Université de Montréal

Résolution du problème d'ordonnancement des activités avec contraintes de ressources et sa généralisation

par

Khaled Moumene

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

Philosophiae Doctor (Ph.D.)

en informatique

Septembre, 2006

Université de Montréal

Cette thèse est intitulée :

Résolution du problème d'ordonnancement des activités avec contraintes de ressources et sa généralisation

présentée par :

Khaled Moumene

a été évaluée par un jury composé des personnes suivantes

Jean-Yves Potvin	président du jury
Jacques A. Ferland	directeur de recherche
Michel Gendreau	membre du jury
Fayez F. Boctor	examinateur externe

Thèse acceptée le : 22 décembre 2006

Sommaire

Le problème d'ordonnancement des activités sous contraintes de ressources (communément noté $RCPSP$) consiste à planifier un ensemble d'activités, consommant des ressources disponibles en quantités limitées entre lesquelles il existe des contraintes de précédence qui associent à chaque activité un ensemble d'activités qui doivent la précéder. L'objectif est de construire un planning minimisant la durée totale d'exécution de toutes les activités.

Le problème est NP-Difficile. Comme c'est le cas pour la majorité des problèmes de ce niveau de complexité, les méthodes heuristiques sont généralement les plus efficaces pour le résoudre alors que les méthodes exactes ne sont efficaces que pour des instances de petite taille.

Dans cette thèse, nous introduisons une nouvelle représentation nommée *séquence d'ensembles d'activités* pour la résolution du $RCPSP$ et sa généralisation. Les heuristiques les plus efficaces pour le $RCPSP$ utilisent la représentation en séquence d'activités et la méthode (de décodage) Serial SGS pour construire le planning réalisable (solution) correspondant. Cette représentation a démontré son efficacité dans plusieurs méthodes de résolution notamment grâce au fait qu'elle est rapidement décodable. En effet, sa forme en liste permet l'utilisation d'une variété d'opérateurs. De plus, l'ensemble de tous les plannings engendrés par la représentation contient toujours un planning optimal.

Par ailleurs, les séquences d'activités engendrent un espace de recherche généralement beaucoup plus grand que celui des plannings puisqu'un même planning peut correspondre à plusieurs représentations différentes de ce type. Nous montrons comment les séquences d'ensembles d'activités peuvent réduire considérablement cet espace et comment y effectuer des déplacements plus efficaces. Nous montrons également que notre nouvelle représentation n'exclut jamais le planning optimal et possède de nombreuses propriétés intéressantes. Des calculs expérimentaux montrent comment, grâce à notre nouvelle représentation, les performances de deux des meilleures heuristiques proposées pour le $RCPSP$ ont été améliorées.

En plus des contraintes du $RCPSP$, la majorité des problèmes de planification réels nécessitent la prise en considération d'une variété de contraintes supplémentaires pour lesquelles la nouvelle représentation peut également s'avérer efficace. Nous généralisons l'utilisation des séquences d'activités et des séquences d'ensembles d'activités pour une très grande catégorie de problèmes de planification incluant le $RCPSP$. La généralisation

des deux représentations est nécessaire car elles sont étroitement liées. Nous montrons comment une séquence d'activités et une séquence d'ensembles d'activités peuvent être décodées en un planning réalisable du problème généralisé et comment, sous certaines conditions, elles n'excluent jamais le planning optimal. Nous montrons également comment les séquences d'ensembles d'activités peuvent, dans ce cas aussi, réduire l'espace de recherche induit par les séquences d'activités.

Nos résultats permettent de concevoir une nouvelle catégorie de méthodes de résolution utilisant notre nouvelle représentation pour améliorer les performances de celles dédiées au $RCPSP$ ou pour la résolution efficace de la généralisation proposée. Ils permettent aussi la réutilisation de la majorité des méthodes de résolution du ($RCSPSP$), qui sont basées sur les séquences d'activités, pour la généralisation du problème.

Mots clés : Problème d'ordonnancement des activités avec contraintes de ressources, RCPSP, Problème de planification, Séquence d'activités, Séquence d'ensembles d'activités, Heuristiques et Métaheuristiques.

Abstract

In a Resource-Constrained Project Scheduling Problem ($RCPSP$), we consider a project including a set of activities to be scheduled under resource and precedence constraints. On the one hand, the precedence constraints are induced by technological requirements to impose that any activity must be scheduled after the completion of all its immediate predecessors. On the other hand, each activity requires a quantity of resources during each period of its execution. The resource constraints are specified to limit the consumption of each resource used to the quantity available in each period. The $RCPSP$ is to determine a starting period for each activity in order to minimize the total duration (makespan) of the project while satisfying the precedence and resource constraints.

Since the problem is known to be NP-Hard, heuristic procedures are generally used to deal with large instances while exact methods are mainly used to solve small instances.

In this thesis, we introduce a new representation named Activity Set List for the solutions of the $RCPSP$ and its generalization. The most efficient heuristics for the $RCPSP$ use the Activity List representation and the serial SGS method to construct the corresponding schedule (solution). This representation has shown its efficiency for the problem in many solution procedures in the literature. Indeed, it is quickly decoded into a feasible schedule and its list form allows the application of various operators. Also, the set of all schedules induced by this representation always contains an optimal schedule.

Furthermore, the Activity List may induce a search space of representations much larger then the space of schedules because the same schedule can correspond to many different Activity List representations. We show how the Activity Set List representation can considerably reduce the search space and how to move more efficiently through it. We also show that this new representation never excludes the optimal schedule and that it has many interesting properties. Computational experiments indicate the efficiency of the Activity Set List. The performance of two of the best heuristics for the problem has been improved by introducing the new representation.

In most of the real life scheduling problems, one has to account for several other constraints in addition to the precedence and resource constraints considered in the $RCPSP$. We define a wide category of scheduling problems that includes various additional constraints for which the Activity List and the Activity Set List representations can be used as efficiently. We mainly show how these representations can be decoded to feasible schedules and how, under some conditions, these representations never exclude

the optimal schedule. We also indicate how to use the Activity Set List to reduce the Activity Lists search space for our problem generalization.

Our results allow the construction of a new category of efficient solution procedures based on the Activity Set List representation for the $RCPSP$ and its generalization. Furthermore, most of the proposed solution procedures for $RCPSP$ based on the Activity List representation can be extended by simply replacing the classic decoding procedure used for the $RCPSP$ by the generalized version introduced here.

Keywords : Resource-Constrained Project Scheduling Problem, RCPSP, Scheduling problem, Activity List, Activity Set List, Heuristics and Meta-heuristics.

Remerciements

Je remercie Monsieur Jacques A. Ferland, mon directeur de recherche, d'avoir si bien dirigé ce travail et dont l'aide et la disponibilité m'ont été très précieuses. J'ai énormément apprécié les longues discussions fertiles que nous avons eues ainsi que sa grande flexibilité de travail.

Mes remerciements vont aussi à Monsieur Jean-Yves Potvin, professeur à l'Université de Montréal, de me faire l'honneur d'évaluer cette thèse et de présider le jury de ma soutenance. Je remercie aussi Monsieur Fayez F. Boctor, professeur à l'Université Laval, d'avoir accepté d'évaluer ce travail et d'agir en tant que membre du jury. J'adresse également mes remerciements à Monsieur Michel Gendreau, professeur à l'Université de Montréal, pour l'évaluation de ce travail et de faire partie du membre du jury.

Je remercie mon épouse Nadjia, dont le soutien et les encouragements n'ont jamais cessé tout au long de ces longues années. Je la remercie aussi d'avoir créé un environnement très favorable au bon déroulement de ce travail. Je remercie mon fils Sami dont le sourire vaut bien plus que milles mots d'encouragements. Je les remercie de m'avoir concédé du temps qui leur appartenait. Mes remerciements vont aussi à ma mère, mon père ainsi que mes frères Adel et Amine pour leurs soutien et encouragements malgré la distance qui nous sépare.

Enfin, je remercie tous les professeurs qui m'ont formé ainsi que toutes les personnes qui ont participé de près ou de loin à l'aboutissement de ce travail.

Table des matières

Liste des tableaux

Table des figures

Introduction

La résolution du problème d'ordonnancement des activités sous contraintes de ressources $RCPSP$ a toujours suscité beaucoup d'intérêt dans la littérature puisqu'il modélise de nombreuses situations où il est question de planifier plusieurs activités, consommant une variété de ressources et soumises à des contraintes de précédence, de sorte à minimiser la durée totale de leur exécution. Vu la complexité du problème, les heuristiques constituent la majorité des méthodes proposées pour sa résolution.

La représentation des solutions (plannings) sous forme de séquence d'activités est utilisée par la plus grande majorité de ces heuristiques. Elle consiste en une liste ordonnée de toutes les activités dont le décodage en un planning réalisable est fait avec la méthode Serial SGS. Cette représentation est rapidement décodable, sa forme en liste permet l'utilisation d'une variété d'opérateurs et l'ensemble de tous les plannings engendrés par cette représentation contient toujours un planning optimal. Par ailleurs, plusieurs séquences d'activités différentes peuvent correspondre à un même planning. Ainsi, le domaine de recherche constitué par toutes les séquences d'activités possibles peut être beaucoup plus grand que celui des plannings. Ainsi, les méthodes se basant sur cette représentation peuvent parcourir plusieurs séquences d'activités inutilement.

Nous proposons une nouvelle représentation notée séquence d'ensembles d'activités généralisant la représentation en séquence d'activités. Nous introduisons un algorithme polynomial simple permettant de construire une séquence d'ensembles d'activités à partir d'une séquence d'activités et lui conférer la propriété de permutation permettant d'affirmer que cette séquence d'ensembles d'activités correspond à plusieurs séquences d'activités différentes dont le décodage induit le même planning. Ceci a pour effet de réduire considérablement le domaine de recherche pouvant être induit par les séquences d'activités. Aussi, la nouvelle représentation n'exclut jamais la solution optimale du problème. Nous montrons comment il est possible d'apporter de simples modifications aux méthodes heuristiques basées sur les séquences d'activités pour améliorer leurs performances grâce aux séquences d'ensembles d'activités. Nous illustrons cela en introduisant la nouvelle représentation dans deux des meilleures heuristiques proposées pour le problème ; la première est basée sur le recuit simulé et la seconde sur les algorithmes génétiques. Nos résultats numériques montrent clairement l'amélioration de leurs performances.

Généralement, les contraintes de précédence et celles sur les ressources que comporte le $RCPSP$ sont insuffisantes pour modéliser la plupart des problèmes de planification

réels. La prise en considération de contraintes supplémentaires s'avère souvent nécessaire. Étant donné que les méthodes heuristiques les plus efficaces pour le $RCPSP$ sont souvent basées sur les séquences d'activités, nous proposons l'utilisation de cette représentation ainsi que celle de séquence d'ensembles d'activités pour une très large généralisation du $RCPSP$. Nous introduisons des conditions simples et non restrictives définissant la généralisation et permettant l'utilisation des deux représentations. Aussi, nous définissons une extension de la méthode de décodage Serial SGS permettant d'associer une solution réalisable du problème généralisé à toute séquence d'activités ou séquence d'ensembles d'activités. Ainsi, pour résoudre le problème généralisé, il est possible de réutiliser la majorité des approches basées sur les séquences d'activités dédiées initialement au $RCPSP$, en y remplaçant uniquement la méthode Serial SGS par notre méthode de décodage généralisée. De plus, nous introduisons la notion d'extensibilité sur les plannings permettant de spécifier des conditions suffisantes assurant l'existence d'une séquence d'activités et d'une séquence d'ensembles d'activités engendrant le planning optimal du problème généralisé. Par conséquent, sous ces conditions, les méthodes de résolution qui utilisent l'une de ces représentations peuvent aboutir à la solution optimale. Il est même possible dans ce cas de résoudre à l'optimum de petites instances du problème généralisé en énumérant toutes les séquences d'ensembles d'activités puisque leur nombre est réduit. Enfin, nous montrons que l'utilisation des séquences d'ensembles d'activités dans les approches de résolution du problème généralisé peut améliorer leurs performances en réduisant considérablement l'espace de recherche induit par les séquences d'activités.

Au Chapitre 1, nous décrivons le problème $RCPSP$ ainsi que certaines de ses variantes puis nous y abordons différentes modélisations et méthodes de résolution proposées dans la littérature. Nous commençons le Chapitre 2 par une brève description de la représentation sous forme de séquences d'activités. Cette représentation est très souvent citée dans les procédures de résolution du $RCPSP$. Nous introduisons ensuite une nouvelle représentation sous forme de séquences d'ensembles d'activités. Aussi, nous spécifions une puissante propriété la caractérisant ainsi que quelques résultats et procédures s'y rattachant. Nous montrons aussi comment les séquences d'ensembles d'activités permettent, grâce à cette propriété, de réduire l'espace de recherche induit par les séquences d'activités. Nous montrons également qu'il existe toujours une séquence d'ensembles d'activités engendrant une solution optimale et comment, dans certaines situations, il possible de l'identifier.

À l'aide d'un exemple, nous illustrons les avantages d'un point de vue numérique

d'utiliser la notion de séquences d'ensembles d'activités. Pour ce faire, nous considérons les séquences d'activités voisines obtenues à partir d'une séquence d'activités spécifique en modifiant la position d'une de ses activités. Ainsi, en se référant à la séquence d'ensembles d'activités associée à la séquence d'activité spécifique, il est possible d'identifier les séquences d'activités voisines correspondant au même planning que celui correspondant à la séquence d'activités spécifique de départ. Une étude numérique plus poussée permet de vérifier que le pourcentage de ces séquences voisines correspondant au même planning est important dans plusieurs cas. Cet exemple montre clairement comment l'introduction de la nouvelle représentation dans les approches basées sur les séquences d'activités peut améliorer leurs performances. Enfin, nous abordons l'extension du décodage de la nouvelle représentation en mode backward. Nous montrons que pour ce mode, nos résultats restent valides.

Le Chapitre 3 est dédié à l'évaluation des performances des séquences d'ensembles d'activités lorsqu'elles sont introduites dans deux des meilleures procédures de résolution pour le $RCPSP$. La première, basée sur le recuit simulé, est celle de Bouleimen et Lecoq [16] et la seconde, basée sur les algorithmes génétiques, est celle de Hartmann [66]. Nos résultats numériques sont obtenus en appliquant ces procédures à toutes les instances de PSPLIB, une bibliothèque fréquemment utilisée dans la littérature, qui regroupe plusieurs problèmes tests. Plusieurs variantes des deux procédures sont implémentées puis comparées selon plusieurs critères de performance, notamment, le pourcentage de déviation par rapport à la solution obtenue par la méthode CPM (Critical Path Method) lorsque les contraintes de ressources sont relâchées, le temps de calcul et le nombre d'itérations ou de générations parcourues avant d'atteindre la meilleure solution. Nous comparons aussi ces procédures suivant le degré de complexité des instances utilisées.

Dans le Chapitre 4, nous définissons la généralisation que nous proposons pour le problème $RCPSP$. Nous introduisons la condition de souplesse permettant le décodage d'une séquence d'activités en un planning réalisable et cela selon les deux modes de planification, forward et backward. Nous montrons également comment, sous cette condition, il est possible de généraliser la méthode de décodage Serial SGS pour ce problème. Plusieurs exemples d'instances du problème satisfaisant la condition de souplesse sont donnés. Nous indiquons par la suite que cette condition n'est pas suffisante pour assurer l'existence d'une séquence d'activité induisant un planning optimal. Ceci nous amène à introduire des conditions supplémentaires assurant l'existence d'une telle séquence d'activités. Plusieurs exemples d'instances vérifiant ces conditions sont donnés. De nombreuses

extensions de techniques initialement développées pour le $RCPSP$ sont aussi abordées pour permettre la résolution du problème généralisé. Enfin, nous montrons comment la majorité des résultats sur les séquences d'ensembles d'activités obtenus dans le contexte du $RCPSP$ peuvent être étendus pour le problème généralisé, notamment, la réduction de l'espace de recherche induit par les séquences d'activités et l'existence d'une séquence d'ensembles d'activités engendrant un planning optimal pour le problème généralisé.

Chapitre 1

Description du problème

1.1 Introduction

La théorie de la planification et du séquencement est l'un des domaines de la recherche opérationnelle qui engendre, virtuellement, un nombre illimité de types de problèmes qu'on peut retrouver dans la réalité. Le problème d'ordonnancement des activités avec contraintes de ressources en fait partie. Il consiste en la planification d'un ensemble d'activités nécessaires à la réalisation d'un projet sous un ensemble de contraintes notamment : des ressources limitées et nécessaires à l'accomplissement de chacune des activités, existence de contraintes temporelles ou de précédence entre les activités.

Le problème trouve son application dans plusieurs domaines comme : l'ingénierie de la construction, le développement informatique, $\cdots$, etc. De plus, le problème est très riche dans le sens où il constitue une formulation généralisée de plusieurs problèmes classiques comme le *Job Shop, machine scheduling, ordonnancement*, $\cdots$, etc.

Le problème d'ordonnancement des activités avec contraintes de ressources consiste à planifier l'ensemble des activités d'un projet en leur affectant des périodes pendant lesquelles elles devront s'accomplir et, éventuellement, la "manière" avec laquelle chacune d'elles devra s'exécuter. Cette planification devra prendre en considération, principalement, deux types de contraintes : des contraintes sur les activités elles-mêmes (qui peuvent être relatives à chacune des activités prise indépendamment ou bien qui lient les activités entre elles) ainsi que des contraintes sur les ressources qui, globalement, garantissent la suffisance de ces dernières tout au long de l'exécution des activités.

En ce qui concerne les objectifs à optimiser, ceux-ci peuvent être nombreux et variés. Ils peuvent concerner, par exemple, la durée du projet ou bien son coût. Vu le nombre

important d'objectifs et de contraintes pouvant être pris en considération, nous pouvons constater qu'il y a un nombre extrêmement important de problèmes. Il s'agit donc bien d'une catégorie de problèmes renfermant plusieurs variantes.

1.2 Termes et appellations

Nous donnons ici une définition plus précise de certains termes se rapportant au problème et que nous utiliserons fréquemment par la suite.

1. *Activité* : Tâche élémentaire faisant partie de l'ensemble des tâches nécessaires à l'accomplissement du projet. L'exécution de toutes les activités signifie l'exécution du projet.
2. *Mode de réalisation d'une activité* : Il se peut qu'une activité puisse être exécutée de plusieurs façons. Chacune de ces façons peut mobiliser des ressources, engendrer des durées de réalisation ou des coûts différents. Ces différentes façons d'exécution sont les modes de réalisation de l'activité.
3. *Période élémentaire* : La période sur laquelle la planification se fait est subdivisée en périodes élémentaires de durées égales. Chacune des activités ne peut commencer (respectivement, se terminer) qu'au début (respectivement, à la fin) d'une période élémentaire. En fait, cette subdivision du temps de planification facilite également la modélisation du problème.
4. *Durée d'une activité* : Temps nécessaire pour exécuter une activité. Il est donné en terme de nombre de périodes élémentaires.
5. *Date* : Désigne un moment précis de la période de planification. Ce moment correspondra toujours au début d'une période élémentaire.
6. *Planning (ou calendrier d'exécution)* : Ce sont toutes les informations nécessaires pour caractériser une solution du problème. Par exemple, un planning peut être donné en spécifiant uniquement la date de début de chacune des activités du projet, si cette information est suffisante pour définir complètement la solution qu'il représente.
7. *Ressources renouvelables* : Ce sont des ressources dont la disponibilité est renouvelée à chaque période élémentaire.
8. *Ressources non renouvelables* : Ce sont des ressources dont la quantité est fixe au début de la première période élémentaire et dont la disponibilité est réduite suite

à sa consommation par les activités au fur et à mesure que le projet se déroule. La quantité de ces ressources à la fin du projet est égale à leur quantité au début du projet moins la somme de la consommation de ces ressources par toutes les activités au cours de toutes les périodes élémentaires.

9. *Contraintes de précédence* : Ce sont des contraintes définies sur les activités du projet. Elles exigent que pour un certain nombre de couples d'activités (i, j), l'activité j doit s'exécuter *après* l'activité i.

10. *Contraintes sur les ressources* : L'exécution de chacune des activités du projet peut nécessiter une ou plusieurs ressources en quantités bien précises. Ces contraintes garantissent globalement la suffisance de ces ressources pour l'accomplissement de toutes les activités.

11. *Contraintes temporelles* : Ce sont des contraintes définies sur les activités du projet. Elles exigent que pour un certain nombre de couples d'activités (i, j), il y ait une durée minimale (ou maximale) entre la date de début (ou de fin) de l'activité i et la date de début (ou de fin) de l'activité j. Mais dans tous les cas, nous considérons que i commence avant j.

1.3 Liste des paramètres

Le Tableau 1.1 résume les paramètres du problème que nous allons utiliser.

1.4 Définition du problème

1.4.1 Le problème $RCPSP$ de base

Comme déjà souligné, le problème possède plusieurs variantes. Nous présentons ici le problème d'ordonnancement des activités avec contraintes de ressources dit de *base* et couramment noté $RCPSP$ dans la littérature.

Le projet à planifier est représenté par un réseau $G = (V', E)$, où V' représente l'ensemble des activités et E correspond aux contraintes de précédence. Les activités sont numérotées de 1 à $n + 2$. Les activités 1 et $n + 2$ sont virtuelles (il existe, donc, réellement n activités). Les activités sont supposées être exécutées *sans interruption* et d_i sera la durée fixe de l'activité $i \in V'$ avec $d_1 = d_{n+2} = 0$. R^ρ sera l'ensemble des ressources renouvelables (il n'y a pas de ressources non renouvelables dans cette

Paramètres	Significations
n	Nombre d'activités.
V	Ensemble des activités ($V = \{2, \cdots, n+1\}$).
V'	Ensemble des activités auquel s'ajoutent les activités virtuelles 1 et $(n+2)$ ($V' = V \cup \{1, (n+2)\}$).
(i, j)	Définit une contrainte de précédence ou temporelle par rapport aux activités i et j.
E	Ensemble des couples (i, j) définissant les contraintes de précédence.
P_j	Ensemble des prédécesseurs *directs* de l'activité j (les activités devant se dérouler juste avant).
S_j	Ensemble des successeurs *directs* de l'activité j (les activités devant se dérouler juste après).
d_j	Temps nécessaire à l'exécution de l'activité j.
R^ρ	Ensemble des ressources renouvelables.
R_k^ρ	Nombre d'unités disponibles de la ressource renouvelable k .
r_{jk}^ρ	Nombre d'unités de la ressource renouvelable k consommées par l'activité j à chaque période élémentaire de son exécution.
μ_j	Ensemble des modes d'exécution de l'activité j .
d_j^m	Temps nécessaire pour exécuter l'activité j selon le mode m.
R^v	Ensemble des ressources non renouvelables.
R_k^v	Nombre d'unités disponibles de la ressource non renouvelable k.
r_{jkm}^ρ	Nombre d'unités de la ressource renouvelable k consommées par l'activité j lorsqu'elle est exécutée selon le mode m, à chaque période élémentaire de son exécution.
r_{jkm}^v	Nombre d'unités de la ressource non renouvelable k consommées par l'activité j lorsqu'elle est exécutée selon le mode m.
P	Un planning donné par toutes les informations suffisantes pour le définir.
ρ_T	Ensemble des plannings réalisables par rapport aux contraintes de précédence ou temporelles.
ρ_R	Ensemble des plannings réalisables par rapport aux contraintes sur les ressources.
$\rho = \rho_T \cap \rho_R$	Ensemble des plannings réalisables.
$\overline{d}$	Date limite de la fin du projet.
T	Ensemble des périodes élémentaires constituant la totalité de la période de référence considérée ($T = \{1, \cdots, \|T\|\}$).
t	Indice des périodes élémentaires ($t = 1, \cdots, \|T\|$).
$[t-1, t[$	Intervalle de temps correspondant à la période élémentaire t.
$r_k(P, t)$	Nombre d'unités de la ressource k consommées à la période élémentaire t en choisissant le planning P.
d_{ij}^{min}, d_{ij}^{max}	Min, max de la longueur de l'intervalle de temps séparant les dates de début des activités i et j .

TAB. 1.1 – Liste des paramètres

variante du problème) et r^{ρ}_{jk} sera le nombre d'unités de la ressource renouvelable k consommées par l'activité j à chaque période élémentaire de son exécution. Chaque ressource renouvelable k est disponible en quantité R^{ρ}_{k} à chaque période élémentaire. L'objectif est de *minimiser la durée totale du projet.*

1.4.2 Autres variantes du problème

Nous abordons ici certaines variantes du problème d'ordonnancement des activités avec contraintes de ressources. Ces variantes peuvent être distinguées selon les trois critères suivants [19] :

1. *Caractéristiques des ressources*
   - Le type de ressources considéré : renouvelables, non renouvelables ou les deux.
   - La quantité des ressources disponibles (dans certains cas, les quantités sont infinies).
   - Dépendance ou pas entre la quantité des ressources consommées, le temps d'exécution des activités (éventuellement, avec un mode donné) et le coût que cela engendre.
2. *Caractéristiques des activités*
   - Existence d'*un ou de plusieurs modes d'exécution* pour chaque activité.
   - Possibilité d'exécuter une activité selon *plusieurs modes adoptés successivement* tout au long du projet.
   - Le temps d'exécution d'une activité peut être *constant*, *aléatoire*, *fonction d'un mode d'exécution* ou d'un *coût.*
   - Une activité peut être *interrompue ou pas* pour être reprise plus tard dans le projet.
   - *Dépendance* ou pas entre les modes d'exécution des activités (en effet, il se peut que l'exécution d'une activité i en un mode m entraîne l'exécution d'une activité j avec un mode $\overline{m}$).
3. *Types d'objectifs*

   Ceux-ci sont variés et multiples. Le problème peut être mono-critère ou multicritère [59]. Voici quelques fonctions objectifs que nous pouvons trouver dans la littérature [53, 70] :
   - Minimiser la durée totale du projet.
   - Minimiser le coût du projet.

- Minimiser les ressources allouées au projet.
- Minimiser les retards sur le projet.
- Minimiser les écarts par rapport à une moyenne d'utilisation des ressources.
- Maximiser la valeur actuelle nette du projet.
- $\cdots$, etc.

Notation des variantes du problème

La manière avec laquelle le problème d'ordonnancement des activités avec contraintes de ressources est abordé dans la littérature est très liée à ses caractéristiques et sa formulation spécifique. Il est donc important de savoir de quelle variante du problème il s'agit avant de développer une stratégie de modélisation et de résolution.

Nous avons trouvé en [19] une assez bonne notation des variantes du problème selon les trois critères que nous avons mentionnés dans la section 1.4.2. Nous utiliserons cette notation pour caractériser ces variantes, aborder leurs modélisations ainsi que leurs méthodes de résolution. Cela dit, nous ne mentionnerons pas ici toutes les notations présentées en [19] qui sont beaucoup plus riches. Nous nous en tiendrons juste aux notations qui vont nous servir par la suite.

Globalement, la notation d'une variante du problème se fera selon trois champs comme suit :

$$(\ \textit{mode de réalisation}\ |\ \textit{caractéristiques des activités}\ |\ \textit{fonction objectif}\ )$$

Les valeurs que peuvent prendre chacun des trois champs sont listées dans le Tableau 1.2[1].

En l'absence de spécifications sur les ressources, nous considérerons que ces dernières sont renouvelables et qu'elles sont disponibles en quantités limitées. Notons que plusieurs valeurs du champ *caractéristiques des activités* peuvent être prises en même temps.

Le problème d'ordonnancement des activités avec contraintes de ressources de *base* $RCPSP$, défini plus haut, pourra donc être noté (PS | Prec | C_{max}).

1.4.3 Complexité du problème

Le problème $RCPSP$ est un problème d'optimisation combinatoire de complexité NP-dur [11]. À partir de 60 activités et 4 types de ressources, sa résolution par une

[1]Nous avons ajouté la notation *intr* pour les caractéristiques des activités. Elle ne figure pas dans [19].

Champs	Valeurs	Signification
Mode de réalisation	PS	Unique mode de réalisation pour chaque activité.
	MPS	Plusieurs modes de réalisation pour chaque activité.
Caractéristiques des activités	Prec	Les contraintes de précédence sont considérées.
	Temp	Les contraintes temporelles sont considérées.
	$d_j = sto$	Les durées d'exécution des activités sont aléatoires.
	$\overline{d}$	Il existe une date limite de fin du projet.
	intr	Une activité peut être interrompue pour être reprise plus tard.
Fonction objectif	C_{max}	Désigne la minimisation de la durée totale du projet.
	(Autres)	Si elle est autre que C_{max}, ce champ aura comme valeur l'expression exacte de la fonction objectif en question.

TAB. 1.2 – Variantes du problème : valeurs des champs

méthode exacte nécessite un temps de calcul assez important [1]. Pour avoir une idée sur la taille réelle de ce type de problèmes, il faut savoir qu'un projet informatique de taille moyenne comprend entre 100 et 1000 activités et nécessite entre 1 à 6 types de ressources [59]. Par ailleurs, notons que pour $|R^v| \geq 2$ et $\mu_j \geq 2$ pour tout $j \in V$, le problème (MPS | Prec | C_{max}) (la variante du problème avec plusieurs modes de réalisation) est NP-complet [85] et que le problème de faisabilité de (PS | Temp | C_{max}) est NP-complet également [70].

