Université de Montréal

Allocation et tarification des accès réseaux

par
Walid Jamoussi

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

Mémoire présenté à la faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en informatique

Décembre 1999

Université de Montréal
Faculté des études supérieures

Ce mémoire de Maîtrise intitulé

Allocation et tarification des accès réseaux

présenté par
Walid Jamoussi

a été évalué par un jury composé des personnes suivantes:

Président:	Michel Gendreau
Directeur de recherche:	Rachida Dssouli
Co-directeur:	Jacques Robert
Membre:	Patrice Marcotte

À mes parents...

SOMMAIRE

Les voies de transmission d'un réseau de télécommunication longue distance *WAN* (*Wide Area Network*) sont généralement coûteuses et se caractérisent par une large capacité de transmission qu'on appelle *bande passante* ou *bande de fréquences*. On est donc amené à partager ces voies entre plusieurs utilisateurs afin de les exploiter plus rationnellement et afin de bénéficier d'économies d'échelle. Ce partage est effectué selon une technique de multiplexage fréquentiel FDM (*Frequency Division Multiplexing*). Cette technique consiste à partager la bande passante disponible en n canaux et à affecter en permanence chacun d'eux à un utilisateur. En contre-partie, ce dernier paye une certaine somme d'argent pour avoir le droit de transmission sur le canal. Dans le cas où une voie de transmission est divisée en n canaux alors que le nombre des utilisateurs est de loin supérieur à n, la politique de fixation de prix pour l'acquisition d'un canal devient importante. Une façon naturelle pour résoudre ce problème est d'utiliser une approche économique dans l'ingénierie des systèmes : *les mécanismes d'enchères*.

Motivé par le concept de bande passante, *Semret et al.* proposent dans [LS 97], une règle d'enchère appelée *règle du second prix progressif* ou *PSP* (*Progressive Second Price*) pour le partage de la bande passante d'un lien de transmission entre plusieurs utilisateurs. Cette règle est une mise en oeuvre d'un mécanisme défini dans la littérature économique pour un problème très général d'allocation et de fixation de prix. Ce mécanisme, appelé mécanisme de Clark-Groves, est inspiré des travaux de Vickrey sur les enchères.

En se basant sur la règle *PSP*, *Semret et al.* proposent un algorithme qui étend celle-ci à un réseau de liens ayant une structure d'arbre. Dans ce contexte, un utilisateur mise sur une combinaison de liens formant un chemin continu et il spécifie le nombre d'unités de bande passante qu'il désire obtenir pour son chemin ainsi que le prix unitaire qu'il est prêt à payer, le prix de chaque lien n'étant pas spécifié. L'inconvénient de cet algorithme est que les utilisateurs sont contraints de miser sur des chemins continus partant nécessairement de la racine de l'arbre.

Ce projet de maîtrise est divisée en deux parties : dans la première partie, nous proposons une version améliorée de l'algorithme proposé par *Semret et al.* pour le partage de la bande passante dans un arbre où nous permettons aux utilisateurs de spécifier un prix pour chaque unité demandée et non pas un seul prix unitaire. Dans la deuxième partie, nous proposons un deuxième algorithme qui se base entièrement sur les techniques de la programmation linéaire pour le partage de la bande passante dans un réseau acyclique selon le même principe utilisé dans le cas d'un arbre tout en permettant aux utilisateurs de miser sur des chemins entre des noeuds quelconques du réseau sans aucune restriction.

Mots clés : Allocation de ressources, accès aux canaux, bande passante, théorie des jeux, mécanismes d'enchères.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

REMERCIEMENTS

Je tiens à exprimer mes profonds remerciements à ma directrice Rachida Dssouli et mon co-directeur Jacques Robert pour leur soutien permanent et l'aide précieuse qu'ils m'ont fournie durant tout mon projet.

Egalement, je tiens à remercier, Robert Gérin-Lajoie, professionnel de recherche au Centre Interuniversitaire de Recherche en Analyse des Organisations (CIRANO) pour ses interventions encourageantes qui m'ont aidé à orienter mes efforts au début de mes travaux.

Un grand merci au CIRANO et au laboratoire universitaire Bell Canada pour le soutien financier qu'il m'ont accordé durant toute la période de mon projet.

CHAPITRE 1

Introduction

Les économies d'échelle jouent un rôle important dans l'industrie des télécommunications. Les coûts de mise en oeuvre et d'entretien d'une ligne de transmission à haut débit, c'est-à-dire à large bande passante, sont pratiquement les mêmes que ceux d'une ligne à faible débit, le coût du support physique de transmission (qu'il soit en cuivre ou en fibre optique) étant en effet négligeable par rapport au coût des travaux de génie civil nécessaires à sa mise en place. Du fait de ces considérations économiques, les exploitants de réseaux ont favorisé la mise en place de systèmes très élaborés qui permettent de partager (ou de multiplexer) entre de nombreux usagers le même support physique de transmission. Les techniques de multiplexage des moyens de transmission peuvent être classées en deux grandes catégories : le multiplexage temporel, *MRT* (*Multiplexage à répartition dans le temps*) ou *TDM* (*Time Division Multiplexing*) et le multiplexage fréquentiel, *MRF* (*Multiplexage par répartition de fréquences*) ou *FDM* (*Frequency Division Multiplexing*) [Tan 96, Hal 96]. Le multiplexage temporel consiste à affecter à chaque utilisateur, pendant un court instant et à tour de rôle, la totalité de la bande passante disponible. Quant au multiplexage fréquentiel, il consiste à partager la bande passante disponible en n canaux plus étroits et à affecter en permanence chacun d'eux à un utilisateur exclusif. Chaque canal peut être utilisé indépendamment des autres. Par exemple, sur une ligne de bande passante

6400 Hz, chaque canal disposera d'une largeur de bande de 6400/*n* Hz. Une condition nécessaire et suffisante pour utiliser la technique FDM est que chaque canal doit utiliser une bande de fréquence propre, suffisamment éloignée des autres canaux pour éviter les interférences.

Le multiplexage FDM est très utilisé en pratique aussi bien sur les lignes de communications électrique (transmission filaire) que radioélectriques (transmission sans fil). Les émissions de radio en AM, par exemple, constituent une bonne illustration de multiplexage fréquentiel. La largeur de la bande passante disponible est de l'ordre d'un mégahertz. Elle se situe approximativement entre 500 kHz et 1,5MHz. Différentes fréquences porteuses appartenant à cette bande de fréquences sont affectées à différentes stations de radiodiffusion, en maintenant toutefois entre elles un espace suffisant pour éviter les interférences. Aussi, les liens d'un réseau de télévision câblé sont divisés en multiples canaux de transmission à 6MHz de bande affectés aux différentes chaînes de télévision. Le partage fréquentiel est aussi utilisé en téléphonie, il permet de réaliser économiquement des liaisons à longue distance par multiplexage des voies à grande capacités telles que des câbles coaxiaux. Lorsque plusieurs canaux téléphoniques sont multiplexés sur un même support de transmission, des filtres appropriés limitent la bande passante utilisable à 3000 Hz par canal téléphonique. Cependant, il est attribué à chacun d'eux une bande de fréquences large de 4000 Hz afin de bien séparer les uns des autres. La mise en oeuvre de multiplexage fréquentiel de signaux téléphonique analogiques est, à quelques exceptions près, la même dans tous les pays du monde; elle est conforme aux normes de l'*UIT* (*Union Internationale des Télécommunications*). L'une d'entre elles, très répandue, consiste à regrouper 12 voies téléphoniques (4000 Hz par voie, soit 3000 Hz par utilisateur et deux espaces inter-bandes de 500 Hz , ce qui conduit à une largeur de bande de 48 kHz) et à les répartir entre 60 et 108 kHz.

Etant donné un canal de transmission multiplexé en plusieurs canaux, le plus souvent, ceux qui sont en compétition pour acquérir un canal ne sont pas des utilisateurs finaux, ce sont plutôt des fournisseurs de services (compagnies privées) qui, pour des raison purement commerciales, achètent le droit de transmission sur un canal et utilisent à leur tour une technique de multiplexage fréquentiel et/ou temporel pour offrir des services de communication (téléphonie, multimedia, vidéo ou télévision à la demande) aux utilisateurs finaux. Dans de telles situations, une fois que la demande de bande passante dépasse l'offre, la politique d'allocation des canaux devient importante. La façon naturelle et la plus directe pour ce type d'allocation est d'utiliser une approche économique dans l'ingénierie des systèmes: les *mécanismes d'enchères.*
Etant donnée une ressource mise aux enchères et un ensemble d'acheteurs en compétition pour acquérir la ressource, un mécanisme d'enchère consiste en:
- *Une soumission des mises*: chaque acheteur soumet une offre dans laquelle il spécifie le prix qu'il est prêt à payer pour obtenir la ressource.
- *Une allocation de la ressource*: la ressource est allouée au plus fort offrant à un prix qui dépend du type d'enchère. En pratique, deux types d'enchères sont communément utilisées, l'une est dite au *premier prix*, l'autre au *second prix,* appelée aussi *enchère de Vickrey* [FT 91, Mye 81, Vick 61]. Dans l'enchère au premier prix, le prix payé est égal à la mise la plus élevée, alors que dans l'enchère au second prix, il est égal à la seconde meilleure mise.
Le caractère équitable des mécanismes d'enchères en fait un cadre particulièrement attrayant, cadre qui est d'ailleurs la doctrine officielle de l'administration en matière d'achat ou de vente publics. En juillet 1994, la *FCC* (*Federal Communications Commission*) a utilisé les mécanismes d'enchères pour l'allocation des licences d'utilisation du spectre fréquentiel pour la communication sans fil [FCC, Cra 94]. Les mécanismes d'enchères ont été aussi utilisés pour l'allocation des ressources de communication dans [Ago 94, WHS 92]

Motivés par le concept de bande passante, *Semret et al.* proposent une nouvelle règle d'enchère appelée *règle du second prix progressif* ou *PSP* (*Progressive Second Price*) pour l'allocation des ressources partageables [LS 97, LS 98a, LS 98b], et en particulier, pour le partage de la bande passante d'une voie de transmission entre plusieurs utilisateurs. Celle-ci est implantée sur le *World Wide Web* sous forme d'un jeu interactif entre plusieurs joueurs appelé *T-REX* (*Telecommunication Resource EXchange*) [Sem 96]. Dans sa version actuelle, ce jeu considère une seule ressource multi-unitaire; chaque joueur soumet une mise par laquelle il spécifie la quantité de la ressource qu'il désire obtenir ainsi que le prix unitaire qu'il est prêt à payer. En fonction de l'ensemble des mises, la règle *PSP* calcule l'allocation de chaque joueur ainsi que le prix unitaire à payer. La fixation des prix est effectuée selon le principe du second prix.

L'enchère selon la règle *PSP* est une mise en oeuvre d'un mécanisme défini dans la littérature économique pour un problème très général d'allocation et de fixation de prix. Ce mécanisme, appelé mécanisme de Clark-Groves, est inspiré des travaux de Vickrey sur les enchères [Cla 71, Gro 73, MWG 95].

Dans l'objectif d'une allocation simultanée de la bande passante au niveau d'un ensemble de liens de transmission interconnectés et non pas au niveau d'un seul lien, *Semret et al.* ont étudié la possibilité d'extension de la règle *PSP* à un ensemble de ressources divisibles qui sont complémentaires [LS97]. Dans ce nouveau contexte, un utilisateur mise sur une combinaison de liens; il spécifie le nombre d'unités demandées ainsi que le prix unitaire qu'il est prêt à payer, le prix de chaque lien n'étant pas spécifié. En donnant au réseau la structure d'un arbre, et en supposant que les utilisateurs misent sur des chemins continus partant nécessairement de la racine de l'arbre vers des noeuds quelconques, *Semret et al.* proposent une règle d'enchère pour l'allocation simultanée de la bande passante de l'ensemble des liens. Le principe de celle-ci est de faire appel à la règle *PSP* pour chaque lien de l'arbre, et de combiner

les allocations par un algorithme de programmation dynamique. Le seul inconvénient de cette règle est que les utilisateurs sont contraints de miser sur des chemins partant nécessairement de la racine de l'arbre.

Notre projet de maîtrise est divisé en deux parties : dans la première partie, nous nous proposons d'améliorer la version originale de l'algorithme proposé par *Semret et al.* pour le partage de la bande passante dans un arbre. Nous commençons par étudier le mécanisme d'enchère proposé dans *T-REX* ainsi que les propriétés la règle d'allocation *PSP*. Ceci nous amène à étudier les concepts fondamentaux de la *théorie des jeux* tels que la stabilité, l'utilité des joueurs, les stratégies dominantes, etc. Etant devenu familier avec les propriétés de *PSP*, nous enchaînons en proposant une nouvelle règle d'enchère appelée *DPSP* (*Dynamic Progressive Second Price*) qui est une généralisation de la règle *PSP* au cas où les utilisateurs spécifient un prix pour chaque unité demandée et non pas un seul prix unitaire. Enfin, en se basant sur cette nouvelle règle, nous proposons un algorithme *Tree_Allocation* qui est une version améliorée de l'algorithme proposé par *Semret et al.* pour le partage de la bande passante dans un arbre où nous permettons aux utilisateurs de spécifier un prix pour chaque unité demandée.

Afin de rendre les conditions de mise plus flexibles nous proposons, dans la deuxième partie de notre projet, un deuxième algorithme *AG_Allocation* (*Acyclic_Graph Allocation*) qui se base sur la programmation linéaire pour le partage de la bande passante dans un réseau acyclique selon le même principe utilisé dans le cas d'un arbre tout en permettant aux utilisateurs de miser sur des chemins entre des noeuds quelconques du réseau sans aucune restriction. Ainsi, avec cet algorithme, si on se restreint à la structure d'arbre qui est un cas particulier de graphe acyclique, l'hypothèse qui fait que les utilisateurs doivent miser sur des chemins partant nécessairement de la racine de l'arbre est levée.

Ce mémoire est organisé de la manière suivante.

Au chapitre 2, nous exposons dans un premier temps les concepts de base de la théorie des jeux et nous présentons une modélisation d'une situation d'enchère sous forme d'un jeu dit non coopératif. Ensuite, nous étudions en détails les deux types d'enchères, au premier prix et au second prix. Enfin, nous considérons un jeu particulier qui correspond à la mise en oeuvre du mécanisme Clark-Groves.

Au chapitre 3, nous commençons par étudier le mécanisme d'enchère proposé dans *T-REX* pour le partage des ressources divisibles. Nous analysons ensuite les propriétés de la règle d'allocation *PSP*, puis nous étudions l'algorithme proposé par *Semret et al.* pour l'allocation de la bande passante sur un arbre.

Au chapitre 4, nous présentons la nouvelle règle d'enchère *DPSP*; ensuite, nous proposons l'algorithme *Tree_Allocation* qui se base sur celle-ci pour le partage de la bande passante sur un arbre.

Au chapitre 5, nous étudions les possibilités d'extension de l'algorithme *Tree_Allocation* vers le cas où les utilisateurs misent sur des chemins entre des noeuds quelconques de l'arbre.

Au chapitre 6, nous présentons l'algorithme *AG_Allocation* qui se base sur la programmation linéaire pour le partage de la bande passante dans un réseau acyclique.

Au chapitre 7, nous décrivons dans un premier temps les implantations de *Tree_Allocation* et *AG_Allocation*, ensuite, nous analysons les résultats d'expérimen-

tation obtenus et nous vérifions que *Tree_Allocation* est un cas particulier de *AG_Allocation*. Enfin, nous analysons le temps d'exécution de chacun des deux algorithmes.

Nous concluons ce mémoire en résumant le travail effectué et en suggérant des extensions possibles.

CHAPITRE 2

La théorie des jeux

La théorie des jeux est l'un des secteurs les plus féconds de l'application des mathématiques en sciences sociales. Elle a pour but de modéliser sous forme de jeux des situations concrètes à priori complexes à analyser et dans lesquelles des agents rationnels sont dans un état d'interdépendance stratégique. Le champ d'application de la théorie des jeux est extrêmement vaste [She 94, MD 88, SCEH 96], il englobe en particulier les mécanismes d'enchères qui sont couramment utilisés en pratique. Dans ces dernières années, la théorie des jeux a été de plus en plus appliquée aux problèmes d'ingénierie liés aux réseaux de communications tels que le contrôle de flux [KL95], le routage [KLO 95, ORS 93] et l'accès aux canaux [KSY 85].

Ce chapitre est divisé en trois parties : la première partie expose les concepts fondamentaux de la théorie des jeux dits *non coopératifs* où toute communication entre les joueurs est exclue. Dans la deuxième partie, nous étudions une situation d'interdépendance stratégique issue d'un contexte économique : l'attribution de marchés à l'aide des mécanismes d'enchères. Enfin, dans la troisième partie nous présentons un mécanisme d'allocation de ressources inspiré des travaux de Vickrey sur les enchères appelé mécanisme de Clark-Groves.

2.1. Définition d'un jeu

Demanage et Ponssard définissent un jeu comme étant un objet mathématique formalisant une situation de conflit entre plusieurs agents (qu'on appelle *joueurs* ou *protagonistes*) [DP 94], c'est-à-dire une situation qu'ils jugent selon des préférences contradictoires et dont ils peuvent influencer certains résultats. Les membres d'une assemblée qui doivent élire l'un d'entre eux pour être le président ou bien les dirigeants de plusieurs firmes en concurrence sur un marché où ils offrent des biens substituables sont des exemples de "joueurs".

2.2. Les jeux sous forme stratégique

2.2.1. Modélisation

La représentation stratégique d'un jeu retient les éléments de base d'une situation d'interaction stratégique à savoir les *joueurs*, leurs *stratégies* disponibles et leur *fonction d'utilité* [DP 94]. Ainsi, un jeu sous forme stratégique est la donnée $(N, S_i, u_i, i \in N)$, où :

- L'ensemble $N = \{1,..., i,..., n\}$ représente l'ensemble des joueurs;
- Pour chaque joueur i, S_i est l'ensemble des possibilités de choix disponibles appelées *stratégies*. Le choix par chaque joueur d'une stratégie détermine l'*issue* d'un jeu.

Soit

s_i : une stratégie de i;

$s = (s_1, \ldots, s_n)$: une issue du jeu;

$S = \Pi_{i \in N} S_i$: l'ensemble des issues possibles;

et aussi :

$s = (s_i, s_{-i})$ où s_{-i} représente les stratégies des joueurs autres que i;

$S_{-i} = \prod_{j \neq i} S_j$: l'ensemble des stratégies des joueurs autres que i;

Pour chaque joueur i, u_i est une fonction réelle sur l'ensemble des issues S, cette fonction est appelée *fonction d'utilité* ou encore *fonction de paiement* [MWG 95, DP 94, Mou 81].

$$u_i : \begin{array}{l} S \rightarrow \mathrm{R} \\ s \rightarrow u_i(s) \end{array}$$

La fonction d'utilité est un outil de modélisation qui permet de représenter les *préférences* des joueurs vis à vis l'ensemble des issues du jeu. Autrement dit,

$u_i(s) > u_i(s')$ signifie que le joueur i préfère strictement l'issue s à l'issue s'

et

$u_i(s) = u_i(s')$ signifie qu'il est indifférent entre les deux issues.

Un joueur est dit *rationnel* s'il choisit une stratégie qui maximise son niveau d'utilité étant donné les stratégies des autres.

Les deux hypothèses suivantes sous-tendent la donnée d'un jeu sous forme stratégique:

- ***Indépendance stratégique*** : *Les joueurs sélectionnent leur stratégie indépendamment les uns des autres.*

Toute coordination formelle entre les joueurs, par exemple sous la forme d'une sélection conjointe d'une issue du jeu, est ainsi exclue. On parle alors de *jeux non coopératifs* et on dit que le comportement des joueurs est *décentralisé*. Cette hypothèse très importante est réalisée par exemple si les stratégies sont sélectionnées simultanément ou en secret.

- ***Information complète*** : *les joueurs connaissent la forme stratégique* $(N, S_i, u_i, i \in N)$.

Les joueurs ont ainsi une connaissance commune de la situation à laquelle ils sont confrontés: ils connaissent les autres joueurs, leur ensemble de stratégies et leur fonction d'utilité.

2.2.2. Exemples de jeux sous forme stratégique

Un jeu est dit fini si tous les ensembles de stratégies sont finis, il est dit infini si les ensembles de stratégies sont infinis. Dans cette section, nous présenterons deux exemples de jeux sous forme stratégique, l'un est fini : le dilemme du prisonnier, l'autre est infini: les jeux d'enchères.

2.2.2.1. Jeu fini : Le dilemme du prisonnier [DP 94]

Le dilemme du prisonnier est l'un des exemples les plus célèbres de la théorie des jeux. Deux individus, soupçonnés d'avoir accompli un sombre méfait, sont placés en garde à vue dans deux cellules différentes. Le juge d'instruction est convaincu de leur culpabilité, mais il ne dispose d'aucune preuve pour les confondre lors du procès. Le juge propose à chacun le marche suivant :
"Avoue ton crime et témoigne contre ton complice, tu bénéficieras d'une réduction de peine. Méfie-toi de lui, s'il est le seul à avouer, tu en prends pour vingt ans".
Le jeu proposé par le juge aux deux prisonniers est de type non coopératif fini. Chaque joueur dispose de deux stratégies :
- une stratégie "pacifique" P : ne pas avouer.
- une stratégie "agressive" A : dénoncer son complice.

