chapitre précédent, il faut considérer ici l'ensemble des rondes. Des contraintes relient les rondes entre elles. De plus, comme nous l'avons déjà souligné à l'introduction et au chapitre 1, au cours des différents échanges que nous avons eus avec les organisateurs du concours *John Molson*, ils nous ont présenté d'autres facettes du problème qu'ils ne nous avaient pas spécifié initialement. Le problème s'en trouve donc enrichi de nouvelles contraintes. L'objectif de ce chapitre est de **développer une approche de résolution de ce problème pratique s'appuyant sur les méthodes proposées au chapitre précédent, et de comparer diverses variantes afin d'identifier celle qui est la plus performante**.

Ce chapitre débute avec un rappel des contraintes qui doivent être prises en compte pour déterminer les affectations. Le problème est ensuite formulé à l'aide d'un modèle de programmation à buts multiples où la somme pondérée des violations des contraintes souples est minimisée. Celles-ci sont classées par ordre de priorité. Les poids que nous utilisons pour refléter leur importance relative ne sont pas choisis arbitrairement, mais sont dérivés d'une analyse théorique. Les résultats permettant de déterminer les valeurs de ces poids ne sont pas valides seulement pour ce *problème sur plusieurs rondes*, mais sont plus généraux. Le modèle inclut aussi des contraintes quadratiques permettant de modéliser l'une des contraintes reliant les rondes entre elles. Nous proposons des linéarisations de ces contraintes afin de pouvoir résoudre le problème avec un solveur de programmes linéaires. Les résultats de cette expérimentation numérique sont présentés dans la section 4.2.

La section suivante (section 4.3) porte sur l'approche de résolution par les métaheuristiques. Nous proposons une stratégie de décomposition permettant de résoudre le problème en résolvant séquentiellement les sous-problèmes associés aux différentes

rondes. Pour résoudre un sous-problème, nous proposons des généralisations des méthodes présentées au chapitre précédent. De plus, nous améliorons significativement les performances de la méthode **RTVS** en explorant différemment le voisinage de la solution courante. Les détails des principales composantes de ces méthodes sont présentées dans les sections 4.4, 4.5 et 4.6, de même que les résultats des expérimentations numériques. La section 4.7 présente une synthèse de ces résultats. Une conclusion complète ce chapitre.

Les instances dont nous nous servons pour tester les différentes méthodes sont générées aléatoirement. Toutes les procédures de résolution sont implantées en utilisant le langage de programmation `Java`. Les tests sont réalisés sur un ordinateur doté d'un processeur *AMD Opteron 246 1993 Mhz* avec *2 GB* de mémoire et fonctionnant sous le système d'opération *Linux*.

4.1 Modélisation du problème

Pour *le problème sur plusieurs rondes*, l'affectation des juges aux matchs est régie par l'ensemble des contraintes suivantes :

Règles incontournables ou contraintes qui doivent être satisfaites :

1. un juge ne peut être affecté à un match impliquant une équipe représentant une Université dans laquelle il a étudié ou à laquelle il est attaché (s'il est professeur). De plus, il ne peut être affecté à un match impliquant une équipe qu'il ne souhaite pas évaluer. Dans les deux cas, on dira que le juge est en *conflit* avec cette équipe. Notons que les organisateurs du concours prennent en compte au plus 4 équipes avec lesquelles un juge peut être *en conflit* ;
2. un juge affecté à un match *francophone* doit être *bilingue* ;
3. 3 ou 5 juges doivent être affectés à chaque match ;
4. au moins un *juge en chef* doit être affecté à chaque match ;
5. au moins un juge *expert*, autre que le *juge en chef*, doit être affecté à chaque match.

Les deux premières contraintes définissent les affectations permises ou admissibles tandis que les trois dernières traduisent les règles définissant la composition du jury. On désignera par *contraintes d'admissibilité* l'ensemble des contraintes incontournables 1 et 2, et par *contraintes de composition* l'ensemble des contraintes incontournables 3, 4 et 5.

Règles souples ou contraintes (ou objectifs) à satisfaire autant que possible :

1. *contrainte d'équilibre* : hormis le *juge en chef*, 2 ou 4 juges additionnels sont affectés à chaque match. Pour ces juges, le nombre des *experts* doit être égal au nombre des *nouveaux* ;
2. *contrainte d'affiliation* : si plusieurs juges représentant des compagnies sont affectés à un match, alors ils doivent être affiliés à des compagnies différentes ;
3. *contrainte de diversité* : les expertises des juges affectés à un même match doivent couvrir le plus grand nombre de domaines d'expertise ;
4. *contrainte de couplage* : une paire spécifique de juges ne peut être affectée plus d'une fois au cours de l'ensemble des rondes ;
5. *contrainte de séquence* : durant les différentes rondes, un juge ne doit pas être affecté à différents matchs impliquant la même équipe ;
6. *contrainte du nombre* : le nombre de matchs avec 5 juges affectés doit être maximisé.

Les contraintes souples 4 et 5 interviennent au fur et à mesure du déroulement des rondes. On les désignera par *contraintes inter rondes*.

4.1.1 Modèle mathématique général

Notre objectif est de déterminer une affectation des juges aux différents matchs des différentes rondes qui est réalisable, dans le sens où elle respecte les règles incontournables d'affectation, et qui satisfait le plus possible aux règles souples. Un modèle de programmation à buts multiples (*goal programming*) semble être approprié pour formaliser un tel problème.

Pour formuler notre modèle, nous utilisons la notation suivante :

- p : l'indice de la ronde, $p = 1, \ldots, P$
- j : l'indice du match d'une ronde, $j = 1, \ldots, M$
- t : l'indice de l'équipe, $t = 1, \ldots, 2M$
- i : l'indice du juge, $i = 1, \ldots, N$
- k : l'indice du domaine d'expertise, $k = 1, \ldots, K$
- c : l'indice de la compagnie, $c = 1, \ldots, C$
- $S = [s_{tjp}]$ où, $s_{tjp} = \begin{cases} 1 & \text{si l'équipe } t \text{ est en compétition dans le match } j \text{ de la ronde } p \\ 0 & \text{sinon} \end{cases}$
- $A = [a_{ijp}]$ où, $a_{ijp} = \begin{cases} 1 & \text{si le juge } i \text{ est admissible au match } j \text{ de la ronde } p \\ 0 & \text{sinon} \end{cases}$

  Rappelons que le juge i est admissible au match j de la ronde p si les deux premières règles incontournables (*contraintes d'admissibilité*) sont respectées.
- $l_i = \begin{cases} 1 & \text{si le juge } i \text{ est un } \textit{juge en chef} \\ 0 & \text{sinon} \end{cases}$
- $v_i = \begin{cases} 1 & \text{si le juge } i \text{ est un juge } \textit{expert} \text{ mais n'est pas un } \textit{juge en chef} \\ 0 & \text{sinon} \end{cases}$

  Rappelons que tous les *juges en chef* sont des juges *experts*, mais que l'inverse est faux.
- $D = [d_{ip}]$ où, $d_{ip} = \begin{cases} 1 & \text{si le juge } i \text{ est disponible pour la ronde } p \\ 0 & \text{sinon} \end{cases}$
- $E = [e_{ik}]$ où, $e_{ik} = \begin{cases} 1 & \text{si le juge } i \text{ possède le domaine d'expertise } k \\ 0 & \text{sinon} \end{cases}$
- $R = [r_{ic}]$ où, $r_{ic} = \begin{cases} 1 & \text{si le juge } i \text{ représente la compagnie } c \\ 0 & \text{sinon} \end{cases}$
- Les variables x_{ijp} représentent l'affectation du juge i au match j de la ronde p :

  $x_{ijp} = \begin{cases} 1 & \text{si le juge } i \text{ est affecté au match } j \text{ de la ronde } p \\ 0 & \text{sinon} \end{cases}$
- Les variables y^3_{jp} et y^5_{jp} indiquent le nombre de juges affectés au match j de la ronde p :

$$y^3_{jp} = \begin{cases} 1 & \text{si 3 juges sont affectés au match } j \text{ de la ronde } p \\ 0 & \text{sinon} \end{cases}$$

$$y^5_{jp} = \begin{cases} 1 & \text{si 5 juges sont affectés au match } j \text{ de la ronde } p \\ 0 & \text{sinon} \end{cases}$$

- Les variables d^{1+}_{jp} et d^{1-}_{jp} servent à modéliser la contrainte d'*équilibre*. Elles représentent l'écart entre le nombre de juges *experts* et *nouveaux* affectés au match j de la ronde p.
- De même, nous associons à chacune des autres règles souples d'affectation une variable d'écart qui prend la valeur 0 si la règle est respectée et qui reflète, le cas échéant, le nombre de violations de cette règle. Nous dénotons par d^2_{cjp}, d^3_{kjp}, d^4_{ir}, et d^5_{it} les variables d'écart associées respectivement aux contraintes d'*affiliation*, de *diversité*, de *couplage*, et de *séquence*.
- Finalement, les $w_1, \ldots, w_6$ représentent les poids associés aux différentes règles souples d'affectation.

Le modèle que nous proposons est résumé dans la figure 4.1. Dans la fonction objectif (4.1), nous minimisons la somme pondérée des violations des règles souples d'affectation. Les contraintes (4.2) indiquent qu'un juge ne peut être affecté à un match d'une ronde que s'il est disponible pour cette ronde. De plus, étant donné que les matchs d'une même ronde se déroulent simultanément, un juge ne peut être affecté à plus d'un match de la même ronde. Les contraintes (4.3) ne permettent pas qu'un juge inadmissible soit affecté à un match. Au moins un *juge en chef* et un autre juge *expert*, différent du *juge en chef*, sont affectés à chaque match de chaque ronde d'après les contraintes (4.4). Les contraintes (4.5) et (4.6) garantissent qu'exactement 3 ou 5 juges sont affectés à chaque match de chaque ronde. Les contraintes (4.7) spécifient que, hormis le *juge en chef*, le nombre de juges *experts* et *nouveaux* affectés à chaque match sont égaux, sinon, la pénalité $w_1(d^{1+}_{jp} + d^{1-}_{jp})$ est ajoutée à la fonction objectif. Chaque compagnie c est représentée au plus une fois dans chaque match d'après les contraintes (4.8), le cas échéant, la pénalité $w_2 d^2_{cjp}$ est encourue. Les contraintes (4.9) stipulent que chaque domaine d'expertise k est couvert au moins une fois dans chaque match, sinon, la pénalité $w_3 d^3_{kjp}$ est ajoutée

$$\min \quad w_1 \sum_{p=1}^{P}\sum_{j=1}^{M}(d_{jp}^{1+}+d_{jp}^{1-}) + w_2 \sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{c=1}^{C} d_{cjp}^{2} + w_3 \sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{k=1}^{K} d_{kjp}^{3} \quad (4.1)$$
$$+ w_4 \sum_{i=1}^{N-1}\sum_{r=i+1}^{N} d_{ir}^{4} + w_5 \sum_{t=1}^{2M}\sum_{i=1}^{N} d_{it}^{5} + w_6 \sum_{p=1}^{P}\sum_{j=1}^{M} y_{jp}^{3}$$

(MG^1) **Sujet à**

$$\sum_{j=1}^{M} x_{ijp} \le d_{ip} \qquad \forall\ i,p \qquad (4.2)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M}(1-a_{ijp})x_{ijp} = 0 \qquad \forall\ i \qquad (4.3)$$

$$\sum_{i=1}^{N}(5l_i+v_i)x_{ijp} \ge 6 \qquad \forall\ j,p \qquad (4.4)$$

$$\sum_{i=1}^{N} x_{ijp} = 3y_{jp}^{3} + 5y_{jp}^{5} \qquad \forall\ j,p \qquad (4.5)$$

$$y_{jp}^{3} + y_{jp}^{5} = 1 \qquad \forall\ j,p \qquad (4.6)$$

$$\sum_{i=1}^{N}(2l_i+2v_i-1)x_{ijp} - d_{jp}^{1+} + d_{jp}^{1-} = 1 \qquad \forall\ j,p \qquad (4.7)$$

$$\sum_{i=1}^{N} r_{ic}x_{ijp} - d_{cjp}^{2} \le 1 \qquad \forall\ c,j,p \qquad (4.8)$$

$$\sum_{i=1}^{N} e_{ik}x_{ijp} + d_{kjp}^{3} \ge 1 \qquad \forall\ k,j,p \qquad (4.9)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M} x_{ijp}x_{rjp} - d_{ir}^{4} \le 1 \qquad \forall\ i,r>i \qquad (4.10)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} - d_{it}^{5} \le 1 \qquad \forall\ i,t \qquad (4.11)$$

$$x_{ijp} = 0 \text{ ou } 1 \qquad \forall\ i,j,p \qquad (4.12)$$

$$y_{jp}^{3}, y_{jp}^{5} = 0 \text{ ou } 1 \qquad \forall\ j,p \qquad (4.13)$$

$$d_{jp}^{1+}, d_{jp}^{1-}, d_{cjp}^{2}, d_{kjp}^{3}, d_{ir}^{4}, d_{it}^{5} \ge 0 \qquad \forall\ i,r>i,t,j,p,c,k \qquad (4.14)$$

Figure 4.1 – Modèle MG^1 formalisant le problème sur plusieurs rondes

à la fonction objectif. Les contraintes (4.10) et (4.11) modélisent les *contraintes inter rondes* : chaque paire spécifique de juges est affectée (respectivement chaque juge est affecté à un match impliquant la même équipe) au plus une fois. Les variables d'écart d_{ir}^4 et d_{it}^5 évaluent les déviations de cette valeur cible, et leur somme pondérée est pénalisée dans la fonction objectif. Finalement, le dernier terme de la fonction objectif reflète la *contrainte du nombre* puisque maximiser le nombre de matchs avec 5 juges affectés est équivalent à minimiser le nombre de matchs avec 3 juges affectés.

4.1.2 Pondérations

Tel que mentionné précédemment, notre objectif consiste à déterminer une affectation des juges aux différents matchs des différentes rondes qui est réalisable, dans le sens où elle respecte les règles incontournables d'affectation, et qui satisfait le plus possible aux règles souples. L'importance relative de ces règles est spécifiée par des priorités relatives. En effet, maximiser le nombre de matchs avec 5 juges affectés (*contrainte du nombre*) est plus prioritaire que satisfaire les différentes autres règles souples. Ces dernières sont à leur tour considérées selon la hiérarchie suivante (par ordre de priorité décroissant) :

- *séquence* ;
- *couplage* ;
- *diversité* ;
- *affiliation* ;
- *équilibre*.

Ainsi, nous devons éviter d'améliorer une règle d'affectation moins prioritaire aux dépens d'une autre plus prioritaire. Pour atteindre cet objectif, une première approche (méthode séquentielle) consiste à optimiser la règle la plus prioritaire. S'il y a plusieurs solutions optimales, alors on retient parmi celles-ci, celles qui optimisent la deuxième règle plus prioritaire et ainsi de suite [105].

La seconde approche consiste à associer à chaque règle souple d'affectation de priorité v un poids relatif $w_v > 0$, et à optimiser un seul objectif qui s'exprime comme une combinaison linéaire des différents objectifs. Notre choix s'est porté sur cette seconde

approche car elle présente l'avantage de résoudre un seul problème plutôt que plusieurs (cf. fonction objectif (4.1)). De plus, dans certains cas, il est possible de déterminer des poids de façon à ce que la résolution du problème avec un seul objectif, génère une solution qui aurait pu être générée par la méthode séquentielle. Un tel ensemble de poids est appelé *ensemble de poids équivalents* [100].

Dans [100], deux algorithmes pour calculer un ensemble de poids équivalents sont proposés. Ils sont basés sur le calcul d'une borne supérieure sur l'ensemble des valeurs que peut prendre une fonction linéaire. Nous présentons ici une technique alternative basée sur une approche similaire et montrons comment cette technique permet d'obtenir des poids de plus petite valeur que ceux générés par les deux algorithmes mentionnés plus haut.

4.1.2.1 Résultats généraux

Considérons le problème multicritères (PM) suivant :

$$\begin{aligned} \min\ f^T(x) &= (f_1(x),\ldots,f_\Upsilon(x))^T \\ \textbf{Sujet à}\quad x &\in X \end{aligned}$$

où, $f_1(x),\ldots,f_\Upsilon(x)$ sont les objectifs. Supposons que $f_1(x) \prec f_2(x) \prec \ldots \prec f_\Upsilon(x)$ où, la notation $f_\nu(x) \prec f_q(x)$ indique que $f_q(x)$ a une plus grande priorité que $f_\nu(x)$. Supposons également que X est discret et que $f_\nu(x)$ est entier $\forall x \in X, \forall\, 1 \leq \nu \leq \Upsilon$.
Soit le problème (PMP) associé à (PM), obtenu en considérant un seul objectif qui s'exprime comme une combinaison linéaire des objectifs de (PM) :

$$\begin{aligned} \min\ f(x) &= \sum_{\nu=1}^{\Upsilon} w_\nu f_\nu(x) \\ \textbf{Sujet à}\quad x &\in X. \end{aligned}$$

Dénotons par $L_\nu = \min_{x\in X} f_\nu(x)$ (respectivement $U_\nu = \max_{x\in X} f_\nu(x)$), la plus petite valeur (respectivement la plus grande valeur) que peut prendre $f_\nu(x)$ lorsque $x \in X$.

Posons :

$$w_q = \begin{cases} 1 & \text{si } q = 1 \\ 1 + \sum_{v=1}^{q-1} w_v(U_v - L_v) & \text{si } 2 \leq q \leq \Upsilon. \end{cases} \tag{4.15}$$

Démontrons maintenant que les poids, tels que définis en (4.15), sont équivalents pour le problème (PMP) dans le sens où une solution optimale de (PMP) est une solution de (PM) qui aurait pu être engendrée par la méthode séquentielle.

Le résultat suivant démontre que les poids définis par (4.15) garantissent que nous ne pouvons pas améliorer les objectifs les moins prioritaires aux dépens d'un objectif plus prioritaire. Notons que la preuve du résultat s'inspire de celle de SHERALI [100].

Propriété 4.1. *Soient x^1 et x^2 deux solutions réalisables du problème (PMP) où l'ensemble des poids $\{w_1, \ldots, w_\Upsilon\}$ est défini par les équations (4.15).*
Dénotons par :

$$\Theta = \{\theta \quad : \quad f_\theta(x^1) < f_\theta(x^2)\}$$

et soit $\overline{\theta} \in \Theta$ tel que $\overline{\theta} = \max_{\theta \in \Theta} \theta$.
Si $f(x^1) < f(x^2)$, alors

$$\Theta \neq \emptyset \tag{4.16}$$

$$f_v(x^1) = f_v(x^2) \qquad \overline{\theta} + 1 \leq v \leq \Upsilon. \tag{4.17}$$

Preuve. Procédons par contradiction.

Supposons que $\Theta = \emptyset$. Alors, $f_v(x^1) \geq f_v(x^2)$ pour tout $1 \leq v \leq \Upsilon$.

Mais alors, $\sum_{v=1}^{\Upsilon} w_v f_v(x^1) \geq \sum_{v=1}^{\Upsilon} w_v f_v(x^2)$ contredisant l'hypothèse que $f(x^1) < f(x^2)$.

Supposons maintenant que (4.17) n'est pas vérifiée, i.e., qu'il existe un indice q, $\overline{\theta} + 1 \leq q \leq \Upsilon$, tel que $f_q(x^1) \neq f_q(x^2)$. Puisque $\overline{\theta} = \max\{\theta \quad : \quad f_\theta(x^1) < f_\theta(x^2\}$, alors nécessairement $f_q(x^1) > f_q(x^2)$, et par conséquent,

$$f_q(x^1) \geq f_q(x^2) + 1 \tag{4.18}$$

car $f_q(x)$ prend des valeurs entières.

Par ailleurs, par définition de Θ, nous avons :

$$f_\nu(x^1) \geq f_\nu(x^2) \qquad \forall \nu \notin \Theta$$

et donc, en particulier :

$$f_\nu(x^1) \geq f_\nu(x^2) \qquad \forall \nu = q+1, \ldots, \Upsilon. \tag{4.19}$$

Évaluons $f(x^2) - f(x^1)$.

$$\begin{aligned} f(x^2) - f(x^1) &= \sum_{\nu=1}^{\Upsilon} w_\nu (f_\nu(x^2) - f_\nu(x^1)) \\ &= \sum_{\nu=1}^{q-1} w_\nu (f_\nu(x^2) - f_\nu(x^1)) + w_q (f_q(x^2) - f_q(x^1)) + \sum_{\nu=q+1}^{\Upsilon} w_\nu (f_\nu(x^2) - f_\nu(x^1)) \\ &\leq \sum_{\nu=1}^{q-1} w_\nu (f_\nu(x^2) - f_\nu(x^1)) - w_q \qquad \text{d'après (4.18) et (4.19).} \end{aligned}$$

Or, pour tout $\nu = 1, \ldots, \Upsilon$ et pour tout $x \in X$ nous avons, $L_\nu \leq f_\nu(x) \leq U_\nu$. Il s'ensuit alors que :

$$\begin{aligned} f(x^2) - f(x^1) &\leq \sum_{\nu=1}^{q-1} w_\nu (U_\nu - L_\nu) - w_q \\ &= -1 < 0 \qquad \text{d'après (4.15).} \end{aligned}$$

Donc, $f(x^2) < f(x^1)$, une contradiction. ∎

Nous pouvons ainsi déduire facilement le résultat suivant.

Corollaire 4.1. *Soient x^1 et x^2 deux solutions réalisables du problème (PMP) où l'ensemble des poids $\{w_1, \ldots, w_\Upsilon\}$ est défini par les équations (4.15).*
Si $f(x^1) < f(x^2)$, alors $f_\Upsilon(x^1) \leq f_\Upsilon(x^2)$.

Preuve. Le résultat découle immédiatemment de la propriété 4.1. En effet, si $\overline{\theta} = \Upsilon$, alors $f_\Upsilon(x^1) < f_\Upsilon(x^2)$ puisque $\overline{\theta} \in \Theta = \{\theta \quad : \quad f_\theta(x^1) < f_\theta(x^2)\}$. Sinon (i.e., si $\overline{\theta} < \Upsilon$), alors $f_\Upsilon(x^1) = f_\Upsilon(x^2)$, ce qui complète la preuve. ∎

Inversement, nous montrons que si une solution améliore les objectifs les plus prioritaires, alors elle améliore nécessairement la valeur de la fonction objectif du problème (PMP).

Propriété 4.2. *Soient x^1 et x^2 deux solutions réalisables du problème (PMP) où l'ensemble des poids $\{w_1,\ldots,w_\Upsilon\}$ est défini par les équations (4.15). Supposons qu'il existe un indice q, $1 \leq q \leq \Upsilon$ tel que :*

$$f_q(x^1) < f_q(x^2) \tag{4.20}$$

$$f_v(x^1) = f_v(x^2) \qquad q+1 \leq v \leq \Upsilon. \tag{4.21}$$

Alors, $f(x^1) < f(x^2)$.

Preuve. La preuve est similaire à celle de la propriété 4.1 et utilise les mêmes arguments. En effet,

$$\begin{aligned}
f(x^1)-f(x^2) &= \sum_{v=1}^{\Upsilon} w_v(f_v(x^1)-f_v(x^2)) \\
&= \sum_{v=1}^{q-1} w_v(f_v(x^1)-f_v(x^2)) + w_q(f_q(x^1)-f_q(x^2)) + \sum_{v=q+1}^{\Upsilon} w_v(f_v(x^1)-f_v(x^2)) \\
&\leq \sum_{v=1}^{q-1} w_v(f_v(x^1)-f_v(x^2)) - w_q \qquad \text{d'après (4.20) et (4.21)} \\
&\leq \sum_{v=1}^{q-1} w_v(U_v-L_v) - w_q \\
&= -1 < 0 \qquad \text{d'après (4.15).}
\end{aligned}$$

Donc, $f(x^1) < f(x^2)$. ■

Le résultat suivant est une conséquence directe de la propriété 4.2.

Corollaire 4.2. *Soient x^1 et x^2 deux solutions réalisables du problème (PMP) où l'ensemble des poids $\{w_1,\ldots,w_\Upsilon\}$ est défini par les équations (4.15).*
Si $f_\Upsilon(x^1) < f_\Upsilon(x^2)$, alors $f(x^1) < f(x^2)$.

Nous pouvons aussi déduire facilement le résultat suivant.

Corollaire 4.3. *Considérons le problème multicritères (PM) où l'ordre de priorité des objectifs est défini comme suit :*

$$f_1(x) \prec f_2(x) \prec \ldots \prec f_\Upsilon(x).$$

Alors, l'ensemble des poids $\{w_1, \ldots, w_\Upsilon\}$ *défini par les équations (4.15) est un ensemble de poids équivalents pour le problème (PMP).*

Preuve. Il suffit de montrer qu'une solution optimale de (PMP) est une solution de (PM) qui aurait pu être engendrée par la méthode séquentielle.
Soit $x^* \in X$ une solution de (PM) générée par la méthode séquentielle. Considérons une autre solution $\bar{x} \in X$ qui ne peut être engendrée par la méthode séquentielle. Alors, il existe un indice q, $1 \leq q \leq \Upsilon$, tel que :

$$\begin{aligned} f_q(x^*) &< f_q(\bar{x}) \\ f_\nu(x^*) &= f_\nu(\bar{x}) \qquad\qquad q+1 \leq \nu \leq \Upsilon. \end{aligned}$$

Se référant à la propriété 4.2, nous déduisons que $f(x^*) < f(\bar{x})$, ce qui complète la preuve. ■

Considérons maintenant le cas particulier où les objectifs $f_1(x), \ldots, f_\Upsilon(x)$ sont linéaires (i.e., pour tout $\nu = 1, \ldots, \Upsilon$, $f_\nu(x) = c_\nu^T x$ où, c_ν est un vecteur d'entiers de taille n), et les variables x sont bornées (i.e., $X = \{x \text{ entier} : Ax = b \text{ et } 0 \leq x \leq u\}$ où la matrice A et le vecteur b possèdent m lignes, et u est un vecteur fini de taille n). Notons par (PML) le problème multicritères correspondant :

$$\begin{aligned} \min f^T(x) &= (c_1^T x, \ldots, c_\Upsilon^T x)^T \\ \textbf{Sujet à} \quad x &\in X \end{aligned}$$

où l'ordre de priorité des objectifs est défini comme suit : $c_1^T x \prec c_2^T x \prec \ldots \prec c_\Upsilon^T x$.

Soit (PMLP) le problème associé à (PML) obtenu en considérant un seul objectif :

$$\begin{aligned} \min f(x) &= \sum_{v=1}^{\Upsilon} w_v c_v^T x \\ \textbf{Sujet à} \quad x &\in X. \end{aligned}$$

Il est clair que le résultat du corollaire 4.3 reste valide dans ce cas, i.e., l'ensemble des poids définis par les équations (4.15) est un ensemble de poids équivalents pour le problème (PMLP). Deux autres ensembles de poids équivalents sont définis dans [100]. Nous montrons dans ce qui suit que, dans certains cas, les poids que nous proposons ont une valeur plus petite que ceux présentés dans [100] et qu'ils leur sont donc préférables. Introduisons d'abord quelques notations utiles. Dénotons par :

- $UB(c_v) = \sum_{j=1}^{n} u_j |c_{vj}|$ une borne supérieure sur les valeurs que peut prendre la fonction linéaire $f_v(x) = c_v^T x$ lorsque $x \in X$;
- $\gamma = \max\{UB(c_v),\ v = 1,\ldots,\Upsilon\}$;
- $M = \gamma + 1$.

Les trois résultats suivants sont dus à SHERALI [100].

Lemme 1 ([100]). *Si les poids* $\{w_1,\ldots,w_\Upsilon\}$ *sont définis par l'ensemble des équations suivant :*

$$w_v = M^{v-1} \qquad 1 \le v \le \Upsilon \tag{4.22}$$

alors, ils forment un ensemble de poids équivalents pour le problème (PMLP).

Lemme 2 ([100]). *Si*

$$w_q = \begin{cases} 1 & si\ q = 1 \\ 1 + UB\left(\sum_{v=1}^{q-1} w_v c_v\right) & si\ 2 \le q \le \Upsilon \end{cases} \tag{4.23}$$

alors, l'ensemble $\{w_1,\ldots,w_\Upsilon\}$ *est un ensemble de poids équivalents pour le problème (PMLP).*

Lemme 3 ([100]). *Soient* $\{\alpha_1,\ldots,\alpha_\Upsilon\}$ *et* $\{\beta_1,\ldots,\beta_\Upsilon\}$ *deux ensembles de poids générés*

en utilisant les équations (4.22) et (4.23) respectivement. Alors, pour tout $\nu = 1, \ldots, \Upsilon$, $\beta_\nu \leq \alpha_\nu$.

Dans plusieurs applications, notamment dans le cas des problèmes de programmation à buts multiples (*goal programming*), les fonctions objectifs $f_\nu(x) = c_\nu^T x$ prennent des valeurs non négatives. Le résultat suivant montre que, dans de tels contextes, il est préférable d'utiliser les poids générés par les équations (4.15) puisque leurs valeurs sont plus petites que celles obtenues via les équations (4.23). En effet, lorsque les coefficients de la fonction objectif sont très grands par rapport aux autres coefficients du problème, si on résout le problème avec l'algorithme du simplexe par exemple, des erreurs de précision peuvent survenir lors de l'étape du pivotage. Ces erreurs s'amplifient au cours du calcul ce qui peut provoquer une erreur dans la solution obtenue.

Propriété 4.3. *Soient* $\{\lambda_1, \ldots, \lambda_\Upsilon\}$ *et* $\{\beta_1, \ldots, \beta_\Upsilon\}$ *deux ensembles de poids générés en utilisant les équations (4.15) et (4.23) respectivement.*
Si $c_\nu \geq 0$, *alors* $\lambda_\nu \leq \beta_\nu$ *pour tout* $\nu = 1, \ldots, \Upsilon$.

Preuve. Nous prouvons ce résultat par récurrence.
Se référant à (4.15) et (4.23), nous avons $\lambda_1 = \beta_1 = 1$.
Soit un indice q, $2 \leq q \leq \Upsilon - 1$. Supposons que $\lambda_\nu \leq \beta_\nu$ pour tout $\nu \leq q$. Montrons que cette affirmation est vraie pour $q+1$, i.e., $\lambda_{q+1} \leq \beta_{q+1}$.
L'ensemble des poids $\{\beta_1, \ldots, \beta_\Upsilon\}$ étant généré en utilisant les équations (4.23), nous avons :

$$\begin{aligned} \beta_{q+1} &= 1 + UB\left(\sum_{\nu=1}^{q} \beta_\nu c_\nu\right) \\ &= 1 + \sum_{j=1}^{n} u_j \left|\left(\sum_{\nu=1}^{q} \beta_\nu c_\nu\right)_j\right|. \end{aligned}$$

Puisque pour tout $\nu = 1, \ldots, \Upsilon$, nous avons $\beta_\nu > 0$ et $c_\nu \geq 0$, alors :

$$\begin{aligned}
\beta_{q+1} &= 1 + \sum_{j=1}^{n} u_j (\sum_{\nu=1}^{q} \beta_\nu c_\nu)_j \\
&= 1 + \sum_{j=1}^{n} u_j \sum_{\nu=1}^{q} \beta_\nu c_{\nu j} \\
&= 1 + \sum_{\nu=1}^{q} \beta_\nu \sum_{j=1}^{n} u_j c_{\nu j} \\
&= 1 + \sum_{\nu=1}^{q} \beta_\nu c_\nu^T u.
\end{aligned}$$

Notons que pour tout $x \in X$ et pour tout $\nu = 1, \ldots, \Upsilon$, nous avons $c_\nu \geq 0$ et $0 \leq x \leq u$. Donc, $c_\nu^T u \geq c_\nu^T x \geq 0 \quad \forall x \in X$. En particulier,

$$c_\nu^T u \geq \max_{x \in X} c_\nu^T x = U_\nu$$

et,

$$\min_{x \in X} c_\nu^T x = L_\nu \geq 0.$$

Par conséquent,

$$c_\nu^T u \geq U_\nu \geq U_\nu - L_\nu.$$

Maintenant, utilisant l'hypothèse de récurrence ($\beta_\nu \geq \lambda_\nu \quad \forall \nu \leq q$), nous obtenons :

$$\beta_\nu c_\nu^T u \geq \lambda_\nu (U_\nu - L_\nu) \qquad \forall \nu \leq q.$$

Ainsi,

$$\begin{aligned}
\beta_{q+1} &= 1 + \sum_{\nu=1}^{q} \beta_\nu c_\nu^T u \\
&\geq 1 + \sum_{\nu=1}^{q} \lambda_\nu (U_\nu - L_\nu) = \lambda_{q+1} \qquad \text{d'après (4.15)}
\end{aligned}$$

ce qui complète la preuve. ■

4.1.2.2 Application au problème sur plusieurs rondes

Nous utilisons les équations (4.15) pour déterminer les valeurs des poids w_v associés aux différentes règles souples d'affectation ($1 \leq v \leq 6$). Cela nécessite de déterminer les bornes inférieure ($L_v = \min_{x \in X} f_v(x)$) et supérieure ($U_v = \max_{x \in X} f_v(x)$) sur la valeur de chacun des cinq premiers termes $f_v(x)$ de la fonction objectif (4.1) (cf. modèle MG^1 introduit au paragraphe 4.1.1). Rappelons que chacun de ces termes correspond à une règle souple d'affectation et s'exprime comme la somme des variables d'écart associées à cette règle.

Avant de discuter des résultats permettant d'identifier ces bornes, il convient de souligner que l'ensemble des solutions réalisables X, tel que défini par les contraintes du modèle MG^1, contient une infinité de solutions qu'il est impossible d'énumérer, et donc, si l'on considère uniquement ces contraintes, il serait impossible d'évaluer les bornes sur les valeurs des $f_v(x)$, particulièrement les bornes supérieures U_v. Pour illustrer notre propos, considérons les variables d'écart d_{jp}^{1+} et d_{jp}^{1-} associées à la contrainte d'*équilibre*. À l'optimalité, leur différence représente certes l'écart entre le nombre de juges *experts* et *nouveaux* affectés au match j de la ronde p ; mais, puisqu'il est possible de leur assigner une infinité de valeurs sans violer les contraintes (4.7) et (4.14) du modèle MG^1, il est possible d'obtenir une infinité de solutions réalisables de sorte que $\max_{x \in X} \sum_{p=1}^{P} \sum_{j=1}^{M} (d_{jp}^{1+} + d_{jp}^{1-})$ n'est pas borné supérieurement.
Il s'ensuit que l'ensemble des solutions réalisables, tel que défini par les contraintes du modèle MG^1, comprend plusieurs solutions qui ne présentent aucun intérêt puisqu'elles ne peuvent en aucun cas être optimales. Nous pouvons donc les éliminer du domaine réalisable sans que cela n'entraîne une modification de la solution optimale du problème. Pour y arriver, il suffit d'ajouter au modèle MG^1 des contraintes supplémentaires qui forcent les variables d'écart à se comporter comme la minimisation les force à le faire. Ces contraintes supplémentaires sont en quelque sorte redondantes lorsque nous considérons la fonction objectif. Nous allons donc considérer ces contraintes, en plus de celles

du modèle MG^1, pour évaluer les bornes supérieures sur les $f_v(x)$ en tirant parti de la structure du problème.

Voici donc les nouvelles contraintes assurant à la fois que les contraintes (4.7) à (4.11) du modèle MG^1 sont satisfaites, et que les valeurs des variables d'écart sont conformes à celles induites par la minimisation de la fonction objectif :

1. Associons à chaque paire (j,p) une variable binaire z^1_{jp}. Nous pouvons alors remplacer les contraintes (4.7) par l'ensemble des contraintes suivantes :

$$\sum_{i=1}^{N}(2l_i+2v_i-1)x_{ijp}-5z^1_{jp} \leq 0 \qquad \forall j,p \qquad (4.24)$$

$$-\sum_{i=1}^{N}(2l_i+2v_i-1)x_{ijp}+z^1_{jp} \leq 1 \qquad \forall j,p \qquad (4.25)$$

$$-\sum_{i=1}^{N}(2l_i+2v_i-1)x_{ijp}+2z^1_{jp}+d^{1+}_{jp} = 1 \qquad \forall j,p \qquad (4.26)$$

$$2z^1_{jp}+d^{1-}_{jp} = 2 \qquad \forall j,p. \qquad (4.27)$$

2. Associons à chaque triplet (c,j,p) une variable binaire z^2_{cjp}. Nous pouvons alors remplacer les contraintes (4.8) par l'ensemble des contraintes suivantes :

$$\sum_{i=1}^{N} r_{ic}x_{ijp}-5z^2_{cjp} \leq 0 \qquad \forall c,j,p \qquad (4.28)$$

$$\sum_{i=1}^{N} r_{ic}x_{ijp}-z^2_{cjp}-d^2_{cjp} = 0 \qquad \forall c,j,p. \qquad (4.29)$$

3. Associons à chaque triplet (k,j,p) une variable binaire z^3_{kjp}. Nous pouvons alors remplacer les contraintes (4.9) par l'ensemble des contraintes suivantes :

$$\sum_{i=1}^{N} e_{ik}x_{ijp}-5z^3_{kjp} \leq 0 \qquad \forall k,j,p \qquad (4.30)$$

$$-\sum_{i=1}^{N} e_{ik}x_{ijp}+z^3_{kjp} \leq 0 \qquad \forall k,j,p \qquad (4.31)$$

$$z^3_{kjp}+d^3_{kjp} = 1 \qquad \forall k,j,p. \qquad (4.32)$$

4. Associons à chaque paire (i,r) une variable binaire z^4_{ir}. Nous pouvons alors remplacer les contraintes (4.10) par l'ensemble des contraintes suivantes :

$$\sum_{p=1}^{P}\sum_{j=1}^{M} x_{ijp}x_{rjp} - Pz^4_{ir} \leq 0 \qquad \forall i,r>i \qquad (4.33)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M} x_{ijp}x_{rjp} - z^4_{ir} - d^4_{ir} = 0 \qquad \forall i,r>i. \qquad (4.34)$$

5. D'une manière analogue, associons à chaque paire (i,t) une variable binaire z^5_{it}. Nous pouvons alors remplacer les contraintes (4.11) par l'ensemble des contraintes suivantes :

$$\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} - Pz^5_{it} \leq 0 \qquad \forall i,t \qquad (4.35)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} - z^5_{it} - d^5_{it} = 0 \qquad \forall i,t. \qquad (4.36)$$

Ainsi, nous obtenons un nouveau modèle MGR[1] qui est équivalent au modèle original MG[1], car la fonction objectif (4.1) reste inchangée, et le domaine réalisable est induit par les contraintes (4.2) à (4.6), (4.24) à (4.36), et (4.12) à (4.14) qui correspondent à celles de MG[1] auxquelles nous ajoutons des contraintes redondantes relativement à l'objectif. L'avantage de considérer MGR[1] dans l'analyse qui suit pour déterminer les poids vient du fait que son domaine réalisable est borné.

Déterminons maintenant pour chaque règle d'affectation de priorité v, $1 \leq v \leq 5$, la plus petite valeur ($L_v = \min_{x\in X} f_v(x)$) et la plus grande valeur ($U_v = \max_{x\in X} f_v(x)$) que peut prendre le terme $f_v(x)$ de la fonction objectif (4.1) qui lui est associé (X étant défini par les contraintes du modèle MGR[1]). Ces résultats font l'objet des lemmes 4 à 9 qui suivent.