1.5 Modélisations et méthodes de résolution existantes

Plusieurs modélisations et méthodes de résolution existent dans la littérature. Une revue globale est faite en [19, 59, 69, 70, 90, 92]. Voici quelques principes d'approches de résolution que nous pouvons y retrouver :

- La programmation dynamique [59]
- Les approches par construction progressive [59, 69]
- Les méthodes d'énumération implicite et de séparation et évaluation [41, 59, 115]
- Les méthodes de recherche locale [1, 2, 6, 16, 66, 95, 118, 125]
- L'intelligence artificielle [59]
- Les réseaux de neurones [59]
- Les algorithmes génétiques [1, 2, 68, 83, 108, 139]
- Les heuristiques basées sur les règles de priorité [3, 14, 13, 38, 55, 59, 113, 134]
- Autres méthodes spécifiques [89]

La majorité des démarches de résolution ne reposent pas sur des modèles bien établis comme un graphe ou un programme mathématique pour solutionner le problème avec des méthodes appropriées à ces derniers. Il existe, par ailleurs, de tels modèles dans la littérature. La plupart d'entre eux sont formulés à l'aide d'un programme mathématique contenant des variables pouvant prendre des valeurs binaires qui sont généralement définies comme suit :

$$x_{it(m)} = \begin{cases} 1 & \text{Si l'activité } i \text{ se termine à la date } t \text{ (et se déroule selon le mode } m) \\ 0 & \text{Sinon} \end{cases}$$

Ce type de modèles présente l'avantage de la relative facilité de traduire les contraintes du problème en équations ou inéquations mathématiques. Ils ont également l'inconvénient d'être, dans la majorité des cas, difficiles à résoudre.

1.5.1 Modélisations mathématiques

Nous présentons, dans ce qui suit, un modèle générique pouvant servir à la modélisation de plusieurs variantes du problème. Ce dernier est inspiré de [106] qui contient une liste très variée de contraintes et de fonctions objectif.

Voici les variables décisionnelles considérées :

$$x_{itm} = \begin{cases} 1 & \text{Si l'activité } i \textit{ se termine à la fin de la période } t \text{ et se déroule selon} \\ & \text{le mode } m \\ 0 & \text{Sinon} \end{cases}$$

$$y_{itm} = \begin{cases} 1 & \text{Si l'activité } i, \textit{ qui peut être interrompue, se déroule durant la période } t \\ & \text{selon le mode } m \\ 0 & \text{Sinon} \end{cases}$$

$$s_{itm} = \begin{cases} 1 & \text{Si l'activité } i, \textit{ qui peut être interrompue, commence à la fin de la période } t \\ & \text{et se déroule selon le mode } m \\ 0 & \text{Sinon} \end{cases}$$

Voici une liste non exhaustive de contraintes pouvant constituer une des variantes du problème :

1. L'activité virtuelle 0 se termine à la période 0 et se déroule selon son unique mode de réalisation 1.

$$x_{001} = 1 \qquad (1)$$

2. Chaque activité doit être complétée et doit se dérouler selon un unique mode de réalisation.

$$\sum_{t \in T} \sum_{m \in \mu_i} x_{itm} = 1 \ , \forall i \in V \qquad (2)$$

3. Une activité qui peut être interrompue doit avoir une seule date de début. La date de reprise d'un activité après son interruption ne peut être considérée comme une date de début.

$$\sum_{t \in T} \sum_{m \in \mu_i} s_{itm} = 1 \ , \forall i \in V \text{ où } i \text{ peut être interrompue} \qquad (3)$$

4. Une activité i qui peut être interrompue ne peut être complétée sans qu'elle se soit déroulée sur un nombre de périodes égal à sa durée.

$$\sum_{t \in T} y_{itm} = \sum_{t \in T} d_i^m x_{itm} \ , \ \forall i \in V \text{ où } i \text{ peut être interrompue, } \forall m \in \mu_i \qquad (4)$$

5. Les contraintes de précédence doivent être respectées. Pour cela, nous considérons deux cas de figure selon que l'activité peut être interrompue ou non.

(a) Si l'activité peut être interrompue.

$$\sum_{t\in T}\sum_{m\in\mu_j} ts_{jtm} - \sum_{t\in T}\sum_{m\in\mu_i} tx_{itm} \geq 0, \qquad (5)$$

$\forall j \in V$ où j peut être interrompue, $\forall i \in P_j$

(b) Si l'activité ne peut être interrompue.

$$\sum_{t\in T}\sum_{m\in\mu_j} (t - d_j^m)x_{jtm} - \sum_{t\in T}\sum_{m\in\mu_i} tx_{itm} \geq 0, \qquad (6)$$

$\forall j \in V$ où j ne peut être interrompue, $\forall i \in P_j$

6. Les contraintes sur les ressources renouvelables doivent être respectées.

$$\sum_{i\in V_1}\sum_{m\in\mu_i}\sum_{t=u}^{Min(u+d_i^m-1,T)} r_{ikm}^{\rho}x_{itm} + \sum_{j\in V_2}\sum_{m\in\mu_j} r_{jkm}^{\rho}y_{ium} \leq R_k^{\rho}, \quad \forall k \in R^{\rho}, \forall u \in T \qquad (7)$$

Où :

$V_1 \subset V$ représente l'ensemble des activités qui ne peuvent être interrompues.

$V_2 \subset V$ représente l'ensemble des activités qui peuvent être interrompues.

7. Les contraintes sur les ressources non renouvelables doivent être respectées.

$$\sum_{i\in V}\sum_{t\in T}\sum_{m\in\mu_i} r_{ikm}^{v}d_i^m x_{itm} \leq R_k^{v}, \quad \forall k \in R^{v} \qquad (8)$$

La minimisation de la durée totale du projet peut s'exprimer comme suit :

$$Min \sum_{t\in T} x_{(n+2)t1} \qquad (9)$$

Ainsi, voici comment certaines variantes particulières du problème peuvent être formulées :

1. (PS | Prec | C_{max}) ou $RCPSP$: En considérant les contraintes (1), (2), (6) et (7) avec l'objectif (9). Bien entendu, *toutes* les activités possèdent un unique mode de réalisation et doivent s'exécuter sans interruption.
2. (MPS | Prec | C_{max}) : En considérant les contraintes (1), (2), (6), (7) et (8) avec l'objectif (9). Dans ce cas, *toutes* les activités doivent s'exécuter sans interruption.
3. (PS | Prec, intr | C_{max}) : En considérant les contraintes (1), (2), (3), (4), (5) et (7) avec l'objectif (9). Dans ce cas, *toutes* les activités possèdent un unique mode de réalisation et peuvent s'exécuter avec interruption.

1.5.2 Méthodes exactes

Vu la complexité du problème (voir la section 1.4.3) et la taille importante de ce dernier dans la plupart des cas réels, les méthodes de résolution exactes sont de plus en plus rares dans la littérature. Presque la totalité des travaux récents sur le problème portent sur sa résolution par des méthodes approchées. Nous mentionnons ici le principe global de la plupart des méthodes exactes proposées pour certaines variantes du problème.

Le problème (PS | Prec | C_{max}) ou $RCPSP$

1. *Les procédures par séparation et évaluation*

   Une variété de procédures par séparation et évaluation ont été proposées pour le problème. La majorité d'entre elles utilisent des *planifications partielles* (planification d'une partie des activités) du projet qui correspondent aux noeuds d'un arbre d'énumération. La procédure de séparation consiste, en général, en l'extension des planifications partielles de différentes façons. Des règles de dominance, des bornes inférieures et des règles de sélection immédiates permettent de réduire le nombre d'alternatives. Voici un exemple d'une méthode de séparation et évaluation proposée en [115] :

   La procédure est initialisée en commençant l'activité virtuelle 1 au temps 0. À chaque niveau de l'arbre, l'ensemble SJ_g des activités planifiées et l'ensemble EJ_g des activités éligibles (ceux dont les prédécesseurs ont déjà été planifiés) sont déterminés. Une activité j_g éligible est sélectionnée. Maintenant, la date de début au plus tôt qui satisfait les contraintes de précédence et de ressources est calculée. Le niveau suivant est par la suite considéré. Si la dernière activité virtuelle $(n+2)$ est éligible, une planification complète est trouvée. Dans ce cas, un backtracking[2] vers les niveaux précédents est prévu. Dans ce cas, la prochaine activité éligible non testée est choisie. Si toutes les activités éligibles sont testées, on remonte encore d'un niveau dans l'arbre .

   Un autre exemple est donné en [48] où une autre procédure de séparation et évaluation est définie mais dans laquelle chaque niveau g de l'arbre est plutôt associé à un *instant* t_g (dit *instant de décision*) auquel les activités peuvent commencer. Dans ce cas, une activité j_g est dite éligible au temps t_g si tous ses prédécesseurs sont planifiés et se terminent avant t_g.

[2] Un mécanisme de retour aux étapes précédentes

Il existe, également, d'autres méthodes par séparation et évaluation qui n'utilisent pas des planifications partielles. Un exemple de ce type de démarche est donné en [24]. Elle est une généralisation de celle proposée en [20, 94] pour le problème de job shop. En [41], les noeuds de l'arbre d'énumération représentent le réseau $G = (V', E)$ associé au problème auquel des contraintes de précédence sont ajoutées pour réduire les conflits sur les ressources.

Certaines remarques pertinentes ont été formulées en [70] par rapport à ce type de procédures :

(a) Les procédures de recherche par séparation et évaluation basées sur la résolution des conflits sur les ressources en retardant des sous-ensembles minimaux d'activités sont les meilleures pour résoudre le problème de façon optimale.

(b) Il est important de trouver un *bon compromis* entre la qualité des bornes et le temps nécessaire pour les calculer.

(c) Une attention particulière doit être accordée à la façon de programmer les procédures de résolution.

Les remarques ci-dessus sont à la base de la *DH-procedure* proposée en [48] qui est l'une des méthodes exactes les plus efficaces pour résoudre ce type de problème [19, 70]. En [49], la *GDH-procedure* est proposée comme une extension de la *DH-procédure* à une généralisation du problème où plusieurs autres types de contraintes sur les activités et sur les ressources sont considérées.

2. *Autres méthodes*

Quelques méthodes de résolution basées sur des techniques de la programmation mathématiques sont proposées en [17, 18] et d'autres sur la programmation dynamique sont proposées en [18, 28].

Le problème (MPS | Prec | C_{max})

Les méthodes de résolution de ce problème sont, pour la plupart, des généralisations de celles dédiées à la résolution du problème (PS | Prec | C_{max}). Parmi elles, nous citerons principalement les méthodes par séparation et évaluation. À titre d'exemple, nous montrons ici comment la procédure de séparation et évaluation mentionnée dans la section 1.5.2 a été généralisée en [128] pour ce problème : dans l'arborescence, une activité éligible j_g et ensuite un mode m_{j_g} sont sélectionnés. Chaque combinaison d'une

activité et d'un de ses modes de réalisation correspond à un descendant du noeud en cours de l'arborescence.

Une autre procédure de séparation et évaluation est proposée en [128]. Par ailleurs, une extension de la *DH-procedure* proposée initialement pour le (PS | Prec | C_{max}) a été faite en [129] pour le problème.

Le problème (PS | Prec, Inter | C_{max})

Pour ce type de problème, notons qu'il est possible de trouver une durée minimale pour le projet qui est *inférieure* à celle pour le problème (PS | Prec | C_{max}). Ceci est dû au fait que toute solution de (PS | Prec | C_{max}) est aussi une solution de (PS | Prec, Inter | C_{max}) (le contraire n'est pas toujours vrai).

La littérature qui traite ce type de problème est presque inexistante. Parmi le peu d'approches qui existent, nous pouvons citer celle en [37] qui fait appel à une énumération implicite et celles en [76, 75] qui utilisent la programmation dynamique. Par ailleurs, en [47], la *DH-procedure* a été étendue pour cette variante du problème.

1.5.3 Méthodes approchées

La littérature portant sur ce type de méthodes de résolution est assez riche. La plupart des méthodes que nous mentionnons ici concernent le problème $RCPSP$ de base car, d'une part, c'est cette variante qui a fait l'objet de plus de travaux de recherche et d'autre part, la majorité des méthodes approchées concernant les autres variantes sont de simples extensions de celles développées initialement pour le $RCPSP$.

Le problème (PS | Prec | C_{max}) ou $RCPSP$

Nous pouvons classer ces méthodes selon les catégories suivantes :

1. *Scheduling Generation Scheme* [87, 91]
   (a) **Serial SGS** : Cette méthode augmente, au fur et à mesure, le nombre d'activités planifiées. À chaque étape, une activité non fictive est sélectionnée parmi l'ensemble des activités éligibles puis planifiée de sorte à débuter au plus tôt et à satisfaire les contraintes de précédence ainsi que celles sur les ressources. L'ensemble des activités éligibles comprend toutes les activités non planifiées dont les prédécesseurs ont déjà été planifiés. La complexité de Serial SGS est $O(|R^{\rho}| \times n^2)$.

S'il n'y a pas de contraintes sur les ressources, cette procédure construit une solution optimale (dans ce cas, nous obtenons le problème d'ordonnancement classique). Serial SGS génère des solutions dites *actives* (aucune activité ne peut être exécutée plus tôt sans retarder d'autres) et elle *n'exclut jamais, à priori, la solution optimale.*

List scheduling est une variante de Serial SGS. Nous considérons une liste d'activités $\lambda = (j_1, \cdots, j_n)$ où chaque activité j_g a tous ses prédécesseurs positionnés dans la liste avant la position de j_g. Étant donné une liste λ, les activités peuvent être planifiées, dans l'ordre où elles apparaissent, à la date au plus tôt de sorte à satisfaire les contraintes sur les ressources. Notons qu'il existe toujours une liste λ^* pour laquelle la solution générée *est optimale* quand une fonction objectif *régulière* est considérée (fonction qui est *non décroissante en terme des dates de fin des activités*).

List Scheduling peut être considérée comme une méthode de décodage, c'est-à-dire, permettant le passage d'une liste ordonnée d'activités à un planning proprement dit. En fait, plusieurs méthodes heuristiques exploitent ce principe [1, 16, 66] en explorant plusieurs listes d'activités, auxquelles une variété d'opérations sont appliquées, pour en retenir celle qui donne le meilleur planning.

Le mode de planification selon le principe de *List Scheduling* qui consiste à planifier au *plus tôt* les activités de λ de j_1 à j_n est appelé le mode *forward.* En fait, il est aussi possible de planifier les activités selon un mode appelé *backward.* Dans ce cas, les activités sont planifiées de j_n à j_1 à leur *dates au plus tard.* Pour une même liste λ, les plannings obtenus selon chaque mode peuvent être complètement différents.

Une planification utilisant les deux modes dite *bidirectionnal planning* a été proposée en [79]. Elle consiste à planifier une partie des activités de la liste en mode forward et l'autre en mode backward. Les deux plannings partiels ainsi obtenus sont par la suite fusionnés pour obtenir le planning final.

Serial SGS ainsi que la représentation en liste d'activités seront plus amplement détaillées dans la section 2.2.

(b) **Parallel SGS** : Contrairement à Serial SGS, cette méthode procède par une incrémentation du *temps.* À chaque itération g, il y a un temps de planification

t_g et un ensemble d'activités éligibles. Une activité est éligible si elle peut commencer à la date t_g en respectant les contraintes de précédence et de ressources. Les activités éligibles sont ainsi choisies jusqu'à ce qu'il ne reste plus d'activités éligibles. Après cela, un prochain temps de planification est calculé. Il est donné par la date de fin la plus petite des activités en cours d'exécution au temps t_g. La complexité de Parallel SGS est $O(|R^\rho| \times n^2)$.

L'inconvénient majeur de cette méthode réside dans la possibilité d'*exclure, à priori, toutes les solutions optimales* lorsqu'une fonction objectif régulière est considérée.

2. *Les règles de priorité*

Ce sont parmi les premières heuristiques proposées. Plusieurs d'entres elles ont été testées expérimentalement [3, 14]. L'avantage de ce type d'heuristiques est qu'elles sont intuitives, faciles à implémenter et rapides en terme de temps de calcul. Par ailleurs, elles ne donnent pas, dans la majorité des cas, des solutions satisfaisantes ne déviant pas trop de la valeur optimale.

En utilisant le principe de Serial SGS, les règles de priorité permettent de construire un planning de la façon suivante : à chaque étape, la prochaine activité à planifier est celle dont les prédécesseurs ont déjà été planifiés *et* dont la *priorité est la plus élevée*. Voici quelques règles de calcul de priorité qui existent dans la littérature et dont plusieurs sont résumées dans [59] :

- *MINSLK (Minimum Job Slack)* : Les activités sont ordonnées selon leur *marge*. La marge associée à une activité est définie comme étant la différence entre ses dates de *début au plus tôt* et de *début au plus tard*. Ces dates sont déduites par la méthode CPM[3]. La priorité est donnée à l'activité ayant la marge la plus petite.
- *MINLFT (Minimum Late Finish Time)* : Les activités sont ordonnées selon leur *date de fin au plus tard*. Ces dates sont calculées par la méthode CPM. La priorité est donnée à l'activité ayant la date de fin au plus tard la plus petite.
- *SIO (Shortest Imminent Operation)* : La priorité est donnée à l'activité ayant la *durée la plus courte*.
- *RAN (Select Job Randomly)* : Consiste à choisir chaque activité, parmi celles qui sont éligibles, de façon aléatoire.

[3]Critical Path Method (pour retrouver le chemin critique) : utilisée pour résoudre le problème d'ordonnancement classique

3. *Les métaheuristiques*

   (a) **Métaheuristiques classiques** : Plusieurs adaptation des métaheuristiques classiques ont été faites pour la résolution du problème. La majorité d'entre elles utilisent le principe de Serial SGS avec des listes d'activités pour codifier les solutions. Parmi ces approches, nous retrouvons la recherche tabou [4, 6, 105, 116], le recuit simulé [15, 16, 32], les algorithmes génétiques [1, 2, 34, 61, 66, 83] et les colonies de fourmis [98].

   Deux des meilleures approches heuristiques proposées pour le problème (l'une basées sur le recuit simulé [16] et l'autre sur les algorithmes génétiques [66]) seront décrites en détails dans les sections 3.2 et 3.3. Par ailleurs, nous donnons ici l'exemple d'une approche de résolution basée sur la recherche tabou et proposée en [80]. Les principaux éléments de l'approche sont les suivants :

      i. *Représentation des solutions* : Elle est faite sous forme de *liste d'activités* selon le principe de serial SGS.

      ii. *Voisinage d'une solution* : Chaque élément du voisinage est obtenu en *permutant deux activités* i et j de la liste d'activités (où i est positionné avant j). Comme la relation de précédence doit être préservée dans la liste d'activité résultante, les successeurs (respectivement prédécesseurs) de i (respectivement j), qui se trouvent entre i et j, seront positionnés juste après i(respectivement avant j), si nécessaire.

      iii. *Liste tabou* : Le but de cette liste est d'éviter de revisiter des solutions récemment parcourues. Dans ce cas, chaque élément de la liste représente *une permutation déjà faite* entre deux activités i et j afin d'éviter, sur un certain nombre d'itérations, de renverser cette permutation.

      iv. *Diversification* : Cette opération est faite lorsqu'un même ensemble de solutions est visité de façon répétée. Dans cette approche, ceci est fait lorsque 3 solutions sont parcourues 2 fois. La diversification consiste à *initialiser la procédure avec une nouvelle solution issue d'une liste d'activités construite aléatoirement.*

4. *Stratégies d'amélioration des plannings*

   Ce sont des méthodes simples qui sont appliquées aux plannings pour tenter d'améliorer leur durée. Dans [139], la méthode de *justification* est proposée. Globalement,

une *justification à droite (respectivement à gauche)* d'un planning P consiste à ordonner ses activités par ordre décroissant (respectivement croissant) de leurs dates de fin (respectivement date de début) et de planifier chacune d'elles, selon cet ordre, à leurs dates de fin (respectivement début) au plus tôt (respectivement au plus tard). Le planning P^D (respectivement P^G) résultant aura une durée *inférieure ou égale* à celle de P. Généralement, une *double justification* est appliquée à P pour obtenir le planning $(P^D)^G$. Dans [137], les auteurs présentent un résumé de plusieurs autres méthodes similaires.

Le problème (MPS | Prec | C_{max})

En [13, 55, 117], des méthodes basées sur les règles de priorité sont proposées. En [125], les auteurs proposent une démarche de résolution basée sur la le recuit simulé. Une adaptation de la recherche tabou est faite en [43] pour la résolution du problème avec une généralisation des contraintes de précédence. Un autre algorithme spécifique de recherche locale est décrit en [89].

À titre d'exemple, nous décrivons ici le principe d'un des algorithmes génétiques les plus efficaces proposé pour le problème [67]. L'algorithme manipule des listes d'activités représentant des solutions du problème. Les solutions correspondantes peuvent être obtenues en leur appliquant Serial SGS. Voici les principales opérations de l'algorithme :

- *Croisement* : Pour générer la *fille* et le *fils*, le croisement entre le *père* et la *mère* se fait comme suit : deux entiers $1 \leq w_1, w_2 \leq n$ sont choisis aléatoirement. Pour la fille (respectivement fils), les w_1 premières activités sont prises de la mère (respectivement père), les activités aux positions $w_1 + 1, \cdots, n$ sont choisies dans l'ordre où elles apparaissent dans le père (respectivement la mère). Bien entendu, les activités déjà choisies de la mère (respectivement du père) ne peuvent être encore choisies du père (respectivement de la mère). Par ailleurs, les modes de réalisation des activités sur les positions $1, \cdots, w_2$ de la fille (respectivement fils) sont pris de la mère (respectivement du père) et le reste (sur les positions $w_2 + 1, \cdots, n$) du père (respectivement de la mère).
- *Mutation* : Deux entiers $1 \leq q_1 < n$, $1 \leq q_2 \leq n$ sont générés aléatoirement. Les activités aux positions q_1 et $q_1 + 1$ sont permutées (en gardant leur mode de réalisation) si le planning ainsi obtenu satisfait toujours les contraintes de précédence. Enfin, un autre mode de réalisation est choisi de façon aléatoire pour l'activité se

trouvant à la position q_2.

- *Stratégie de "culling" pour générer la prochaine population* : Une variante de la méthode consiste à sélectionner les meilleurs individus (par rapport à la durée des plannings qu'ils représentent) pour construire la nouvelle génération.

Autres types de problèmes

1. (PS | $d_j = sto$ | C_{max})

   Dans la pratique, il ne suffit pas de fixer des durées constantes pour les activités pour pouvoir évaluer un bon planning puisque ces durées sont généralement des estimations grossières sujettes à des changements imprévisibles. La prise en considération d'un vecteur des durées *aléatoires* pour les activités s'impose et rend la formulation plus réaliste. Cette façon de faire est motivée essentiellement par le fait que $C_{max}(E(d_1), \cdots, E(d_n)) \leq E(C_{max}(d_1, \cdots, d_n))$ [19]. En d'autres termes, souvent les durées minimales calculées pour les projets avec des temps d'exécution estimés et fixés au départ, sont *des sous-estimations des durées minimales moyennes (ou espérées) réelles.*

   Pour ce type de problème, plusieurs méthodes existent. Globalement, elles peuvent être classées dans l'une ou l'autre des catégories suivantes [19] :
   - Méthodes de simulation [26, 123, 132, 140].
   - Méthodes pour borner ou calculer l'espérance de la durée du projet [51, 54].
   - Méthodes pour l'analyse du chemin "le plus critique" [126, 127].
   - Méthodes pour borner la fonction de distribution de la durée du projet [52, 63].

2. (PS | Prec, $d_j = sto$ | C_{max})

   La modélisation de ce type de problème conduit souvent au domaine de la programmation dynamique où il est question de trouver des *politiques* ou des *stratégies* optimales. Des approches de résolutions pour ce type de problèmes sont décrites dans [101, 102].

1.5.4 Bornes inférieures pour le *RCPSP*

Le calcul d'une borne inférieure pour ce problème peut s'avérer très utile. Déjà, elle donne une idée sur la durée minimale du projet qu'il n'est pas possible d'améliorer. Aussi, cela peut servir dans des procédures par séparation et évaluation pour résoudre des problèmes ayant plus de contraintes. Plus de telles bornes sont rapides à calculer et

leur valeur ne s'éloigne pas trop de la valeur optimale de la fonction objectif, plus les procédures de résolutions qui les utilisent seront efficaces.

Généralement, les bornes inférieures pour ce problème sont calculées en relaxant certaines de ses contraintes puis en résolvant le problème résultant jusqu'à l'optimalité lorsque cela est possible. Une des façons les plus simples de le faire est de relaxer les contraintes sur les ressources. Ceci mène au problème d'ordonnancement classique pour lequel le calcul de la durée minimale du projet est facile (la méthode CPM le résout en un temps polynomial). En [131], une amélioration de cette borne est proposée en modifiant le réseau $G = (V', E)$ associé au problème. Deux méthodes de calcul d'une borne inférieure sont décrites dans [81]. D'autres bornes, dont l'évaluation est basée sur la résolution d'un programme linéaire, sont présentées en [27, 99].

Notons qu'il existe aussi dans la littérature des bornes pour les autres variantes du problème. Parmi elles, nous pouvons citer celle présentée en [23] pour une généralisation du problème (MPS | Prec | C_{max}).

1.6 Facteurs et paramètres ayant un lien avec la complexité du *RCPSP*

Il est difficile d'isoler les facteurs permettant de générer des instances "faciles" ou "difficiles" à résoudre et d'après lesquels il est possible de prévoir les efforts de calculs nécessaires.

Plusieurs auteurs ont tenté d'identifier de tels facteurs. Malheureusement, il n'est pas possible de prouver de façon rigoureuse leur pertinence quant à leur lien avec la complexité réelle du problème [70]. Voici quelques uns de ces facteurs :

1. *NC* (*Coefficient of network complexity*) : Ce paramètre a été introduit en [111]. Il est défini comme étant le rapport entre le nombre d'arcs et le nombre de sommets du réseau activités-noeuds $G = (V', E)$ associé au problème. Nous obtenons une plus grande complexité en rajoutant des arcs au réseau (donc des contraintes de précédence).
2. *RF* (*Resource factor*) : Ce facteur a été introduit en [111] et reflète la portion moyenne des ressources nécessaires par activité. Sa formulation est la suivante :

$$RF = \frac{1}{|V|}\frac{1}{|R^\rho|}\sum_{j=1}^{V}\sum_{k\in R^\rho}\delta_{jk}$$

où

$$\delta_{jk} = \begin{cases} 1 & \text{si } r^\rho_{jk} > 0 \\ 0 & \text{sinon.} \end{cases}$$

Si $RF = 1$, alors chaque activité requiert toutes les ressources et si $RF = 0$, aucune activité ne requiert de ressources.

3. RS (*Resource strength*) : Ce paramètre a été introduit en [36] et redéfini ensuite en [93]. Il exprime la relation entre les demandes des activités, en terme de ressources, par rapport à leur disponibilité. Cette quantité est associée à une ressource $k \in R^\rho$ et peut s'exprimer comme suit :

$$RS_k = \frac{R^\rho_k - r^{min}_k}{r^{max}_k - r^{min}_k}$$

Où :

- $r^{min}_k = max_{i=1,\cdots,|V|}\ r^\rho_{ik}$.
- r^{max}_k : La demande maximale de la ressource k lorsque nous considérons le planning obtenu en ne tenant compte que des contraintes de précédence et où les activités commencent à leur dates au plus tôt.

Notons ici que si $RS_k = 1$, le problème n'est plus contraint par les ressources dans le sens où ces dernières sont disponibles en quantités importantes qui ne retardent pas la planification des activités.

4. OS (*Order Strength*) : Il est défini comme étant le rapport entre le nombre de relations de précédence (incluant celles qui sont transitives) par le nombre $\frac{n(n-1)}{2}$ [97].

5. Autres : D'autres paramètres existent aussi dans la littérature comme l'index de complexité (*Complexity Index* (CI)) [9] et la Restrictivité (*Restrictiveness* (RT)) [39].

1.7 Problèmes tests

Des problèmes dits *tests* ont été mis au point par différents auteurs dans la littérature. Ce sont des instances du problème avec lesquelles l'efficacité des procédures de résolution est évaluée.

Deux de ces ensembles de problèmes tests sont souvent mentionnés et utilisés : les 110 *problèmes tests de Patterson* [114] et ceux de la bibliothèque *PSPLIB* [93] générés par un programme nommé ProGen. Cela dit, l'efficacité de la majorité des heuristiques récentes est évaluée avec la bibliothèque PSPLIB.

1.7.1 Les 110 problèmes tests de Patterson

Ces problèmes, présentés en [114], ont été collectés de différentes sources. Ils ont longtemps constitué une référence importante pour évaluer l'efficacité des procédures de résolution.

1.7.2 La bibliothèque PSPLIB

Le programme ProGen (Parameter Driven Project Generator), présenté en [93], a été utilisé pour générer des instances du problème selon différents types de *paramètres* fixés au départ. Ces instances ont été largement utilisées dans la littérature et sont regroupées dans une bibliothèque nommée *PSPLIB* qui est disponible sur *http ://www.bwl.uni-kiel.de/prod/psplib/*. Pour le $RCPSP$, elle contient des instances ayant 30, 60, 90 et 120 activités non fictives (notés J30, J60, J90 et J120 respectivement). Il existe 480 instances pour chacun des ensembles J30, J60 et J90 et 600 instances pour J120.

Les instances de J30, J60, J90 et J120 ont été générées en veillant à ce qu'il aient des valeurs bien précises des paramètres NC, RF et RS présentés dans la section 1.6. Pour les ensembles J30, J60 et J90, ces paramètres prennent leurs valeurs dans le Tableau 1.3. Pour J120, ces paramètres prennent leurs valeurs dans le Tableau 1.4.

NC	1.5	1.8	2.1	
RF	0.25	0.5	0.75	1
RS	0.2	0.5	0.7	1

TAB. 1.3 – Valeurs des paramètres pour J30, J60 et J90

NC	1.5	1.8	2.1		
RF	0.25	0.5	0.75	1	
RS	0.1	0.2	0.3	0.4	0.5

TAB. 1.4 – Valeurs des paramètres pour J120

Pour chacune des $3 \times 4 \times 4 = 48$ combinaisons des valeurs des paramètres sur le Tableau 1.3, 10 instances sont générées pour chacun des ensembles J30, J60 et J90. C'est de cette façon que nous retrouvons 480 instances pour chacun de ces ensembles. De même pour J120, 10 instances sont générées pour chacune des $3 \times 4 \times 5 = 60$ combinaisons des paramètres du Tableau 1.4 pour obtenir un total de 600 instances.