La matrice de gain de ce jeu est la suivante :

$$\begin{array}{cc} & \textit{joueur 2} \\ & \begin{array}{cc} P & A \end{array} \\ \textit{joueur 1} \begin{array}{c} P \\ A \end{array} & \begin{bmatrix} 1,1 & -1,2 \\ 2,-1 & 0,0 \end{bmatrix} \end{array}$$

chaque élément de la matrice représente une issue du jeu où on y lit deux nombres qui sont respectivement les niveaux d'utilités des deux joueurs.

Ce jeu place les joueurs dans une situation conflictuelle intéressante. D'après la matrice de paiement, chaque joueur est naturellement tenté de faire le choix A, il a intérêt à avouer quoi que fasse son complice (puisque dans tous les cas A lui procure un meilleur paiement) ce qui lui permet d'augmenter son gain et de diminuer le gain de

l'autre. Pourtant, s'ils pouvaient se concerter, il n'auraient collectivement pas intérêt à avouer, ils gagneraient plus en faisant en commun le choix coopératif P (ils choisiraient conjointement l'issue (P, P)).

Cet exemple célèbre illustre de façon criante le conflit possible entre la rationalité individuelle et une démarche collective. De nombreuses situations économiques peuvent être représentées par le dilemme du prisonnier. Par exemple, dans une situation de duopole, la stratégie agressive correspond à un prix faible, et la stratégie pacifique à un prix élevé.

2.2.2.2. Jeu infini : *Les jeux d'enchères* [FT 91]

Un objet est mis au enchères. Tout acheteur potentiel i attribue une valeur θ_i à l'objet. Son niveau d'utilité (ou son gain) est $\theta_i - p$ s'il obtient l'objet et paye un prix p, 0 sinon.

Dans les modèles d'enchères, nous supposons que θ_i est une information privée, c'est-à-dire qu'uniquement le joueur i connaît θ_i.

Considérons les enchères sous pli scellé : chaque joueur fait une seule offre, soumise dans une enveloppe cachetée. En pratique, deux types d'enchères sont communément utilisées, l'une est dite au *premier prix*, l'autre au *second prix* appelée aussi *enchère de Vickrey* [Vick 61]. Dans les deux cas, l'objet est alloué au plus offrant. Dans l'enchère au premier prix le bénéficiaire de l'objet paye son enchère, alors que dans l'enchère au second prix, il paye seulement la seconde meilleure enchère.

Une stratégie pour le joueur i, s_i, spécifie une mise b_i pour chaque valeur possible de θ_i. Si l'ensemble des valeurs possibles de θ_i est $\Theta_i = [0, \bar{\theta}_i]$, alors une stratégie est une fonction :

$$s_i : \begin{array}{l} \Theta_i \rightarrow \mathrm{R}^+ \\ \theta_i \rightarrow b_i \end{array}$$

Etant donné une issue $b = (b_1, \ldots, b_n)$ représentant le profil des mises, ordonnons les mises :

$$b_j \geq b_k \geq \ldots$$

alors j gagne l'enchère, son gain est $\theta_j - \mathrm{p}$ où :
$\mathrm{p} = b_j$ pour l'enchère au premier prix
$\mathrm{p} = b_k$ pour l'enchère au second prix
Les gains de tous les autres joueurs sont nuls.

Etant donné les gains des joueurs, on peut calculer les gains espérés d'un joueur qui utilise une stratégie s_i étant donnée les stratégies des autres. Il est à noter que les gains espérés dépendent des fonctions de répartition des paramètres $\theta = (\theta_1, \ldots, \theta_n)$.

$$u_i(s_1, \ldots, s_n) = \int_0^{\bar{\theta}_i} (\theta_i - s_i(\theta_i)) \left[\prod_{j \neq i} \mathrm{F}_j(s_j^{-1}(s_i(\theta_i))) \right] d\mathrm{F}_i(\theta_i)$$

Ainsi, nous avons modélisé une situation d'enchère comme un jeu sous forme stratégique.

2.3. Hypothèse de rationalité

L'hypothèse fondamentale de la théorie des jeux est que chaque joueur cherche à maximiser son niveau d'utilité indépendamment des autres et connaissant les données des jeux à savoir (N, S_i, u_i, $i \in N$). En d'autres termes, parmi l'ensemble des stratégies qui lui sont disponibles, il essaye de choisir celle qui maximise son utilité espérée. La question qui se pose alors est la suivante : Quelle stratégie un joueur rationnel doit t-il choisir pour maximiser son niveau d'utilité ?
La réponse à cette question nous conduit à étudier la relation de dominance entre les stratégies.

2.3.1. Relation de dominance entre stratégies

2.3.1.1. Dominance stricte

Pour un joueur donné, deux stratégies sont comparables sans ambiguïté si l'une donne un paiement strictement meilleur que l'autre quelles que soient les stratégies des autres joueurs, on dit que la première domine strictement l'autre.

En termes mathématiques, dans le jeu (N, S_i, u_i, $i \in N$), on dit que la stratégie $s_i \in S_i$ du joueur i domine strictement sa stratégie $s'_i \in S_i$ si on a :

$$u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i}) \qquad \forall s_{-i} \in S_{-i}$$

Une stratégie d'un joueur s_i est *strictement dominante* si elle domine strictement toutes ses autres stratégies; elle est *strictement dominée* s'il existe une stratégie qui la domine strictement.

Si un joueur rationnel possède une stratégie strictement dominante, celle-ci est unique et toutes les autres stratégies sont strictement dominées. Le joueur peut alors la jouer sans aucune hésitation puisqu'elle lui procure le meilleur paiement indépendamment des stratégies des autres. Il lui est inutile de prévoir le comportement d'autrui puisque son meilleur choix est indépendant. Dans ce cas, le jeu est immédiatement résolu, on dit qu'on a un *équilibre en stratégies strictement dominantes*. Cet équilibre est unique.

2.3.1.2. Dominance faible

Dans un jeu $(N, S_i, u_i, i \in N)$, on dit que la stratégie $s_i \in S_i$ du joueur i est une stratégie *faiblement dominante* ou tout simplement dominante si on a :

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}), \qquad \forall s'_i \in S_i, \ \forall s_{-i} \in S_{-i}$$

autrement dit, une stratégie s_i est (*faiblement*) *dominante* si elle domine (faiblement) toutes ses autres stratégies; elle est (*faiblement*) *dominée* s'il existe une stratégie qui la domine (faiblement).

Dire que la stratégie s'_i du joueur i est dominée par sa stratégie s_i revient à dire que quel que soit le choix stratégique des autres joueurs, employer s_i est au moins aussi profitable pour i qu'employer s'_i, et que parfois la stratégie s_i est strictement meilleure que s'_i. C'est en ce sens précis qu'on postule alors que la stratégie s_i est meilleure que la stratégie s'_i ce qui revient à admettre qu'un joueur rationnel doit employer une stratégie non dominée.

Remarquons que cet axiome est parfaitement compatible avec le comportement décentralisé des joueurs : pour déterminer ses stratégies non dominées un joueur n'a nul besoin d'information sur le comportement réel des autres joueurs: en fait le joueur i a seulement besoin de connaître les possibilités stratégiques offertes aux autres joueurs (c-a-d $S_{-i} = \prod_{j \neq i} S_j$), toute information supplémentaire (par exemple les préférences des autres joueurs) est superflue.

Soit $D_i(u_i)$ l'ensemble des stratégies dominantes du joueur i :

$$s_i \in D_i(u_i) \Leftrightarrow s_i \ \textit{domine} \ s'_i \qquad \forall (s'_i \in S_i)$$

En fait, une stratégie dominante $s_i \in D_i(u_i)$ du joueur i n'est rien d'autre qu'une solution optimale commune de tous les problèmes d'optimisation

$$\max_{s_i \in S_i} u_i(s_i, s_{-i})$$

2.3.2. Equilibre en stratégies dominantes

L'étude du concept de stratégies dominantes et dominées nous conduit à définir un premier concept d'équilibre obtenu par élimination successive des stratégies strictement dominées.

Définition : On appelle équilibre en stratégie dominantes, toute issue $s = (s_1, \ldots, s_n)$ telle que, pour tout i, s_i est une stratégie dominante du joueur i ($s_i \in D_i(u_i)$).

2.4. Equilibre de Nash

Considérons un jeu sous forme stratégique. Supposons qu'avant de jouer les joueurs se rencontrent et tentent d'harmoniser leurs stratégies. Supposons en outre que si un accord est conclu, sa violation par l'un des joueurs n'entraîne aucune pénalité. Dans de telles conditions les joueurs se doivent de rechercher une issue qui respecte une certaine stabilité interne, dans le sens où aucun d'entre eux ne peut en changeant ultérieurement de stratégie, augmenter son niveau d'utilité. Ceci conduit à la définition de l'équilibre de Nash (ou équilibre stratégique).

Une issue s^* est un équilibre de Nash si :

$$u_i(s^*) \geq u_i(s_i, s^*_{-i}) \qquad \forall i \in N, \forall s_i \in S_i$$

En présence de l'équilibre de Nash, un joueur n'aura aucune incitation à dévier de sa stratégie si les autres joueurs ne dévient pas, et il sait qu'il en est de même pour les autres joueurs.

2.4.1. Relations avec les stratégies strictement dominantes

- En équilibre de Nash, une stratégie n'est jamais strictement dominée.

En situation d'équilibre, si s_i^* est strictement dominée par s_i, le joueur i a intérêt à dévier pour utiliser s_i. Par contre, s_i^* n'est pas forcement une stratégie strictement

dominante. En effet, elle est aussi bonne que toute autre stratégie vis-à-vis des comportements des autres joueurs à l'équilibre considéré, mais pas nécessairement vis-à-vis de tous leurs comportements possibles.

- *Si un joueur possède une stratégie strictement dominante, il doit la jouer en un équilibre de Nash.*

En conséquence, si chacun des joueurs a une stratégie strictement dominante, l'équilibre de Nash est unique et coïncide avec l'équilibre en stratégies strictement dominantes.

2.4.2. Relations avec les stratégies dominantes

Si chaque joueur a une stratégie faiblement dominante et l'utilise, on obtient un équilibre de Nash, Ainsi, les issues de stratégies faiblement dominantes (ou équilibre en stratégies faiblement dominantes) sont des équilibres de Nash. Dans un jeu, il peut donc y avoir plusieurs équilibres de Nash.

2.5. Application économique : les mécanismes d'enchères

Les mécanismes d'enchères sont couramment utilisés, par exemple pour l'émission de bons du Trésor, de valeurs boursières, l'attribution de marchés, des concessions d'exploitation, pour la vente d'oeuvres d'art, etc. De tels mécanismes créent un cadre compétitif censé faire émerger le "prix du marché".
Le caractère équitable des mécanismes d'enchères en fait un cadre particulièrement attrayant, cadre qui est d'ailleurs la doctrine officielle de l'administration en matière d'achats publics.

Dans cette section, nous mettons en oeuvre les concepts de base de la théorie des jeux pour analyser le comportement des différents protagonistes dans une situation d'enchère. Nous étudierons les deux cas d'enchères au premier prix et au second prix.

2.5.1. L'enchère au premier prix

Rappelons que dans les enchères au premier prix, le bénéficiaire de l'objet paye sa mise s_i. Il mise évidemment moins que sa propre évaluation θ_i, car sinon son niveau d'utilité $(\theta_i - s_i)$ serait au mieux nul. L'enchère optimale est donc toujours inférieure ou égale à l'évaluation privée.

2.5.2. L'enchère au second prix

Dans l'enchère au second prix (ou enchère de Vickrey), le prix effectivement payé est égal à la seconde meilleure enchère. Dans ce cas, annoncer sa propre évaluation pour l'objet, $s_i^*(\theta_i) = \theta_i$, est une stratégie dominante pour le joueur i. En effet,

$$u_i(s_i^*(\theta_i), s_{-i}) \geq u_i(s_i(\theta_i), s_{-i}) \qquad \forall s_i \in S_i, \forall s_{-i} \in S_{-i}$$

Preuve

Nous savons que le gain d'un joueur i est $\theta_i - p$ s'il gagne l'enchère et paye p où $p = max\{s_k(\theta_k)/k \neq i\}$, 0 sinon.

Pour toute valeur de p, le maximum de gain que peut obtenir le joueur i est $max\{\theta_i - p, 0\}$. Or le gain de i est 0 si $s_i(\theta_i) \leq p$, et $\theta_i - p$ si $s_i(\theta_i) \geq p$. Le joueur i peut donc se garantir en tout temps le maximum des deux en misant $s_i(\theta_i) = \theta_i$.

Miser sa propre évaluation θ_i est donc une stratégie dominante.

La stratégie d'équilibre dans l'enchère au second prix consiste à annoncer sa vraie évaluation de l'objet. Ce résultat n'est guère surprenant car cette stratégie est dominante: Enchérir moins ferait perdre au joueur des opportunités d'obtenir l'objet; enchérir plus lui ferait payer parfois pour l'objet plus que son évaluation.

2.5.3. Conclusion

Dans les enchères au premier prix, le bénéficiaire de l'objet enchérit moins que sa propre évaluation, sinon son niveau d'utilité serait au mieux nul. La stratégie optimale dans ce cas est donc toujours inférieure ou égale à l'évaluation privée, alors que dans l'enchère au second prix, la stratégie optimale est toujours égale à l'évaluation privée.

2.6. Mécanisme de Clark-Groves

Pour un problème d'allocation très général, nous considérons un jeu particulier qui correspond à la mise en oeuvre d'un mécanisme inspiré des travaux de Vickrey sur les enchères appelé mécanisme de Clark-Groves [Cla 71, Gro 73, MWG 95].

2.6.1. Modélisation

Soit un ensemble de n joueurs $N = \{1, ..., n\}$. Les préférences (ou gain économique) d'un joueur $i \in N$ sont données par :

$$u_i(x) = v_i(x, \theta_i) - t_i \qquad (2.1)$$

où

- $x \in X$ correspond à une allocation quelconque d'une ressource, et X est l'ensemble des allocations possibles;
- θ_i est un paramètre de préférence que seul i connaît;
- t_i est un paiement versé par i au vendeur.
- $v_i(x, \theta_i)$ est une fonction qui donne l'évaluation du joueur i de son l'allocation.

Par exemple, dans le cas de l'allocation d'une ressource indivisible,

$$X = \left\{ x = (x_1, ..., x_n) \middle| \; x_i \in \{0, 1\}, \sum_{i=1}^{n} x_i = 1 \right\}$$

et

$$v_i(x, \theta_i) = x_i \theta_i$$

Les préférences du vendeur sont données par :

$$u_0(x) = v_0(x) + \sum_{i=1}^{n} t_i \qquad (2.2)$$

Ainsi, d'après (2.1) et (2.2), la somme des gains économiques totaux est :

$$v_0(x) + \sum_{i=1}^{n} v_i(x, \theta_i) \tag{2.3}$$

Soit $\theta = (\theta_1, \ldots, \theta_n)$, $x^*(\theta)$ est l'allocation qui maximise les gains économiques totaux donnés par (2.3). Autrement dit,

$$v_0(x^*(\theta)) + \sum_{i=1}^{n} v_i(x^*(\theta), \theta_i) \geq v_0(x) + \sum_{i=1}^{n} v_i(x, \theta_i) \qquad \forall x \in X \tag{2.4}$$

Le jeu défini par le mécanisme de Clark-Groves est le suivant :

(*i*)- chaque joueur soumet au vendeur une mise $\hat{\theta}_i$

(*ii*)- le vendeur met en place l'allocation $x^*(\hat{\theta})$

(*iii*)- le joueur i paie :

$$t_i(\hat{\theta}) = \left[v_0(x^*(0, \hat{\theta}_{-i})) + \sum_{j \neq i} v_j(x^*(0, \hat{\theta}_{-i}), \theta_j)\right] - \left[v_0(x^*(\hat{\theta}_i, \hat{\theta}_{-i})) + \sum_{j \neq i} v_j(x^*(\hat{\theta}_i, \hat{\theta}_{-i}), \theta_j)\right] \tag{2.5}$$

où :

- $x^*(\hat{\theta}_i, \hat{\theta}_{-i})$ est l'allocation optimale calculée en présence de i;
- $x^*(0, \hat{\theta}_{-i})$ est l'allocation optimale calculée en l'absence de i.

2.6.2. Stratégie dominante

Dans ce qui suit, nous montrons que dans le jeu défini par le mécanisme de Clark-Groves, miser $\hat{\theta}_i = \theta_i$ est une stratégie dominante. Autrement dit,

$$v_i(x^*(\theta_i, \theta_{-i}), \theta_i) - t_i(\theta_i, \theta_{-i}) \geq v_i(x^*(\hat{\theta}_i, \theta_{-i}), \theta_i) - t_i(\hat{\theta}_i, \theta_{-i}) \qquad \forall \hat{\theta}$$

Preuve

Supposons que les joueurs autres que i annoncent θ_{-i}. Alors, pour un $\hat{\theta}_i$ quelconque, d'après (2.1) et (2.5) les gains de i sont donnés par :

$$v_i(x^*(\hat{\theta}_i, \theta_{-i}), \theta_i) + \left[v_0(x^*(\hat{\theta}_i, \theta_{-i})) + \sum_{j \neq i} v_j(x^*(\hat{\theta}_i, \theta_{-i}), \theta_j)\right] - \left[v_0(x^*(0, \theta_{-i})) + \sum_{j \neq i} v_j(x^*(0, \theta_{-i}), \theta_j)\right] \qquad (2.6)$$

Le dernier terme de (2.6) noté A est indépendant de $\hat{\theta}_i$.

Par définition de $x^*(\theta)$ on a $\forall \hat{\theta}_i$:

$$v_i(x^*(\theta), \theta_i) + \left[v_0(x^*(\theta)) + \sum_{j \neq i} v_j(x^*(\theta), \theta_j)\right] \geq v_i(x^*(\hat{\theta}_i, \theta_{-i}), \theta_i) + \left[v_0(x^*(\hat{\theta}_i, \theta_{-i})) + \sum_{j \neq i} v_j(x^*(\hat{\theta}_i, \theta_{-i}), \theta_j)\right]$$

Ainsi, $\forall \hat{\theta}_i$ nous avons :

$$v_i(x^*(\theta), \theta_i) + \left[v_0(x^*(\theta)) + \sum_{j \neq i} v_j(x^*(\theta), \theta_j)\right] - A \geq v_i(x^*(\hat{\theta}_i, \theta_{-i}), \theta_i) + \left[v_0(x^*(\hat{\theta}_i, \theta_{-i})) + \sum_{j \neq i} v_j(x^*(\hat{\theta}_i, \theta_{-i}), \theta_j)\right] - A$$

et donc :

$$v_i(x^*(\theta), \theta_i) - t_i(\theta) \geq v_i(x^*(\hat{\theta}_i, \theta_{-i}), \theta_i) - t_i(\hat{\theta}_i, \theta_{-i}) \qquad \forall \hat{\theta}_i$$

Donc les gains de i en misant θ_i sont toujours supérieurs ou égaux à ceux obtenus en misant $\hat{\theta}_i$ et donc miser $\hat{\theta}_i = \theta_i$ est une stratégie dominante.

2.6.3. Conclusion

Le mécanisme de Clark-Groves est un jeu particulier qui s'inspire des enchères au second prix pour un problème d'allocation très général. Dans ce mécanisme, miser sa propre évaluation du bien procure le meilleur gain économique, elle est donc une stratégie dominante.

CHAPITRE 3

L'allocation des ressources partageables

T-REX (Telecommunication Resource EXchange) est un jeu distribué interactif développé au laboratoire de recherche en télécommunication de l'université de Columbia par *Nemo Semret* [Sem 96]. Disponible sur le Web, il consiste en un simple marché virtuel qui effectue l'allocation d'une seule ressource partageable (ou divisible) entre plusieurs joueurs. Cette allocation est réalisée par un mécanisme d'enchère où chaque joueur soumet une mise à travers laquelle il spécifie la quantité de la ressource qu'il désire obtenir ainsi que le prix unitaire qu'il est prêt à payer. Le partage de la ressource est effectué selon une règle d'enchère introduite par *Semret et al.* appelée règle du *second prix progressif* ou *PSP* (*Progressive Second Price*) [LS 97, LS 98a, LS 98b]. Cette règle est une application du mécanisme de Clark-Groves dans le contexte des ressources multi-unitaires et où une mise est définie par un couple (prix, quantité).

Bien que la règle *PSP* est très générale et peut s'appliquer à n'importe quelle ressource multi-unitaire, elle a été étudiée dans le cadre des réseaux de télécommunication pour le partage de la bande passante des lignes de transmission entre plusieurs utilisateurs.

Ce chapitre est organisé comme suit : dans un premier temps, nous analysons le mécanisme d'enchère utilisé par *T-REX* pour l'allocation de la bande d'un lien de transmission. Ensuite, nous donnons les propriétés de la règle d'allocation *PSP*, et enfin, nous étudions un algorithme proposé par *Semret et al.* qui étend la règle *PSP* à un ensemble de liens reliés sous forme d'un arbre.

3.1. L'enchère d'une ressource divisible : *T-REX*

3.1.1. Formulation du problème

Soit une voie de transmission de quantité finie de bande passante Q. Cette quantité est partagée d'une manière équitable en plusieurs canaux par multiplexage fréquentiel. Chaque canal fait l'objet d'une unité de bande passante. Soit $\mathcal{N} = \{1, ..., n\}$ l'ensemble des utilisateurs (ou acheteurs) en compétition pour le partage de Q, la mise d'un utilisateur i est définie ainsi:

$$s_i = (q_i, p_i) \in S_i = [0, Q] \times [0, \infty)$$

s_i est sélectionnée à partir d'un espace de mise S_i qui est l'espace des mises possibles. s_i spécifie la quantité q_i de bande passante que i désire obtenir, ainsi que le prix unitaire p_i qu'il est prêt à payer.