Lemme 4. *Pour tout* v, $1 \leq v \leq 5$, $L_v = 0$.

Preuve. Le résultat découle directement du fait que les différents objectifs (termes de la fonction objectif (4.1)) reflètent le nombre de violations des différentes règles d'af-

fectation. Ainsi, si la règle d'affectation de priorité v est satisfaite dans tous les matchs de toutes les rondes, alors le terme de la fonction objectif (4.1) qui lui est associé vaut 0. ■

Lemme 5. *Pour toute solution réalisable du modèle* MGR^1,

$$\sum_{p=1}^{P}\sum_{j=1}^{M}(d_{jp}^{1+}+d_{jp}^{1-}) \leq 4PM.$$

Preuve. Rappelons que les variables d'écart d_{jp}^{1+} et d_{jp}^{1-} servent à modéliser la contrainte d'*équilibre* stipulant que, dans chaque match de chaque ronde, hormis le *juge en chef*, nous devons avoir le même nombre de juges *experts* et de juges *nouveaux*.
Pour faciliter l'écriture, dénotons par $L_{jp}(x) = \sum_{i=1}^{N} l_i x_{ijp}$ le nombre de *juges en chef* affectés au match j de la ronde p, et par $E_{jp}(x) = \sum_{i=1}^{N} v_i x_{ijp}$ le nombre de juges de la catégorie *Expert* affectés à ce match (ceux qui sont *experts* mais qui ne sont pas des *juges en chef*). Soit $N_{jp}(x) = \sum_{i=1}^{N}(1-l_i-v_i)x_{ijp}$ le nombre de juges *nouveaux* affectés à ce match. Le terme $\sum_{i=1}^{N}(2l_i+2v_i-1)x_{ijp}$ peut donc se réécrire sous la forme : $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)$. Notons que la valeur de ce terme ne peut en aucun cas dépasser 5 puisque :

$$\begin{aligned} L_{jp}(x)+E_{jp}(x)-N_{jp}(x) &\leq L_{jp}(x)+E_{jp}(x)+N_{jp}(x) \\ &= \sum_{i=1}^{N} x_{ijp} \\ &\leq 5 \qquad \text{d'après (4.5), (4.6) et (4.13).} \end{aligned}$$

Discutons les différents cas selon la valeur de $(L_{jp}(x)+E_{jp}(x)-N_{jp}(x))$ et déterminons, dans chaque cas, une borne supérieure sur la valeur de $(d_{jp}^{1+}+d_{jp}^{1-})$.

Cas 1. Supposons que $L_{jp}(x)+E_{jp}(x)-N_{jp}(x) > 1$. Alors, les contraintes (4.24), et le fait que z_{jp}^{1} soit binaire assurent que $z_{jp}^{1}=1$. Il s'ensuit donc de (4.27) que $d_{jp}^{1-}=0$.

Par ailleurs, se référant aux contraintes (4.26), nous avons :

$$\begin{aligned} d_{jp}^{1+} &= L_{jp}(x) + E_{jp}(x) - N_{jp}(x) - 1 \\ &= L_{jp}(x) + E_{jp}(x) + N_{jp}(x) - 2N_{jp}(x) - 1 \\ &\leq L_{jp}(x) + E_{jp}(x) + N_{jp}(x) - 1. \end{aligned}$$

Or, $L_{jp}(x) + E_{jp}(x) + N_{jp}(x) = \sum_{i=1}^{N} x_{ijp}$ n'est rien d'autre que le nombre total de juges affectés au match j de la ronde p. Se référant aux contraintes (4.5), (4.6) et (4.13), nous avons :

$$\begin{aligned} d_{jp}^{1+} &\leq L_{jp}(x) + E_{jp}(x) + N_{jp}(x) - 1 \\ &= \sum_{i=1}^{N} x_{ijp} - 1 \\ &\leq 5 - 1 = 4. \end{aligned}$$

Ainsi, si $L_{jp}(x) + E_{jp}(x) - N_{jp}(x) > 1$, alors

$$d_{jp}^{1+} + d_{jp}^{1-} \leq 4. \tag{4.37}$$

Cas 2. Supposons maintenant que $L_{jp}(x) + E_{jp}(x) - N_{jp}(x) < 1$.

Montrons d'abord que $L_{jp}(x) + E_{jp}(x) \geq 2$ en procédant par absurde.

Supposons que $L_{jp}(x) + E_{jp}(x) < 2$. Considérons les deux cas de figure suivants. Si $E_{jp}(x) = 0$, alors la contradiction se déduit facilement de (4.4) puisque, dans ce cas, nous avons $6 \leq 5L_{jp}(x) + E_{jp}(x) = 5L_{jp}(x)$, ce qui implique que $L_{jp}(x) \geq 2$. Considérons maintenant le cas où $E_{jp}(x) \geq 1$. Alors,

$$\begin{aligned} 5L_{jp}(x) + E_{jp}(x) &= 5(L_{jp}(x) + E_{jp}(x)) - 4E_{jp}(x) \\ &< 10 - 4 = 6 \end{aligned}$$

contredisant les contraintes (4.4).

Nous montrons maintenant que si $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)<1$, alors nécessairement $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)=-1$.
En effet, si $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)<-1$, alors $2\leq L_{jp}(x)+E_{jp}(x)<N_{jp}(x)-1$, et donc,

$$\begin{aligned}\sum_{i=1}^{N} x_{ijp} &= L_{jp}(x)+E_{jp}(x)+N_{jp}(x)\\ &> 5\end{aligned}$$

contredisant les contraintes (4.5), (4.6) et (4.13).

Supposons maintenant que $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)>-1$. Il est clair que ceci implique que $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)=0$ puisque nous avons également supposé que $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)<1$. Il s'ensuit donc que :

$$\begin{aligned}\sum_{i=1}^{N} x_{ijp} &= L_{jp}(x)+E_{jp}(x)+N_{jp}(x)\\ &= 2N_{jp}(x).\end{aligned}$$

La contradiction se déduit encore une fois de (4.5), (4.6) et (4.13) puisque ces contraintes impliquent que $\sum_{i=1}^{N} x_{ijp}$ est un nombre impair (égal à 3 ou 5).

Maintenant, par (4.25), par le fait que z_{jp}^{1} soit binaire, et par le résultat que nous venons de prouver ($L_{jp}(x)+E_{jp}(x)-N_{jp}(x)=-1$), nous déduisons que $z_{jp}^{1}=0$. Il s'ensuit donc de (4.27) que $d_{jp}^{1-}=2$. Finalement, se référant aux contraintes (4.26), nous obtenons que $d_{jp}^{1+}=0$.
Ainsi, si $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)<1$, alors

$$d_{jp}^{1+}+d_{jp}^{1-}=2. \tag{4.38}$$

Cas 3. Considérons le cas où $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)=1$. Utilisant (4.24) et le fait que z_{jp}^{1} soit binaire, nous obtenons que $z_{jp}^{1}=1$, et par conséquent, $d_{jp}^{1+}=d_{jp}^{1-}=0$ par les contraintes (4.26) et (4.27).

Ainsi, si $L_{jp}(x)+E_{jp}(x)-N_{jp}(x)=1$, alors

$$d_{jp}^{1+}+d_{jp}^{1-}=0. \tag{4.39}$$

Finalement de (4.37), (4.38) et (4.39), nous concluons que pour tout $j=1,\ldots,M$ et pour tout $p=1,\ldots,P$, $d_{jp}^{1+}+d_{jp}^{1-}\leq \max(4,2,0)=4$, et par conséquent,

$$\begin{aligned}\sum_{p=1}^{P}\sum_{j=1}^{M}(d_{jp}^{1+}+d_{jp}^{1-}) &\leq \sum_{p=1}^{P}\sum_{j=1}^{M}4\\ &= 4PM\end{aligned}$$

ce qui complète la preuve. ■

Lemme 6. *Pour toute solution réalisable du modèle MGR*[1]*,*

$$\sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{c=1}^{C}d_{cjp}^{2}\leq 4PM.$$

Preuve. Rappelons que les variables d'écart d_{cjp}^{2} reflètent le nombre de violations de la contrainte d'*affiliation* spécifiant que si plusieurs juges représentant des compagnies sont affectés à un match, alors ils doivent être affiliés à des compagnies différentes. En effet, rappelons que $r_{ic}=1$ si le juge i représente la compagnie c et 0 sinon. $\sum_{i=1}^{N} r_{ic}x_{ijp}$ n'est donc rien d'autre que le nombre de juges affectés au match j de la ronde p représentant la compagnie c. Notons que la valeur de ce terme ne peut en aucun cas dépasser 5 puisque :

$$\begin{aligned}\sum_{i=1}^{N} r_{ic}x_{ijp} &\leq \sum_{i=1}^{N} x_{ijp}\\ &\leq 5 \qquad\qquad \text{d'après (4.5), (4.6) et (4.13).}\end{aligned}$$

Considérons les trois cas de figure suivants :

Cas 1. Si $\sum_{i=1}^{N} r_{ic}x_{ijp} = 0$, alors se référant aux contraintes (4.29), $z^2_{cjp} + d^2_{cjp} = 0$, et donc, $z^2_{cjp} = d^2_{cjp} = 0$ puisque $z^2_{cjp}, d^2_{cjp} \geq 0$.

Cas 2. Si $\sum_{i=1}^{N} r_{ic}x_{ijp} = 1$, alors se référant aux contraintes (4.28), et puisque z^2_{cjp} est binaire, nous déduisons que $z^2_{cjp} = 1$. Il s'ensuit donc de (4.29) que $d^2_{cjp} = 0$.

Cas 3. Si $\sum_{i=1}^{N} r_{ic}x_{ijp} > 1$, alors les contraintes (4.28), et le fait que z^2_{cjp} soit binaire assurent que $z^2_{cjp} = 1$. Se référant aux contraintes (4.29), nous déduisons que $d^2_{cjp} = \sum_{i=1}^{N} r_{ic}x_{ijp} - 1 > 0$.

Il est facile de voir que pour toutes valeurs de c, j et p, la plus grande valeur que peut prendre d^2_{cjp} est égale à 4 lorsque les 5 juges affectés au match j de la ronde p sont de la même compagnie (i.e., $\sum_{i=1}^{N} r_{ic}x_{ijp} = 5$). Or, comme ceci ne peut se produire que pour une seule compagnie, il s'ensuit que $\sum_{c=1}^{C} d^2_{cjp} \leq 4$. Par conséquent, $\sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{c=1}^{C} d^2_{cjp} \leq 4PM$. ■

Lemme 7. *Pour toute solution réalisable du modèle MGR*[1],

$$\sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{k=1}^{K} d^3_{kjp} \leq (K-1)PM.$$

Preuve. Rappelons que les variables d'écart d^3_{kjp} reflètent le nombre de violations de la contrainte de *diversité* requérant que les expertises des juges affectés à un même match couvrent le plus grand nombre de domaines d'expertise.

En effet, rappelons que $e_{ik} = 1$ si le juge i possède le domaine d'expertise k et 0 sinon. $\sum_{i=1}^{N} e_{ik}x_{ijp}$ n'est donc rien d'autre que le nombre de juges affectés au match j de la ronde p ayant le domaine d'expertise k. Notons que la valeur de ce terme ne peut en aucun cas dépasser 5 puisque :

$$\begin{aligned} \sum_{i=1}^{N} e_{ik}x_{ijp} &\leq \sum_{i=1}^{N} x_{ijp} \\ &\leq 5 \qquad \text{d'après (4.5), (4.6) et (4.13).} \end{aligned}$$

Considérons les deux cas de figure suivants :

Cas 1. Si $\sum_{i=1}^{N} e_{ik}x_{ijp} = 0$, alors les contraintes (4.31) assurent que $z^3_{kjp} = 0$, et donc, $d^3_{kjp} = 1$ par les contraintes (4.32).

Cas 2. Si $\sum_{i=1}^{N} e_{ik}x_{ijp} \geq 1$, alors se référant aux contraintes (4.30), et puisque z^3_{kjp} est binaire, nous déduisons que $z^3_{kjp} = 1$. Il s'ensuit donc de (4.32) que $d^3_{kjp} = 0$.

Ainsi, pour toutes valeurs de k, j et p, nous avons $d^3_{kjp} \leq 1$.

Par ailleurs, puisque chaque juge possède au moins un domaine d'expertise, alors il existe au moins un indice $k' \in \{1,\ldots,K\}$ tel que $d^3_{k'jp} = 0$.

Donc, $\sum_{k=1}^{K} d^3_{kjp} = \sum_{\substack{k=1 \\ k\neq k'}}^{K} d^3_{kjp} \leq K-1$, et par conséquent, $\sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{k=1}^{K} d^3_{kjp} \leq (K-1)PM$. ■

Lemme 8. *Pour toute solution réalisable du modèle MGR*[1],

$$\sum_{i=1}^{N-1}\sum_{r=i+1}^{N} d^4_{ir} \leq 10(P-1)M.$$

Preuve. Rappelons que les variables d'écart d^4_{ir} reflètent le nombre de violations de la contrainte de *couplage* stipulant qu'une paire spécifique de juges ne peut être affectée plus d'une fois au cours de l'ensemble des rondes.

Le résultat du lemme se déduit facilement en observant que $\sum_{j=1}^{M} x_{ijp}x_{rjp} \leq 1$, car un juge ne peut être affecté à plus d'un match de la même ronde d'après les contraintes (4.2), et donc, $\sum_{p=1}^{P}\sum_{j=1}^{M} x_{ijp}x_{rjp} \leq P$. Ainsi, par une démarche analogue à celle que nous avons utilisée dans la preuve du lemme 6, il est facile de voir que les contraintes (4.33) et (4.34) assurent que $d^4_{ir} \leq P-1$, et que $d^4_{ir} = P-1$ si et seulement si la paire de juges i et r est affectée à un match de chaque ronde.

Maintenant, dénotons par Λ l'ensemble des paires de juges i et r qui sont affectées à un match de chaque ronde ; ainsi, $\Lambda = \{(i,r) \quad : \quad d^4_{ir} = P-1\}$. Par les contraintes (4.5), (4.6) et (4.13), les juges affectés à un match représentent au plus 10 paires différentes, et donc, au plus $10PM$ paires de juges sont représentées au cours de l'ensemble des matchs de

toutes les rondes. Or, puisque chaque paire de juges $(i,r) \in \Lambda$ est nécessairement affectée à P matchs, il s'ensuit que $|\Lambda| \leq 10M$.

D'autre part, il est facile de vérifier que si $|\Lambda| = 10M$, alors $d^4_{ir} = 0 \quad \forall (i,r) \notin \Lambda$. En effet, supposons qu'il existe une paire de juges $(i',r') \notin \Lambda$ telle que $d^4_{i'r'} > 0$. Alors, se référant aux contraintes (4.34), $\sum_{p=1}^{M} \sum_{j=1}^{M} x_{i'jp} x_{r'jp} \geq 1$. Il existerait donc j' et p' tels que $x_{i'j'p'} = x_{r'j'p'} = 1$. Mais alors, le nombre de paires de juges affectées au match j' de la ronde p' dépasserait 10 (puisque $|\Lambda| = 10M$ implique que 10 paires de juges de Λ sont déjà affectées à chaque match) contredisant les contraintes (4.5), (4.6) et (4.13).

De ce qui précède, nous concluons que $\sum_{i=1}^{N-1} \sum_{r=i+1}^{N} d^4_{ir} \leq 10(P-1)M$. ■

Lemme 9. *Pour toute solution réalisable du modèle MGR*[1]*,*

$$\sum_{t=1}^{2M} \sum_{i=1}^{N} d^5_{it} \leq \frac{25}{4}(P-1)M.$$

Preuve. Rappelons que les variables d'écart d^5_{it} reflètent le nombre de violations de la contrainte de *séquence* spécifiant que, durant les différentes rondes, un juge ne doit pas être affecté à différents matchs impliquant la même équipe.

Rappelons également que les équipes sont partitionnées en groupes, et qu'à chaque ronde, chaque équipe rencontre une des autres équipes de son groupe. Ainsi, chaque groupe compte $P+1$ équipes (P étant le nombre de rondes), et $\frac{P+1}{2}$ matchs de chaque groupe ont lieu à chaque ronde.

Il est également clair que si un juge viole la contrainte de *séquence*, alors il est nécessairement affecté à au moins deux matchs du même groupe (puisque durant les différentes rondes les équipes ne rencontrent que les autres équipes de leur groupe).

Soit G un groupe donné. Supposons que nous disposons d'un ensemble J formé de $5\frac{P+1}{2}$ juges tels que, à chaque ronde, chaque juge de cet ensemble est affecté à un match du groupe G. Partitionnons J en $\frac{P+1}{2}$ ensembles disjoints dont chacun comporte 5 juges. Dénotons ces ensembles par $J_\tau, \quad \tau = 1, \ldots, \frac{P+1}{2}$.

Sans perte de généralité, supposons que les équipes de G sont indexées de 1 à $P+1$ et

que chaque juge $i \in J_1$ est affecté aux P matchs impliquant l'équipe $t = 1$. Rappelons que $s_{tjp} = 1$ si l'équipe t est en compétition dans le match j de la ronde p et 0 sinon. $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp}$ n'est donc rien d'autre que le nombre de fois que le juge i est affecté à un match de l'équipe t. Notons que la valeur de ce terme ne peut en aucun cas dépasser P car $\sum_{j=1}^{M} x_{ijp} \leq 1$ par les contraintes (4.2). Ainsi, l'hypothèse ci-dessus se traduit par $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{1jp} x_{ijp} = P \quad \forall i \in J_1$. Les contraintes (4.35) et (4.36) assurent respectivement que $z_{i1}^5 = 1$ pour tout $i \in J_1$, et que $d_{i1}^5 = P - 1$. Puisque l'ensemble J_1 comporte 5 juges, alors

$$\sum_{i \in J_1} d_{i1}^5 = 5(P-1). \tag{4.40}$$

Par ailleurs, puisqu'à chaque ronde l'équipe 1 rencontre une des autres équipes de son groupe, alors chaque juge $i \in J_1$ est affecté exactement une fois à un match impliquant les équipes $t = 2, \ldots, P+1$ (i.e., $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp} = 1$ pour tout $i \in J_1$ et $t = 2, \ldots, P+1$). Utilisant, encore une fois les contraintes (4.35) et (4.36), nous déduisons que, pour ces équipes, $d_{it}^5 = 0$, et donc,

$$\sum_{i \in J_1} d_{it}^5 = 0 \qquad \forall t = 2, \ldots, P+1. \tag{4.41}$$

Finalement, il est clair qu'aucun juge $i \in J_1$ n'est affecté à un match impliquant une équipe ne faisant pas partie de G (i.e., $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp} = 0 \quad \forall i \in J_1 , t \notin G$), car les matchs d'une ronde se déroulent simultanément et donc, un juge ne peut être affecté à plus de P matchs. Les contraintes (4.36) nous permettent alors de déduire que, pour tout $i \in J_1$ et pour ces équipes $t \notin G$ (ou encore pour tout $t > P+1$), $d_{it}^5 = 0$. Par conséquent,

$$\sum_{i \in J_1} d_{it}^5 = 0 \qquad \forall t > P+1. \tag{4.42}$$

De (4.40), (4.41) et (4.42), nous concluons que :

$$\sum_{t=1}^{2M} \sum_{i \in J_1} d_{it}^5 = 5(P-1).$$

Considérons maintenant l'ensemble des juges J_2. Il est clair qu'aucun de ces juges ne peut être affecté à un match impliquant l'équipe $t = 1$ (car les juges de J_1 sont affectés aux P matchs de cette équipe et $|J_1| = 5$). Ceci se traduit par $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{1jp} x_{ijp} = 0 \quad \forall i \in J_2$. Utilisant, les contraintes (4.36), nous déduisons que $d_{i1}^5 = 0 \quad \forall i \in J_2$, et donc,

$$\sum_{i \in J_2} d_{i1}^5 = 0. \tag{4.43}$$

Quant aux matchs impliquant l'équipe 2, les juges de J_2 peuvent être affectés à au plus $P-1$ de ces matchs. En effet, nous avons supposé que les juges formant J_1 sont affectés à tous les matchs impliquant l'équipe 1. Donc, en particulier, ils sont affectés au match dans lequel les équipes 1 et 2 sont en compétition. Supposons donc que chaque juge $i \in J_2$ est affecté à $P-1$ matchs impliquant l'équipe 2 (i.e., $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{2jp} x_{ijp} = P-1 \quad \forall i \in J_2$). Se référant aux contraintes (4.35) et (4.36), nous obtenons que $d_{i2}^5 = P-2$, et donc, puisque $|J_2| = 5$:

$$\sum_{i \in J_2} d_{i2}^5 = 5(P-2). \tag{4.44}$$

Maintenant, utilisant l'hypothèse que chaque juge est affecté à P matchs impliquant les équipes du groupe G, il s'ensuit qu'il existe deux équipes d'indices $\bar{t}$ et $\tilde{t}$, $3 \leq \bar{t}, \tilde{t} \leq P+1$ telles que les juges de J_2 sont affectés à deux matchs impliquant chacune de ces équipes (i.e., $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{\bar{t}jp} x_{ijp} = \sum_{p=1}^{P} \sum_{j=1}^{M} s_{\tilde{t}jp} x_{ijp} = 2 \quad \forall i \in J_2$) ; le premier match lorsque $\bar{t}$ (respectivement $\tilde{t}$) sont en compétition avec l'équipe 2, et le second match lorsque $\bar{t}$ rencontre $\tilde{t}$. Il découle donc de (4.35) et (4.36) que $d_{i\bar{t}}^5 = d_{i\tilde{t}}^5 = 1$. Par ailleurs, les juges de J_2 sont affectés exactement une fois aux matchs impliquant le reste des équipes du groupe G (i.e., $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp} = 1 \quad \forall t \in G$ et $t \neq 2, \bar{t}, \tilde{t}$). Ces matchs sont ceux où ces équipes

rencontrent l'équipe 2. Par conséquent, pour ces équipes, $d_{it}^5 = 0$. En somme, puisque $|J_2| = 5$, nous avons :

$$\sum_{t=3}^{P+1} \sum_{i \in J_2} d_{it}^5 = 10. \tag{4.45}$$

Finalement, les juges $i \in J_2$ ne sont affectés à aucun match impliquant une équipe ne faisant pas partie de G (i.e., $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp} = 0 \quad \forall t > P+1$). Les contraintes (4.35) et (4.36) nous permettent alors de déduire que $d_{it}^5 = 0$, et par conséquent,

$$\sum_{i \in J_2} d_{it}^5 = 0 \qquad \forall t > P+1. \tag{4.46}$$

De (4.43), (4.44), (4.45) et (4.46), nous concluons que :

$$\sum_{t=1}^{2M} \sum_{i \in J_2} d_{it}^5 = 5P.$$

Nous montrons maintenant, qu'en général, pour tout $\tau = 1, \ldots, \frac{P+1}{2}$, nous avons :

$$\sum_{t=1}^{2M} \sum_{i \in J_\tau} d_{it}^5 = 5(P + \tau - 2). \tag{4.47}$$

Décomposons la sommation de (4.47) comme suit :

$$\sum_{t=1}^{2M} \sum_{i \in J_\tau} d_{it}^5 = \sum_{t=1}^{\tau-1} \sum_{i \in J_\tau} d_{it}^5 + \sum_{i \in J_\tau} d_{i\tau}^5 + \sum_{t=\tau+1}^{P+1} \sum_{i \in J_\tau} d_{it}^5 + \sum_{t=P+2}^{2M} \sum_{i \in J_\tau} d_{it}^5 \tag{4.48}$$

et déterminons la plus grande valeur que peut prendre chacun des quatre termes de droite de (4.48) .

Terme 1. Soit $t < \tau$. Par hypothèse, tous les matchs de l'équipe t ne se voient affecter que des juges de J_θ où $\theta \leq t$. Donc, aucun juge $i \in J_\tau$ ne peut être affecté à un match de cette équipe ce qui se traduit par :

$$\sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp} = 0. \tag{4.49}$$

Les contraintes (4.36) nous permettent alors de déduire que $d_{it}^5 = 0$, et par conséquent,

$$\sum_{t=1}^{\tau-1} \sum_{i \in J_\tau} d_{it}^5 = 0. \tag{4.50}$$

Terme 2. Considérons maintenant le cas où $t = \tau$ et soit $i \in J_\tau$. Il est clair que la plus grande valeur que pourrait prendre $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{\tau jp} x_{ijp}$ serait égale à P si le juge i était affecté à tous les matchs de l'équipe τ. Or, comme les juges des ensembles J_θ, où $\theta < \tau$, sont affectés à un match impliquant l'équipe τ (soit lorsque θ rencontre τ), alors $i \in J_\tau$ peut être affecté au plus $(P - \sum_{\theta=1}^{\tau-1} 1)$ fois aux matchs de l'équipe τ, et donc, la plus grande valeur que peut prendre $\sum_{p=1}^{P} \sum_{j=1}^{M} s_{\tau jp} x_{ijp}$ est $P - \tau + 1$.

Supposons donc que :

$$\sum_{p=1}^{P} \sum_{j=1}^{M} s_{\tau jp} x_{ijp} = P - \tau + 1. \tag{4.51}$$

Il s'ensuit alors de (4.35) et (4.36) que $d_{i\tau}^5 = P - \tau$ pour tout $i \in J_\tau$. Puisque l'ensemble J_τ comprend 5 juges, nous obtenons :

$$\sum_{i \in J_\tau} d_{i\tau}^5 = 5(P - \tau). \tag{4.52}$$

Terme 3. Soient $\tau + 1 \leq t \leq P + 1$ et $i \in J_\tau$. Utilisant l'hypothèse que le juge i est affecté à P matchs impliquant les équipes du groupe G, nous avons :

$$\begin{aligned} \sum_{t \in G} \sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp} &= \sum_{t=1}^{P+1} \sum_{p=1}^{P} \sum_{j=1}^{M} s_{tjp} x_{ijp} \\ &= 2P. \end{aligned}$$

Ainsi,

$$\begin{aligned} 2P &= \sum_{t=1}^{P+1}\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} \\ &= \sum_{t=1}^{\tau-1}\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} + \sum_{p=1}^{P}\sum_{j=1}^{M} s_{\tau jp}x_{ijp} + \sum_{t=\tau+1}^{P+1}\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} \\ &= 0 + (P-\tau+1) + \sum_{t=\tau+1}^{P+1}\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} \qquad \text{d'après (4.49) et (4.51).} \end{aligned}$$

Par conséquent, $\sum_{t=\tau+1}^{P+1}\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} = P+\tau-1$.

Maintenant, il est facile de voir que $i \in J_\tau$ est affecté au moins une fois aux matchs impliquant chacune des équipes $t = \tau+1,\ldots,P+1$ (i.e., $\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} \geq 1 \; \forall t = \tau+1,\ldots,P+1$). Ces matchs sont ceux où ces équipes rencontrent l'équipe τ. Se référant aux contraintes (4.35) et (4.36), nous obtenons :

$$\begin{aligned} \sum_{t=\tau+1}^{P+1} d_{it}^5 &= \sum_{t=\tau+1}^{P+1}\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} - \sum_{t=\tau+1}^{P+1} 1 \\ &= (P+\tau-1) - (P+1-\tau-1+1) \\ &= 2\tau-2 \end{aligned}$$

et donc,

$$\sum_{t=\tau+1}^{P+1}\sum_{i\in J_\tau} d_{it}^5 = 10(\tau-1) \tag{4.53}$$

car l'ensemble J_τ comprend 5 juges.

Terme 4. Par hypothèse, les juges $i \in J_\tau$ ne sont affectés qu'aux matchs impliquant les équipes du groupe G (i.e., $\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} = 0 \quad \forall t > P+1$). Les contraintes (4.36) nous permettent alors de déduire que, pour ces équipes $t > P+1$, $d_{it}^5 = 0$, et par conséquent,

$$\sum_{t=P+2}^{2M}\sum_{i\in J_\tau} d_{it}^5 = 0. \tag{4.54}$$

Nous sommes finalement en mesure de déduire le résultat (4.47). Il découle immédiatement de (4.48), (4.50), (4.52), (4.53) et (4.54).

Maintenant, si on considère l'ensemble des juges J affectés aux matchs impliquant les équipes du groupe G, nous avons :

$$\begin{aligned}
\sum_{t=1}^{2M}\sum_{i\in J} d_{it}^5 &= \sum_{t=1}^{2M}\sum_{\tau=1}^{\frac{P+1}{2}}\sum_{i\in J_\tau} d_{it}^5 \\
&= \sum_{\tau=1}^{\frac{P+1}{2}}\sum_{t=1}^{2M}\sum_{i\in J_\tau} d_{it}^5 \\
&= \sum_{\tau=1}^{\frac{P+1}{2}} 5(P+\tau-2) \qquad \text{d'après (4.47)} \\
&= 5((P-2)\frac{P+1}{2} + \sum_{\tau=1}^{\frac{P+1}{2}}\tau) \\
&= 5(\frac{(P+1)(P-2)}{2} + \frac{(P+1)(P+3)}{8}) \\
&= \frac{25}{8}(P+1)(P-1).
\end{aligned}$$

La situation décrite plus haut est la situation où il y a le maximum de violations de la contrainte de *séquence* en considérant un groupe G. La valeur $\frac{25}{8}(P+1)(P-1)$ constitue donc une borne supérieure sur le nombre de violations de cette contrainte en ne considérant que les juges affectés aux matchs d'un groupe donné. Si on considère tous les groupes et donc tous les juges, nous obtenons :

$$\begin{aligned}
\sum_{t=1}^{2M}\sum_{i=1}^{N} d_{it}^5 &\leq \frac{2M}{P+1}\frac{25}{8}(P+1)(P-1) \\
&= \frac{25}{4}(P-1)M.
\end{aligned}$$

(Notons que le nombre de groupes est le nombre total des équipes ($2M$) divisé par le nombre d'équipes dans chaque groupe ($P+1$).) ■

Résumons maintenant l'ensemble de ces résultats dans le tableau 4.1 où pour chaque règle d'affectation identifiée par son nom (colonne 2), et caractérisée par le terme cor-

respondant dans la fonction objectif (4.1) $f_v(x)$ (colonne 3), nous retrouvons les valeurs des bornes inférieure L_v (colonne 4) et supérieure U_v (colonne 5).

v	Contrainte	$f_v(x)$	L_v	U_v
1	*Équilibre*	$\sum_{p=1}^{P}\sum_{j=1}^{M}(d_{jp}^{1+}+d_{jp}^{1-})$	0	$4PM$
2	*Affiliation*	$\sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{c=1}^{C}d_{cjp}^{2}$	0	$4PM$
3	*Diversité*	$\sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{k=1}^{K}d_{kjp}^{3}$	0	$(K-1)PM$
4	*Couplage*	$\sum_{i=1}^{N-1}\sum_{r=i+1}^{N}d_{ir}^{4}$	0	$10(P-1)M$
5	*Séquence*	$\sum_{t=1}^{2M}\sum_{i=1}^{N}d_{it}^{5}$	0	$\frac{25}{4}(P-1)M$

Tableau 4.1 – Valeurs de L_v et U_v

Considérant ces valeurs L_v et U_v dans le tableau 4.1, nous sommes maintenant en mesure de spécifier les valeurs des poids à l'aide du système d'équations (4.15) :

$$
\begin{aligned}
w_1 &= 1\\
w_2 &= 4w_1PM+1\\
w_3 &= 4w_1PM+4w_2PM+1\\
w_4 &= 4w_1PM+4w_2PM+w_3(K-1)PM+1\\
w_5 &= 4w_1PM+4w_2PM+w_3(K-1)PM+10w_4(P-1)M+1\\
w_6 &= 4w_1PM+4w_2PM+w_3(K-1)PM+10w_4(P-1)M+\frac{25}{4}w_5(P-1)M+1.
\end{aligned}
$$

En résolvant ce système et en réarrangeant les termes nous obtenons :

$$\begin{aligned}
w_1 &= 1 \\
w_2 &= 4PM+1 \\
w_3 &= (4PM+1)^2 \\
w_4 &= (4PM+1)^2((K-1)PM+1) \\
w_5 &= (4PM+1)^2((K-1)PM+1)(10(P-1)M+1) \\
w_6 &= (4PM+1)^2((K-1)PM+1)(10(P-1)M+1)(\frac{25}{4}(P-1)M+1).
\end{aligned}$$

4.1.3 Linéarisation du modèle

Dans les sections suivantes nous présentons un certain nombre de méthodes de résolution basées sur des métaheuristiques. Dans le but d'évaluer leur efficacité numérique et la qualité des solutions qu'elles génèrent, nous introduisons maintenant une linéarisation du modèle MG[1] afin de pouvoir le résoudre avec le logiciel CPLEX. En fait, seules les contraintes (4.10) qui modélisent la contrainte de *couplage* sont quadratiques :

$$\sum_{p=1}^{P}\sum_{j=1}^{M} x_{ijp}x_{rjp} - d_{ir}^4 \leq 1 \qquad \forall\, i,r>i. \tag{4.10}$$

Nous présentons trois linéarisations des contraintes (4.10) qui diffèrent par le nombre de variables et de contraintes nécessaires pour les linéariser.
La première approche plus classique consiste à associer à chaque quadruple (i,r,j,p) une nouvelle variable $z_{irjp} \geq 0$ indiquant si la paire de juges (i,r) est affectée au match j de la ronde p ou non. Il suffit donc de considérer les contraintes supplémentaires suivantes :

$$x_{ijp}+x_{rjp}-z_{irjp} \leq 1 \qquad \forall i,r>i,j,p \tag{4.55}$$

$$-x_{ijp}+z_{irjp} \leq 0 \qquad \forall i,r>i,j,p \tag{4.56}$$

$$-x_{rjp}+z_{irjp} \leq 0 \qquad \forall i,r>i,j,p \tag{4.57}$$

et de remplacer dans (4.10) le terme quadratique $x_{ijp}x_{rjp}$ par z_{irjp} pour obtenir les nou-

velles contraintes :

$$\sum_{p=1}^{P}\sum_{j=1}^{M} z_{irjp} - d_{ir}^{4} \leq 1 \qquad \forall i, r > i. \tag{4.58}$$

Même si cette approche permet de linéariser les contraintes de *couplage*, il n'en reste pas moins que la taille du modèle augmente tant en nombre de variables qu'en nombre de contraintes. Ceci entraîne une réduction de la dimension des problèmes qu'il est possible de résoudre en utilisant CPLEX.

Nous présentons maintenant quelques ajustements pour réduire le nombre de variables et de contraintes supplémentaires. Tout d'abord, observons que dans toute solution réalisable, $\sum_{j=1}^{M} x_{ijp}x_{rjp}$ ne peut prendre que l'une des deux valeurs suivantes : 1 si la paire de juges (i,r) est affectée à un même match de la ronde p et 0 sinon. En effet, selon les contraintes (4.2) du modèle MG[1], un juge ne peut être affecté à plus d'un match d'une ronde donnée. Aussi, plutôt que d'associer une variable $z_{irjp} \geq 0$ à chaque quadruple (i,r,j,p), associons-nous à chaque triplet (i,r,p) une variable $z_{irp} \geq 0$ permettant de vérifier si la paire de juges (i,r) est affectée à un même match de la ronde p.
Considérons donc l'ensemble des contraintes suivantes :

$$-\sum_{j=1}^{M}(x_{ijp}+x_{rjp}) + z_{irp} \leq 0 \qquad \forall i, r > i, p \tag{4.59}$$

$$x_{ijp} - x_{rjp} + z_{irp} \leq 1 \qquad \forall i, r > i, j, p \tag{4.60}$$

$$-x_{ijp} + x_{rjp} + z_{irp} \leq 1 \qquad \forall i, r > i, j, p \tag{4.61}$$

$$x_{ijp} + x_{rjp} - z_{irp} \leq 1 \qquad \forall i, r > i, j, p. \tag{4.62}$$

Ainsi,

- si ni i ni r ne sont affectés à un match de la ronde p, alors $z_{irp} = 0$ (par les contraintes (4.59)) ;
- si i seulement est affecté à un match de la ronde p, alors $z_{irp} = 0$ (par les contraintes (4.60)) ;
- si r seulement est affecté à un match de la ronde p, alors $z_{irp} = 0$ (par les contraintes (4.61)) ;

- si i et r sont affectés à deux matchs différents de la ronde p, alors $z_{irp} = 0$ (par les contraintes (4.60) ou (4.61)) ;
- si i et r sont affectés à un même match de la ronde p, alors $z_{irp} = 1$ (par les contraintes (4.60) ou (4.61), et (4.62)).

Pour compléter la linéarisation, il suffit de substituer dans (4.10) la somme quadratique $\sum_{j=1}^{M} x_{ijp}x_{rjp}$ par z_{irp} pour obtenir les nouvelles contraintes :

$$\sum_{p=1}^{P} z_{irp} - d_{ir}^4 \leq 1 \qquad \forall i, r > i. \tag{4.63}$$

Notons que le fait d'avoir aggrégé les variables supplémentaires ($z_{irp} = \sum_{j=1}^{M} z_{irjp}$) conduit à une réduction importante du nombre de variables. Ce faisant, cette reformulation se traduit par un accroissement non négligeable du nombre de contraintes.

Examinons maintenant de plus près l'ensemble des contraintes (4.60). Encore une fois, puisqu'un juge ne peut être affecté à plus d'un match d'une ronde donnée, nous constatons qu'il n'est pas nécessaire d'associer une contrainte à chaque match j. Une seule contrainte par ronde de la forme suivante suffit :

$$\sum_{j=1}^{M} (x_{ijp} - x_{rjp}) + z_{irp} \leq 1 \qquad \forall i, r > i, p. \tag{4.64}$$

Par contre, notons qu'il n'est pas possible de remplacer également les contraintes (4.61) par un ensemble de contraintes (4.65) (avec une seule contrainte par ronde) de la forme :

$$\sum_{j=1}^{M} (x_{rjp} - x_{ijp}) + z_{irp} \leq 1 \qquad \forall i, r > i, p. \tag{4.65}$$

En effet, dans le cas où les contraintes (4.64) et (4.65) remplaceraient respectivement les contraintes (4.60) et (4.61), rien ne forcerait z_{irp} à être égale à 0 lorsque les juges i et r sont affectés à des matchs différents d'une ronde p.

En somme, une troisième linéarisation des contraintes quadratiques (4.10) est obtenue

en les remplaçant par les contraintes linéaires (4.63), (4.59), (4.64), (4.61) et (4.62). Notons que nous pourrions également utiliser les contraintes (4.60) et (4.65) au lieu des contraintes (4.64) et (4.61).

Les caractéristiques de ces transformations en termes du nombre de variables et de contraintes nécessaires pour linéariser les contraintes (4.10) du modèle MG^1 sont résumés dans le tableau 4.2. Pour des raisons évidentes, nous avons retenu la troisième transformation pour poursuivre nos comparaisons numériques.

Linéarisation	Variables	Contraintes
(4.55), (4.56), (4.57) et (4.58)	$\frac{M}{2}[N(N-1)P]$	$\frac{3M}{2}[N(N-1)P]$
(4.59), (4.60), (4.61), (4.62) et (4.63)	$\frac{1}{2}[N(N-1)P]$	$\frac{3M+1}{2}[N(N-1)P]$
(4.59), (4.64), (4.61), (4.62) et (4.63)	$\frac{1}{2}[N(N-1)P]$	$[M+1][N(N-1)P]$

Tableau 4.2 – Nombre de variables et de contraintes supplémentaires induites par les différentes linéarisations

4.2 Résolution à l'aide d'un solveur de programmes linéaires

Résumons d'abord la méthodologie utilisée pour générer les problèmes tests pour notre expérimentation numérique. Nous présenterons ensuite les résultats obtenus en utilisant le solveur de programmes linéaires CPLEX 9.13.