1.8 Résultats expérimentaux existants pour *RCPSP*

L'évaluation des démarches de résolution qui existent dans la littérature se fait, généralement, selon les deux critères suivants :

1. *La qualité de la meilleure solution obtenue* mesurée par le *pourcentage de déviation.* Cette quantité mesure l'écart entre la valeur que la solution donne à l'objectif et une valeur de référence (souvent, la valeur optimale ou une borne à cette dernière). Sa formulation est la suivante :

$$\left(\frac{\text{Valeur de l'objectif donnée par la solution - Valeur de référence}}{\text{Valeur de référence}} \times 100\right)\%$$

2. *Le temps de calculs* nécessaire pour l'obtenir. Ceci peut être donné en terme de temps proprement dit (mesuré en secondes généralement) ou bien en terme de *nombre de solutions (plannings) qu'il a fallu construire durant toute la procédure de résolution.* Cette dernière quantité semble être la plus utilisée dans la littérature.

1.8.1 Méthodes exactes

Nous présentons ici deux tableaux comparatifs 1.5 et 1.6 de quelques méthodes exactes dont certaines ont été déjà mentionnées plus haut. Le Tableau 1.5 concerne les 110 problèmes tests de Patterson et le Tableau 1.6 concerne les 480 instances de J30 de la bibliothèque PSPLIB (le temps est donné en secondes).

Il apparaît clairement que la méthode de Demeulemeester et Harrolen [50, 48] est parmi les meilleures méthodes exactes pour le problème.

Par ailleurs, la procédure présentée en [41] pour la résolution du problème avec une généralisation des contraintes de précédence, se révèle également assez performante. Elle est capable de résoudre à l'optimum des instances pouvant avoir jusqu'à 100 activités en un temps raisonnable. La procédure a été testée sur un ensemble d'instances générées avec une variante du programme ProGen et peut être utilisée pour la résolution du problème *RCPSP*.

La majorité des méthodes exactes que nous avons recensées ne sont pas récentes. Ceci est probablement dû à la complexité du problème (voir la section 1.4.3) qui incite à investir plus dans des approches heuristiques pour résoudre des instances de taille plus importante.

Auteurs	Configuration	Temps moyen	Temps limite	# Optimums
Davis et al. [37]	Amdahl 470/V8	14.02	300	96
Talbot et al. [133]	Amdahl 470/V8	14.98	300	97
Stinson et al. [131]	Amdahl 470/V8	0.82	300	110
Christofides et al. [33]	IBM 3090	56.82	Non	110
Bell et al. [10]	Apple Macintoch Plus	340.57	Non	110
Demeulemeester et al. [48]	80386 25Mhz	0.21	Non	110
Demeulemeester et al. [50](DH1)	80386 25Mhz	0.02	Non	110
Demeulemeester et al. [50](DH2)	Pentium Pro 200Mhz	0.002	Non	110

TAB. 1.5 – Comparaison des méthodes sur les 110 problèmes tests de Patterson [70]

Auteurs	Configuration	Temps moyen	Temps limite	# Optimums
Demeulemeester et al. [48]	80386 25Mhz	79.91	3600	428
Mingozi et al. [99]	80386 25Mhz	Non mentionné	Non	480
Demeulemeester et al. [50](DH1)	80386 25Mhz	12.33	3600	479
Brucker et al. [24](DH2)	Pentium Pro 200Mhz	0.37	Non	480
	Sun/Sparc 20/801	17.88	3600	407

TAB. 1.6 – Comparaison des méthodes sur les 480 instances de J30 (30 activités) [70]

1.8.2 Méthodes heuristiques

Une comparaison récente des performances de plusieurs méthodes heuristiques a été faite en [90]. Les tableaux 1.7, 1.8 et 1.9 listent les 20 meilleures heuristiques recensées pour le problème $RCPSP$ pour chaque ensemble d'instances J30, J60 et J120 de PSPLIB avec une limite de 5000 plannings générés (nombre de solutions parcourues avant l'arrêt des calculs). Sur ces tableaux, la colonne *Déviation* donne la déviation moyenne sur toutes les instances d'un même ensemble par rapport à la durée des plannings optimaux pour J30 (puisqu'ils sont connus pour ces instances) et par rapport aux bornes inférieures

données par la méthode CPM pour J60, J90 et J120.

Selon ces résultats, une variété d'heuristiques enregistrent de bonnes performances (algorithmes génétiques, recherche tabou, recuit simulé, colonies de fourmis, recherche dispersée, ..., etc.). Kolisch et al. [90] soulignent que les méthodes les plus performantes sont celles de Kochetov et Stolyar [82], Valls et al. [138], Alcaraz et al. [2], Debels et al. [45] et Hartmann [68].

Auteurs	Algorithmes	Déviations
Kochetov et Stolyar [82]	Algorithmes Génétiques et Recherche Tabou	0.04
Valls et al. [138]	Algorithmes Génétiques hybride	0.06
Alcaraz et al. [2]	Algorithmes Génétiques	0.06
Debels et al. [45]	Scatter search	0.11
Alcaraz et Maroto [1]	Algorithmes Génétiques	0.12
Tormos et Lova [136]	Sampling	0.13
Nonobe et Ibaraki [105]	Recherche Tabou	0.16
Tormos et Lova [137]	Sampling	0.16
Tormos et Lova [135]	Sampling	0.17
Klein [80]	Recherche Tabou	0.17
Valls et al. [139]	Algorithmes Génétiques	0.2
Hartmann [68]	Algorithmes Génétiques	0.22
Bouleimen et Lecocq [16]	Recuit Simulé	0.23
Hartmann [66]	Algorithmes Génétiques - variante 1	0.25
Valls et al. [139]	Sampling	0.28
Coelho et Tavares [34]	Algorithmes Génétiques	0.33
Schirmer [121]	Sampling	0.44
Baar et al. [6]	Recherche Tabou	0.44
Kolisch et Drexel [88]	Sampling	0.52
Hartmann [66]	Algorithmes Génétiques - variante 2	0.56

TAB. 1.7 – Heuristiques - résultats sur **J30** avec 5000 plannings générés [90]

Auteurs	Algorithmes	Déviations
Debels et al. [45]	Scatter search	11.1
Valls et al. [138]	Algorithmes Génétiques hybride	11.1
Kochetov et Stolyar [82]	Algorithmes Génétiques et Recherche Tabou	11.17
Alcaraz et al. [2]	Algorithmes Génétiques	11.19
Valls et al. [139]	Algorithmes Génétiques	11.27
Tormos et Lova [136]	Sampling	11.62
Hartmann [68]	Algorithmes Génétiques	11.7
Tormos et Lova [135]	Sampling	11.82
Alcaraz et Maroto [1]	Algorithmes Génétiques	11.86
Tormos et Lova [137]	Sampling	11.87
Hartmann [66]	Algorithmes Génétiques - variante 1	11.89
Bouleimen et Lecocq [16]	Recuit Simulé	11.9
Klein [80]	Recherche Tabou	12.03
Nonobe et Ibaraki [105]	Recherche Tabou	12.18
Valls et al. [139]	Sampling	12.35
Schirmer [121]	Sampling	12.58
Coelho et Tavares [34]	Algorithmes Génétiques	12.63
Kolisch et Drexel [88]	Sampling	13.06
Hartmann [66]	Algorithmes Génétiques - variante 2	13.32
Hartmann [66]	Algorithmes Génétiques - variante 3	13.32

TAB. 1.8 – Heuristiques - résultats sur **J60** avec 5000 plannings générés [90]

Auteurs	Algorithmes	Déviations
Valls et al. [138]	Algorithmes Génétiques hybride	32.54
Debels et al. [45]	Scatter search	33.1
Valls et al. [139]	Algorithmes Génétiques	33.24
Kochetov et Stolyar [82]	Algorithmes Génétiques et Recherche Tabou	33.36
Alcaraz et al. [2]	Algorithmes Génétiques	33.91
Valls et al. [139]	Algorithme basé sur les populations	34.02
Tormos et Lova [136]	Sampling	34.41
Hartmann [68]	Algorithmes Génétiques	35.39
Merkle et al. [98]	Colonnies de fourmis	35.43
Tormos et Lova [135]	Sampling	35.56
Tormos et Lova [137]	Sampling	35.81
Alcaraz et Maroto [1]	Algorithmes Génétiques	36.57
Hartmann [66]	Algorithmes Génétiques - variante 1	36.74
Valls et al. [139]	Sampling	37.47
Bouleimen et Lecocq [16]	Recuit Simulé	37.68
Nonobe et Ibaraki [105]	Recherche Tabou	37.88
Coelho et Tavares [34]	Algorithmes Génétiques	38.41
Hartmann [66]	Algorithmes Génétiques - variante 3	38.49
Schirmer [121]	Sampling	38.7
Kolisch [87]	Sampling	38.75

TAB. 1.9 – Heuristiques - résultats sur **J120** avec 5000 plannings générés [90]

Chapitre 2

Représentation en séquence d'ensembles d'activités

2.1 Introduction

Tout au long de ce chapitre, nous nous intéressons *uniquement* à la variante $RCPSP$ du problème.

Une multitude d'approches de résolution du problème $RCPSP$ existent dans la littérature. Dans plusieurs d'entre elles, la solution est codifiée selon une représentation qui est manipulée jusqu'à l'obtention d'une codification donnant satisfaction. La solution finale est déduite de la codification ainsi obtenue.

Une des étapes les plus importantes dans cette catégorie d'approches est le choix de la représentation ainsi que des opérations à appliquer à ces dernières.

La majorité des meilleures heuristiques connues jusqu'à présent pour le problème $RCPSP$ utilisent une représentation (ou codification) *en séquence d'activités*[1] ainsi que Serial SGS (voir la section 1.5.3) comme technique de décodage (passage de la codification au planning proprement dit) [1, 2, 16, 66, 105].

Nous proposons ici une nouvelle représentation des solutions sous forme de *séquence d'ensembles d'activités* et nous montrons comment, à travers ses nombreuses et puissantes propriétés, elle peut se révéler plus efficace que la représentation sous forme de séquence d'activités.

[1] Appellation anglophone : Activity list representation

2.2 Séquence d'activités

Une séquence d'activités **(s)** est une liste ordonnée de toutes les activités. Nous dirons, par la suite, qu'une séquence est **(vp)** (**vérifie les contraintes de précédence**) si toute activité *i y est positionnée après tous ses prédécesseurs* P_i.

Rappelons que la méthode Serial SGS procède comme suit : les activités de la (s)(vp) sont sélectionnées une par une selon leur ordre. Chaque activité sélectionnée est planifiée à sa date au plus tôt de sorte à satisfaire les contraintes sur les ressources et les contraintes de précédence. Comme la séquence d'activités est (vp), lorsqu'une activité doit être planifiée, ses prédécesseurs sont déjà tous planifiés.

Dans leur article, Alcaraz et Maroto [1] soulignent une caractéristique très importante concernant les (s)(vp) et la méthode Serial SGS : *un même planning peut être engendré par plusieurs (s)(vp) différentes.* Par ailleurs, la méthode Serial SGS associe toujours *un unique* planning à chaque (s)(vp). Ceci signifie que les approches utilisant les (s)(vp) et Serial SGS *peuvent parcourir beaucoup plus de (s)(vp) différentes que de plannings différents.* La Figure 2.1 montre un exemple de problème $RCPSP$, représenté sous forme d'un réseau, avec un unique type de ressource disponible en quantité 6 par période élémentaire avec 8 activités à planifier. La Figure 2.2 montre *un planning pouvant être engendré par les deux (s)(vp) distinctes* A et B suivantes :

$$A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

$$B = [1, 2, 5, 3, 4, 6, 8, 7, 9, 10]$$

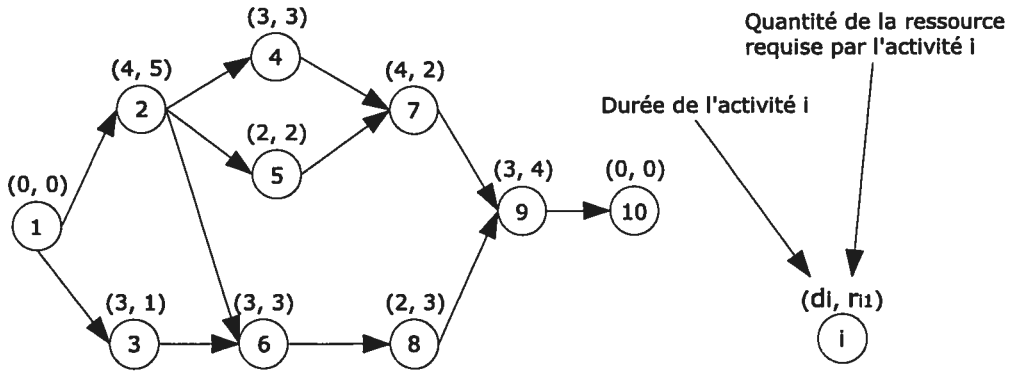

FIG. 2.1 – Un exemple de problème $RCPSP$

Nous proposons, dans ce qui suit, une nouvelle représentation *généralisant la représentation sous forme de (s)(vp)* et permettant une *réduction considérable de l'espace de*

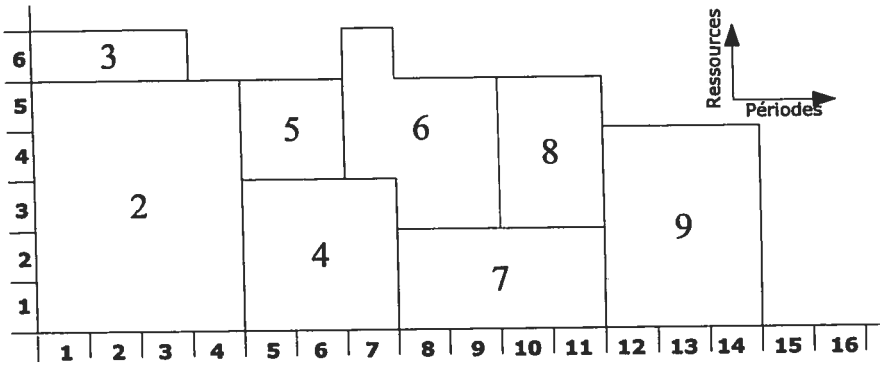

FIG. 2.2 – Planning engendré par deux (s)(vp) différentes

recherche induit par ces dernières. Nous montrons que chaque représentation de ce type correspond à *plusieurs représentations différentes* du type (s)(vp) induisant toutes *un même planning.* Aussi, comme pour les (s)(vp), nous démontrons que cette nouvelle représentation *n'exclut jamais la solution optimale du problème* (si elle existe) et que tout planning pouvant être obtenu grâce à une (s)(vp) peut également être obtenu avec cette nouvelle représentation.

Notons, par ailleurs, que dans les heuristiques exploitant les (s)(vp), le temps calcul le plus important est généralement monopolisé par le passage de la (s)(vp) au planning proprement dit [16] (temps nécessaire pour exécuter Serial SGS). En réduisant le risque de passage à des (s)(vp) engendrant un même planning, une exploration plus efficace de l'ensemble des plannings possibles pourra être faite.

2.3 Séquences d'ensembles d'activités

Nous donnons ici les définitions nécessaires à la caractérisation de la nouvelle représentation ainsi que ses propriétés.

Définition 1 Une séquence d'ensembles d'activités **(se)** est une liste ordonnée d'ensembles d'activités distincts et non vides qui partitionnent l'ensemble de toutes les activités $V \cup \{1,(n+2)\}$. En d'autres termes, une (se) sera représentée comme suit :

$$[A_1, A_2, \cdots, A_p]$$

telle que :

- $\forall i \in V \cup \{1,(n+2)\},\ \exists! z \in \{1,\cdots,p\}/i \in A_z.$
- $\forall z \in \{1,\cdots,p\},\ A_z \neq \emptyset.$

- $A_1, A_2, \cdots, A_p$ est une liste ordonnée.

Par analogie avec les (s)(vp) nous donnons la définition suivante :

Définition 2 Une (se) : $[A_1, A_2, \cdots, A_p]$ est **(vp)** (vérifie les contraintes de précédence) si toute activité i appartenant à un ensemble A_z ($z \in \{1, \cdots, p\}$) a tous ses prédécesseurs P_i dans $\cup_{v=1}^{v=z} A_v$.

Avec ces deux définitions, nous pouvons déjà dire que les (se)(vp) sont une *généralisation* des (s)(vp). En effet, une (s)(vp) n'est qu'une (se)(vp) ayant $|V \cup \{1, (n+2)\}| = (n+2)$ ensembles. Par définition des (se), chaque ensemble contiendra donc forcément une unique activité dont les prédécesseurs sont positionnés avant.

Nous ajoutons deux dernières définitions qui concernent une propriété extrêmement puissante caractérisant les (se)(vp).

Définition 3 Soit A_z un ensemble d'une (se)(vp) : $[A_1, A_2, \cdots, A_p]$ ($z \in \{1, \cdots, p\}$). La liste o_z *est* (vp) si elle constitue une liste ordonnée des activités de A_z de sorte qu'une activité de o_z a tous ses prédécesseurs, qui appartiennent à A_z, positionnés avant elle dans o_z. Une telle liste est dite *ordre.*
O_z dénote l'ensemble de tous les ordres o_z pouvant être construits avec l'ensemble A_z.

Définition 4 Une (se) : $[A_1, A_2, \cdots, A_p]$ possède la propriété de permutation **(PRMT)** si elle est (vp) et $\forall (o_1, o_2, \cdots, o_p) \in O_1 \times O_2 \times \cdots \times O_p$, les (s) formées par $o_1, o_2, \cdots, o_p$ *correspondent toutes à un même planning.*

Pour illustrer cela, nous donnons ici un exemple d'une (se) qui est (PRMT) en se référant au problème de la Figure 2.1. Nous considérons la (se) SE suivante :

$$[\{1,2,3,4,5\}, \{6,7,8,9,10\}],$$

où $A_1 = \{1,2,3,4,5\}$ et $A_2 = \{6,7,8,9,10\}$. Le Tableau 2.1 donne la liste de tous les ordres (vp) possibles pouvant être construits avec chacun de ces deux ensembles .

Nous pouvons facilement vérifier que *SE est bien (PRMT).* Ainsi, toutes les $8 \times 3 = 24$ (s), formées par la concaténation ordonnée des ordres (vp) de A_1 *puis* de A_2, sont bien (vp) et *correspondent toutes au même planning*, celui de la Figure 2.2. Notons que la (s)(vp) B, définie plus haut, peut être construite en concaténant les ordres $[1,2,5,3,4] \in O_1$ et $[6,8,7,9,10] \in O_2$.

Ensembles	Ordres (vp) possibles
$A_1 = \{1,2,3,4,5\}$	$O_1 = \{[1,2,3,4,5],[1,2,3,5,4],[1,3,2,4,5],$ $[1,3,2,5,4],[1,2,4,5,3],[1,2,5,4,3],$ $[1,2,4,3,5],[1,2,5,3,4]\}$
$A_2 = \{6,7,8,9,10\}$	$O_2 = \{[7,6,8,9,10],[6,7,8,9,10],[6,8,7,9,10]\}$

TAB. 2.1 – Listes des ordres (vp) possibles avec les ensembles A_1 et A_2

Il est clair, à travers cet exemple, que la notation sous forme de (se)(PRMT) est bien plus *compacte* et plus intéressante que celles des 24 (s)(vp) équivalentes. De plus, nous pouvons affirmer qu'il est *inutile* d'adopter des transformations sur les (s)(vp) donnant lieu à *des permutations des activités d'un même ensemble d'une (se)(PRMT)*, car cela engendre forcément le même planning. Par ailleurs, *il est possible* que certaines permutations impliquant *des ensembles différents, engendrent tout de même le même planning.*

2.3.1 Un algorithme pour la construction des (SE)(PRMT)

La codification sous forme de (se) peut se révéler assez efficace notamment grâce à la propriété (PRMT). Notons qu'il ne suffit pas de partitionner l'ensemble des activités en ensembles puis les ordonner pour obtenir cette propriété. Si nous procèdons de cette façon, même si on peut construire des ordres (vp) pour chaque sous-ensemble, la concaténation ordonnée de ces ordres peut ne pas donner, dans tous les cas, des (s) qui soient (vp) et qui correspondent toutes au même planning. Pour illustrer cela, reconsidérons le problème de la Figure 2.1.

1. La (se) : [{1, 2, 3, 7, 8}{4, 5, 6, 9, 10}] *n'est pas (vp)* puisque l'activité 7 appartient au premier ensemble et ses prédécesseurs $P_7 = \{4,5\}$ appartiennent au second. Par conséquent, toute (s) issue de cette (se) ne sera pas (vp) puisque l'activité 7 y sera positionnée avant ses prédécesseurs.

2. Toutes les combinaisons des ordres de O_1 et O_2 pour la (se) :[{1, 2, 3, 4, 5, 6}{7, 8, 9, 10}] *forment des (s)(vp) mais ne correspondent pas toutes au même planning.* En effet, les (s)(vp) : $[1,2,3,4,5,6,7,8,9,10]$ et $[1,2,3,4,6,5,7,8,9,10]$ appartiennent bien à $O_1 \times O_2$ mais *représentent deux plannings différents* de durées 14 et 16 respectivement.

Nous proposons ici *un algorithme polynomial simple*, nommé **ConstruireSEPRMT**, permettant de construire des (se) qui sont (PRMT). L'algorithme reçoit en entrée n'importe quelle (s)(vp) et donne en sortie une (se) qui est (PRMT) dont l'ensemble des (s)(vp) qu'elle représente contient la (s)(vp) en entrée.

Globalement, l'algorithme procède comme suit : il sélectionne les activités dans l'ordre dans lequel elles apparaissent dans la (s). Un premier ensemble vide est créé et la première activité y est placée. Si la date au plus tôt à laquelle la prochaine activité peut être planifiée, en ne considérant *que les contraintes de précédence*, est égale à la date au plus tôt à laquelle elle peut être planifiée, en considérant *et les contraintes de précédence et celles sur les ressources*, alors l'activité est ajoutée dans l'ensemble en cours. Sinon, la construction de l'ensemble en cours est terminée et un nouvel ensemble, contenant cette activité, est créé. Nous obtenons ainsi une (se) dont les ensembles sont ordonnés selon leur ordre de création. L'algorithme ConstruireSEPRMT est détaillé sur la Figure 2.3.

À titre d'exemple, la (se)(PRMT) : $[\{1,2,3,4,5\},\{6,7,8,9,10\}]$ est obtenue en appliquant ConstruireSEPRMT à la (s)(vp) : $[1,2,3,4,5,6,7,8,9,10]$.

En ce qui concerne la complexité de l'algorithme, ce dernier est polynomial car :

- En (*) : la boucle s'exécute en $O(n)$ fois.
- En (**) : les fonctions $T_P(.)$ et $T_{PR}(.)$ sont polynomiales. Elles sont évaluées en $O(c.n^2)$, où c est une constante positive.

ConstruireSEPRMT est très similaire à Serial SGS dans le sens où la procédure planifie, à la date au plus tôt, les activités en tenant compte des contraintes de précédence et celles sur les ressources. Le seul effort de calcul supplémentaire par rapport à Serial SGS vient de *la comparaison des deux valeurs $T_P(.)$ et $T_{PR}(.)$ pour délimiter les ensembles de la (se)(PRMT) en sortie.* Notons que ces deux valeurs sont aussi calculées dans Serial SGS.

Le résultat suivant démontre que l'algorithme ConstruireSEPRMT donne bien en sortie des (se)(PRMT).

Théorème 1 Initié avec une (s)(vp), l'algorithme *ConstruireSEPRMT* construit une (se) ayant la propriété (PRMT).

Preuve 1 Soit s la (s) en entrée de l'algorithme et soit $se : [A_1, A_2, \cdots, A_p]$ la (se) générée. Montrons que $\forall (o_1, o_2, \cdots, o_p) \in O_1 \times O_2 \times \cdots \times O_p$ alors la (s) $s^0 : o_1, o_2, \cdots, o_p$

- est (vp).
- produit le même planning que celui de s.

1. – s : La (s)(vp) en entrée.
   - l : Le nombre d'ensembles en cours de la (se) en sortie.
   - $s[k]$: L'activité à la position k de la (s) s, avec $k \in \{1, \cdots, (n+2)\}$.
   - $T_P(i)$: La date au plus tôt à laquelle l'activité i peut être planifiée, en considérant *uniquement les contraintes de précédence*, et cela étant donné la planification faite des activités qui précèdent i dans s.
   - $T_{PR}(i)$: La date au plus tôt à laquelle l'activité i peut être planifiée, en considérant *et les contraintes de précédence et les contraintes sur les ressources*, étant donné la planification faite des activités qui précèdent i dans s.
2. – Créer le premier ensemble $A_1 = \{s[1]\}$ de la (se).
   - l :=1.
3. **Pour** $k = 2$ **à** $(n+2)$ **faire**$^{(*)}$
   - **Si** $T_P(s[k]) = T_{PR}(s[k])^{(**)}$ **alors** $A_l := A_l \cup \{s[k]\}$
   - **Sinon**
     - $l := l + 1$.
     - Créer un nouvel ensemble $A_l = \{s[k]\}$.
4. La (se) recherchée est $A_1, A_2, \cdots, A_l$.

FIG. 2.3 – La procédure ConstruireSEPRMT

1. **s^0 est (vp)**

   Soit i une activité de la (s) s^0 et soit $j \in P_i$ un de ses prédécesseurs. Supposons que $i \in o_k, k \in \{1, \cdots, p\}$.

   Comme l'algorithme construit les ensembles $A_1, A_2, \cdots, A_p$ en choisissant les activités dans l'ordre où elles apparaissent dans la (s) s, alors j appartient forcément à $A_1 \cup \cdots \cup A_k$.

   (a) Si $j \in A_1 \cup A_2 \cup \cdots \cup A_{(k-1)}$: alors j fait partie de l'un des ordres $o_1, o_2, \cdots, o_{(k-1)}$ et dans ce cas, j précède i dans s^0.

   (b) Si $j \in A_k$: les activités i et j font partie de l'ordre o_k qui est, par définition de O_k, (vp). Ainsi, j précède i dans s^0 dans ce cas également.

2. **s^0 produit le même planning que s**

   Nous montrons, par récurrence sur k, que les activités dans $A_1 \cup A_2 \cup \cdots A_k$ $(k \leq p)$ sont planifiées selon s^0 aux mêmes dates qu'elles le sont selon s.

   (a) Pour $k = 1$: se référant à l'algorithme, toutes les activités de A_1 sont planifiées selon s à une date $T_P(i) = T_{PR}(i)$. De plus, comme $\forall o_1 \in O_1, o_1$ est (vp), toutes les activités i de A_1 seront planifiées, selon s^0, aux mêmes dates.

   (b) Pour $k = r$ l'affirmation est vrai $\Rightarrow$ pour $k = (r+1)$ l'affirmation est vrai, avec $r \in \{\mathbf{1}, \cdots, (p-1)\}$:

   Donc, supposons que les activités de $A_1 \cup A_2 \cup \cdots A_r$ sont planifiées selon s^0 aux même dates qu'elles le sont selon s et montrons que cette affirmation est vrai pour $A_1 \cup A_2 \cup \cdots A_{(r+1)}$. Pour cela, nous considérons la planification des activités de $A_1 \cup A_2 \cup \cdots A_r$ selon s^0 et nous voulons maintenant planifier les activités selon l'ordre $o_{(r+1)}$. Se référant à l'algorithme, nous pouvons constater que dans $A_{(r+1)}$, il existe une *unique* activité i_0 qui a été planifiée selon s à la date $T_{PR}(i_0)(> T_P(i_0))$ car c'est le critère selon lequel un nouvel ensemble de *se* est créé. Le reste des activités $i \in A_{(r+1)} - \{i_0\}$ sont toutes planifiées aux dates $T_P(i) = T_{PR}(i)$.

   Or, pour n'importe quel ordre $o_{(r+1)} \in O_{(r+1)}$, nous considérons les cas suivants :

   i. Si i_0 est la première activité à planifier dans $o_{(r+1)}$:

      Alors, i_0 sera planifiée à la même date $T_{PR}(i_0)(> T_P(i_0))$ selon s^0 car i_0 a, également, été la première à être planifiée selon s dans $A_{(r+1)}$ et cela a été fait à cette date.

Étant donné la planification de i_0, le reste des activités $i \in A_{(r+1)} - \{i_0\}$ peuvent toutes être planifiées à la date $T_P(i) = T_{PR}(i)$ selon s^0 puisque cela a été possible selon s. Ces activités ne peuvent être planifiées plus tôt, à cause des contraintes de précédence.

ii. Si i_0 n'est pas la première activité à planifier dans $o_{(r+1)}$:

Considérons les ensembles :

$A^{av}_{(r+1)}$: ensemble des activités de $A_{(r+1)} - \{i_0\}$ positionnées, dans $o_{(r+1)}$, *avant* i_0.

$A^{ap}_{(r+1)}$: ensemble des activités de $A_{(r+1)} - \{i_0\}$ positionnées, dans $o_{(r+1)}$, *après* i_0.

- Les activités i de $A^{av}_{(r+1)}$ seront planifiées selon s^0 aux dates $T_P(i) = T_{PR}(i)$, car cela a été possible selon s même en présence du reste des activités de $A_{(r+1)}$. Par ailleurs, elles ne peuvent être planifiées selon s^0 plus tôt à cause des contraintes de précédence.
- En ce qui concerne la planification de i_0, elle a été la première a être planifiée selon s dans $A_{(r+1)}$ et cela a été fait à la date $T_{PR}(i_0)(> T_P(i_0))$. Donc, i_0 ne peut être planifiée plus tôt selon s^0. Par ailleurs, i_0 ne sera pas planifiée plus tard, car il a été possible que i_0 débute en $T_{PR}(i_0)(> T_P(i_0)$ en présence des activités de $A^{av}_{(r+1)}$ dans la planification issue de s.
- Lorsqu'il sera question de planifier les activités i de $A^{ap}_{(r+1)}$ selon s^0, ces dernières peuvent être planifiées aux dates $T_P(i) = T_{PR}(i)$ car, dans la planification donnée selon s, il a été possible que ces activités commencent à ces dates en présence des activités de $A^{av}_{(r+1)} \cup \{i_0\}$ planifiées aux mêmes dates données selon s^0 pour ces activités. Par ailleurs, les activités de $A^{ap}_{(r+1)}$ ne peuvent être planifiées plus tôt selon s^0 à cause des contraintes de précédence.