$s = (s_1^T, \ldots, s_n^T)^T$ est le profil des mises de tous les utilisateurs. Il appartient à l'ensemble $S = \Pi_{i \in \mathcal{N}} S_i$:

$$s = \begin{bmatrix} s_1 \\ s_2 \\ \ldots \\ s_n \end{bmatrix} = \begin{bmatrix} q_1 & p_1 \\ q_2 & q_2 \\ \ldots & \ldots \\ q_n & p_n \end{bmatrix}$$

La i^{eme} ligne indique la mise de l'utilisateur numéro i.

Soient deux vecteurs qs et ps tel que : $qs = s(1,0)^T = (q_1,\ldots,q_n)^T$; $qs_i = q_i$ et $ps = s(0,1)^T = (p_1,\ldots,p_n)^T$; $ps_i = p_i$.

En suivant la notation standard de la théorie des jeux, soit $s_{-i} \equiv (s_1^T,\ldots,s_{i-1}^T,s_{i+1}^T,\ldots,s_n^T)^T$, le profil des mises des utilisateurs autres que i obtenu à partir de s en supprimant la ligne i. Ainsi, si on souhaite s'intéresser à la mise d'un utilisateur i, on écrit le profil des mises (s_i,s_{-i}).

3.1.2. Règle d'allocation

L'allocation de la ressource est effectuée selon une règle A de la forme :

$$A : \quad S \rightarrow S$$

$$s = (qs,ps) \rightarrow A(s) = (qA(s), pA(s))$$

La i^{eme} ligne de A, $A_i(s) = (qA_i(s),\ pA_i(s))$, est l'allocation de l'utilisateur i. Il obtient une quantité $qA_i(s)$ à un prix unitaire $pA_i(s)$.

La règle d'allocation A est réalisable si $\forall s$:

$$\sum_{i \in \mathcal{N}} qA_i(s) \leq Q \tag{3.1}$$

$$qA_i(s) \leq qs_i \quad \forall\ i \in \mathcal{N} \tag{3.2}$$

$$pA_i(s) \leq ps_i \quad \forall\ i \in \mathcal{N} \tag{3.3}$$

La formulation ci-dessus est une généralisation de ce qui est habituellement sous-entendu par enchère où une seule ressource non divisible est à vendre. En effet, si pour un utilisateur $w \in \mathcal{N}$; $qA_w(s) = Q$ et $qA_i(s) = 0\ \forall i \neq w$, alors on retrouve le mécanisme d'enchère traditionnel (une ressource unique) pour lequel la théorie est bien développée [FT 91, Mil 87, Mye 81].

Hypothèses:
Dans la formulation ci-dessus, *Semret et al.* supposent que :

H1: Chaque utilisateur $i \in \mathcal{N}$ a une évaluation propre $\theta_i \geq 0$ pour une unité de la ressource. Cette évaluation est une information privée (uniquement i connaît θ_i).
H2: Les utilisateurs sont rationnels. Chacun cherche à maximiser une certaine fonction d'utilité u_i.
H3: Le vendeur est vu comme un utilisateur A_0. Il impose un prix unitaire minimum appelé *prix de réserve* en fixant une mise $s_0 = (qs_0,\ ps_0) \equiv (Q, ps_0)$ [LS 97].

3.1.3. L'utilité des utilisateurs

L'utilité d'un utilisateur $i \in \mathcal{N}$ est donnée par sa fonction u_i :

$$u_i : \begin{array}{l} S \rightarrow [0, +\infty) \\ s \rightarrow u_i(s) \end{array}$$

Etant donné que chaque utilisateur i a une évaluation propre $\theta_i \geq 0$ pour une unité de la ressource, la valeur totale de son allocation est $\theta_i qA_i(s)$. Pour un profil de mises s, sous la règle d'allocation A, *Semret et al.* définissent l'utilité d'un joueur i comme la valeur de ce qu'il obtient moins son coût; en termes mathématiques,

$$u_i(s) = \theta_i qA_i(s) - qA_i(s)pA_i(s)$$

3.2. La règle du second prix progressif *PSP*

Etant donnée une ressource partageable Q et un profil de mise tel qu'il a été défini à la section précédente, la règle d'enchère *PSP* introduite dans [LS 97, LS 98a] permet de calculer l'allocation ainsi que le prix unitaire à payer par chaque utilisateur. Elle est définie comme suit :

$$qA_i(s) = min\left\{ qs_i,\ Q - \sum_{\{j|(ps_j > ps_i)\}} qA_j(s) \right\} \tag{3.4}$$

$$pA_i(s) = \begin{cases} \dfrac{\sum_{j \neq i} ps_j [qA_j(0;s_{-i}) - qA_j(s_i;s_{-i})]}{\sum_{j \neq i} [qA_j(0;s_{-i}) - qA_j(s_i;s_{-i})]}, & si\ qA_i(s) > 0 \\ 0, & si\ qA_i(s) = 0 \end{cases} \tag{3.5}$$

où :

- $qA_i(s)$ est la quantité allouée à l'utilisateur i, c'est le minimum entre ce qu'il a demandé q_i et la quantité restante de Q après avoir servi tous les autres utilisateurs qui ont proposé un prix plus élevé que le sien. Si deux utilisateurs misent le même prix, alors celui qui a enregistré sa mise le premier est privilégié.
- $pA_i(s)$ est le prix unitaire que va payer l'utilisateur i.
- $qA_j(0;s_{-i})$ est l'allocation de j $qA_j(s)$ en absence de l'utilisateur i.
- $qA_j(s_i;s_{-i})$ est l'allocation de j $qA_j(s)$ en présence de l'utilisateur i.

Etant donné que le vendeur (l'utilisateur A_0) a fixé sa mise à (Q,ps_0) avec $ps_0 > 0$ on a:

$$\sum_{i=0}^{n} qA_i(s) = Q$$

ainsi le dénominateur de $pA_i(s)$ est toujours positif lorsque $qA_i(s) > 0$.

3.2.1. Principe de *PSP*

Le principe de *PSP* est un principe d'*exclusion-compensation* (voir figure 3.1) : l'utilisateur i paie un prix unitaire qui est la moyenne de tous les prix $p_{j \neq i}$ misés par les autres utilisateurs dont chacun est pondéré par la quantité supplémentaire qu'aurait obtenue j en absence de i.

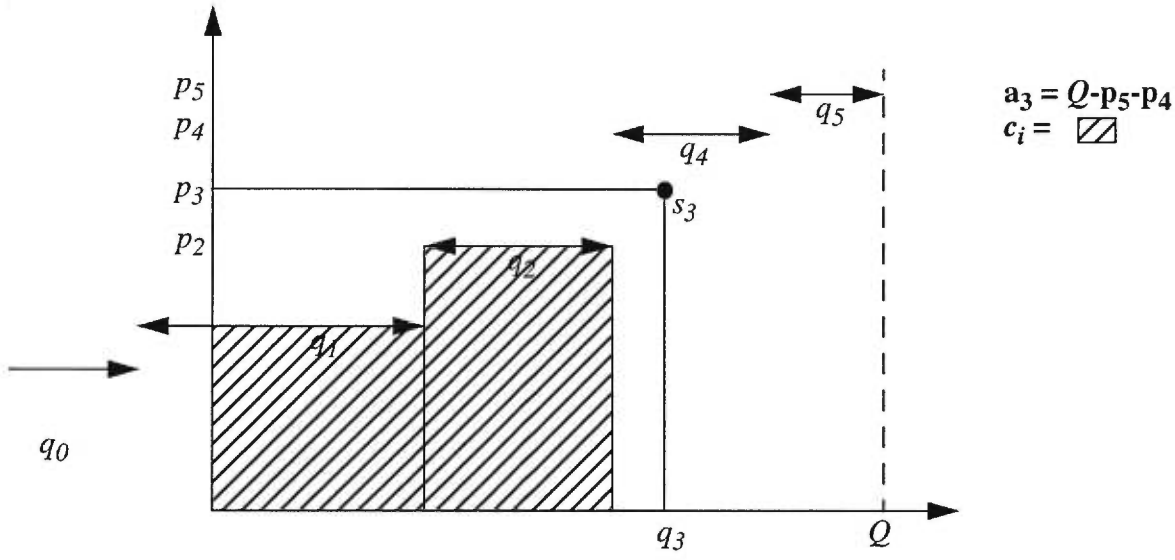


Figure 3.1. Principe d'exclusion-compensation de *PSP*

Le principe d'exclusion-compensation utilisé par la règle *PSP* n'est autre que le principe du mécanisme de Clark-Groves. En effet, la règle *PSP* n'est pas autre chose qu'une mise en oeuvre du mécanisme Clark-Groves dans le contexte de ressources multiples.

3.2.2. Mise en oeuvre du mécanisme de Clark-Groves

Nous rappelons que le mécanisme Clark-Groves est défini comme suit :

(*i*)- chaque joueur soumet au vendeur une mise $\hat{\theta}_i$

(*ii*)- le vendeur met en place l'allocation $x^*(\hat{\theta})$

(*iii*)- le joueur *i* paie :

$$t_i(\hat{\theta}) = \left[v_0(x^*(0, \hat{\theta}_{-i})) + \sum_{j \neq i} v_j(x^*(0, \hat{\theta}_{-i}), \theta_j)\right] - \left[v_0(x^*(\hat{\theta}_i, \hat{\theta}_{-i})) + \sum_{j \neq i} v_j(x^*(\hat{\theta}_i, \hat{\theta}_{-i}), \theta_j)\right]$$

L'utilité d'un joueur i est donné par :

$$v_i(x, \theta_i) - t_i$$

La règle *PSP* n'est pas autre chose qu'une mise en oeuvre du mécanisme Clark-Groves, en effet,

- $\hat{\theta}_i$ correspond à $s_i = (q_i, p_i)$
- $x^*(\hat{\theta})$ correspond à $qA(s)$
- $v_i(x^*(\hat{\theta}_i, \hat{\theta}_{-i}), \theta_i)$ correspond à $qA_i(s_i;s_{-i})ps_i$.
  Nous allons voir plus loin dans ce chapitre qu'avec la règle *PSP*, miser sa propre évaluation $ps_i = \theta_i$ est une stratégie dominante. Ainsi, $qA_i(s_i;s_{-i})ps_i$ n'est autre que $qA_i(s_i;s_{-i})\theta_i$.
- $t_i(\hat{\theta})$ correspond à $pA_i(s)qA_i(s)$.
- $v_i(x, \theta_i) - t_i$ correspond à $\theta_i qA_i(s) - qA_i(s)pA_i(s)$

3.2.3. Complexité de *PSP*

La complexité de *PSP* est faible. Une implémentation efficace devrait dans le pire des cas trier les mises en un temps $O(n\log n)$, effectuer (3.4) en un temps linéaire et (3.5) en un temps $O(n^2)$. Ainsi, la complexité du calcul des allocations est dans $O(n^2)$ [LS97].

3.2.4. Exemple

Soit une ressource R de quantité Q = 10 unités à partager entre 5 utilisateurs $\{A_1,A_2,A_3,A_4,A_0\}$ selon la règle d'enchère *PSP*. L'ensemble des mises est:
s_1=(4,18); s_2=(3,16); s_3=(4,14); s_4=(1,12); s_0=(10,10);

$qA_1(s)$= min{4, (10 - 0)} = 4
$qA_2(s)$= min{3, (10 - 4)} = 3
$qA_3(s)$= min{4, (10 - [4+3])} = 3
$qA_4(s)$= min{1, (10 - [4+3+3])} = 0
$qA_0(s)$= min{10, (10 - [4+3+3])} = 0

$$pA_1(s) = \frac{16(3-3)+14(4-3)+12(1-0)+10(2-0)}{(3-3)+(4-3)+(1-0)+(2-0)} = 11.5 \text{ UM (unité monétaire)}$$

$$pA_2(s) = \frac{18(4-4)+14(4-3)+12(1-0)+10(1-0)}{(4-4)+(4-3)+(1-0)+(1-0)} = 12 \text{ UM}$$

$$pA_3(s) = \frac{18(4-4)+16(3-3)+12(1-0)+10(2-0)}{(4-4)+(3-3)+(1-0)+(2-0)} = 10.66 \text{ UM}$$

$pA_4(s) = pA_0(s) = 0$ puisque $qA_4(s) = qA_0(s) = 0$.

3.2.5. Propriétés de *PSP*

3.2.5.1. Réalisabilité

- La quantité allouée à l'utilisateur i $qA_i(s)$ est le minimum entre ce qu'il a demandé q_i et la quantité restante de Q après avoir servi tous les autres utilisateurs qui ont proposé un prix plus élevé que le sien; *PSP* vérifie (3.1) et (3.2).

- Selon la règle *PSP*, lorsqu'un utilisateur i obtient une unité de ressource, il paye le prix p_j misé par l'utilisateur j qui a été privé de cette unité; p_j est nécessairement inférieur ou égal à p_i sinon c'est j qui aurait obtenu cette unité. Ainsi, *PSP* vérifie la contrainte (3.3).

Par suite, la règle *PSP* vérifie les contraintes de réalisabilité (3.1), (3.2) et (3.3) donc elle est réalisable.

3.2.5.2. Stratégie dominante

La propriété clé de *PSP* est que chaque joueur ne peut faire mieux que de dire simplement la vérité c'est à dire d'affecter $ps_i = \theta_i$. *Miser son évaluation est une stratégie dominante* [LS 97]

Lemme [LS 97] : pour chaque utilisateur $i \in \mathcal{N}$, $\forall (s_i; s_{-i}) \in S$,

$$u_i((qs_i, \theta_i), s_{-i}) \geq u_i(s_i, s_{-i})$$

3.2.5.3. Équilibre de Nash

Etant donné que miser sa propre évaluation θ_i est une stratégie dominante, alors dans le jeu d'enchère donné par la règle *PSP*, il existe un équilibre de Nash $s^* \in T$ avec

$$T = \Pi_{i \in N} T_i \quad \text{et} \quad T_i = \{s_i \in S_i, ps_i = \theta_i\}$$

3.2.5.4. Mesure d'efficacité

La mesure d'efficacité d'une règle d'allocation est une fonction de la forme

$$m(A(s)) = \sum_{i \in N} m_i(A_i(s))$$

Une règle d'allocation est m-efficace en s si $m(A(s)) \geq m(A'(s))$ pour toute règle A' réalisable.

L'efficacité d'une règle d'allocation peut être mesurée par la somme des utilités des joueurs $\sum u_i(A_i(s))$. Dans le cas des enchères, la mesure d'efficacité la plus courante est le revenu du vendeur; par exemple, dans le cas d'une allocation d'une ressource partageable, si une portion de la ressource reste non allouée et il existe un joueur j dont la demande est insatisfaite, alors la règle d'enchère est inefficace car on peut avoir un revenu plus élevé en donnant le reste de la ressource à j. La mesure d'efficacité de la règle *PSP* est mesurée par la volonté à payer des utilisateurs $\sum ps_i qA_i(s)$.

3.2.5.5. Équité

Une autre propriété de la règle *PSP* est qu'elle a un caractère équitable dans le sens où la ressource est partagée parmi les participants qui la souhaitent le plus, c'est-à-dire ceux qui obtiennent la ressource sont ceux qui ont les plus fortes évaluations.

3.3. L'enchère d'un ensemble de ressources partageables

Dans cette section, nous présentons un mécanisme d'enchère proposé par *Semret et al.* qui étend l'enchère d'une seule ressource multi-unitaire à un ensemble de ressources (divisibles) complémentaires. Les ressources considérées sont les liens de transmission d'un réseau de télécommunication.

3.3.1. Formulation du problème [LS 97]

Soit un réseau de télécommunication à L liens de transmissions. L'ensemble des liens est $\mathcal{L} = \{1,\ldots,L\}$ pour lesquelles les capacités de bande passante sont $Q = \{Q^1,\ldots,Q^L\}$. $\mathcal{N}$ est l'ensemble d'utilisateurs $\mathcal{N} = \{1,\ldots,n\}$.

Les mises des utilisateurs sont de la forme $s_i = (q_i,\ p_i\ ,r_i)$, s_i est une mise sur une route r_i de bande passante q_i et à un prix p_i.

Une route $r_i \in \{0,1\}^L$ est un ensemble de liens qui ne forment pas nécessairement un chemin continu. $r_{i,l} = 1$ si le lien $l \in \mathcal{L}$ est sur le chemin de i.

Soit $rs = (r_1^T, \ldots, r_n^T)^T$; la $i^{ème}$ ligne de rs est la route r_i de l'utilisateur i.

3.3.2. Règle d'allocation

Etant donnée la formulation ci-dessus, une règle d'allocation F est de la forme :

$$F : \quad [0,Q]^n \times [0,\infty)^n \times \{0,1\}^{L\times n} \rightarrow [0,Q]^n \times [0,\infty)^n \times \{0,1\}^{L\times n}$$
$$s = (qs, ps, rs) \quad \rightarrow F(s) = (qF(s), pF(s), rF(s))$$

où :

- $qF(s) = (qF_1(s), \ldots, qF_n(s))$; $qF_i(s)$ est la quantité de bande passante allouée à l'utilisateur i pour sa route r_i.
- $pF(s) = (pF_1(s), \ldots, pF_n(s))$; $pF_i(s)$ est le prix unitaire à payer pour i.
- $rF(s) = (rF_1(s), \ldots, rF_n(s))$; $rF_i(s)$ est l'ensemble des liens alloués à i.

Hypothèse:

Dans [LS 97], *Semret et al.* supposent que chaque utilisateur obtient soit l'ensemble des liens demandés, même avec une quantité de bande passante inférieure à ce qu'il a demandé, soit rien du tout ; autrement dit, $qF_i(s) \leq q_i$ et $rF(s) = rs$.

Avec cette hypothèse, une règle d'allocation F est réalisable si $\forall s$,

$$qF(s)rs \leq Q \tag{3.5}$$

$$(qF(s),pF(s)) \leq (qs,ps) \tag{3.6}$$

$$rF(s) = rs \tag{3.7}$$

Lorsque les mises se rapportent à des routes quelconques (voir figure 3.2), une règle d'allocation réalisable et efficace est difficile à calculer.

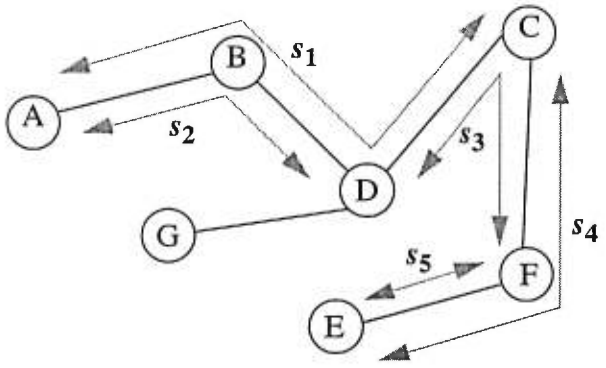


Figure 3.2. Exemple illustratif de mises sur des routes quelconques

Une règle d'allocation efficace dans ce cas, devrait être comme suit: $\forall s$ calculer $F(s)$ est équivalent à résoudre

$$\max_{F(s)} m(F(s))$$

où m est une fonction de mesure d'efficacité.

Sous les contraintes (3.5)-(3.7). Ceci signifie résoudre un problème d'optimisation de grande taille. Avec une restriction additionnelle $qF_i(s) = q_i$ ou 0, ce problème optimisation est en fait NP-complet [HRP 95].

3.3.3. Partage des ressources sur un arbre

En donnant au réseau une structure particulière, celle d'un arbre, et en supposant que les utilisateurs misent sur des routes formant des chemins continus partant nécessairement de la racine de l'arbre vers des noeuds quelconques, *Semret et al.* [LS 97] proposent un algorithme qui calcule d'une manière efficace $F(s)$.

3.3.3.1. Principe

L'algorithme commence à partir des feuilles et remonte jusqu'à la racine, et à chaque noeud de l'arbre, il

- extrait le sous ensemble de mises qui demandent le lien directement rattaché au noeud considéré et les classer par ordre de prix du plus élevé au plus faible,
- effectue une allocation en faisant appel à la règle d'allocation *PSP*,
- effectue une mise à jour des mises selon l'allocation sur le lien courant, et
- supprime le noeud et passe au suivant jusqu'à arriver à la racine.

3.3.3.2. Calcul de $F(s)$

Soit la notation $PSP(Q^l, .)$ pour distinguer le lien l de quantité Q^l sur lequelle PSP s'applique.

Pour $\mathcal{J} \subset \mathcal{N}$, soit $1_{\mathcal{J}}$ l'opérateur qui supprime toutes les lignes i tel que $i \notin \mathcal{J}$, ainsi, $1_{\mathcal{J}}(s)$ est le profil des mises du sous-ensemble d'utilisateurs $\mathcal{J}$.

La règle d'allocation $F(s)$ pour un arbre est donnée par l'algorithme suivant :

Algorithme

* données:
L'ensemble des noeuds V de l'arbre, l'ensemble des liens L, $Q = \{Q^1,\ldots,Q^L\}$ et $s = \{(q_i,p_i,r_i)\}_{i=1}^{n}$

1. - choisir une feuille quelconque $v \in V$; soit $l_v \in \mathcal{L}$ l'unique lien relié à v.
- soit $\mathcal{N}(l_v) = \{i \in \mathcal{N} \ ; \ r_{i,l_v} = 1\}$ le sous ensemble d'utilisateurs qui veulent le lien l_v.