4.2.1 Problèmes tests

La méthode de génération des problèmes reprend les principes utilisés pour générer les problèmes dont nous nous sommes servis pour tester les méthodes développées au chapitre 3. Nous avons conservé l'idée de distinguer les problèmes en fonction de la valeur (nulle ou positive) du dernier terme de la fonction objectif (4.1). En effet, eu égard aux résultats numériques présentés à la section 3.2, il semble qu'une valeur non nulle de ce terme (i.e., lorsque le nombre de juges disponibles est insuffisant pour affecter 5

juges à chaque match) induit les instances les plus difficiles à résoudre avec CPLEX, notamment lorsque la taille du problème augmente. Par contre, nous n'avons pas repris la distinction en termes de la valeur (nulle ou positive) des autres composantes de la fonction objectif. D'une part, les expérimentations numériques menées au chapitre 3 ne nous ont pas permis de tirer de conclusion quant à la difficulté de résolution relative des instances d'une catégorie par rapport à l'autre lorsque le problème est résolu avec CPLEX. Quant aux méthodes heuristiques, les résultats des tests réalisés au chapitre précédent semblent indiquer que les instances pour lesquelles la valeur du premier terme de la fonction objectif est nulle sont généralement très faciles à résoudre, dans le sens où des solutions optimales sont obtenues très rapidement quelle que soit la méthode ou la variante utilisée. D'autre part, en raison du nombre accru et de la complexité des contraintes qui régissent le problème étudié dans ce chapitre, il est plus difficile de prévoir si la valeur des 5 premiers termes de la fonction objectif (4.1) est nulle ou positive. En somme, nous considérons dans ce chapitre deux ensembles de problèmes P_1 et P_2. Les problèmes dans P_2 sont générés de façon à ce qu'il existe une solution où tous les matchs comptent 5 juges affectés (i.e., le dernier terme de la fonction objectif (4.1) est égal à 0) alors que pour les problèmes dans P_1, certains matchs comptent 3 juges affectés seulement (i.e., le dernier terme de la fonction objectif (4.1) est positif).

Comme pour les problèmes réels (spécifiques dans le cadre du *concours international d'étude de cas MBA John Molson*), les problèmes que nous avons générés comportent 5 rondes, et par conséquent, le nombre d'équipes est un multiple de 6. (Rappelons que les équipes sont partitionnées en groupes de 6, et qu'à chaque ronde, chaque équipe rencontre une des autres équipes de son groupe.) Afin d'étudier le comportement des procédures de résolution en fonction de la dimension des problèmes, nous considérons des problèmes avec un total de 75, 150 et 450 matchs. Le nombre de rondes étant fixé à 5, chacun des ensembles P_1 et P_2 comporte trois sous-ensembles avec 15, 30 et 90 matchs par ronde. Finalement, chacun des 6 sous-ensembles contient 10 instances différentes.

Pour chaque instance dans P_1, le nombre de juges disponibles pour chacune des 5 rondes est fixé à $4M$, M étant le nombre de matchs par ronde, de manière à ce qu'il soit impossible d'affecter 5 juges à tous les matchs. Il est de $5M$ pour les instances

dans P_2. Afin de garantir le respect des contraintes incontournables de *composition* et faciliter la satisfaction de la contrainte souple d'*équilibre*, les $4M$ (respectivement $5M$) juges sont répartis comme suit : M *juges en chef*, $2M$ juges de la catégorie *Expert*, et M (respectivement $2M$) juges de la catégorie *Nouveau*. Notons que plutôt que de choisir aléatoirement le nombre de juges de chaque catégorie, nous l'avons fixé au préalable. Cela limite l'apparition de biais dans les instances et permet de disposer d'instances plus homogènes. Ce nombre est également moins important que celui que l'on retrouve dans les problèmes réels. Il est restreint au strict minimum requis afin d'induire des instances *a priori* plus difficiles à résoudre. De plus, dans le but de contraindre la satisfaction des contraintes *inter rondes*, les différentes instances sont générées de sorte à retrouver les mêmes juges dans toutes les rondes (i.e., les mêmes $4M$ (respectivement $5M$) juges sont disponibles pour les 5 rondes).

Pour l'ensemble des instances, nous utilisons les mêmes règles pour générer les autres données concernant les juges (admissibilité, expertise, affiliation). Chaque juge est en conflit avec exactement 4 équipes différentes, choisies aléatoirement parmi celles qui sont en compétition dans les matchs (i.e., n'est pas admissible aux matchs impliquant ces 4 équipes). Rappelons que pour les problèmes réels, chaque juge peut être en conflit avec 4 équipes au plus. La règle que nous utilisons nous permet donc de disposer d'instances *a priori* difficiles à résoudre puisque le fait de réduire le nombre de matchs auxquels chaque juge est admissible rendrait plus ardue la satisfaction des règles souples d'affectation. Toutefois, il convient de souligner que, telles qu'elles sont générées, les instances ont la particularité suivante : la difficulté pourrait décroître en fonction du nombre de matchs. En effet, pour les instances avec 15 matchs par ronde, le pourcentage de matchs auxquels chaque juge est admissible est d'environ 46% tandis qu'il est d'environ 73% pour les instances avec 30 matchs par ronde. Les possibilités pour affecter les juges sont alors plus restreintes dans le premier cas que dans le second. Donc, théoriquement, il serait plus difficile de satisfaire les règles souples d'affectation si le nombre de matchs par ronde est 15 que si ce nombre est 30. Dans le même ordre d'idées, les instances avec 90 matchs par ronde sont caractérisées par un pourcentage encore plus important de matchs auxquels chaque juge est admissible (environ 91%), et seraient par conséquent encore

plus faciles à résoudre. Notons que nous aurions pu déterminer le nombre d'équipes avec lesquels chaque juge est en conflit proportionnellement au nombre total d'équipes plutôt que de le fixer à 4, mais cela ne serait pas réaliste. En effet, pour les instances avec 30 (respectivement 90) matchs par ronde, cela impliquerait que chaque juge est en conflit avec 8 (respectivement 24) équipes.

La deuxième contrainte qui régit l'admissibilité des juges aux matchs stipule qu'un juge affecté à un match *francophone* doit être *bilingue*. Rappelons qu'un match est *francophone* si au moins l'une des deux équipes en compétition dans ce match présente en français. Le nombre d'équipes présentant en français est inspiré des problèmes réels. Il constitue 10% du nombre total d'équipes. Pour chaque catégorie de juges (*Juge en chef*, *Expert*, *Nouveau*), le pourcentage de ceux qui sont *bilingues* est fixé à 30%. Notons que ce pourcentage est moins important que celui que l'on retrouve dans les problèmes réels afin de limiter encore plus le nombre de juges admissibles aux matchs *francophones*, et compliquer davantage la satisfaction des règles souples d'affectation en particulier celles *inter rondes*.

Comme pour les problèmes réels, le nombre de domaines d'expertise est fixé à 6. Pour déterminer les domaines d'expertise que possède chaque juge, un nombre entier α est d'abord choisi aléatoirement selon une loi uniforme dans l'intervalle $[1,6]$. Par la suite, α nombres entiers sont choisis aléatoirement dans le même intervalle. Ces nombres représentent les indices des domaines d'expertise que possède le juge.

Finalement, le nombre de compagnies est fixé à 50. Là encore, ce nombre est moins important que celui que l'on retrouve dans les problèmes réels dans le but de contraindre la satisfaction de la contrainte d'*affiliation*. L'indice de la compagnie que représente chaque juge correspond à un nombre aléatoire généré à partir d'une distribution uniforme discrète sur l'intervalle $[0,50]$. Notons que l'indice 0 signifie que le juge n'est affilié à aucune compagnie (i.e., il est professeur universitaire).

Les caractéristiques de chaque sous-ensemble de problèmes sont rappelées dans le tableau 4.3 dans lequel M désigne le nombre de matchs par ronde (colonne 1), et T représente le nombre d'équipes en compétition dans les différents matchs des 5 rondes (colonne 2). Nous indiquons également, dans les colonnes qui suivent, le nombre de

juges de chaque catégorie disponibles pour chaque ronde.

M	T	*Juge en chef*		*Expert*		*Nouveau*	
		P_1	P_2	P_1	P_2	P_1	P_2
15	30	15	15	30	30	15	30
30	60	30	30	60	60	30	60
90	180	90	90	180	180	90	180

Tableau 4.3 – Caractéristiques des instances de test en fonction du nombre de matchs par ronde

4.2.2 Expérimentations numériques

Nous avons tenté de résoudre avec le logiciel CPLEX 9.13 le modèle MG^1, introduit au paragraphe 4.1.1, où les contraintes quadratiques (4.10) sont remplacées par les contraintes linéaires (4.59), (4.64), (4.61), (4.62) et (4.63) telles que décrites dans la troisième approche de linéarisation du paragraphe 4.1.3. De plus, nous avons exploité les caractéristiques particulières du problème afin de spécifier des bornes supérieures sur les valeurs des variables. L'objectif étant de rendre plus performante l'étape de préoptimisation (fixation des variables, élimination des contraintes, ...) lors de l'exécution de CPLEX 9.13. Les tests ont été réalisés sur l'ensemble des instances décrites au paragraphe précédent.

Dans un premier temps, nous avons limité la durée totale de résolution à 24 heures. Nous avons conservé les valeurs par défaut des autres paramètres de CPLEX. Nous avons constaté que, pour toutes les instances de notre expérimentation numérique, le solveur n'a pas été en mesure de résoudre la relaxation linéaire du problème à la racine de l'arbre de recherche.

Nous avons donc procédé à un réglage plus affiné des paramètres de CPLEX. Ainsi, nous avons considéré plusieurs combinaisons de paramètres censés rendre plus efficace la résolution et favoriser l'obtention de solutions entières. Ces paramètres concernent

entre autres, la fréquence d'utilisation de l'heuristique destinée à trouver des solutions entières aux différents nœuds durant la procédure de séparation et coupes (*branch-and-cut*), le choix des coupes à générer, les algorithmes utilisés pour résoudre la relaxation linéaire aussi bien à la racine de l'arbre de recherche qu'à chacun de ses nœuds, le choix des variables pour effectuer le branchement ... Sans entrer dans les détails de cette expérimentation numérique, mentionnons que certaines combinaisons ont eu pour effet de réduire considérablement le temps de résolution de la relaxation linéaire et ont permis l'exploration de quelques nœuds. Les performances de CPLEX demeurent toutefois mauvaises puisque, pour toutes les instances considérées, aucune solution entière n'a pu être identifiée par le solveur dans une limite de temps de 10 heures.

Dans le but d'avoir quand même un point de comparaison afin d'évaluer l'efficacité numérique de nos méthodes basées sur des métaheuristiques, nous avons finalement résolu la relaxation linéaire du problème. Nous nous servirons de la valeur de la solution associée à cette relaxation comme borne inférieure sur la valeur de la solution optimale (pour mesurer le pourcentage de déviation de cette valeur de référence). De plus, afin d'améliorer cette borne, l'un des objectifs (règle souple d'affectation), en l'occurence celui qui vise à maximiser le nombre de matchs avec 5 juges affectés, est transformé en contrainte. Des tests préliminaires ont montré qu'il est préférable de résoudre la formulation duale de ce problème avec l'algorithme primal du simplexe.
Les résultats obtenus au cours de cette dernière expérimentation sont résumés dans le tableau 4.4. Les colonnes désignent dans l'ordre, l'ensemble des problèmes, le nombre de matchs par ronde (*Taille*), le pourcentage des instances résolues (*%Res*), et finalement le temps minimal (*Min CPU*), maximal (*Max CPU*) et moyen (*Ave CPU*) requis pour ces instances. Les temps de résolution sont indiqués en secondes.

4.3 Approche de résolution par les métaheuristiques

L'approche de résolution s'inspire du principe de décomposition dans le sens où les sous-problèmes associés à chacune des rondes sont optimisés en considérant les affectations des autres rondes fixées à leurs valeurs courantes. Nous pouvons noter éga-

	Taille	*%Res*	*Min CPU*	*Max CPU*	*Ave CPU*
P_1	15	100	60,22	2 848,47	982,46
	30	100	621,54	1 658,81	893,05
	90	0	-	-	-
P_2	15	100	111,31	916,38	350,83
	30	100	1 806,47	4 475,85	2 400,34
	90	0	-	-	-

Tableau 4.4 – Résultats obtenus en utilisant le solveur CPLEX

lement une analogie entre cette approche et les méthodes de type «*cyclic coordinate descent*» [75] (comme celles de *Gauss-Seidel* et de *Jacobi*), utilisées pour optimiser une fonction à plusieurs variables.

Globalement, nous procédons comme suit : partant d'une solution initiale générée au cours de la phase 1, nous résolvons séquentiellement les (ou certains) sous-problèmes qui composent le problème global, et nous modifions progressivement la solution complète du problème. Puisqu'à chaque fois que nous considérons un sous-problème, nous fixons les solutions des $(P-1)$ autres sous-problèmes, nous pouvons résoudre le problème résultant en utilisant l'une des méthodes développées au chapitre 3 ; mais ajustées, bien entendu, pour tenir compte des nouvelles contraintes. La solution initiale x_p^0, servant à amorcer la résolution du sous-problème associé à la ronde p, est fonction de la méthode de résolution utilisée. Évidemment, au cours de la résolution, l'évaluation des violations des contraintes inter rondes (*séquence* et *couplage*) se fait en considérant les affectations des autres rondes fixées à leurs valeurs courantes. Ce processus est réitéré aussi longtemps que le critère d'arrêt n'est pas satisfait. Dans notre implémentation, celui-ci est spécifié en termes d'un temps CPU maximal alloué *tempsmax*. La procédure de résolution s'arrête également dès qu'une solution «optimale» est générée. Notons que, puisqu'aucune des instances de notre expérimentation numérique n'a pu être résolue avec CPLEX, nous considérons qu'une solution est «optimale» si toutes les règles d'affectation sont respectées ; nous y reviendrons dans le paragraphe 4.4.5. Le schéma général de résolution est présenté dans la figure 4.2.

Dans le reste de ce chapitre, nous allons considérer diverses variantes de la procédure

Phase 1

$xinit_p$, la solution initiale du sous-problème associé à la ronde p (ensemble des affectations dans les matchs de la ronde p)

$xinit := \bigcup_{p=1}^{P} xinit_p$, la solution initiale du problème

$xbestG := xinit$, la meilleure solution rencontrée jusqu'à présent

$xbestLast := xbestG$, la meilleure solution récemment obtenue

$x := xbestG$ ou $xbestLast$ suivant la variante

Phase 2

Tant que (le critère d'arrêt n'est pas satisfait) **faire**

  Identifier Ω l'ensemble des rondes à considérer

  Pour chaque ronde $p \in \Omega$ (considérant séquentiellement les rondes de Ω selon un ordre aléatoire) **faire**

    Fixer les affectations des rondes $q \neq p$ à leurs valeurs courantes dans x

    RÉSOLUTION DU SOUS-PROBLÈME ASSOCIÉ À LA RONDE p

    Résoudre le sous-problème associé à la ronde p avec x_p^0 comme solution initiale (x_p^0 dépend de la méthode de résolution). Soit x_p^* la meilleure solution ainsi obtenue

    MISE À JOUR

    $xbestLast := (xbestLast - \{xbestLast_p\}) \cup \{x_p^*\}$

    Si ($xbestLast$ est meilleure que $xbestG$) **Alors**

      $xbestG := xbestLast$

    Fin Si

    Mettre à jour x

  Fin Pour

Fait

$xbestG$ est la meilleure solution générée

Figure 4.2 – Schéma général de résolution du problème sur plusieurs rondes

de résolution décrite ci-haut. Ces variantes sont caractérisées par :

1. la façon de spécifier l'ensemble Ω des rondes à considérer à chaque itération majeure de la phase 2 : nous considérons deux façons différentes de spécifier cet ensemble. La première consiste à considérer toutes les rondes, i.e., $\Omega = \{1,\ldots,P\}$. La seconde consiste à ne considérer que les rondes où certaines contraintes du problème sont violées ;

2. la façon de spécifier la solution x utilisée pour fixer les solutions des $(P-1)$ sous-problèmes, autres que le sous-problème que l'on s'apprête à résoudre : nous considérons aussi deux façons différentes pour spécifier cette solution. La première considère les affectations que nous retrouvons dans la meilleure solution rencontrée jusqu'à présent *xbestG*. La deuxième utilise plutôt les affectations que nous retrouvons dans la meilleure solution récemment obtenue *xbestLast*. Notons que *xbestLast* est initialisée avec *xbestG* à la fin de la phase 1. Elle est régulièrement mise à jour au cours de la phase 2. En effet, chaque fois qu'un sous-problème associé à une ronde p est résolu, une nouvelle solution x_p^* de ce sous-problème est générée. Mettre à jour *xbestLast* consiste alors à remplacer les affectations dans les matchs de la ronde p par celles que nous retrouvons dans x_p^* ; les affectations dans les matchs des rondes $q \neq p$ sont maintenues à leurs valeurs courantes. Le parallèle avec les principes d'intensification et de diversification est immédiat. Dans le premier cas ($x := xbestG$), on suit plutôt une stratégie d'intensification. Le second cas ($x := xbestLast$) correspond davantage à une stratégie de diversification ;

3. la méthode utilisée pour résoudre les sous-problèmes associés aux rondes : nous considérons trois méthodes différentes. Ce sont des généralisations des trois métaheuristiques introduites au chapitre 3.

Le processus de résolution est donc régi par l'action conjuguée des trois éléments susmentionnés, et chaque triplet (Ω, x, méthode) donne lieu à une variante de la procédure de résolution. Les caractéristiques particulières de ces variantes seront présentées dans les sections qui suivent.

Dans l'approche de résolution que nous venons de décrire, le fait de passer d'un

sous-problème associé à une ronde à un autre correspond à un changement de voisinage. En effet, l'ensemble des solutions qu'il est possible d'obtenir en ne modifiant qu'une partie de la solution, en l'occurence les affectations dans les matchs de la ronde considérée, définit une structure de voisinage. Également, lorsque la recherche s'arrête dans un optimum local relativement à une structure de voisinage, elle est relancée, mais dans une autre structure de voisinage. Ceci rappelle la métaheuristique de recherche à voisinage variable (RVV), dans laquelle toutefois l'ensemble des structures de voisinage utilisé est fixé préalablement et non ajusté en fonction des résultats obtenus comme c'est le cas ici (variantes où l'ensemble Ω est limité aux rondes comportant des violations).

Ce qui distingue aussi notre approche de résolution de la RVV est le fait que la recherche n'est pas nécessairement relancée à partir d'une solution choisie dans le voisinage de la meilleure solution connue. En fait, cette stratégie s'avère parfois pénalisante comme le soulignent les auteurs dans [55] : « It appears from this study that local optima tend to coalesce at the top of large mountains. However, there may be several of them. This poses a problem for VNS : indeed a few iterations will quite quickly determine the best solution in a large region (the top of the highest mountain in that region). But then no better solution will be found near to it. Moreover, solutions in neighborhoods far from the incumbent will necessarily be rather different from it and possibly of poor value. So they will not provide indications as to where are better solutions (higher mountains, if any). If a single descent (or ascent) is made the local optimum found may not be very good either. In other words, when one must move very far from the incumbent solution, basic VNS tends to degenerate into multi-start, which is not among the best metaheuristics.» Cette observation est d'ailleurs corroborée par notre étude expérimentale. Nous verrons plus loin, lorsque nous comparerons numériquement les différentes variantes, que les variantes utilisant *xbestLast* arrivent plus facilement à s'échapper des optima locaux que celles utilisant *xbestG*. Notons que la stratégie que nous adoptons pour surmonter la difficulté soulignée par les auteurs dans [55] est sensiblement différente de la leur (qui est la recherche à voisinage variable biaisée), dans la mesure où nous acceptons toute solution obtenue suite à la recherche plutôt que de ne l'accepter que si elle est suffisamment distante de la meilleure solution connue.

Enfin, une dernière différence entre la RVV et notre approche de résolution concerne la structure de voisinage utilisée lorsqu'une amélioration de la fonction objectif est obtenue : dans le cas de la RVV, la recherche est relancée dans le premier voisinage alors que dans notre cas les structures de voisinage sont considérées séquentiellement, car rien ne justifie le déplacement au premier voisinage.

4.4 Variantes utilisant la méthode de recherche avec tabous

Cette section est consacrée aux variantes utilisant la méthode de recherche avec tabous pour résoudre les sous-problèmes associés aux rondes. Nous introduisons d'abord le modèle mathématique sur lequel se base la résolution des sous-problèmes (paragraphe 4.4.1). Ensuite, dans le paragraphe 4.4.2, nous présentons les caractéristiques des variantes que nous considérons. Nous détaillons leurs différentes composantes dans les paragraphes 4.4.3 et 4.4.4, puis comparons la qualité de leurs solutions ainsi que leur efficacité dans le paragraphe 4.4.5.

4.4.1 Modèle mathématique

Pour retrouver le modèle mathématique du sous-problème associé à la ronde p, il suffit de fixer, dans le modèle MG^1, les variables associées aux rondes $q \neq p$ à leurs valeurs courantes, ce qui génère un modèle MG^1_p. Le modèle MG^2_p, que nous allons décrire dans ce qui suit, est une version pénalisée du modèle MG^1_p. Il s'agit aussi d'une généralisation du modèle M^2 (cf. paragraphe 3.4.2) obtenue en modifiant la façon d'évaluer la contrainte de *diversité* et en ajoutant les contraintes associées aux juges *experts*, celles d'*équilibre*, d'*affiliation*, et les contraintes inter rondes de *couplage* et de *séquence*.

Désignons par I_p et I_p^f respectivement les ensembles des juges disponibles et fictifs pour la ronde p ($I_p = \{i \quad : \quad d_{ip} = 1\}$). Se référant à la notation utilisée dans le paragraphe 3.4.2, si $|I_p| \geq 5M$, alors $(|I_p| - 5M)$ nœuds destination fictifs sont ajoutés et $|I_p^f| = 0$. Ces nœuds sont associés à un match fictif $(M+1)_p$. Chaque nœud source est associé à un juge i. Si par contre $|I_p| < 5M$, alors nous devons affecter 3 juges à chaque match, et une paire de juges additionnelle à $\left\lfloor \frac{|I_p|-3M}{2} \right\rfloor$ matchs. Nous ajoutons

donc $\kappa_p = (|I_p| - 3M) \bmod 2$ nœud destination fictif que nous associons au match fictif $(M+1)_p$, et $|I_p^f| = (5M - |I_p| + \kappa_p)$ nœuds source que nous associons à des juges fictifs. Soit

$$\overline{N}_p = |I_p| + |I_p^f| = \begin{cases} 5M + \kappa_p & \text{si } |I_p| < 5M \\ |I_p| & \text{sinon} \end{cases}$$

et dénotons par

$$\alpha_p = \begin{cases} \sum_{q=1}^{p-1} \overline{N}_q & \text{si } p \geq 2 \\ 0 & \text{sinon.} \end{cases}$$

Alors, les 5 nœuds associés au match j de la ronde p sont $(\alpha_p + j)$, $(\alpha_p + M + j)$, $(\alpha_p + 2M + j)$, $(\alpha_p + 3M + j)$ et $(\alpha_p + 4M + j)$.

La fonction objectif comprend 9 termes (au lieu de seulement 4 comme dans le paragraphe 3.4.2) : $ad_j(x_p)$, $nj_j(x_p)$, $lj_j(x_p)$, $ej_j(x_p)$, $af_j(x_p)$, $dv_j(x_p)$, $cp_{ir}(x_p)$, et $se_{it}(x_p)$ associés respectivement aux contraintes d'*admissibilité*, de *composition*, d'*équilibre*, d'*affiliation*, de *diversité*, de *couplage*, et de *séquence*. Les valeurs de ces termes sont spécifiés comme suit :

1. Admissibilité :

$$ad_j(x_p) = \sum_{\beta=0}^{4} \sum_{i \in I_p} (1 - a_{ijp}) x_{i(\alpha_p + \beta M + j)} \quad j = 1, \ldots, M.$$

2. Exactement 3 ou 5 juges doivent être affectés à chaque match :
   dénotons par

$$F_j(x_p) = \sum_{\beta=0}^{4} \sum_{i \in I_p^f} x_{i(\alpha_p + \beta M + j)}$$

   le nombre de juges fictifs affectés au match j dans la solution x_p. Nous devons avoir $nj_j(x_p) = 0$ si et seulement si $F_j(x_p) = 0$ ou 2. L'expression

$$nj_j(x_p) = \max\left(\left\lfloor \frac{F_j(x_p)}{2} \right\rfloor - 1, F_j(x_p) - 2 \left\lfloor \frac{F_j(x_p)}{2} \right\rfloor \right) \quad j = 1, \ldots, M$$

   nous permet de formaliser la contrainte d'avoir 3 ou 5 juges affectés à chaque

match j de la ronde p.

3. Un *juge en chef* doit être affecté à chaque match :

   dénotons par

$$L_j(x_p) = \sum_{\beta=0}^{4} \sum_{i \in I_p} l_i x_{i(\alpha_p+\beta M+j)}$$

   le nombre de *juges en chef* affectés au match j dans la solution x_p. Alors, la valeur de $lj_j(x_p)$ doit être égale à 0 si et seulement si $L_j(x_p) \geq 1$, i.e.,

$$lj_j(x_p) = \max\big(1 - L_j(x_p), 0\big) \quad j = 1, \ldots, M.$$

4. Un juge *expert*, autre que le *juge en chef*, doit être affecté à chaque match :

   dénotons par

$$E_j(x_p) = \sum_{\beta=0}^{4} \sum_{i \in I_p} v_i x_{i(\alpha_p+\beta M+j)}$$

   le nombre de juges de la catégorie *Expert* affectés au match j dans la solution x_p. Alors, la valeur de $ej_j(x_p)$ doit être égale à 0 si et seulement si, soit $E_j(x_p) \geq 1$ ou bien $L_j(x_p) \geq 2$ (car tous les *juges en chef* sont des juges *experts*), i.e.,

$$ej_j(x_p) = \max\big(1 - \max(E_j(x_p), L_j(x_p) - 1), 0\big) \quad j = 1, \ldots, M.$$

5. Contrainte d'*équilibre* :

   dénotons par

$$N_j(x_p) = \sum_{\beta=0}^{4} \sum_{i \in I_p} (1 - l_i - v_i) x_{i(\alpha_p+\beta M+j)}$$

   le nombre de juges *nouveaux* affectés au match j dans la solution x_p. Alors, la valeur de $eq_j(x_p)$ doit être égale à 0 si et seulement, hormis le *juge en chef* affecté à ce match, le nombre des juges *experts* et *nouveaux* sont égaux. Ainsi,

$$eq_j(x_p) = \big|L_j(x_p) + E_j(x_p) - N_j(x_p) - \min(L_j(x_p), 1)\big| \quad j = 1, \ldots, M.$$

6. Contrainte d'*affiliation* :

soit

$$A_{cj}(x_p) = \sum_{\beta=0}^{4} \sum_{i \in I_p} r_{ic} x_{i(\alpha_p+\beta M+j)}$$

le nombre de juges représentant la compagnie c, affectés au match j dans la solution x_p. Nous souhaitons que chaque compagnie soit représentée au plus une fois dans ce match. Par conséquent,

$$af_j(x_p) = \sum_{c=1}^{C} \max\left(A_{cj}(x_p) - 1, 0\right) \quad j = 1, \ldots, M.$$

7. Contrainte de *diversité* :

   dénotons par

$$D_{kj}(x_p) = \sum_{\beta=0}^{4} \sum_{i \in I_p} e_{ik} x_{i(\alpha_p+\beta M+j)}$$

   le nombre de juges possédant le domaine d'expertise k, affectés au match j dans la solution x_p. Pour favoriser la représentation de tous les domaines d'expertise dans le match j, nous définissons $dv_j(x_p)$ comme suit :

$$dv_j(x_p) = \sum_{k=1}^{K} \max\left(1 - D_{kj}(x_p), 0\right) \quad j = 1, \ldots, M.$$

8. Contrainte de *couplage* :

   soit

$$C_{ir}(x_p) = \sum_{q=1}^{P} \sum_{j=1}^{M} \left(\sum_{\beta=0}^{4} x_{i(\alpha_q+\beta M+j)} \right) \left(\sum_{\beta=0}^{4} x_{r(\alpha_q+\beta M+j)} \right)$$

   le nombre de fois que la paire de juges (i, r) est affectée à un même match. La contrainte de *couplage* stipule qu'une paire spécifique de juges ne peut être affectée plus d'une fois au cours de l'ensemble des rondes. Par conséquent, la valeur de $cp_{ir}(x_p)$ doit être égale à 0 si et seulement si $C_{ir}(x_p) \leq 1$. Il s'ensuit que :

$$cp_{ir}(x_p) = \max\left(C_{ir}(x_p) - 1, 0\right) \quad i, r \in I_p \text{ et } r > i.$$

9. Contrainte de *séquence* :

dénotons par

$$S_{it}(x_p) = \sum_{q=1}^{P} \sum_{j=1}^{M} \sum_{\beta=0}^{4} s_{tjq} x_{i(\alpha_q+\beta M+j)}$$

le nombre de fois que le juge i est affecté à un match impliquant l'équipe t. Alors, pour que la contrainte de *séquence* soit satisfaite, il suffit de définir :

$$se_{it}(x_p) = \max(S_{it}(x_p) - 1, 0) \quad i \in I_p \text{ et } t = 1, \ldots, 2M.$$

Rappelons que

$$\overline{N}_p = |I_p| + |I_p^f| = \begin{cases} 5M + \kappa_p & \text{si } |I_p| < 5M \\ |I_p| & \text{sinon.} \end{cases}$$

Le nouveau modèle MG_p^2 peut être formulé comme le problème d'affectation suivant :

$$\min \quad w_1 \sum_{j=1}^{M} eq_j(x_p) + w_2 \sum_{j=1}^{M} af_j(x_p) + w_3 \sum_{j=1}^{M} dv_j(x_p) + w_4 \sum_{i \in I_p} \sum_{\substack{r \in I_p \\ r>i}} cp_{ir}(x_p) \quad (4.66)$$

$$+ w_5 \sum_{t=1}^{2M} \sum_{i \in I_p} se_{it}(x_p) + w_6 \sum_{j=1}^{M} \left(ad_j(x_p) + nj_j(x_p) + lj_j(x_p) + ej_j(x_p) \right)$$

(MG_p^2) **Sujet à**

$$\sum_{\mu=\alpha_p+1}^{\alpha_p+\overline{N}_p} x_{i\mu} = 1 \qquad i \in I_p \cup I_p^f \quad (4.67)$$

$$\sum_{i \in I_p \cup I_p^f} x_{i\mu} = 1 \qquad \mu = \alpha_p + 1, \ldots, \alpha_p + \overline{N}_p \quad (4.68)$$

$$x_{i\mu} = 0 \text{ ou } 1 \qquad i \in I_p \cup I_p^f \quad (4.69)$$
$$\mu = \alpha_p + 1, \ldots, \alpha_p + \overline{N}_p$$

Figure 4.3 – Modèle MG_p^2

Notons qu'aucun terme de la fonction objectif (4.66) n'est associé à la contrainte du *nombre* (visant à maximiser le nombre de matchs avec 5 juges affectés), car la modélisation que nous proposons veille implicitement au respect de cette contrainte.
$w_1, \ldots, w_5$ représentent les poids associés aux différentes règles souples d'affectation. Nous associons aux règles incontournables le même poids w_6. Les valeurs des poids sont déterminées en se référant aux équations (4.15). Ainsi,

$$\begin{aligned}
w_1 &= 1 \\
w_2 &= 5M+1 \\
w_3 &= (5M+1)(4M+1) \\
w_4 &= (5M+1)(4M+1)((K-1)M+1) \\
w_5 &= (5M+1)(4M+1)((K-1)M+1)(10M+1) \\
w_6 &= (5M+1)(4M+1)((K-1)M+1)(10M+1)^2.
\end{aligned}$$

4.4.2 Caractéristiques de ces variantes

Pour générer la solution initiale au cours de la phase 1, les rondes sont considérées séquentiellement. Pour chaque ronde $p = 1, \ldots, P$, les $\overline{N}_p$ juges (nœuds source) sont d'abord affectés aux $\overline{N}_p$ matchs (nœuds destination) de façon aléatoire. Cette solution initiale de la ronde p est ensuite améliorée en utilisant une recherche avec tabous qui fera l'objet du paragraphe suivant. L'évaluation des violations des contraintes inter rondes se fait en tenant compte des solutions des rondes déjà considérées. Notons que le choix de la solution initiale de la ronde est motivé par les résultats des expérimentations numériques présentés au paragraphe 3.4.6. En effet, ces expérimentations ont indiqué que les meilleurs résultats sont obtenus avec une solution générée aléatoirement plutôt qu'avec une solution générée avec une heuristique.

À chaque itération majeure de la phase 2, les rondes dans Ω sont considérées séquentiellement dans un ordre aléatoire. Pour chaque ronde $p \in \Omega$, dépendemment de la variante considérée, les affectations courantes que nous retrouvons dans *xbestG* ou dans *xbestLast* constituent x_p^0, la solution utilisée pour réinitialiser la résolution du sous-

problème associé à p. Une généralisation de la stratégie de diversification présentée au paragraphe 3.4.5 est d'abord utilisée pour modifier x_p^0. Ceci génère une nouvelle solution de la ronde p. La même recherche avec tabous employée au cours de la phase 1 est ensuite utilisée pour améliorer cette nouvelle solution. Les solutions des $(P-1)$ autres rondes sont maintenues fixées à leurs valeurs courantes dans *xbestG* ou dans *xbestLast* suivant la variante.

Le fait d'appliquer d'abord la stratégie de diversification a un impact sur le choix des rondes à inclure dans Ω. En effet, puisque la stratégie de diversification repose sur les violations, alors seules les rondes comportant des violations peuvent faire partie de Ω. Ainsi, seule cette façon de définir Ω est considérée dans les variantes utilisant la méthode de recherche avec tabous. Nous allons maintenant détailler les deux composantes de cette méthode.

4.4.3 Recherche avec tabous

Rappelons que l'objectif est d'améliorer la solution d'une ronde donnée p (solution générée aléatoirement au cours de la phase 1 ou obtenue suite à l'application de la stratégie de diversification s'il s'agit de la phase 2). La méthode utilisée est très similaire à celle présentée au paragraphe 3.4.4. Nous résumons ici ses principaux éléments en soulignant, s'il y a lieu, les différences entre les deux méthodes.

Le voisinage d'une solution réalisable x_p du modèle MG_p^2 est généré en échangeant deux juges i et r affectés présentement à deux nœuds destination associés respectivement à deux matchs différents j et l de la ronde p. Afin de minimiser le risque de cycler, les deux paires (i,j) et (r,l) sont introduites dans la liste tabou LT lorsque $x_p \oplus (i,j,r,l)$ remplace x_p. De plus, pour mieux guider la recherche, les échanges (i,j,r,l) impliquant deux juges fictifs (i.e., $i \in I_p^f$ et $r \in I_p^f$) ne sont pas considérés. En effet, ces échanges n'ont aucune influence sur la valeur de la solution courante (cf. fonction objectif (4.66)).

Un échange (i,j,r,l) est considéré tabou si $(i,l) \in LT$ et $(r,j) \in LT$. Notons que dans la méthode décrite au paragraphe 3.4.4, tout échange est tabou dès que l'une des paires (i,l) ou (r,j) figure dans la liste tabou LT. Des résultats numériques préliminaires ont indiqué que ce dernier critère est très restrictif et qu'il pénalise l'exploration du

voisinage. Il convient en effet de noter que la fonction objectif varie ici sur un plus grand intervalle de valeurs. Plus explicitement, dans le cas du problème étudié au chapitre précédent, la fonction objectif prend un nombre restreint de valeurs et le nombre de solutions de même qualité est très élevé ("larges vallées"). Interdire un petit nombre de solutions augmenterait les chances de confiner la recherche dans une vallée car elle aurait tendance à visiter toujours les mêmes solutions. En interdisant un grand nombre de solutions, nous espérions augmenter les chances de s'échapper des vallées et visiter des solutions variées.

Le même critère d'aspiration est employé ici en révoquant le statut tabou d'une solution lorsque sa valeur est meilleure que celle de la meilleure solution rencontrée jusqu'à présent.

Nous utilisons la stratégie *première amélioration* puisque celle-ci s'est avérée plus efficace que la stratégie *meilleure amélioration*, si l'on se réfère aux résultats de l'expérimentation numérique du chapitre 3 (cf. paragraphe 3.4.6). Ainsi, nous choisissons comme solution courante pour la prochaine itération, la première solution voisine non tabou améliorant la valeur de la solution courante, ou la première solution tabou vérifiant le critère d'aspiration. De plus, lorsque tout le voisinage est énuméré sans pouvoir identifier de telles solutions, nous utilisons le second critère de sélection présenté au paragraphe 3.4.4 pour départager les solutions candidates de même qualité. Ce critère qui repose sur une mémoire à long terme, permet de biaiser la sélection vers les échanges les moins fréquemment utilisés.

Finalement, la recherche s'arrête si aucune amélioration n'est possible (i.e., dans tous les matchs de la ronde p, toutes les règles d'affectation sont respectées) ou si *nitermax* itérations successives ont été complétées sans qu'il y ait eu amélioration de la fonction objectif.

4.4.4 Stratégie de diversification

Au cours de la phase 2, la recherche avec tabous, décrite au paragraphe précédent, est initialisée avec une nouvelle solution de la ronde p générée suite à l'application d'une stratégie de diversification. Le principe de cette stratégie est le même que celui

présenté au paragraphe 3.4.5. Seule la façon de générer les deux ensembles W et G diffère légèrement vu la présence des nouvelles contraintes.

Se référant au schéma de la figure 4.2, soit x_p^0 la solution de départ utilisée pour générer la nouvelle solution de la ronde p. Dénotons par $I(j,x_p^0)$ l'ensemble des 5 juges affectés au match j dans cette solution. Notons que ces juges sont ceux que nous retrouvons dans *xbestG* ou dans *xbestLast* suivant la variante considérée. Les juges faisant partie de l'ensemble W sont choisis comme suit :

1. si $ad_j(x_p^0) > 0$, alors inclure tous les juges $i \in I(j,x_p^0)$ qui ne sont pas admissibles au match j ;
2. si $nj_j(x_p^0) > 0$, alors nous considérons les cas de figure suivants selon le nombre de juges fictifs $F_j(x_p^0)$ affectés au match j :
   (a) si $F_j(x_p^0) = 1$, alors inclure le juge fictif $i \in I(j,x_p^0)$,
   (b) si $F_j(x_p^0) = 3$, alors inclure l'un des juges fictifs $i \in I(j,x_p^0)$,
   (c) si $F_j(x_p^0) = 4$, alors inclure deux des juges fictifs $i, i' \in I(j,x_p^0)$;
3. si $lj_j(x_p^0) > 0$, alors inclure l'un des juges $i \in I(j,x_p^0)$;
4. si $ej_j(x_p^0) > 0$, alors inclure l'un des juges $i \in I(j,x_p^0)$;
5. si $eq_j(x_p^0) > 0$, alors inclure l'un des juges $i \in I(j,x_p^0)$, sélectionné parmi ceux de la catégorie où il y a un surplus (*experts* ou *nouveaux*) ;
6. si $af_j(x_p^0) > 0$, alors, pour chaque paire de juges représentant la même compagnie, inclure l'un des juges $i \in I(j,x_p^0)$;
7. si $dv_j(x_p^0) > 0$, alors inclure l'un des juges $i \in I(j,x_p^0)$;
8. si $cp_{ir}(x_p^0) > 0$, alors inclure l'un des juges i ou r en autant bien entendu qu'il existe un match j de la ronde p tel que $i \in I(j,x_p^0)$ et $r \in I(j,x_p^0)$;
9. si $se_{it}(x_p^0) > 0$, alors inclure i en autant qu'il existe un match j de la ronde p tel que $i \in I(j,x_p^0)$, et l'équipe t est en compétition dans j.