Ainsi, les activités de $A_1 \cup A_2 \cup \cdots \cup A_k, (k \leq p)$ sont planifiées selon s^0, aux mêmes dates qu'elles le sont selon s.

Donc, *se* est bien (PRMT). □

2.3.2 Résultats supplémentaires sur les (SE)

Les résultats suivants établissent des propriétés très pratiques pour exploiter les (se) dans une approche de résolution.

Théorème 2 Si la (se) construite par l'algorithme *ConstruireSEPRMT* comporte un unique ensemble, alors le planning optimal (solution optimale du problème *RCPSP*) correspond au planning obtenu par la méthode CPM[2].

Preuve 2 En analysant de près l'algorithme, si la (se) obtenue est composée d'un unique ensemble, alors toute activité i que l'algorithme a planifiée, vérifie $T_P(i) = T_{PR}(i)$. Ainsi, toutes les activités ont pu être planifiées à des dates au plus tôt qui peuvent être obtenues rien qu'en tenant compte des contraintes de précédence. Ainsi, le planning obtenu est identique à celui donné par la méthode CPM. De plus, il est forcément optimal car la méthode CPM construit des solutions optimales. □

Théorème 3 Si le problème *RCPSP* possède une solution optimale, alors il existe une (se)(PRMT) permettant d'obtenir une solution optimale.

Preuve 3 Kolisch [86], Sprecher et al. [130] montrent que si le problème *RCPSP* possède une solution optimale, alors il existe toujours une (s)(vp) qui représente une solution optimale. Il suffit d'appliquer l'algorithme ConstruireSEPRMT à cette (s)(vp) pour obtenir une (se)(PRMT) donnant la même solution. □

2.3.3 Avantages numériques

Pour mieux illustrer la puissance de notre nouvelle représentation, voici comment il est possible de l'exploiter dans plusieurs approches qui utilisent la transformation suivante : déplacer une activité de la (s)(vp) courante en l'insérant à une autre position de sorte que la nouvelle (s) reste (vp). Une nouvelle (s)(vp) est ainsi obtenue et le planning correspondant est calculé.

Selon nos résultats, dans ce type de transformations certains déplacements sont *inutiles* car ils engendrent le même planning. Voici un cas de figure qui l'illustre : nous avons considéré la première instance de 30 activités de la bibliothèque PSPLIB [93] (voir la section 1.7.2) pour laquelle nous avons construit une (s)(vp) grâce à la règle de priorité

[2](Critical Path Method), méthode permettant de donner une solution optimale au problème classique d'ordonnancement

MINSLK (voir la section 1.5.3) puis nous avons déduit, grâce à l'algorithme ConstruireSEPRMT, la (se)(PRMT) qui en découle et qui est illustrée à la Figure 2.4 (pour connaître la (s) construite avec la règle MINSLK, il suffit d'enlever les symboles "{" et "}" et garder l'ordre des activités).

[{1, 4*, 10, 16, 2, **9**, 11, 21, 26}{5, 15}{6}{3, 13}{18, 20, 25, 7, 8}{27, 28, 31, 19}
{29, 12, 14*}{17, 22, 23, 24, 30, 32}]

FIG. 2.4 – La (se) obtenue avec la première instance de 30 activités de PSPLIB

Dans cet exemple, supposons que l'activité 9 doit être déplacée. Pour compléter cette opération de sorte que la (s) reste (vp), il suffit de positionner l'activité 9 n'importe où, *après* l'activité 4 (*premier prédécesseur de 9, à sa gauche dans la séquence*) et *avant* l'activité 14 (*premier successeur de 9, à sa droite dans la séquence*). Ceci représente **22** possibilités. Avec la (se)(PRMT) construite, nous savons qu'il est *inutile* de déplacer l'activité 9 à l'intérieur de son ensemble, c'est-à-dire, *après l'activité* 4 *et avant l'activité* 5. Le nombre de possibilités est maintenant réduit à **16**, soit une diminution de **27%** des cas. De plus, chacune de ces 16 possibilités engendre un *nouveau planning*. Notons que la majorité des approches de ce type choisissent la nouvelle position de l'activité au hasard, et cette opération est répétée plusieurs fois. Par ailleurs, la construction d'un planning à partir d'une (s) est généralement l'opération la plus coûteuse [16] (en terme de temps de calcul). Nous avons donc intérêt à *maximiser les chances de tomber sur des séquences donnant lieu à des plannings différents.*

Des tests similaires ont été effectués sur la totalité des instances J30, J60 et J90 de PSPLIB. Pour chacune de ces instances :

1. 100 (s)(vp) différentes ont été générées aléatoirement selon la règle RAN (voir la section 1.5.3).
2. Pour chaque (s)(vp) générée, le test effectué plus haut en utilisant une activité sélectionnée aléatoirement, est répété 100 fois.
3. Pour chaque activité sélectionnée, deux types de *pourcentages de déplacements inutiles* sont évalués. Le premier concerne le déplacement de l'activité sélectionnée à l'*intérieur* de son ensemble (ce qui induit forcément le même planning). Le second est dû au déplacement de l'activité sélectionnée à l'*extérieur* de son ensemble *et induisant le même planning.* Ainsi, la somme de ces deux quantités donne le pour-

centage de *tous les déplacements possibles* de l'activité sélectionnée *engendrant le même planning* parmi tous les déplacements permettant à la (s) de rester (vp).

4. Enfin, une moyenne des pourcentages pour ces deux types de déplacements est calculée pour chacune des instances.

Rappelons que les instances J30, J60 et J90 de PSPLIB ont été générées grâce aux paramètres NC, RF et RS (voir la section 1.7.2). Pour chaque combinaison des valeurs de ces paramètres, un groupe de 10 instances a été généré dans PSPLIB. Nous avons, donc, évalué une moyenne des deux pourcentages de déplacements inutiles sur chacun de ces groupes. Les résultats de ces calculs sont résumés sur le Tableau 2.2. Les colonnes *J30-int*, *J60-int* et *J90-int* (respectivement *J30-ext*, *J60-ext* et *J90-ext*) donnent la moyenne du pourcentage de déplacements *inutiles* d'une activité à l'*intérieur* (respectivement à l'*extérieur*) de son ensemble pour J30, J60 et J90 respectivement et pour chaque combinaison des valeurs des paramètres NC, RF et RS. La colonne *Moy.-int* (respectivement *Moy.-ext*) donne la moyenne des valeurs dans les colonnes *J30-int*, *J60-int* et *J90-int* (respectivement *J30-ext*, *J60-ext* et *J90-ext*) pour chaque combinaison des paramètres. Pour simplifier l'analyse, nous avons trié nos résultats suivant les valeurs de *Moy.-int*.

Le Tableau 2.2 illustre l'efficacité des (se). *Le pourcentage des déplacements inutiles à l'intérieur d'un ensemble varie entre* 14.99% *et* 86.49%. La moyenne de ce pourcentage est de 48.22%, 46.61% et 46.33% pour les ensembles J30, J60 et J90 respectivement. Ainsi, *ni le pourcentage des déplacements inutiles à l'intérieur d'un ensemble pour chaque combinaison des paramètres, ni sa moyenne ne varient de façon significative avec la taille du problème.* Par aillleurs, *le pourcentage de déplacements inutiles à l'extérieur d'un ensemble varie entre* 0% *et* 39.45%. La moyenne de ce pourcentage est de 17.60%, 20.02% et 20.74% pour les ensembles J30, J60 et J90 respectivement. Ici aussi, *ni le pourcentage des déplacements inutiles à l'extérieur d'un ensemble pour chaque combinaison des paramètres, ni sa moyenne ne varient de façon significative avec la taille du problème.* Le Tableau 2.2 montre bien aussi que le déplacement d'une activité à l'extérieur de son ensemble *peut réduire de façon significative le risque d'engendrer le même planning* (selon les colonnes *J30-int*, *J60-int*, *J90-int* et *Moy.-int*). En effectuant ce type de déplacements, ce risque bien que réduit existe toujours et son importance varie dépendamment de l'instance considérée (selon les colonnes *J30-ext*, *J60-ext*, *J90-ext* et *Moy.-ext*). Notons que pour les cas où $RS = 1$, les ressources sont suffisament disponibles pour que toutes les activités soient planifiées juste après leurs prédécesseurs, ce qui veut dire que pour

ces instances, les (se) possèdent *un unique ensemble.* Ceci explique le pourcentage nul de déplacements inutiles à l'extérieur d'un ensemble dans les colonnes *J30-ext*, *J60-ext*, *J90-ext* et *Moy.-ext.*

Par ailleurs, dans le Tableau 2.2 il est difficile d'identifier un impact clair des paramètres NC et RF sur les pourcentages de déplacements inutiles. Par contre, *l'impact du paramètre RS sur les deux pourcentages est assez significatif.* Cela dit, le Tableau 2.3 permet d'évaluer de façon plus précise l'impact des 3 paramètres. Les colonnes *Pourc. int.* (respectivement *Pourc. ext.*) donnent la moyenne sur la colonne *Moy.-int.* (respectivement *Moy.-ext.*) du Tableau 2.2 pour chaque valeur des paramètres. Le Tableau 2.3 indique que *le pourcentage des déplacements inutiles à l'intérieur d'un ensemble augmente avec les valeurs des paramètres NC et RS et diminue avec les valeurs du paramètre RF.* Notons que ce pourcentage augmente de façon significative avec la valeur du paramètre RS. Par ailleurs, *le pourcentage des déplacements inutiles à l'extérieur d'un ensemble diminue avec les valeurs des paramètres NC, RF et RS.* Ici aussi, l'impact du paramètre RS sur ce pourcentage paraît plus important que celui des paramètres NC et RF.

Il semble que l'impact significatif du paramètre RS puisse s'expliquer par le fait que ce dernier mesure la relation entre la quantité des ressources nécessaires aux activités par rapport à leurs disponibilités. Ainsi, plus cette disponibilité augmente par rapport à la demande des activités, plus la taille des ensembles dans les (se) aura tendance à croître et plus le nombre de positions auxquels une activité peut être déplacée à l'extérieur de son ensemble aura tendance à diminuer.

Dans tous ce qui suit, lorsque nous faisons référence aux *déplacements inutiles* d'une activité, il sera question de déplacements à l'*intérieur* de son ensemble.

NC	RF	RS	J30-int	J30-ext	J60-int	J60-ext	J90-int	J90-ext	Moy.-int	Moy.-ext
2.10	1.00	1.00	86.49%	0.00%	82.82%	0.00%	80.98%	0.00%	83.43%	0.00%
2.10	0.25	1.00	85.64%	0.00%	82.31%	0.00%	81.03%	0.00%	82.99%	0.00%
2.10	0.50	1.00	85.95%	0.00%	81.74%	0.00%	81.25%	0.00%	82.98%	0.00%
2.10	0.75	1.00	85.30%	0.00%	82.12%	0.00%	81.39%	0.00%	82.94%	0.00%
1.80	1.00	1.00	82.62%	0.00%	79.92%	0.00%	80.54%	0.00%	81.03%	0.00%
1.80	0.75	1.00	81.97%	0.00%	80.74%	0.00%	80.20%	0.00%	80.97%	0.00%
1.80	0.50	1.00	81.41%	0.00%	80.05%	0.00%	80.64%	0.00%	80.70%	0.00%
1.80	0.25	1.00	81.65%	0.00%	79.93%	0.00%	80.13%	0.00%	80.57%	0.00%
1.50	0.50	1.00	78.54%	0.00%	78.36%	0.00%	78.57%	0.00%	78.49%	0.00%
1.50	0.75	1.00	79.68%	0.00%	77.87%	0.00%	77.28%	0.00%	78.28%	0.00%
1.50	1.00	1.00	78.41%	0.00%	77.73%	0.00%	78.33%	0.00%	78.16%	0.00%
1.50	0.25	1.00	77.23%	0.00%	78.49%	0.00%	76.96%	0.00%	77.56%	0.00%
2.10	0.25	0.70	59.71%	20.19%	53.03%	22.86%	54.06%	22.73%	55.60%	21.93%
1.80	0.25	0.70	54.36%	22.62%	52.54%	23.59%	49.11%	26.56%	52.01%	24.26%
2.10	0.50	0.70	53.47%	21.38%	50.28%	22.81%	50.33%	23.71%	51.36%	22.63%
2.10	0.25	0.50	59.15%	19.84%	45.38%	28.76%	45.66%	29.07%	50.06%	25.89%
2.10	0.75	0.70	45.51%	20.40%	49.41%	21.63%	48.88%	22.37%	47.94%	21.47%
2.10	1.00	0.70	46.62%	19.41%	48.29%	20.67%	48.47%	21.55%	47.79%	20.55%
1.80	0.50	0.70	47.14%	21.52%	49.67%	21.90%	46.23%	25.08%	47.68%	22.84%
1.80	1.00	0.70	43.11%	19.77%	47.04%	19.73%	48.44%	20.37%	46.19%	19.96%
1.50	0.25	0.70	45.99%	24.96%	45.55%	25.62%	46.54%	26.54%	46.03%	25.71%
1.80	0.75	0.70	45.82%	18.71%	45.54%	23.25%	46.26%	23.20%	45.87%	21.72%
1.80	0.25	0.50	51.38%	21.79%	42.92%	30.32%	41.83%	30.33%	45.38%	27.48%
1.50	1.00	0.70	40.86%	19.38%	44.35%	20.24%	44.80%	20.90%	43.34%	20.17%
1.50	0.50	0.70	43.22%	23.56%	42.97%	23.60%	42.82%	25.46%	43.01%	24.21%
1.50	0.75	0.70	39.23%	21.00%	42.47%	23.07%	42.98%	23.84%	41.56%	22.64%
2.10	0.50	0.50	46.54%	23.10%	38.29%	29.91%	38.31%	29.84%	41.05%	27.61%
1.50	0.25	0.50	41.80%	27.90%	37.82%	32.38%	36.00%	32.96%	38.54%	31.08%
2.10	0.75	0.50	40.15%	22.53%	36.17%	26.46%	38.41%	26.91%	38.25%	25.30%
1.80	0.50	0.50	40.34%	24.94%	36.26%	29.99%	35.05%	30.87%	37.22%	28.60%
2.10	1.00	0.50	35.98%	20.23%	35.82%	22.83%	37.18%	24.79%	36.33%	22.62%
1.80	1.00	0.50	36.35%	20.24%	36.64%	22.59%	35.91%	23.88%	36.30%	22.24%
1.80	0.75	0.50	36.79%	22.69%	33.97%	26.31%	36.09%	26.67%	35.62%	25.22%
2.10	0.25	0.20	40.68%	30.35%	34.66%	36.15%	30.79%	39.21%	35.37%	35.24%
1.50	0.75	0.50	32.76%	22.97%	32.90%	26.54%	33.41%	26.65%	33.02%	25.39%
1.80	0.25	0.20	40.36%	29.32%	30.85%	37.19%	27.83%	39.42%	33.01%	35.31%
1.50	0.50	0.50	32.55%	26.76%	33.37%	30.11%	32.74%	30.62%	32.88%	29.17%
1.50	1.00	0.50	30.89%	20.43%	33.57%	23.24%	33.58%	24.01%	32.68%	22.56%
1.50	0.25	0.20	31.08%	33.33%	27.30%	37.28%	23.98%	39.45%	27.45%	36.69%
2.10	0.50	0.20	25.72%	29.52%	22.31%	35.68%	22.33%	36.51%	23.45%	33.90%
1.80	0.50	0.20	24.71%	29.29%	21.92%	34.07%	20.66%	35.41%	22.43%	32.92%
1.50	0.50	0.20	20.05%	31.77%	19.42%	34.03%	18.97%	34.03%	19.48%	33.28%
1.80	0.75	0.20	17.58%	24.22%	18.52%	27.58%	19.21%	28.15%	18.43%	26.65%
2.10	0.75	0.20	17.07%	24.33%	18.64%	28.95%	18.05%	28.07%	17.92%	27.12%
2.10	1.00	0.20	16.34%	19.24%	16.96%	21.40%	18.10%	22.80%	17.14%	21.15%
1.80	1.00	0.20	16.03%	21.43%	17.07%	21.25%	18.18%	22.88%	17.09%	21.85%
1.50	0.75	0.20	14.99%	25.03%	17.25%	26.56%	17.51%	28.27%	16.58%	26.62%
1.50	1.00	0.20	15.43%	20.82%	16.28%	22.47%	18.03%	22.55%	16.58%	21.95%

TAB. 2.2 – Pourcentage des déplacements inutiles

NC	Pourc. int.	Pourc. ext.	RF	Pourc. int.	Pourc. ext.	RS	Pourc. int.	Pourc. ext.
1.5	43.98%	19.97%	0.25	52.05%	21.96%	0.2	22.08%	29.39%
1.8	47.53%	19.32%	0.50	46.73%	21.26%	0.5	38.11%	26.10%
2.1	49.66%	19.09%	0.75	44.78%	18.51%	0.7	47.36%	22.34%
			1	44.67%	16.09%	1	80.67%	0%

TAB. 2.3 – Moyennes des pourcentages des déplacements inutiles

2.3.4 Utilisation en mode Backward

Jusqu'à présent, nous avons utilisé Serial SGS avec le mode *forward* (voir la section 1.5.3) pour décoder une (s)(vp) en un planning. Par conséquent, la représentation en (se) a été introduite en considérant ce mode de planification. En fait, une (s)(vp) peut aussi être décodée avec Serial SGS mais avec le mode *backward*. Selon ce mode, l'activité i_{n+2} est la première à être planifiée. i_{n-j+3} est la j^{ime} activité à être planifiée *le plus tard possible* se sorte à satisfaire les contraintes de précédence ainsi que celles sur les ressources.

En se référant à l'exemple de la Figure 2.1, l'application de Serial SGS selon le mode backward décode la (s)(vp) suivante :

$$[1, 2, 5, 3, 6, 8, 4, 7, 9, 10]$$

pour générer le planning illustré à la Figure 2.5. Ce planning est différent de celui donné par le mode forward illustré à la Figure 2.6.

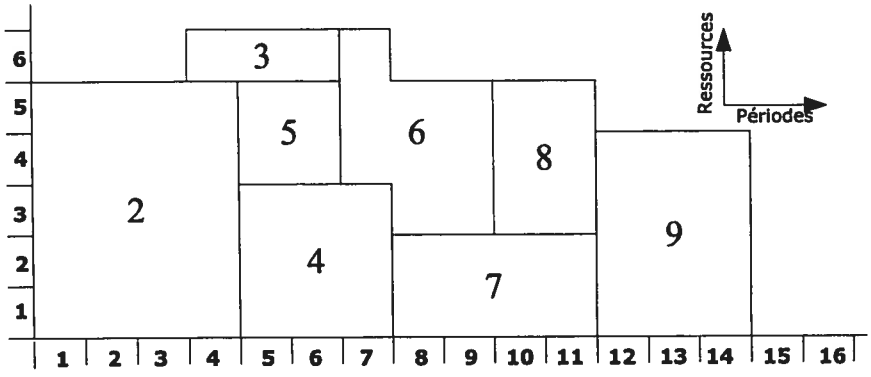

FIG. 2.5 – Planning selon le mode backward

Une extension de la propriété de permutation (PRMT) peut être faite pour le mode backward. Pour ce faire, la Définition 4 peut être reprise comme suit :

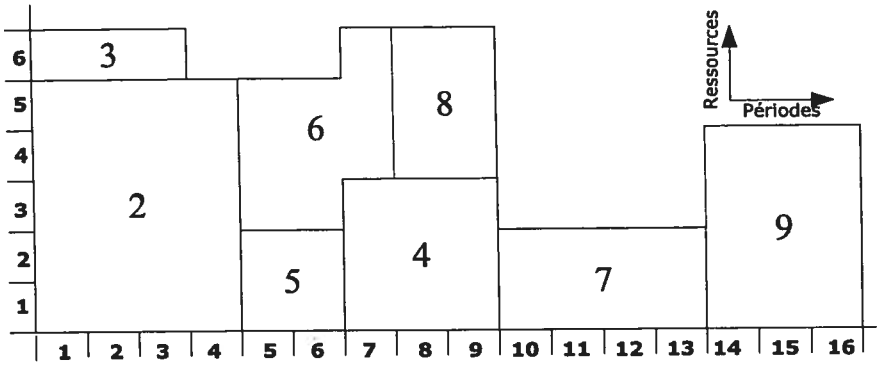

FIG. 2.6 – Planning selon le mode forward

Définition 5 Une (se) : $[A_1, A_2, \cdots, A_p]$ possède la propriété de permutation selon le mode backward **(PRMTb)** si elle est (vp) et $\forall(o_1, o_2, \cdots, o_p) \in O_1 \times O_2 \times \cdots \times O_p$, les (s) formées par $o_1, o_2, \cdots, o_p$ *correspondent toutes à un même planning lorsqu'elles sont décodées avec le mode backward.*

De façon similaire, la procédure ConstruireSEPRMT peut être facilement modifiée pour obtenir la procédure **ConstruireSEPRMTb** permettant de générer des (se) (PRMTb) en l'initisant avec n'importe quelle (s)(vp). Globalement, la procédure sélectionne les activités de i_{n+2} à i_1 pour les planifier à leur *date au plus tard.* Si la date de planification au plus tard $L_P(i_j)$ d'une activité sélectionnée i_j, en ne considérant que les contraintes de précédence, est égale à la date de planification au plus tard $L_{PR}(i_j)$, en considérant et les contraintes de précédence et celles sur les ressources, alors i_j est insérée dans l'ensemble en cours. Sinon, si $L_{PR}(i_j) < L_P(i_j)$, un nouvel ensemble est initialisé avec i_j. Une preuve similaire à celle du Théorème 1 peut être faite pour montrer que la (se) :

$$[A_l, A_{l-1}, \cdots, A_1]$$

donné par ConstruireSEPRMTb est bien (PRMTb).

À titre d'exemple, si la procédure ConstruireSEPRMTb est initiée par la (s)(vp) :

$$[1, 2, 5, 3, 6, 8, 4, 7, 9, 10]$$

la (se) obtenue sera :

$$[\{1, 2, 5\}\{3, 6, 8, 4, 7, 9, 10\}]$$

Notons que cette (se)(PRMTb) est différente de la (se)(PRMT) :

$$[\{1,2,5,3,6,8\}\{4,7,9,10\}]$$

donnée par ConstruireSEPRMT lorsqu'elle est initiée avec la même (s)(vp). Ainsi, il est possible de générer des plannings et des (se) différents selon le mode de planification choisi.

Le Théorème 2 reste valide pour le mode backward. Si une (se)(PRMTb) générée par ConstruireSEPRMTb comporte un unique ensemble, le planning optimal peut être obtenu par la méthode CPM.

Chapitre 3

Implémentation de procédures avec les séquences d'ensembles d'activités

3.1 Introduction

Pour illustrer l'efficacité de la représentation sous forme de séquence d'ensembles d'activités **(se)**, nous avons implémenté plusieurs variantes de deux des meilleures procédures heuristiques existantes. La première est basée sur le *recuit simulé* et proposée par Bouleimen et Lecoq [16]. La seconde est basée sur les *algorithmes génétiques* et proposée par Hartmann [66].

Plusieurs tests sont faits avec les instances de la bibliothèque PSPLIB [93] (voir la section 1.7.2). Toutes les procédures sont codées en C++ sous l'environnement Borland C++Builder 6.0 et implémentées sur un PC ayant un processeur Pentium 1.6 Ghz et 256 Mo de RAM avec le système d'exploitation Microsoft Windows XP.

Il est important de noter que notre objectif ici n'est pas d'obtenir de meilleurs résultats que ceux des plus performantes heuristiques pour le problème, mais plutôt de montrer *comment des heuristiques performantes peuvent être encore améliorées en y introduisant les séquences d'ensembles d'activités.*

3.2 Approche basée sur le recuit simulé

Comme dans la procédure de Bouleimen et Lecoq, nous utilisons la représentation sous forme de séquence d'activités **(s)(vp)**. La recherche est faite sur C *chaînes*. Chaque chaîne comporte S *étapes* et pour chaque étape $s \in \{1, \cdots, S\}$, N_s (s)(vp) sont générées

à partir de la (s)(vp) en cours. N_s est initialisé avec la valeur N_0 au début de chaque chaîne et sa valeur est progressivement augmentée d'une étape à une autre selon la règle $N_{s+1} = N_s(1 + (s \times h))$. Le paramètre h sert à moduler la longueur d'une étape. Après la fin de la dernière chaîne, le voisinage de la meilleure (s)(vp) trouvée est complètement exploré.

Dans notre procédure, et contrairement à celle de Bouleimen et Lecoq, au lieu de déplacer une activité à n'importe quelle position entre le dernier prédécesseur et le premier successeur dans la (s) pour générer un voisin, ce mouvement est restreint à être à *l'extérieur de l'ensemble incluant l'activité à déplacer* tel que décrit à la section 2.3.3. De plus, le critère d'arrêt de notre procédure est un nombre maximum $NbPlg$ de plannings ou de voisins générés. Pour certaines valeurs des paramètres C, S et h, N_0 est calculé comme étant la valeur minimale induisant un nombre de plannings supérieur ou égal à $NbPlg$ si toutes les chaînes et étapes sont complétées. Si ce nombre est supérieur à $NbPlg$, la procédure est interrompue lorsque $NbPlg$ plannings sont générés. La température initiale T_0 est calculée de sorte que le premier voisin du planning initial puisse être accepté avec la probabilité 0.01 s'il engendre une dégradation de 20% de la durée du planning. La procédure de résolution proposée est détaillée sur la Figure 3.1.

3.2.1 Expérimentations

Quatre différentes procédures ont été implémentées et testées :

1. RS_**SE** : notre procédure décrite à la Figure 3.1 où le voisinage est généré grâce aux (se).
2. RS_**S** : identique à la procédure RS_SE mais où le voisinage de Bouleimen et Lecoq est utilisé (voisinage généré grâce aux (s)(vp)).
3. RS_SE_**B** : identique à la procédure RS_SE mais où les plannings sont générés avec le mode **backward**.
4. RS_S_**B** : identique à la procédure RS_S mais où les plannings sont générés avec le mode **backward**.

Les quatre procédures utilisent la même approche de base résumée à la Figure 3.1. Les différences entre ces dernières résident dans la façon avec laquelle le voisinage est généré (soit avec les (se) ou avec les (s)) et le mode de planification utilisé (soit forward ou backward).

1. Générer la solution initiale x_0 (sous forme de (s)(vp)), selon la règle de priorité SIO (la priorité est donnée aux activités ayant les durées les plus courtes), ainsi que le planning qui lui correspond et qui sera de durée $f(x_0)$. Poser aussi $x_{Best} = x_0$ et $f_{Best} = f(x_0)$; meilleure solution rencontrée dans la procédure ainsi que la durée du planning qui lui correspond.
2. Soient $x_{EnCours} = x_0$ et $f_{EnCours} = f_0$, la solution en cours ainsi que la durée du planning qui lui correspond.
3. Les paramètres de calcul sont les suivants :
   - N_0 : Le nombre initial de voisins.
   - h : Un coefficient utilisé pour augmenter le nombre de voisins à générer d'une étape à une autre.
   - T_0 : La température initiale.
   - T : La température en cours.
   - α : Paramètre (entre 0 et 1) utilisé pour diminuer la température d'une étape à une autre.
   - C : Le nombre de chaînes.
   - S : Le nombre d'étapes à parcourir dans chaque chaîne.
4. Lire les valeurs des paramètres h, α, C, S et $NbPlg$.
5. Calculer N_0 comme étant la valeur minimale induisant un nombre de plannings supérieur ou égal à $NbPlg$ si toutes les chaînes et les étapes sont complétées.
6. Calculer T_0 de sorte que le voisin de x_0 puisse être accepté avec une probabilité 0.01 s'il engendre une dégradation de 20% de $f(x_0)$.
7. **Pour** C chaînes **faire**
   - $T = T_0$
   - $N_s = N_0$
   - **Pour** $s = 1, ..., S$ **faire**
     - $N_s = N_s(1 + (h \times s))$
     - **Pour** N_s voisins **faire**
       - **Si** le nombre de voisins générés depuis le début de la procédure est ($NbPlg$ - le nombre de voisins de x_{Best}) **alors aller à** l'étape 8.
       - Calculer un voisin x_v de $x_{EnCours}$.
       - Calculer $f(x_v)$, la durée du planning qui correspond à x_v.
       - Calculer $\Delta = f(x_v) - f(x_{EnCours})$.
       - **Si** $\Delta < 0$ **alors**
         - $x_{EnCours} = x_v$ et $f_{EnCours} = f(x_v)$.
         - **Si** $f(x_v) < f_{Best}$ **alors** $x_{Best} = x_v$ et $f_{Best} = f(x_v)$.
         - **Si** $f(x_v) =$ durée CPM **alors** terminer.
       - **Sinon si** $e^{(-\Delta/T)} >$ une valeur tirée au hasard entre 0 et 1 **alors**
         - $x_{EnCours} = x_v$ et $f_{EnCours} = f(x_v)$.
     - **Fin pour** N_s
     - $T = \alpha T$
   - **Fin pour** s
   - **Fin pour** C
8. Explorer le voisinage de de la meilleure solution x_{Best} .