2. - calculer $\bar{s} = PSP(Q^{lv}, 1_{\mathcal{N}(l_v)}(s))$, effectuer l'allocation sur le lien l_v .

3. - Pour chaque $i \in \mathcal{N}(l_v)$, $qs_i = qF_i(s)$, effectuer une mise à jour des s_i.

4. - $V = V - \{v\}$, et $\mathcal{L} = \mathcal{L} - \{l_v\}$, supprimer cette feuille et ce lien de l'arbre.

5. - si $\mathcal{L} \neq \varnothing$ aller à 1, sinon $F(s) = s$ et fin.

Etant donné que l'algorithme s'exécute tout en remontant la route d'un utilisateur, à l'étape 3, sa mise pour le prochain lien est automatiquement réduite à son allocation actuelle sur le lien courant.

3.3.3.3. Calcul du mécanisme Clark-Groves

L'algorithme présenté ci-dessus, est une mise en oeuvre du mécanisme de Clark-Groves dans le contexte d'un ensemble de ressources complémentaires. Les ressources considérées sont des liens de transmission interconnectés sous forme d'un arbre.

Par analogie avec le mécanisme de Clark-Groves, nous avons :

- $\hat{\theta}$ correspond à $s = \{(q_i, p_i, r_i)\}_{i=1}^{n}$
- $x^*(\theta)$ correspond à $qF(s)$
- $t_i(\hat{\theta})$ correspond à $pF_i(s)qF_i(s)$.
- $v_i(x, \theta_i) - t_i$ correspond à $\theta_i qF_i(s) - qF_i(s)pF_i(s)$

3.3.3.4. Exemple

Considérons la spécification du réseau ayant la structure d'un arbre de la figure 3.2, et soit l'ensemble des mises suivant :

s : s_1 = (3, [34], [AD])
s_2 = (2, [32], [AD])
s_3 = (2, [40], [AE])
s_4 = (2, [36], [AE])
s_5 = (1, [38], [AC])
s_6 = (1, [42], [AB])

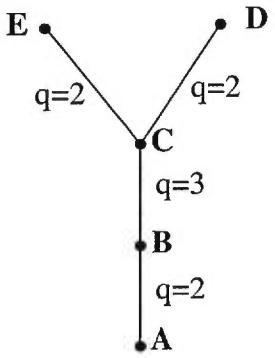


Figure 3.3. Spécification d'un réseau ayant la structure arbre

V = {A, B, C, D, E}; L ={ [AB], [BC], [CD], [CE] };

Calcul de $F(s)$

***1ère itération*:**

v = D; l = [CD] ; $Q^{[CD]} = 2$; $1_{\mathcal{N}(l_v)}(s) = \{s_1, s_2\}$.

s_1 = (3, [34], [AD])

s_2 = (2, [32], [AD])

$qA_1(s)$= 2, $pA_1(s)$= 32;

$qA_2(s)$= 0, $pA_2(s)$= 0;

V = {A, B, C, E}; L ={ [AB], [BC], [CE] };

Selon les allocations sur le lien [CD], s_1 et s_2 deviennent :

s_1= (2, [34], [AD]);

s_2= (0, [32], [AD]);

***2ème itération*:**

v = E; l = [CE] ; $Q^{[CE]} = 2$; $1_{\mathcal{N}(l_v)}(s) = \{s_3, s_4\}$.

s_3= (2, [40], [AE])

s_4= (2, [36], [AE])

$qA_3(s)$= 2, $pA_3(s)$= 36; s_3= (2, [40], [AE]);

$qA_4(s)$= 0, $pA_4(s)$= 0; s_4= (0, [36], [AE]);

V = {A, B, C}; L ={ [AB], [BC]};

***3ème itération*:**

v = C; l = [BC] ; $Q^{[BC]} = 3$; $1_{\mathcal{N}(l_v)}(s) = \{s_1, s_3, s_5\}$.

s_1 = (2, [34], [AD])

s_3 = (2, [40], [AE])

s_5 = (1, [38], [AC])

$qA_1(s)$= 0, $pA_1(s)$= 0; s_1= (0, [34], [AD]);

$qA_3(s)$= 2, $pA_3(s)$= 34; s_3= (2, [40], [AE]);

$qA_5(s)$= 1, $pA_5(s)$= 34; s_5 = (1, [38], [AC])

V = {A, B}; L ={ [AB]};

***4ème itération*:**

v = B; l = [AB] ; $Q^{[AB]} = 2$; $1_{\mathcal{N}(l_v)}(s) = \{s_3, s_5, s_6\}$.

s_3 = (2, [40], [AE])

s_5 = (1, [38], [AC])

s_6 = (1, [42], [AB])

$qA_3(s)$= 1, $pA_3(s)$= 38; s_3= (1, [40], [AE]);
$qA_5(s)$= 0, $pA_5(s)$= 0; s_5 = (0, [38], [AC])
$qA_6(s)$= 1, $pA_6(s)$= 40; s_6= (1, [39], [AB]);
$V = \{A\}$; $L = \varnothing$;

allocation finale :
$qF_3(s) = 1$; $pF_3(s) = max\{34, 36, 38\} = 38$ UM.
$qF_6(s) = 1$; $pF_6(s) = 40$ UM.
$qF_i(s) = pF_i(s) = 0$ pour $i = 1, 2, 4, 5$.

3.4. Conclusion

T-REX implémente une règle d'enchère appelée *PSP* (*Progressive Second Price*) introduite par *Semret et al.* dans [LS 97] pour l'allocation des ressources divisibles, et en particulier, pour le partage de la bande passante des lignes de transmission entre plusieurs utilisateurs. Cette règle est une application du mécanisme de Clark-Groves dans le contexte d'une seule ressource multi-unitaire, elle a plusieurs propriétés désirables dont la plus importante est qu'elle possède un équilibre de Nash lorsque chaque utilisateur mise sa propre évaluation [LS 97].
En se basant sur la règle *PSP*, *Semret et al.* proposent un algorithme qui calcule le mécanisme de Clark-Groves dans un contexte de réseau de liens de transmission ayant la structure d'un arbre. Les utilisateurs misent ainsi sur des combinaisons de liens en ne spécifiant qu'un seul prix, le prix de chaque lien n'étant pas spécifié. La limite de cette algorithme est que les utilisateurs sont contraints de miser sur des chemins continus partant nécessairement de la racine de l'arbre.

CHAPITRE 4

Généralisation de la règle *PSP*

Dans le mécanisme d'enchère proposé dans *T-REX* pour le partage des ressources divisibles, *Semret et al.* supposent que les utilisateurs ont la même évaluation θ_i pour chaque unité acquise de la ressource. Avec cette hypothèse, il y a une perte de flexibilité du fait que les utilisateurs peuvent avoir une évaluation qui décroît pour chaque unité additionnelle à partir d'une certaine quantité minimale.
Dans le but de rendre ce mécanisme plus flexible, nous proposons dans ce chapitre, une généralisation de la règle d'enchère *PSP* vers le cas où les utilisateurs ont une évaluation décroissante pour chaque unité additionnelle de la ressource :
$\theta_i^1 \geq \theta_i^2 \geq \ldots \geq \theta_i^k$, où θ_i^k est l'évaluation de l'utilisateur i pour la k^{eme} unité acquise.
Nous introduisons ainsi, une nouvelle règle d'enchère appelée *DPSP* (*Dynamic Progressive Second Price*).

Après avoir définit formellement le problème, nous présentons dans la deuxième section de ce chapitre la nouvelle règle d'enchère *DPSP*. En se basant sur cette nouvelle règle, nous enchaînons dans la troisième section, en présentant une version améliorée de l'algorithme proposé par *Semret et al.* qui calcule le mécanisme de Clark-Groves pour le partage de la bande passante dans un arbre.

4.1. L'enchère d'une ressource divisible

Dans cette section, nous allons présenter un nouveau mécanisme d'enchère pour les ressources partageables où nous supposons que les utilisateurs ont une évaluation décroissante pour chaque unité additionnelle de la ressource: $\theta_i^1 \geq \theta_i^2 \geq \ldots \geq \theta_i^k$, avec θ_i^k est l'évaluation de l'utilisateur i pour la k^{eme} unité acquise.

4.1.1. Formulation du problème

Soit Q la quantité totale de bande passante d'un canal de transmission multiplexé en K canaux plus étroits dont chacun représente une unité de bande passante, et $\mathcal{N} = \{1, \ldots, n\}$ un ensemble d'utilisateurs qui sont en compétition pour le partage de Q.

On définit la mise d'un utilisateur $i \in \mathcal{N}$ comme suit :

$$s_i = (q_i, p_i^1, \ldots, p_i^{q_i})$$

où :

q_i est le nombre d'unités de bande passante demandées par le i^{eme} utilisateur,

p_i^k est appelée *mise élémentaire*, c'est le prix que i est prêt à payer pour obtenir la k^{eme} unité demandée avec $1 \leq k \leq q_i$ et $p_i^1 \geq p_i^2 \geq \ldots \geq p_i^{q_i}$.

Soient q et p deux opérateurs tels que :

- $qs = (qs_1, \ldots\ldots, qs_n)^T$ avec $qs_i = q_i$
- $ps = (ps_1, \ldots\ldots, ps_n)^T$ avec $ps_i = p_i^1, \ldots, p_i^{q_i}$

4.1.2. Règle d'allocation

La règle d'allocation est de la forme :

$$A : \quad S \rightarrow S$$
$$s = (qs, ps) \rightarrow A(s) = (a(s), c(s))$$

où $S = \Pi_{i \in \mathcal{N}} S_i$

La $i^{\text{ème}}$ ligne de A, $A_i(s) = (a_i(s), c_i(s))$ représente l'allocation de l'utilisateur i, ainsi que la charge totale à payer. Il obtient une quantité $a_i(s)$ à une charge $c_i(s)$.

La règle d'allocation A est réalisable si $\forall s$:

$$\sum_{i \in \mathcal{N}} a_i(s) \leq Q \tag{4.1}$$

$$a_i(s) \leq q_i \quad \forall \ i \in \mathcal{N} \tag{4.2}$$

$$c_i(s) \leq \sum_{k=1}^{a_i(s)} p_i^k \quad \forall \ i \in \mathcal{N} \tag{4.3}$$

Hypothèse :
D'une manière similaire à *T-REX*, nous supposons que le vendeur (A_0) impose un prix unitaire minimum appelé *prix de réserve* en soumettant une mise $s_0 = (Q, p_0^1, \ldots, p_0^K)$ avec $p_0^1 = \ldots = p_0^K > 0$.

Ainsi, étant donné que A_0 mise sur toute la quantité disponible, (4.1) devient :

$$\sum_{i=0}^{n} a_i(s) = Q$$

4.1.3. L'utilité des utilisateurs

On définit l'utilité d'un utilisateur $i \in \mathcal{N}$, comme étant la valeur de ce qu'il obtient moins le coût; étant donné que chaque utilisateur i attribue une évaluation propre $\theta_i^1 \geq \theta_i^2 \geq \ldots \geq \theta_i^k$ pour chaque unité demandée, l'utilité de i est :

$$u_i(s) = \begin{cases} \sum_{k=1}^{a_i(s)} \theta_i^k - c_i(s) & si \ \ a_i(s) > 0 \\ 0 & si \ \ a_i(s) = 0 \end{cases}$$

4.2. La règle *DPSP* (*Dynamic Progressif Second Price*)

Dans cette section, nous proposons une nouvelle règle d'enchère pour le ressources divisibles appelée *DPSP* qui est une généralisation de la règle *PSP* vers le cas où les utilisateurs spécifient un prix pour chaque unité demandée. La règle d'allocation *DPSP* est définie comme suit :

$$a_i(s) = min\left\{q_i,\ Q - \sum_{\{j|(p_j^1 > p_i^1)\}} a_j(s)\right\} \tag{4.4}$$

$$c_i(s) = \sum_{j \neq i} \sum_{k=1}^{x_j} p_j^{k + a_j(s_i;s_{-i})} \qquad \text{si } x_j > 0 \tag{4.5}$$

avec

$$x_j = a_j(0;s_{-i}) - a_j(s_i;s_{-i})$$

où :

- $a_i(s)$ est la quantité allouée à l'utilisateur i.
- $c_i(s)$ est la charge que va payer i pour la quantité $a_i(s)$ obtenue.
- x_j est la quantité supplémentaire qu'aurait obtenu j en absence de i.
- $a_j(0;s_{-i})$ est l'allocation de j $a_j(s)$ en absence de l'utilisateur i.
- $a_j(s_i;s_{-i})$ est l'allocation de j $a_j(s)$ en présence de l'utilisateur i.

La charge d'un utilisateur i peut être exprimée autrement : considérant l'ensemble des mises élémentaires rejetées triées par ordre décroissant $X = \left\{\bigcup_{j \neq i} p_j^k\right\}$, le prix que va payer $i \in \mathcal{N}$ pour la $k^{ème}$ unité acquise n'est autre que le $k^{ème}$ élément de X. Ainsi, $c_i(s)$ est la somme des $a_i(s)$ premiers éléments de X.

$$c_i(s) = \sum_{k=1}^{a_i(s)} X_k \tag{4.6}$$

4.2.1. Principe de *DPSP*

Le principe de *DPSP* est le même que celui de *PSP*, c'est un principe d'*exclusion-compensation* : pour chaque unité de ressource acquise, le bénéficiaire paye le prix qu'était prêt à payer l'utilisateur j privé de cette unité. Ainsi, la règle d'enchère *DPSP* est une mise en oeuvre du mécanisme Clark-Groves dans le contexte de ressources multi-unitaires, et où les agents spécifient un prix pour chaque unité demandée.

4.2.2. Réalisabilité de *DPSP*

- Le nombre d'unités allouée à l'utilisateur i est le minimum entre ce qu'il a demandé q_i et la quantité restante de Q après avoir servi toutes les autres mises élémentaires des autres utilisateurs qui sont plus élevées que les siennes. Ainsi, *DPSP* vérifie (4.1) et (4.2).

- Selon la règle *DPSP*, lorsqu'un utilisateur i obtient une unité de ressource, il paye le prix p_j misé par l'utilisateur j qui a été privé de cette unité, p_j est nécessairement inférieur ou égal à p_i sinon c'est j qui aurait obtenu cette unité. Ainsi, *DPSP* vérifie la contrainte (4.3).

Par suite, la règle *DPSP* vérifie les contraintes (4.1), (4.2) et (4.3) de la page 49, donc elle est réalisable.

4.2.3. Complexité de *DPSP*

La complexité de *DPSP* est inférieure à celle de *PSP*, en effet, une bonne implémentation devrait dans le pire des cas trier les mises en un temps $n\log n$, effectuer (4.4) en un temps linéaire et (4.6) en un temps $n\log n$. Ainsi, la complexité de *DPSP* est de l'ordre $O(n)$.

4.2.4. Exemple

Considérant la nouvelle règle *DPSP*, soit un lien de transmission divisé en 5 canaux et un ensemble de 5 utilisateurs $\{A_1,A_2,A_3,A_4,A_0\}$ dont les mises respectives sont :

s_1 = (4, [**18**,**17**,12,10])
s_2 = (3, [**16**,**15**,12])
s_3 = (2, [14,13])
s_4 = (1, [**20**])
s_0 = (5, [10,10,10,10,10])

Allocation

$a_1(s)$= 2; $a_2(s)$= 2; $a_3(s)$= 0; $a_4(s)$= 1; $a_0(s)$= 0;
$c_1(s)$= ?

Pour calculer $c_1(s)$, on doit d'abord calculer l'allocation en absence de l'utilisateur 1, c'est à dire on suppose que $a_1(s)$= 0.

s_2 = (3, [**16**,**15**,12])
s_3 = (2, [**14**,**13**])
s_4 = (1, [**20**])
s_0 = (5, [10,10,10,10,10])
$a_1(s)$=0; $a_2(s)$= 2; $a_3(s)$= 2; $a_4(s)$= 1; $a_0(s)$= 0;

l'utilisateur 3 aurait obtenu 2 unités si 1 n'était pas présent, ainsi, 1 paie les prix qu'était prêt à payer 3 pour ces deux unités, $c_1(s) = 14 + 13 = 27$ UM.

$c_2(s)$= ? $a_2(s)$= 0

s_1 = (4, [**18**,**17**,12,10])

s_3 = (2, [**14**,**13**])

s_4 = (1, [**20**])

s_0 = (5, [10,10,10,10,10])

$c_2(s)$ = 14 + 13 = 27 UM

$c_4(s)$= ? $a_4(s)$= 0

s_1 = (4, [**18**,**17**,12,10])

s_2 = (3, [**16**,**15**,12])

s_3 = (2, [**14**,13])

s_0 = (5, [10,10,10,10,10])

$c_4(s)$ = 14 UM

$c_3(s) = c_0(s)$= 0.

Allocation finale :

$a_1(s)$= 2; $c_1(s)$ = 27 UM

$a_2(s)$= 2; $c_2(s)$ = 27 UM

$a_3(s)$= 0; $c_3(s)$ = 0 UM

$a_4(s)$= 1; $c_4(s)$ = 14 UM

$a_0(s)$= 0; $c_0(s)$= 0 UM

4.3. Partage des ressources dans un arbre

4.3.1. Formulation du problème

Considérant un réseau de télécommunication à L liens $\mathcal{L} = \{1. . . L\}$ interconnectés selon la structure d'un arbre et pour les quels, les quantités de bande passante respectives sont $Q = \{Q^1,...,Q^L\}$.

En se basant sur la règle *DPSP* présentée à la section précédente, nous proposons une règle d'allocation F pour l'allocation simultanée de la bande passante au niveau de l'ensemble de l'arbre.

Soit un ensemble d'utilisateurs $\mathcal{N} = \{1, ..., n\}$ qui misent sur des combinaisons de liens d'une certaine quantité de bande passante et formant des chemins continus partant de la racine de l'arbre vers des noeuds quelconques.

Les mises des utilisateurs sont de la forme $s_i = (q_i,\ p_i^1,...,p_i^{q_i}\ ,r_i)$, s_i est une mise sur une route r_i de bande passante q_i, p_i^k le prix misé pour la k^{eme} unité demandé

Une route $r_i \in \{0,1\}^L$ est un ensemble de liens qui forment un chemin continu partant de la racine de l'arbre vers un noeud quelconque. $r_{i,l} = 1$ si le lien $l \in \mathcal{L}$ est dans le chemin de i.

Soit $rs = (r_1^T\ , ...,r_n^T\ )^T$; la i^{eme} ligne de rs est la route r_i de l'utilisateur i.

4.3.2. Règle d'allocation

La règle d'allocation F est de la forme:

$$F: \quad [0,Q]^n \times [0,\infty)^n \times \{0,1\}^{L\times n} \rightarrow [0,Q]^n \times [0,\infty)^n \times \{0,1\}^{L\times n}$$
$$s = (qs, ps, rs) \quad \rightarrow F(s) = (f(s), c(s), rF(s))$$

où :

- $f(s) = (f_1(s), \ldots\ldots, f_n(s))$; $f_i(s)$ est la quantité de bande passante allouée à i pour sa route r_i.

- $c(s) = (c_1(s), \ldots\ldots, c_n(s))$; $c_i(s)$ est la charge à payer de i.

- $rF(s) = (rF_1(s), \ldots\ldots, rF_n(s))$; $rF_i(s)$ est l'ensemble des liens alloués à i.

Hypothèse :

Comme dans [LS 97], on suppose que chaque utilisateur obtient soit l'ensemble des liens demandés, même avec une quantité de bande passante inférieure à ce qu'il a demandé, soit rien du tout, autrement dit, $f_i(s) \leq q_i$ et $rF(s) = rs$.

Ainsi, F est réalisable si $\forall s$,

$$f_i(s) \leq qs_i$$

$$c_i(s) \leq \sum_{k=1}^{f_i(s)} p_i^k$$

$$f(s)rs \leq Q$$

$$rF(s) = rs$$

4.3.3. Principe d'allocation

L'allocation est donnée par un algorithme qui commence à partir des feuilles et qui remonte jusqu'à la racine en effectuant une allocation au niveau de chaque lien selon la règle *DPSP*. Etant donné que les utilisateurs misent sur des chemins continus partant de la racine de l'arbre vers des noeuds quelconques, l'algorithme s'exécute tout en remontant les routes des utilisateurs. Ainsi, en remontant la route d'un utilisateur, au niveau de chaque lien, sa mise est automatiquement mise à jour en fonction de son allocation actuelle sur le lien courant.

4.3.4. Calcul de $F(s)$

Soit la notation $DPSP(Q^l, .)$ pour indiquer que *DPSP* est entrain de s'appliquer sur la ressource l pour laquelle il existe une quantité Q^l .

Pour $\mathcal{J} \subset I$, soit $1_{\mathcal{J}}$ l'opérateur qui supprime toutes les lignes i tel que $i \notin \mathcal{J}$, ainsi, $1_{\mathcal{J}}(s)$ est le profil des mises du sous-ensemble d'utilisateurs $\mathcal{J}$.

Pour chaque utilisateur $i \in \mathcal{N}$, on définit deux vecteurs $\mathrm{P}_{i,l}$ et CO_i de taille q_i.
- $\mathrm{P}_{i,l}$ est un vecteur prix, $\mathrm{P}_{i,l}^k$ $(1 \le k \le a_i)$ est le prix de la $k^{ème}$ unité obtenue calculé selon $DPSP(Q^l, .)$.

- CO_i est un vecteur coût d'option, CO_i^k $(1 \le k \le a_i)$ est le prix courant que va effectivement payer i pour la $k^{ème}$ unité obtenue. En remontant la route d'un utilisateur, selon l'allocation sur le lien courant, CO_i est automatiquement mis à jour.