Tous les autres juges font partie de l'ensemble G.

Dans les cas (3, 4, 5, 6, 7 et 8) où un juge est choisi parmi un ensemble pour être inclus dans W, nous biaisons le choix en faveur de celui qui a été le moins souvent

inclus dans un ensemble W dans le passé (plutôt que de le choisir aléatoirement comme au paragraphe 3.4.5). Pour ce faire, dénotons par $n(i)$ le nombre de fois que le juge i a été sélectionné dans le passé pour faire partie d'un ensemble W. La probabilité $p(i)$ de choisir i parmi l'ensemble de juges $\mathscr{E}$ est alors définie comme suit :

$$p(i) = \frac{n(i)}{\sum_{r \in \mathscr{E}} n(r)}.$$

Finalement, comme nous l'avons déjà souligné, pour évaluer les violations des contraintes inter rondes de *couplage* et de *séquence* (valeurs de $cp_{ir}(x_p^0)$ et $se_{it}(x_p^0)$ respectivement), nous considérons les affectations des autres rondes $q \neq p$ fixées à leurs valeurs courantes dans *xbestG* ou dans *xbestLast* suivant la variante considérée.

4.4.5 Expérimentations numériques

Nous comparons maintenant les deux variantes de la procédure de résolution utilisant la recherche avec tabous. Comme nous l'avons déjà mentionné, pour les deux variantes, l'ensemble Ω des rondes à considérer à chaque itération majeure de la phase 2 consiste en l'ensemble des rondes comportant des violations, et la méthode utilisée pour résoudre les sous-problèmes associés aux rondes est la recherche avec tabous. Les deux variantes diffèrent par la façon de spécifier la solution x (cf. figure 4.2). Nous dénotons la variante utilisant *xbestLast* par **RT-Div-Partiel** et celle utilisant *xbestG* par **RT-Int-Partiel**.

Les paramètres de la recherche avec tabous sont fixés comme dans le paragraphe 3.4.6 : lorsqu'on résoud le sous-problème associé à la ronde p, la durée des interdictions est choisie aléatoirement dans l'intervalle $[t_{min}, t_{max}] = [\lfloor 0,8|I_p| \rfloor, \lceil 1,2|I_p| \rceil]$, et le nombre maximal *nitermax* d'itérations successives sans amélioration considéré est égal à $|I_p|$. Rappelons que $|I_p|$ représente le nombre de juges disponibles pour la ronde p. Le choix de ces valeurs est motivé par le fait qu'elles ont permis d'obtenir de bons résultats (si l'on se réfère aux résultats numériques présentés au paragraphe 3.4.6). Le paramètre *tempsmax* utilisé pour spécifier le critère d'arrêt de la procédure de résolution est fixé

à $3N$ secondes, N étant le nombre total de juges. Notons que la procédure peut s'arrêter avant *tempsmax* secondes. C'est le cas lorsqu'une solution «optimale» est générée. Puisque nous ne disposons pas de résultats prouvant l'optimalité d'une solution, la solution que nous qualifions d'«optimale» n'est en fait qu'une solution pour laquelle il est certain, vu la structure de problème, qu'il est impossible d'obtenir une solution meilleure. Pour déterminer sa valeur, nous utilisons des bornes inférieures sur le nombre de violations des différentes règles souples d'affectation. (Rappelons que notre objectif est de déterminer une affectation des juges aux matchs des différentes rondes qui respecte les règles incontournables d'affectation, et qui satisfait le plus possible aux règles souples.) Ces bornes sont définies comme suit : pour chaque instance, nous nous basons sur le nombre de juges disponibles pour chaque ronde pour calculer la borne inférieure sur le nombre de violations des contraintes souples d'*équilibre* et du *nombre*. Les bornes inférieures sur le nombre de violations des autres règles souples d'affectation correspondent à la valeur 0.

Pour illustrer comment on peut déterminer les valeurs des bornes inférieures sur le nombre de violations des contraintes souples d'*équilibre* et du *nombre*, considérons les instances dans les ensembles P_1 et P_2 décrites au paragraphe 4.2.1. Pour les instances dans P_1, le nombre de juges disponibles pour chaque ronde est $|I_p| = 4M$, M étant le nombre de matchs par ronde. Il est alors clair que, pour chaque ronde, le nombre de matchs avec 5 juges ne peut dépasser $\lfloor \frac{|I_p|-3M}{2} \rfloor = \lfloor \frac{M}{2} \rfloor$, et donc, celui des matchs avec 3 juges ne peut être inférieur à $(M - \lfloor \frac{M}{2} \rfloor)$. Par conséquent, $5(M - \lfloor \frac{M}{2} \rfloor)$ constitue une borne inférieure sur le nombre de violations de la contrainte du *nombre* (5 étant le nombre de rondes). D'autre part, rappelons que pour ces instances, les $4M$ juges disponibles pour chaque ronde sont répartis comme suit : M *juges en chef*, $2M$ juges de la catégorie *Expert*, et M juges de la catégorie *Nouveau*. Étant donnée une ronde p, supposons que 3 juges sont affectés à $(M - \lfloor \frac{M}{2} \rfloor)$ matchs, et que dans chacun de ces matchs, on retrouve un *juge en chef*, un juge de la catégorie *Expert*, et un juge de la catégorie *Nouveau* (i.e., la contrainte d'*équilibre* est respectée dans ces matchs). Alors, la plus petite valeur que pourrait prendre le nombre de violations de la contrainte d'*équilibre* dans cette ronde serait égale à $\lfloor \frac{M}{2} \rfloor$ si dans chacun des matchs avec 5 juges on retrou-

vait un *juge en chef*, 3 juges de la catégorie *Expert*, et un juge de la catégorie *Nouveau*. Par conséquent, $5\lfloor\frac{M}{2}\rfloor$ constitue une borne inférieure sur le nombre de violations de la contrainte d'*équilibre*.

Considérons maintenant les instances dans P_2. Puisque pour chaque ronde le nombre de juges disponibles est suffisant pour affecter 5 juges à chaque match ($|I_p| = 5M$), et puisque ces juges sont répartis comme suit : *M juges en chef*, $2M$ juges de la catégorie *Expert*, et $2M$ juges de la catégorie *Nouveau*, alors les bornes inférieures sur le nombre de violations des contraintes d'*équilibre* et du *nombre* correspondent à la valeur 0.

Nous utilisons l'ensemble des instances de test décrites au paragraphe 4.2.1 pour évaluer et comparer les deux variantes **RT-Div-Partiel** et **RT-Int-Partiel**. Chaque instance est résolue 10 fois avec une valeur d'initialisation différente pour le générateur des nombres aléatoires. Afin de disposer d'une plateforme de comparaison objective, les mêmes valeurs sont utilisées pour chaque variante.

Avant d'analyser les performances des deux variantes, il convient de préciser que pour toutes les instances de test que nous considérons, aucune solution entière n'a pu être identifiée avec CPLEX dans une limite de temps de 10 heures. L'absence de tels résultats proscrit donc toute analyse qualitative absolue. Par conséquent, tout au long de ce chapitre, pour évaluer la qualité des solutions générées par les différentes méthodes que nous proposons, nous les comparons avec la solution de référence décrite plus haut (solution «optimale»). De plus, puisque pour certaines instances nous disposons de la valeur optimale de la relaxation linéaire, nous considérons également (lorsque cela est possible) l'écart relatif entre la valeur des solutions générées et cette valeur. En effet, cette dernière constitue une meilleure borne inférieure sur la valeur optimale (comparativement à la valeur de la solution «optimale»). Elle nous permet donc d'évaluer de façon plus précise la qualité des solutions obtenues. Notons que ce critère est utilisé dans [88] entre autres.

La comparaison des deux variantes se fait donc à l'aide des cinq critères suivants :

1. l'écart moyen entre les solutions générées et les solutions «optimales». Pour cela nous procédons comme suit :
   - Pour chaque instance, nous considérons l'ensemble des 10 solutions générées et

nous calculons la moyenne des violations de chaque règle souple d'affectation

- Pour chaque instance et chaque règle souple d'affectation, nous évaluons la déviation de la moyenne du nombre de violations (que nous avons calculée) par rapport au nombre «optimal» de violations (i.e., nombre de violations de la règle d'affectation considérée dans la solution «optimale»)
- Enfin, pour chaque règle souple d'affectation, une moyenne globale de déviation est calculée pour chaque sous-ensemble de problèmes (rappelons que nous disposons de 9 sous-ensembles de problèmes et que chacun comprend 10 instances). Nous dénotons par $Ave\,dev - Nbre$, $Ave\,dev - Seq$, $Ave\,dev - Coup$, $Ave\,dev - Div$, $Ave\,dev - Aff$, et $Ave\,dev - Equ$ respectivement, les moyennes de déviation associées aux contraintes du *nombre*, de *séquence*, de *couplage*, de *diversité*, d'*affiliation*, et d'*équilibre*

2. *%AveGap*, l'écart relatif moyen entre les solutions générées et la solution optimale de la relaxation linéaire. Nous évaluons d'abord la valeur de cet écart pour chaque instance. La formule utilisée est la suivante :

$$\frac{Ave\, z_H - z_{RL}}{z_{RL}} \text{x} 100$$

où $Ave\, z_H$ est la valeur moyenne des solutions générées par la variante en considérant les 10 résolutions, et z_{RL} est la valeur optimale de la relaxation linéaire. Une moyenne globale est ensuite calculée pour chaque sous-ensemble de problèmes. Notons que z_H et z_{RL} sont évaluées en utilisant les poids du modèle MG[1] tels que définis à la fin du paragraphe 4.1.2 ;

3. *%Feas*, le pourcentage des solutions réalisables obtenues ;
4. *%Opt*, le pourcentage des solutions «optimales» obtenues ;
5. *AveCPU*, le temps moyen de résolution nécessaire pour générer les solutions. Il est indiqué en secondes.

Les résultats des expérimentations numériques sont résumés dans le tableau 4.5. Pour chaque sous-ensemble de problèmes et chaque variante, nous indiquons les valeurs obtenues pour les différents critères de comparaison susmentionnés. Notons que la moyenne

de déviation associée à la contrainte du *nombre* (*Ave dev – Nbre*) ne figure pas dans ce tableau car elle est toujours nulle (i.e., dans toutes les solutions générées par les deux variantes, 5 juges sont affectés au maximum de matchs). Le critère *%Feas* n'apparaît pas non plus dans ce tableau car toutes les solutions générées par les deux variantes sont toujours réalisables. Quant au critère *%Ave Gap*, pour certains sous-ensembles de problèmes, la valeur reportée est NA. Ceci indique que la valeur optimale de la relaxation linéaire z_{RL} est nulle pour toutes les instances de ces sous-ensembles, et par conséquent, le critère *%Ave Gap* ne s'applique pas. Lorsqu'aucune valeur n'est reportée, cela signifie que nous ne disposons pas de la valeur optimale de la relaxation linéaire.
Nous donnons également, dans la dernière colonne du tableau 4.5, les résultats obtenus avec CPLEX. Pour cette colonne, nous indiquons dans les cellules correspondantes à *Ave CPU* le temps moyen de résolution de la relaxation linéaire. Lorsqu'aucun temps n'est reporté, cela signifie qu'aucune instance du sous-ensemble de problèmes n'a pu être résolue. Les autres critères ne s'appliquent pas à CPLEX comme indiqué par NA dans les cellules qui leur correspondent.

Si l'on effectue une comparaison entre CPLEX et les deux variantes, celles-ci le surclassent indiscutablement. En premier lieu, les deux variantes ont pu produire assez rapidement des solutions de très bonne qualité pour toutes les instances de test considérées alors qu'il a été impossible d'obtenir une solution entière avec CPLEX dans la limite de temps allouée (10 heures), et ce en dépit de toutes les simplifications apportées au modèle et malgré le fait que nous ayons considéré plusieurs combinaisons de paramètres. De plus, pour les gros problèmes avec 90 matchs par ronde, CPLEX n'a même pas été en mesure de résoudre la relaxation linéaire alors que les deux variantes ont toujours réussi à trouver des solutions «optimales» en quelques secondes.
Comparons les solutions produites par les deux variantes avec la solution optimale de la relaxation linéaire obtenue avec CPLEX. On constate que *%Ave Gap* n'excède que très exceptionnellement 1%. Cela signifie que les valeurs des solutions générées par les deux variantes sont très proches de la borne inférieure obtenue avec CPLEX, et met donc en évidence l'efficacité des deux variantes à produire des solutions d'excellente

		Taille	**RT-Div-Partiel**	**RT-Int-Partiel**	CPLEX
Ave dev – Seq	P_1	15	2,41	2,44	NA
		30	0	0	NA
		90	0	0	NA
	P_2	15	14,9	15,15	NA
		30	0	0	NA
		90	0	0	NA
Ave dev – Coup	P_1	15	2,3	0,6	NA
		30	0	0,01	NA
		90	0	0	NA
	P_2	15	15,71	15,11	NA
		30	0	0	NA
		90	0	0	NA
Ave dev – Div	P_1	15	5,92	3,77	NA
		30	0	0,03	NA
		90	0	0	NA
	P_2	15	1,61	0,67	NA
		30	0	0	NA
		90	0	0	NA
Ave dev – Aff	P_1	15	3,19	0,39	NA
		30	0	0	NA
		90	0	0	NA
	P_2	15	7,65	4,01	NA
		30	0	0,04	NA
		90	0	0	NA
Ave dev – Equ	P_1	15	4,15	0,15	NA
		30	0	0	NA
		90	0	0	NA
	P_2	15	40,80	33,68	NA
		30	0	0,84	NA
		90	0	0	NA
%Ave Gap	P_1	15	7,01E-03	7,19E-03	NA
		30	0	1,48E-08	NA
		90	-	-	NA
	P_2	15	0,18	1,73	NA
		30	NA	NA	NA
		90	-	-	NA
%Opt	P_1	15	0	0	NA
		30	100	96	NA
		90	100	100	NA
	P_2	15	0	0	NA
		30	100	71	NA
		90	100	100	NA
Ave CPU (sec)	P_1	15	180,35	180,23	982,46
		30	4,74	18,94	893,05
		90	11,72	11,69	-
	P_2	15	225,59	225,51	350,83
		30	16,72	163,56	2 400,34
		90	22,26	22,47	-

Tableau 4.5 – Comparaison de l'efficacité des variantes utilisant la RT en fonction des sous-ensembles de problèmes

qualité. Également, le temps de résolution de celles-ci est uniformément inférieur à celui de CPLEX. En particulier, si l'on considère les instances avec 30 matchs par ronde, le temps de résolution est réduit en moyenne d'un facteur de 153 pour la variante **RT-Div-Partiel**. De plus, toutes les solutions générées par cette variante sont «optimales» et bien entendu entières.

En somme, les deux variantes permettent de générer des solutions de très bonne qualité. La perte d'optimalité (parfois) est largement compensée par l'excellente performance au niveau du temps de résolution et de la taille des problèmes pouvant être résolus.

Comparons maintenant la variante **RT-Div-Partiel** avec la variante **RT-Int-Partiel**. L'évaluation de la qualité des solutions se fera par rapport aux solutions «optimales».

À la lecture des résultats présentés dans le tableau 4.5, il ressort que **RT-Div-Partiel** domine **RT-Int-Partiel**. Globalement, on observe une réduction du nombre de violations des règles souples d'affectation les plus prioritaires. Le pourcentage des solutions «optimales» obtenues est augmenté en moyenne de 5,50%, et le temps moyen de résolution est réduit d'un facteur de 1,35.

Examinons maintenant les résultats par rapport aux ensembles de problèmes (en fixant la taille des problèmes). Nous nous intéressons ici uniquement à la qualité des solutions générées. Nous reviendrons un peu plus loin pour discuter des temps de résolution.

Considérons d'abord les instances avec 15 matchs par ronde. On constate qu'il est plus difficile de satisfaire les règles souples d'affectation pour les instances dans P_2 que pour celles dans P_1, ce qui est logique. D'abord, pour les deux sous-ensembles, le fait que les mêmes juges soient disponibles pour toutes les rondes, et que le nombre de matchs auquel chaque juge est admissible soit réduit (rappelons que chaque juge ne peut être affecté aux matchs impliquant 4 équipes différentes soit environ 46% des matchs de chaque ronde), explique en grande partie le fait que le nombre de violations des différentes règles souples d'affectation soit élevé. D'un autre côté, pour les instances dans P_2, le nombre de juges disponibles pour chaque ronde permet d'affecter exactement 5 juges à chaque match alors que pour celles dans P_1, certains matchs comptent 3 juges affectés seulement (cf. tableau 4.3). Il est clair qu'il est beaucoup plus difficile de satisfaire

les règles souples d'affectation (autres que celle du *nombre*) lorsqu'il s'agit d'affecter 5 juges à un match que lorsqu'il faut en affecter 3 seulement (rappelons que la contrainte du *nombre*, stipulant qu'il faut maximiser le nombre de matchs avec 5 juges affectés, est plus prioritaire que toutes les autres règles souples d'affectation). Les instances dans P_2 sont donc plus difficiles à résoudre que celles dans P_1. Ceci expliquerait le fait que la différence entre les deux variantes **RT-Div-Partiel** et **RT-Int-Partiel** soit plus prononcée pour les instances dans le premier sous-ensemble que pour celles dans le second. En effet, en termes de *%Ave Gap*, l'écart entre les deux variantes est quasiment nul pour les instances dans P_1 alors qu'il est d'un facteur de 9,61 pour celles dans P_2.

On peut faire le même constat en comparant les solutions obtenues par les deux variantes pour les instances dans les deux sous-ensembles avec 30 matchs par ronde : la différence entre les deux variantes est plus prononcée pour les instances dans P_2 que pour celles dans P_1. En particulier, si l'on considère le pourcentage des solutions «optimales» obtenues, la différence est de 29% pour le premier sous-ensemble alors qu'elle est seulement de 4% pour le second.

Par ailleurs, il est intéressant de noter que le nombre de violations des règles souples d'affectation diminue avec la taille des problèmes, ce qui était tout à fait prévisible. En effet, comme nous l'avons déjà souligné au paragraphe 4.2.1, plus le nombre de matchs par ronde est élevé, moins il y a de restrictions au niveau de l'admissibilité (le pourcentage de matchs auxquels chaque juge est admissible est élevé). Il est alors plus facile de satisfaire les règles souples d'affectation. Ceci explique le fait que toutes les solutions obtenues par les deux variantes pour les instances avec 90 matchs par ronde ne comportent aucune violation et confirme le caractère «facile» de ces instances. Il est donc tout à fait logique que les solutions obtenues par les deux variantes soient similaires aussi bien pour les instances dans P_1 que pour celles dans P_2.

Analysons plus soigneusement les résultats relativement à la taille des problèmes. La figure 4.4 illustre l'évolution de l'écart moyen entre la valeur des solutions obtenues et la valeur «optimale», et l'évolution du temps moyen de résolution en fonction du nombre de matchs par ronde. Notons que la valeur d'une solution n'est rien d'autre que la somme pondérée du nombre de violations des différentes règles d'affectation. La

valeur «optimale» représente la valeur de la solution «optimale».

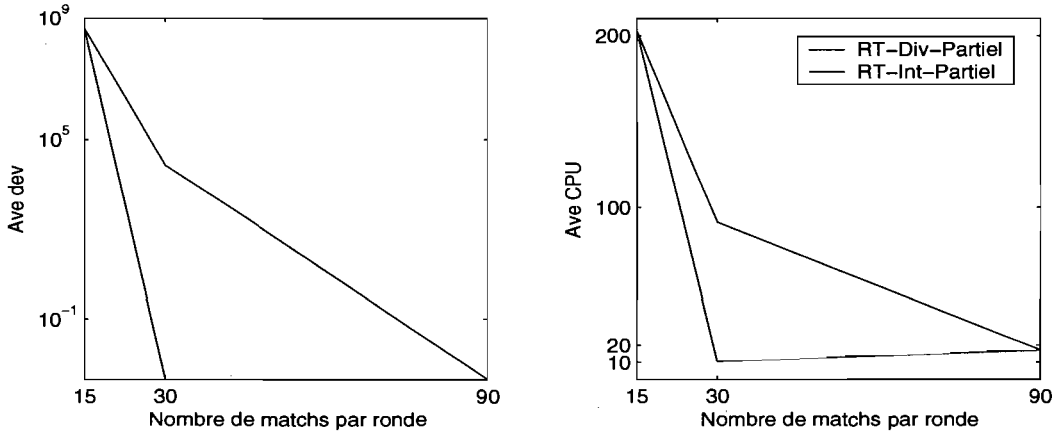

Figure 4.4 – Comparaison de l'efficacité des variantes utilisant la RT en fonction de la taille des problèmes

Comme nous l'avons mentionné plus haut, les instances avec 90 matchs par ronde ne semblent pas poser de problèmes à aucune des deux variantes. Considérons les autres instances avec 15 et 30 matchs par ronde. Si l'on s'intéresse à la qualité des solutions générées (cf. premier graphique), on constate que **RT-Div-Partiel** domine uniformément **RT-Int-Partiel**. Si l'on s'attache au temps de résolution (cf. deuxième graphique), on remarque que la variante **RT-Div-Partiel** se démarque pour les instances avec 30 matchs par ronde, vraisemblablement en raison du fait qu'un plus grand nombre de solutions «optimales» est obtenu (cf. tableau 4.5). En effet, rappelons que le critère d'arrêt des différentes variantes est spécifié en termes d'un temps maximal de résolution *tempsmax*. Elles s'arrêtent également dès qu'une solution «optimale» est générée. Ainsi, plus le pourcentage des solutions «optimales» obtenues est élevé, plus le temps moyen de résolution est réduit.

La même raison est valable pour justifier l'irrégularité de l'écart entre les deux variantes. En effet, plus la différence en termes de *%Opt* est importante, plus l'écart en termes de *Ave CPU* est prononcé. Ainsi, pour les instances avec 15 matchs par ronde, les temps de résolution des deux variantes sont similaires car, pour l'ensemble des résolutions, ni

l'une ni l'autre n'a pu identifier une solution «optimale» (cf. tableau 4.5), et par conséquent, elles se sont toujours arrêtées après *tempsmax* secondes.

Maintenant, pour les deux variantes, on note que les temps de résolution n'augmentent pas vraiment en fonction de la taille des problèmes, en partie, parce qu'il est plus facile d'obtenir des solutions «optimales» pour les instances de grande taille que pour celles de petite taille. En effet, comme nous venons de le mentionner, il existe une forte corrélation entre le *%Opt* et le *Ave CPU*. Or, comme nous l'avons déjà souligné, la façon avec laquelle les instances sont générées fait en sorte qu'il est plus facile de satisfaire les règles souples d'affectation lorsque le nombre de matchs est élevé. Rappelons que la solution que nous qualifions d'«optimale» est une solution où toutes les règles d'affectation sont respectées. Il est donc tout à fait logique que les temps de résolution soient moindres lorsqu'il est plus facile d'obtenir des solutions «optimales».

Les conclusions que nous venons d'ébaucher sont basées sur les valeurs moyennes (*Ave dev* et *Ave CPU*). Pour approfondir la comparaison, nous avons effectué le test de WILCOXON pour échantillons appariés sur les résultats de l'ensemble des résolutions. Comme au chapitre 3, pour chaque paire de variantes (A,B), nous avons testé l'hypothèse nulle selon laquelle les performances des deux variantes (qualité des solutions générées ou temps de résolution) sont identiques versus l'hypothèse alternative que la variante A est meilleure que la variante B. Les résultats de ce test sont résumés dans le tableau 4.6. Pour chaque entrée (A,B), nous indiquons la *p_valeur* obtenue. Le seuil de confiance utilisé étant de 0,05, une *p_valeur* inférieure à 0,05 indique que la variante A (dont le nom figure à la colonne 1 du tableau) domine la variante B (dont le nom figure à la ligne 2 du tableau). À l'opposé, une *p_valeur* supérieure à 0,05 indique que l'hypothèse nulle n'est pas rejetée, et donc, il n'y a pas de différence statistiquement significative entre les deux variantes A et B. Nous indiquons en caractère gras de telles valeurs.

	Qualité de la solution RT-Int-Partiel	**Temps de résolution RT-Int-Partiel**
RT-Div-Partiel	**0,98**	0,01

Tableau 4.6 – Résultats du test de WILCOXON pour les variantes utilisant la RT

Les résultats obtenus nous permettent de conclure, qu'au niveau de la qualité des solutions, il n'y a pas de différence statistiquement significative entre les deux variantes **RT-Div-Partiel** et **RT-Int-Partiel**. Cependant, si l'on tient compte du temps de résolution, **RT-Div-Partiel** domine.

Comme nous l'avons déjà mentionné, les instances avec 90 matchs par ronde ne semblent pas poser de problèmes à aucune des deux variantes. Celles avec 30 matchs par ronde semblent être assez faciles à résoudre. Aussi, pour illuster le comportement des deux variantes, présentons-nous dans la figure 4.5 l'évolution de *Ave dev* en fonction du temps lors de la résolution des instances avec 15 matchs par ronde. Comme au chapitre précédent, chaque courbe est associée à une variante et montre les valeurs moyennes de *Ave dev* durant les 10 résolutions de ces instances, valeurs calculées à différents moments de la résolution.

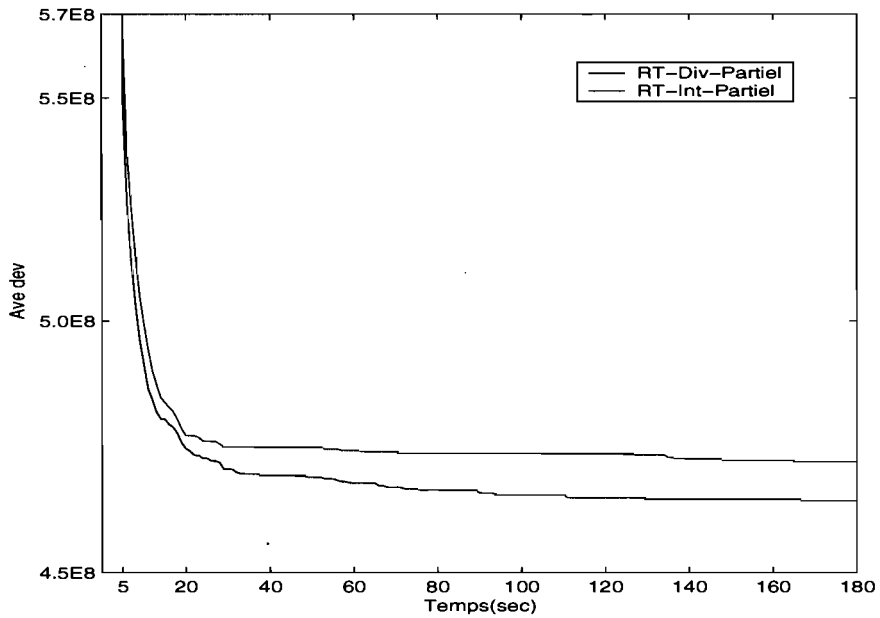

Figure 4.5 – Évolution de *Ave dev* des variantes utilisant la RT pour les problèmes avec 15 matchs par ronde

À quelques exceptions près, les deux variantes ont le même comportement. Durant les 5 premières secondes, leurs courbes sont confondues. Cela est certainement dû au fait

que les deux variantes sont équivalentes. En effet, durant cette période, la solution comporte un grand nombre de violations des différentes règles d'affectation (rappelons que la solution initiale est générée aléatoirement). Par conséquent, des améliorations sont régulièrement obtenues. Dès lors, *xbestG* n'est rien d'autre que *xbestLast* et les deux variantes obtiennent les mêmes résultats.

Sur l'intervalle [5, 20], les valeurs de *Avedev* continuent à décroître avec toutefois un léger avantage en faveur de **RT-Div-Partiel**. L'écart entre les deux variantes s'accentue sur l'intevalle [20, 40]. En effet, **RT-Int-Partiel** parvient rarement à améliorer la valeur de *Avedev*. Le fait de concentrer l'exploration autour de *xbestG* semble piéger la recherche autour des optima locaux.

RT-Div-Partiel maintient son avance sur **RT-Int-Partiel** durant le reste du temps de résolution. En effet, après la $40^{ème}$ seconde, les améliorations obtenues par les deux variantes sont marginales et reflètent le plus souvent une diminution du nombre de violations de la contrainte d'*équilibre*. On note toutefois que **RT-Div-Partiel** parvient plus souvent à améliorer la valeur de *Avedev* que **RT-Int-Partiel**.

4.5 Variantes utilisant la méthode de recherche à voisinage variable

Cette section est dédiée aux variantes utilisant la recherche à voisinage variable. La résolution des sous-problèmes associés aux rondes se base sur le modèle MG_p^2 présenté au paragraphe 4.4.1. Nous décrivons d'abord les caractéristiques des variantes considérées, puis présentons les résultats des expérimentations numériques.

4.5.1 Caractéristiques de ces variantes

Ces variantes suivent le même schéma algorithmique que celui des variantes utilisant la méthode de recherche avec tabous. La phase 1 est identique. La phase 2 diffère puisque la méthode de recherche à voisinage variable (RVV) est utilisée pour résoudre les sous-problèmes associés aux rondes. De plus, lorsque nous considérons une ronde $p \in \Omega$, nous passons directement à la recherche à voisinage variable sans appliquer une stratégie de diversification d'abord. En effet, la phase perturbation de la RVV peut être assimilée à

une diversification.

Pour résoudre le sous-problème associé à la ronde p, nous pouvons utiliser la méthode décrite à la section 3.5, car seule la fonction objectif de MG_p^2 est différente de celle de M^2. Puisque la RVV explore séquentiellement les voisinages proches, puis de plus en plus éloignés de la meilleure solution connue (cf. figure 3.15), alors seule $xbestG_p$ (i.e., les affectations courantes que nous retrouvons dans $xbestG$) peut servir de solution initiale pour amorcer la résolution du sous-problème. Ainsi, seule cette façon de définir x_p^0 est considérée dans ces variantes. Par contre, il est possible de considérer les deux façons pour fixer les affectations des autres rondes $q \neq p$ (leurs valeurs courantes dans $xbestG$ ou dans $xbestLast$). Également, puisque la stratégie de perturbation, utilisée pour générer les solutions de départ dans les différents voisinages $V_k(xbestG_p)$, ne repose pas sur les violations, il est possible de considérer les deux façons de définir l'ensemble Ω. Cependant, on peut noter d'emblée que la combinaison ($\{1,\ldots,P\}$, $xbestG$, RVV) affichera de mauvaises performances, et par conséquent, il est inutile de la considérer. En effet, si la solution de la ronde p est «optimale» (i.e, dans tous les matchs de p, toutes les règles d'affectation sont respectées), alors la solution globale du problème obtenue à l'issue de la résolution du sous-problème associé à p ne pourra être meilleure que $xbestG$. Ainsi, lorsque nous entamerons la résolution du sous-problème suivant, nous considérerons la même solution $xbestG$ dont nous disposions avant de résoudre le sous-problème associé à p car celle-ci n'aura pas changé. Le fait de considérer les rondes «optimales» ne présente donc aucun intérêt lorsque nous utilisons $xbestG$. Bien au contraire, cela représente une perte de temps puisqu'on disperse inutilement les efforts en explorant de nombreuses régions sans se concentrer sur les régions les plus prometteuses (rondes comportant des violations que l'on pourrait éventuellement améliorer ce qui permettrait de changer $xbestG$). Qui plus est, le critère d'arrêt de la procédure est spécifié en termes d'un temps maximal alloué $tempsmax$.

4.5.2 Expérimentations numériques

Nous dénotons par **RVV-Div-Partiel**, **RVV-Int-Partiel** et **RVV-Div-Global** les trois variantes implémentées. **Div** et **Int** font référence à la solution utilisée lors de la réso-

lution du sous-problème associé à la ronde p pour fixer les affectations dans les rondes $q \neq p$ (*xbestLast* et *xbestG* respectivement). **Partiel** et **Global** indiquent la façon utilisée pour spécifier l'ensemble Ω des rondes à considérer à chaque itération majeure de la phase 2 (les rondes comportant des violations et toutes les rondes respectivement).

Les paramètres de la recherche à voisinage variable sont fixés comme dans le paragraphe 3.5.4, et, à l'instar des variantes utilisant la recherche avec tabous, ici aussi un temps maximal $tempsmax = 3N$ secondes est alloué à chaque variante (N étant le nombre total de juges).

Nous reprenons l'ensemble des instances de test décrites au paragraphe 4.2.1 pour évaluer et comparer les trois variantes. Chaque instance est résolue 10 fois avec une valeur d'initialisation différente pour le générateur des nombres aléatoires. Notons que les mêmes valeurs sont utilisées pour chaque variante. De plus, ces valeurs sont identiques à celles employées pour les variantes utilisant la recherche avec tabous.

La comparaison se fait à l'aide des cinq critères utilisés dans le paragraphe 4.4.5 à savoir, l'écart moyen entre les solutions générées et les solutions «optimales» en termes du nombre de violations des différentes règles souples d'affectation ($Ave\,dev - Nbre$, $Ave\,dev - Seq$, $Ave\,dev - Coup$, $Ave\,dev - Div$, $Ave\,dev - Aff$ et $Ave\,dev - Equ$), l'écart relatif moyen entre les solutions générées et la solution optimale de la relaxation linéaire ($\%Ave\,Gap$), le pourcentage des solutions réalisables obtenues ($\%Feas$), le pourcentage des solutions «optimales» trouvées ($\%Opt$), et le temps moyen de résolution en secondes ($Ave\,CPU$). Les résultats obtenus pour chaque sous-ensemble de problèmes sont présentés dans le tableau 4.7. Celui-ci a la même structure que le tableau 4.5 présenté au paragraphe 4.4.5. Là encore, les deux critères $Ave\,dev - Nbre$ et $\%Feas$ ne figurent pas dans le tableau puisque leurs valeurs sont toujours égales à 0 et 100% respectivement (i.e., dans toutes les solutions générées par les trois variantes, 5 juges sont affectés au maximum de matchs, et toutes ces solutions sont réalisables). La figure 4.6 illustre l'évolution de l'écart moyen entre la valeur des solutions obtenues et la valeur «optimale», et l'évolution du temps moyen de résolution en fonction de la taille des problèmes.

Si l'on effectue une comparaison entre CPLEX et les trois variantes, celles-ci le surclassent indiscutablement indiquant encore une fois qu'il est plus avantageux d'uti-

		Taille	**RVV-Div-Partiel**	**RVV-Int-Partiel**	**RVV-Div-Global**	CPLEX
Avedev − Seq	P_1	15	2,30	2,62	2,30	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	14,90	15,37	14,90	NA
		30	0	0	0	NA
		90	0	0	0	NA
Avedev − Coup	P_1	15	0,05	0,61	0,04	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	14,57	14,93	14,57	NA
		30	0	0	0	NA
		90	0	0	0	NA
Avedev − Div	P_1	15	3,64	3,13	3,68	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	1,28	0,24	1,28	NA
		30	0	0	0	NA
		90	0	0	0	NA
Avedev − Aff	P_1	15	2,08	0,44	2,33	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	6,75	1,66	6,75	NA
		30	0	0,04	0	NA
		90	0	0	0	NA
Avedev − Equ	P_1	15	2,96	0,02	3,16	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	38,98	18,64	38,96	NA
		30	0,82	0,72	4,28	NA
		90	0	0	0	NA
%AveGap	P_1	15	6,25E-03	8,38E-03	6,25E-03	NA
		30	0	0	0	NA
		90	-	-	-	NA
	P_2	15	0,17	3,32	0,17	NA
		30	NA	NA	NA	NA
		90	-	-	-	NA
%Opt	P_1	15	5	0	3	NA
		30	100	100	100	NA
		90	100	100	100	NA
	P_2	15	0	0	0	NA
		30	75	72	8	NA
		90	100	100	100	NA
AveCPU (sec)	P_1	15	174,89	180,19	176,66	982,46
		30	5,18	12,46	6,19	893,05
		90	11,87	11,85	11,97	-
	P_2	15	225,45	225,25	225,45	350,83
		30	130,08	154,88	416,71	2 400,34
		90	22,60	22,65	22,86	-

Tableau 4.7 – Comparaison de l'efficacité des variantes utilisant la RVV en fonction des sous-ensembles de problèmes

liser l'approche métaheuristique. En effet, les faibles valeurs de *%Ave Gap* indiquent l'excellente qualité des solutions générées par les trois variantes. De plus, les temps de résolution de celles-ci sont sans commune mesure avec ceux de CPLEX. En particulier, pour les problèmes du sous-ensemble P_1 avec 30 matchs par ronde, les valeurs nulles de *%Ave Gap* confirment que toutes les solutions générées par les trois variantes sont optimales. *Ave CPU* est pour sa part réduit d'un facteur de 172, 144, et 71 respectivement pour les variantes **RVV-Div-Partiel**, **RVV-Div-Global**, et **RVV-Int-Partiel**. Rappelons que, pour CPLEX, *Ave CPU* indique le temps moyen requis pour résoudre la relaxation linéaire et qu'aucune solution entière n'a pu être identifiée par le solveur dans une limite de temps de 10 heures.

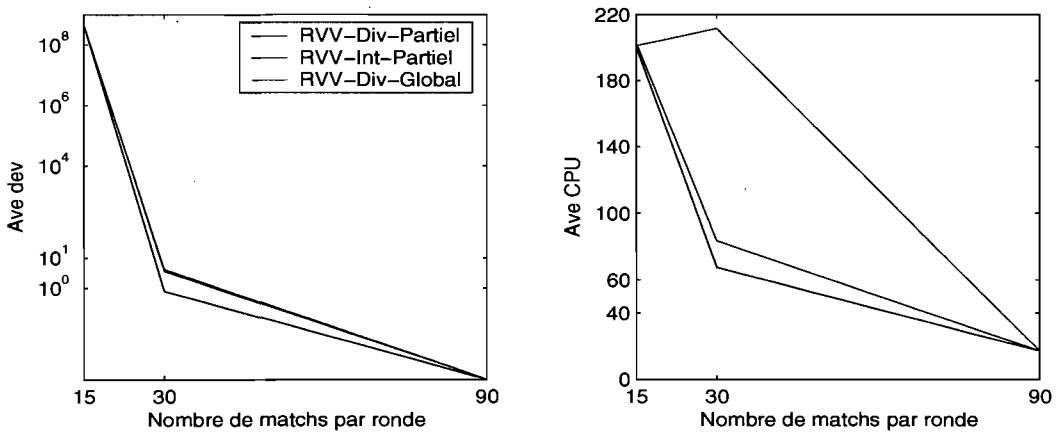

Figure 4.6 – Comparaison de l'efficacité des variantes utilisant la RVV en fonction de la taille des problèmes

Comparons maintenant les trois variantes entre elles. Globalement, **RVV-Div-Partiel** semble la plus performante. Comparativement aux deux autres variantes **RVV-Int-Partiel** et **RVV-Div-Global**, le pourcentage des solutions «optimales» obtenues est augmenté en moyenne de 1,33% et 11,50% respectivement. Quant au temps de résolution, il est réduit en moyenne d'un facteur de 1,06 et 1,50 respectivement. Comme on peut le constater sur la figure 4.6, sa supériorité est beaucoup plus significative pour les instances avec 30 matchs par ronde. Nous expliquerons un peu plus loin pourquoi.