FIG. 3.1 – Schéma global des différentes variante de la procédure de résolution

Les tests suivants permettent d'évaluer l'impact des (se) sur la performance des procédures RS_S et RS_S_B. Au total, quatre tests sont effectués sur la bibliothèque PSPLIB. Ces tests sont destinés à évaluer l'impact de l'utilisation des (se) sur *la qualité de la solution obtenue* (les trois premiers tests) et sur le *nombre de solutions qu'il a fallu générer avant d'atteindre la meilleure solution* (le quatrième test). Notons que dans la littérature sur le sujet, la qualité d'une solution est généralement évaluée par la déviation de sa durée par rapport à celle du planning généré avec la méthode CPM pour les instances J60, J90 et J120 et par rapport à la durée optimale pour les instances de J30.

- **Test 1** : RS_SE, RS_S, RS_SE_B, RS_S_B sont testées et comparées avec les instances J30, J60, J90 et J120 en utilisant toutes les combinaisons des valeurs suivantes des paramètres : $C \in \{1, 2, 5\}, h \in \{0, 1, 2\}$ et $S \in \{5\}$. Le nombre de plannings $NbPlg$ générés est fixé à 5000. Ces valeurs ont également été utilisées dans [16] pour des tests sur J30.
- **Test 2** : RS_SE et RS_S sont testées et comparées avec les valeurs suivantes pour les paramètres : $C = 1, h = 1$ et $S = 5$. Ce choix des paramètres est motivé par les bons résultats qu'ils ont permis d'obtenir dans le Test 1. Ici, le nombre de plannings $NbPlg$ générés est fixé à 1000 fois le nombre d'activités dans le but de compléter une exploration du domaine des solutions qui soit proportionnelle à la taille de l'instance.
- **Test 3** : En considérant les résultats du Test 1, RS_SE et RS_S sont comparées pour chaque combinaison des valeurs de NC, RF et RS (voir la section 1.7.2) afin de mesurer l'impact des (se) selon le degré de complexité des instances. Pour cela, nous procédons comme suit :

  1. Pour chaque ensemble d'instances J30, J60, J90 et J120, nous considérons le groupe des 10 instances générées avec les mêmes valeurs de NC, RF et RS.
  2. Pour chacun de ces groupes, nous calculons la moyenne du pourcentage de déviation pour chacune des 9 combinaisons des valeurs : $C \in \{1, 2, 5\}, h \in \{0, 1, 2\}$ et $S \in \{5\}$. Ainsi, pour chaque instance, 9 moyennes du pourcentage de déviation sont obtenues pour chaque combinaison des valeurs de NC, RF et RS.
  3. Une *moyenne globale du pourcentage de déviation* est calculée sur ces 9 moyennes pour chacune des 48 (respectivement 60) combinaisons des valeurs de NC, RF et RS pour les ensembles J30, J60 et J90 (respectivement pour J120).

4. Enfin, la différence entre les moyennes globales des pourcentages de déviation obtenus avec RS_SE et RS_S (amélioration apportée par RS_SE par rapport à RS_S) est calculée pour chacune des 48 combinaisons des paramètres.

– **Test 4** : Ce test est similaire au Test 3 sauf que *la moyenne du nombre de plannings générés avant d'atteindre la meilleure solution* est calculée au lieu de la moyenne du pourcentage de déviation.

Dans tous ces tests, $\alpha = 0.25$ comme suggéré dans [16].

3.2.2 Résultats numériques et analyses

Les résultats du Test 1 sont résumés dans les Tableaux 3.1, 3.2, 3.3 et 3.4. Les 4 valeurs, dans chaque cellule de ces tableaux, correspondent aux méthodes de résolution suivantes :

RS_SE (RS_S)
RS_SE_B (RS_S_B)

Ces valeurs donnent la *moyenne du pourcentage de déviation* (*Moy. Dev.*) ou le pourcentage de *déviation maximum* (*Max. Dev.*) sur toutes les instances et pour chaque combinaison des valeurs de C et h (rappelons que S prend uniquement la valeur 5). Aussi, pour chaque procédure, nous indiquons en caractère gras la meilleure moyenne du pourcentage de déviation pour chaque ensemble d'instances.

	C=1		C=2		C=5	
	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.
h=0	0.19 (**0.21**)	4.76 (5.43)	0.17 (**0.21**)	4.41 (5.26)	0.19 (0.25)	4.69 (5.17)
	0.18 (0.22)	5.17 (5.45)	0.21 (**0.20**)	7.89 (5.26)	0.17 (0.21)	6.58 (3.95)
h=1	**0.16** (0.24)	5.17 (5.45)	0.21 (0.23)	5.17 (5.17)	0.25 (0.26)	5.17 (5.97)
	0.19 (0.22)	5.45 (5.26)	**0.16** (0.21)	5.17 (5.19)	0.27 (0.28)	7.84 (6.90)
h=2	0.30 (0.37)	5.26 (6.78)	0.30 (0.37)	5.26 (6.78)	0.30 (0.37)	5.26 (6.78)
	0.30 (0.35)	5.17 (6.58)	0.30 (0.35)	5.17 (6.58)	0.30 (0.35)	5.17 (6.58)

TAB. 3.1 – **Test 1** : Résultats pour **J30** avec 5000 plannings générés

Les Tableaux 3.1, 3.2, 3.3 et 3.4 montrent bien la supériorité de la version des procédures qui utilisent les (se) par rapport à celles qui utilisent les (s) en considérant la moyenne du pourcentage de déviation. En effet, seulement dans un unique cas de figure

	C=1		C=2		C=5	
	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.
h=0	12.21 (12.33)	107.79 (111.69)	12.24 (12.39)	114.29 (118.18)	12.39 (12.52)	112.99 (111.69)
	12.19 (12.40)	114.29 (116.88)	12.20 (12.36)	115.58 (114.29)	12.43 (12.56)	116.88 (115.58)
h=1	**12.04 (12.24)**	109.09 (112.99)	12.40 (12.40)	112.99 (118.18)	12.61 (12.66)	114.29 (115.58)
	12.17 (12.24)	118.18 (118.18)	12.41 (12.43)	118.18 (116.88)	12.71 (12.84)	118.18 (120.78)
h=2	12.74 (12.93)	120.78 (118.18)	12.74 (12.93)	120.78 (118.18)	12.74 (12.93)	120.78 (118.18)
	12.74 (12.77)	118.18 (122.08)	12.74 (12.77)	118.18 (122.08)	12.74 (12.77)	118.18 (122.08)

TAB. 3.2 – **Test 1** : Résultats pour **J60** avec 5000 plannings générés

	C=1		C=2		C=5	
	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.
h=0	**12.51** (12.77)	116.95 (116.95)	12.63 (12.85)	116.95 (120.34)	12.90 (12.91)	116.95 (120.34)
	12.47 (12.64)	116.86 (113.56)	12.68 (12.77)	118.64 (113.56)	12.84 (13.04)	116.95 (118.64)
h=1	12.59 (**12.60**)	116.95 (113.56)	12.68 (12.75)	116.95 (116.95)	13.24 (13.28)	118.64 (122.03)
	12.48 (12.65)	116.95 (113.56)	12.76 (12.85)	116.95 (118.64)	13.36 (13.47)	118.64 (118.64)
h=2	13.23 (13.27)	122.03 (122.03)	13.23 (13.27)	122.03 (122.03)	13.23 (13.27)	122.03 (122.03)
	13.06 (13.22)	120.34 (116.95)	13.06 (13.22)	120.34 (116.95)	13.06 (13.22)	120.34 (116.95)

TAB. 3.3 – **Test 1** : Résultats pour **J90** avec 5000 plannings générés

	C=1		C=2		C=5	
	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.	Moy. Dev.	Max. Dev.
h=0	41.31 (42.10)	225.35 (218.31)	**41.03 (41.92)**	220.20 (218.18)	41.56 (42.59)	223.23 (218.31)
	40.98 (41.53)	222.54 (218.81)	40.44 (41.00)	223.23 (223.23)	41.04 (41.55)	220.79 (223.76)
h=1	41.06 (42.07)	218.18 (221.21)	41.46 (42.15)	223.23 (223.23)	42.15 (42.81)	228.28 (225.25)
	40.33 (40.59)	220.79 (220.20)	40.72 (41.28)	224.75 (222.77)	41.75 (42.05)	229.29 (225.25)
h=2	42.12 (42.88)	224.75 (225.25)	42.12 (42.88)	224.75 (225.25)	42.12 (42.88)	224.75 (225.25)
	40.89 (41.20)	222.22 (219.80)	40.89 (41.20)	222.22 (219.80)	40.89 (41.20)	222.22 (219.80)

TAB. 3.4 – **Test 1** : Résultats pour **J120** avec 5000 plannings générés

($h = 1$, $C = 2$) dans le Tableau 3.2, RS_SE donne une moyenne du pourcentage de déviation égale à celle de RS_S. Aussi, dans un autre cas ($h = 0$, $C = 2$) dans le Tableau 3.1, RS_S_B donne une meilleure moyenne du pourcentage de déviation par rapport à RS_SE_B mais avec une différence de 0.01% seulement. *Sur les 70 autres cas, RS_SE et RS_SE_B donnent de meilleurs résultats que RS_S et RS_S_B* respectivement.

Les résultats du Test 2, comparant RS_SE et RS_S sur les instances J30, J60, J90 et J120 selon différents critères d'arrêt (*NbPlg* égal à 1000 × 30, 1000 × 60, 1000 × 90, 1000 × 120 pour J30, J60, J90 et J120, respectivement), sont résumés sur le Tableau 3.5.

	Déviation		Nombre de plannings		Temps de calcul (Sec)	
	Moy	Max	Moy	Amélioration	Moy	Max
J30	0.07 (0.07)	6.67 (2.67)	1149.56 (1485.09)	22.59%	3.34 (3,33)	8 (9)
J60	11.14 (11.19)	106.49 (107.79)	5906.59 (6938.20)	14.87%	8.91 (8.69)	34 (33)
J90	10.83 (10.89)	106.78 (106.78)	10898.63 (11356.80)	4.03%	15.91 (15.91)	63 (63)
J120	33.90 (34.01)	204.04 (203.03)	38141.16 (40575.16)	5.99%	59.50 (59.71)	115 (113)

TAB. 3.5 – **Test 2** : Comparaison de RS_SE et RS_S

En plus de *la moyenne et le maximum du pourcentage de déviation* (deuxième colonne), le Tableau 3.5 comporte aussi le *nombre moyen de plannings générés* par RS_SE et RS_S (qui est entre parenthèses) *avant d'atteindre la meilleure solution* ainsi que le *pourcentage d'amélioration qu'apporte RS_SE par rapport à RS_S en tenant compte de ce nombre* (troisième colonne). Finalement, la dernière colonne indique *la moyenne et le maximum du temps de calcul (en secondes) pour générer NbPlg plannings ou avant d'atteindre la valeur CPM.*

En général, le temps de calcul de RS_SE est légèrement supérieur à celui de RS_S sauf pour J90 où il y a égalité et pour J120 où RS_SE donne un meilleur résultat sur la moyenne. Cette différence est dû à l'effort de calcul supplémentaire fait dans RS_SE pour garantir qu'une activité *soit déplacée à l'extérieur de son ensemble.* Les résultats sur le Tableau 3.5 indiquent également *la supériorité de RS_SE par rapport RS_S au niveau de la moyenne du pourcentage de déviation (sauf pour J30 ou il y a égalité) et du nombre de plannings générés avant d'atteindre la meilleure solution.*

Les résultats du Test 3 sont résumés dans les Tableaux 3.6 et 3.7. Les trois premières colonnes du Tableau 3.6 regroupent les valeurs de NC, RF et RS. Les trois colonnes suivantes indiquent, respectivement, le pourcentage de la déviation moyenne globale de

RS_S, celui de RS_SE et la différence entre ces deux valeurs (amélioration) pour les instances J30. Lorsque cette différence est positive (respectivement négative) RS_SE donne de meilleurs résultats (respectivement de moins bons résultats) que ceux de RS_S. Les colonnes restantes indiquent les mêmes résultats pour J60 et J90. Le Tableau 3.7 possède la même structure et regroupe les résultats de J120.

Le Test 3 *n'indique aucun impact clair des paramètres NC, RF et RS sur l'amélioration apportée par RS_SE.* Cependant, une analyse plus approfondie de ces résultats permet de construire le Tableau 3.8 qui donne le *pourcentage du nombre de combinaisons des valeurs de NC, RF et RS où RS_SE donne de meilleurs, les mêmes ou de moins bons résultats que RS_S* (les lignes *% Meils*, *% Égales* et *% Pires*, respectivement). Pour les cas où RS_SE donne de meilleurs résultats, nous calculons la moyenne et le maximum du pourcentage d'amélioration (*Moy. Dev. Meil* et *Max. Dev. Meil*, respectivement). De façon similaire, nous calculons la moyenne *Moy. Dev. Pire* et le maximum *Max. Dev. Pire* lorsque RS_SE donne de moins bons résultats.

Pour l'ensemble des instances J30, J60 et J90, les résultats du Tableau 3.8 indiquent que RS_SE et RS_S génèrent des résultats similaires pour presque la moitié des combinaisons des valeurs de *NC*, *RF* et *RS*. Par ailleurs, *le pourcentage du nombre de combinaisons où RS_SE donne de meilleurs résultats que RS_S est significativement plus important que ce même nombre lorsque le contraire se produit.* De plus, *le pourcentage d'amélioration lorsque RS_SE génère de meilleurs résultats est plus important que ce pourcentage lorsque RS_S génère de meilleurs résultats.* En ce qui concerne les instances J120, *cette supériorité est même plus significative.*

Par ailleurs, nous avons effectué le *test de Wilcoxon pour échantillons appariés* [141] sur les valeurs des colonnes *(s)* et *(se)* des Tableau 3.6 et 3.7 pour chaque ensemble J30, J60, J90 et J120 afin de constater si l'impact des (se) est statistiquement significatif. Tous ces tests se sont révélés *significatifs avec un niveau de confiance égal à* 5%.

Les Tableaux 3.9, 3.10, 3.11, 3.12 indiquent les résultats du Test 4. Ces derniers donnent les moyennes de l'amélioration sur le nombre de plannings générés avant d'atteindre la meilleure solution pour chaque valeur des paramètres *NC*, *RF* et *RS* par rapport à chaque ensemble d'instances J30, J60, J90 et J120.

Les résultats numériques du Test 4 *n'indiquent pas non plus un impact clair des paramètres NC, RF et RS* sur le nombre de plannings générés avant d'atteindre la meilleure solution. Cependant, nous pouvons voir que *la différence entre les nombres de plannings générés avant d'atteindre la meilleure solution entre RS_SE et RS_S augmente avec la*

NC	RF	RS	J30			J60			J90		
			(s)	(se)	Amél	(s)	(se)	Amél	(s)	(se)	Amél
1.50	0.25	0.20	0.00	0.00	0.00	10.92	10.74	0.18	13.97	13.99	-0.02
1.50	0.25	0.50	0.00	0.00	0.00	2.41	2.41	0.00	0.00	0.00	0.00
1.50	0.25	0.70	0.00	0.00	0.00	1.98	2.02	-0.04	0.00	0.00	0.00
1.50	0.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.50	0.20	0.55	0.27	0.28	35.60	35.47	0.13	38.85	38.67	0.18
1.50	0.50	0.50	0.00	0.00	0.00	1.79	1.51	0.28	2.04	1.91	0.13
1.50	0.50	0.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.50	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.75	0.20	1.32	0.97	0.35	51.41	50.93	0.49	64.02	63.32	0.70
1.50	0.75	0.50	0.67	0.54	0.13	0.90	0.87	0.04	0.67	0.48	0.19
1.50	0.75	0.70	0.22	0.00	0.22	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.75	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	1.00	0.20	2.21	2.16	0.05	79.49	78.96	0.54	74.70	74.86	-0.16
1.50	1.00	0.50	1.09	1.00	0.09	3.03	2.89	0.14	0.82	0.75	0.07
1.50	1.00	0.70	0.04	0.00	0.04	0.00	0.00	0.00	0.00	0.00	0.00
1.50	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	0.25	0.20	0.00	0.00	0.00	10.71	10.53	0.18	12.05	11.70	0.35
1.80	0.25	0.50	0.00	0.00	0.00	0.85	0.81	0.03	0.00	0.00	0.00
1.80	0.25	0.70	0.00	0.00	0.00	1.30	1.30	0.00	0.00	0.00	0.00
1.80	0.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	0.50	0.20	0.54	0.39	0.15	39.49	38.83	0.66	42.01	42.43	-0.42
1.80	0.50	0.50	0.00	0.00	0.00	2.42	2.20	0.22	1.85	1.56	0.28
1.80	0.50	0.70	0.00	0.04	-0.04	0.40	0.40	0.00	0.00	0.00	0.00
1.80	0.50	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	0.75	0.20	1.86	1.47	0.39	57.50	57.52	-0.02	64.98	65.66	-0.68
1.80	0.75	0.50	0.36	0.30	0.06	3.60	3.47	0.13	4.69	4.39	0.30
1.80	0.75	0.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	0.75	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	1.00	0.20	1.65	1.56	0.10	78.09	77.53	0.56	73.13	72.92	0.22
1.80	1.00	0.50	0.37	0.39	-0.02	6.44	6.28	0.16	4.75	4.52	0.23
1.80	1.00	0.70	0.00	0.02	-0.02	0.00	0.00	0.00	0.26	0.11	0.14
1.80	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	0.25	0.20	0.07	0.00	0.07	10.90	10.95	-0.05	9.75	9.42	0.33
2.10	0.25	0.50	0.00	0.00	0.00	2.33	2.35	-0.02	1.49	1.50	-0.01
2.10	0.25	0.70	0.00	0.00	0.00	1.32	1.32	0.00	0.10	0.10	0.00
2.10	0.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	0.50	0.20	0.05	0.00	0.05	39.56	38.75	0.82	44.22	43.48	0.73
2.10	0.50	0.50	0.00	0.03	-0.03	2.81	2.66	0.16	2.57	2.08	0.49
2.10	0.50	0.70	0.00	0.00	0.00	0.12	0.12	0.00	0.00	0.00	0.00
2.10	0.50	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	0.75	0.20	0.83	0.69	0.13	62.82	62.50	0.32	68.78	68.11	0.67
2.10	0.75	0.50	0.30	0.23	0.07	4.86	4.68	0.19	5.29	5.19	0.10
2.10	0.75	0.70	0.06	0.02	0.04	0.29	0.27	0.02	0.00	0.00	0.00
2.10	0.75	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	1.00	0.20	0.49	0.46	0.03	81.95	80.61	1.34	84.80	84.88	-0.08
2.10	1.00	0.50	0.55	0.43	0.11	9.10	8.97	0.12	8.06	7.83	0.23
2.10	1.00	0.70	0.13	0.02	0.10	0.03	0.00	0.03	0.03	0.00	0.03
2.10	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TAB. 3.6 – **Test 3** : Déviations par combinaison de paramètres - **J30**, **J60** et **J60**

NC	RF	RS	J120		
			(s)	(se)	Amél.
1.5	0.25	0.1	37.73%	37.41%	0.32%
1.5	0.25	0.2	24.35%	22.84%	1.51%
1.5	0.25	0.3	4.42%	3.61%	0.81%
1.5	0.25	0.4	3.85%	2.12%	1.73%
1.5	0.25	0.5	1.05%	0.20%	0.85%
1.5	0.5	0.1	107.42%	106.56%	0.86%
1.5	0.5	0.2	52.66%	51.59%	1.07%
1.5	0.5	0.3	26.76%	24.52%	2.25%
1.5	0.5	0.4	8.21%	6.90%	1.31%
1.5	0.5	0.5	2.72%	1.58%	1.14%
1.5	0.75	0.1	141.27%	139.71%	1.56%
1.5	0.75	0.2	75.33%	74.32%	1.01%
1.5	0.75	0.3	30.50%	30.01%	0.50%
1.5	0.75	0.4	16.30%	14.96%	1.34%
1.5	0.75	0.5	3.27%	2.18%	1.10%
1.5	1	0.1	177.38%	177.18%	0.20%
1.5	1	0.2	79.58%	78.74%	0.84%
1.5	1	0.3	40.35%	39.47%	0.89%
1.5	1	0.4	21.21%	19.52%	1.69%
1.5	1	0.5	6.46%	4.54%	1.91%
1.8	0.25	0.1	31.64%	31.49%	0.15%
1.8	0.25	0.2	18.29%	17.16%	1.13%
1.8	0.25	0.3	2.84%	1.79%	1.05%
1.8	0.25	0.4	2.17%	1.78%	0.39%
1.8	0.25	0.5	1.62%	1.48%	0.14%
1.8	0.5	0.1	93.72%	92.92%	0.80%
1.8	0.5	0.2	50.33%	49.35%	0.99%
1.8	0.5	0.3	17.19%	16.12%	1.07%
1.8	0.5	0.4	9.11%	7.77%	1.35%
1.8	0.5	0.5	3.62%	2.78%	0.84%

NC	RF	RS	J120		
			(s)	(se)	Amél.
1.8	0.75	0.1	142.77%	141.16%	1.61%
1.8	0.75	0.2	58.34%	58.04%	0.30%
1.8	0.75	0.3	34.94%	34.70%	0.24%
1.8	0.75	0.4	18.34%	17.40%	0.94%
1.8	0.75	0.5	6.83%	5.76%	1.07%
1.8	1	0.1	159.14%	159.22%	-0.09%
1.8	1	0.2	86.06%	85.66%	0.40%
1.8	1	0.3	38.11%	37.75%	0.36%
1.8	1	0.4	20.52%	19.19%	1.33%
1.8	1	0.5	4.38%	3.44%	0.93%
2.1	0.25	0.1	31.88%	31.56%	0.32%
2.1	0.25	0.2	17.83%	17.63%	0.20%
2.1	0.25	0.3	7.83%	6.92%	0.91%
2.1	0.25	0.4	4.00%	2.60%	1.40%
2.1	0.25	0.5	0.46%	0.27%	0.18%
2.1	0.5	0.1	96.06%	95.07%	0.99%
2.1	0.5	0.2	51.00%	50.54%	0.46%
2.1	0.5	0.3	24.89%	24.96%	-0.07%
2.1	0.5	0.4	12.23%	11.47%	0.75%
2.1	0.5	0.5	3.44%	3.17%	0.27%
2.1	0.75	0.1	156.94%	155.90%	1.05%
2.1	0.75	0.2	75.32%	74.66%	0.66%
2.1	0.75	0.3	34.12%	33.34%	0.78%
2.1	0.75	0.4	18.19%	17.87%	0.32%
2.1	0.75	0.5	5.54%	4.35%	1.19%
2.1	1	0.1	184.98%	184.80%	0.18%
2.1	1	0.2	83.92%	83.55%	0.37%
2.1	1	0.3	45.59%	45.33%	0.26%
2.1	1	0.4	22.84%	22.31%	0.54%
2.1	1	0.5	10.85%	10.35%	0.50%

TAB. 3.7 – **Test 3** : Déviations par combinaison de paramètres - **J120**

	J30	J60	J90	J120
% Meils	39.58%	45.83%	37.50%	96.66%
% Pires	8.33%	8.33%	12.50%	3.33%
% Égales	52.08%	45.83%	50%	0%
Moy. Dev. Meil	0.13%	0.31%	0.30%	0.85%
Moy. Dev. Pire	-0.03%	-0.03%	-0.23%	-0.08%
Max. Dev. Meil	0.39%	1.34%	0.73%	2.25%
Max. Dev. Pire	-0.04%	-0.05%	-0.68%	-0.09%

TAB. 3.8 – **Test 3** : Comparaison globale des résultats de RS_SE et RS_S

taille du problème. En effet, les valeurs de la moyenne de cette différence sont de 84.30, 97.67, 102.81 et 176.28 pour J30, J60, J90 et J120 respectivement. Ceci indique clairement que *l'impact de l'utilisation des (se) sur le nombre de plannings générés avant d'atteindre la meilleure solution augmente avec la taille du problème.*

Pour résumer, les résultats des Tests 3 et 4 ne permettent pas d'établir un lien clair des valeurs de NC, RF et RS avec l'amélioration de la qualité de la solution ou avec le nombre de plannings générés avant d'atteindre la meilleure solution lorsque les (se) sont utilisées. Cependant, ces résultats permettent d'affirmer l'avantage de l'utilisation des (se) et, qu'en général, leur impact augmente avec la taille du problème.

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	90.14	0.25	39.43	0.2	17
1.8	43.05	0.5	64.10	0.5	177.69
2.1	119.72	0.75	98.24	0.7	142.52
		1	135.43	1	0

TAB. 3.9 – **Test 4** : Évolution de l'amélioration sur le nombre de plannings pour **J30**

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	97.87	0.25	108.79	0.2	168.61
1.8	137.71	0.5	67.54	0.5	114.35
2.1	57.44	0.75	79.01	0.7	107.74
		1	135,36	1	0

TAB. 3.10 – **Test 4** : Évolution de l'amélioration sur le nombre de plannings pour **J60**

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	73.47	0.25	127.26	0.2	29.85
1.8	44.54	0.5	49.76	0.5	212.75
2.1	190.44	0.75	153.93	0.7	168.68
		1	80.31	1	0

TAB. 3.11 – **Test 4** : Évolution de l'amélioration sur le nombre de plannings pour **J90**

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	227.47	0.25	103.02	0.1	119.37
1.8	208.08	0.5	190.13	0.2	51.49
2.1	93.29	0.75	282.40	0.3	183.81
		1	129.57	0.4	398.19
				0.5	128.54

TAB. 3.12 – **Test 4** : Évolution de l'amélioration sur le nombre de plannings pour **J120**

3.3 Approche basée sur les algorithmes génétiques

La variante de l'algorithme génétique considérée peut être résumée comme suit. La procédure est initialisée avec une population d'individus (solutions) de taille *POP paire*. Ces individus (*parents*) sont regroupés en couples, de façon aléatoire, et l'opérateur de *croisement* est appliqué à chacun de ces couples pour générer deux individus (*enfants*). L'opérateur de *mutation* est par la suite appliqué à chaque enfant avec une certaine probabilité. Enfin, une stratégie de *sélection* est utilisée pour choisir les *POP* individus, parmi les *POP* parents et les *POP* enfants, pour construire la population de la prochaine génération. La procédure est ainsi répétée jusqu'à ce qu'un certain nombre de générations soient parcourues.

Nous commençons par décrire la procédure génétique de Hartmann [66], notée **AG_S**, et nous montrons comment tirer avantage de la représentation en (se) pour obtenir la procédure **AG_SE**. Hartmann propose plusieurs versions de son algorithme. Pour nos tests, nous considérons seulement la version la plus performante telle que rapportée dans [66]. Les étapes communes des procédures AG_SE et AG_S sont résumées dans la Figure 3.2.

La différence entre AG_SE et AG_S réside dans la façon avec laquelle *les points de croisement q_1 et q_2 sont choisis* (étape 3a en caractère gras dans la Figure 3.2). Dans

1. **Représentation des solutions** : elle est faite sous forme de $(s)(vp)$.
2. **Population initiale** : POP $(s)(vp)$ sont générées. Chacune d'elle est construite de façon aléatoire activité par activité. Une activité est sélectionnée parmi celles dont les prédécesseurs ont déjà été sélectionnés. La probabilité de sélectionner une activité est proportionnelle à sa date de fin au plus tard.
3. **Croisement** : Toutes les $(s)(vp)$ sont regroupées en couples. L'opérateur de croisement est appliqué sur chaque couple de $(s)(vp)$, P et M, pour générer les enfants, G et F, comme suit :
   (a) **Deux points de croisement q_1 et q_2 sont choisis aléatoirement avec** $1 \leq q_1 < q_2 \leq (N+2)$.
   (b) Pour générer l'enfant F (respectivement G), les activités aux positions $1, \cdots, q_1$ sont les mêmes que dans M (respectivement P). Les positions $q_1+1, \cdots, q_2$ sont occupées par les $q_2 - q_1$ premières activités dans P (respectivement M), qui n'ont pas été sélectionnées, selon leur ordre dans P (respectivement M). Le reste des activités sont placées aux positions $q_2+1, \cdots, N+2$ selon leur ordre dans M (respectivement P).
4. **Mutation** : Toutes les $(s)(vp)$ générées par l'opérateur de croisement subissent une mutation de la façon suivante :
   (a) Sélectionner les activités selon leur ordre dans la $(s)(vp)$.
   (b) L'activité à la position i est permutée avec l'activité à la position $(i+1)$ avec une probabilité p_{mut}.
   (c) La permutation est faite seulement si la (s) résultante reste (vp).
5. **Sélection** : Les $(s)(vp)$ enfants obtenus sont ajoutés aux parents. Les meilleures POP $(s)(vp)$ (en terme de leur durée) sont sélectionnées pour former la prochaine génération.

FIG. 3.2 – Les étapes principales des procédures génétiques AG_SE et AG_S

AG_S, ces points sont choisis de façon aléatoire alors que dans AG_SE ce choix est fait de sorte à *réduire le risque d'engendrer un enfant correspondant au même planning que celui de l'un de ses parents.*

Pour illustrer la sélection de q_1 et q_2 dans AG_SE, considérons les deux (s)(vp) suivantes en se référant au problème illustré sur la Figure 2.1.

$$M_1 = [1, 3, 6, 2, 5, 4, 7, 8, 9, 10]$$

$$P_1 = [1, 3, 2, 6, 4, 5, 7, 8, 9, 10]$$

Supposons que les points $q_1 = 2$ et $q_2 = 3$ ont été sélectionnés aléatoirement pour effectuer un croisement. Ainsi, les enfants générés sont les suivants :

$$F_1 = [1, 3, 2, 6, 5, 4, 7, 8, 9, 10]$$

$$G_1 = [1, 3, 6, 2, 4, 5, 7, 8, 9, 10]$$

Nous pouvons facilement vérifier qu'en fait, G_1 peut être déduite de P_1 rien qu'en permutant les positions des activités 2 et 6. À présent, étant donné que la (se)(vp) générée à partir de P_1 est :

$$[\{1, 3, 2, 6, 4\}\{5, 7, 8, 9, 10\}]$$

G_1 et P_1 *correspondent au même planning puisque les activités* 2 *et* 6 *appartiennent au même ensemble.*

Dans la procédure AG_SE, nous tentons d'éviter cette situation de la façon suivante :

1. Nous choisissons d'abord une valeur pour q_1 puis pour q_2 telle que $q_2 > q_1$. À présent, supposons que les activités aux positions $q_1 + 1, \cdots, N + 2$ appartiennent au dernier ensemble de la (se) qui correspond à un des parents. Ainsi, l'un des enfants sera similaire à ce parent pour les positions $1, \cdots, q_1$. Le reste des positions de l'enfant seront occupées par les activités de ce parent se trouvant aux positions $q_1 + 1, \cdots, N + 2$ mais possiblement avec un ordre différent. Comme ces activités appartiennent à un même ensemble, l'enfant correspondra au même planning que celui du parent en question.