La règle d'allocation F pour un arbre est donnée par l'algorithme *Tree_Allocation* suivant :

Algorithme *Tree_Allocation*

* données:

L'ensemble de noeuds V de l'arbre, l'ensemble des liens $\mathcal{L}$, $Q = \{Q^1, \ldots\ldots, Q^L\}$ et $s = \{s_i\}_{i=1}^{n}$, $\{CO_i\}_{i=1}^{n}$, $\{\mathrm{P}_{i,l}\}_{i=1}^{n}$.

1. - choisir une feuille $v \in V$; soit $l_v \in \mathcal{L}$ l'unique lien relié à V.

- soit $\mathcal{N}(l_v) = \{i \in \mathcal{N} \ ; \ r_{i,l_v} = 1\}$ le sous ensemble d'utilisateurs demandant l_v.

2. - calculer $(a_i(s), \mathrm{P}_{i,l}) = DPSP(Q^{lv}, 1_{\mathcal{N}(l_v)}(s))$, on effectue l'allocation de l_v .

3. - Pour chaque $s_i \in 1_{\mathcal{N}(l_v)}(s)$

$qs_i = a_i(s)$, on effectue une mise à jour de s_i.

$CO_i = max\{CO_i \cup \mathrm{P}_{i,l}\}$, on remet à zero CO_i et on lui affecte à les $a_i(s)$ plus grand éléments des deux vecteurs CO_i et $\mathrm{P}_{i,l}$.

4. - $V = V - \{v\}$, et $\mathcal{L} = \mathcal{L} - \{l_v\}$, on supprime cette feuille et ce lien de l'arbre.

5. - si $\mathcal{L} \neq \varnothing$ aller à 1, sinon

pour $i \in \mathcal{N}$ $\quad f_i(s) = qs_i$

$$c_i(s) = \sum_{j=1}^{a_i(s)} CO_i^j$$

fin.

4.3.5. Complexité de *Tree_Allocation*

Etant donné un arbre de L liens, pour calculer les allocations, *Tree_Allocation* exécute L fois la règle *DPSP*. Ainsi, la complexité de l'algorithme est de l'ordre $O(L \times n) \cong O(n)$.

4.3.6. Exemple

Considérant la spécification du réseau illustrée par la figure 4.1, et soit l'ensemble des mises s.

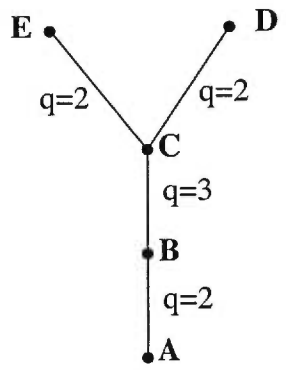


Figure 4.1. Spécification d'un réseau ayant la structure arbre

s: s_1= (2, [34,27], [AE])

s_2= (2, [32,31], [AE])

s_3= (1, [26], [AE])

s_4= (1, [25], [AE])

s_5 = (2, [34,32], [AD])

s_6 = (2, [33,28], [AD])

s_7 = (1, [32], [AD])

s_8 = (1, [27], [AD])

s_9 = (1, [32], [AC])

s_{10} = (1, [27], [AC])

s_{11} = (1, [30], [AB])

s_{12} = (1, [27], [AB])

Calcul de $F(s)$

V = {A, B, C, D, E}; $\mathcal{L}$ ={ [AB], [BC], [CD], [CE] };

***1^{ere} itération*:**

v = E; l = [CE] ; $Q^{[CE]} = 2$; $1_{\mathcal{N}((l_v)}(s) = \{s_1, s_2, s_3, s_4\}$.

s_1= (2, [**34**,27], [AE])

s_2= (2, [**32**,31], [AE])

s_3= (1, [26], [AE])

s_4= (1, [25], [AE])

$a_1(s)$= 1; $P_{1,l}$ = [31, 0]; s_1= (1, [34], [AE]); CO_1 = [31, 0];

$a_2(s)$= 1; $P_{2,l}$ = [27, 0]; s_2= (1, [32], [AE]); CO_2 = [27, 0];

$a_3(s)$= 0; $c_3(s)$= 0; s_3= (0, [26], [AE])

$a_4(s)$= 0; $c_4(s)$= 0; s_4= (0, [25], [AE])

V = {A, B, C, D}; $\mathcal{L}$ ={ [AB], [BC], [CD] }; s = {s_1, s_2, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}}

***2^eme itération*:**

v = D; l = [CD] ; $Q^{[CD]} = 4$; $1_{\mathcal{N}(l_v)}(s) = \{s_5, s_6, s_7, s_8\}$.

s_5= (2, [**34**,**32**], [AD])

s_6= (2, [**33**,28], [AD])

s_7= (1, [**32**], [AD])

s_8= (1, [27], [AD])

$a_5(s)$= 2; $P_{5,l}$ = [28, 27]; s_5= (2, [34,32], [AD]); CO_5 = [28, 27];

$a_6(s)$= 1; $P_{6,l}$ = [27, 0]; s_6= (1, [33], [AD]); CO_6 = [27, 0];

$a_7(s)$= 1; $P_{7,l}$ = [28, 0]; s_7= (1, [32], [AD]); CO_7 = [27];

$a_8(s)$= 0; $c_8(s)$= 0; s_8= (0, [27], [AD]);

V = {A, B, C}; $\mathcal{L}$={ [AB], [BC]}; s = {s_1, s_2, s_5, s_6, s_7, s_9, s_{10}, s_{11}, s_{12}}

***3^eme itération*:**

v = C; l = [BC] ; $Q^{[BC]} = 3$; $1_{\mathcal{N}(l_v)}(s) = \{s_1, s_2, s_5, s_6, s_7, s_9, s_{10}\}$.

s_1= (1, [**34**], [AE])

s_2= (1, [32], [AE])

s_5= (2, [**34**,32], [AD])

s_6= (1, [**33**], [AD])

s_7= (1, [32], [AD])

s_9= (1, [32], [AC])

s_{10}= (1, [27], [AC])

$a_1(s)$= 1; $P_{1,l}$ = [32, 0]; s_1= (1, [34], [AE]); $CO_1 = max\{31, 32\}$ = [32, 0]

$a_5(s)$= 1; $P_{5,l}$ = [32, 0]; s_5= (1, [34], [AD]); $CO_5 = max\{27, 28, 32\}$ = [32, 0]

$a_6(s)$= 1; $P_{6,l}$ = [32]; s_6= (1, [33], [AD]); $CO_6 = max\{27, 32\}$ = [32, 0]

V = {A, B}; $\mathcal{L}$={ [AB]}; s = {s_1, s_5, s_6, s_{11}, s_{12}}

4eme itération:

V = B; l = [AB] ; $Q^{[AB]} = 4$; $1_{\mathcal{N}(l_v)}(s) = \{s_1, s_5, s_9, s_{11}, s_{12}\}$.

s_1= (1, [**34**], [AE])

s_5= (1, [**34**], [AD])

s_6= (1, [**33**], [AD])

s_{11}= (1, [**30**], [AB])

s_{12}= (1, [27], [AB])

$a_1(s)$= 1; $P_{1,l}$ = [27]; s_1= (1, [**34**], [AE]); $CO_1 = max\{32, 27\} = [32, 0]$

$a_5(s)$= 1; $P_{5,l}$ = [27]; s_5= (1, [**34**], [AD]); $CO_5 = max\{32, 27\} = [32, 0]$

$a_6(s)$= 1; $P_{6,l}$ = [27]; s_6= (1, [**33**], [AD]); $CO_6 = max\{32, 27\} = [32, 0]$

$a_{11}(s)$= 1; $P_{11,l}$ = [27]; s_{11}= (1, [**30**], [AB]); $CO_{11} = max\{0, 27\} = [27]$

$V = \{A\}$; $\mathcal{L} = \varnothing$;

allocation finale :

$f_1(s)$ = 1; $c_1(s)$= 32 UM;

$f_5(s)$ = 1; $c_5(s)$= 32 UM;

$f_6(s)$ = 1; $c_6(s)$= 32 UM;

$f_{11}(s)$ = 1; $c_{11}(s)$= 27 UM;

$f_i(s) = c_i(s) = 0$ pour $i \neq 1, 5, 6, 11$

4.4. Conclusion

Dans ce chapitre, nous avons proposé une amélioration du mécanisme d'enchère introduit par *Semret et al.* dans *T-REX* qui le rend plus flexible du fait que les utilisateurs ayant une évaluation décroissante de la ressource peuvent maintenant spécifier un prix pour chaque unité demandée. Nous avons introduit ainsi une nouvelle règle d'enchère appelée *DPSP* (*Dynamic Progressive Second Price*) qui est généralisation de *PSP* (*Progressive Second Price*) au cas où les mises contiennent un vecteur de prix et non plus un seul prix unitaire. En se basant sur cette nouvelle règle, nous avons présenté une version améliorée de algorithme proposé par *Semret et al.* pour le partage et la fixation de prix de la bande passante dans un arbre selon le mécanisme de Clark-Groves.

CHAPITRE 5

Possibilités d'extension de *Tree_Allocation*

Dans les deux chapitres précédents, nous avons présenté deux algorithmes qui calculent le mécanisme de Clark-Groves pour le partage de la bande passante d'un ensemble de liens de transmission interconnectés selon la structure d'un arbre. Dans ce contexte, un utilisateur mise sur une combinaison de liens qui forment un chemin continu d'une certaine quantité de bande passante partant de la racine de l'arbre vers un noeud quelconque.

Le deuxième algorithme *Tree_Allocation* que nous avons présenté au chapitre précédent, est une version améliorée du premier algorithme proposé par *Semret et al.* du fait que les conditions de mise sont devenues plus souples en permettant aux utilisateurs de spécifier un prix pour chaque unité demandée et non pas un seul prix unitaire. L'inconvénient de ces deux algorithmes, est que, les utilisateurs sont contraints à miser sur des chemins continus partant nécessairement de la racine de l'arbre. Ceci entraîne une perte de flexibilité du fait que, un utilisateur ne peut pas miser sur un chemin entre deux noeuds qui soient tous les deux différents de la racine, par exemple, un chemin entre deux feuilles de l'arbre. Par conséquent, il ne pourra pas participer à

l'enchère, ce qui réduit le nombre d'utilisateurs potentiels, affaiblit la concurrence et par suite, le prix de vente.

Dans le but de rendre la politique de mise plus flexible, nous nous proposons dans ce chapitre d'étudier les possibilités d'extension de l'algorithme *Tree_Allocation* vers le cas plus général où les utilisateurs peuvent miser sur des chemins entre deux noeuds quelconques de l'arbre sans aucune restriction. Dans un premier temps, nous commençons par étudier les limites de l'algorithme qui font que les chemins misés doivent nécessairement partir de la racine de l'arbre.

5.1. Limites de *Tree_Allocation*

Considérons la spécification de l'arbre donnée par la figure 5.1,

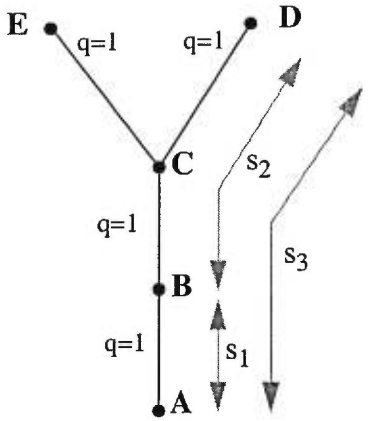


Figure 5.1. Spécification d'un réseau ayant la structure arbre

Supposons que les utilisateurs peuvent miser entre des noeuds quelconques de l'arbre, et soit un ensemble de trois utilisateurs A_1, A_2 et A_3 dont les mises respectives sont :

s : s_1= (1, [15], [AB])
s_2= (1, [25], [BD])
s_3= (1, [30], [AD])

Selon l'algorithme *Tree_Allocation*, en parcourant l'arbre des feuilles jusqu'à la racine, au niveau de chaque lien, c'est toujours la mise s_3 qui est la plus élevée, ainsi l'allocation finale est :

$a_1(s) = 0$; $c_1(s) = 0$ UM
$a_2(s) = 0$; $c_2(s) = 0$ UM
$a_3(s) = 1$; $c_3(s) = 25$ UM

Cette allocation n'est pas efficace du fait que, A_3 est prêt à payer 30 UM pour le chemin [AD] alors que A_1 et A_2 ensemble sont prêts à payer 40 UM pour ce même chemin. Ainsi, l'allocation optimale dans ce cas serait d'attribuer les tronçons [BC] et [CD] à A_2 et le tronçon [AB] à A_1.

D'après l'ensemble des mises, un chemin de A vers D est composé par les chemins [AB] et [BD] des mises s_1 et s_2. Si ces deux mises étaient fusionnées ensemble en une seule mise s_4= (1, [40], [AD]), alors c'est cette dernière qui aurait obtenu le chemin [AD] et non s_1. Ainsi, cet exemple montre clairement que l'algorithme *Tree_Allocation* ne traite pas le cas des chemins composés. ce qui explique pourquoi les utilisateurs sont contraints à miser sur des chemins continus partant toujours de la racine de l'arbre.

5.2. Possibilité d'extension de *Tree_Allocation*

Dans le but de résoudre le problème de chemins composés et permettre ainsi aux utilisateurs de miser sur des chemins entre des noeuds quelconque de l'arbre, nous avons essayé d'adopter l'approche suivante :

1- Diviser l'arbre en plusieurs sous-arbres comme le montre la figure 5.2.
2- Permettre aux utilisateurs de miser sur des chemins entre des noeuds quelconques appartenant à des sous-arbres différents (exemple s_2).

3- Pour chaque sous-arbre, considérer le sous-ensemble de mises de ce sous-arbre et effectuer une allocation selon l'algorithme *Tree_Allocation*, les allocations au niveau de l'ensemble des sous-arbres, sont effectuées d'une manière séquentielle, c'est-à-dire que l'allocation du sous-arbre a_i s'effectue en fonction de l'allocation du sous-arbre a_{i-1}. Ainsi, si la route d'un utilisateur porte sur deux sous arbres a_i et a_{i+1}, alors sa mise au niveau de a_{i+1} sera mis à jour en fonction de son allocation au niveau de a_i.

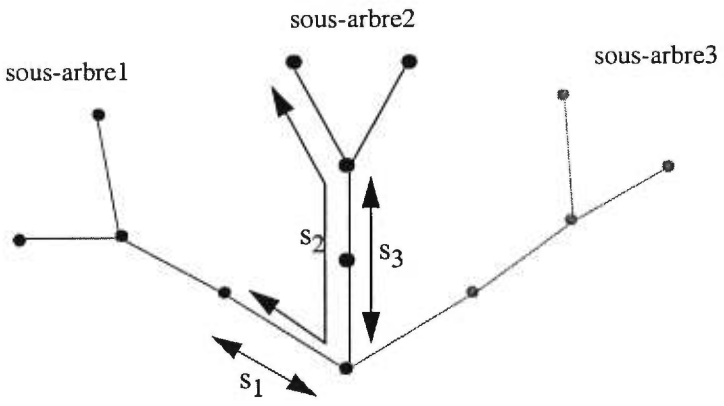


Figure 5.2. Division d'un arbre en plusieurs sous-arbres

5.3. Problèmes rencontrés

P1- Encore une fois, les utilisateurs ne peuvent pas miser sur des chemins entre deux noeuds quelconques d'un même sous-arbre, sinon on retourne au problème de départ et on ne peut plus appliquer *Tree_Allocation* pour le calcul de l'allocation au niveau du sous-arbre.

P2- Etant donné qu'une mise peut porter sur deux sous-arbres différents, malgré plusieurs alternatives pour calculer une allocation optimale au niveau de l'ensemble de l'arbre, nous étions à chaque fois confrontés au problème de chemins composés. En effet, on risque de satisfaire la mise s_2 au lieu de s_1 et s_3 bien que ces deux dernières, ensemble, soient plus élevées que s_2 (voir figure 5.2).

5.4. Direction vers la programmation linéaire

A la suite d'une série de tentatives sans succès pour généraliser l'algorithme *Tree_Allocation* vers le cas où les mises peuvent se faire sur des chemins entre des noeuds quelconques de l'arbre, nous avons cherché à résoudre le même problème autrement en utilisant les techniques de la programmation linéaire.

Dans le chapitre suivant, nous présentons un algorithme qui se base sur la programmation linéaire pour le partage de la bande passante sur un arbre selon le même principe de *Tree_Allocation* et en permettant aux utilisateurs de miser sur des chemins entre des noeuds quelconques de l'arbre sans aucune restriction.

CHAPITRE 6

Partage de la bande passante dans un réseau acyclique

L'inconvénient de l'algorithme *Tree_Allocation* présenté au chapitre 4, est que, les utilisateurs sont contraints de miser sur des chemins continus partant nécessairement de la racine de l'arbre. Toujours dans le cas d'un arbre, dans le but de rendre la politique de mise plus flexible et en se basant sur les techniques de la programmation linéaire, nous proposons dans ce chapitre un algorithme qui calcule le mécanisme de Clark-Groves pour le partage de la bande passante dans un arbre tout en permettant aux utilisateurs de miser sur des chemins entre des noeuds quelconques de l'arbre sans aucune restriction. Avec cet algorithme, l'hypothèse qui fait que les chemins misés doivent nécessairement partir de la racine de l'arbre est levée, ainsi, nous pouvons considérer la structure arbre du réseau comme étant un graphe acyclique puisque la racine n'a plus aucune importance. Nous parlons alors d'allocation dans un réseau acyclique.

6.1. Formulation du problème

(a)- soit un réseau de télécommunication acyclique à L liens de transmissions; $\mathcal{L} = \{1,\ldots,L\}$ pour lesquels les quantités de bande passante sont $Q = \{Q^1,\ldots,Q^L\}$ (voir figure 6.1), et un ensemble d'utilisateurs $\mathcal{N} = \{1,\ldots,n\}$.

(b)- Les mises des utilisateurs sont de la forme $s_i = (q_i,\ p_i^1,\ldots,p_i^{q_i}\ ,\ r_i)$, où :
- q_i est le nombre d'unités de bande passante demandées par le i^{eme} utilisateur pour sa route r_i,
- p_i^k est appelée *mise élémentaire*, c'est le prix que i est prêt à payer pour obtenir la k^{eme} unité demandée avec $1 \le k \le q_i$ et $p_i^1 \ge p_i^2 \ge \ldots \ge p_i^{q_i}$.

(c)- Une route $r_i \in \{0,1\}^L$ est un ensemble de liens qui forment un chemin continu entre deux noeuds quelconques du réseau. $r_{i,l} = 1$ si le lien $l \in \mathcal{L}$ est dans le chemin de i.

(d)- Pour chaque lien $l \in \mathcal{L}$, le vendeur impose un prix de réserve p_0^l.

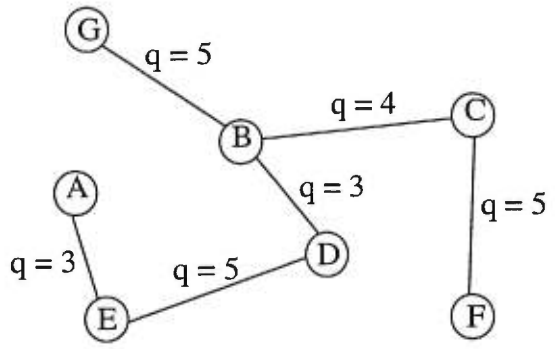


Figure 6.1. Réseau de télécommunication acyclique

6.2. Règle d'allocation

Etant donné la formulation ci-dessus, une règle d'allocation F est de la forme :

$$F : \quad [0,Q]^n \times [0,\infty)^n \times \{0,1\}^{L\times n} \rightarrow [0,Q]^n \times [0,\infty)^n \times \{0,1\}^{L\times n}$$
$$s = (qs, ps, rs) \quad \rightarrow F(s) = (f(s), c(s), rF(s))$$

où :

- $f(s) = (f_1(s), \ldots\ldots, f_n(s))$; $f_i(s)$ est la quantité de bande passante allouée à i pour la route demandée r_i.

- $c(s) = (c_1(s), \ldots\ldots, c_n(s))$; $c_i(s)$ est la charge à payer de i.

- $rF(s) = (rF_1(s), \ldots\ldots, rF_n(s))$; $rF_i(s)$ est l'ensemble des liens alloués à i.

Un utilisateur obtient soit l'ensemble des liens demandés, même avec une quantité de bande passante inférieure à ce qu'il a demandé, soit rien du tout, autrement dit, $f_i(s) \leq q_i$ et $rF(s) = rs$. F est ainsi réalisable si $\forall s$,

$$f_i(s) \leq qs_i$$

$$c_i(s) \leq \sum_{k=1}^{f_i(s)} p_i^k$$

$$f(s)rs \leq Q$$

$$rF(s) = rs$$

6.3. Principe d'allocation

L'allocation est donnée par un algorithme qui se base entièrement sur la programmation linéaire. L'idée de base est de transformer le problème en un programme linéaire équivalent dont la fonction objective consiste à maximiser la volonté à payer et dont les contraintes technologiques sont données par les capacités finies des liens.

Après avoir défini le programme linéaire, le calcul de la règle d'allocation F s'effectue en deux étapes:

Etape 1: dans la première étape, on calcule la solution optimale du programme linéaire généré, et à partir de celle-ci, on déduit les quantités allouées des utilisateurs $f_i(s)$.