Comparons maintenant la variante **RVV-Int-Partiel** avec la variante **RVV-Div-Global**. Analysons les résultats relativement à la taille des problèmes. Considérons d'abord ceux des instances avec 15 matchs par ronde. Si l'on s'intéresse à la qualité des solutions générées, on voit clairement que les performances de **RVV-Int-Partiel** se situent nettement en retrait par rapport à celles de **RVV-Div-Global** (cf. tableau 4.7). En particulier, pour les instances dans P_2 (qui semblent être les plus difficiles à résoudre), *%Ave Gap* est beaucoup plus élevé (d'un facteur de 19,53). Par contre, si l'on s'attache au temps de résolution, on note que les performances des deux variantes sont très comparables : pour les instances dans P_2, elles sont similaires alors que pour celles dans P_1, **RVV-Div-Global** est légèrement meilleure. Ceci tient au fait que **RVV-Div-Global** a été en mesure de générer quelques solutions «optimales» pour certaines instances dans P_1 alors que pour toutes les instances dans P_2, aucune solution «optimale» n'a été générée par les deux variantes. (Rappelons que les différentes variantes s'arrêtent lorsqu'une solution «optimale» est générée ou lorsque le temps de résolution maximal *tempsmax* s'est écoulé.)

Considérons maintenant les instances avec 30 matchs par ronde. Au niveau de la qualité des solutions, la variante **RVV-Int-Partiel** est toujours celle qui affiche les plus mauvaises performances. Bien qu'elle soit meilleure que **RVV-Div-Global** en termes de $\%Opt$, il n'en demeure pas moins qu'en termes de la valeur moyenne des solutions générées, cette dernière est meilleure. En effet, si l'on se réfère aux valeurs de $Ave\,dev-Aff$ indiquées dans le tableau 4.7, on note que certaines solutions générées par **RVV-Int-Partiel** comptent des violations de la contrainte d'*affiliation*, ce qui n'est pas le cas pour **RVV-Div-Global**. (Rappelons que la contrainte d'*affiliation* est plus prioritaire que celle d'*équilibre* et donc, la valeur du poids qui lui est associé est plus élevée.) Au niveau du temps de résolution, les résultats sont partagés. Pour les instances dans P_1, **RVV-Int-Partiel** est la plus lente. Le contraire se produit pour celles dans P_2. En effet, comme nous venons de le mentionner, **RVV-Int-Partiel** trouve un plus grand nombre de solutions «optimales». Il est donc tout à fait logique que ses temps de résolution soient moindres.

Finalement, pour les instances avec 90 matchs par ronde, les résultats obtenus par les

deux variantes sont très comparables tant au niveau de la qualité des solutions générées qu'au niveau du temps de résolution. Ceci tient au fait que ces problèmes sont faciles à résoudre comme cela a déjà été souligné au paragraphe 4.4.5. Du coup, les deux variantes obtiennent très rapidement des solutions «optimales».

Comparons maintenant les deux variantes **RVV-Div-Partiel** et **RVV-Div-Global**. Considérons d'abord les instances avec 15 matchs par ronde. On constate que les résultats obtenus par les deux variantes sont très similaires. Une explication plausible réside dans le fait qu'il est difficile voire impossible de satisfaire toutes les règles souples d'affectation pour la quasi-totalité de ces instances (comme il a déjà été souligné au paragraphe 4.4.5). De plus, si l'on observe de plus près les meilleures solutions obtenues par les deux variantes, on constate que pour la majorité des instances considérées, toutes les rondes comportent des violations d'au moins une des règles d'affectation. Dès lors, les deux variantes **RVV-Div-Partiel** et **RVV-Div-Global** sont équivalentes et obtiennent les mêmes résultats. Concernant les autres instances pour lesquelles les deux variantes ne sont pas équivalentes (i.e., dans les meilleures solutions obtenues, certaines rondes ne comportent aucune violation), **RVV-Div-Global** est légèrement meilleure que **RVV-Div-Partiel** (cf. valeurs de $Ave\,dev - Coup$ pour P_1 et $Ave\,dev - Eq$ pour P_2). Ainsi, pour ces instances, le fait de considérer toutes les rondes à chaque itération majeure de la phase 2 semble bénéfique : limiter l'ensemble Ω aux rondes comportant des violations semble limiter l'exploration autour d'un optimum local (surtout lorsque le nombre de ces rondes est réduit).

Considérons maintenant les instances avec 30 matchs par ronde. Si l'on s'intéresse à la qualité des solutions générées, on constate que **RVV-Div-Partiel** est légèrement meilleure que **RVV-Div-Global**. Cette baisse de performance de **RVV-Div-Global** peut être justifiée par le temps passé à résoudre les sous-problèmes associés aux rondes ne comportant aucune violation. En effet, comme nous l'avons déjà mentionné au paragraphe 4.4.5, les instances avec 30 matchs par ronde sont assez faciles à résoudre. Ainsi, après quelques itérations de la phase 2, $xbestG$ comporte seulement quelques violations de la contrainte d'*équilibre* constatées le plus souvent dans une seule ronde. Il n'est donc pas nécessaire de considérer toutes les rondes à chacune des itérations suivantes

pour améliorer *xbestG*. Bien au contraire, ceci retarde la convergence vers les solutions «optimales». Nous retrouvons les mêmes constatations si l'on s'intéresse au temps de résolution : **RVV-Div-Partiel** affiche de meilleures performances que **RVV-Div-Global**. Les mêmes raisons évoquées ci-dessus sont valables pour justifier les résultats obtenus. La différence entre les deux variantes est plus prononcée pour les instances dans P_2 que pour celles dans P_1, car l'écart en termes de *%Opt* est plus important dans le premier cas que dans le second.

Le fait de considérer toutes les rondes à chaque itération majeure de la phase 2 semble aussi avoir un impact négatif dans le cas des instances avec 90 matchs par ronde (qui semblent être les plus faciles à résoudre). En effet, quoique toutes les solutions générées par **RVV-Div-Global** soient «optimales», celle-ci se révèle la plus lente des trois variantes.

En somme, le fait de considérer toutes les rondes à chaque itération majeure de la phase 2 engendre parfois un gain qualitatif et contribue à s'échapper des optima locaux. Néanmoins, il présente l'inconvénient d'augmenter les temps de résolution lorsque les problèmes sont faciles à résoudre.

Pour poursuivre la comparaison des trois variantes, nous avons effectué le test de WILCOXON pour échantillons appariés sur les résultats de l'ensemble des résolutions. Pour chaque paire de variantes (A,B), nous avons testé l'hypothèse nulle selon laquelle les performances des deux variantes (qualité des solutions générées ou temps de résolution) sont identiques versus l'hypothèse alternative que la variante A est meilleure que la variante B. Les résultats de ce test sont résumés dans le tableau 4.8. Pour chaque entrée (A,B), nous indiquons la *p_valeur* obtenue. Nous soulignons cette valeur en caractère gras lorsque la différence entre A et B n'est pas statistiquement significative (i.e., si la *p_valeur* obtenue est supérieure au seuil de confiance utilisé qui est de 0,05). Une *p_valeur* inférieure à 0,05 révèle que la variante A (dont le nom figure à la colonne 1 du tableau lorsque nous comparons la qualité des solutions ou la colonne 4 lorsqu'il s'agit du temps de résolution) est statistiquement meilleure que la variante B (dont le nom figure à la ligne 2 du tableau).

Les résultats du test ne remettent pas en cause nos conclusions concernant la qualité des

	Qualité de la solution			Temps de résolution	
	RVV-Div-Global	RVV-Int-Partiel		RVV-Int-Partiel	RVV-Div-Global
RVV-Div-Partiel	3.87E-3	5.99E-14	**RVV-Div-Partiel**	**0,93**	2.06E-12
RVV-Div-Global	-	1.57E-9	**RVV-Int-Partiel**	-	<2.20E-16

Tableau 4.8 – Résultats du test de WILCOXON pour les variantes utilisant la RVV

solutions : la variante **RVV-Div-Partiel** se classe sans conteste en premier, **RVV-Div-Global** en second, et **RVV-Int-Partiel** en dernier. Au niveau du temps de résolution, les résultats du test indiquent qu'il n'y a pas de différence statistiquement significative entre **RVV-Div-Partiel** et **RVV-Int-Partiel**. Ces deux variantes surclassent **RVV-Div-Global**.

La figure 4.7 illustre le comportement de *Ave dev* durant la résolution des instances avec 15 matchs par ronde. Chaque courbe est associée à une variante et montre les valeurs moyennes de *Ave dev* durant les 10 résolutions de ces instances, valeurs calculées à différents moments de la résolution.

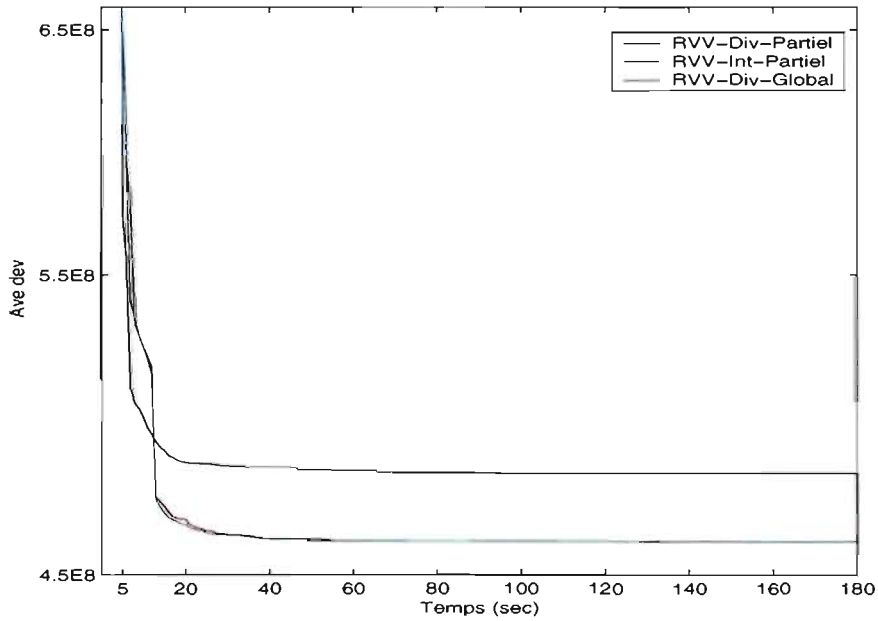

Figure 4.7 – Évolution de *Ave dev* des variantes utilisant la RVV pour les problèmes avec 15 matchs par ronde

Il n'y a pas de différence significative entre les trois variantes durant les premières se-

condes. Deux raisons viennent expliquer cette similitude : d'une part, puisqu'au début la solution est de mauvaise qualité et compte un grand nombre de violations des différentes règles d'affectation, des améliorations sont pratiquement obtenues chaque fois que l'on résoud un sous-problème associé à une ronde. Par conséquent, *xbestG* et *xbestLast* coïncident, et donc, **RVV-Int-Partiel** et **RVV-Div-Partiel** sont équivalentes. D'autre part, puisque ces violations sont présentes le plus souvent dans toutes les rondes, les deux façons de spécifier l'ensemble Ω concordent, ce qui implique que **RVV-Div-Partiel** et **RVV-Div-Global** sont équivalentes.

Les différences entre les trois variantes sont décelées environ vers la 5ème seconde. Dans le cas de **RVV-Int-Partiel**, la valeur moyenne de *Ave dev* continue à décroître rapidement. Elle surpasse les deux autres variantes. Cependant, dès la 10ème seconde, les améliorations sont très marginales, et on note le début d'une période de stagnation. De plus, après la 90ème seconde et jusqu'à la fin du processus, aucune amélioration n'est enregistrée, ce qui laisse supposer que cette variante est plus facilement piégée dans les optima locaux.

Cet aspect de stagnation que l'on constate précocement pour la variante **RVV-Int-Partiel**, apparaît plus tardivement pour les deux autres variantes. On peut également remarquer que leurs courbes stagnent à des valeurs moins élevées de *Ave dev* que **RVV-Int-Partiel**. Sur l'intervalle [10,60], **RVV-Div-Global** affiche des performances légèrement meilleures que **RVV-Div-Partiel**. L'écart entre les deux variantes est quasiment nul sur l'intervalle [60,180].

4.6 Variantes utilisant la méthode de recherche avec tabous à voisinages structurés

Cette section est consacrée aux variantes utilisant la méthode de recherche avec tabous à voisinages structurés. Après avoir décrit le modèle mathématique sur lequel elles se basent, nous présentons leurs caractéristiques. Nous détaillons par la suite leurs principales composantes. Nous terminons cette section par les résultats des expérimentations numériques.

4.6.1 Modèle mathématique

Nous proposons une version modifiée du modèle mathématique MG^1 décrit au paragraphe 4.1.1 où, pour chaque ronde p, le surplus de juges disponibles est affecté à un match fictif $(M+1)_p$. De plus, dans ce nouveau modèle, les matchs avec un seul juge affecté sont acceptés, mais engendrent un coût dont la pondération est beaucoup plus élevée ($w_7 \gg w_6$).

Pour formuler le modèle, nous introduisons les variables y^1_{jp} afin d'indiquer si un seul juge est affecté au match j de la ronde p :

$$y^1_{jp} = \begin{cases} 1 & \text{si un seul juge est affecté au match } j \text{ de la ronde } p \\ 0 & \text{sinon.} \end{cases}$$

Les contraintes stipulant qu'au moins un *juge en chef* et un autre juge *expert* doivent être affectés à chaque match de chaque ronde ne pouvant être satisfaites pour les matchs j tels que $y^1_{jp} = 1$, nous exigeons plutôt que le seul juge affecté à ces matchs soit un *juge en chef*. Nous devons donc modifier en conséquence les contraintes (4.4) du modèle MG^1. Ces contraintes sont remplacées par les deux ensembles de contraintes (4.70) et (4.71) :

$$\sum_{i=1}^{N} l_i x_{ijp} \geq y^1_{jp} \qquad \forall \quad j,p \tag{4.70}$$

$$\sum_{i=1}^{N} (5l_i + v_i) x_{ijp} \geq 6(y^3_{jp} + y^5_{jp}) \qquad \forall \quad j,p. \tag{4.71}$$

Ainsi, si un seul juge est affecté au match j de la ronde p, alors la contrainte (4.70) assure que c'est un *juge en chef* et la contrainte (4.71) est inactive (i.e., satisfaite automatiquement). Si 3 ou 5 juges sont affectés au match j de la ronde p, alors la contrainte (4.71) est équivalente à la contrainte (4.4) de MG^1 pour assurer qu'au moins un *juge en chef* et un autre juge *expert* sont affectés à ce match, et la contrainte (4.70) est inactive.

Le nouveau modèle mathématique MG^3 est résumé dans la figure 4.8. Pour retrouver le modèle mathématique du sous-problème associé à la ronde p, il suffit de fixer, dans MG^3, les variables associées aux rondes $q \neq p$ à leurs valeurs courantes, ce qui génère

$$\min \quad w_1 \sum_{p=1}^{P}\sum_{j=1}^{M}(d_{jp}^{1+}+d_{jp}^{1-}) + w_2 \sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{c=1}^{C} d_{cjp}^{2} + w_3 \sum_{p=1}^{P}\sum_{j=1}^{M}\sum_{k=1}^{K} d_{kjp}^{3} \quad (4.72)$$
$$+ w_4 \sum_{i=1}^{N-1}\sum_{r=i+1}^{N} d_{ir}^{4} + w_5 \sum_{t=1}^{2M}\sum_{i=1}^{N} d_{it}^{5} + w_6 \sum_{p=1}^{P}\sum_{j=1}^{M} y_{jp}^{3} + w_7 \sum_{p=1}^{P}\sum_{j=1}^{M} y_{jp}^{1}$$

(MG^3) **Sujet à**

$$\sum_{j=1}^{(M+1)_p} x_{ijp} = d_{ip} \qquad \forall \ i,p \qquad (4.73)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M}(1-a_{ijp})x_{ijp} = 0 \qquad \forall \ i \qquad (4.3)$$

$$\sum_{i=1}^{N} l_i x_{ijp} \geq y_{jp}^{1} \qquad \forall \ j,p \qquad (4.70)$$

$$\sum_{i=1}^{N}(5l_i+v_i)x_{ijp} \geq 6(y_{jp}^{3}+y_{jp}^{5}) \qquad \forall \ j,p \qquad (4.71)$$

$$\sum_{i=1}^{N} x_{ijp} = y_{jp}^{1} + 3y_{jp}^{3} + 5y_{jp}^{5} \qquad \forall \ j,p \qquad (4.74)$$

$$y_{jp}^{1} + y_{jp}^{3} + y_{jp}^{5} = 1 \qquad \forall \ j,p \qquad (4.75)$$

$$\sum_{i=1}^{N}(2l_i+2v_i-1)x_{ijp} - d_{jp}^{1+} + d_{jp}^{1-} = 1 \qquad \forall \ j,p \qquad (4.7)$$

$$\sum_{i=1}^{N} r_{ic}x_{ijp} - d_{cjp}^{2} \leq 1 \qquad \forall \ c,j,p \qquad (4.8)$$

$$\sum_{i=1}^{N} e_{ik}x_{ijp} + d_{kjp}^{3} \geq 1 \qquad \forall \ k,j,p \qquad (4.9)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M} x_{ijp}x_{rjp} - d_{ir}^{4} \leq 1 \qquad \forall \ i,r>i \qquad (4.10)$$

$$\sum_{p=1}^{P}\sum_{j=1}^{M} s_{tjp}x_{ijp} - d_{it}^{5} \leq 1 \qquad \forall \ i,t \qquad (4.11)$$

$$x_{ijp} = 0 \text{ ou } 1 \qquad \forall \ i,j,p \qquad (4.12)$$

$$y_{jp}^{1}, y_{jp}^{3}, y_{jp}^{5} = 0 \text{ ou } 1 \qquad \forall \ j,p \qquad (4.76)$$

$$d_{jp}^{1+}, d_{jp}^{1-}, d_{cjp}^{2}, d_{kjp}^{3}, d_{ir}^{4}, d_{it}^{5} \geq 0 \qquad \forall \ i,r>i,t,j,p,c,k \qquad (4.14)$$

Figure 4.8 – Modèle équivalent MG^3

un modèle MG_p^3. Les valeurs des poids $\{w_1, \ldots, w_7\}$ de MG_p^3 sont calculées en utilisant les équations (4.15). Ainsi,

$$\begin{aligned}
w_1 &= 1 \\
w_2 &= 4M+1 \\
w_3 &= (4M+1)^2 \\
w_4 &= (4M+1)^2((K-1)M+1) \\
w_5 &= (4M+1)^2((K-1)M+1)(10M+1) \\
w_6 &= (4M+1)^2((K-1)M+1)(10M+1)^2 \\
w_7 &= (4M+1)^2((K-1)M+1)(10M+1)^2(M+1).
\end{aligned}$$

Il est intéressant de noter que MG_p^3 est une généralisation du modèle M^3 (cf. paragraphe 3.6.1) obtenue en introduisant des termes supplémentaires dans la fonction objectif, et des contraintes supplémentaires pour tenir compte des nouvelles contraintes introduites dans le modèle MG^1.

4.6.2 Caractéristiques de ces variantes

Pour générer la solution initiale au cours de la phase 1, nous avons développé une heuristique constructive. Celle-ci sera présentée au paragraphe suivant 4.6.3.

À chaque itération majeure de la phase 2, les rondes dans Ω sont considérées séquentiellement dans un ordre aléatoire. Le sous-problème associé à chaque ronde $p \in \Omega$ est résolu en utilisant une généralisation de la méthode de recherche avec tabous à voisinages structurés (RTVS) présentée à la section 3.6. Des modifications sont requises à cause des contraintes supplémentaires que nous retrouvons dans le modèle MG_p^3. De plus, l'ordre dans lequel les trois composantes de cette méthode sont appliquées diffère de celui de la méthode présentée à la section 3.6. En effet, nous appliquons d'abord une stratégie de diversification (étape 1) pour générer une nouvelle solution initiale de la ronde p, puis une recherche avec tabous à voisinages structurés (étape 2) pour réduire le nombre de matchs de cette ronde avec 1 et 3 juges affectés. Lors de la dernière étape

(étape 3), nous appliquons une autre recherche avec tabous pour améliorer la qualité de la solution au regard des autres contraintes du modèle MG_p^3.

Le fait d'utiliser la méthode de recherche avec tabous à voisinages structurés influe sur le choix de la solution initiale x_p^0 utilisée pour amorcer la résolution du sous-problème associé à la ronde p. En effet, puisque la stratégie de diversification de cette méthode consiste à construire une nouvelle solution à partir de la meilleure solution générée au cours de la procédure de résolution et une autre solution choisie aléatoirement parmi l'ensemble des meilleures solutions rencontrées, alors seule $xbestG_p$ (i.e., les affectations courantes que nous retrouvons dans $xbestG$) peut servir de solution initiale pour amorcer la résolution du sous-problème. Ainsi, seule cette façon de définir x_p^0 est considérée dans ces variantes. Par contre, il est possible de considérer les deux façons pour fixer les affectations des autres rondes $q \neq p$ (leurs valeurs courantes dans $xbestG$ ou dans $xbestLast$). Également, puisque la stratégie de diversification ne repose pas sur les violations, il est possible de considérer les deux façons de définir l'ensemble Ω.

Finalement, il convient de souligner que le choix de la solution initiale (phase 1) est motivé par les résultats des expérimentations numériques présentés au paragraphe 3.6.6. En effet, ces expérimentations ont indiqué que les meilleurs résultats de la RTVS sont obtenus avec une solution initiale de bonne qualité plutôt qu'avec une solution générée aléatoirement. D'un autre côté, les deux recherches avec tabous des étapes 2 et 3 (utilisées pour résoudre le sous-problème associé à la ronde p au cours de la phase 2) explorent l'ensemble des solutions réalisables du modèle MG_p^3 et utilisent la stratégie *meilleure amélioration* pour sélectionner la solution dans le voisinage de la solution courante. En effet, cette stratégie s'est avérée plus efficace que la stratégie *première amélioration* si l'on se réfère aux résultats de l'expérimentation numérique du chapitre 3 (cf. paragraphe 3.6.6). Notons que l'heuristique que nous utilisons au cours de la phase 1 pour générer la solution initiale peut produire des solutions non réalisables (i.e., qui ne respectent pas les contraintes du modèle MG_p^3). Cependant, la stratégie de diversification de l'étape 1 inclut un processus de réparation permettant de restaurer la réalisabilité.

4.6.3 Heuristique constructive

Nous introduisons maintenant une méthode heuristique permettant de construire une solution initiale pour l'approche de résolution basée sur la recherche avec tabous à voisinages structurés. Cette méthode s'apparente à la méthode **HLA-HOA** présentée au chapitre 3 pour le problème où une seule ronde est considérée (cf. section 3.3).

Pour faciliter la présentation, introduisons quelques définitions utiles pour préciser l'ordre dans lequel les matchs sont sélectionnés afin de procéder à l'affectation des juges, et quel juge ou quelle paire de juges leur sera affecté.

L'affectation d'un juge i à un match j est *éligible* si i est admissible à j, et si les deux contraintes inter rondes de *séquence* et de *couplage* sont satisfaites. L'affectation d'un juge i à un match j est *acceptable* si i est admissible à j, si la contrainte de *séquence* est satisfaite, mais celle de *couplage* est violée. Finalement, l'affectation d'un juge i à un match j est *tolérable* si i est admissible à j, mais les deux contraintes de *séquence* et de *couplage* sont violées.

Par ailleurs, rappelons que les matchs se divisent en deux catégories : les matchs *francophones* et ceux *anglophones*. Un match est *francophone* si au moins l'une des deux équipes en compétition dans ce match présente en français. Le cas échéant (i.e., si les deux équipes présentent en anglais) le match est *anglophone*. D'un autre côté, tous les juges sont *anglophones*, mais quelques uns seulement sont *bilingues*.

La génération de la solution initiale procède en quatre grandes étapes :

Étape 1 : Un *juge en chef* est affecté à chaque match de chaque ronde.

Étape 2 : Un juge *expert* est affecté à chaque match de chaque ronde.

Étape 3 : Un troisième juge est affecté à chaque match de chaque ronde.

Étape 4 : Une paire de juges additionnelle est affectée au plus grand nombre de matchs.

Présentons d'abord un cadre général pour générer les affectations au cours des trois premières étapes. Notons que tous les matchs de toutes les rondes sont traités de façon séquentielle à chaque étape. Il faut donc préciser l'ordre dans lequel ils seront sélectionnés. De plus, une fois qu'un match est sélectionné, il faut déterminer comment sélectionner le juge ou la paire de juges qui lui sera affecté. Ces sélections sont complétées

selon une suite de critères qui sont appliqués de façon hiérarchique. La logique derrière cette hiérarchie vise d'une part, à générer une solution de bonne qualité, et d'autre part, à faciliter les affectations futures. Ce dernier aspect est très important dans un processus heuristique d'affectation où les retours en arrière sont inexistants.

Analysons d'abord l'ordre de sélection des matchs. Le premier niveau hiérarchique stipule de toujours choisir parmi les matchs *francophones* avant de considérer ceux *anglophones*. Ceci est dû au fait que les matchs *francophones* requièrent des juges *bilingues*.

La sélection au deuxième niveau est complétée différemment suivant l'ensemble retenu au premier niveau. Ainsi, dans le cas où les matchs sont *francophones*, nous choisissons le premier sous-ensemble des matchs retenus au premier niveau qui n'est pas vide selon l'ordre suivant :

i. sous-ensemble des matchs pour lesquels il existe au moins un juge *bilingue* de type requis (*juge en chef*, *expert* ou *nouveau*) non affecté *éligible* ;

ii. sous-ensemble des matchs pour lesquels il existe au moins un juge *bilingue* de type requis non affecté *acceptable* ;

iii. sous-ensemble des matchs pour lesquels il existe au moins un juge *bilingue* de type requis non affecté *tolérable*.

Dans le cas où l'ensemble des matchs retenus au premier niveau contient des matchs *anglophones*, le nombre de sous-ensembles à considérer est doublé. Ainsi, nous choisissons le premier sous-ensemble des matchs retenus au premier niveau qui n'est pas vide selon l'ordre suivant :

iv. sous-ensemble des matchs pour lesquels il existe au moins un juge *anglophone* de type requis non affecté *éligible* ;

v. sous-ensemble des matchs pour lesquels il existe au moins un juge *bilingue* de type requis non affecté *éligible* ;

vi. sous-ensemble des matchs pour lesquels il existe au moins un juge *anglophone* de type requis non affecté *acceptable* ;

vii. sous-ensemble des matchs pour lesquels il existe au moins un juge *bilingue* de type requis non affecté *acceptable* ;

viii. sous-ensemble des matchs pour lesquels il existe au moins un juge *anglophone* de type requis non affecté *tolérable* ;

ix. sous-ensemble des matchs pour lesquels il existe au moins un juge *bilingue* de type requis non affecté *tolérable*.

Si le sous-ensemble sélectionné au deuxième niveau contient plus d'un élément, alors nous procédons au troisième niveau pour sélectionner le match. À ce niveau, la sélection diffère selon le sous-ensemble choisi au deuxième niveau. Ainsi, pour les sous-ensembles i, iv ou v, nous considérons d'abord le sous-ensemble des matchs comptant le plus petit nombre de juges de type requis non affectés *éligibles*. Si ce sous-ensemble comporte plus d'un élément, nous considérons le sous-ensemble des matchs comptant le plus petit nombre de juges de type requis non affectés *acceptables*. Si ce sous-ensemble comporte plus d'un élément, nous considérons le sous-ensemble des matchs comptant le plus petit nombre de juges de type requis non affectés *tolérables*. Si ce sous-ensemble comporte plus d'un élément, nous choisissons finalement un match aléatoirement. Dans le cas des sous-ensembles ii, vi ou vii, nous considérons successivement le sous-ensemble des matchs comptant le plus petit nombre de juges de type requis non affectés *acceptables*, puis *tolérables*. Finalement, pour les sous-ensembles iii, viii ou ix, nous considérons uniquement le sous-ensemble des matchs comptant le plus petit nombre de juges de type requis non affectés *tolérables*.

Pour sélectionner le juge de type requis à affecter au match sélectionné, le sous-ensemble des juges considéré dépend du type (*francophone* ou *anglophone*) de ce match. Nous considérons séquentiellement une sous-suite hiérarchique des sous-ensembles suivants :

a. sous-ensemble des juges de type requis qui sont *éligibles* pour le moins de matchs auxquels aucun juge n'a encore été affecté au cours de l'étape d'affectation courante ;

b. sous-ensemble des juges de type requis qui sont *acceptables* pour le moins de

matchs auxquels aucun juge n'a encore été affecté au cours de l'étape d'affectation courante ;

c. sous-ensemble des juges de type requis qui sont *admissibles* pour le moins de matchs auxquels aucun juge n'a encore été affecté au cours de l'étape d'affectation courante ;

d. sous-ensemble des juges de type requis qui ont été affectés le moins de fois aux matchs impliquant les deux équipes en compétition dans le match sélectionné ;

e. sous-ensemble des juges de type requis qui ont été couplés le moins de fois avec les juges déjà affectés au match sélectionné ;

f. sous-ensemble des juges de type requis qui couvrent le plus grand nombre de domaines d'expertise manquants dans le match sélectionné ;

g. sous-ensemble des juges de type requis affiliés aux compagnies qui sont représentées le moins de fois dans le match sélectionné.

Notons que les sous-ensembles d à g sont reliés à la satisfaction des règles souples d'affectation dans le match sélectionné, et qu'ils sont spécifiés dans l'ordre de priorité de celles-ci.

Si le match sélectionné fait partie du sous-ensemble du deuxième niveau i, iv, ou v, alors la sous-suite des sous-ensembles considérée est constituée de ceux en a, f, et g. S'il fait plutôt partie du sous-ensemble du deuxième niveau ii, vi, ou vii, ou du sous-ensemble iii, viii, ou ix, alors les sous-suites des sous-ensembles considérées sont respectivement (b, e, f, et g) et (c, d, e, f, et g).

Précisons maintenant l'application du cadre général au cours de chacune des trois premières étapes. Pour l'étape 1, le type de juges requis est un *juge en chef*. De plus, puisqu'il s'agit du premier juge assigné à chaque match, les notions de juge *éligible*, *acceptable* ou *tolérable* ne s'appliquent pas. Seule l'admissibilité du *juge en chef* au match et la satisfaction de la contrainte inter rondes de *séquence* entrent en ligne de compte. Au cours de l'étape 2, le type de juges requis est un *juge en chef* ou un juge de la catégorie *Expert*, car les *juges en chef* non affectés à l'étape 1 sont considérés à titre d'*expert*. Finalement à l'étape 3, tous les juges qui restent sont considérés comme étant

de type requis.

Pour l'étape 4, nous devons adapter le cadre général précédent en fonction du fait qu'une paire de juges additionnelle doit être affectée au maximum de matchs. Comme pour les étapes précédentes, le premier niveau hiérarchique pour sélectionner les matchs stipule de considérer les matchs *francophones* avant de considérer ceux *anglophones*.
Dans le cas des matchs *francophones*, nous considérons (au deuxième niveau) dans un premier temps le sous-ensemble des matchs pour lesquels il existe au moins une paire de juges *bilingue* non affectée qui respecte les trois contraintes d'admissibilité, de *séquence*, et de *couplage*. Si ce sous-ensemble est vide, nous relaxons successivement les critères de sélection dans l'ordre suivant. D'abord, nous acceptons que la contrainte de *couplage* ne soit pas respectée, et ensuite, nous tolérons que les deux contraintes de *séquence* et de *couplage* soient violées. Lorsque nous considérons les matchs *anglophones*, les mêmes sous-ensembles sont considérés dans le même ordre, sauf que nous n'imposons pas que les juges soient *bilingues*. Notons que contrairement aux étapes 1, 2 et 3, le nombre de sous-ensembles n'est pas doublé, car nous ne considérons pas en premier lieu les juges *anglophones* puis ceux *bilingues*. En effet, choisir avec parcimonie les juges *bilingues* n'a plus aucune importance puisqu'il s'agit de la dernière étape d'affectation et aucun autre juge (ou paire de juges) ne sera affecté(e) aux matchs *francophones*.
Les critères de sélection au troisième niveau reposent sur le même principe utilisé aux étapes 1, 2 et 3. Ils visent à faciliter la satisfaction des contraintes d'admissibilité, de *séquence* et de *couplage* lors des affectations futures. Toutefois, nous considérons cette fois-ci le nombre de paires de juges qui satisfont les conditions du deuxième niveau plutôt que le nombre de juges.
Finalement, les critères utilisés pour sélectionner la paire de juges sont également ajustés et visent d'une part, à faciliter les affectations futures, et d'autre part, à minimiser le nombre de violations des différentes contraintes dans le match sélectionné.

Dans l'approche de résolution que nous venons de décrire, nous avons considéré un seul problème d'affectation avec PM matchs, P et M étant respectivement le nombre de rondes et le nombre de matchs par ronde. Nous aurions pu résoudre séquentiellement les P sous-problèmes associés aux différentes rondes en tenant compte, lors de la résolution

de chaque sous-problème, des solutions des sous-problèmes associés aux rondes déjà considérées. Cette approche présente l'avantage de traiter successivement des problèmes de petite taille. Toutefois, la qualité des solutions générées dépend de l'ordre dans lequel les rondes sont considérées. Dans l'approche que nous avons retenue (un seul problème avec *PM* matchs), le problème est certes de plus grande taille, mais comme l'approche repose sur une vision globale, elle est nettement moins sensible au facteur de l'ordre, et donc, est plus efficace. Des tests numériques préliminaires ont corroboré cette observation.

Par ailleurs, notons que la contrainte d'*équilibre* (stipulant que, hormis le *juge en chef*, le même nombre de juges *experts* et *nouveaux* doit être affecté à chaque match) n'est pas prise en compte dans le processus de résolution. Elle peut cependant y être intégrée facilement. En effet, il suffit d'ajuster les critères de sélection à l'étape 3 de sorte à restreindre dans un premier temps le type de juges requis aux juges de la catégorie *Nouveau*, puis par la suite, si aucun juge de cette catégorie n'a pu être identifié, considérer tous les juges non affectés indépendamment de leur catégorie. De même, à l'étape 4, il suffit d'imposer d'abord que la paire de juges soit composée d'un juge *expert* et d'un juge *nouveau*, puis relaxer cette condition en cas d'échec (i.e., considérer toutes les paires de juges disponibles indépendamment de la catégorie des juges qui les forment).

Finalement, il convient de souligner que l'heuristique peut produire des solutions non réalisables (si, à l'une des étapes 1, 2 ou 3, tous les sous-ensembles considérés au deuxième niveau de sélection des matchs sont vides).

4.6.4 Stratégie de diversification

Rappelons qu'au cours de la phase 2, les rondes dans Ω sont considérées séquentiellement. Pour chaque ronde $p \in \Omega$, la recherche avec tabous à voisinages structurés (étape 2) est initialisée avec une nouvelle solution de cette ronde générée suite à l'application d'une stratégie de diversification (étape 1). Le principe de cette stratégie est le même que celui présenté au paragraphe 3.6.5. Seules certaines phases doivent êtres ajustées pour tenir compte du fait qu'il faut que chaque match comporte au moins un juge *expert* en plus d'un *juge en chef*.

Nous allons adopter une notation similaire à celle du paragraphe 3.6.5. Dénotons par Γ_p l'ensemble des ρ meilleures solutions de la ronde p générées jusqu'à présent. Notons qu'au cours de la phase 1 (heuristique constructive), les ensembles Γ_p, $1 \leq p \leq P$, sont mis à jour à la fin de chacune des quatre étapes de l'heuristique. À titre d'exemple, à la fin de l'étape 1, chaque ensemble Γ_p comprend une solution où un *juge en chef* est affecté à chaque match $j \neq (M+1)_p$, et où les autres juges disponibles pour p sont affectés au match fictif $(M+1)_p$. Soient $xbest_p$ la meilleure solution de Γ_p, et $\bar{x}_p \neq xbest_p$ une autre solution choisie aléatoirement dans Γ_p. Nous générons une première affectation x_p^0 en complétant un croisement uniforme entre $xbest_p$ et $\bar{x}_p$ utilisant l'opérateur décrit au paragraphe 3.6.5.

x_p^0 peut être non réalisable parce qu'un même juge i peut être affecté à deux matchs différents $j_1 \in M_{best\,p}$ et $j_2 \notin M_{best\,p}$. ($M_{best\,p}$ dénote le sous-ensemble des matchs pour lesquels nous retenons les affectations de $xbest_p$. Pour les autres matchs, les affectations sont retenues de $\bar{x}_p$.) Pour favoriser la présence des affectations de $xbest_p$ dans x_p^0, nous éliminons i de j_2 à moins que $j_1 = (M+1)_p$, ou si en supprimant i de j_1, au moins un *juge en chef* (si j_1 compte 2 ou 3 juges affectés) ou un *juge en chef* et un juge *expert* (si j_1 compte 4 ou 5 juges juges affectés) restent affectés à ce match .