   Pour éviter cette situation, nous allons restreindre le choix de q_1 de sorte que les activités aux positions $q_1 + 1, \cdots, N + 2$ de chacun des parents *n'appartiennent pas toutes au dernier ensemble.* Donc, si les nombres d'activités, dans les derniers

ensembles des (se) correspondant aux deux parents, sont nb_1 et nb_2 alors le choix de q_1 se fera *au hasard* tel que :

$$q_1 < min(N+2-nb_1, N+2-nb_2)$$

Rappelons que si un parent possède un unique ensemble, il correspond à une solution optimale (voir le Théorème 2 à la section 2.3.2).

2. Après le choix de q_1, nous pouvons également restreindre le choix de q_2 pour réduire encore le risque d'engendrer un même planning. Comme déjà vu dans l'exemple de croisement précédent, bien que les activités aux positions $q_1+1, \cdots, N+2$ ne soient pas toutes dans le dernier ensemble pour chacun des parents P_1 et $M1$, l'enfant G_1 engendre le même planning que P_1. La restriction du choix de q_2 se fera afin d'éviter la situation suivante. Supposons que nous voulons construire un enfant G à partir du croisement de P et M. Les q_1 premières positions sont prises de P, ensuite, les $q_2 - q_1$ premières activités de M non encore choisies dans G, sont sélectionnées. A ce stade, les q_2 premières activités de G sont sélectionnées. Or, si les q_2 activités en question *sont dans un ordre pouvant être induit par un ou plusieurs ensembles successifs de la (se) correspondante à* P, alors G engendrera forcément le même planning que P. Dans l'exemple précédent, les $q_1(=2)$ premières positions de P_1 sont 1 et 3 respectivement. Ainsi, les $q_2 - q_1(=3-2=1)$ premières activités non sélectionnées pour G_1 dans M_1 se réduisent à l'unique activité 6. Ainsi, les $q_2 = 3$ premières activités de G sont respectivement 1, 3 et 6. Or, cet ordre des activités peut être induit par le premier ensemble de P_1.

   Soient n_1 et n_2 les nombres d'ensembles contenant les activités aux positions $1, \cdots, q_2$ dans P et M respectivement. Alors, q_2 *devra être choisie de sorte que les* q_2 *premières activités de* G *(respectivement* F*) ne soient pas toutes dans les* n_1 *(respectivement* n_2*) premiers ensembles de* P *(respectivement* M*).* Pour ce faire, nous commençons par vérifier cette condition pour la valeur $q_2 = \mu = q_1 + 1$. Si cette condition n'est pas vérifiée, nous incrémentons la valeur de μ jusqu'à ce que la condition soit vérifiée. Par la suite, nous choisirons q_2 au hasard parmi les valeurs $\mu, \cdots, N+2$. Dans l'exemple précédent, pour $q_1 = 2$, q_2 doit être choisie parmi les valeurs $5, \cdots, 10$ ($\mu = 5$).

Notons, par ailleurs, que ce choix de q_1 et q_2 n'assure pas qu'aucun des enfants ne corresponde au même planning de l'un des parents. Le but de cette démarche et de

réduire le plus possible le risque que cela se produise en évitant les cas de figure où il est certain que cette situation se présente.

3.3.1 Expérimentations

Afin d'évaluer l'impact de l'utilisation des (se) pour sélectionner les points de croisement q_1 et q_2, nous comparons les méthodes AG_SE et AG_S selon les tests suivants. Les deux premiers tests servent à l'analyse de l'impact sur la *qualité de la solution* évaluée par la *déviation* de sa durée par rapport à celle du planning optimal pour J30 et par rapport à la durée du planning donné par la méthode CPM (lorsque les contraintes sur les ressources sont relaxées) pour J60, J90 et J120 :

- **Test 5** : Les instances J30, J60, J90 et J120 sont résolues avec AG_SE et AG_S en utilisant les mêmes valeurs des paramètres ayant induit les meilleures performances dans [66], i.e., p_{mut}=0.05 et une population de taille 40. Ces instances sont résolues deux fois. La première avec une limite de 1000 plannings générés (25 générations) et la seconde avec une limite de 5000 plannings générés (125 générations).
- **Test 6** : Ce test est similaire au Test 3 (voir la section 3.2.1). Les résultats obtenus avec AG_SE et AG_S sont comparés pour chaque combinaison des valeurs de NC, RF et RS en utilisant une limite de 5000 plannings générés. L'objectif de ce test est d'évaluer l'impact de l'utilisation des (se) suivant le degré de complexité des instances. Pour chaque ensemble d'instances J30, J60, J90 et J120 nous calculons une moyenne du pourcentage de déviation pour chaque groupe de 10 instances générées avec les mêmes valeurs de NC, RF et RS. Ainsi, nous obtenons 48 (respectivement 60) moyennes de pourcentage de déviation pour chaque ensemble d'instances J30, J60 et J90 (respectivement pour les instances de J120) et pour chacune des procédures AG_SE et AG_S. L'amélioration qu'apporte AG_SE par rapport à AG_S, en considérant la moyenne du pourcentage de déviation, est ensuite calculée pour chaque combinaison des valeurs des paramètres.

Le troisième test est fait pour analyser l'impact sur le *nombre de générations parcourues avant d'atteindre la meilleure solution* :

- **Test 7** : Ce test est similaire au Test 6 sauf que c'est le nombre de générations parcourues avant d'atteindre la meilleure solution qui est considéré. Nous comparons AG_SE et AG_S pour chaque combinaison des valeurs des paramètres NC, RF et RS.

3.3.2 Résultats numériques et analyses

Les résultats du Test 5 sont résumés dans le Tableau 3.13. Pour chaque ensemble d'instances J30, J60, J90 et J120, la première ligne correspond aux résultats avec une limite de 1000 plannings générés et la seconde, avec une limite de 5000 plannings générés. La première composante de chaque couple de valeurs sur le tableau correspond au résultat de AG_SE et la seconde composante, qui est entre parenthèses, correspond au résultat de AG_S.

La seconde colonne du tableau (*% Déviation - Moy.*) donne la moyenne du pourcentage de déviation. La troisième colonne (*% Déviation - Amél.*) indique la différence entre la moyenne du pourcentage de déviation de AG_SE et AG_S afin de mesurer l'amélioration apportée par AG_SE. Le maximum de la moyenne du pourcentage de déviation est donné sur la quatrième colonne (*% Déviation - Max*). La cinquième colonne (*Nombre de générations - Moy.*) comporte la moyenne du nombre de générations parcourues avant d'atteindre la meilleure solution. Le pourcentage de l'amélioration qu'apporte AG_SE par rapport AG_S, en considérant ce nombre, est reporté sur la sixième colonne (*Nombre de générations - Amél%.*). Si la valeur est positive (respectivement négative), AG_SE a une meilleure (respectivement moins bonne) performance que AG_S. Les deux dernières colonnes (*Temps de calcul - Moy.* et *Temps de calcul - Max*) indiquent la moyenne et le maximum du temps de calcul, respectivement.

	% Déviation			Nombre de générations		Temps de calcul (Sec)	
	Moy.	Amél.	Max	Moy.	Amél.%	Moy.	Max
J30	0.84 (0.86)	0.02	10.29 (13.95)	3.72 (3.97)	6.30	0.3 (0.2)	1 (1)
	0.71 (0.77)	0.06	10.29 (13.95)	6.93 (6.72)	-3.03	1.53 (1.1)	2 (2)
J60	13.93 (14.04)	0.11	119.48 (122.08)	5.85 (6.44)	9.16	0.55 (0.43)	1 (1)
	13.39 (13.39)	0	116.88 (111.69)	16.64 (18.25)	8.82	2.88 (2.09)	3 (3)
J90	14.26 (14.39)	0.13	118.64 (116.95)	7.42 (7.32)	-1.35	0.86 (0.65)	1 (1)
	13.47 (13.69)	0.22	116.95 (116.95)	19.82 (20.68)	4.16	4.37 (3.19)	5 (4)
J120	43.86 (44.03)	0.17	229.29 (229.70)	16.13 (16.08)	-0.31	1.16 (0.88)	2 (1)
	41.69 (41.90)	0.21	224.24 (220.20)	49.69 (48.98)	-1.43	5.83 (4.41)	7 (6)

TAB. 3.13 – **Test 5** : Comparaison des résultats de AG_SE et AG_S

Les résultats du Tableau 3.13 indiquent clairement une supériorité de AG_SE en considérant la moyenne du pourcentage de déviation. En effet, mis à part les instances

J60 où il y a égalité pour une limite de 5000 plannings générés, *AG_SE donne une moyenne du pourcentage de déviation meilleure que celle de AG_S. Cette supériorité devient encore plus importante lorsque le nombre de plannings générés augmente* (sauf pour J60). Par ailleurs, *il n'y a pas une dominance claire de l'une des procédures par rapport au nombre de générations parcourues avant d'atteindre la meilleure solution.* Enfin, *la moyenne du temps de calcul de AG_SE est légèrement plus grande que celle de AG_S.* Dans le pire des cas, (pour J120 avec 5000 plannings générés), cette différence est de 1.42 secondes. Ceci est dû à l'effort de calcul supplémentaire pour choisir les points de croisement dans AG_SE. Cela dit, le temps de calcul de AG_SE reste raisonnable étant donné l'amélioration de la moyenne du pourcentage du déviation.

Comme c'était le cas pour RS_SE (voir la section 3.2), les résultats du Test 6 *n'indiquent aucun impact clair des valeurs des paramètres NC, RF et RS sur l'amélioration apportée par AG_SE* (les Tableaux 3.14 et 3.15, ayant la même structure que celle des Tableaux 3.6 et 3.7 (voir la section 3.2.2), résument les résultats du test). Cependant, une analyse plus approfondie de ces résultats mène aux valeurs qui figurent dans le Tableau 3.16 qui possède la même structure que celle du Tableau 3.8 (voir la section 3.2.2) pour les procédures RS_SE et RS_S. Les mêmes conclusions faites pour les Tableaux 3.6 et 3.7 peuvent également être faites pour les Tableaux 3.14 et 3.15.

NC	RF	RS	J30			J60			J90		
			(s)	(se)	Amél	(s)	(se)	Amél	(s)	(se)	Amél
1.50	0.25	0.20	1.60	0.46	1.14	11.80	13.42	-1.62	16.21	15.96	0.26
1.50	0.25	0.50	0.00	0.00	0.00	2.41	2.41	0.00	0.00	0.00	0.00
1.50	0.25	0.70	0.00	0.00	0.00	2.13	2.13	0.00	0.00	0.00	0.00
1.50	0.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.50	0.20	1.59	1.38	0.21	37.50	39.38	-1.88	42.59	42.16	0.43
1.50	0.50	0.50	2.19	0.92	1.27	3.33	3.54	-0.21	3.31	2.92	0.40
1.50	0.50	0.70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.50	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.75	0.20	3.65	3.64	0.01	54.10	53.40	0.70	66.67	66.31	0.36
1.50	0.75	0.50	1.46	1.30	0.15	3.98	3.22	0.77	1.42	1.69	-0.26
1.50	0.75	0.70	0.41	0.48	-0.07	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.75	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	1.00	0.20	4.02	4.47	-0.45	82.47	81.42	1.04	74.76	74.94	-0.18
1.50	1.00	0.50	2.45	2.24	0.21	4.41	4.44	-0.03	2.13	2.37	-0.24
1.50	1.00	0.70	0.17	0.69	-0.52	0.00	0.00	0.00	0.00	0.00	0.00
1.50	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	0.25	0.20	0.21	0.21	0.00	13.04	13.06	-0.02	14.49	13.35	1.14
1.80	0.25	0.50	0.48	0.48	0.00	0.81	0.81	0.00	0.33	0.33	0.00
1.80	0.25	0.70	0.00	0.00	0.00	1.30	1.30	0.00	0.00	0.00	0.00
1.80	0.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	0.50	0.20	1.19	0.59	0.60	41.71	40.80	0.91	45.07	43.68	1.39
1.80	0.50	0.50	0.61	0.43	0.18	4.29	3.78	0.52	2.76	2.63	0.13
1.80	0.50	0.70	0.15	0.00	0.15	0.73	0.40	0.33	0.00	0.00	0.00
1.80	0.50	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	0.75	0.20	2.96	3.29	-0.32	60.85	60.15	0.69	67.00	65.10	1.90
1.80	0.75	0.50	1.21	1.51	-0.30	5.11	5.18	-0.07	6.20	6.12	0.09
1.80	0.75	0.70	0.00	0.00	0.00	0.00	0.00	0.00	0.26	0.24	0.02
1.80	0.75	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.80	1.00	0.20	3.20	2.90	0.30	78.82	78.81	0.01	73.39	72.66	0.73
1.80	1.00	0.50	1.58	1.47	0.11	8.26	7.81	0.46	7.12	5.83	1.29
1.80	1.00	0.70	0.38	0.57	-0.19	0.00	0.00	0.00	0.29	0.43	-0.14
1.80	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	0.25	0.20	0.49	0.18	0.31	12.08	11.80	0.28	10.53	10.22	0.32
2.10	0.25	0.50	0.00	0.00	0.00	3.08	2.61	0.46	1.93	1.80	0.13
2.10	0.25	0.70	0.00	0.00	0.00	1.32	1.32	0.00	0.10	0.10	0.00
2.10	0.25	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	0.50	0.20	0.77	0.93	-0.16	41.10	42.11	-1.01	46.29	45.72	0.57
2.10	0.50	0.50	0.38	0.31	0.07	4.08	4.06	0.01	4.37	3.91	0.47
2.10	0.50	0.70	0.00	0.00	0.00	0.36	0.36	0.00	0.36	0.12	0.24
2.10	0.50	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	0.75	0.20	0.66	0.91	-0.25	63.80	64.33	-0.53	68.95	67.86	1.09
2.10	0.75	0.50	1.02	1.22	-0.20	5.69	6.40	-0.70	6.53	5.65	0.88
2.10	0.75	0.70	0.82	0.97	-0.15	0.54	0.54	0.00	0.00	0.10	-0.10
2.10	0.75	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2.10	1.00	0.20	1.29	1.40	-0.11	82.51	82.74	-0.23	84.92	84.66	0.25
2.10	1.00	0.50	1.66	1.32	0.34	10.67	10.61	0.06	9.01	9.37	-0.36
2.10	1.00	0.70	0.36	0.00	0.36	0.44	0.42	0.03	0.25	0.35	-0.10
2.10	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

TAB. 3.14 – **Test 6** : Déviations par combinaison de paramètres - **J30**, **J60** et **J90**

NC	RF	RS	J120		
			(s)	(se)	Amél.
1.5	0.25	0.1	37.61%	38.17%	-0.56%
1.5	0.25	0.2	23.04%	23.99%	-0.95%
1.5	0.25	0.3	3.64%	3.84%	-0.19%
1.5	0.25	0.4	3.74%	3.75%	-0.01%
1.5	0.25	0.5	0.40%	0.25%	0.14%
1.5	0.5	0.1	104.17%	104.49%	-0.32%
1.5	0.5	0.2	49.33%	49.65%	-0.32%
1.5	0.5	0.3	23.45%	23.43%	0.01%
1.5	0.5	0.4	7.02%	7.86%	-0.85%
1.5	0.5	0.5	2.16%	1.66%	0.50%
1.5	0.75	0.1	135.47%	133.47%	2.00%
1.5	0.75	0.2	73.16%	72.94%	0.22%
1.5	0.75	0.3	30.18%	30.12%	0.06%
1.5	0.75	0.4	15.75%	15.56%	0.18%
1.5	0.75	0.5	3.00%	2.77%	0.24%
1.5	1	0.1	172.38%	172.18%	0.21%
1.5	1	0.2	76.42%	76.61%	-0.19%
1.5	1	0.3	38.07%	37.49%	0.58%
1.5	1	0.4	19.19%	19.69%	-0.50%
1.5	1	0.5	5.42%	5.47%	-0.04%
1.8	0.25	0.1	34.34%	33.64%	0.70%
1.8	0.25	0.2	19.30%	18.41%	0.89%
1.8	0.25	0.3	2.84%	2.84%	0.00%
1.8	0.25	0.4	2.06%	2.58%	-0.52%
1.8	0.25	0.5	1.93%	2.17%	-0.23%
1.8	0.5	0.1	91.90%	95.97%	-4.07%
1.8	0.5	0.2	49.62%	49.03%	0.59%
1.8	0.5	0.3	18.94%	18.87%	0.06%
1.8	0.5	0.4	9.80%	8.57%	1.23%
1.8	0.5	0.5	5.14%	4.88%	0.26%

NC	RF	RS	J120		
			(s)	(se)	Amél.
1.8	0.75	0.1	139.86%	138.78%	1.08%
1.8	0.75	0.2	58.61%	57.41%	1.20%
1.8	0.75	0.3	34.94%	34.96%	-0.02%
1.8	0.75	0.4	19.45%	18.11%	1.34%
1.8	0.75	0.5	7.39%	7.62%	-0.23%
1.8	1	0.1	156.71%	153.99%	2.73%
1.8	1	0.2	82.99%	82.04%	0.95%
1.8	1	0.3	38.89%	38.21%	0.68%
1.8	1	0.4	20.53%	20.38%	0.15%
1.8	1	0.5	5.77%	5.38%	0.38%
2.1	0.25	0.1	34.70%	32.84%	1.86%
2.1	0.25	0.2	19.04%	19.71%	-0.68%
2.1	0.25	0.3	9.70%	8.82%	0.88%
2.1	0.25	0.4	4.62%	3.84%	0.78%
2.1	0.25	0.5	0.47%	0.58%	-0.11%
2.1	0.5	0.1	97.40%	97.48%	-0.09%
2.1	0.5	0.2	51.49%	51.48%	0.01%
2.1	0.5	0.3	25.92%	25.46%	0.46%
2.1	0.5	0.4	13.76%	12.72%	1.04%
2.1	0.5	0.5	4.16%	3.86%	0.30%
2.1	0.75	0.1	154.07%	154.77%	-0.69%
2.1	0.75	0.2	73.71%	71.49%	2.22%
2.1	0.75	0.3	34.39%	33.56%	0.84%
2.1	0.75	0.4	17.92%	18.91%	-0.99%
2.1	0.75	0.5	7.05%	6.74%	0.31%
2.1	1	0.1	181.53%	182.13%	-0.60%
2.1	1	0.2	81.85%	81.92%	-0.07%
2.1	1	0.3	44.17%	44.82%	-0.65%
2.1	1	0.4	22.29%	22.16%	0.14%
2.1	1	0.5	10.89%	10.93%	-0.04%

TAB. 3.15 – **Test 6** : Déviations par combinaison de paramètres - **J120**

	J30	J60	J90	J120
% Meils	31.25%	29.16%	41.66%	60%
% Pires	22.91%	20.83%	14.58%	40%
% Égales	45.83%	50%	43.75%	0%
Moy. Dev. Meil	0.36%	0.45%	0.60%	0.70%
Moy. Dev. Pire	-0.25%	-0.63%	-0.20%	-0.54%
Max. Dev. Meil	1.27%	1.04%	1.90%	2.73%
Max. Dev. Pire	-0.52%	-1.88%	-0.36%	-4.07%

TAB. 3.16 – **Test 6** : Comparaison globale des résultats de AG_SE et AG_S

Par ailleurs, nous avons effectué dans ce cas aussi le test de Wilcoxon pour échantillons appariés avec un niveau de confiance de 5% sur les valeurs des colonnes *(s)* et *(se)* des Tableau 3.14 et 3.15 pour chaque ensemble J30, J60, J90 et J120. Contrairement à RS_SE et RS_S, *ces tests ne se sont révélés significatifs que pour les ensembles J90 et J120.* Il semblerait que pour les méthodes AG_SE et AG_S, l'impact des (se) sur le pourcentage de déviation n'est significatif que pour des instances de tailles relativement importantes.

Les résultats du Test 7 *n'indiquent également pas d'impact clair des paramètres NC, RF et RS sur le nombre de générations parcourues avant d'atteindre la meilleure solution* (les Tableaux 3.17, 3.18, 3.19 et 3.20, dont la structure est similaire à celle des Tableaux 3.9, 3.10, 3.11 et 3.12 (voir la section 3.2.2), résume les résultats du test). Cette situation est similaire à celle observée pour les procédures RS_SE et RS_S (voir la section 3.2.2). Aussi, la moyenne de la différence entre les nombres de générations parcourues avant d'atteindre la meilleure solution pour AG_SE et AG_S ne semble pas augmenter avec l'augmentation de la taille du problème puisque les valeurs obtenus sont -0.21, 1.61, 0.85 et -0.71 pour J30, J60, J90 et J120 respectivement (une différence négative signifie que AG_S parcourt moins de générations avant d'atteindre la meilleure solution).

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	-1.65	0.25	-0.58	0.2	-3.37
1.8	0.56	0.5	1.15	0.5	2.12
2.1	0.46	0.75	0.025	0.7	0.41
		1.0	-1.42	1	0

TAB. 3.17 – **Test 7** : Évolution de l'amélioration sur le nombre de générations pour J30

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	1.31	0.25	0.89	0.2	0.72
1.8	3.21	0.5	0.82	0.5	4.5
2.1	0.3	0.75	1.45	0.7	1.21
		1.0	3.27	1	0

TAB. 3.18 – **Test 7** : Évolution de l'amélioration sur le nombre de générations pour J60

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	1.03	0.25	1.05	0.2	4
1.8	0.26	0.5	-1.76	0.5	-1.04
2.1	1.27	0.75	0.3	0.7	0.46
		1.0	3.83	1	0

TAB. 3.19 – **Test 7** : Évolution de l'amélioration sur le nombre de générations pour J90

NC	*NbPlg*	*RF*	*NbPlg*	*RS*	*NbPlg*
1.5	0.065	0.25	2.88	0.1	-0.25
1.8	1.25	0.5	-3.8	0.2	2.3
2.1	-3.44	0.75	-3.29	0.3	-1.15
		1	1.38	0.4	-5.95
				0.5	1.51

TAB. 3.20 – **Test 7** : Évolution de l'amélioration sur le nombre de générations pour J120

Chapitre 4

Séquences d'ensembles d'activités pour la généralisation du problème

4.1 Introduction

Dans les chapitres 2 et 3, nous avons introduit la représentation sous forme de (se) et nous avons indiqué comment elle peut être exploitée pour améliorer les performances des méthodes de résolution du $RCPSP$ se basant sur la représentation en (s). Ces deux représentations restent très liées puisque les méthodes de résolution qui exploitent les (se) restent tout de même basées sur les (s) pour représenter les solutions. Les (se) n'interviennent que lors de certaines transformations sur les (s) pour réduire l'espace de recherche induit par ces dernières.

Nous proposons dans ce chapitre une extension de l'utilisation des (se) à une large généralisation du problème $RCPSP$. Pour ce faire, nous abordons d'abord une extension des (s) pour cette généralisation puisque la notion de (se) ne peut être complètement définie sans celle de (s).

Notre intérêt envers ces extensions vient, d'une part, de l'efficacité des (s) dans un très grand nombre de méthodes de résolution proposées pour le $RCPSP$ [1, 16, 58, 66, 73, 105] ainsi que de l'amélioration importante que la représentation en (se) peut apporter. Nous espérons ainsi transposer ces résultats et ré-exploiter ces méthodes pour la généralisation proposée du problème. Notons que la majorité des problèmes réels de planification comportent généralement beaucoup plus de contraintes que le $RCPSP$.

4.2 Généralisation du problème $RCPSP$

Nous proposons ici une généralisation du problème $RCPSP$ qui est notée (ORD)+(C). Le bloc (ORD) comporte *une fonction objectif à minimiser* ainsi que l'*ensemble de toutes les contraintes de précédence entre les activités.* Cet ensemble peut être vide. Le Bloc (C) comporte un *ensemble de contraintes arbitraires.* Cet ensemble peut également être vide.

Selon cette définition, le problème $RCPSP$ est un problème (ORD)+(C) où la fonction objectif de (ORD) est la durée totale du projet et où (C) contient les contraintes sur les ressources. Aussi, le problème d'ordonnancement classique (qui peut être résolu de façon optimale avec la méthode CPM), est un problème (ORD)+(C) où la fonction objectif de (ORD) est la durée totale du planning et où (C) est vide.

4.3 Représentation en séquences d'activités

Pour le problème (ORD)+(C), lorsque le bloc (ORD) contient des contraintes de précédence, seules ces contraintes sont considérées pour construire une (s). Dans ce cas, une (s) est dite *(vp) lorsque toute activité dans la (s) est positionnée après ses prédécesseurs.*

Pour décoder une (s)(vp), une méthode similaire à Serial SGS (voir la section 2.2) est utilisée. Les activités sont sélectionnées selon leur ordre dans la (s) puis planifiées à leur date au plus tôt *après leurs prédécesseurs* de sorte que *toutes les contraintes de (ORD)+(C) soient satisfaites.*

De nouvelles conditions doivent être définies pour que cette méthode de décodage soit applicable à toute (s)(vp). Pour illustrer cela, considérons l'exemple simple suivant où deux activités 1 et 2, de durées 2, doivent être planifiées. Comme contraintes, supposons que ces activités ne peuvent se dérouler en même temps et que l'activité 1 ne peut commencer après la période 2. Ceci est bien un problème (ORD)+(C) où il n'y a pas de contraintes de précédence et où le bloc (C) contient les contraintes définies plus haut. Pour cet exemple, il existe uniquement deux (s)(vp) : $a_1 = [1, 2]$ et $a_2 = [2, 1]$. a_1 peut être décodée en un planning où les activités 1 et 2 commencent aux périodes 1 et 3 respectivement. Par ailleurs, a_2 *ne peut être décodée.* En effet, une fois que l'activité 2 est planifiée pour commencer à la période 1, la première contrainte impose à l'activité 1 de commencer à partir de la période 3, ce qui ne satisfait pas la seconde contrainte.

Dans ce qui suit, nous introduisons une condition simple sur les contraintes (C) pour éviter ce type de situations.

4.3.1 Contraintes souples

Définition 6 Soit a une (s)(vp) pour le problème $(ORD)+(C)$. Une contrainte particulière $c \in (C)$ est **souple** si, en ne considérant que la contrainte c, toute activité j dans a peut être planifiée à une période $D_j(c)$ ou **à n'importe quelle période plus tard.** Une telle contrainte est notée c^s.

Notons que dans l'exemple précédent, la deuxième contrainte imposant que l'activité 1 ne peut commencer après la période 2 *n'est pas souple.*

Définition 7 Le bloc (C) est **souple** si toutes les contraintes $c \in (C)$ sont c^s. Un tel bloc est noté (C^s).

Ainsi, dans une (s)(vp) pour $(ORD)+(C^s)$, toute activité i peut être planifiée à une période $D_i = Max_{c\in(C)}D_i(c)$. Il est facile de vérifier que toute (s)(vp) pour ce problème peut être décodée avec la variante de la méthode Serial SGS proposée plus haut. Par ailleurs, notons que *les contraintes sur les ressources dans $RCPSP$ sont souples*, et donc $RCPSP$ est *un cas particulier* de problème $(ORD)+(C^s)$.

Selon la Définition 7, l'ajout de n'importe quel nombre de contraintes souples à un problème $RCPSP$ engendre également un problème $(ORD)+(C^s)$. Voici quelques exemples de contraintes *souples* :

1. Une activité ne peut commencer *avant* une certaine période.
2. Certaines activités doivent être séparées par, *au moins*, une durée DR.
3. Certaines activités *ne peuvent être planifiées dans une certaine fenêtre de temps.*

Par contre, les contraintes suivantes *ne sont pas souples* :

1. Une activité ne peut commencer *après* une certaine période.
2. Certaines activités doivent *se dérouler en même temps.*
3. Certaines activités *doivent être planifiées dans une certaine fenêtre de temps.*

Pour mieux illustrer la notion de *souplesse*, nous considérons l'exemple du problème $(ORD)+(C^s)$ suivant pour lequel des contraintes souples sont ajoutées à un problème $RCPSP$. Le problème est illustré à la Figure 4.1. Les activités 1 et 10 sont virtuelles et il existe une unique ressource renouvelable disponible en quantité 6 durant chaque période élémentaire.

Les contraintes souples additionnelles sont les suivantes :

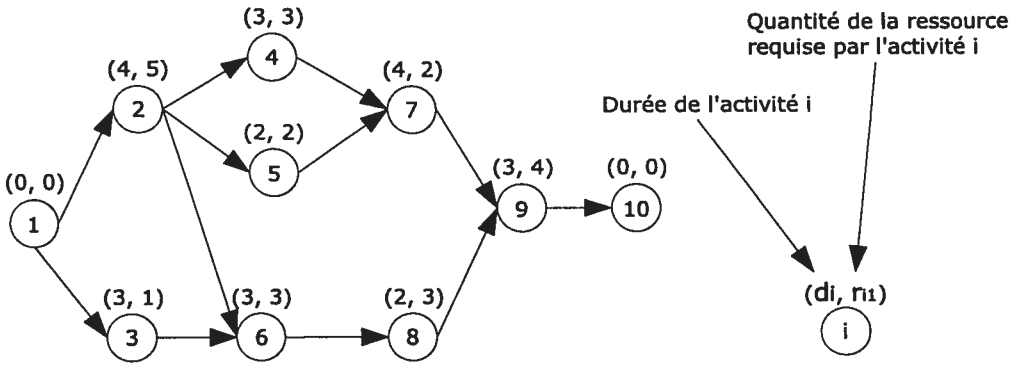

FIG. 4.1 – Exemple d'un problème $(ORD)+(C^s)$

1. Les activités 2, 4 et 8 ne peuvent commencer respectivement avant les périodes 2, 7 et 9.
2. Les activités 7 et 8 ne peuvent se dérouler en même temps.