Etape 2: Dans la deuxième étape, on calcule les charges à payer $c_i(s)$; pour cela, on considère le sous-ensemble d'utilisateurs $\mathcal{M} = \{1, ..., m\}$ ayant reçu des allocations. Pour chaque $i \in \mathcal{M}$, $c_i(s)$ est calculée moyennant une petite modification du programme linéaire original et une réoptimisation de celui-ci en partant de la solution optimale calculée à l'étape 1.

Ainsi, le calcul de règle d'allocation F s'effectue moyennant une optimisation, plus m réoptimisations.

6.4. L' algorithme *AG_Allocation*

La règle d'allocation $F(s)$ pour un graphe acyclique, est donnée par l'algorithme *AG_Allocation* (*Acyclic Graph Allocation*) qui consiste en trois étapes, à savoir :

Etape 1: Génération du programme linéaire

(a)- fonction objective

La règle d'allocation $F(s)$ est efficace si elle maximise la volonté à payer des utilisateurs. La fonction objective du programme linéaire à définir se doit alors de maximiser cette quantité. Elle peut se définir comme suit :

Pour chaque mise $s_i = (q_i, p_i^1,\ldots,p_i^{q_i}, r_i)$, on associe une variable de décision x_i^j à chaque unité demandée ($j = 1\ldots q_i$) dont le coût est p_i^j : le prix que i est prêt à payer pour cette unité. La variable x_i^j vaut 1 si i obtient cette unité, 0 sinon. Ainsi,

$$0 \leq x_i^j \leq 1 \qquad (6.1)$$

La fonction objective z s'écrit alors :

$$max\ z = \sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} p_i^j x_i^j \qquad (6.2)$$

(b)- Les contraintes technologiques

Etant donné que les capacités de transmission des liens sont finies, on associe à chaque lien $l \in \mathcal{L}$ une contrainte de capacité de la forme :

$$\sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} \theta_i^l \, x_i^j \leq Q^l \qquad (6.3)$$

θ_i^l est 1 si le lien $l \in r_i$, 0 sinon.

Exemple : si on a un ensemble de 3 mises s_1, s_2, s_3 qui demandent le lien [AB] tel que :

s_1=(3, [60, 50, 30], [AB])
s_2=(2, [40, 20], [AB])
s_3=(1, [10], [AB])

alors la contrainte de capacité associée à [AB] serait :

$$x_1^1 + x_1^2 + x_1^3 + x_2^1 + x_2^2 + x_3^1 \leq 3$$

D'après (6.1), (6.2) et (6.3), le programme linéaire résultant est :

$$max\ z = \sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} p_i^j x_i^j \tag{6.4}$$

$$\text{sujet à :} \quad \sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} \theta_i^l x_i^j \leq Q^l \qquad l \in \mathcal{L} \tag{6.5}$$

$$0 \leq x_i^j \leq 1 \qquad i = 1, \ldots, n\ ,\ j = 1 \ldots q_i \tag{6.6}$$

En ajoutant des variables d'écarts pour écrire les contraintes technologiques sous forme d'équations, le programme linéaire (6.4)-(6.6) devient :

$$max\ z = \sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} p_i^j x_i^j \tag{6.7}$$

$$\text{sujet à :} \quad \sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} \theta_i^l x_i^j + e_l = Q^l \qquad l \in \mathcal{L} \tag{6.8}$$

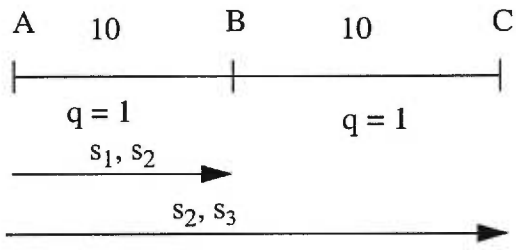
$$0 \leq x_i^j \leq 1 \qquad i = 1, \ldots, n\ ,\ j = 1 \ldots q_i \tag{6.9}$$

$$e_l \geq 0 \qquad l \in \mathcal{L} \tag{6.10}$$

la variable e_l, peut être vue comme étant le nombre d'unités du lien l allouées au vendeur lui même (initialement $e_l = Q^l$). Etant donné que le vendeur attribue à l un prix de réserve $p_0^l > 0$, deux interprétations du problème sont possibles. Pour bien les comprendre, nous nous proposons de les présenter à travers l'exemple suivant :

Exemple : Soit un réseau à deux liens [AB], [BC], ayant chacun une capacité de une unité de bande passante et un prix de réserve de 10 UM. Soit un ensemble de quatre utilisateurs A_1 , A_2 , A_3 , A_4 dont les mises sont :

s_1=(1, [13], [AB])
s_2=(1, [13], [AB])
s_3=(1, [22], [AC])
s_4=(1, [22], [AC])



1ère *interprétation*: L'allocation optimale est d'allouer [AB] à A_1 et [BC] au vendeur lui-même. Le revenu théorique du vendeur est 13 + 10 = 23 UM, alors que son revenu réel n'est que 13 UM. En adoptant cette interprétation, on doit injecter les prix de réserve dans la fonction objective du programme linéaire (6.7)-(6.10), autrement dit, chaque variable d'écart e_l qui représente l'allocation du vendeur pour le lien l, aura p_0^l comme coût et non un coût nul. Par conséquent, le programme linéaire (6.7)-(6.10) devient :

$$max\ z = \sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} p_i^j x_i^j + \sum_{l \in \mathcal{L}} p_0^l e_l$$

$$\text{sujet à :} \quad \sum_{i \in \mathcal{N}} \sum_{j=1}^{q_i} \theta_i^l x_i^j + e_l = Q^l \qquad l \in \mathcal{L}$$

$$0 \le x_i^j \le 1 \quad i = 1, \ldots, n \ , \ j = 1 \ldots q_i$$

$$e_l \ge 0 \quad l \in \mathcal{L}$$

$2^{ème}$ *interprétation*: L'allocation optimale est d'allouer [AB] et [BC] à A_3, le revenu réel du vendeur est alors 22 UM. Dans ce cas, les variables d'écarts e_l auront des coûts nuls et le programme linéaire (6.7)-(6.10) reste inchangé.

Dans ce qui suit, nous optons pour la deuxième interprétation pour être conforme avec *Tree_Allocation*. En effet, dans ce dernier, l'allocation calculée maximise le revenu réel du vendeur et non théorique.

Etape 2 : Calcul des allocations $f_i(s)$

Pour calculer les quantités allouées des utilisateurs $f_i(s)$, on calcule d'abord la solution optimale du programme linéaire généré et à partir de celle-ci, étant donné que x_i^j est égal à 1 si i obtient la $j^{ème}$ unité demandée, 0 sinon, l'allocation $f_i(s)$ d'un utilisateur i se calcule comme suit :

$$f_i(s) = \sum_{j=1}^{q_i} x_i^j$$

Etape 3: Calcul des charges à payer $c_i(s)$

Soit $\mathcal{M}$ l'ensemble d'utilisateurs dont la demande a été totalement ou partiellement satisfaite, $\mathcal{M} = \{1, ..., m\}$. Pour calculer la charge à payer $c_i(s)$ par un utilisateur $i \in \mathcal{M}$, on affecte aux variables de décision x_i^j $(j = 1...q_i)$ le coût 0 dans la fonction objective du programme linéaire, et en partant de la solution optimale, on réoptimise le programme linéaire modifié.

Soient :

- z^* : la valeur de la fonction objective associée à l'ancienne solution optimale (calculée à l'étape 2). C'est le montant que les utilisateurs ayant reçu des allocations (y compris i) sont prêts à payer pour les unités de ressources obtenues.

- z^*_{-i} : la valeur de la fonction objective associée à la solution optimale du programme linéaire modifié. Celle-ci n'est autre que la valeur optimale de la fonction objective du programme linéaire qu'on aurait généré si i n'était pas présent.

Ainsi, selon le principe d'exclusion-compensation, la charge à payer $c_i(s)$ d'un utilisateur i se calcule comme suit :

$$c_i(s) = max\left\{\sum_{j=1}^{a_i(s)} p_i^j - (z^* - z^*_{-i}) \ , cm_i\right\}$$

avec

$$cm_i = f_i(s) \sum_{l \in \mathcal{L}} p_0^l \, r_{i,l}$$

cm_i est la charge minimale de l'utilisateur i pour son allocation $f_i(s)$. C'est le nombre d'unités obtenues $f_i(s)$, multiplié par le prix de réserve de son chemin qui est égal à la somme des prix de réserves des liens appartenant à ce chemin ($\sum_{l \in L} p_0^l \, r_{i,l}$ avec $r_{i,l} = 1$ si le lien l est dans le chemin de i).

Par conséquent, pour calculer $c(s)$, il faut effectuer au total m réoptimisations.

6.5. Exemple

Considérons la spécification du réseau acyclique illustrée par la figure 6.3, supposons que le prix de réserve de chacun des liens est 10 UM. Soit un ensemble de 4 utilisateurs $\{A_1, A_2, A_3, A_4\}$ dont les mises sont :

s_1=(1, [22], [BC])
s_2=(2, [28, 26], [CD])
s_3=(1, [30], [CE])
s_4=(1, [32], [DE])

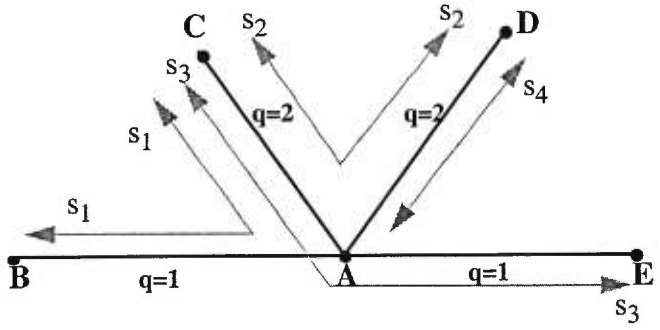


Figure 6.2. Spécification d'un réseau acyclique

Etape 1 : Génération du programme linéaire

Le programme linéaire associé à cette enchère est :

$$max\ z = 22x_1^1 + 28x_2^1 + 26x_2^2 + 30x_3^1 + 32x_4^1$$

$$\begin{aligned}
&[AB] \quad x_1^1 \le 1\\
&[AC] \quad x_1^1 + x_2^1 + x_2^2 + x_3^1 \le 2\\
&[AD] \quad x_2^1 + x_2^2 + x_4^1 \le 2\\
&[AE] \quad x_3^1 \le 1
\end{aligned}$$

$$0 \le x_i^j \le 1 \quad i = 1, \ldots, 4 \ , \ j = 1, \ldots, q_i$$

En écrivant ce programme linéaire sous forme d'équations, on obtient :

$$max\ z = 22x_1^1 + 28x_2^1 + 26x_2^2 + 30x_3^1 + 32x_4^1$$

$$\begin{aligned}
&x_1^1 + e_1 = 1 && [AB]\\
&x_2^1 + e_2 = 1\\
&x_2^2 + e_3 = 1\\
&x_3^1 + e_4 = 1 && [AE]\\
&x_4^1 + e_5 = 1\\
&x_1^1 + x_2^1 + x_2^2 + x_3^1 + e_6 = 2 && [AC]\\
&x_2^1 + x_2^2 + x_4^1 + e_7 = 2 && [AD]
\end{aligned}$$

$$0 \le x_i^j \le 1 \qquad i = 1, \ldots, 4 \ , \ j = 1, \ldots, q_i$$

Etape 2 : Calcul des allocations $f_i(s)$

Calcul de la solution optimale : en utilisant la méthode révisée du simplexe [Wins 94, BT 97, SM 89] pour la résolution du programme linéaire ci-dessus, on obtient à l'optimum :

$x_B = (x_2^1, x_2^2, x_3^1, x_4^1, e_1, e_3, e_7) = (1, 0, 1, 1, 1, 1, 0)$
$c_B = (28, 26, 30, 32, 0, 0, 0)$; $z^* = 90$

$x_N = (x_1^1, e_2, e_4, e_5, e_6)$
$c_N = (-4, -2, -4, -32, -26)$

Ainsi, les allocations des utilisateurs sont :
$f_1(s) = 0; f_2(s) = 1; f_3(s) = 1; f_4(s) = 1.$

Etape 3 : Calcul des charges à payer.

<u>charge de A2</u> : $c_2(s)$?

Pour calculer $c_2(s)$, on décrémente à zero les coûts des variables de décisions relatives à s_2 $c(x_2^1) = 0$, $c(x_2^2) = 0$, et on teste si la solution de base associée à z^* est toujours optimale.

$x_B = (x_2^1, x_2^2, x_3^1, x_4^1, e_1, e_3, e_7)$
$c_B = (\mathbf{0}, \mathbf{0}, 30, 32, 0, 0, 0)$

Test de l'optimalité: en calculant les nouveaux coûts marginaux des variables hors-base nous obtenons :

$x_N = (x_1^1, e_2, e_4, e_5, e_6)$
$c_N = (22, 0, -30, -32, 0)$

$c(x_1^1)$ devient positif, alors la solution courante n'est plus optimale.

Calcul de la nouvelle solution optimale:
$x_B = (x_1^1, x_3^1, x_4^1, e_1, e_2, e_3, e_7) = (1, 1, 1, 0, 1, 1, 1)$
$c_B = (22, 30, 32, 0, 0, 0, 0)$; $z^*_{-2} = 84$

$x_N = (x_2^1, x_2^2, e_4, e_5, e_6)$;
$c_N = (-22, -22, -8, -32, -22)$;

Ainsi, $c_2(s) = max\{28 - (90 - 84), 20\} = 22$ UM

<u>charge de A3</u> : $c_3(s)$?

on affecte à x_3^1 le coût 0;
$x_B = (x_2^1, x_2^2, x_3^1, x_4^1, e_1, e_3, e_7)$
$c_B = (28, 26, \mathbf{0}, 32, 0, 0, 0)$;

Test de l'optimalité :
$x_N = (x_1^1, e_2, e_4, e_5, e_6)$

$c_N = (4, 2, -26, 32, 26)$

Calcul de la nouvelle solution optimale:

$x_B = (x_1^1, x_2^1, x_2^2, x_4^1, e_1, e_3, e_4) = (1, 1, 0, 1, 0, 1, 1)$

$c_B = (22, 28, 26, 32, 0, 0, 0)$; $z^*_{-3} = 82$

$x_N = (x_3^1, e_2, e_5, e_6, e_7)$

$c_N = (-22, -2, -28, -22, -4)$

Ainsi, $c_3(s) = max\{30 - (90 - 82),\ 20\} = 22$ UM

<u>charge de A4</u> : $c_4(s)$?

on affecte à x_4^1 le coût 0;

$x_B = (x_2^1, x_2^2, x_3^1, x_4^1, e_1, e_3, e_7)$

$c_B = (28, 26, 30, \mathbf{0}, 0, 0, 0)$;

Test de l'optimalité :

$x_N = (x_1^1, e_2, e_4, e_5, e_6)$

$c_N = (-4, -2, -4, -32, -26)$

La solution de base associée à z^* est toujours optimale. En affectant le coût 0 à x_4^1, z^* devient égale à $z^*_{-4} = 58$.

Ainsi, $c_4(s) = max\{32 - (90 - 58),\ 10\} = 10$ UM qui est le prix de réserve de [AD].

6.6. Limite de *AG_Allocation*

Dans le programme linéaire généré (6.7)-(6.10), en supposant que $0 \leq x_i^j \leq 1$ on risque dans certains cas particuliers d'avoirs des solutions fractionnaires bien que dans les simulations que nous avons effectuées nous avons toujours obtenu des résultats entiers. Dans ce qui suit, nous présentons un cas où nous obtenons des résultats fractionnaires.

Considérons la structure du réseau donnée par la figure 6.3.

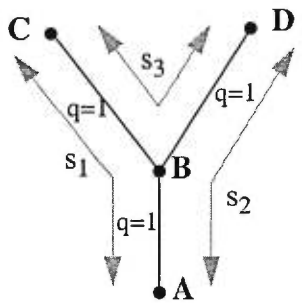


Figure 6.3. Spécification d'un réseau acyclique

Soit un ensemble de 3 utilisateurs A_1 , A_2 , A_3 dont les mises sont :

s_1=(1, [10], [AC])

s_2=(1, [8], [AD])

s_3=(1, [6], [BC])

L'allocation optimale serait d'allouer à chacun des utilisateurs une demi-unité, soit $f_1(s) = f_2(s) = f_3(s) = 1/2$ ce qui maximise la volonté à payer qui est dans ce cas de $10/2 + 8/2 + 6/2 = 12$. Ce résultat ne peut pas être accepté car on cherche a avoir des solutions entières étant donné qu'une unité est indivisible. Le résultat voulu dans ce cas est d'allouer les tronçons [AB] et [AC] à l'utilisateur A_1.

6.7. Conclusion

Dans ce chapitre, nous avons proposé un algorithme *AG_Allocation* qui calcule le mécanisme de Clark-Groves pour le partage et la tarification de bande passante dans un réseau acyclique tout en permettant aux utilisateurs de miser sur des chemins entre des noeuds quelconques du réseau sans aucune restriction. L'idée de l'algorithme est de transformer le problème en un programme linéaire équivalent dont la fonction objective consiste à maximiser la volonté à payer et dont les contraintes sont définies à partir des capacités finies des liens de transmission. Dans une première étape, l'algorithme calcule les quantités allouées en effectuant une optimisation du programme linéaire généré. Ensuite, dans une deuxième étape, il calcule les charges à payer, en effectuant une série de réoptimisations, une pour chaque utilisateurs ayant reçu une allocation. L'inconvénient de cette algorithme est que dans certains cas nous risquons d'obtenir des solutions fractionnaires. Ainsi, cette algorithme serait plus efficace et plus utile si nous considérions des quantités continues et non des unités.

CHAPITRE 7

Implantation et résultats d'expérimentation

Dans les chapitres précédents, nous avons présenté deux algorithmes pour le partage et la tarification de la bande passante dans les réseau de télécommunication selon le principe d'exclusion-compensation du mécanisme Clark-Groves. Le premier algorithme *Tree_Allocation*, considère un réseau ayant la structure d'un arbre et suppose que les utilisateurs misent sur des chemins partant nécessairement de la racine de l'arbre. Le deuxième algorithme *AG_Allocation*, considère un réseau acyclique et permet aux utilisateurs de miser sur des chemins entre des noeuds quelconques du réseau. Afin de tester le bon fonctionnement de ces deux algorithmes, nous les avons implanté en langage de programmation JAVA.

Ce chapitre est organisé comme suit : dans un premier temps, nous décrivons notre implantation pour chacun des deux algorithmes, ensuite, nous analysons les résultats obtenus par chacun des algorithmes et nous vérifions que *Tree_Allocation* est un cas particulier de *AG_Allocation*. Enfin, nous donnons le temps d'exécution de chacun des deux algorithmes en faisant varier la taille du réseau et le nombre de mises.

7.1. Implantation

7.1.1. Spécification des données

Afin d'augmenter la convivialité de l'interface, nous avons opté pour une entrée de données textuelle. Les données fournies en entrée sont : la spécification du réseau et la liste mises.

7.1.1.1. Spécification du réseau

La spécification du réseau, correspond à l'ensemble des liens du réseau, leur capacité de transmission, ainsi que leur prix de réserve. La structure de réseau utilisée pour implanter chacun des deux algorithmes, est celle d'un arbre.

La spécification est fournie sous forme d'un fichier texte dont chaque ligne comprend: un *noeud*, son *père*, la *capacité du lien* entre ces deux noeuds et son *prix de réserve*. La toute première ligne du fichier spécifie la racine de l'arbre. Considérant le réseau illustré par la figure 7.1, la spécification textuelle relative à ce réseau est donnée par la figure 7.2.

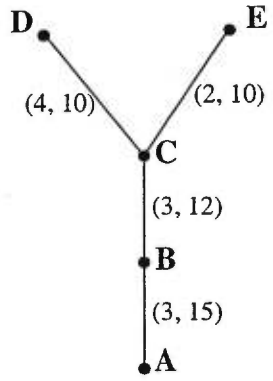


Figure 7.1. Réseau à 4 liens ayant la structure arbre,
(x, y) : x est la capacité du lien, y est son prix de réserve

```
A       null      null      null
B       A         3         15
C       B         3         12
D       C         4         10
E       C         2         10
```

Figure 7.2. Spécification textuelle d'un réseau

7.1.1.2. Spécification des mises

Etant donné un réseau de télécommunication à L liens de transmission interconnectés sous forme d'un arbre, les mises sont de la forme :

$$s_i = (q_i, \ p_i^1, \ldots, p_i^{q_i}, r_i)$$

où :

- q_i est le nombre d'unités de bande passante demandées par le i^{eme} utilisateur,
- p_i^k est une mise élémentaire, c'est le prix que i est prêt à payer pour la k^{eme} unité demandée.

- $r_i \in \{0,1\}^L$ est un ensemble de liens qui forment :
  1/ un chemin continu partant de la racine de l'arbre dans le cas de *Tree_Allocation*.
  2/ un chemin continu entre deux noeuds quelconques de l'arbre dans le cas de *AG_Allocation*.

$r_{i,l} = 1$ si le lien l est dans le chemin de i.

En soumettant sa mise, un utilisateur ne spécifie pas la route r_i qui correspond à son chemin mais plutôt :

1/ le noeud destination de son chemin dans le cas de *Tree_Allocation*, sous entendu que le noeud de départ est la racine de l'arbre.

2/ les deux noeuds extrémités de son chemin dans le cas de *AG_Allocation*.

lors de l'enregistrement des mises, les routes r_i sont calculées.