Le *processus de réparation* doit aussi être modifié pour permettre de générer une solution où, d'une part, chaque match compte au moins un *juge en chef* affecté, et d'autre part, chaque match avec 3 ou 5 juges affectés compte au moins un autre juge *expert*. Ainsi, dans un premier temps, un *juge en chef* est affecté à chaque match en utilisant la phase 1 du processus de réparation décrite au paragraphe 3.6.5. Une seconde phase très similaire à la première est utilisée pour tenter d'affecter un juge *expert* à chaque match. Pour faciliter la présentation, dénotons par $L_j(x_p^0)$ et $E_j(x_p^0)$ respectivement le nombre de *juges en chef* et de juges de la catégorie *Expert* affectés au match j dans la solution x_p^0. Soit :

$$Ex_p = \{j \neq (M+1)_p \quad : \quad L_j(x_p^0) = 1 \text{ et } E_j(x_p^0) = 0\}.$$

À chaque itération de cette seconde phase, un match $l \in Ex_p$ est d'abord sélectionné

aléatoirement. Les *juges en chef* et ceux de la catégorie *Expert* admissibles à l sont ensuite considérés séquentiellement dans un ordre aléatoire :

- si le juge i est un *juge en chef* affecté au match fictif $(M+1)_p$ ou à un match $j \neq (M+1)_p$ tel que $L_j(x_p^0) + E_j(x_p^0) \geq 3$, alors nous le retirons de j pour le réaffecter à l ;
- si i est un juge de la catégorie *Expert* affecté à $(M+1)_p$ ou à un match $j \neq (M+1)_p$ tel que $L_j(x_p^0) + E_j(x_p^0) \geq 3$, alors nous le retirons également de j pour le réaffecter à l ;
- si i est le seul *juge en chef* affecté à $j \neq (M+1)_p$ (i.e., $L_j(x_p^0) = 1$), et si r est un autre *juge en chef* admissible à j et affecté à $(M+1)_p$ ou à un match $\overline{j} \neq (M+1)_p$ tel que $L_{\overline{j}}(x_p^0) \geq 2$ et $L_{\overline{j}}(x_p^0) + E_{\overline{j}}(x_p^0) \geq 3$, alors nous réaffectons r à j et i à l ;
- Si i est un juge de la catégorie *Expert* affecté à un match $j \neq (M+1)_p$ tel que $L_j(x_p^0) = E_j(x_p^0) = 1$, et si r est un *juge en chef* admissible à j et affecté à $(M+1)_p$ ou à un match $\overline{j} \neq (M+1)_p$ tel que $L_{\overline{j}}(x_p^0) \geq 2$ et $L_{\overline{j}}(x_p^0) + E_{\overline{j}}(x_p^0) \geq 3$, alors nous réaffectons r à j et i à l ;
- si i est un juge de la catégorie *Expert* affecté à un match $j \neq (M+1)_p$ tel que $L_j(x_p^0) = E_j(x_p^0) = 1$, et si r est un juge de la catégorie *Expert* admissible à j et affecté à $(M+1)_p$ ou à un match $\overline{j} \neq (M+1)_p$ tel que $L_{\overline{j}}(x_p^0) + E_{\overline{j}}(x_p^0) \geq 3$, alors nous réaffectons r à j et i à l.

Notons qu'il est peut être impossible d'affecter un juge *expert* à un certain match l. Dans ce cas, nous conservons dans l un *juge en chef* et nous réaffectons, s'il y a lieu, tous les autres juges affectés à l au match fictif $(M+1)_p$.

La dernière phase du *processus de réparation* vise à satisfaire la contrainte stipulant qu'exactement 1, 3 ou 5 juges doivent être affectés à chaque match $j \neq (M+1)_p$. Pour y arriver, nous considérons séquentiellement et dans un ordre aléatoire les matchs l comptant 2 ou 4 juges affectés. S'il existe un juge i affecté à $(M+1)_p$ et admissible à l, alors nous le retirons de $(M+1)_p$ pour le réaffecter à l. Sinon (i.e., si aucun juge de $(M+1)_p$ n'est admissible à l), nous procédons comme suit : si l compte 2 juges affectés, alors nous en choisissons un pour le réaffecter à $(M+1)_p$ en s'assurant que le juge qui reste affecté à l est un *juge en chef*. Si l compte 4 juges affectés, alors nous en choisissons un

de façon aléatoire pour le réaffecter à $(M+1)_p$ en s'assurant qu'il reste dans l un *juge en chef* et un juge de la catégorie *Expert* (ou un deuxième *juge en chef*).

4.6.5 Recherche avec tabous à voisinages structurés

Une fois que nous avons complété la première étape et généré une nouvelle solution réalisable de la ronde p, nous utilisons la recherche avec tabous à voisinages structurés suivante pour réduire le nombre de matchs de p avec 1 et 3 juges affectés.

Le voisinage est défini comme au paragraphe 3.6.3, i.e., il est généré en réaffectant une paire de juges (i,r) affectée présentement à un certain match j à un autre match $l \neq j$. La définition d'une réaffectation réalisable est toutefois plus complexe. Elle doit être modifiée pour assurer qu'après la réaffectation, d'une part, les deux matchs j et l comptent au moins un *juge en chef*, et d'autre part, si j ou l comptent 3 ou 5 juges affectés, alors un de ceux-ci est un juge *expert* (un autre *juge en chef* ou un juge de la catégorie *Expert*). La réaffectation est donc réalisable et la solution générée appartient au voisinage de x_p si et seulement si :

- les juges i et r sont admissibles au match l :

$$a_{ilp} = a_{rlp} = 1$$

- il reste au moins un *juge en chef* affecté au match j :

$$\exists \bar{i} \in I(j,x_p) - \{i,r\} \quad \text{tel que} \quad l_{\bar{i}} = 1$$

  (Rappelons que $I(j,x_p)$ représente l'ensemble des juges affectés au match j dans la solution x_p .)

- si le match j compte présentement 5 juges affectés, il reste au moins un juge *expert* (autre que le *juge en chef*) affecté à j :

$$\exists i' \in I(j,x_p) - \{i,r\} \text{ tel que soit } (l_{i'} = 1 \text{ et } i' \neq \bar{i}) \text{ ou bien } (v_{i'} = 1)$$

- si le match l compte présentement un seul juge affecté, alors au moins l'un des

deux juges i ou r est un juge *expert* :

$$l_i + l_r + v_i + v_r \geq 1$$

– au plus 5 juges seront affectés au match l :

$$|I(l, x_p) \cup \{i, r\}| \leq 5.$$

Nous tirons avantage de la structure du problème pour partitionner $V(x_p)$ (l'ensemble des réaffectations réalisables pour la solution x_p) de la même façon qu'au paragraphe 3.6.3, et nous obtenons ainsi les mêmes huit sous-ensembles $\bigcup_{\tau=1}^{8} V_\tau(x_p) \subseteq V(x_p)$. Comme il a déjà été souligné au paragraphe 4.6.2, nous utilisons la stratégie *meilleure amélioration* comme stratégie de sélection dans le voisinage (généré en utilisant l'un des sous-ensembles de réaffectations $V_\tau(x_p)$). Générer tous les éléments du voisinage pour choisir la prochaine solution courante peut requérir beaucoup de temps. C'est pourquoi, au paragraphe 3.6.3, nous avons induit la valeur de la meilleure modification pour arrêter la génération du voisinage (dès qu'une réaffectation induit cette valeur). Ceci avait permis d'accélérer de manière significative l'exploration. Malheureusement, les tests préliminaires ont indiqué que cette technique n'est pas appropriée pour le problème étudié dans ce chapitre. Nous avons constaté que, le plus souvent, le voisinage est généré au complet car aucune solution voisine n'induit la modification souhaitée. Ceci est dû au fait que la fonction objectif du modèle MG_p^3 comporte un nombre accru de composantes comparativement à celle du modèle M^3. En effet, la valeur de la meilleure modification est fixée au préalable. Elle est calculée en supposant que la paire de juges (i, r) viole le plus de contraintes possible dans j (match duquel elle sera supprimée), et qu'elle violerait le moins de contraintes possible dans l (match dans lequel elle sera insérée). Or, en réalité, cette situation n'est pas toujours rencontrée car au fur et à mesure du déroulement du processus de résolution, certaines contraintes ne sont violées dans aucun match. Cette lacune n'a pas été relevée dans le cas du *problème sur une seule ronde* (étudié au chapitre précédent) car celui-ci comporte une seule contrainte souple.

Ces constatations nous ont conduits à apporter les deux modifications suivantes à la stratégie d'exploration du voisinage :

- calculer la valeur de la meilleure modification au moment de l'examen du voisinage (évaluation dynamique plutôt que statique) ;
- générer en premier les solutions voisines susceptibles d'induire la meilleure modification. En effet, nous avons constaté que, quoique nous ajustions les valeurs des meilleures modifications dynamiquement, l'approche initiale utilisée pour examiner les solutions voisines nous conduit parfois à parcourir le voisinage au complet.

Ces deux modifications font l'objet de la discussion qui suit. Nous expliquons d'abord comment nous pouvons déterminer les valeurs des meilleures modifications pour les différents sous-ensembles $V_\tau(x_p)$, puis nous montrons comment nous pouvons identifier les solutions susceptibles d'induire ces modifications.

Introduisons d'abord quelques notations utiles. Pour chaque match $j \neq (M+1)_p$, dénotons par $nv_1(j)$, $nv_2(j)$, $nv_3(j)$, $nv_4(j)$, et $nv_5(j)$ respectivement le nombre de violations des contraintes d'*équilibre*, d'*affiliation*, de *diversité*, de *couplage*, et de *séquence* dans ce match. Se référant au modèle MG^3_p, nous pouvons les définir comme suit :

- $nv_1(j) = d^{1+}_{jp} + d^{1-}_{jp}$
- $nv_2(j) = \sum_{c=1}^{C} d^2_{cjp}$
- $nv_3(j) = \sum_{k=1}^{K} d^3_{kjp}$
- $nv_4(j) = \sum_{i \in I(j,x_p)} \sum_{\substack{r \in I(j,x_p) \\ r>i}} d^4_{ir}$, $I(j,x_p)$ étant l'ensemble des juges affectés au match j dans la solution x_p
- $nv_5(j) = \sum_{i \in I(j,x_p)} (d^5_{it_1} + d^5_{it_2})$, t_1 et t_2 étant les deux équipes en compétition dans le match j.

(Rappelons que pour obtenir MG^3_p, il suffit de fixer, dans le modèle MG^3 résumé dans la figure 4.8, les variables associées aux rondes $q \neq p$ à leurs valeurs courantes.)

Soient $\widetilde{nv}_1(j)$, $\widetilde{nv}_2(j)$, $\widetilde{nv}_3(j)$, $\widetilde{nv}_4(j)$, et $\widetilde{nv}_5(j)$ le nombre de violations de ces contraintes dans le match j après qu'on ait modifié les affectations dans ce match (insertion ou suppression d'une paire de juges).

Pour chaque réaffectation réalisable (i,r,j,l), dénotons par :

- $\delta_v(i,r,j,l)$, $v = 1,\ldots,7$, la modification induite dans le $v^{\text{ème}}$ terme de la fonction objectif du modèle MG_p^3
- $\Delta(i,r,j,l) = \sum_{v=1}^{7} \delta_v(i,r,j,l)$, la modification totale induite dans la fonction objectif.

Soit $\Delta_\tau(x_p) = \min_{(i,r,j,l)\in V_\tau(x_p)} \Delta(i,r,j,l)$, la meilleure modification pouvant être induite en considérant l'ensemble des réaffectations $(i,r,j,l) \in V_\tau(x_p)$.

Finalement, rappelons que $M_1(x_p)$, $M_3(x_p)$ et $M_5(x_p)$ désignent respectivement les sous-ensembles des matchs comptant 1, 3 et 5 juges affectés dans la solution x_p.

Nous allons maintenant expliquer, pour chaque $V_\tau(x_p)$, comment nous pouvons obtenir des bornes inférieures sur les valeurs des modifications pouvant être induites, en tenant compte de l'état actuel de la recherche.

Considérons le premier voisinage $V_1(x_p) = \{(M+1)_p, M_1(x_p)\}$. Associons à chaque match $l \in M_1(x_p)$ la valeur $G(l) = w_3 \max(1-K, -nv_3(l))$ (K étant le nombre de domaines d'expertise), et soit $l^* = \arg\min_{l\in M_1(x_p)} G(l)$. Le résultat suivant garantit que la modification induite par toute réaffectation $(i,r,j,l) \in V_1(x_p)$ ne peut être meilleure que $\Delta_1(x_p) = -w_7 + w_6 + G(l^*)$.

Propriété 4.4. *Pour toute réaffectation* $(i,r,j,l) \in V_1(x_p)$,

$$\Delta(i,r,j,l) \geq -w_7 + w_6 + G(l^*) = \Delta_1(x_p).$$

Preuve. Soit $(i,r,j,l) \in V_1(x_p) = \{(M+1)_p, M_1(x_p)\}$. Identifions des bornes inférieures sur les valeurs des $\delta_v(i,r,j,l)$, $v = 1,\ldots,7$.

Considérons d'abord le terme $\delta_1(i,r,j,l)$ correspondant à la contrainte d'*équilibre*. Il est clair que :

$$\delta_1(i,r,j,l) = w_1(\widetilde{nv}_1(l) - nv_1(l))$$

car $j = (M+1)_p$ est un match fictif. Par ailleurs, puisque le nombre de violations de la contrainte d'*équilibre* dans le match l ne peut qu'augmenter (ou garder la même valeur)

lorsqu'on insère une paire de juges dans ce match, alors $\widetilde{nv}_1(l) \geq nv_1(l) = 0$, et ainsi :

$$\delta_1(i,r,j,l) \geq 0.$$

Un argument analogue nous permet de tirer une conclusion similaire pour les modifications correspondant aux contraintes d'*affiliation*, de *couplage*, et de *séquence*. Nous avons donc : $\delta_2(i,r,j,l) \geq 0$, $\delta_4(i,r,j,l) \geq 0$, et $\delta_5(i,r,j,l) \geq 0$.
Considérons maintenant le terme $\delta_3(i,r,j,l)$ correspondant à la contrainte de *diversité*. Là encore, il est facile de voir que :

$$\delta_3(i,r,j,l) = w_3\left(\widetilde{nv}_3(l) - nv_3(l)\right).$$

Par ailleurs, puisque $nv_3(l) \leq K-1$ et $\widetilde{nv}_3(l) \geq 0$, alors d'une part :

$$\widetilde{nv}_3(l) - nv_3(l) \geq 1 - K$$

et, d'autre part :

$$\widetilde{nv}_3(l) - nv_3(l) \geq -nv_3(l).$$

Il s'ensuit alors que :

$$\delta_3(i,r,j,l) \geq w_3 \max\left(1-K, -nv_3(l)\right).$$

Finalement, puisque $(i,r,j,l) \in V_1(x_p) = \{(M+1)_p, M_1(x_p)\}$, cette réaffectation implique qu'il y a un nouveau match avec 3 juges affectés, et que le nombre de matchs avec un seul juge affecté est réduit de 1 ; donc, $\delta_6(i,r,j,l) = w_6$ et $\delta_7(i,r,j,l) = -w_7$.
Maintenant, en utilisant les valeurs des différentes bornes inférieures ci-dessus, nous

concluons que :

$$\begin{aligned}\Delta(i,r,j,l) &= \sum_{\nu=1}^{7} \delta_\nu(i,r,j,l) \\ &\geq w_3 \max(1-K, -nv_3(l)) + w_6 - w_7 \\ &= -w_7 + w_6 + G(l) \\ &\geq -w_7 + w_6 + G(l^*) = \Delta_1(x_p)\end{aligned}$$

ce qui complète la preuve. ■

Considérons maintenant le second voisinage $V_2(x_p) = \{M_5(x_p), M_1(x_p)\}$. Associons à chaque match $j \in M_5(x_p)$ la valeur suivante :

$$H(j) = w_2 \max(-2, -nv_2(j)) + w_4 \max(-6, -nv_4(j)) + w_5 \max(-4, -nv_5(j)).$$

Soient $j^* = \arg\min_{j \in M_5(x_p)} H(j)$ et $l^* \in M_1(x_p)$ tel que défini précédemment, i.e., $l^* = \arg\min_{l \in M_1(x_p)} G(l)$ où, $G(l) = w_3 \max(1-K, -nv_3(l))$. Le résultat suivant précise une borne inférieure sur la meilleure réaffectation dans $V_2(x_p)$.

Propriété 4.5. *Pour toute réaffectation* $(i,r,j,l) \in V_2(x_p)$,

$$\Delta(i,r,j,l) \geq -w_7 + 2w_6 + G(l^*) + H(j^*) = \Delta_2(x_p).$$

Preuve. Soit $(i,r,j,l) \in V_2(x_p) = \{M_5(x_p), M_1(x_p)\}$. Nous procédons comme dans la preuve de la propriété 4.4 pour identifier des bornes inférieures sur les valeurs des $\delta_\nu(i,r,j,l)$, $\nu = 1,\ldots,7$. Avant de les expliciter, notons que pour tout $\nu = 1,\ldots,5$,

$$\delta_\nu(i,r,j,l) = w_\nu\left((\widetilde{nv}_\nu(j) - nv_\nu(j)) + (\widetilde{nv}_\nu(l) - nv_\nu(l))\right).$$

$\nu = 1$ **(contrainte d'*équilibre*).** D'une part,

$$((\widetilde{nv}_1(j) - nv_1(j)) + (\widetilde{nv}_1(l) - nv_1(l))) \geq -2 + 2 = 0.$$

En effet, rappelons que pour tout match $j \neq (M+1)_p$:

$$nv_1(j) = d_{jp}^{1+} + d_{jp}^{1-}$$

où, d_{jp}^{1+} et d_{jp}^{1-} représentent respectivement l'écart positif et négatif entre le nombre de juges *experts* et *nouveaux* affectés à j. La valeur -2 est atteinte si aucun juge de la catégorie *Nouveau* n'est affecté au match j, i.e., $d_{jp}^{1+} = 4$ et $d_{jp}^{1-} = 0$. Mais alors, après la réaffectation la valeur de d_{lp}^{1+} deviendrait 2 alors qu'elle était 0.
D'autre part, en utilisant un argument similaire à celui utilisé dans la preuve de la propriété 4.4, nous avons $\widetilde{nv}_1(l) - nv_1(l) \geq 0$, et puisque $\widetilde{nv}_1(j) \geq 0$, alors :

$$(\widetilde{nv}_1(j) - nv_1(j)) + (\widetilde{nv}_1(l) - nv_1(l)) \geq -nv_1(j).$$

Il s'ensuit donc que :

$$\delta_1(i,r,j,l) \geq w_1 \max(0, -nv_1(j)) = 0.$$

$v = 2$ **(contrainte d'*affiliation*).** Là encore, nous avons :

$$(\widetilde{nv}_2(j) - nv_2(j)) + (\widetilde{nv}_2(l) - nv_2(l)) \geq -nv_2(j)$$

car $\widetilde{nv}_2(j) \geq 0$, et $\widetilde{nv}_2(l) - nv_2(l) \geq 0$ puisque le nombre de violations de la contrainte d'*affiliation* ne peut qu'augmenter lorsqu'on insère une paire de juges dans le match l.
Aussi,

$$(\widetilde{nv}_2(j) - nv_2(j)) + (\widetilde{nv}_2(l) - nv_2(l)) \geq \widetilde{nv}_2(j) - nv_2(j) \geq -2$$

car il ne peut y avoir une réduction de plus de deux violations de la contrainte d'*affiliation* dans le match j, ce qui est le cas lorsque chacun des deux juges i et r représente la même compagnie qu'un autre juge affecté à j.

Ainsi, nous obtenons :

$$\delta_2(i,r,j,l) \geq w_2 \max(-2, -nv_2(j)).$$

$v = 3$ **(contrainte de *diversité*).** D'une part,

$$(\widetilde{nv}_3(j) - nv_3(j)) + (\widetilde{nv}_3(l) - nv_3(l)) \geq -nv_3(l)$$

car $\widetilde{nv}_3(l) \geq 0$ et $\widetilde{nv}_3(j) - nv_3(j) \geq 0$ puisque le nombre de domaines d'expertise non couverts dans le match j ne peut qu'augmenter (ou garder la même valeur) lorsqu'on supprime une paire de juges de ce match.
D'autre part,

$$(\widetilde{nv}_3(j) - nv_3(j)) + (\widetilde{nv}_3(l) - nv_3(l)) \geq \widetilde{nv}_3(l) - nv_3(l) \geq 1 - K.$$

Il s'ensuit donc que :

$$\delta_3(i,r,j,l) \geq w_3 \max(1 - K, -nv_3(l)).$$

$v = 4$ **(contrainte de *couplage*).** D'une part,

$$(\widetilde{nv}_4(j) - nv_4(j)) + (\widetilde{nv}_4(l) - nv_4(l)) \geq -7 + 1 = -6.$$

Cette borne inférieure est atteinte dans le cas où la paire de juges (i,r) viole la contrainte de *couplage*, et chacun des juges formant cette paire viole cette contrainte avec chacun des trois autres juges affectés au match j, mais ni i ni r ne violent cette contrainte avec le juge affecté au match l.
D'autre part, par un argument similaire à celui utilisé pour les contraintes d'*équilibre* et d'*affiliation*, nous avons :

$$(\widetilde{nv}_4(j) - nv_4(j)) + (\widetilde{nv}_4(l) - nv_4(l)) \geq -nv_4(j).$$

Ainsi, nous obtenons :

$$\delta_4(i,r,j,l) \geq w_4 \max(-6, -nv_4(j)).$$

$\underline{v = 5}$ **(contrainte de *séquence*).** Encore une fois,

$$(\widetilde{nv}_5(j) - nv_5(j)) + (\widetilde{nv}_5(l) - nv_5(l)) \geq \widetilde{nv}_5(j) - nv_5(j) \geq -4$$

car il ne peut y avoir une réduction de plus de quatre violations de la contrainte de *séquence* dans le match j, ce qui est le cas lorsque chacun des juges i et r viole cette contrainte pour chacune des deux équipes en compétition dans j alors que ni i ni r ne violent cette contrainte pour les deux équipes en compétition dans l. Par ailleurs,

$$(\widetilde{nv}_5(j) - nv_5(j)) + (\widetilde{nv}_5(l) - nv_5(l)) \geq -nv_5(j).$$

Donc,

$$\delta_5(i,r,j,l) \geq w_5 \max(-4, -nv_5(j)).$$

$\underline{v = 6 \text{ et } v = 7}$ **(nombre de matchs avec 3 et 1 juge(s) affecté(s) respectivement).**
Finalement, la réaffectation $(i,r,j,l) \in V_2(x_p) = \{M_5(x_p), M_1(x_p)\}$ implique que deux matchs avec 3 juges affectés sont créés alors que le nombre de matchs avec un seul juge affecté est réduit de 1. Les deux derniers termes prennent donc les valeurs $\delta_6(i,r,j,l) = 2w_6$ et $\delta_7(i,r,j,l) = -w_7$.

Maintenant de ce qui précède, et en se référant aux définitions de $H(j)$ et $G(l)$, nous concluons que :

$$\begin{aligned} \Delta(i,r,j,l) &= \sum_{v=1}^{7} \delta_v(i,r,j,l) \\ &\geq -w_7 + 2w_6 + G(l) + H(j) \\ &\geq -w_7 + 2w_6 + G(l^*) + H(j^*) = \Delta_2(x_p) \end{aligned}$$

ce qui complète la preuve. ■

Une démarche similaire à celles utilisées dans les preuves des deux propriétés précédentes nous permet de déterminer $\Delta_\tau(x_p)$, la valeur de la meilleure modification pouvant être induite par toute réaffectation $(i,r,j,l) \in V_\tau(x_p)$, pour chacun des autres voisinages $V_\tau(x_p)$, $\tau = 3,\ldots,8$.

Les résultats obtenus pour chaque voisinage $V_\tau(x_p) = \{M_{out}, M_{in}\}$ sont résumés dans le tableau 4.9 où, $G(l^*) = \min_{l \in M_{in}} G(l)$ et $H(j^*) = \min_{j \in M_{out}} H(j)$. Les valeurs de $H(j)$, $G(l)$, et $\Delta_\tau(x_p)$ sont données respectivement dans les troisième, quatrième, et dernière colonne de ce tableau.

τ	$V_\tau(x_p)$	$H(j)$	$G(l)$	$\Delta_\tau(x_p)$
1	$\{(M+1)_p, M_1(x_p)\}$	0	$w_3 \max(1-K, -nv_3(l))$	$-w_7 + w_6 + G(l^*)$
2	$\{M_5(x_p), M_1(x_p)\}$	$w_2 \max(-2, -nv_2(j)) + w_4 \max(-6, -nv_4(j)) + w_5 \max(-4, -nv_5(j))$	$w_3 \max(1-K, -nv_3(l))$	$-w_7 + 2w_6 + G(l^*) + H(j^*)$
3	$\{(M+1)_p, M_3(x_p)\}$	0	$w_1 \max(-2, -nv_1(l)) + w_3 \max(1-K, -nv_3(l))$	$-w_6 + G(l^*)$
4	$\{M_5(x_p), M_3(x_p)\}$	$w_2 \max(-2, -nv_2(j)) + w_4 \max(-6, -nv_4(j)) + w_5 \max(-4, -nv_5(j))$	$w_3 \max(1-K, -nv_3(l))$	$G(l^*) + H(j^*)$
5	$\{M_3(x_p), M_1(x_p)\}$	$w_2 \max(-1, -nv_2(j)) + w_4 \max(-2, -nv_4(j)) + w_5 \max(-4, -nv_5(j))$	$w_3 \max(1-K, -nv_3(l))$	$G(l^*) + H(j^*)$
6	$\{M_5(x_p), (M+1)_p\}$	$w_1 \max(-2, -nv_1(j)) + w_2 \max(-2, -nv_2(j)) + w_4 \max(-7, -nv_4(j)) + w_5 \max(-4, -nv_5(j))$	0	$w_6 + H(j^*)$
7	$\{M_3(x_p), M_3(x_p)\}$	$w_2 \max(-1, -nv_2(j)) + w_4 \max(-2, -nv_4(j)) + w_5 \max(-4, -nv_5(j))$	$w_3 \max(1-K, -nv_3(l))$	$w_7 - 2w_6 + G(l^*) + H(j^*)$
8	$\{M_3(x_p), (M+1)_p\}$	$w_1 \max(-2, -nv_1(j)) + w_2 \max(-2, -nv_2(j)) + w_4 \max(-3, -nv_4(j)) + w_5 \max(-4, -nv_5(j))$	0	$w_7 - w_6 + H(j^*)$

Tableau 4.9 – Partition de $V(x_p)$

Introduisons maintenant des conditions nécessaires qui caractérisent les solutions voisines pouvant induire les meilleures modifications $\Delta_\tau(x_p)$ données dans la dernière colonne du tableau 4.9.

Propriété 4.6. *Si le voisinage $V_\tau(x_p)$ comporte une solution voisine $x_p \oplus (i,r,j,l)$ telle que $\Delta(i,r,j,l) = \Delta_\tau(x_p)$, alors cette solution correspond à une solution obtenue en supprimant une paire de juges (i,r) d'un match $j^* = \arg\min_{j \in M_{out}} H(j)$ et en la réinsérant dans un match $l^* = \arg\min_{l \in M_{in}} G(l)$.*

Preuve. Remarquons que pour chaque voisinage $V_\tau(x_p)$, $\tau = 1,\ldots,8$, la valeur de la meilleure modification peut se réécrire sous la forme $\Delta_\tau(x_p) = constante_\tau + G(l^*) + H(j^*)$. Procédons par absurde pour démontrer la propriété.
Supposons qu'il existe une réaffectation $(\bar{i},\bar{r},\bar{j},\bar{l}) \in V_\tau(x_p)$ telle que $\Delta(\bar{i},\bar{r},\bar{j},\bar{l}) = \Delta_\tau(x_p)$, mais que $\bar{j} \neq \arg\min_{j \in M_{out}} H(j)$ ou $\bar{l} \neq \arg\min_{l \in M_{in}} G(l)$. Examinons les trois cas de figure possibles suivants :

Cas 1. Si $H(\bar{j}) > \min_{j \in M_{out}} H(j)$ et $G(\bar{l}) > \min_{l \in M_{in}} G(l)$, alors se référant aux propriétés 4.4 et 4.5, et à la remarque ci-dessus, nous obtenons facilement une contradiction :

$$\begin{aligned} \Delta_\tau(x_p) = \Delta(\bar{i},\bar{r},\bar{j},\bar{l}) &\geq constante_\tau + G(\bar{l}) + H(\bar{j}) \\ &> constante_\tau + G(l^*) + H(j^*) = \Delta_\tau(x_p). \end{aligned}$$

Cas 2. Si $H(\bar{j}) > \min_{j \in M_{out}} H(j)$ et $\bar{l} = \arg\min_{l \in M_{in}} G(l)$, alors la contradiction se déduit immédiatement comme dans le premier cas.

Cas 3. Si $\bar{j} = \arg\min_{j \in M_{out}} H(j)$ et $G(\bar{l}) > \min_{l \in M_{in}} G(l)$, alors la contradiction se déduit comme dans les deux cas précédents.

■

Les résultats précédents nous permettent de proposer une fouille du voisinage mieux dirigée, en considérant d'abord les réaffectations (i,r,j^*,l^*) impliquant les matchs $j^* = \arg\min_{j \in M_{out}} H(j)$ et $l^* = \arg\min_{l \in M_{in}} G(l)$. Ceci donne lieu à une forme révisée de la méthode telle que résumée dans la figure 4.9

Initialisation
x_p^0, une solution initiale
$x_p := x_p^0$, la solution courante
$x_p^* := x_p^0$, la meilleure solution générée au cours de l'étape 2
Γ_p, l'ensemble des ρ meilleures solutions générées jusqu'à présent
$LT := \emptyset$, la liste tabou
$niter := 0$, le nombre d'itérations successives sans améliorer la fonction objectif
$fin :=$ **faux**
$\tau := 1$, l'indice du sous-ensemble de réaffectations utilisé présentement
Algorithme
Tant que ($fin =$ **faux**) **faire**
  $niter := niter + 1$, $Z := \emptyset$
  EXPLORATION DES VOISINS
  Identifier les matchs $j^* = \arg\min_{j \in M_{ow}} H(j)$ et $l^* = \arg\min_{l \in M_{in}} G(l)$
  Évaluer la valeur de la meilleure modification $\Delta_\tau(x_p)$
  $V_\tau^*(x_p) = \{x_p \oplus (i, r, j, l) \quad : \quad j = j^* \text{ et } l = l^*\}$
  $s := 1$, $fin-rech :=$ **faux**
  Tant que ($s \leq |V_\tau^*(x_p)|$ **ET** $fin - rech =$ **faux**) **faire**
    $x'_p \in V_\tau^*(x_p)$ (considérées séquentiellement)
    Si (x'_p n'est pas tabou **OU** vérifie le critère d'aspiration) **Alors**
      Si ($f(x'_p) - f(x_p) = \Delta_\tau(x_p)$) **Alors**
        $Z := \{x'_p\}$, $fin - rech :=$ **vrai**
      Sinon
        $Z := Z \cup \{x'_p\}$, $s := s + 1$
      Fin Si
    Sinon
      $s := s + 1$
    Fin Si
  Fait
  Si ($fin - rech =$ **faux**) **Alors**
    Pour chaque solution voisine réalisable $x'_p \in V_\tau(x_p) - V_\tau^*(x_p)$ **faire**
      Si (x'_p n'est pas tabou **OU** vérifie le critère d'aspiration) **Alors**
        $Z := Z \cup \{x'_p\}$
      Fin Si
    Fin Pour
  Fin Si
  REMPLACEMENT DE LA SOLUTION COURANTE
  Si ($Z \neq \emptyset$) **Alors** $x_p := \arg\min_{z \in Z} \{f(z)\}$
  SÉLECTION DE LA PROCHAINE STRUCTURE DE VOISINAGE
  Si ($Z \neq \emptyset$) **Alors**
    Si ($\tau = 4$) **Alors** $\tau := 2$
    Si ($\tau \geq 5$) **Alors** $\tau := 1$
  Sinon
    Si ($\tau < 8$) **Alors** $\tau := \tau + 1$ **Sinon** $\tau := 1$
  Fin Si
  MISE À JOUR
  Si ($Z \neq \emptyset$) **Alors**
    Mettre à jour LT et Γ_p
    Si ($f(x_p) < f(x_p^*)$) **Alors** $x_p^* := x_p$, $niter := 0$
  Fin Si
  ÉVALUATION DES CRITÈRES D'ARRÊT
  Si ($|M_5(x_p)| = M$ **OU** ($|M_1(x_p)| = 0$ **ET** $I((M+1)_p, x_p) = 0$ ou 1) **OU** $niter = nitermax$) **Alors** $fin :=$ **vrai**
Fait
x_p^* est la meilleure solution générée au cours de l'étape 2

Figure 4.9 – Forme révisée de l'étape 2 de la RTVS

4.6.6 Recherche avec tabous

La solution générée à la seconde étape est utilisée pour initialiser la recherche avec tabous de la dernière étape. L'objectif consiste maintenant à réduire le nombre de violations des contraintes d'*équilibre*, d'*affiliation*, de *diversité*, de *couplage*, et de *séquence* dans les matchs de la ronde p.

Le voisinage est défini comme au paragraphe 3.6.4, i.e., il est généré en échangeant l'affectation de deux juges i et r affectés présentement à deux matchs différents j et l. Nous devons toutefois modifier la définition d'un échange réalisable pour tenir compte du fait qu'il faut que chaque match comporte au moins un juge *expert* en plus d'un *juge en chef* (bien entendu si le match compte 3 ou 5 juges affectés). Ainsi, l'échange est réalisable et la solution générée appartient au voisinage de x_p si et seulement si :

- les juges i et r sont respectivement admissibles aux matchs l et j :

$$a_{ilp} = 1 \quad \text{et} \quad a_{rjp} = 1$$

- il existe au moins un *juge en chef* dans chacun des matchs j et l :

$$\exists\, \bar{i} \in I(j,x_p) - \{i\} \cup \{r\} \quad \text{tel que} \quad l_{\bar{i}} = 1$$

$$\exists\, \bar{\bar{i}} \in I(l,x_p) - \{r\} \cup \{i\} \quad \text{tel que} \quad l_{\bar{\bar{i}}} = 1$$

- si le match j (respectivement l) compte 3 ou 5 juges affectés, il existe au moins un juge *expert*, autre que le *juge en chef*, affecté à j (respectivement à l) :

$$\exists i' \in I(j,x_p) - \{i\} \cup \{r\} \text{ tel que soit } (l_{i'} = 1 \text{ et } i' \neq \bar{i}) \text{ ou bien } (v_{i'} = 1)$$

(respectivement $\exists i'' \in I(l,x_p) - \{r\} \cup \{i\}$ tel que soit $(l_{i''} = 1$ et $i'' \neq \bar{\bar{i}})$ ou bien $(v_{i''} = 1)$.)

Comme il a déjà été souligné au paragraphe 4.6.2, nous utilisons la stratégie *meilleure amélioration* comme stratégie de sélection dans le voisinage. Contrairement à la méthode décrite au paragraphe 3.6.4, fixer la valeur de la meilleure modification à une valeur

prédéterminée et interrompre la génération du voisinage dès qu'une solution voisine induisant cette modification est atteinte, ne permet malheureusement pas d'accélérer la phase d'exploration. En effet, nous avons constaté lors des tests préliminaires que, le plus souvent, le voisinage au complet est généré car aucune solution voisine n'induit la modification souhaitée ; la solution courante étant de bonne qualité et les contraintes du problème n'étant pas nécessairement toutes violées. Aussi, plutôt que de générer le voisinage au complet, proposons-nous de concentrer l'exploration sur les régions les plus prometteuses de celui-ci. En fait, une partition du voisinage permet parfois d'ignorer plusieurs solutions voisines.

Cette modification de la stratégie d'exploration du voisinage fait l'objet des résultats qui suivent. Nous montrons comment l'algorithme proposé (dénoté par *FR*) n'exclut aucune solution voisine et permet d'accélérer en moyenne l'étape 3 comparativement à l'algorithme *FC* où le voisinage complet est généré à chaque itération. Nous montrons également comment cet algorithme (*FR*) peut se révéler plus efficace que l'algorithme *FC* pour effectuer une recherche plus exhaustive de l'ensemble des solutions.

Avant de présenter les deux algorithmes, quelques définitions et notations doivent être introduites. Dénotons par ϑ l'ensemble de tous les échanges (réalisables et non réalisables) impliquant deux juges i et r affectés à deux matchs différents j et l. Nous dirons qu'un échange $(i,j,r,l) \in \vartheta$ est *autorisé* s'il est réalisable et non tabou, ou s'il est réalisable, tabou, mais vérifie le critère d'aspiration.

Pour chaque échange $(i,j,r,l) \in \vartheta$, dénotons par $\Delta(i,j,r,l)$ la modification induite dans la fonction objectif lorsqu'on applique cet échange à la solution courante x_p. Notons qu'un facteur de pénalité très élevé est accordé aux échanges non réalisables. De plus, soit $\Delta_{min} = \min_{(i,j,r,l)\in\vartheta} \{\Delta(i,j,r,l) : \quad (i,j,r,l) \text{ est } autorisé\}$, la meilleure modification pouvant être induite par tous les échanges *autorisés* $(i,j,r,l) \in \vartheta$ à une itération donnée.

Partitionnons également l'ensemble des échanges ϑ en quatre ensembles disjoints :

- $\vartheta_1 = \{(i,j,r,l) \quad : \quad (i,j,r,l)$ est *autorisé* et $\Delta(i,j,r,l) = \Delta_{min}\}$
- $\vartheta_2 = \{(i,j,r,l) \quad : \quad (i,j,r,l)$ n'est pas *autorisé* et $\Delta(i,j,r,l) = \Delta_{min}\}$
- $\vartheta_3 = \{(i,j,r,l) \quad : \quad (i,j,r,l)$ n'est pas *autorisé* et $\Delta(i,j,r,l) < \Delta_{min}\}$
- $\vartheta_4 = \{(i,j,r,l) \quad : \quad \Delta(i,j,r,l) > \Delta_{min}\}$.

Pour obtenir la prochaine solution courante, plutôt que de générer le voisinage au complet et de retenir le premier échange *autorisé* induisant la modification Δ_{min} (algorithme *FC*), dans l'algorithme *FR*, nous choisissons aléatoirement un échange $(i,j,r,l) \in \vartheta_1$. Par contre, la phase de mise à jour dans l'algorithme *FR* requiert en plus la mise à jour de Δ_{min}, et des ensembles ϑ_1, ϑ_2 et ϑ_3. Les deux algorithmes *FC* et *FR* sont résumés dans les figures 4.10 et 4.11 respectivement.

Nous montrons d'abord que la solution pouvant être obtenue par l'algorithme *FC* peut également être obtenue en utilisant l'algorithme *FR*, et que cet algorithme (*FR*) n'exclut aucune solution voisine.