Le planning associé à la (s)(vp)

$$s_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

est illustré à la Figure 4.2.

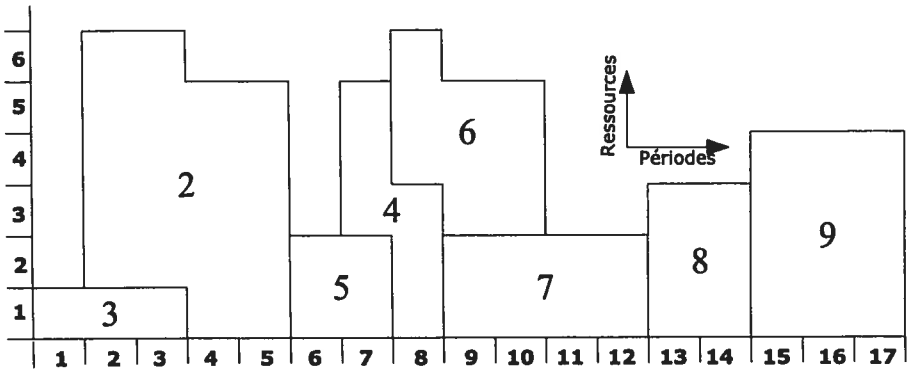

FIG. 4.2 – Planning associé à s_1 pour le problème $(ORD)+(C^s)$

Voici quelques détails sur le décodage de s_1 :

- L'activité 2 est planifiée à la période 2 car elle ne peut commencer plus tôt.
- L'activité 4 est planifiée à la période 7 car :
  - Elle a l'activité 2 comme prédécesseur (donc, ne peut commencer avant la période 6).
  - Elle ne peut commencer avant la période 7.
- L'activité 6 est planifiée à la période 8 car :

- Elle a les activités 2 et 3 comme prédécesseurs (donc, ne peut commencer avant la période 6).
- Les ressources sont insuffisantes jusqu'à la période 8.

- L'activité 8 est planifiée à la période 13 car :
  - Elle a l'activité 6 comme prédécesseur (donc, ne peut commencer avant la période 11).
  - Elle ne peut commencer avant la période 9.
  - Les ressources sont insuffisantes jusqu'à la période 11.
  - Elle ne peut être exécutée en même temps que l'activité 7 (donc, ne peut s'exécuter aux périodes 9, 10, 11 et 12).

4.3.2 Le mode Backward

Jusqu'à présent, nous avons généralisé l'utilisation des (s)(vp) pour le problème $(ORD)+(C^s)$ en tenant compte du mode *forward* (voir la section 1.5.3) pour lequel les activités sont planifiées de la première à la dernière dans une (s)(vp) à la date au plus tôt en veillant à respecter l'ensemble des contraintes du problème.

Plusieurs heuristiques performantes utilisant les deux modes de planification sont proposées dans la littérature [1, 2, 79]. Dans cette section, nous discutons des conditions permettant le décodage des (s)(vp) pour le problème $(ORD)+(C^s)$ selon le mode *backward*.

Les Définitions 6 et 7 portant sur la souplesse, sont adaptées au mode backward comme suit :

Définition 8 Soit a une (s)(vp) pour le problème $(ORD)+(C)$. Une contrainte particulière $c \in (C)$ est **souple selon le mode backward** si, en ne considérant que la contrainte c, toute activité j dans a peut être planifiée à une période $D_j^B(c)$ ou **à n'importe quelle période plus tôt.** Une telle contrainte est notée c^{sb}.

Définition 9 Le bloc (C) est **souple selon le mode backward** si toutes les contraintes $c \in (C)$ sont c^{sb}. Un tel bloc est noté (C^{sb}).

Le résultat suivant démontre que les notions de *souplesse* et *souplesse selon le mode backward* sont équivalentes.

Théorème 4 Le bloc (C) est souple si et seulement si il est souple selon le mode backward.

Preuve 4 Nous montrons, tout d'abord, que si (C) est souple, *alors* il est souple selon le mode backward.

Par l'absurde, supposons que (C) est souple et ne l'est pas selon le mode backward. Ainsi, il existe une (s)(vp) et une première activité i de cette dernière qui ne peut être planifiée selon le mode backward à partir d'une date D_i^t ou plus tôt.

Soient :

- A_{av}^i et A_{ap}^i : les ensembles des activités positionnées respectivement avant et après i dans la (s)(vp).
- $P_{A_{ap}^i}^B$: le *planning partiel* des activités A_{ap}^i selon le mode backward (ces activités peuvent être planifiées selon ce mode puisque i est la première qui ne peut l'être).
- P^F : le planning associé à la (s)(vp) en mode forward.
- DR_j^B : la durée, dans $P_{A_{ap}^i}^B$, qui sépare la date de début d'une activité $j \in A_{ap}^i$ de la date de début de l'activité dans A_{ap}^i qui *commence le plus tôt* dans $P_{A_{ap}^i}^B$.

À présent, nous ajoutons plusieurs contraintes souples $(\overline{C})$ au problème $(ORD)+(C)$. Les contraintes $(\overline{C})$ se résument comme suit : $\forall j \in A_{ap}^i$, les activités i et j doivent être séparées par au moins une durée $D_{max} + DR_j^B$. D_{max} est une durée assez grande choisie de sorte que toutes les activités A_{ap}^i soient planifiées *après la fin* l'activité i en mode forward. Notons que le bloc $(\mathrm{C}) + (\overline{C})$ est souple puisque (C) et $(\overline{C})$ sont souples individuellement.

Générons maintenant le planning associé à la (s)(vp), en mode forward, pour le problème $(ORD) + (\mathrm{C}) + (\overline{C})$. Dans ce planning, chaque activité $j \in A_{ap}^i$ sera planifiée à une date la séparant de i d'*exactement* une durée de $D_{max} + DR_j^B$ puisque :

1. Les activités de A_{ap}^i sont planifiées aux dates au plus tôt.
2. Dans le planning P^F, les activités de A_{ap}^i ont pu être planifiées plus tôt. Comme (C) est souple, ces activités peuvent être retardées jusqu'à ce que les contraintes dans $(\overline{C})$ soient satisfaites.
3. Les "positions relatives" des activités de A_{ap}^i entre elles pourraient peut-être entraîner une violation des contraintes. Or, ceci n'est pas possible dans ce cas car notre choix des durées $DR_j^B, j \in A_{ap}^i$ induit un positionnement des activités entre elles qui est *identique à celui dans* $P_{A_{ap}^i}^B$ qui est un planning partiel en mode backward qui n'induit pas une violation des contraintes entre les activités de A_{ap}^i.

Maintenant, si nous générons le planning de la (s)(vp), selon le mode backward, pour le problème $(ORD) + (\mathrm{C}) + (\overline{C})$, la planification des activités A_{ap}^i donne le même planning

partiel $P^B_{A^i_{ap}}$ (obtenu sans $(\overline{C})$) car les contraintes $(\overline{C})$ n'influent en rien leur planification, puisque i est planifiée après les activités de A^i_{ap}. Comme cela a été souligné plus haut, la planification selon le mode forward donne des "positions relatives" des activités A^i_{ap} entre elles identiques à ceux du planning partiel $P^B_{A^i_{ap}}$, donc identique à celui en mode backward. Or, puisque i a pu être planifiée, selon le mode forward, avec la planification partielle $P^B_{A^i_{ap}}$, cela sera possible selon le mode backward aussi. En effet, il suffit de planifier, en mode backward, l'activité i à une période la séparant d'une durée $D_{max}+DR^B_j$ de chaque activité $j \in A^i_{ap}$. De plus, nous pouvons augmenter encore la valeur de D_{max} de n'importe quelle quantité pour pouvoir affirmer qu'*il existe une date à partir de laquelle l'activité i peut être planifiée selon le mode backward pour le problème* $(ORD)+(C)+(\overline{C})$. À présent, puisque i peut être ainsi planifiée selon le mode backward à partir d'une certaine date avec les contraintes (C) + $(\overline{C})$, cela sera forcément possible avec *uniquement* les contraintes (C). Ceci contredit notre supposition initiale.
Ainsi, si le bloc (C) est souple, alors il l'est aussi selon le mode backward. Une démarche similaire peut être suivie pour montrer que si le bloc (C) est souple selon le mode backward, alors il est aussi souple. □

Nous pouvons ainsi déduire facilement le résultat suivant :

Corollaire 1 Toute (s)(vp) peut être décodée, selon les modes forward et backward, en un planning réalisable pour le problème $(ORD)+(C^s)$.

Pour illustrer le mode de planification en backward, nous reconsidérons l'exemple de la Figure 4.1. Le planning associé à la (s)(vp)

$$s_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

selon le mode backward est illustré à la Figure 4.3. Notons que ce planning est différent de celui de la Figure 4.2 qui correspond aussi à s_1 mais qui est construit selon le mode forward.

Dans ce qui suit, nous utiliserons uniquement la notion de souplesse puisque la souplesse selon le mode backward est équivalente.

4.3.3 Conditions d'existence d'une codification optimale

Comme déjà mentionné, si le problème $RCPSP$ possède une solution optimale, alors il existe toujours une (s)(vp) qui correspond à une solution optimale [86, 130]. Dans cette section, nous abordons cette question pour le problème $(ORD)+(C^s)$.

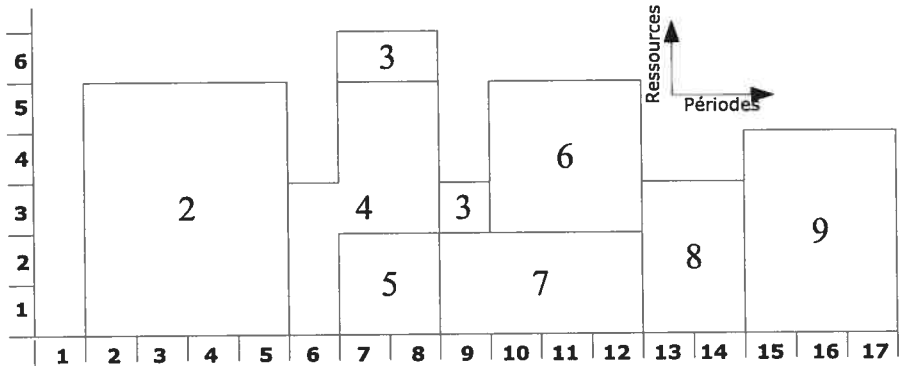

FIG. 4.3 – Planning associé à s_1 obtenu selon le mode backward pour $(ORD)+(C^s)$

Tout d'abord, il est nécessaire d'introduire quelques définitions. Certaines d'entre elles ont été introduites par Backer [7] pour le problème de Job Shop puis étendues et formalisées par Sprecher et al. [130] pour le problème $RCPSP$. À présent, nous allons les étendre au problème $(ORD)+(C^s)$.

Définition 10 Un *shift à gauche* (left shift) d'une activité j dans un planning réalisable consiste à planifier plus tôt l'activité j sans modifier les dates de planification des autres activités pour obtenir un nouveau planning réalisable.

Définition 11 Un *shift à gauche d'une période* (one-period left shift) d'une activité j est un shift à gauche de j où j commence une période plus tôt.

Définition 12 Un *shift local à gauche* (local left shift) d'une activité j est obtenu par une succession de shifts à gauche d'une période de j.

Définition 13 Un *shift global à gauche* (global left shift) d'une activité j est un shift à gauche de j qui ne peut être obtenu par un shift local à gauche.

Définition 14 Une solution est *semi-active* (semi-active) si elle est réalisable et ne peut subir aucun shift local à gauche.

Définition 15 Une solution est *active* (active) si elle est réalisable et ne peut subir ni un shift local à gauche ni un shift global à gauche.

Il est facile de vérifier que *toute (s)(vp) engendre une solution active pour* $(ORD)+(C^s)$ puisque chaque activité est planifiée à sa date au plus tôt et donc elle ne peut être planifiée plus tôt sans changer la date de planification des autres activités.

Définition 16 Dans un problème de minimisation, la fonction objectif est dite *régulière* (regular) si elle est non décroissante en fonction des dates de fin des activités.

Notons que la minimisation de la *durée totale d'un projet* (une fonction objectif fréquemment utilisée dans plusieurs problèmes de planification tel que le $RCPSP$) *est une fonction régulière.* Plusieurs autres fonctions régulières sont présentées dans [112, 124] pour les problèmes de planification.

Définition 17 Soient P un planning réalisable, i une activité et AP_i l'ensemble des activités qui commencent à la même date que i ou plus tard. Un *planning partiel* P_i de P est un planning où seulement les activités $AP_i \cup \{i\}$ apparaissent et qui commencent aux mêmes dates que dans P.

Définition 18 Soient i une activité, P_i un planning partiel du planning réalisable P et $\bar{P}_i$ le planning P duquel les activités AP_i sont enlevées.

- $C\bar{O}N_i$: un sous-ensemble des contraintes du problème $(ORD)+(C^s)$ tel que *toutes* les activités concernées par ces contraintes apparaissent *uniquement* dans $\bar{P}_i$.
- CON_i : un sous-ensemble des contraintes du problème $(ORD)+(C^s)$ qui ne sont pas dans $C\bar{O}N_i$.

Définition 19 Soient i une activité et d_i sa date de début dans le planning partiel P_i. P_i est *extensible localement* si un shift local à gauche de i dans P_i de $(d_i - 1)$ périodes est possible en ne considérant que les contraintes CON_i.

Définition 20 Un planning réalisable P est *extensible globalement* si, pour toute activité i, le planning partiel P_i est extensible localement.

Dans tous ce suit, nous discutons uniquement de l'existence d'une (s)(vp) donnant un planning optimal au problème $(ORD)+(C^s)$ *ayant une fonction objectif régulière.* Ce problème sera noté $(ORD^r)+(C^s)$.

La condition de souplesse

Selon nos résultats à la section 4.3.1, la souplesse des contraintes suffit pour le décodage de n'importe quelle (s)(vp) en un planning réalisable pour le problème $(ORD)+(C^s)$. Dans cette section, nous vérifions si cette condition reste suffisante pour l'existence d'une (s)(vp) engendrant une solution optimale pour le problème.

Considérons l'exemple simple du problème $(ORD^r)+(C^s)$ suivant. Il est composé d'un problème $RCPSP$ auquel des contraintes souples sont ajoutées. La Figure 4.4 illustre les principales données du problème $RCPSP$ pour lequel une unique ressource est disponible en quantité 4 à chaque période élémentaire.

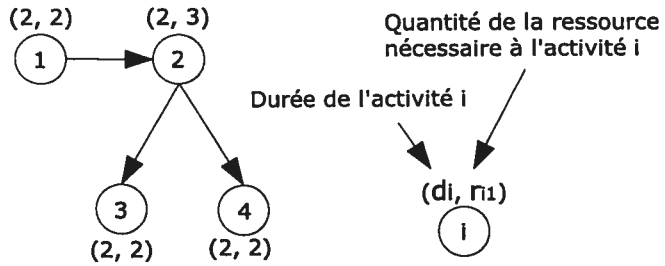

FIG. 4.4 – Exemple d'un problème $(ORD^r) + (C^s)$

Les contraintes souples ajoutées sont les suivantes :

1. Les activités 1 et 2 ne doivent pas être séparées par 2 périodes.
2. Les activités 2 et 3 ne doivent pas être séparées par 2 périodes.
3. L'activité 3 ne peut commencer avant la période 7.

Pour cet exemple, *il existe uniquement* 2 *(s)(vp)* : $a_1 = [1, 2, 3, 4]$ et $a_2 = [1, 2, 4, 3]$. a_1 et a_2 engendrent le même planning lorsqu'elles sont décodées selon le mode forward. Ce planning est représenté à la Figure 4.5 et il est de durée 9.

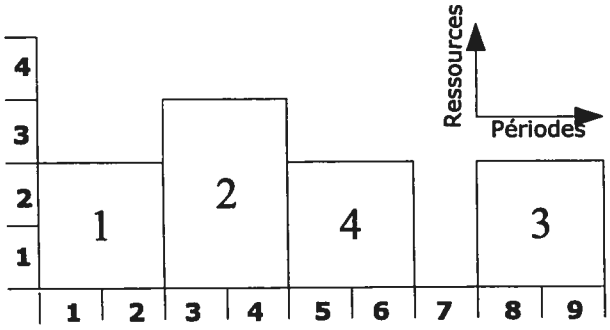

FIG. 4.5 – Planning donné par a_1 et a_2 en mode forward

À présent, considérons le planning réalisable de la Figure 4.6. Ce dernier est de durée 8 et il est donc *meilleur que celui donné par* a_1 *et* a_2.

Cet exemple montre clairement que *l'ensemble des plannings engendrés par toutes les (s)(vp) ne contient pas le planning optimal.* Ainsi, *la condition de souplesse n'est pas suffisante* pour garantir l'existence d'une d'une (s)(vp) engendrant un planning optimal.

Cet exemple montre aussi que l'ensemble des solutions engendrées par toutes les (s)(vp) *peut ne pas contenir toutes les solutions actives* car les solutions aux Figures 4.5 et 4.6 sont actives.

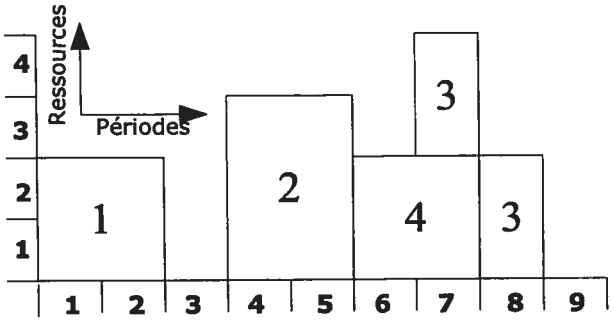

FIG. 4.6 – Planning meilleur que celui donné par a_1 et a_2

Par ailleurs, même *si nous considérons aussi le mode backward, l'ensemble des plannings engendrés peut ne pas contenir aussi le planning optimal.* Pour l'exemple précédent, la Figure 4.7 illustre le planning généré par a_1 et a_2 selon le mode backward qui est aussi de durée 9. Il est donc *moins bon* que celui de la Figure 4.6.

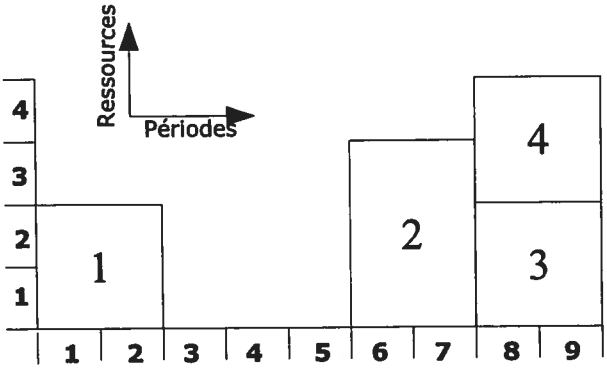

FIG. 4.7 – Planning donné par a_1 et a_2 en mode backward

Conditions suffisantes

Dans cette section, nous donnons une nouvelle condition garantissant l'existence d'une (s)(vp) engendrant un planning optimal pour le problème $(ORD^r)+(C^s)$. Pour cela, nous utilisons les deux principaux résultats suivants :

Théorème 5 Si le problème $(ORD^r) + (C^s)$ possède une solution optimale, alors il possède une solution optimale *active.*

Preuve 5 Soit a_{opt} une solution optimale de $(ORD^r) + (C^s)$. Si a_{opt} est active, alors le résultat est vrai.
Si a_{opt} n'est pas active, alors il est possible d'appliquer un shift à gauche à a_{opt}. Comme a_{opt} est optimale et la fonction objectif est régulière, sa valeur ne changera pas après cette opération. Appliquons donc, de façon successive, autant de shifts à gauche que possible. Le nombre de shifts à gauche à faire est forcément fini car, à chaque fois, une activité est avancée d'au moins une période et le nombre de périodes est fini. À la fin, nous obtenons une solution a'_{opt} donnant la même valeur à la fonction objectif que a_{opt} et pour laquelle aucun shift à gauche n'est possible. a'_{opt} est donc une solution optimale active pour le problème. □

Notons que pour le Théorème 5, la souplesse de (C^s) n'est pas exploitée de façon explicite dans la démonstration. Cependant, cette condition reste requise pour assurer que toute (s)(vp) puisse être décodée en un planning réalisable.

Notre second résultat introduit une condition garantissant l'existence d'une (s)(vp) pour chaque solution active.

Théorème 6 Considérons le problème $(ORD^r) + (C^s)$. Si *tout planning réalisable P est extensible globalement*, alors il existe une (s)(vp) pour toute solution active.

Preuve 6 Soient P une solution réalisable active et a_P une (s)(vp) construite comme suit : à chaque étape n, la n^{ime} activité de a_P est sélectionnée. L'ordre des activités dans a_P est l'ordre dans lequel les activités sont sélectionnées. À une étape n, l'ensemble des activités candidates est créé. Il est composé des activités qui n'ont aucun prédécesseur non planifié. Pour chaque activité candidate, nous déterminons une date de planification au plus tôt où l'activité peut être planifiée en considérant uniquement les activités déjà planifiées. La n^{ime} activité choisie pour construire a_P, est une activité candidate qui possède une date de planification au plus tôt *égale* à sa date de planification dans P.
Il est clair que si toutes les activités sont choisies, la (s)(vp) a_P correspondra à P. Ainsi, par cette méthode constructive, il serait possible d'associer à toute solution active une (s)(vp).
À présent, supposons que cela ne soit pas possible ; c'est-à-dire, qu'à une certaine étape n_0, toutes les activités candidates *ont une date de planification au plus tôt différente de celle dans P*. Dans ce cas, soient j une activité candidate, d_j sa date de planification au plus tôt et d_j^P sa date de planification dans P. Examinons les deux cas de figure suivants :

1. Si $d_j^P < d_j$

   Dans ce cas, l'activité j n'a pas pu être planifiée plus tôt que d_j dans P et cela en présence d'une partie seulement des activités planifiées aux mêmes dates que dans P. Donc, *l'activité j ne pourrait être planifiée plus tôt que d_j dans P également.* Or, la date de planification de l'activité j dans P est $d_j^P < d_j$. Par conséquent, P ne serait pas *réalisable.* Ceci contredit nos suppositions initiales. Donc, nous ne pouvons avoir $d_j^P < d_j$.

2. Si $d_j < d_j^P$

   Selon le cas précédent, chaque activité candidate j doit satisfaire $d_j < d_j^P$. Dans ce cas, il est clair que les contraintes qui l'empêcheraient d'être planifiée à la date d_j dans P *sont celles qui la lient aux activités non encore planifiées.* Ce sont ces contraintes qui ont retardé la date de planification de l'activité j jusqu'à d_j^P.

   À présent, soit i l'activité candidate qui commence le plus tôt dans P et P_i le planning partiel de P. Ainsi, ce sont les contraintes CON_i qui empêchent i d'être planifiée à la date d_i alors que les contraintes $C\bar{O}N_i$ le permettent. Par ailleurs, notons que les activités qui figurent dans P_i sont celles qui n'ont pas encore été planifiées (i est l'activité candidate qui commence le plus tôt dans P et le reste sont des successeurs, directs ou indirects, des activités candidates).

   Or, comme tout planning réalisable est extensible globalement, cela est particulièrement vrai pour P. Ainsi, P_i est extensible localement. Donc, il sera toujours possible de planifier plus tôt (par un shift local de $(d_i^P - 1)$) l'activité i dans P_i sous les contraintes CON_i. Par conséquent, *les contraintes qui lient i aux activités non encore planifiées ne l'empêchent pas d'être planifiée à la date d_i au lieu de d_i^P.* Ainsi, aucune contrainte du problème n'empêche l'activité i d'être planifiée à la date d_i dans P. Ceci contredit le fait que P soit une solution active.

   Donc, il est toujours possible de construire une (s)(vp) pour chaque solution active.□

Le résultat important suivant est une conséquence directe des Théorèmes 5 et 6.

Corollaire 2 Soit $(ORD^r)+(C^s)$ un problème ayant une solution optimale. Si tout planning réalisable est extensible globalement, alors il existe une (s)(vp) donnant la solution optimale.

Selon le Théorème 5, si le problème possède une solution optimale alors il possède une solution optimale active. En considérant le Théorème 6, il existe une (s)(vp) représentant la solution optimale active.

À présent, considérons deux problèmes $(ORD^r) + (C_1^s)$ et $(ORD^r) + (C_2^s)$ ayant des plannings optimaux. Supposons aussi que pour chacun de ces problèmes, chaque planning réalisable est globalement extensible. Soit $(C_{1,2}^s)$ l'union des contraintes (C_1^s) et (C_2^s). Il est facile de vérifier que pour le problème $(ORD^r) + (C_{1,2}^s)$, *chaque planning réalisable est globalement extensible.* En effet, si chaque planning partiel réalisable P_i pour $(ORD^r) + (C_{1,2}^s)$ est localement extensible pour les contraintes (C_1^s) (respectivement (C_2^2)), il est aussi localement extensible en considérant les contraintes $(C_{1,2}^s)$. Ceci signifie que *si un ensemble est composé de contraintes possédant individuellement cette propriété, cet ensemble possède aussi cette propriété.* Comme la souplesse est aussi préservée, le problème résultant possède une (s)(vp) représentant un planning optimal selon le Théorème 6.

Il est intéressant de noter que pour le cas particulier du problème $RCPSP$, chaque planning réalisable est globalement extensible. Puisque les contraintes sur les ressources sont aussi souples, alors le problème possède une (s)(vp) donnant un planning optimal. Ce résultat a déjà été démontré dans [87, 130].

Pour l'exemple de la Figure 4.4, les contraintes souples "les activités 1 (respectivement 2) et 2 (respectivement 3) ne doivent pas être séparées par 2 périodes" *engendrent au moins un planning partiel qui n'est pas localement extensible.* Par exemple, le planning partiel P_2 de la Figure 4.6 n'est pas localement extensible.

Exemples de problèmes ayant une (s)(vp) engendrant un planning optimal

Comme déjà souligné, l'existence d'une (s)(vp) engendrant un planning optimal pour le problème $(ORD^r)+(C^s)$ peut être établi en montrant que :

1. Chaque contrainte c, prise individuellement dans (C^s), est souple.
2. Chaque planning réalisable pour le problème est globalement extensible.

Nous donnons ici une liste non exhaustive d'exemples de contraintes vérifiant ces conditions.

1. Un couple d'activités doit être séparé par, *au moins*, une durée DR.
2. Un ensemble d'activités ne peuvent être *exécutées simultanément.*
3. Une activité ne peut commencer *avant* une certaine période.
4. Une activité *ne peut être planifiée dans une certaine fenêtre de temps.*
5. Contraintes sur les ressources similaires à celles du $RCPSP$.

6. Contraintes sur les ressources similaires à celles du $RCPSP$ mais où *les quantités disponibles ne sont pas nécessairement identiques à chaque période élémentaire.* Dans ce cas, la disponibilité des ressources doit permettre à chaque activité d'être planifiée individuellement à n'importe quelle période.

7. Contraintes sur les ressources similaires à celles du $RCPSP$ mais où *les quantités des ressources requises pour une activité change tout au long de sa durée d'exécution.* Par exemple, une activité de durée 2 peut nécessiter 2 unités d'une ressource à sa première période d'exécution et 3 unités à sa seconde période d'exécution. Dans ce cas aussi, la disponibilité des ressources doit permettre à chaque activité d'être planifiée individuellement à n'importe quelle période.

4.4 Extensions de techniques du $RCPSP$ au problème généralisé

Plusieurs méthodes de résolution développées pour le problème $RCPSP$ utilisent les (s)(vp). Par exemple, [1, 66, 73] (algorithmes génétiques), [16] (recuit simulé) et [105] (recherche tabou). La majorité de ces procédures peuvent être adaptées à la résolution du problème $(ORD^r)+(C^s)$ en *ajustant simplement la façon avec laquelle une (s)(vp) est décodée pour tenir compte des contraintes de* $(ORD^r)+(C^s)$.

Dans [139], Valls et al. ont introduit l'opération de *justification*. Globalement, une *justification à droite* (right justification) (respectivement *à gauche* (left justification)) d'un planning P consiste à ordonner les activités par ordre décroissant de leur dates de fin (respectivement croissant de leur date de début) puis les replanifier, dans cet ordre, le plus tard (respectivement le plus tôt) possible pour obtenir un nouveau planning *réalisable* P^D (respectivement P^G). Après cette opération, P^D (respectivement P^G) sera *aussi bon ou meilleur* que P. Généralement, une *double justification* (double justifying) est appliquée à un planning réalisable P pour obtenir le planning $(P^D)^G$. D'autres techniques très similaires à la justification sont résumées en [90], parmi elles nous pouvons citer aussi la technique de Tormos et al. [137].

Ces techniques peuvent être facilement étendues au problème $(ORD^r)+(C^s)$ avec, comme fonction objectif, la durée totale du projet. Pour cela, il suffit de respecter les contraintes de $(ORD^r)+(C^s)$ lorsque les activités sont replanifiées.

Klein [79] a introduit la notion de *bidirectionnal planning* pour le problème $RCPSP$.

Elle utilise simultanément les deux modes de planification forward et backward pour construire un planning. Chaque activité est planifiée selon un des deux modes, ensuite, les plannings partiels obtenus selon chaque mode sont fusionnés pour construire un planning final réalisable. Puisque les modes de planification forward et backward ont été définis pour le problème $(ORD)+(C^s)$ (voir les sections 4.3 et 4.3.2), cette méthode peut également être étendue au problème.