L'ensemble des mises est fourni en entrée sous forme d'un fichier texte dont chaque ligne spécifie une mise de la forme 7.1 dans le cas d'un arbre et 7.2 dans le cas d'un graphe acyclique.

```
#ID   To.     q     VP[]                    (7.1)
#ID   From    To    q     VP[]              (7.2)
```

où :

- `#ID` : est l'identificateur de l'utilisateur.
- `From` : spécifie le noeud de départ du chemin.
- `To` : spécifie le noeud destination du chemin.
- `q` : est la quantité demandée
- `VP[]` : est le vecteur des mises élémentaires.

7.1.2. Structures de données

7.1.2.1. Structure du réseau

Pour implanter chacun des deux algorithmes *Tree_Allocation* et *AG_Allocation*, nous avons utilisé la structure arbre. Cette structure est définie comme étant un tableau de noeuds `NodeList[]`, chaque noeud est défini par la structure `Node` suivante :

```
Struct Node {
  string name, fatherName;
  int    capacity;
  int    linkNumber;
  int    NbrOfChildren;
  double minimumPrice;
}
```

- `name` : nom du noeud.
- `fatherName` : nom du noeud père.
- `capacity` : capacité du lien entre `name` et `fatherName`.
- `linkNumber` : indice du lien dans le vecteur route.
- `NbrOfChildren` : nombre de fils du noeud. Ce champs est mis à jour au fur et à mesure les noeuds sont insérés.
- `minimumPrice` : prix de réserve du lien entre `name` et `fatherName`.

7.1.2.2. Structure des mises

Pour enregistrer les mises, nous avons utilisé la structure `Bid` definie comme suit :

```
Struct Bid {
   int      bidderID;
   int      quantity;
   double   prices[];
   int      route[];
   double   minimumTopay;
}
```

- `bidderID` : c'est l'identificateur de l'utilisateur.
- `quantity` : c'est la quantité de bande passante demandée.
- `prices[]` : c'est un vecteur de mises élémentaires.
- `route[]` : c'est la route relative au chemin misé,

  `route[i]=1`, si le $i^{ème}$ lien de l'arbre appartient au chemin.

- `minimumTopay` : c'est le prix de réserve du chemin misé.

l'ensemble des mises est enregistré dans un tableau `BidList[]` de structure `Bid`.

7.1.3. Génération automatique des mises : `BidsRandom()`

Afin de pouvoir tester nos algorithmes sur un grand nombre de mises, nous avons défini une méthode qui génère des mises aléatoires définies ainsi :

`BidsRandom(Tree tree, int Nbr)` où :

- `tree` : est la structure du réseau.
- `Nbr` : est le nombre de mises à générer.

Cette méthode se base entièrement sur la fonction de génération de nombre aléatoires. Les mises générées sont de la forme (7.1) dans le cas de *Tree_Allocation*, et (7.2) dans le cas de *AG_Allocation*.

7.1.4. Calcul des routes : `FindRoute()`

Etant donné une mise fournie en entrée, pour calculer la route du chemin spécifié par cette mise, nous avons défini une méthode `FindRoute()` qui prend en arguments la structure de l'arbre et deux noeuds quelconques. Cette méthode, se doit de trouver un ensemble des liens formant un chemin continu entre ces deux noeuds et retourner la route associée.

7.1.5. Implantation de *Tree_Allocation*

Nous rappelons que l'algorithme *Tree_Allocation* consiste en 4 étapes :

Etape 1 : choisir une feuille de l'arbre.

Etape 2 : Allouer le lien directement rattaché à cette feuille selon le règle *DPSP*.

Etape 3 : Mettre à jour l'ensemble des mises et les allocations courantes.

Etape 4 : Supprimer cette feuille et ce lien de l'arbre et retourner à l'étape 1 si l'arbre n'est pas vide.

Architecture de l'implantation de Tree_Allocation

La figure 7.3 illustre le shéma de calcul des allocations selon l'algorithme *Tree_Allocation*.

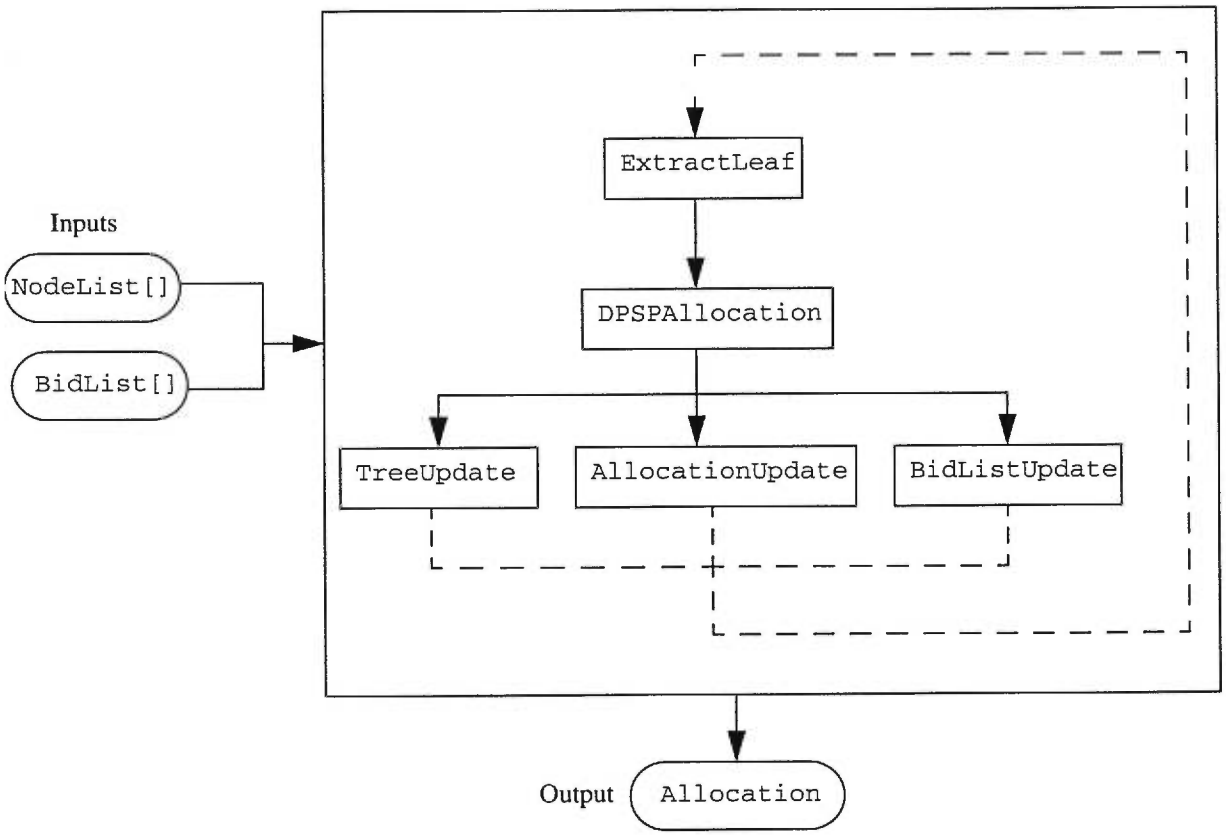


Figure 7.3. Schéma de calcul des allocations selon *Tree_Allocation*

Description des modules

- **ExtractLeaf** : extraction d'une feuille de l'arbre.
- **DPSPAllocation** : allocation du lien rattaché à la feuille extraite selon la règle *DPSP*.
- **AllocationUpdate** : mise à jour des allocations courantes.
- **BidListUpdate** : mise à jour de l'ensemble des mises.
- **TreeUpdate** : suppression de la feuille extraite de l'arbre

Ces modules sont utilisés d'une manière itérative jusqu'à ce que l'arbre devient vide.

7.1.6. Implantation de *AG_Allocation*

Nous rappelons que l'algorithme *AG_Allocation* consiste en 4 étapes à :

Etape 1: Transformer le problème en un programme linéaire PL à maximiser.

Etape 2: Calculer la solution optimale de PL

Etape 3: Déduire les quantités allouées à partir de la solution optimale calculée.

Etape 4: Effectuer une série de réoptimisations de PL pour calculer les charges à payer.

Architecture de l'implantation de AG Allocation

La figure 7.4 illustre le schéma de calcul des allocations selon l'algorithme *AG_Allocation*.

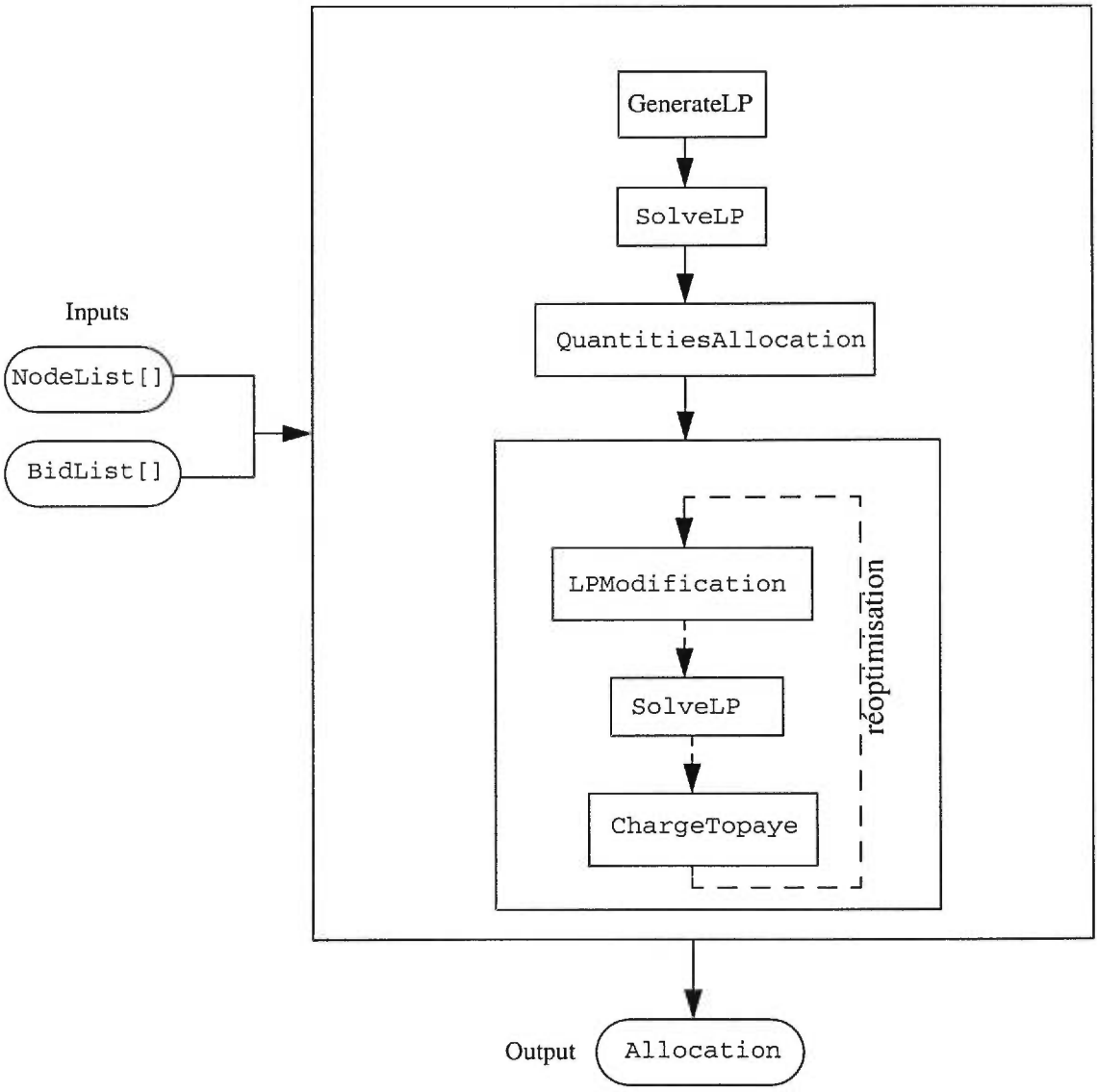


Figure 7.4. Schéma de calcul des allocations selon *AG_Allocation*

Implantation de l'algorithme de simplexe

Pour la résolution du programme généré, nous n'avons pas utilisé un outil existant (comme le progiciel *C-plex*); plutôt, nous avons défini une méthode `SolveLP` qui implante la méthode révisée du simplexe.

Description des modules

- **`GenerateLP`** : ce module consiste à générer à partir de la structure du réseau et la liste des mises un programme linéaire dont la fonction objective consiste à maximiser la volonté à payer, et dont les contrainte sont définies à partir des capacités finies des liens du réseau.
- **`SolveLP`** : ce module se doit de calculer la solution optimale du programme linéaire généré selon la méthode révisée du simplexe.
- **`QuantitiesAllocation`** : ce module consiste à calculer les quantités allouées à partir de la solution optimale du programme linéaire.
- **`LPModification`** : ce module consiste à modifier le programme linéaire original afin de réduire la mise d'un utilisateur à zéro.
- **`ChargeToPay`** : ce module consiste à calculer la charge à payer d'un utilisateur à partir de la solution optimale du programme linéaire original et celle du programme linéaire modifié.

7.2. Résultats d'expérimentation

7.2.1. *Tree_Allocation*

Spécification du réseau

Considérons le réseau de télécommunication R à 10 liens interconnectés sous forme d'un arbre comme le montre la figure 7.5. Soit A la racine de l'arbre, toutes les mises partent ainsi de A vers n'importe quel noeud.

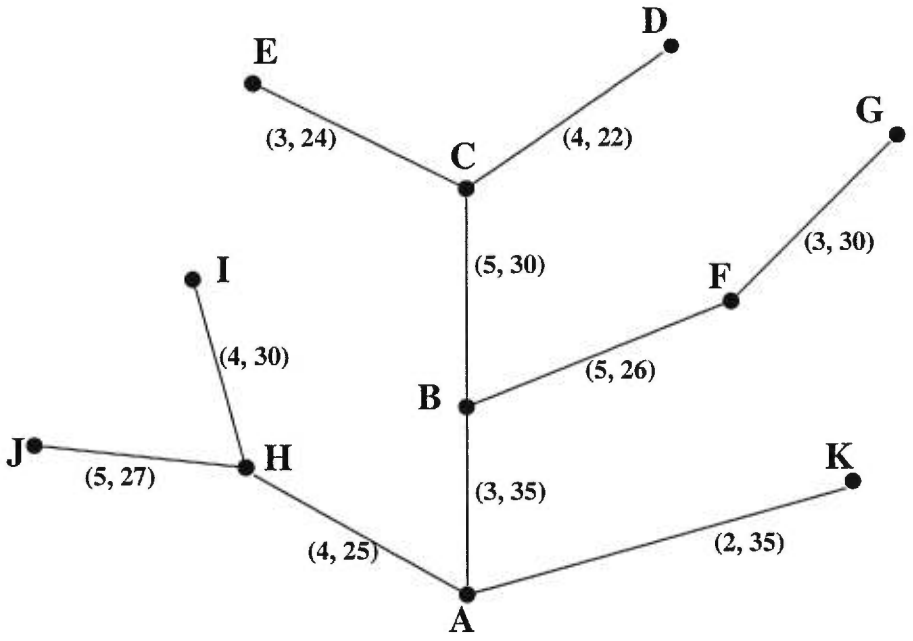


Figure 7.5. Réseau acyclique à 11 noeuds

Liste des mises

Etant donné le réseau R de la figure 7.5, nous avons généré un ensemble $S1$ de 40 mises aléatoires. Ces mises sont données par le tableau 7.1.

Tableau 7.1. Liste des mises $S1$

#ID	To	q	VP[ ]
1	E	2	[79.0 75.0]
2	G	4	[110.0 108.0 88.0 80.0]
3	C	1	[87.0]
4	E	5	[81.0 81.0 75.0 70.0 70.0]
5	E	4	[106.0 99.0 89.0 86.0]
6	F	1	[72.0]
7	B	3	[105.0 98.0 61.0]

Tableau 7.1. Liste des mises *S*1

#ID	To	q	VP[ ]
8	D	1	[107.0]
9	E	1	[71.0]
10	D	2	[79.0 72.0]
11	E	3	[110.0 87.0 81.0]
12	E	1	[94.0]
13	B	1	[88.0]
14	I	3	[109.0 82.0 81.0]
15	D	4	[107.0 77.0 72.0 68.0]
16	J	1	[83.0]
17	G	4	[105.0 104.0 72.0 43.0]
18	F	4	[88.0 86.0 82.0 79.0]
19	E	1	[88.0]
20	H	5	[93.0 78.0 70.0 68.0 60.0]
21	I	1	[73.0]
22	J	1	[80.0]
23	C	2	[96.0 74.0]
24	B	5	[107.0 71.0 66.0 61.0 57.0]
25	F	3	[90.0 86.0 80.0]
26	E	4	[99.0 98.0 83.0 80.0]
27	J	2	[73.0 70.0]
28	J	3	[98.0 96.0 84.0]
29	G	2	[78.0 73.0]
30	J	3	[67.0 66.0 64.0]
31	J	3	[84.0 77.0 72.0]
32	H	3	[94.0 72.0 66.0]
33	C	3	[107.0 101.0 76.0]
34	B	3	[99.0 95.0 77.0]
35	H	5	[70.0 61.0 60.0 59.0 51.0]
36	E	3	[108.0 87.0 86.0]
37	K	5	[76.0 66.0 63.0 63.0 57.0]
38	C	5	[107.0 77.0 71.0 65.0 64.0]
39	B	1	[72.0]
40	G	1	[63.0]

Résultats d'allocation

Etant donné le réseau R et l'ensemble des mises $S1$, les résultats d'allocation obtenus sont donnés par le tableau 7.2.

Tableau 7.2. ***Tree_Allocation*(*R*, *S*1)**

#ID	$f_i(s)$	$c_i(s)$
2	2	215.0
11	1	108.0
14	1	93.0
28	2	177.0
32	1	93.0
37	2	70.0

Analyse et discussion

- Prenant comme exemple le #2, il obtient les 2 premières unités de bande passante demandées pour son chemin de A vers E à une charge totale de 215 UM et pour les quelles il était prêt à payer 218 UM (voir tableau 7.2). D'après le principe d'exclusion compensation, la charge de #2 est la somme de ce qui étaient prêt à payer ceux qui auraient obtenu ces deux unités en absence de #2. Pour vérifier ce résultat, nous avons enlevé #2 de la liste des mises $S1$, et recalculé les allocations en absence de #2; les résultats obtenus sont donnés par le tableau 7.3.

Tableau 7.3. ***Tree_Allocation*****(*R*, *SI*\{#2})**

#ID	$f_i(s)$	$c_i(s)$
8	1	107.0
11	1	108.0
14	1	93.0
28	2	177.0
32	1	93.0
36	1	107.0
37	2	70.0

En comparant 7.1 et 7.2, nous constatons qu'en l'absence de #2, #8 et #36 obtiennent chacun sa première unité demandée pour la quelle, #8 est prêt à payer 107 UM, et 36 est prêt à payer 108 UM (voir tableau 7.3). Ainsi, nous vérifions bien que #2 paie pour les deux unités obtenues $107 + 108 = 215$ UM.

- D'après 7.3. et 7.4, le #11 obtient une unité pour laquelle il paye 108 UM en présence de #2, et 107 UM en absence de #2. Ainsi, nous vérifions bien que les utilisateurs sont en situation d'interdépendance stratégique.

- En effectuant le même calcul en absence de #28, nous obtenons les résultats donnés par le tableau 7.4.

Tableau 7.4. ***Tree_Allocation*(*R*, *S1*\{#28})**

#ID	$f_i(s)$	$c_i(s)$
2	2	215.0
11	1	108.0
14	1	83.0
20	1	83.0
31	2	83.0
32	1	83.0
37	1	70.0

En absence de #28, #20 et #31 obtiennent chacun une unité. D'après *S*1, #20 est prêt à payer pour sa première unité demandée 93 UM, et #31 est prêt à payer 84 UM, Ainsi, nous vérifions bien que #28 paye bel et bien 93 + 84 = 177 UM.

7.2.2. AG_Allocation

Spécification du réseau

Reprenons le réseau *R* de la figure 7.5, et considérons-le comme un graphe acyclique du fait que les utilisateurs peuvent maintenant miser sur des chemins entre des noeuds quelconques.

Liste des mises

Etant donnée le réseau *R*, soit l'ensemble *S*2 des mises générées aléatoirement du tableau 7.5

Tableau 7.5. Liste des mises *S2*

#ID	Fom	To	q	VP[ ]
1	C	E	5	[72.0 70.0 68.0 63.0 61.0]
2	E	B	5	[108.0 101.0 97.0 88.0 86.0]
3	F	C	2	[95.0 87.0]
4	B	K	5	[106.0 95.0 88.0 84.0 81.0]
5	F	C	3	[94.0 90.0 88.0]
6	K	J	6	[107.0 102.0 98.0 90.0 79.0 74.0]
7	E	B	6	[88.0 77.0 76.0 71.0 70.0 70.0]
8	E	D	5	[110.0 91.0 87.0 79.0 75.0]
9	J	H	6	[110.0 99.0 89.0 82.0 77.0 77.0]
10	E	B	5	[82.0 78.0 72.0 67.0 66.0]
11	F	C	5	[106.0 93.0 92.0 70.0 67.0]
12	I	D	6	[110.0 108.0 84.0 71.0 70.0 65.0]
13	I	J	1	[80.0]
14	K	F	4	[76.0 72.0 66.0 60.0]
15	D	K	2	[83.0 75.0]
16	J	H	2	[98.0 79.0]
17	G	B	4	[88.0 79.0 77.0 74.0]
18	F	D	5	[84.0 76.0 70.0 68.0 64.0]
19	J	C	6	[109.0 106.0 96.0 90.0 81.0 72.0]
20	H	E	6	[97.0 90.0 88.0 83.0 81.0 77.0]
21	I	C	7	[109.0 93.0 80.0 78.0 71.0 70.0]
22	F	K	6	[94.0 85.0 81.0 80.0 72.0 65.0]
23	B	E	2	[98.0 86.0]
24	I	G	4	[109.0 84.0 76.0 64.0]
25	K	E	3	[77.0 66.0 64.0]
26	B	G	6	[108.0 94.0 91.0 87.0 84.0 75.0]
27	J	I	4	[92.0 89.0 81.0 77.0]
28	D	J	5	[76.0 76.0 70.0 64.0 64.0]
29	C	K	5	[105.0 94.0 76.0 72.0 65.0]
30	C	I	4	[85.0 83.0 75.0 55.0]

Résultats d'allocation

Etant donné le réseau R et l'ensemble des mises $S2$, les résultats d'allocation sont donnés par le tableau 7.6.