Propriété 4.7. *Soit $x_p \oplus (i,j,r,l)$ la solution choisie comme prochaine solution courante en utilisant l'algorithme FC. Alors, $(i,j,r,l) \in \vartheta_1$.*

Preuve. Trivialement, il est clair que l'échange (i,j,r,l) ne fait partie ni de ϑ_2 ni de ϑ_3 car il ne serait pas *autorisé*, et donc, il ne pourrait être retenu en utilisant l'algorithme *FC*. De même, $(i,j,r,l) \notin \vartheta_4$. En effet, dans le cas contraire (i.e., si $(i,j,r,l) \in \vartheta_4$), alors, si (i,j,r,l) n'est pas *autorisé* le même raisonnement précédent s'applique. Sinon (i.e., si $(i,j,r,l) \in \vartheta_4$ est *autorisé*), il existerait un autre échange *autorisé* $(\bar{i},\bar{j},\bar{r},\bar{l})$ tel que $\Delta(\bar{i},\bar{j},\bar{r},\bar{l}) = \Delta_{min} < \Delta(i,j,r,l)$, et donc, $x_p \oplus (i,j,r,l)$ ne pourrait être retenue comme prochaine solution courante en utilisant l'algorithme *FC*.
Ainsi, puisque $\vartheta = \vartheta_1 \cup \vartheta_2 \cup \vartheta_3 \cup \vartheta_4$, alors nécessairement $(i,j,r,l) \in \vartheta_1$. ■

Propriété 4.8. *Soient x_p la solution courante et $(i,j,r,l) \in \vartheta_1$. Alors, $x_p \oplus (i,j,r,l)$ pourrait être choisie comme prochaine solution courante en utilisant l'algorithme FC.*

Preuve. Par absurde. Supposons que $x_p \oplus (i,j,r,l)$ ne peut être choisie comme prochaine solution courante en utilisant l'algorithme *FC*. Alors, soit (i,j,r,l) n'est pas *autorisé*, soit il existe un autre échange *autorisé* $(\bar{i},\bar{j},\bar{r},\bar{l})$ induisant une meilleure modification de la fonction objectif (i.e., $\Delta(\bar{i},\bar{j},\bar{r},\bar{l}) < \Delta(i,j,r,l)$). Dans les deux cas une contradiction s'ensuit de la définition de Δ_{min} et de ϑ_1. ■

Initialisation
x_p^*, la meilleure solution générée à l'étape 2, et $x_p := x_p^*$, la solution courante
$x_p^{**} := x_p^*$, la meilleure solution générée au cours de l'étape 3
Δ, la matrice contenant la valeur des échanges
Γ_p, l'ensemble des ρ meilleures solutions générées jusqu'à présent
$LT := \emptyset$, la liste tabou
$niter := 0$, le nombre d'itérations successives sans améliorer la fonction objectif
$fin :=$ **faux**
Algorithme
Tant que ($fin =$ **faux**) **faire**
  $niter := niter + 1$, $E := \emptyset$, $\Delta_{min} := \infty$, $s := 1$
  EXPLORATION DES VOISINS
  Tant que ($s \leq |\vartheta|$) **faire**
    $(i,j,r,l) \in \vartheta$ (considérés séquentiellement)
    Si ((i,j,r,l) est *autorisé*) **Alors**
      Si ($\Delta(i,j,r,l) < \Delta_{min}$) **Alors**
        $\Delta_{min} := \Delta(i,j,r,l)$, $E := \{(i,j,r,l)\}$, $s := s+1$
      Sinon
        $s := s+1$
      Fin Si
    Sinon
      $s := s+1$
    Fin Si
  Fait
  REMPLACEMENT DE LA SOLUTION COURANTE
  Si ($E \neq \emptyset$) **Alors**
    $x_p := x_p \oplus (i,j,r,l)$ ((i,j,r,l), étant l'échange contenu dans E)
  Fin Si
  MISE À JOUR
  Si ($E \neq \emptyset$) **Alors**
    Mettre à jour LT, Δ et Γ_p
    Si ($f(x_p) < f(x_p^{**})$) **Alors**
      $x_p^{**} := x_p$, $niter := 0$
    Fin Si
  Fin Si
  ÉVALUATION DES CRITÈRES D'ARRÊT
  Si ($f(x_p) = b_{inf}$ **OU** $niter = nitermax$) **Alors** $fin :=$ **vrai**
Fait
x_p^{**} est la meilleure solution générée au cours de l'étape 3

Figure 4.10 – Forme classique de l'étape 3 de la RTVS (algorithme FC)

Initialisation
x_p^*, la meilleure solution générée à l'étape 2, et $x_p := x_p^*$, la solution courante
$x_p^{**} := x_p^*$, la meilleure solution générée au cours de l'étape 3
Δ_{min}, la valeur de la meilleure modification en considérant l'ensemble des échanges *autorisés*
$\vartheta_1 = \{(i,j,r,l) \quad : \quad (i,j,r,l)$ est *autorisé* et $\Delta(i,j,r,l) = \Delta_{min}\}$
$\vartheta_2 = \{(i,j,r,l) \quad : \quad (i,j,r,l)$ n'est pas *autorisé* et $\Delta(i,j,r,l) = \Delta_{min}\}$
$\vartheta_3 = \{(i,j,r,l) \quad : \quad (i,j,r,l)$ n'est pas *autorisé* et $\Delta(i,j,r,l) < \Delta_{min}\}$
Γ_p, l'ensemble des ρ meilleures solutions générées jusqu'à présent
$LT := \emptyset$, la liste tabou
$niter := 0$, le nombre d'itérations successives sans améliorer la fonction objectif
$fin :=$ **faux**
Algorithme
Tant que $(fin =$ **faux**$)$ **faire**
    $niter := niter + 1$
    Si $(\vartheta_1 \neq \emptyset)$ **Alors**
        CHOIX D'UN ÉCHANGE
        Choisir aléatoirement $(i,j,r,l) \in \vartheta_1$
        REMPLACEMENT DE LA SOLUTION COURANTE
        $x_p := x_p \oplus (i,j,r,l)$
    Fin Si
    MISE À JOUR
    Si $(\vartheta_1 \neq \emptyset)$ **Alors**
        Mettre à jour LT et Γ_p
        Si $(f(x_p) < f(x_p^{**}))$ **Alors**
            $x_p^{**} := x_p$, $niter := 0$
        Fin Si
    Fin Si
    Mettre à jour Δ_{min} ainsi que les ensembles ϑ_1, ϑ_2 et ϑ_3
    ÉVALUATION DES CRITÈRES D'ARRÊT
    Si $(f(x_p) = b_{inf}$ **OU** $niter = nitermax)$ **Alors** $fin :=$ **vrai**
Fait
x_p^{**} est la meilleure solution générée au cours de l'étape 3

Figure 4.11 – Forme révisée de l'étape 3 de la RTVS (algorithme FR)

Propriété 4.9. *En utilisant l'algorithme FR, toutes les solutions voisines de la solution courante x_p sont explorées.*

Preuve. À chaque itération de l'algorithme FR, la valeur de Δ_{min} est mise à jour et pour ce faire, tous les échanges $(i,j,r,l) \in \vartheta$ sont considérés. Par conséquent, toutes les solutions voisines de la nouvelle solution courante sont explorées. ■

Comme nous l'avons mentionné plus haut, comparativement à l'algorithme FC, la phase de mise à jour dans l'algorithme FR requiert en plus la mise à jour de Δ_{min}, et des ensembles ϑ_1, ϑ_2 et ϑ_3. Nous montrons que cette mise à jour supplémentaire n'augmente pas la complexité de l'algorithme FR par rapport à l'algorithme FC et que le pire cas de cette phase est $O(N_p^2)$ où N_p est le nombre de juges disponibles pour la ronde p, ce qui est similaire à la complexité de la phase d'exploration des voisins dans l'algorithme FC. Néanmoins, en moyenne, l'algorithme FR permet d'accélérer l'étape 3 du processus de résolution des sous-problèmes associés aux rondes.

Propriété 4.10. *La mise à jour de Δ_{min} n'augmente pas la complexité de l'algorithme.*

Preuve. La mise à jour de Δ_{min} nécessite de considérer tous les échanges $(i,j,r,l) \in \vartheta$, ce qui est similaire à la mise à jour de la matrice Δ dans l'algorithme FC. ■

Propriété 4.11. *Au pire des cas, la mise à jour des ensembles ϑ_1, ϑ_2 et ϑ_3 nécessite la génération du voisinage de la nouvelle solution courante au complet.*

Preuve. Pour faciliter la présentation, dénotons par x_p la solution courante et par $x'_p := x_p \oplus (i',j',r',l')$ la solution retenue par l'algorithme FR comme solution courante pour la prochaine itération. Dénotons par $\widetilde{\Delta}(i,j,r,l)$ la valeur de la modification induite dans la fonction objectif si on applique l'échange (i,j,r,l) à la prochaine itération. Observons que pour tout échange $(i,j,r,l) \in \vartheta$, la valeur de la modification correspondante ne change que s'il implique des juges affectés à l'un des matchs j' ou l'.
Partitionnons donc l'ensemble de tous les échanges ϑ en deux sous-ensembles disjoints $\overline{\vartheta}$ et $\overline{\overline{\vartheta}}$:

- $\overline{\vartheta} = \{(i,j,r,l) \quad : \quad j = j' \text{ ou } l = l'\}$
- $\overline{\overline{\vartheta}} = \{(i,j,r,l) \quad : \quad j \neq j' \text{ et } l \neq l'\}$.

Ainsi,

$$\vartheta = \bigcup_{s=1}^{4} \left((\vartheta_s \cap \overline{\vartheta}) \cup (\vartheta_s \cap \overline{\overline{\vartheta}}) \right) \tag{4.77}$$

Soit $\widetilde{\Delta}_{min} = \min_{(i,j,r,l)\in\vartheta} \widetilde{\Delta}(i,j,r,l)$ la valeur de la meilleure modification pouvant être induite dans la fonction objectif à la prochaine itération, et dénotons par :

- $\widetilde{\vartheta}_1 = \{(i,j,r,l) \quad : \quad (i,j,r,l) \text{ est } \textit{autorisé} \text{ et } \widetilde{\Delta}(i,j,r,l) = \widetilde{\Delta}_{min}\}$
- $\widetilde{\vartheta}_2 = \{(i,j,r,l) \quad : \quad (i,j,r,l) \text{ n'est pas } \textit{autorisé} \text{ et } \widetilde{\Delta}(i,j,r,l) = \widetilde{\Delta}_{min}\}$
- $\widetilde{\vartheta}_3 = \{(i,j,r,l) \quad : \quad (i,j,r,l) \text{ n'est pas } \textit{autorisé} \text{ et } \widetilde{\Delta}(i,j,r,l) < \widetilde{\Delta}_{min}\}$.

Examinons les trois cas de figure possibles suivants :

Cas 1. $\widetilde{\Delta}_{min} < \Delta_{min}$

Il est clair que si $(i,j,r,l) \in \vartheta_1 \cap \overline{\overline{\vartheta}}$ ou si $(i,j,r,l) \in \vartheta_2 \cap \overline{\overline{\vartheta}}$, alors il ne fait partie ni de $\widetilde{\vartheta}_1$ ni de $\widetilde{\vartheta}_2$ ni de $\widetilde{\vartheta}_3$. En effet, dans ce cas :

$$\widetilde{\Delta}(i,j,r,l) = \Delta(i,j,r,l) = \Delta_{min} > \widetilde{\Delta}_{min}.$$

Le même raisonnement s'applique pour les échanges $(i,j,r,l) \in \vartheta_4 \cap \overline{\overline{\vartheta}}$ puisque dans ce cas :

$$\widetilde{\Delta}(i,j,r,l) = \Delta(i,j,r,l) > \Delta_{min} > \widetilde{\Delta}_{min}.$$

Ainsi, se référant à (4.77), pour déterminer les ensembles $\widetilde{\vartheta}_1$, $\widetilde{\vartheta}_2$ et $\widetilde{\vartheta}_3$, il suffit de considérer les échanges :

$$\begin{aligned} (i,j,r,l) \quad &\in \quad (\vartheta_1 \cap \overline{\vartheta}) \cup (\vartheta_2 \cap \overline{\vartheta}) \cup (\vartheta_3 \cap \overline{\vartheta}) \cup (\vartheta_3 \cap \overline{\overline{\vartheta}}) \cup (\vartheta_4 \cap \overline{\vartheta}) \\ &= \quad \overline{\vartheta} \cup (\vartheta_3 \cap \overline{\overline{\vartheta}}). \end{aligned}$$

Cas 2. $\widetilde{\Delta}_{min} = \Delta_{min}$

Il est inutile de considérer les échanges $(i,j,r,l) \in \vartheta_4 \cap \overline{\overline{\vartheta}}$ car :

$$\widetilde{\Delta}(i,j,r,l) = \Delta(i,j,r,l) > \Delta_{min} = \widetilde{\Delta}_{min}.$$

Ainsi, se référant encore une fois à (4.77), pour déterminer les ensembles $\widetilde{\vartheta}_1$, $\widetilde{\vartheta}_2$ et $\widetilde{\vartheta}_3$, il suffit de considérer les échanges :

$$(i,j,r,l) \in \overline{\vartheta} \cup \left((\vartheta_1 \cup \vartheta_2 \cup \vartheta_3) \cap \overline{\overline{\vartheta}} \right).$$

Cas 3. $\widetilde{\Delta}_{min} > \Delta_{min}$

C'est le seul cas où, pour déterminer des ensembles $\widetilde{\vartheta}_1$, $\widetilde{\vartheta}_2$ et $\widetilde{\vartheta}_3$, il faut considérer tous les échanges $(i,j,r,l) \in \vartheta$.

En somme, pour déterminer les ensembles $\widetilde{\vartheta}_1$, $\widetilde{\vartheta}_2$ et $\widetilde{\vartheta}_3$, hormis le cas 3 ($\widetilde{\Delta}_{min} > \Delta_{min}$) où nous devons générer le voisinage au complet, au lieu de générer les $\frac{N_p(N_p-1)}{2}$ solutions voisines (N_p étant le nombre de juges disponibles pour la ronde p), nous n'avons qu'à en générer $(|\overline{\vartheta}| + |\vartheta_3 \cap \overline{\overline{\vartheta}}|)$ lorsque $\widetilde{\Delta}_{min} < \Delta_{min}$, et $(|\overline{\vartheta}| + |\vartheta_1 \cap \overline{\overline{\vartheta}}| + |\vartheta_2 \cap \overline{\overline{\vartheta}}| + |\vartheta_3 \cap \overline{\overline{\vartheta}}|)$ lorsque $\widetilde{\Delta}_{min} = \Delta_{min}$. Notons que :

$$\begin{aligned} |\overline{\vartheta}| &= |\{(i,j,r,l) \quad : \quad j = j' \text{ ou } l = l'\}| \\ &\leq 10(N_p - 5). \end{aligned}$$

De plus, en considérant les échanges $(i,j,r,l) \in \overline{\overline{\vartheta}}$, il suffit de vérifier leur statut (si l'échange est *autorisé* ou non) puisqu'on sait d'ores et déjà que la valeur de la modification pouvant être induite dans la fonction objectif si on applique cet échange à la prochaine itération n'a pas changé (i.e., $\widetilde{\Delta}(i,j,r,l) = \Delta(i,j,r,l)$). ■

Finalement, notons que l'utilisation de l'algorithme *FR* plutôt que l'algorithme *FC* à l'étape 3 du processus de résolution du sous-problème associé à une ronde permet de diversifier davantage la recherche et pourrait remédier aux problèmes de cyclage. En effet, vu la structure du problème, plusieurs solutions voisines ont la même valeur et les optima locaux sont nombreux. Par conséquent, au cours de la recherche nous sommes souvent dans le cas 2 de la propriété 4.11 ($\widetilde{\Delta}_{min} = \Delta_{min}$). Nous avons vu que dans ce cas, mettre à jour les ensembles $\widetilde{\vartheta}_1$, $\widetilde{\vartheta}_2$ et $\widetilde{\vartheta}_3$ ne nécessite pas la génération du voisinage au complet. Ainsi, l'algorithme *FR* permet d'accélérer en moyenne l'étape 3 du processus de résolution du sous-problème associé à une ronde (comparativement à l'algorithme *FC*). Par

ailleurs, étant donné que le critère d'arrêt de la procédure de résolution est spécifié en termes d'un temps maximal alloué, ceci permet d'effectuer en moyenne un plus grand nombre d'itérations de la phase 2 (cf. figure 4.2), et donc, une recherche plus exhaustive de l'ensemble des solutions.

D'autre part, supposons qu'à une itération de la phase 2, la solution utilisée pour amorcer l'étape 3 de la procédure de résolution (obtenue à l'issue de l'étape 2) est similaire à celle obtenue à une itération antérieure. Dans ce cas, plutôt que de choisir la première solution rencontrée induisant la meilleure modification (algorithme *FC*), en utilisant l'algorithme *FR*, nous sélectionnons l'une de ces solutions aléatoirement ce qui peut remédier aux problèmes de cyclage. En réduisant le risque de cycler, une exploration plus diversifiée de l'ensemble des solutions pourrait être faite.

4.6.7 Expérimentations numériques

Nous comparons numériquement trois variantes différentes de la procédure de résolution. Comme dans la section 4.5, ces variantes sont spécifiées en termes de la solution utilisée lors de la résolution du sous-problème associé à la ronde p pour fixer les affectations dans les rondes $q \neq p$ (*xbestLast* et *xbestG*), et la façon utilisée pour spécifier l'ensemble Ω des rondes à considérer à chaque itération majeure de la phase 2 (les rondes comportant des violations et toutes les rondes) : **RTVS-Div-Partiel**, **RTVS-Int-Partiel** et **RTVS-Div-Global**. Notons qu'ici aussi, nous ne considérons pas la variante **RTVS-Int-Global**, i.e., la combinaison ($\{1,\ldots,P\}$, *xbestG*, RTVS), et ce pour les mêmes raisons évoquées au paragraphe 4.5.1. Par ailleurs, des expérimentations numériques préliminaires nous ont permis de constater qu'il est préférable de ne pas affecter les paires de juges additionnelles lors de la phase 1 de la procédure de résolution. Autrement dit, pour les trois variantes, au cours de cette phase, nous tentons d'affecter 3 juges à chaque match de chaque ronde en utilisant les étapes 1, 2 et 3 de l'heuristique constructive décrite au paragraphe 4.6.3. Les paires de juges additionnelles sont affectées au cours de la phase 2, lors de la résolution des sous-problèmes associés aux rondes. Finalement, pour les trois variantes, au cours des étapes 2 et 3 de la **RTVS**, nous utilisons les formes révisées dont les pseudo-codes sont donnés dans les figures 4.9 et 4.11.

Les paramètres de la recherche avec tabous à voisinages structurés (**RTVS**) sont fixés comme dans le paragraphe 3.6.6. Le paramètre de la procédure de résolution *tempsmax* est fixé comme pour les autres variantes utilisant la recherche avec tabous (**RT**) et la recherche à voisinage variable (**RVV**) à savoir $3N$ secondes (N étant le nombre total de juges).

Les tests sont complétés en utilisant l'ensemble des instances décrites au paragraphe 4.2.1. Chaque instance est résolue 10 fois avec une valeur d'initialisation différente pour le générateur des nombres aléatoires. Les mêmes valeurs sont utilisées pour chaque variante.

Pour évaluer et comparer les trois variantes, nous utilisons les mêmes critères de comparaison considérés pour les variantes utilisant la **RT** et la **RVV** : l'écart moyen entre les solutions générées et les solutions «optimales» en termes du nombre de violations des différentes règles souples d'affectation ($Ave\,dev-Nbre$, $Ave\,dev-Seq$, $Ave\,dev-Coup$, $Ave\,dev-Div$, $Ave\,dev-Aff$ et $Ave\,dev-Equ$), l'écart relatif moyen entre les solutions générées et la solution optimale de la relaxation linéaire ($\%Ave\,Gap$), le pourcentage des solutions réalisables obtenues ($\%Feas$), le pourcentage des solutions «optimales» trouvées ($\%Opt$), et le temps moyen de résolution en secondes ($Ave\,CPU$). Toutes les solutions générées par les trois variantes comptent 5 juges affectés au maximum de matchs et sont toujours réalisables. Par conséquent, les critères $Ave\,dev-Nbre$ et $\%Feas$ ne figurent pas dans le tableau 4.10 qui récapitule les résultats obtenus pour chaque sous-ensemble de problèmes. Ce tableau a la même structure que le tableau 4.5 présenté au paragraphe 4.4.5.

À l'instar des variantes utilisant la **RT** et la **RVV**, les trois variantes utilisant la **RTVS** surclassent CPLEX. Elles parviennent toujours à générer des solutions d'excellente qualité comme l'indiquent les faibles valeurs de $\%Ave\,Gap$, et le gain en termes du temps de résolution est très considérable. En particulier, pour les problèmes du sous-ensemble P_1 avec 30 matchs par ronde, les valeurs nulles de $\%Ave\,Gap$ confirment que toutes les solutions générées par les trois variantes sont optimales. $Ave\,CPU$ est, pour sa part, réduit d'un facteur de 804, 218, et 153 respectivement pour les variantes **RTVS-Div-Global**, **RTVS-Div-Partiel**, et **RTVS-Int-Partiel**. Qui plus est, aucune solution entière n'a pu

		Taille	**RTVS-Div-Partiel**	**RTVS-Int-Partiel**	**RTVS-Div-Global**	CPLEX
Ave dev − Seq	P_1	15	2,30	2,36	2,30	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	14,90	15,07	14,90	NA
		30	0	0	0	NA
		90	0	0	0	NA
Ave dev − Coup	P_1	15	0	0,13	0	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	14,54	14,59	14,54	NA
		30	0	0	0	NA
		90	0	0	0	NA
Ave dev − Div	P_1	15	1,80	3,99	1,78	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	1,28	0,25	1,25	NA
		30	0	0	0	NA
		90	0	0	0	NA
Ave dev − Aff	P_1	15	1,86	0,58	2,06	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	7,51	1,70	7,51	NA
		30	0	0	0	NA
		90	0	0	0	NA
Ave dev − Equ	P_1	15	3,43	0,06	3,84	NA
		30	0	0	0	NA
		90	0	0	0	NA
	P_2	15	40,16	17,30	40,18	NA
		30	2,54	0,66	6,76	NA
		90	0	0	0	NA
%Ave Gap	P_1	15	6,25E-03	6,65E-03	6,25E-03	NA
		30	0	0	0	NA
		90	-	-	-	NA
	P_2	15	0,17	1,30	0,17	NA
		30	NA	NA	NA	NA
		90	-	-	-	NA
%Opt	P_1	15	10	1	4	NA
		30	100	100	100	NA
		90	100	100	100	NA
	P_2	15	0	0	0	NA
		30	33	71	0	NA
		90	100	100	100	NA
Ave CPU (sec)	P_1	15	164,70	178,31	174,87	982,46
		30	4,09	5,82	1,11	893,05
		90	3,47	3,43	3,46	-
	P_2	15	225,18	225,14	225,17	350,83
		30	344,25	181,17	450,77	2 400,34
		90	4,52	4,50	4,84	-

Tableau 4.10 – Comparaison de l'efficacité des variantes utilisant la RTVS en fonction des sous-ensembles de problèmes

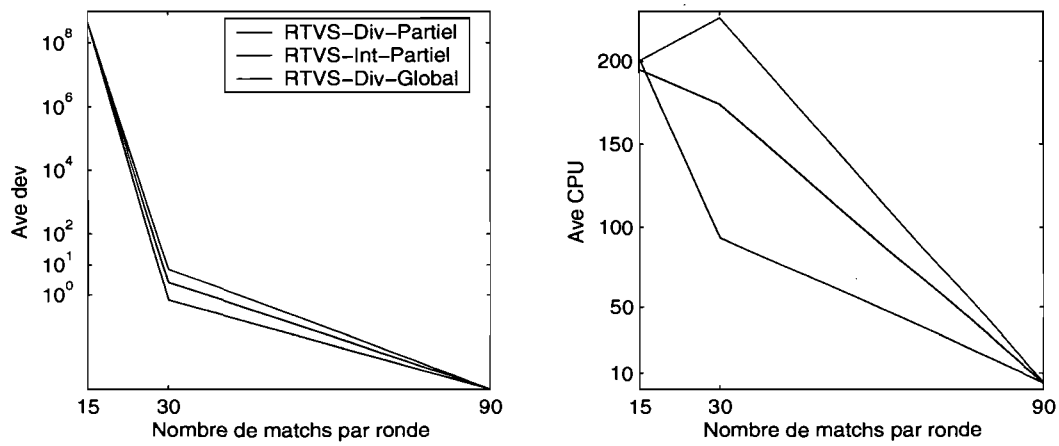

Figure 4.12 – Comparaison de l'efficacité des variantes utilisant la RTVS en fonction de la taille des problèmes

être identifiée avec CPLEX dans une limite de temps de 10 heures, et le *AveCPU* reporté dans la dernière colonne du tableau 4.10 représente le temps moyen requis pour résoudre la relaxation linéaire.

Pour comparer les trois variantes entre elles, nous nous référons aux résultats du tableau 4.10 et à la figure 4.12 qui montre l'évolution de l'écart moyen entre la valeur des solutions obtenues et la valeur «optimale», et l'évolution du temps moyen de résolution en fonction de la taille des problèmes. Globalement, les résultats sont très similaires au cas des variantes utilisant la **RVV**. En effet, là encore, des trois variantes considérées, la variante **RTVS-Int-Partiel** semble être la moins performante (cf. tableau 4.10). On note toutefois que, pour les problèmes du sous-ensemble P_2 avec 30 matchs par ronde, les temps de résolution de cette variante sont sensiblement moins longs que ceux des deux autres variantes, et les solutions qu'elle génère sont légèrement meilleures.

Quant aux deux variantes **RTVS-Div-Partiel** et **RTVS-Div-Global**, les résultats sont en tous points similaires au cas des variantes utilisant la **RVV** : pour les problèmes avec 15 matchs par ronde (qui semblent être les plus difficiles à résoudre), les deux variantes obtiennent le plus souvent les mêmes résultats, mais parfois, on note un très léger avantage en faveur de **RTVS-Div-Global** (cf. valeurs de $Avedev - Div$). Concernant les problèmes

avec 30 matchs par ronde (qui semblent être assez faciles à résoudre), les résultats sont partagés : **RTVS-Div-Global** semble bien se comporter pour les problèmes du sous-ensemble P_1. Elle l'est moins dans le cas des problèmes du sous-ensemble P_2 pour lesquels elle est surclassée par les deux autres variantes tant en termes de la qualité des solutions générées qu'en termes du temps de résolution. Finalement, pour les problèmes avec 90 matchs par ronde (les plus faciles à résoudre), les résultats obtenus par les deux variantes sont très comparables. Les mêmes raisons évoquées au paragraphe 4.5.2 sont valables ici pour expliquer l'ensemble de ces résultats.

Pour compléter l'analyse, nous avons effectué le test des rangs pour échantillons appariés de WILCOXON sur l'ensemble des résultats obtenus. Les résultats de ce test sont résumés dans le tableau 4.11 qui a la même structure que le tableau 4.8 présenté au paragraphe 4.5.2. Rappelons que pour chaque entrée (A,B), nous indiquons la *p_valeur* obtenue que nous soulignons en caractère gras lorsque la différence entre les deux variantes A et B n'est pas statistiquement significative (i.e., si la *p_valeur* obtenue est supérieure au seuil de confiance utilisé qui est de 0,05). Une *p_valeur* inférieure à 0,05 révèle que la variante A (dont le nom figure à la colonne 1 du tableau lorsque nous comparons la qualité des solutions, ou la colonne 4 lorsqu'il s'agit du temps de résolution) est statistiquement meilleure que la variante B (dont le nom figure à la ligne 2 du tableau).

	Qualité de la solution			Temps de résolution	
	RTVS-Div-Global	**RTVS-Int-Partiel**		**RTVS-Div-Partiel**	**RTVS-Div-Global**
RTVS-Div-Partiel	6,47E-4	1,99E-6	**RTVS-Int-Partiel**	5,71E-6	3,43E-10
RTVS-Div-Global	-	9,79E-5	**RTVS-Div-Partiel**	-	**0,07**

Tableau 4.11 – Résultats du test de WILCOXON pour les variantes utilisant la RTVS

Les résultats de ce test nous permettent de conclure qu'au niveau de la qualité des solutions, **RTVS-Div-Partiel** surclasse les deux autres variantes. **RTVS-Div-Global** se classe en second et **RTVS-Int-Partiel** en dernier. En revanche, au niveau du temps de résolution, c'est cette dernière variante qui domine les deux autres, lesquelles ne sont pas statistiquement différentes.

Nous présentons dans la figure 4.13 l'évolution de *Avedev* en fonction du temps lors de la résolution des problèmes avec 15 matchs par ronde.

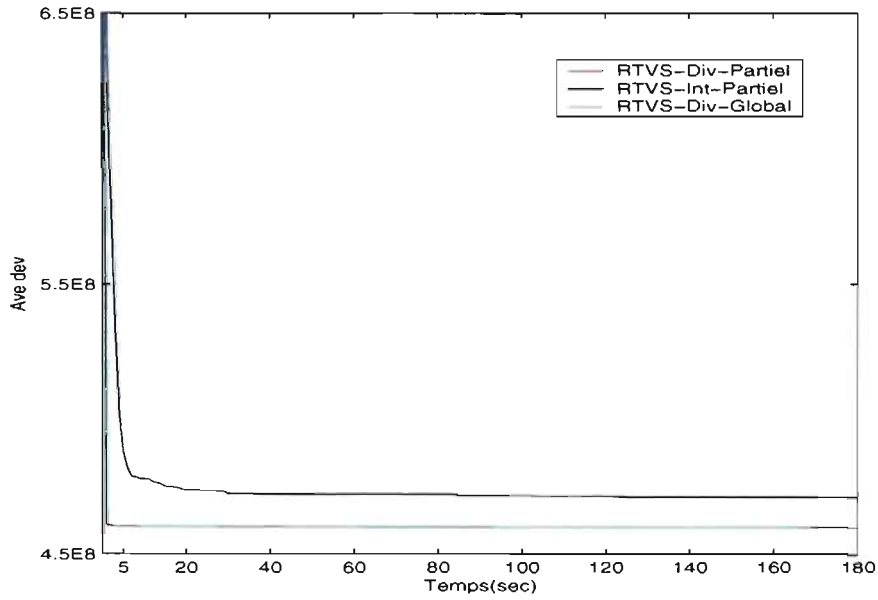

Figure 4.13 – Évolution de *Ave dev* des variantes utilisant la RTVS pour les problèmes avec 15 matchs par ronde

Globalement, les courbes associées aux trois variantes ont a peu près la même allure. Durant les deux premières secondes elles sont monotones décroissantes. On constate toutefois que **RTVS-Div-Partiel** et **RTVS-Div-Global** convergent plus rapidement que **RTVS-Int-Partiel**. Notons que, contrairement aux variantes utilisant la recherche à voisinage variable (**RVV**), la différence entre les trois variantes apparaît ici dès le début du processus de résolution, vraisemblablement en raison de la bonne qualité de la solution initiale. En effet, rappelons que dans le cas des variantes utilisant la **RVV**, la solution initiale est générée aléatoirement. Par conséquent, elle n'est pas réalisable et compte un nombre élevé de violations des différentes règles souples d'affectation. Comme nous l'avons déjà expliqué au paragraphe 4.5.2, tant que toutes les rondes comportent des violations, et que des améliorations sont obtenues chaque fois que l'on considère le sous-problème associé à une ronde, les trois variantes sont équivalentes. De fait, les différences entre les variantes utilisant la **RVV** n'apparaissent que vers la $5^{ème}$ seconde. Ceci n'est pas le cas ici puisque la solution initiale est générée en utilisant les trois premières

étapes de l'heuristique constructive présentée au paragraphe 4.6.3. Toutes les solutions produites par cette heuristique sont réalisables. Certes, 3 juges seulement sont affectés à chaque match de chaque ronde, mais le plus souvent, après la première itération de la phase 2, une paire de juges additionnelle est affectée au maximum de matchs, et les solutions obtenues comptent un nombre modéré de violations des autres règles souples d'affectation. De plus, le temps moyen requis par l'heuristique et la première itération de la phase 2 ne dépasse jamais une seconde.

Sur l'intervalle [2,5], les courbes associées aux trois variantes ont des allures différentes. Pour les variantes **RTVS-Div-Partiel** et **RTVS-Div-Global**, il semble que la bonne qualité de la solution limite l'impact de la méthode de résolution : les améliorations sont très marginales et ne concernent que la satisfaction de la contrainte d'*équilibre*. Dans le cas de la **RTVS-Int-Partiel**, les améliorations sont beaucoup plus importantes. Elles le sont toutefois moins sur l'intervalle [5,20]. En effet, durant cette période, les courbes associées aux trois variantes commencent à se stabiliser chacune autour d'une valeur donnée, laquelle est toutefois moins élevée pour les deux variantes **RTVS-Div-Partiel** et **RTVS-Div-Global** que pour **RTVS-Int-Partiel**. Ceci confirmerait l'hypothèse que cette dernière variante est plus facilement piégée dans les optima locaux. On en voit d'ailleurs la conséquence sur le graphique puisque, jusqu'à la fin du processus de résolution, *Ave dev* de **RTVS-Int-Partiel** ne parvient pas à rejoindre celles des deux autres variantes.

Finalement, sur l'intevalle [20,180], le comportement des trois variantes est à peu près identique à ce que l'on a observé pour les variantes utilisant la **RVV** : les trois variantes stagnent et les améliorations obtenues deviennent de plus en plus rares. L'écart entre les trois variantes utilisant la **RTVS** est cependant moins accentué que dans le cas des variantes utilisant la **RVV**.

Enfin, nous nous sommes intéressés à évaluer l'impact des modifications apportées à la stratégie d'exploration du voisinage (tant à l'étape 2 qu'à l'étape 3) sur les performances de la procédure de résolution. En effet, dans la version de base de la **RTVS**, à l'étape 2 comme à l'étape 3, à chaque itération, nous considérons séquentiellement les solutions voisines de la solution courante jusqu'à ce qu'une solution induisant la

meilleure modification (dont la valeur est connue *a priori*) soit atteinte, ou jusqu'à ce que tout le voisinage soit généré. La valeur de cette modification est fixée et ne dépend pas de l'état actuel de la recherche. Nous avons procédé différemment en utilisant aux étapes 2 et 3 les formes révisées présentées respectivement dans les figures 4.9 et 4.11. Comme nous l'avons mentionné dans les paragraphes 4.6.5 et 4.6.6, les modifications que nous proposons pourraient accroître l'efficacité de la procédure de résolution : la forme révisée utilisée à l'étape 2 permettrait d'accélérer l'exploration du voisinage alors que celle employée à l'étape 3 permettrait en plus d'effectuer une recherche plus exhaustive de l'ensemble des solutions.

Pour confirmer le bien fondé de nos propos, nous avons effectué un comparatif des performances des deux versions (la version que nous proposons versus la version de base). Pour ce faire, nous avons procédé comme suit :

- nous avons testé les trois variantes dans les mêmes conditions que précédemment (mêmes paramètres, mêmes valeurs d'initialisation du générateur des nombres aléatoires, 10 résolutions pour chaque instance). Les mêmes solutions initiales sont utilisées. Seules les stratégies d'exploration du voisinage diffèrent. Nous dénotons par $(Avedev)'$ et $(AveCPU)'$ respectivement l'écart moyen entre les valeurs des solutions générées et la valeur «optimale», et le temps moyen de résolution (en secondes) ;
- pour chaque sous-ensemble de problèmes et chaque variante, nous avons calculé le gain moyen en termes de la qualité des solutions ($Gdev = (Avedev)' - Avedev$), et en termes du temps de résolution ($GCPU = \frac{(AveCPU)'}{AveCPU}$). Ainsi, des valeurs de $Gdev$ et de $GCPU$ supérieures à 0 et à 1 respectivement indiquent que les modifications apportées aux stratégies d'exploration du voisinage ont été bénéfiques.

Nous reportons dans la figure 4.14 les résultats obtenus. Les deux graphiques représentent respectivement, en abscisse le nombre de matchs par ronde, et en ordonnée les valeurs de $Gdev$ et $GCPU$. Chaque courbe est associée à une variante.

Globalement, il ressort un avantage à utiliser la version révisée plutôt que la version de base : pour les problèmes avec 15 matchs par ronde, le gain au niveau de la qualité des solutions est probant. Il l'est moins au niveau du temps de résolution car le pourcentage

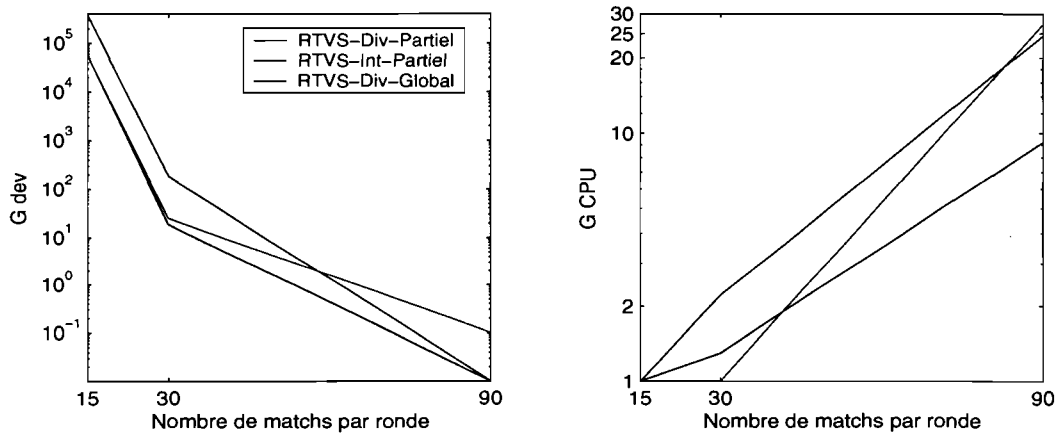

Figure 4.14 – Impact des modifications apportées aux étapes 2 et 3 sur les performances des variantes utilisant la RTVS

des solutions «optimales» obtenues est quasiment le même. Pour les problèmes avec 30 matchs par ronde, on note aussi une amélioration de la qualité des solutions, certes moins importante, mais qui est accompagnée le plus souvent d'une réduction du temps de résolution. Finalement, pour les problèmes avec 90 matchs par ronde, l'amélioration au niveau de la qualité des solutions est presque négligeable, néanmoins, en termes du temps de résolution, le gain est tangible. Les améliorations les plus importantes ont lieu pour les variantes **RTVS-Div-Global** et **RTVS-Int-Partiel**.

Pour conclure, les résultats des tests réalisés indiquent que les modifications apportées aux étapes 2 et 3 de la **RTVS** contribuent considérablement à accroître l'efficacité de la procédure de résolution. En général, elles permettent de produire de meilleures solutions et accélèrent de manière significative la convergence.

4.7 Comparaison des meilleures variantes

Dans la section 4.3, nous avons introduit une approche de résolution du *problème sur plusieurs rondes* par les métaheuristiques. Se référant au schéma général de résolution présenté dans la figure 4.2, trois facteurs sont supposés influencer les performances de la procédure proposée :

- la façon de spécifier l'ensemble Ω des rondes à considérer à chaque itération majeure de la phase 2 ;
- la façon de spécifier la solution x utilisée pour fixer les affectations des rondes autres que la ronde considérée ;
- la méthode utilisée pour résoudre le sous-problème associé à une ronde.

Dans les trois dernières sections, nous avons évalué et analysé l'impact des deux premiers facteurs (Ω et x). En effet, nous avons fixé la méthode utilisée pour résoudre le sous-problème associé à une ronde (**RT**, **RVV** et **RTVS**) et, pour chaque méthode, nous avons effectué des comparaisons en faisant varier les deux autres facteurs. Au vu des résultats obtenus, il ressort qu'il est toujours plus avantageux de limiter Ω à l'ensemble des rondes comportant des violations. En effet, le fait de considérer toutes les rondes contibue parfois à obtenir de bons résultats puisqu'il permet d'effectuer une recherche plus exhaustive de l'ensemble des solutions, mais il s'accompagne d'une convergence lente puisqu'aucune amélioration ne peut être obtenue lors de la résolution des sous-problèmes associés aux rondes dont les affectations sont «optimales». De même, les résultats des expérimentations numériques indiquent qu'il est plus bénéfique de fixer les affectations des rondes, autres que la ronde considérée, à leurs valeurs courantes dans *xbestLast*, la meilleure solution récemment obtenue. En effet, il semble que les variantes utilisant cette solution arrivent plus facilement à s'échapper des optima locaux que celles utilisant *xbestG*. Ainsi, les trois meilleures variantes sont **RT-Div-Partiel**, **RVV-Div-Partiel** et **RTVS-Div-Partiel**.