4.5 Représentation en séquences d'ensembles d'activités

Étant donné que la notion de (s)(vp) a été complètement définie pour le problème $(ORD) + (C^s)$, nous avons tous les éléments nécessaires pour généraliser la notion de séquence d'ensembles d'activités (se) pour $(ORD) + (C^s)$.

Dans la section 2.3.1, nous avons introduit l'algorithme ConstruireSEPRMT (voir la Figure 2.3) pour construire des (se) ayant la propriété (PRMT) pour le problème $RCPSP$. Cet algorithme peut être facilement étendu au problème $(ORD)+(C^s)$ en redéfinissant la quantité $T_{PR}(i)$ comme étant *la date au plus tôt où l'activité i peut être planifiée en considérant toutes les contraintes du problème.* Pour montrer que l'algorithme ainsi obtenu construit bien des (se)(PRMT) pour le problème $(ORD)+(C^s)$, il est possible de suivre les mêmes étapes de la démonstration du Théorème 1 (voir la section 2.3.1).

À titre d'exemple, nous reconsidérons le problème $(ORD)+(C^s)$ décrit sur la Figure 4.1. En appliquant l'algorithme à la (s)

$$s_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

nous obtenons la (se)(PRMT)

$$se_1 = [\{1, 2, 3\}\{4, 5\}\{6, 7\}\{8, 9, 10\}].$$

Le Tableau 4.1 montre les différents ordres possibles pour chacun des 4 ensembles de se_1. Ainsi, il est possible de construire $2 \times 2 \times 2 \times 1 = 8$ différentes (s)(vp) qui correspondent toutes au même planning de la Figure 4.2.

Ainsi la notion de (se)(PRMT) peut aussi être utilisée pour le problème $(ORD)+(C^s)$ afin de *réduire l'espace de recherche induit par les (s)(vp) en évitant celles qui engendrent le même planning.* Un raisonnement similaire peut être suivi pour montrer que

Ensembles	Ordres
$\{1,2,3\}$	$[1,2,3]$, $[1,3,2]$
$\{4,5\}$	$[4,5]$, $[5,4]$
$\{6,7\}$	$[6,7]$, $[7,6]$
$\{8,9,10\}$	$[8,9,10]$

TAB. 4.1 – Ordres correspondant à chaque ensemble de se_1

les (se)(PRMTb) peuvent aussi être construites avec le mode *backward* en généralisant l'algorithme ConstruireSEPRMTb pour le problème $(ORD)+(C^s)$.

Le résultat du Théorème 2 (voir la section 2.3.2) peut aussi être généralisé pour le problème $(ORD^r)+(C^s)$ ayant comme objecif la minimisation de la durée totale du projet. En effet, si une (se)(PRMT) possède un unique ensemble, alors la solution optimale du problème $(ORD^r)+(C^s)$ peut être obtenue avec la méthode CPM.

Enfin, nous avons montré à travers le Théorème 3 (voir la section 2.3.2), qu'il existe une (se)(PRMT) engendrant une solution optimale pour un problème $RCPSP$. Pour un problème $(ORD^r)+(C^s)$ ayant une solution optimale, une telle (se)(PRMT) existe *sous la condition du Corrolaire* 2 (tout planning réalisable est extensible globalement). En effet, sous cette condition, il existe une (s)(vp) donnant une solution optimale au problème $(ORD^r)+(C^s)$. Il suffit d'appliquer la version modifiée de l'algorithme ConstruireSEPRMT à cette (s)(vp) pour obtenir une (se)(PRMT) engendrant une solution optimale du problème.

Conclusion

Nous présentons une nouvelle représentation sous forme de séquence d'ensembles d'activités pour le $RCPSP$. Les meilleures heuristiques pour le problème utilisent la représentation en séquence d'activités. Nous montrons comment la nouvelle représentation peut être considérée comme une généralisation des séquences d'activités et comment une même séquence d'ensembles d'activités peut correspondre à plusieurs séquences d'activités différentes engendrant un même planning. Cette propriété permet de construire une nouvelle catégorie d'heuristiques utilisant un espace de recherche beaucoup plus réduit induisant ainsi plus d'efficacité dans la recherche des meilleurs plannings.

Plusieurs résultats numériques, issus de l'utilisation de la nouvelle représentation dans deux des meilleures approches de résolution pour le problème, montrent l'efficacité des séquences d'ensembles d'activités. La performance des deux approches en question a été améliorée par rapport à plusieurs critères.

Nous définissons par la suite une très large généralisation du problème $RCPSP$ pour laquelle une extension de l'utilisation des séquences d'activités et des séquences d'ensembles d'activités est faite. Nous montrons comment ces deux représentations peuvent être décodées en un planning réalisable et nous donnons des conditions suffisantes permettant de garantir l'existence d'une séquence d'activités et d'une séquence d'ensembles d'activités engendrant un planning optimal. Dans ce cas aussi, nous indiquons comment les séquences d'ensembles d'activités permettent de réduire l'espace de recherche induit par les séquences d'activités.

Pour cette généralisation, nos résultats démontrent que la majorité des approches de résolution dédiées au $RCPSP$ et basées sur les séquences d'activités, peuvent être réutilisées pour résoudre le problème généralisé. Il suffit pour cela d'utiliser la procédure de décodage généralisée que nous proposons au lieu de la procédure classique dédiée au $RCPSP$. De plus, il est tout à fait possible de résoudre, sous certaines conditions, de petites instances de la généralisation à l'optimum en énumérant toutes les séquences d'activités possibles. Enfin, la performance de toutes ces approches peut être améliorée en y introduisant les séquences d'ensembles d'activités qui peuvent considérablement réduire l'espace de recherche induit par les séquences d'activités.

Nous avons illustré comment les séquences d'ensembles d'activités peuvent être exploitées dans deux types de transformations. Dans une adaptation du recuit simulé, nous avons exploité la nouvelle représentation pour déterminer les séquences d'activités voi-

sines en modifiant la position des activités dans la séquence de départ, alors que dans une adaptation des algorithmes génétiques, nous l'avons exploité pour déterminer les points de croisement de deux séquences d'activités.

Comme pistes de recherche, il serait intéressant d'explorer comment la nouvelle représentation peut être utilisée dans d'autres types de transformations sur les séquences d'activités. En particulier, comment dans un algorithme génétique elle peut permettre de générer une meilleure population initiale et de définir des croisements plus adaptés au $RCPSP$ et à sa généralisation. Aussi, il serait également intéressant d'explorer l'extension des séquences d'ensembles d'activités au mode "bidirectional planning" combinant les modes forward et backward et de voir comment cette représentation pourrait réduire le nombre de séquences d'activités à énumérer pour retrouver celle induisant une solution optimale pour de petites instances. Notons que ces voies de recherche peuvent aussi bien être exploitées pour la résolution du $RCPSP$ que pour sa généralisation.

Bibliographie

[1] Alcaraz J., Maroto C., *A Robust Genetic Algorithm for Resource Allocation in Project Scheduling*, Annals of Operations Research (102) 2001, 83-109

[2] Alcaraz J., Maroto C., Ruiz R., *Improving the performance of genetic algorithms for RCPS problem*, in : Proceedings of the Ninth International Workshop on Project Management and Scheduling, Nancy 2004, 40-43

[3] Alvarez-Valdes R., Tamarit J.M., *Heuristic algorithms for resource-constrained project scheduling : A review and empirical analysis*, in : R. Sowinski, J. Weglarz (Eds), Advances in Project Scheduling, Elsevier, Amterdam 1989, 114-134

[4] Artigues C., Michelon C., Reusser S., *Insertion techniques for static and dynamic resource-constrained project scheduling*, European Journal of Operational Research (149) 2003, 249-267

[5] Aubin J., Ferland J. A., *A Large Scale Timetabling Problem*, Computers and Operations Research (16) 1989, 67-77

[6] Baar T., Brucker P., Knust S., *Tabu-search algorithms and lower bounds for resource-constrained scheduling problem*, in Meta-heuristics : Advances and Trends in Local Search Paradigms for Optimization, eds. S. Voss, S. Martello, I. Osman and C. Roucairol (Kluwer Academic) 1998, 1-18

[7] Backer K. R., *Introduction to sequencing and scheduling*, Wiley, New York 1974, p 183

[8] Baptiste P., Le Pape C., Nuijten W., *Satisfiability tests and time-bound adjustments for cumulative scheduling problems*, Annals of Operations Research (92) 1999, 305-333

[9] Bein W.W., Kamburowski J., Mstallmann M.F., *Optimal reduction of two-terminal directed acyclic graphs*, SIAM Journal on Computing (21) 1992, 1112-1129

[10] Bell C.A., Park K., *Solving resource-constrained project scheduling problem A* search*, Naval Research Logistics (37) 1990, 61-84

[11] Blazewicz J., Lenstra J., Rinnoy Kan A., *Scheduling projects to resource constraints : Classification and complexity*, Discrete Applied Mathematics (5) 1983, 11-24

[12] Boctor F.F., *Heuristics for Scheduling Projects with Resource Restrictions and Several Resource-Duration Modes*, Internation Journal of Production Research (31) 1993, 2547-2558

[13] Boctor F.F., *A new effcient heuristic for scheduling projects with resource restrictions and multiple execution modes*, European Journal of Operational Research (90) 1996, 349-361

[14] Boctor F.F., *Some effcient multi-heuristic procedures for resource-constrained project scheduling*, European Journal of Operational Research (49) 1990, 3-13

[15] Boctor F.F., *Resource-constrained project scheduling by simulated annealing*, International Journal in Production Research (34) 1996, 2335-2351

[16] Bouleimen M., Lecocq H., *A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple version*, European Journal of Operational Research (149) 2003, 268-281

[17] Bowman E.H., *The schedule sequencing problem*, Operations Research (7) 1959, 621-624

[18] Brand J.D., Meyer W.L., Shaffer L.R, *The resource scheduling problem in construction*, Civil Engineering Studies, Construction Research Series No. 5, University of Illinois, Urbana 1964

[19] Brucker P., Drexl A., Mohring R., Neumann K., *Resource-constrained project scheduling : Notation, classification, models and methods*, European Journal of Operational Research (112) 1999, 3-41

[20] Brucker P., Jurisch B., Sievers B., *A branch and bound algorithm for the job-shop problem*, Discrete Applied Mathematics (49) 1994, 107-127

[21] Brucker P., Knust S., *A linear programming and constraint propagation-based lower bound for the RCPSP*, European Journal of Operational Research (127) 2000, 355-362

[22] Brucker P., Knust S., *Resource-Constrained Project Scheduling and Timetabling*, In Lecture Notes in Computer Science 2079 PATAT III, Bruke E. and Erben W., Eds, Springer 2001, 277-293

[23] Brucker P., Knust S., *Lower bounds for resource-constrained project scheduling problems*, European Journal of Operational Research (149) 2003, 302-313

[24] Brucker P., Knust S., Schoo A., Thiele O., *A branch-and-bound algorithm for the resource constrained project scheduling problem*, European Journal of Operational Research (107) 1998, 272-288

[25] Brucker P., Schumacher D., *A New Tabu Search Procedure for Audit-Scheduling Problem*, Journal of Scheduling (2) 1999, 157-173

[26] Burt J.M., Garman M.B., *Conditional Monte Carlo : A simulation technique for stochastic network analysis*, Management Science (18) 1971, 207-217

[27] Carlier J., Néron E., *On linear bounds for the resource constrained project scheduling problem*, European Journal of Operational Research (149) 2003, 314-324

[28] Carruthers J.A., Battersby A., *Advances in critical path methods*, Operational Research Quarterly (16) 1966, 559-586

[29] Cerny V., *Thermodynamical Approach to the Traveling Salesman Problem : an Efficient Simulation Algorithm*, Journal of Optimization Theory and Applications (45) 1985

[30] Cesta A., Oddi A., *A Constraint-Based Method for Project Scheduling with Time-Windows*, Journal of Heuristics (8) 2002, 109-136

[31] Chee-Kit L., *Neural methods in combinatorial optimization*, Computers and Operations Research (19) 1992, 191-208

[32] Cho J. -H., Kim Y. -D., *A simulated annealing algorithm for resource-constrained project scheduling problems*, Journal of the Operational Research Society (48) 1997, 735-744

[33] Christofides N., Alvarez-Valdes R., Tamarit M., *Project scheduling with resource constraints : A branch and bound approach*, European Journal of Operational Research (29) 1987, 262-273

[34] Coelho J., Tavares L., *Comparative analysis of metaheuristics for the resource-constrained project scheduling problem*, Technical report, Department of Civil Engineering. Instito Superior Tecnico, Portugal 2003

[35] Colorni A., Dorigo M., Maniezzo V., *Distributed Optimization by ant colonies*, In Toward a Practice of Autonomous Systems : Proceedings of the First European Conference on Artificial Life (ECAL 91), edited by F. Varela and P. Bourgine, Cambridge, England : MIT Press, 134-142

[36] Cooper D.F., *Heuristics for scheduling resource-constrained projects : An experimental comparaison*, Management Science (22) 1976, 1186-1194

[37] Davis E.W., Heidorn G.E., *An algorithm for optimal project scheduling under multiple resource constraints*, Management Science (27) 1971, B803-B816

[38] Davis E.W., Patterson J.H., *A comparison of heuristic and optimal solutions in resource-constrained project scheduling*, Management Science (21) 1975, 944-955

[39] De Reyck B., *On the use of restrictiveness as a measure of complexity for resource constrained project scheduling*, Research Report 9535, Department of Applied Economics, K.U. Leuven,1995

[40] De Reyck B., Harroelen W., *On the use of the complexity index as the measure of complexity in activity networks*, European Journal of Operational Research (91) 1996, 347-366

[41] De Reyck B., Herroelen W., *A branch-and-bound procedure for resource-constrained project scheduling problem with generalized precedence relations*, European Journal of Operational Research (111) 1998, 152-174

[42] De Reyck B., Herroelen W., *An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations*, Computers and Operations Research (25) 1998, 1-17

[43] De Reyck B., Herroelen W., *The multi-mode resource-constrained project scheduling problem with generalized precedence relations*, European Journal of Operational Research (119) 1999, 538-556

[44] de Werra D., *An Introduction to Timetabling*, European Journal of Operational Research (19) 1985, 151-162

[45] Debels D., De Reyck B., Leus R., Vanhoucke M., *A hybrid scatter search/Electromagnetism meta-heuristic for project scheduling*, European Journal of Operational Research ((2) 169) 2006, 638-653

[46] Demeulemeester E., Herroelen W., *Project Scheduling : A research Handbook*, Kluwer Academic Publishers, Boston 2002

[47] Demeulemeester E., Herroelen W., *An effcient optimal solution procedure for the preemptive resource-constrained project scheduling problem*, European Journal of Operational Research (90) 1996, 334-348

[48] Demeulemeester E., Herroelen W., *A branch-and-bound procedure for multiple resource-constrained project scheduling problem*, Management Science (38) 1992, 1803-1818

[49] Demeulemeester E., Herroelen W., *A branch-and-bound procedure for the generalized resource-constrained project scheduling problem*, Operations Research (45) 1997, 201-212

[50] Demeulemeester E., Herroelen W., *New benchmarking results for the resource constrained project scheduling problem*, Management Science (43) 1995, 1485-1492

[51] Devroye L.P., *Inequalities for the completion times of stochastic PERT network*, Mathematics of Operations Research (4) 1979, 441-447

[52] Dobin B., *Bounding the project completion time distribution in PERT networks*, Operations Research (33) 1985, 862-881

[53] Doersch R. H., Patterson J. H., *Scheduling a project to maximize its present value : A zero-one programming approach*, Management Science (23, 8) 1977, 882-889

[54] Downwy P.J., *Distribution-free bounds on expectation on the maximum with scheduling applications*, Operations Research Letters (9) 1990, 189-201

[55] Drexl A., *Scheduling of project networks by job assignment*, Management Science (37) 1991, 1590-1602

[56] Elmaghraby S.E., Herroelen W., *On the mesurement of complexity in activity networks*, European Journal of Operational Research (5) 1980, 223-234

[57] Ferland J.A., Costa D., *Heuristic Search Methods for Combinatorial Programming Problems*, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal (# 1193) Mars 2001

[58] Fleszar K., Hindi K., *Solving the resource-constrained project scheduling problem by variable neighbourhood search*, European Journal of Operaional Research (155) 2004, 402-413

[59] Gagnon M., *Modélisation et résolution heuristique de l'allocation des ressources en gestion de projets*, Thèse de Phd, Université Laval Avril 2002

[60] Glover F., *Future Paths for Integer Programming and Links to Artificial Intelligence*, Computers and Operations Research (13) 1986, 533-549

[61] Gonçalves J., Mendes J., *A random key based genetic algorithm for the resource-constrained project scheduling problem*, Technical report, Departamento de Engenharia, Universidade do porto 2003

[62] Gondran M., Minoux M., *Graphes et Algorithmes*, Editions EYROLLES,1979

[63] Hagstrom J.N., *Computing the probability distribution of project duration in pert network*, Networks 201990, 231-244

[64] Hansen P., *The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming*, Presented at Congress on Numerical Methods in Combinatorial Optimization, Capri, Itali.

[65] Hansen P., Maldenovic N., *Variable Neighborhood Search : Principles and applications*, European Journal of Operations Research (130) 2001, 449-467

[66] Hartmann S., *A competitive genetic algorithm for resource-constrained project scheduling*, Naval Research Logistics (456) 1998, 733-750

[67] Hartmann S., *Project scheduling with multiple modes : A genetic algorithm*, Manuskripte aus den Instituten fur Betriebswirtschaftslehre der Universitat Kiel (435) 1997

[68] Hartmann S., *A self-adapting genetic algorithm for project scheduling under resource constraints*, Naval Research Logistics (49) 2002, 433-448

[69] Hartmann S., Kolisch R., *Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem*, European Journal of Operational Research (127) 2000, 394-407

[70] Herroelen W., Reyck B., Demeulemeester E., *Resource-constrained project scheduling : A survey of recent developments*, Computers and Operations Research (4) 1998, 279-302

[71] Herroelen W., Leus R., *Project Scheduling under uncertainty : Survey and research potentials*, European Journal of Operational Research (165) 2005, 289-306

[72] Hertz A., Widmer M., *Guidelines for use of meta-heuristics in combinatorial optimization*, European Journal of Operational Research (151) 2003, 247-252

[73] Hindi K. S., Yang H., Fleszar K., *An evolutionary algorithm for resource-constrained project scheduling*, IEEE Transactions on Evolutionary Computation (6) 2002, 512-518

[74] Hopfield J.J., Tank D.W., *Neural computation of descisions in optimization problems*, Biological Cybernetics (52) 1985, 141-152

[75] Kaplan L., *Resource-constrained project scheduling with setup times*, Unpublished paper

[76] Kaplan L., *Resource-constrained project scheduling with preemption of jobs*, PhD. Dissertation, University of Michigan 1988

[77] Kelley J. E., *The critical-path method : Resources planning and scheduling*, in : industrial planning scheduling, eds. J.F. Muth and G.L. Thompson (Prentice-Hall)1963, 347-365

[78] Kirkpatrick S., Gelatt C.D.Jr., Vecchi M.P., *Optimization by Simulated Annealing*, Science (220) 1983, 671-680

[79] Klein R., *Bidirectional planning : improving priority rule-based heuristics for scheduling resource-constrained projects*, European Journal of Operational Research (127) 2000, 619-638

[80] Klein R., *Project scheduling with time-varying resource constraints*, International Journal of Production Research, 38 (16) 2000, 3937-3952

[81] Klein R., Scholl A., *Computing lower bounds by destructive improvement : An application to resource-constrained project scheduling*, European Journal of Operational Research (112) 1999, 322-346

[82] Kochetov Y., Stolyar A., *Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem*, in : Proceedings of the 3rd International Workshop of Computer Science and Information Technologies, Russia 2003

[83] Kohlmorgen U., Schmeck H., Haase K., *Experiences with fine-grained parallel genetic algorithms*, Annals of Operations Research (90) 1999, 203-219

[84] Kolisch R., Drexl A., *Adaptative search for solving hard project scheduling problem*, Naval Research Logistics (43) 1996, 23-40

[85] Kolisch R., *Project scheduling under resource constraints - Effcient Heuristics for Several Problem Classes*, Physica, Heidelberg 1995

[86] Kolisch R., *Efficient priority rules for the resource-constrained project scheduling problem*, Journal of Operations Management (14) 1996, 179-192

[87] Kolisch R., *Serial and Parallel resource-constrained project scheduling methods revisited : Theory and computation*, European Journal of Operational Research (90) 1996, 320-333

[88] Kolisch R., Drexl A., *Adaptative search for solving hard project scheduling problems*, Naval Research Logistics (43) 1996, 23-40

[89] Kolisch R., Drexl A., *Local Search for nonpreemptive multi-mode resource-constrained project scheduling*, IEE Transactions (29) 1997, 987-999

[90] Kolisch R., Hartmann S., *Experimental investigation of heuristics for resource-constrained project scheduling : An update*, European Journal of Operational Research, to appear 2005

[91] Kolisch R., Hartmann S., *Heuristic algorithms for solving resource-constrained project scheduling problem : Classification and computation analysis*, in : J. Weglarz (Ed.), Project Scheduling : Recent Models, Algorithms and Applications, Kluwer Academic Publisher, Boston 1999, 147-178

[92] Kolisch R., Padman R., *An integrated survey of deterministic project scheduling*, OMEGA International Journal of Management Science 29 (3) 2001, 249-272

[93] Kolisch R., Sprecher A., Drexl A., *Characterization and generation of general class of resource-constrained project scheduling problems*, Management Science (41) 1995, 1693-1703

[94] Kramer A., *Branch and bound methods for scheduling problems with multiprocessor tasks on dedicated processors*, OR Spektrom (19) 1997, 219-227

[95] Leon V.J., Ramamoorthy B., *Strength and adaptability of problem-space based neighborhoods for resourse-constrained scheduling*, OR Spektrum (17) 1995, 173-182

[96] Li K. Y., Willis R. J., *An interactive scheduling technique for resource-constrained project scheduling problem*, European Journal of Operational Research (56) 1992, 370-379

[97] Mastor A.A., *An experimental and comparative evaluation of production line balancing techniques*, Management Science (16) 1970, 728-746

[98] Merkle D., Middendorf M., Schmeck H., *Ant colony optimization for resource-constrained project scheduling*, IEEE Transactions on Evolutionary Computation (6) 2002, 333-346

[99] Mingozi A., Maniezzo V., Ricciardelli S., Bianco L., *An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation*, Management Science (44) 1998, 714-729

[100] Mladenovic N., Hansen P., *Variable Neighborhood Search*, Computers and Operations Research (24) 1997, 1097-1100

[101] Mohring R.H., Radermacher F.J., Weiss G., *Stochastic scheduling problems I - Set strategies*, Zeitschrift Fur Operations Research Ser. (A28) 1984, 193-260

[102] Mohring R.H., Radermacher F.J., Weiss G., *Stochastic scheduling problems II - Set strategies*, Zeitschrift Fur Operations Research Ser. (A29) 1985, 65-104

[103] MOUMENE K., *Le problème de l'emploi du temps, approche théorique et stratégies pratiques*, Mémoire de Magister (Master), Université des Sciences et de la Technologie Houari Boumediene, Septembre 2001

[104] Moumene K., Ferland J. A., *Activity Set List Representation for the Resource-Constrained Project Scheduling Problem*, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal (Publication no 1223) March 2005

[105] Nonobe K., Ibaraki T., *Formulation and tabu search algorithm for the resource constrained project scheduling problem*, in : C.C. Ribeiro, P. Hansen (Eds) . Essays and Surveys in Metaheuristics, Kluwer Academic Publishers 2002, 557-588

[106] Nudtasomboon N., Randhawa S. U., *Resource-constrained project scheduling with renewable and non-renewable resources and time-resource tradeoffs*, Computers & Industrial Engineering (32) 1997, 227-242

[107] Oguz O., Bala H., *A comparative study of computational produces for the resource constrained project scheduling problem*, European Journal of Operational Research (72) 1994, 406-416

[108] Ozdamar L., *A genetic algorithm approach to a general category project scheduling problem*, Research Report, Mamara University, Istanbul 1996

[109] Ozdamar L., Ulusoy G., *A note on interactive forward/backward scheduling technique with a reference to a procedure of Li and Willis*, European Journal of Operational Research (89) 1996, 400-407

[110] Palpant M., Artingues C., Michelon P., *LSSPER : Solving the resource-constrained project scheduling problem with large neighbourhood search*, Annals of Operations Research (131) 2004, 237-257

[111] Pascoe T.L., *Allocation of resources-CPM*, Revue Française de Recherche Opérationnelle (38) 1996, 31-38

[112] Patterson J. H., Slowinski R., Talbot F. B., Weglarz J., *Computational experience with a backtracking algorithm for solvimg a general class of precedence and resource-constrained scheduling problem*, European Journal of Operational Research (49) 1990, 68-79

[113] Patterson J.H., *Project scheduling : The effects of problem structure on heuristic performance*, Naval Research Logistics (23) 1976, 95-123

[114] Patterson J.H., *A comparison of exact approaches for solving the multiple constrained resource project scheduling problem*, Research Report, Department of Industrial and Manufacturing System Engineering, Lehigh University 1984

[115] Patterson J.H., Slowinski R., Talbot F.B., Weglarz J., *An algorithm for a general class of precedence and resource constrained scheduling problems*, in :R. Sowinski, 17.J.Weglarz (Eds), Advances in project scheduling, Elsevier, Amsterdam 1989, 3-28

[116] Pinson E., Prins C., Rullier F., *Using tabu search for solving the resource-constrained project scheduling problem*, in : Proceedings of the 4th International Workshop on Project Management and Scheduling, Leuven, Belgium 1994, 102-106

[117] Provan J.S., Ball M.O., *The complexity of counting cuts and the probability that a graph is connected*, SIAM Journal and Computing (12) 1983, 777-788

[118] Sampson S.E., Weiss E.N., *Local search techniques for the generalized resource-constrained project scheduling problem*, Naval Research Logistics (40) 1993, 665-675

[119] Schaerf A., *Local Search Techniques for Large High School Timetabling Problems*, IEEE Transactions on Systems, Man and Cybernetics, Part (A29) 1999, 368-377

[120] Schaerf A., *A Survey of Automated Timetabling*, Artificial Intelligence Review (13) 1999, 87-127

[121] Schirmer A., *Case-based reasoning and improved adaptive search for project scheduling*, Naval Research Logistics (47) 2000, 201-222

[122] Schirmer A., Riesenberg S., *Class-based control schemes for parameterized project scheduling heuristics, Working paper*, Instituten fur Betriebswirtschaftslehre, Universitat Kiel, Kiel, Germany 1998

[123] Sigal C.E., Pritsker A.A.B., Solberg J.J., *The use of cutset in Monte Carlo analysis of stochastic networks*, Mathematics and Computers in Simulation (21) 1979, 379-384

[124] Slowinski R., *Multiobjective project scheduling under multiple-category resource constraints*, In : R. Slowinski and J. Weglarz (eds.), Advances in Project Scheduling, Elsevier, Amsterdam 1989, 151-167

[125] Slowinski R., Soniewicki B., Weglarz J., *DSS for multiple-objective project scheduling subject to multiple-category resource constraints*, European Journal of Operational Research (79) 1994, 220-229

[126] Soroush H., *Risk taking in stochastic PERT networks*, European Journal of Operational Research (67) 1993, 221-241

[127] Soroush H.M., *The most critical path in PERT network*, Journal of Operational Research Society (45) 1994, 287-300

[128] Sprecher A., Drexl A., *Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm*, European Journal of Operational Research (107) 1998, 431-450

[129] Sprecher A., Hartmann S., Drexl A., *An exact algorithm for project scheduling with multiple modes*, OR Spektrum (19) 1997, 195-203

[130] Sprecher A., Kolisch R., Drexl A., *Semi-Active, active and non-delay schedules for resource-constrained project scheduling problem*, European Journal of Operational Research (80) 1995, 94-102

[131] Stinson J.P., Davis E.W., Khumawala B.M., *Multiple resource-constrained scheduling using branch-and-bound*, AIIE Transactions (10) 1978, 252-259

[132] Sullivan R.S., Hayya J.C., *A comparison of the method of bounding distribution (MBD) and Mote Carlo simulation for analyzing stochastic acyclic networks*, Operations Research (28) 1980, 614-617

[133] Talbot B., Patterson J.H., *An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problem*, Management Science (24) 1978, 1163-1174

[134] Thesen A., *Heuristic scheduling of activities under resource and precedence restrictions*, Management Science (23) 1976, 412-422

[135] Tormos P., Lova A., *An efficient multi-pass heuristic for project scheduling with constrained resources*, International Journal of Production Research 41 (5) 2003, 1071-1086

[136] Tormos P., Lova A., *Integrating heuristics for resource constrained project scheduling : one step forward. Technical report*, Department of Statistics and Operations Research, Universidad Politecnica de Valencia 2003

[137] Tormos S., Lova .A, *A competitive heuristic solution technique for resource-constrained project scheduling*, Annals of Operations Research (102) 2001, 65-81

[138] Valls V., Ballestin F., Quintanilla M.S., *A hybrid genetic algorithm for the RCPSP*, Technical report, Department of Statistics and Operations Research, University of Valencia 2003

[139] Valls V., Ballestin F., Quintanilla S., *Justification and RCPSP : A technique that pays*, European Journal of Operational Research (165) 2005, 375-386

[140] Van Slyke R.M., *Monte Carlo methods and the PERT problem*, Operations Research (11) 1963, 860-893

[141] Wilcoxon F., *Individual comparisons by ranking methods*, Biometrics (1) 1945, 80-83

[142] Youssef H., Sadiq M.S., Adiche H., *Evolutionary algorithms, simulated annealing and tabu search : a comparative study*, Engineering Application of Artificial Intelligence (14) 2001, 167-181