Nous comparons maintenant ces variantes pour identifier cette fois-ci la meilleure méthode pour résoudre le sous-problème associé à une ronde. Le tableau 4.12 récapitule les résultats obtenus. Pour chaque taille de poblèmes, nous indiquons le pourcentage des solutions «optimales» obtenues ($\%Opt$) et le temps moyen de résolution ($AveCPU$) en secondes. Ce tableau contient également les résultats obtenus avec CPLEX. Rappelons que pour CPLEX, $AveCPU$ indique le temps moyen requis pour résoudre la relaxation linéaire, et qu'aucune solution entière n'a pu être identifiée par le solveur dans une limite de temps de 10 heures. De plus, pour les problèmes avec 90 matchs par ronde, même la solution optimale de la relaxation linéaire n'a pas pu être identifiée.

Taille	CPLEX		RT-Div-Partiel		RVV-Div-Partiel		RTVS-Div-Partiel	
	%Opt	*Ave CPU*	*%Opt*	*Ave CPU*	*%Opt*	*Ave CPU*	*%Opt*	*Ave CPU*
15	NA	666,65	0	202,97	2,50	200,17	5	194,94
30	NA	1 646,70	100	10,73	87,50	67,63	66,50	174,17
90	NA	-	100	16,99	100	17,23	100	3,99

Tableau 4.12 – Récapitulatif des résultats des meilleures variantes

À la lecture de ces résultats, il ressort qu'au niveau de la qualité des solutions, aucune variante ne domine clairement les autres. On note que la variante **RTVS-Div-Partiel** parvient plus souvent à obtenir des solutions «optimales» pour les problèmes avec 15 matchs par ronde (les plus difficiles à résoudre) alors que le contraire se produit pour ceux avec 30 matchs par ronde. Quant aux problèmes avec 90 matchs par ronde (les plus faciles à résoudre), toutes les variantes obtiennent systématiquement des solutions «optimales».

Si l'on s'attache au temps de résolution, les résultats sont également partagés : pour les problèmes avec 15 matchs par ronde, les trois variantes sont très comparables. Pour ceux avec 30 matchs par ronde, on note que la variante **RT-Div-Partiel** domine suivie par **RVV-Div-Partiel**, et finalement **RTVS-Div-Partiel**. Cependant, pour les problèmes avec 90 matchs par ronde, c'est cette dernière variante qui prend l'avantage.

Les résultats du test de WILCOXON, présentés dans le tableau 4.13, nous permettent de clarifier davantage la relation entre les trois variantes. En effet, ils indiquent la nette supériorité de la variante **RTVS-Div-Partiel** tant au niveau de la qualité des solutions générées qu'au niveau du temps de résolution. **RVV-Div-Partiel** produit de meilleures solutions que **RT-Div-Partiel**, mais ceci se fait au détriment du temps de résolution. En effet, à ce niveau, **RVV-Div-Partiel** est la variante qui affiche les plus mauvaises performances.

	Qualité de la solution			Temps de résolution	
	RVV-Div-Partiel	**RT-Div-Partiel**		**RT-Div-Partiel**	**RVV-Div-Partiel**
RTVS-Div-Partiel	5,54E-04	<2,2E-16	**RTVS-Div-Partiel**	<2,2E-16	< 2,2E-16
RVV-Div-Partiel	-	<2,2E-16	**RT-Div-Partiel**	-	1,90E-3

Tableau 4.13 – Résultats du test de WILCOXON pour les meilleures variantes

Nous avons regroupé dans la figure 4.15 l'évolution de *Ave dev* durant la résolution des problèmes avec 15 matchs par ronde. Chaque courbe est associée à une variante.

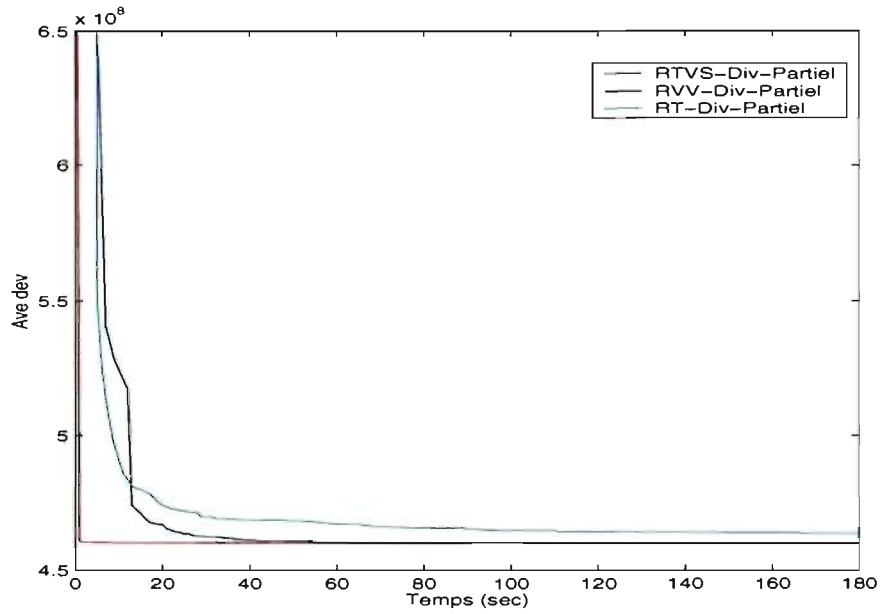

Figure 4.15 – Évolution de *Ave dev* des meilleures variantes pour les problèmes avec 15 matchs par ronde

La figure montre clairement que **RTVS-Div-Partiel** parvient à obtenir des solutions de très bonne qualité et ce dès la première seconde. Elle maintient son avance sur les deux autres variantes durant le reste du temps de résolution. Notons toutefois qu'après la 2ème seconde, les améliorations obtenues par cette variante sont très marginales. La capacité de la variante à trouver rapidement de bons optima locaux est sans doute due à la bonne qualité de la solution initiale.

Les courbes associées aux deux autres variantes ont une allure sensiblement différente de celle associée à **RTVS-Div-Partiel**. Initialement, le comportement de **RT-Div-Partiel** et **RVV-Div-Partiel** est identique, et les valeurs moyennes de *Ave dev* décroissent constamment à peu près de la même manière. Sur l'intervalle [5,15], **RT-Div-Partiel** affiche de meilleures performances que **RVV-Div-Partiel**, mais le contraire se produit sur l'inter-

valle [15, 20]. Ceci confirmerait l'hypothèse que la stratégie de diversification de la **RT** est plus efficace que la stratégie de perturbation de la **RVV** lorsque le nombre de violations des contraintes est élevé, dans la mesure où elle accélère la convergence (rappelons que les deux variantes diffèrent uniquement par la façon de générer la solution initiale pour amorcer la résolution du sous-problème associé à une ronde : diversification versus perturbation). En effet, nous avons fait les mêmes constatations au chapitre 3 (cf. section 3.7). Après les 20 premières secondes, les améliorations obtenues par les deux variantes ne sont pas très importantes et leurs courbes ont tendance à stagner, mais cette stagnation s'opère à des valeurs moins élevées de *Ave dev* pour **RVV-Div-Partiel**.
Enfin, comme on peut l'observer sur la figure 4.15, aucune variante n'a été en mesure de réduire la valeur moyenne de l'écart (*Ave dev*) à 0. En fait, nous avons constaté que pour 95% des instances considérées, aucune des 10 solutions générées par chacune des huit variantes étudiées n'est «optimale». Rappelons que nous qualifions de solution «optimale» une solution dont la valeur correspond à une borne inférieure sur la valeur optimale et que les écarts sont calculées par rapport à cette solution. Nous avons pu également constater que toutes les variantes convergent souvent vers la même solution. Toute la question est de savoir si cette solution correspond à un optimum global ou si ce n'est qu'un optimum local. Nous n'avons pas de réponse puisqu'aucune instance n'a pu être résolue avec CPLEX.

En conclusion, les résultats des différents tests réalisés tendent en faveur de **RTVS-Div-Partiel**. Elle constitue la variante la plus performante pour résoudre le *problème sur plusieurs rondes* surtout si l'on tient compte des temps de résolution. Par ailleurs, les performances affichées par les huit variantes étudiées dans ce chapitre sont sans commune mesure avec celles de CPLEX.

4.8 Conclusion

L'objectif de ce chapitre était de développer une approche de résolution du *problème sur plusieurs rondes* s'appuyant sur les méthodes proposées au chapitre précédent, et de comparer diverses variantes afin d'identifier celle qui est la plus performante.

Nous avons d'abord formulé ce problème à l'aide d'un modèle de programmation à buts multiples (*goal programming*) où la fonction objectif correspond à la somme pondérée des violations des règles souples d'affectation (ou objectifs). Cette formulation nécessite d'identifier l'importance relative des objectifs considérés, puis de déterminer un ensemble de poids traduisant cette importance. Dans notre cas, il est possible de classer les règles souples d'affectation par ordre de priorité. Nous avons donc développé un algorithme général qui permet d'obtenir un ensemble de poids de sorte à respecter la hiérarchie lexicographique des objectifs. Nous l'avons ensuite appliqué pour calculer les valeurs des poids que nous avons utilisées. Par la suite, nous avons reformulé notre problème avec des contraintes linéaires pour que celui-ci puisse être résolu avec le logiciel CPLEX. En dépit de toutes les modifications que nous avons apportées au modèle, le problème demeure très difficile à résoudre. Nous avons pu constater que même trouver une solution optimale de la relaxation linéaire du problème peut s'avérer extrêmement laborieux notamment lorsque le problème est de grande taille. Les résultats obtenus mettent donc en évidence les limites de CPLEX pour résoudre ce problème. Cette limitation qui apparaît déjà sur des problèmes aussi simples (relaxation linéaire) rend impossible la résolution de l'application réelle avec CPLEX dans une limite de temps raisonnable. Nous nous sommes alors tournés vers la résolution approchée.

Nous avons proposé des méthodes de résolution basées sur le même mécanisme fonctionnel. L'approche repose sur le principe de décomposition et nous offre donc la possibilité de résoudre séparément les sous-problèmes qui composent le problème global. Ceci permet, d'une part, de mieux gérer l'exploration du voisinage et, d'autre part, de séparer davantage les éléments reliant les rondes de ceux dépendants des sous-problèmes associés à chaque ronde. Les méthodes utilisées pour résoudre les sous-problèmes associés aux rondes se présentent comme des généralisations de celles développées au chapitre 3 puisqu'elles reprennent les mêmes principes en intégrant toutefois quelques modifications. Les modifications proposées portent aussi bien sur l'amélioration de l'existant en vue d'accroître l'efficacité de ces méthodes, que sur l'ajout de nouvelles composantes destinées à tenir compte des contraintes additionnelles que nous ne retrouvons pas dans le problème étudié dans le chapitre précédent.

Bien que les méthodes présentées dans ce chapitre soient destinées à résoudre le problème d'affectation des juges dans le contexte particulier du concours *John Molson*, la méthodologie globale demeure générique et peut être appliquée dans d'autres contextes.

CONCLUSION

Cette thèse s'inscrit dans un contexte opérationnel de résolution des problèmes d'affectation des juges lors des compétitions. En effet, nous avons illustré ce problème dans le cas particulier du *concours international d'étude de cas MBA John Molson* organisé annuellement par l'université Concordia. Plus précisément, nous avons étudié deux problèmes : *le problème sur une seule ronde* qui est une version simplifiée de l'application réelle, et *le problème sur plusieurs rondes* qui se veut plus général et plus complexe, et reflète la situation réelle rencontrée dans le contexte du concours *John Molson*. Notre objectif était de proposer des méthodes et des outils pour résoudre ces deux problèmes. Nous soulignons dans ce qui suit ce qui nous semble être les principales contributions de cette thèse.

Pour résoudre *le problème sur une seule ronde*, il a fallu d'abord élaborer un modèle mathématique qui le formalise. Nous avons formulé le problème à l'aide d'un modèle de programmation linéaire en nombres entiers que nous avons tenté de résoudre avec le logiciel CPLEX. On doit reconnaître que les résultats obtenus avec CPLEX ont été satisfaisants dans certains cas. Nous pensons en particulier aux situations où le nombre de juges disponibles est suffisant pour affecter 5 juges à chaque match. Néanmoins, lorsque cette condition n'est pas satisfaite, CPLEX s'avère moins efficace et peu d'instances ont pu être résolues à l'optimalité. Or, ce sont ces situations qui correspondent le mieux à la réalité. Une autre difficulté, mais qui était tout à fait prévisible, est liée au fait que les temps de résolution sont prohibitifs lorsqu'il s'agit d'instances de grande taille.

Ces limitations nous ont amené à résoudre le problème avec des méthodes approchées. Notre objectif consistait à proposer des méthodes de résolution, les évaluer, puis comparer leurs performances afin d'identifier la méthode la plus efficace.

Nous avons d'abord développé des heuristiques basées sur une approche séquentielle. Dans une seconde étape, nous avons mis en œuvre des méthodes de résolution basées sur les métaheuristiques : une méthode de recherche avec tabous (RT), une méthode de recherche à voisinage variable (RVV), et une méthode de recherche avec tabous à voisinages structurés (RTVS). Pour les deux premières méthodes, nous avons repris le schéma

classique des métaheuristiques RT et RVV en l'enrichissant toutefois d'éléments supplémentaires permettant d'exploiter à bon escient les informations recueillies au cours de la résolution. Les résultats des comparaisons indiquent d'ailleurs clairement la pertinence de ces ajouts. La troisième méthode que nous avons développée (RTVS) repose sur une approche hybride. Elle exploite à la fois les principes de la recherche à voisinage variable, de la recherche avec tabous, et des algorithmes génétiques.

Pour évaluer les trois méthodes basées sur les métaheuristiques, nous avons étudié quatre variantes de chaque méthode. L'analyse du comportement des différentes variantes a permis de mettre en évidence certaines faiblesses de ces méthodes. Néanmoins, il demeure que toutes les méthodes sont très efficaces et surclassent CPLEX.

Enfin, la dernière étape avait pour objet de comparer les différentes méthodes de résolution proposées. Il est généralement admis que les méthodes basées sur les métaheuristiques permettent de produire de meilleures solutions que les méthodes qui reposent sur des heuristiques, mais elles requièrent souvent plus de temps. Les méthodes de résolution que nous avons développées ne font pas exception à cette règle. En fait, il est plus intéressant de considérer les deux approches (heuristique et métaheuristique) comme complémentaires plutôt que concurrentes. C'est ce qu'indiquent les excellents résultats obtenus avec la RTVS lorsque la solution initiale est générée avec l'heuristique constructive que nous avons développée : les résultats des comparaisons révèlent que cette combinaison constitue l'approche la plus performante pour résoudre *le problème sur une seule ronde*.

Le second problème étudié dans cette thèse est plus complexe que le premier. En effet, le fait de considérer toutes les rondes augmente la combinatoire déjà importante. Le problème est également enrichi par de nouvelles contraintes. De plus, sa complexité s'est accrue par la présence de contraintes quadratiques. Il était difficile de s'attendre à une bonne performance de CPLEX. De ce fait, notre travail a été structuré autour de deux objectifs : d'une part, élaborer une approche de résolution par les métaheuristiques de manière à exploiter les résultats de notre première étude, et d'autre part, évaluer et comparer les méthodes de résolution qui en découlent. Nous avons toutefois tenté de résoudre le problème avec CPLEX. Notre objectif consistait à obtenir les valeurs optimales afin de juger de la qualité des solutions générées par les méthodes de résolution que nous

proposons. En effet, le problème ayant des spécificités particulières propres au contexte du concours *John Molson*, nous ne disposons malheureusement d'aucun référentiel pour pouvoir évaluer l'efficacité de ces méthodes.

Pour résoudre le problème avec CPLEX, il a fallu d'abord linéariser les contraintes quadratiques du modèle. Tirant parti de la structure du problème, nous avons proposé une linéarisation permettant de réduire considérablement le nombre de variables et de contraintes supplémentaires comparativement aux linéarisations classiques. Malgré toutes les modifications et les simplifications apportées pour réduire la complexité du modèle, aucune solution entière n'a pu être identifiée avec le solveur dans une limite de temps de 10 heures.

Notre approche de résolution par les métaheuristiques repose sur une stratégie de décomposition où les sous-problèmes associés aux rondes sont considérés séquentiellement. L'ensemble des solutions qu'il est possible d'obtenir en ne modifiant qu'une partie de la solution, en l'occurence les affectations dans une ronde donnée, définit une structure de voisinage. Lorsque la recherche s'arrête dans un optimum local relativement à une structure de voisinage, elle est relancée dans une autre structure de voisinage (nous considérons un autre sous-problème). En ce sens, l'approche de résolution peut être vue comme une recherche à voisinage variable, et le fait de changer de structure de voisinage permet de passer d'un sous-problème à un autre. Toutefois, l'ensemble des structures de voisinage utilisés peut être adapté en fonction des résultats obtenus (nous pouvons ne pas considérer les sous-problèmes associés aux rondes pour lesquelles des solutions optimales ont été obtenues).

Pour résoudre le sous-problème associé à une ronde, nous avons adapté les méthodes de résolution développées dans le cadre de notre première étude. Ainsi, nous avons proposé des généralisations de la RT et de la RVV permettant de tenir compte des différentes contraintes additionnelles. Dans le cas de la RTVS, nous avons en plus modifié la stratégie d'exploration du voisinage ce qui a permis d'améliorer significativement les performances de cette méthode.

De même, pour chacune des méthodes utilisant la RT, la RVV et la RTVS, nous avons considéré plusieurs variantes de la procédure de résolution, et nous avons analysé leurs

comportements en fonction de la nature et de la complexité des instances dans le but d'identifier la variante la plus performante.

Les résultats des expérimentations numériques témoignent de la pertinence de notre approche de résolution. Toutes les variantes considérées sont efficaces. Dans certains cas, nous retrouvons les résultats «optimaux» (dans le sens où les solutions générées satisfont à toutes les contraintes du problème), et dans d'autres, nous sommes très proches des valeurs optimales de la relaxation linéaire. En tout état de cause, l'obtention de solutions de bonne qualité est déjà très satisfaisante dans la mesure où on ne peut obtenir aucun résultat avec CPLEX dans une limite de temps raisonnable. Toutefois, les résultats des comparaisons révèlent la supériorité de la variante utilisant la RTVS. Elle se distingue aussi bien au niveau de la qualité des solutions générées qu'au niveau du temps de résolution.

Notre contribution se situe également sur un plan théorique bien qu'elle soit très modeste. En effet, nous avons été amenés par la nature du problème à associer aux différentes règles d'affectation des poids pour refléter leur importance relative. Plutôt que de fixer les valeurs de ces poids arbitrairement, nous avons développé un algorithme permettant d'obtenir des poids de plus petite valeur que ceux générés par deux algorithmes proposés dans la littérature. Cela nous a alors permis de calculer les poids pour les différents modèles que nous avons développés. Nous pensons toutefois que notre algorithme pourra être utile dans d'autres contextes en autant que l'ordre de priorité des objectifs soit prédéterminé.

Notre dernière contribution revêt un caractère opérationnel. En effet, le second problème abordé dans cette thèse étant réaliste et d'importance pratique, nous avons pu tester et valider nos méthodes de résolution en temps réel. Ainsi, lors des trois dernières éditions du concours *John Molson*, nous avons collaboré avec les organisateurs de ce concours pour établir les affectations des juges avec les méthodes que nous avons développées. Ces expériences nous ont permis d'une part, de faire évoluer notre modèle pour intégrer de plus en plus de contraintes réalistes, et d'autre part, d'ajuster nos approches de résolution à cette évolution du contexte réel. Mentionnons enfin que l'une des méthodes de résolution a été intégrée dans un logiciel interactif permettant aux or-

ganisateurs de générer les affectations eux mêmes.

Avec les deux problèmes étudiés dans cette thèse, nous avons constaté que les méthodes de résolution que nous avons développées présentent plusieurs avantages :

- leur efficacité : elles produisent des solutions d'excellente qualité ;
- leur aptitude à résoudre des problèmes de grande taille en des temps très raisonnables ;
- leur flexibilité et leur adaptabilité : les différentes généralisations présentées au chapitre 4 fournissent une bonne illustration de la manière dont les modèles et les méthodes de résolution peuvent être modifiés pour tenir compte d'autres contraintes.

Une suite logique à nos travaux serait donc d'adapter ces méthodes de résolution à d'autres contextes. Il serait intéressant de considérer certains problèmes présentés au chapitre 2. Nous pensons en particulier au problème d'affectation des arbitres aux tournois sportifs et celui d'affectation des articles aux évaluateurs. Il nous semble également qu'il serait intéressant d'utiliser l'approche de décomposition pour des problèmes d'affectation sur un horizon comportant plusieurs périodes.

D'autres avenues de recherche s'ouvrent également pour le second problème étudié dans cette thèse. Un premier axe de recherche consiste à développer des algorithmes qui adaptent la solution générée aux changements survenant dans les données. En effet, les organisateurs du concours sont parfois confrontés au problème de désistement de certains juges après que les affectations soient déterminées. Quoiqu'elles surviennent occasionnellement, ces situations sont embarassantes d'autant plus qu'elles surviennent souvent à la dernière minute. Les organisateurs se voient alors contraints à modifier rapidement les affectations et d'en générer des nouvelles, mais il est important qu'il y ait le moins de changements possible dans les affectations originales. Pour tenir compte de cet aspect du problème, l'approche envisagée serait de résoudre un nouveau problème d'affectation dont la solution initiale (complète ou partielle) est la solution originale, et d'introduire un coût supplémentaire de «modification» servant à pénaliser les affectations qui diffèrent de celles que l'on retrouve dans la solution initiale.

Une autre perspective de recherche concerne l'ajout de nouvelles contraintes au problème. En effet, les organisateurs du concours *John Molson* nous ont récemment fait

part d'autres contraintes qu'ils souhaiteraient intégrer. Elles concernent essentiellement l'adaptation de la composition du jury en fonction du nombre de juges qui le composent, des restrictions sur le nombre de matchs auxquels les juges *nouveaux* peuvent être affectés, et des restrictions sur la catégorie des juges affectés à chaque match de la dernière ronde en fonction des scores obtenus par les équipes aux matchs des rondes précédentes. Enfin, la programmation par contraintes a été largement utilisée dans le cadre de problèmes d'affectation complexes et s'est révélée très efficace. Une avenue prometteuse serait donc d'explorer les possibilités de résoudre les problèmes étudiés dans cette thèse avec cette approche.

BIBLIOGRAPHIE

[1] W. P. Adams et H. D. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32:1274–1290, 1986.

[2] R. K. Ahuja, T. L. Magnanti et J. B. Orlin. *Network flows : theory, algorithms and applications*. Englewood Cliffs, NJ :Prentice Hall, Inc, 1993.

[3] M. M. Amini et M. Racer. A hybrid heuristic for the generalized assignment problem. *European Journal of Operational Research*, 87(2):343–348, 1995.

[4] C. Archetti, A. Hertz et M. G. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13:49–76, 2007.

[5] B. M. Baker et C. Benn. Assigning pupils to tutor groups in a comprehensive school. *Journal of Operational Research Society*, 52:623–629, 2001.

[6] K. R. Baker et S. G. Powell. Methods for assigning students to groups : a study of alternative objective functions. *Journal of Operational Research Society*, 53: 397–404, 2002.

[7] E. Balas et M. J. Saltzman. An algorithm for the three-index assignment problem. *Operations Research*, 39(1):150–161, 1991.

[8] M. S. Bazaraa et H. D. Sherali. New approaches for solving the quadratic assignment problem. *Operations Research Verfahren*, 32:29–46, 1979.

[9] M. S. Bazaraa et H. D. Sherali. Bender's partitioning scheme applied to a new formulation of the quadratic assignment problem. *Naval Research Logistics Quarterly*, 27:29–41, 1980.

[10] S. Benferhat et J. Lang. Conference paper assignment. *International Journal of Intelligent Systems*, 16:1183–1192, 2001.

[11] J. Bhadury, E. J. Mighty et H. Damar. Maximizing workforce diversity in project teams : a network flow approach. *Omega*, 28:143–153, 2000.

[12] R. .E Burkard. Locations with spatial interactions : the quadratic assignment problem. Dans *Discrete Location Theory,*, pages 387–437. P.B.Mirchandani and R.L.Francis, eds, Wiley, New York, 1991.

[13] R. E. Burkard et T. Bonniger. A heuristic for quadratic boolean programs with applications to quadratic assignment problem. *European Journal of Operational Research*, 13:374–386, 1983.

[14] R. E. Burkard et E. Cela. Quadratic and three-dimensional assignments. Dans *Annotated bibliographies in combinatorial optimization*, pages 373–391. M.Dell'Amico, F.Maffioli, S.Martello(Eds.), 1997.

[15] R. E. Burkard et F. Rendl. A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 17:169–174, 1984.

[16] S. Butenko, J. Gil-Lafuente et P. Pardalos. *Economics, Management and Optimization in Sports*. Springer, Berlin, 2004.

[17] D. G. Cattryse et L. N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60:260–272, 1992.

[18] D. G. Cattrysse, M. Salomon et L. N. Van Wassenhove. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, 72(1):167–174, 1994.

[19] E. Cella. *The quadratic assignment problem : theory and algorithms*. Kluwer Academic Publishers, Boston, 1997.

[20] N. Christofides et M. Gerrard. Special cases of the quadratic assignment problem. Rapport technique, Management Science Research Report 391, Carnegie Mellon University, April 1976.

[21] P.C. Chu et J.E. Beasley. A genetic algorithm for the generalized assignment problem. *Computers and Operations Research*, 24(1):17–23, 1997.

[22] M. Dell'Amico et S. Martello. Linear assignment. Dans *Annotated bibliographies in combinatorial optimization*, pages 355–371. M.Dell'Amico, F.Maffioli, S.Martello(Eds.), 1997.

[23] M. Dell'Amico et P. Toth. Algorithms and codes for dense assignment problems : the state of the art. *Discrete Applied Mathematics*, 100:17–48, 2000.

[24] J. Desrosiers, N. Mladenovic et D. Villeneuve. Design of balanced MBA student teams, 2002. Les Cahiers du GERAD, G-2002-49.

[25] J.H. Dinitz et D.R. Stinson. On assigning referees to tournament schedules. *Bulletin of the Institute of Combinatorics and its Applications*, 44:22–28, 2005.

[26] J. Dréo, A. Pétrowski, P. Siarry et E. Taillard. *Métaheuristiques pour l'optimisation difficile*. Eyrolles, Paris, 2003.

[27] A.R. Duarte, C.C. Ribeiro et S. Urrutia. Referee assignment in sports tournaments. Dans *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling PATAT*, pages 394–397, Brno, Czech Republic, 2006.

[28] A.R. Duarte, C.C. Ribeiro et S. Urrutia. A hybrid ils heuristic to the referee assignment problem with an embedded MIP strategy. Dans *Lecture Notes in Computer Science 4771*, pages 82–95. Springer, 2007.

[29] A.R. Duarte, C.C. Ribeiro, S. Urrutia et E.H. Haeusler. Referee assignment in sports leagues. Dans *Lecture Notes in Computer Science 3867*, pages 158–173. Springer, 2007.

[30] S.T. Dumais et J. Nielsen. Automating the assignment of submitted manuscripts to reviewers. Dans *Research and Development in Information Retrieval*, pages 233–244, 1992.

[31] M. Guignard et M. Rosenwein. An improved dual-based algorithm for the generalized assignment problem. *Operations Research*, 37(4):658–663, 1989.

[32] J. R. Evans. A microcomputer-based decision support system for scheduling umpires in the american baseball league. *Interfaces*, 18(6):42–51, 1988.

[33] J. R. Evans, J. E. Hebert et R. F. Deckro. Play ball - the scheduling of sports officials. *Perspectives in Computing*, 4(1):18–29, 1984.

[34] J. A. Ferland et D. Costa. Heuristic search methods for combinatorial programming problems. Rapport technique 1193, département d'informatique et de recherche opérationnelle, université de Montréal, 2001.

[35] M. L. Fisher, R. Jaikumar et L. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science*, 32(9):1095–1103, 1986.

[36] C. Fleurent et J. A. Ferland. Genetic hybrids for the quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 16:173–187, 1994.

[37] L. R. Ford et D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1966.

[38] L. R. Francis et J. A. White. *Facility layout and location*. Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[39] A. M. Frieze et J. Yadegar. An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice. *Journal of the Operational Research Society*, 32(11):989–995, 1981.

[40] L. Gambardella, E. Taillard et M.Dorigo. Ant colonies for QAP. Rapport technique, Technical report 97-4, IDSIA, Lugano, Switzerland, 1997.

[41] R. S. Garfinkel et M. R. Rao. The bottleneck transportation problem. *Naval Research Logistics Quarterly*, 18:465–472, 1971.

[42] M. Gendreau, A. Hertz et G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40:1276–1290, 1994.

[43] J. Gil-Lafuente. The best systems to appoint referees. Dans *Economics, Management and Optimization in Sports*, pages 101–120. Springer, Berlin, 2004.

[44] K. C. Gilbert et R. B. Hofstra. Multidimensional assignment problems. *Decision Sciences*, 19(2):306–321, 1988.

[45] P. .C Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *J.SIAM*, 10:305–313, 1962.

[46] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13:533–549, 1986.

[47] F. Glover. Tabu search-part I. *ORSA Journal on Computing*, 1:190–206, 1989.

[48] F. Glover et M. Laguna. *Tabu Search.* Kluwer Academic Publishers, Boston, 1997.

[49] F. Glover, M. Laguna et R. Marti. Principles of tabu search. Dans *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, 2007.

[50] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Boston, 1989.

[51] J. Goldsmith et R. H. Sloan. The AI conference paper assignment problem. À paraître dans Proc. AAAI 2007 Workshop on Preference Handling for Artificial Intelligence.

[52] O. Gross. The bottleneck assignment problem, 1959. The Rand Corporation, Paper P-1630. Présenté au the RAND Symposium on Mathematical Programming (Linear Programming and Extensions), 16-20 Mars 1959.

[53] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial optimization, 1986. Présenté au Congress on Numerical Methods in Combinatorial Optimization, Capri (Italie).

[54] P. Hansen et N. Mladenovic. Recherche à voisinage variable, 2000. Les Cahiers du GERAD, G-2000-08.

[55] P. Hansen et N. Mladenovic. Variable neighborhood search : principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[56] P. Hansen et N. Mladenovic. A tutorial on variable neighborhood search, 2003. Les Cahiers du GERAD, G-2003-46.

[57] D. Hartvigsen, J. C. Wei et R. Czuchlewski. The conference paper-reviewer assignment problem. *Decision Sciences*, 30(3):865–876, 1999.

[58] A. V. Hill, J. D. Naumann et N. L. Chervany. Scat and spat : large-scale computer based optimization systems for the personnel assignment problem. *Decision Sciences*, 14:207–220, 1983.

[59] R. Jonker et A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340, 1987.

[60] K. Jornsten et M. Nasberg. A new lagrangean relaxation approach to the generalized assignment problem. *European Journal of Operational Research*, 27: 313–323, 1986.

[61] K. Jornsten et P. Varbrand. Two algorithms for the generalized assignment problem. Working paper, LiTH-Mat-R-87-02, Linkoping Institute of Technology, Sweden, 1987.

[62] T. C. Koopmans et M. J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

[63] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Logistics Quarterly*, 2(1):83–97, 1955.

[64] M. Laguna, J. P. Kelly, J. L. Gonzalez-Velarde et F. Glover. Tabu search for the multilevel generalized assignment problem. *European Journal of Operational Research*, 82(1):176–189, 1995.

[65] M. Laguna, R. Marti et V. Campos. Intensification and diversification with elite tabu search solutions for the LOP. *Computers and Operations Research*, 26:1217–1230, 1999.

[66] A. Lamghari et J. A. Ferland. Metaheuristic methods based on tabu search for assigning judges to competitions. *À paraître dans Annals of Operations Research. Special Issue CI-Sched.*

[67] A. Lamghari et J. A. Ferland. Heuristic techniques to assign judges in competitions. Dans *V. Prahlad, W. W. Tan, and A. P. Loh (eds.). Proceeding of the Third International Conference on Computational Intelligence, Robotics and Autonomous System (CIRAS)*, Singapore, 2005.

[68] A. Lamghari et J. A. Ferland. Structured neighborhood tabu search for assigning judges to competitions. Dans *D. Srinivasan (ed.). Proceedings of IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)*, pages 238–245, Honolulu, 2007.

[69] E. L. Lawler. *Combinatorial optimization : networks and matroids*. Holt, Rinehart and Winston, New York, 1976.

[70] S. M. Lee et M. J. Schniederjans. A multicriteria assignment problem : a goal programming approach. *Interfaces*, 13:75–81, 1983.

[71] Y. Li, P. M. Pardalos et M. G. C. Resende. A greedy randomized adaptive search procedure for quadratic assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 16:237–261, 1994.

[72] T. T. Liang et T. J. Thompson. A large-scale personnel assignment model for the navy. *Decision Sciences*, 18:234–249, 1987.

[73] N. Lorena et M. G. Narciso. Relaxation heuristics for a generalized assignment problem. *European Journal of Operational Research*, 91:600–610, 1996.

[74] H. L. Lourenco et D. Serra. Adaptive search heuristics for the generalized assignment problem. *MathWare and Soft Computing*, 9(2,3):209–234, 1995.

[75] D. G. Luenberger. *Linear and Nonlinear Programming, 2^{nd} edition*. Addison-Wesley, Massachusetts, 1984.

[76] D. Magos. Tabu search for the planar three-index assignment problem. *Journal of Global Optimization*, 8(1):35–48, 1996.

[77] D. Magos et P. Miliotos. An algorithm for the planar three-index assignment problem. *European Journal of Operational Research*, 77(1):141–153, 1994.

[78] S. Martello et P. Toth. An algorithm for the generalized assignment problem. *Operational Research*, 81(2,3):589–603, 1981.

[79] J. B. Mazzola et A. W. Neebe. An algorithm for the bottleneck generalized assignment problem. *Computers and Operations Research*, 20(4):355–362, 1993.

[80] J. Mingers et F. A. O'brien. Creating students groups with similar characteristics : a heuristic approach. *Omega*, 23:313–332, 1995.

[81] N. Mladenovic et P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.

[82] N. Mladenovic, M. Labbé et P. Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42:48–64, 2003.

[83] M. G. Narciso et N. Lorena. Lagrangean-surrogate relaxation for generalized assignment problems. *European Journal of Operational Research*, 114:165–177, 1999.

[84] R. M. Nauss. Solving the generalized assignment problem : An optimizing and heuristic approach. *INFORMS Journal on Computing*, 15(3):249–266, 2003.

[85] I. H. Osman. Heuristics for the generalized assignment problem : simulated annealing and tabu search approaches. *OR Spectrum*, 17(4):211–225, 1995.

[86] P. M. Pardalos, K. A. Murthy et T. P. Harrison. A computational comparison of local search heuristics for solving quadratic assignment problem. *Informatica*, 4 (12):172–187, 1993.

[87] P. M. Pardalos et L. S. Pitsoulis. *Nonlinear Assignment Problems : Algorithms and Applications*. Kluwer Academic Publishers, 2000.

[88] A. Pepin, G. Desaulniers, A. Hertz et D. Huisman. Comparison of heuristic approaches for the multiple depot vehicle scheduling problem, 2006. Les Cahiers du GERAD, G-2006-65.

[89] U. Pferschy. Solution methods and computational investigations for the linear bottleneck assignment problem. *Computing*, 59(3):237–258, 1997.

[90] W. P. Pierskalla. The multidimensional assignment problem. *Operations Research*, 16(2):422–431, 1968.

[91] L. G. Proll. A simple method of assigning projects to students. *Operational Research Quarterly*, 23:195–202, 1972.

[92] R Development Core Team. *R : A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

[93] A. Ravindran et V. Ramaswami. On the bottleneck assignment problem. *Journal of Optimization Theory and Applications*, 21(4):451–458, 1977.

[94] G. R. Reeves et E. P. Hickman. Assigning MBA students to field study project teams : a multicriteria approach. *Interfaces*, 22:52–58, 1992.

[95] F. Rendl. Ranking scalar products to improve bounds for the quadratic assignment problem. *European Journal of Operational Research*, 20:363–372, 1985.

[96] S. Sahni et T. Gonzalez. P-complete approximation problems. *Journal of the Association of Computing Machinery*, 23:555–565, 1976.

[97] S. E. Sampson. Optimization of volunteer labor assignments. *Journal of Operations Management*, 24:363–377, 2006.

[98] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations research*, 45(6):831–841, 1997.

[99] A. Schirrer, K. F. Doerner et R. F. Hartl. Reviewer assignment for scientific articles using memetic algorithms. Dans *Metaheuristics*, pages 113–134. Springer, New York, 2007.

[100] H. D. Sherali. Equivalent weights for lexicographic multi-objective programs : Characterizations and computations. *European Journal of Operational Research*, 11:367–379, 1982.

[101] W. Shish et J. A. Sullivan. Dynamic course scheduling for college faculty via zero-one programming. *Decision Sciences*, 8:711–721, 1977.

[102] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA Journal of Computing*, 2(1):33–45, 1990.

[103] P. Soriano et M. Gendreau. Fondements et applications des méthodes de recherche avec tabous. Rapport technique 908, département d'informatique et de recherche opérationnelle, université de Montréal, 1994.

[104] F. S. R. Spieksma. Multi index assignment problems : Complexity, approximation and extensions. Dans *Nonlinear Assignment Problems : Algorithms and Applications*, pages 1–12. Kluwer Academic Publishers, 2000.

[105] R. E. Steuer. *Multiple criteria optimization : theory, computation and application.* Robert E. Krieger Publishing Company, 1989.

[106] Y. Sun, J. Ma, Z. Fan et J. Wang. A hybrid knowledge and model approach for reviewer assignment. *Expert Systems with Applications*, 34(2):817–824, 2008.

[107] G. Syswerda. Uniform crossover in genetic algorithms. Dans *J.D. Schaffer (ed.). Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann Publishers*, pages 2–9, San Mateo, California, 1989.

[108] E. Taillard. Robust tabu search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[109] D. M. Tate et A. E. Smith. A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, 22:73–83, 1995.

[110] R. R. Weitz et M. T. Jelassi. Assigning students to groups : a multi-criteria decision support system approach. *Decision Sciences*, 23:746–757, 1992.

[111] R. R. Weitz et S. Lakshminarayanan. An empirical comparison of heuristic methods and graph theoritic methods for creating maximally diverse groups. *Omega*, 25(4):473–482, 1997.

[112] R. R. Weitz et S. Lakshminarayanan. An empirical comparison of heuristic methods for creating maximally diverse groups. *Journal of the Operational Research Society*, 49:635–646, 1998.

[113] M. R. Wilhelm et T. L. Ward. Solving quadratic assignment problem by simulated annealing. *IEEE Transactions*, 19(1):107–119, 1987.

[114] M. B. Wright. Scheduling english cricket umpires. *Journal of the Operational Research Society*, 42(6):447–452, 1991.

[115] M. Yagiura, T. Ibaraki et F. Glover. An ejection chain approach for the generalized assignment problem. *INFORMS Journal on Computing*, 16(2):133–151, 2004.

[116] M. Yagiura, T. Ibaraki et F. Glover. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, 169(2):548–569, 2006.

[117] M. Yagiura, T. Yamaguchi et T. Ibaraki. A variable depth search algorithm for the generalized assignment problem. Dans *Metaheuristics : advances and trends in local search paradigms for optimization*, pages 459–471. Kluwer, Boston, 1999.

[118] M. Yavuz, U. H. Inan et A. Figlali. Fair referee assignments for professional football leagues. *Computers and Operations Research*, 35:2937–2951, 2008.