Tableau 7.6. ***AG_Allocation*(*R*, *S2*)**

ID	$f_i(s)$	z^*	z^*_{-i}	$c_i(s)$
2	2	1512.0	1495.0	192.0
3	1	1512.0	1511.0	94.0
4	1	1512.0	1424.0	70.0
8	1	1512.0	1490.0	88.0
9	3	1512.0	1464.0	250.0
11	1	1512.0	1500.0	94.0
16	1	1512.0	1503.0	89.0
26	3	1512.0	1474.0	255.0
27	1	1512.0	1502.0	82.0
29	1	1512.0	1505.0	100.0

Analyse et discussion

- D'après le tableau 7.6 l'utilisateur #9, par exemple, obtient 3 unités pour lesquelles il paye 250 UM. D'après le principe d'exclusion-compensation, la charge de #9 est la somme de ce qui étaient prêt à payer ceux qui auraient obtenu ces deux unités en absence de #9. Pour vérifier ce résultat, nous avons enlevé #9 de la liste des mises $S2$, et recalculé les allocations; les résultats obtenus sont donnés par le tableau 7.7.

Tableau 7.7. ***AG_Allocation*(*R*, *S2*\{#9})**

#ID	$f_i(s)$	z^*	z^*_{-i}	$c_i(s)$
2	2	1464.0	1447.0	192.0
3	1	1464.0	1463.0	94.0
4	1	1464.0	1385.0	70.0
8	1	1464.0	1442.0	88.0
11	1	1464.0	1452.0	94.0
16	1	1464.0	1371.0	27.0
26	3	1464.0	1426.0	255.0
27	3	1464.0	1286.0	171.0
13	1	1464.0	1463.0	79.0
29	1	1464.0	1457.0	100.0

D'après les tableaux 7.6 et 7.7, nous vérifions bien qu'en l'absence de #9, la volonté à payer pour les allocations reçues devient $z^* = 1464$ UM, et que : le #27 obtient sa deuxième et troisième unités demandées pour lesquelles il est prêt à payer respectivement 89 et 81 UM, Aussi, le #13 obtient sa première unité demandée pour la quelle il est prêt à payer 80 UM. Ainsi, d'après le principe d'exclusion-compensation, #9 doit payer pour ces 3 unités obtenues $89 + 81 + 80 = 250$ UM, ce qui vérifie bien que le résultat donné par le tableau 7.6.

7.2.3. Comparaison des résultats

Le premier algorithme *Tree_Allocation* suppose que le réseau a la structure d'un arbre et que les utilisateurs misent sur des chemins partant nécessairement de la racine de

l'arbre. Le deuxième algorithme *AG_Allocation*, considère cette même structure du réseau comme étant un graphe acyclique et suppose que les utilisateurs peuvent miser sur des chemins entre des noeuds quelconques du réseau. Ainsi, on peut dire que *Tree_Allocation* est un cas particulier de *AG_Allocation*. Par conséquent, si on se restreint aux conditions d'allocation dans un arbre, *Tree_Allocation* et *AG_Allocation* devraient donner les mêmes résultats pour une même spécification de données. Pour vérifier ce fait, nous avons repris le réseau R de la figure 7.5 et l'ensemble des mises $S1$ donné par le tableau 7.1, et nous avons recalculé les allocations avec cette fois-ci l'algorithme *AG_Allocation*. Les résultats obtenus sont donnés par le tableau 7.8.

Tableau 7.8. *AG_Allocation*(*R*, *S1*)

#ID	$f_i(s)$	z^*	z^*_{-i}	$c_i(s)$
2	2	867.0	864.0	215.0
11	1	867.0	865.0	108.0
14	1	867.0	851.0	93.0
28	2	867.0	850.0	177.0
32	1	867.0	866.0	93.0
37	2	867.0	825.0	70.0

Analyse et discussion

En comparant les tableaux 7.7 et 7.8 nous vérifions bien que dans les conditions d'allocation de *Tree_Allocation*, les deux algorithmes *Tree_Allocation* et *AG_Allocation* fournissent les mêmes résultats. Comme dans le cas de *Tree_Allocation*, nous avons enlevé #2 de la liste des mises, et recalculé les allocations. Les résultats obtenus sont donnés par le tableau 7.9.

Tableau 7.9. ***AG_Allocation*(*R*, *S1*\{#2})**

#ID	$f_i(s)$	z^*	z^*_{-i}	$c_i(s)$
8	1	864.0	864.0	107.0
11	1	864.0	861.0	107.0
14	1	864.0	848.0	93.0
28	2	864.0	847.0	177.0
32	1	864.0	863.0	93.0
36	1	864.0	863.0	107.0
37	2	864.0	722.0	70.0

Nous vérifions très bien, comme dans le cas de *Tree_Allocation*, c'est le #8 et #36 qui auraient partagé les deux unités obtenues par #2 si ce dernier n'était pas présent.

Ainsi, *Tree_Allocation* est bel et bien un cas particulier de *AG_Allocation*.

7.2.4. Variante de l'algorithme *AG_Allocation*

Dans le chapitre précédent, nous avons présenté deux interprétations possibles de l'efficacité de la règle d'allocation:

$1^{ère}$ *interprétation*: la première interprétation maximise le revenu théorique du vendeur peu importe son revenu réel. Dans ce cas, on injecte les prix de réserve dans la fonction objective du programme linéaire généré.

$2^{ème}$ *interprétation*: la deuxième interprétation maximise le revenu réel du vendeur peu importe son revenu théorique. Dans ce cas on n'injecte pas les prix de réserve dans la fonction objective du programme linéaire généré. Cette interprétation est utilisée dans *AG_Allocation*.

Pour pouvoir comparer les résultats d'allocation donnés par chacune des deux interprétations pour une même spécification de données, nous avons implanté une deuxième version de l'algorithme *AG_Allocation* qui calcule les allocations selon la première interprétation. Celle-ci, est désignée par *AG_Allocation2*.

Reprenons le réseau R de la figure 7.1 et l'ensemble des mises $S1$ du tableau 7.2, en calculant les allocations selon *AG_Allocation2* nous obtenons les résultats donnés par le tableau 7.10.

Tableau 7.10. ***AG_Allocation2(R, S1)***

#ID	$f_i(s)$	z^*	z^*_{-i}	$c_i(s)$
7	1	1582.0	1572.0	95.0
14	1	1582.0	1575.0	102.0
20	2	1582.0	1554.0	143.0
24	1	1582.0	1573.0	98.0
32	1	1582.0	1559.0	71.0
34	1	1582.0	1581.0	98.0
37	2	1582.0	1510.0	70.0

Analyse et discussion

D'après les tableaux 7.8 et 7.10, dans le cas de *AG_Allocation* le revenu réel du vendeur $\sum c_i(s)$ = 756 UM, alors que celui ci n'est que 677 UM dans le cas de *AG_Allocation2* bien que la volonté à payer z^* donnée par *AG_Allocation2* (1582UM) est supérieure à celle obtenue par *AG_Allocation* (867UM).

7.3. Analyse du temps d'exécution

Dans cette section, nous donnons le temps moyen d'exécution de chacun des deux algorithmes en faisant varier la taille du réseau et le nombre de mises. Pour ceci, nous avons considéré deux réseaux : le premier contient 50 noeuds, le deuxième 100 noeuds.

Cas de Tree_Allocation

La figure 7.6 donne le temps moyen d'exécution de *Tree_Allocation* en fonction des mises dans le cas du réseau à 50 noeuds (1) et celui à 100 noeuds (2).

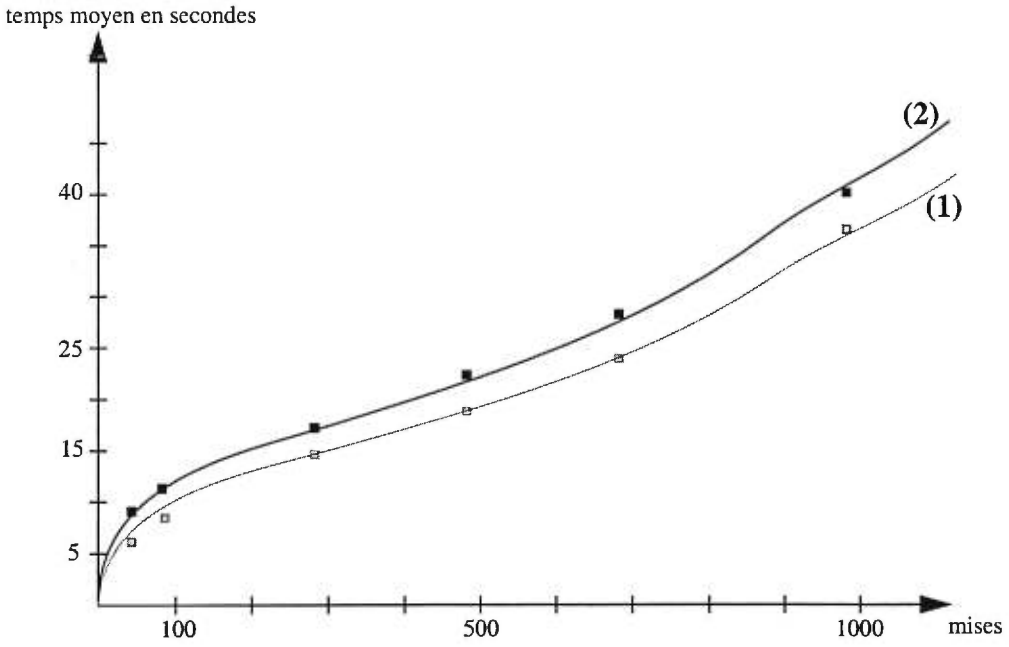


Figure 7.6. Temps moyen d'exécution de *Tree_Allocation*

Cas de AG_Allocation

La figure 7.7 donne le temps moyen d'exécution de *AG_Allocation* en fonction du nombre de mises dans le cas du réseau à 50 noeuds (1) et celui à 100 noeuds (2).

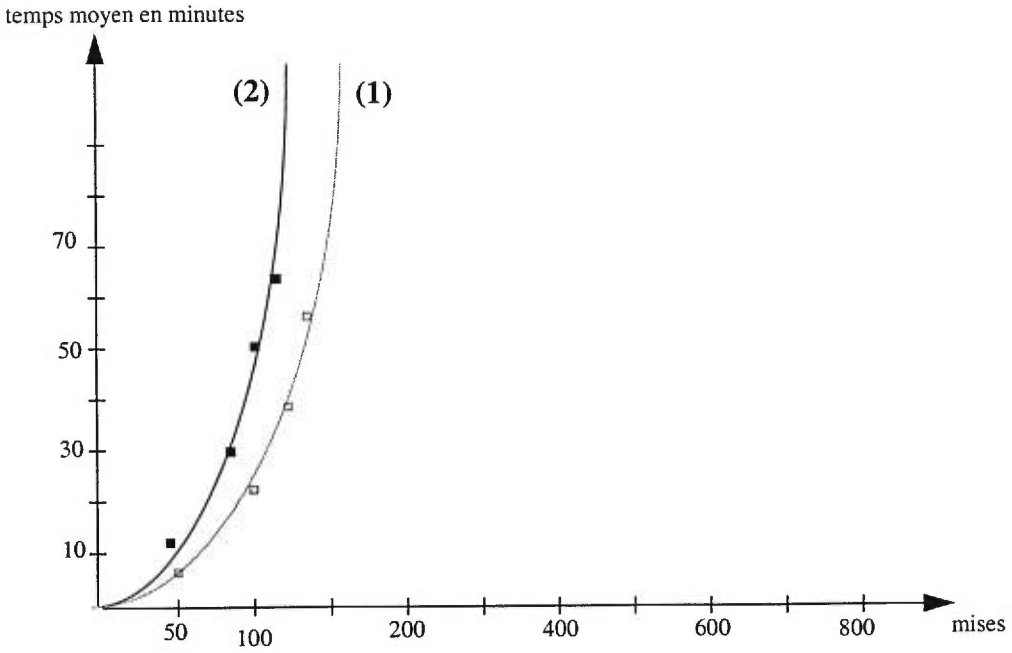


Figure 7.7. Temps moyen d'exécution de *AG_Allocation*

Performance de la machine utilisée

Nous tenons à préciser que la simulation a été effectuée sur une station *sun* SPARCstation 5 dont la fréquence est 80Mhz.

Analyse et discussion

Les résultats de simulation montrent clairement que *Tree_Allocation* est beaucoup plus rapide que *AG_Allocation*. A titre de comparaison, pour un réseau à 100 noeuds et un ensemble de 100 mises, *Tree_Allocation* nécessite en moyenne 12 secondes pour calculer l'allocation alors que *AG_Allocation* exige en moyenne 50 minutes. Ceci se justifie par le fait que, avec *Tree_Allocation* nous calculons L fois (nombre de liens du réseau) la règle *DPSP* en un temps linéaire, chaque fois sur un sous-ensemble de mises, alors que dans le cas *AG_Allocation*, nous considérons simultanément la totalité des mises, ce qui explose rapidement le nombre des variables de décision du programme linéaire généré, et par conséquent le temps nécessaire pour une optimisation de celui-ci, devient important.

7.4. Conclusion

Dans ce chapitre, nous avons vérifié que les deux algorithmes *Tree_Allocation* et *AG_Allocation* calculent bel et bien les allocations des utilisateurs selon le principe d'exclusion-compensation. Aussi, nous avons vérifié que si l'on se restreint aux conditions d'allocation dans un arbre, les deux algorithmes fournissent le même résultat ce montre que *Tree_Allocation* est un cas particulier de *AG_Allocation*. En terme de temps d'exécution, dans le cas particulier d'un arbre *Tree_Allocation* est beaucoup plus rapide que *AG_Allocation*, ceci se justifie par le fait que pour calculer l'allocation, *Tree_Allocation* exécute un certain nombre de fois la règle *DPSP* en un temps linéaire, alors que *AG_Allocation* effectue une optimisation du programme linéaire généré pour calculer les quantités allouées, plus, une série de réoptimisations pour calculer les charges à payer, et étant donné que la taille du programme linéaire augmente très rapidement lorsque le nombre de mises s'élève, une optimisation devient coûteuse en terme de temps d'exécution.

Conclusion

Par nature, les réseaux de communication sont des systèmes distribués où de multiples utilisateurs sont en compétition pour des ressources finies. Dans ce mémoire, nous avons considéré une approche par la théorie des jeux: les mécanismes d'enchères, pour l'allocation de la bande passante des voies de transmission.
Dans une première partie de notre travail, nous avons proposé une nouvelle règle d'enchère appelée *DPSP* (*Dynamic Progressive Second Price*) pour le partage de la bande passante d'un lien de transmission entre plusieurs utilisateurs selon le mécanisme de Clark-Groves. Cette règle, est une généralisation de la règle *PSP* (*Progressive Second Price*) proposée par *Semret et al.* dans [LS 97] vers le cas où les utilisateurs spécifient un prix pour chaque unité demandée et non pas un seul prix unitaire. En se basant sur cette nouvelle règle, nous avons proposé un algorithme *Tree_Allocation* qui est une version améliorée de l'algorithme proposé par *Semret et al.* pour la mise en oeuvre du mécanisme Clark-Groves dans le contexte d'un réseau de liens ayant la structure d'un arbre. L'inconvénient de cet algorithme, est que, les utilisateurs sont contraints à miser sur des chemins continus partant nécessairement de la racine de l'arbre.

Afin de rendre les conditions de mise plus flexibles, en se base sur les outils de la programmation linéaire, nous avons proposé dans la deuxième partie de notre travail, un deuxième algorithme *AG_Allocation* qui calcule le mécanisme Clark-Groves dans le contexte d'un réseau de liens acyclique tout en permettant aux utilisateurs de miser sur des chemins entre des noeuds quelconques du réseau sans aucune restriction.

Ainsi, avec cet algorithme, si on se restreint à la structure arbre qui est un cas particulier de graphe acyclique, l'hypothèse qui fait que les chemins misés doivent nécessairement partir de la racine de l'arbre est levée. Par conséquent, nous pouvons dire que *Tree_Allocation* est un cas particulier de *AG_Allocation*. L'inconvénient de cet algorithme est que dans certains cas on peut obtenir des résultats fractionnaires. Ainsi, cet algorithme serait plus efficace si nous considérons des quantités continues et non des unités.

D'après les résultats d'expérimentation, si on se restreint aux conditions de *Tree_Allocation*, nous avons bien vérifié que les deux algorithmes fournissent les mêmes résultats pour une même spécification de données, sauf que dans ce cas particulier, *Tree_Allocation* est plus efficace de point de vue temps d'exécution. En effet, pour calculer les allocations, *Tree_Allocation* exécute L fois (nombre de liens de l'arbre) la règle *DPSP* en un temps linéaire, chaque fois sur un sous ensemble de mises, alors que *AG_Allocation*, considère toute les mises à la fois, à partir desquelles il génère un programme linéaire et effectue une série d'optimisations de celui-ci, ainsi, étant donné qu'à chaque mise est associée plusieurs variables de décision dans le programme linéaire généré, lorsque le nombre de mises augmente, le nombre de variables de décisions explose rapidement, et par suite le temps nécessaire pour une optimisation du programme linéaire devient important.

Comme une continuation à notre travail, il serait intéressant de considérer le cas plus général où les utilisateurs spécifient dans leur mises des quantités continues et de l'appliquer sur des réseaux arbitraires qui peuvent contenir des cycles entre les noeuds. Ainsi, en considérant ce type de réseaux si l'un des liens formant le chemin demandé par un utilisateur est saturé, on peut laisser tomber ce chemin et emprunter éventuellement un autre.

Bibliographie

[Ago 94] Agorics Technical Report ADd004P. *Real-time video delivery with market-based ressource allocation*. Technical report, Sun Microsystems Labs, SunConnect, Agorics, Inc., 1994.

[BT 97] D. Bertsimas et J. N. Tsitsiklis, K. *Introduction to Linear Optimization*, Athena Scientific, Belmont, Massachusetts, 1997.

[CIRANO] Centre Interuniversitaire de Recherche en ANalyse des Organisations http://www.cirano.umontreal.ca

[Cla 71] E. H. Clark. *Multipart pricing of public goods*. Public Choice, 8:17-33, 1971

[Cram 94] P. Cramton, *Money Out of Thin Air : The Nationwide Narrowband PCS Auction*, Working Paper, University of Maryland, 1994

[DP 94] G. Demanage et J. P. Ponssard, *Théorie des jeux et analyse économique*. Presses Universitaires de France, 1994.

[FCC] Federal Communications Commission, http://www.fcc.gov/wtb/auctions.html.

[FT 91] D. Fudenberg et J. Tirole., *Game Theory*. The IMT Press, Cambrige, Massachusetts, London, England, 1991.

[Gro 73] T. Groves. *Incentives in teams*. Econometrica, 41(3):617-631, July 1973.

[Hal 96] F. Halsall, *Data Communications, Computer Networks and Open Systems*. Addison-Wesley Publishers Ltd. United Kingdom, 1996.

[HRP 95] R. M. Harstad, M. H. Rothkopf, et A. Pekec, *Computationally manageable combinatorial auctions*. Technical Report 95-09, DIMACS, University of Rutgers, 1995.

[KL 95] Y. A. Korilis et A. A. Lazar, *On the existence of equilibria in noncooperative optimal flow control*, J. ACM. vol. 42, 95 1995.

[KLO 95] Y. A. Korilis et A. A. Lazar et A. Orda, *Architecting noncooperative networks*, IEEE J. Select. Areas Commun., vol. 13, September 1995.

[KSY 85] J. Kurose, M. Schwartz, et Y. Yemini, *A microeconomic approach to decentralized optimization of channel access policies in multiaccess networks*, in Proc. 5th Int. Conf. on Distributed Computing Systems IEEE, pp.70-77, May 1985.

[LS 96] A. A. Lazar et N. Semret, *A Resource Allocation Game with Application to Wireless Spectrum.* Center for Telecommunications Reseach, Columbia University, New York. Technical Report CU/CTR/TR 456-96-22, October 1997.

[LS 97] A. A. Lazar et N. Semret, N., *Auctions for Network Ressource Sharing*, Technical Report CU/CTR/TR 468-97-02, Center for Telecommunications Reseach, Columbia University, New York, February 1997.

[LS 98a] A. A. Lazar et N. Semret, N., *The Progressive Second price Auction Mechanism for Network Resource Sharing*. Center for Telecommunications Reseach, Columbia University, New York, January 1998.

[LS 98b] A. A. Lazar et N. Semret, N., *Design, Analysis and Simulation of the Progressive Second Price Auction for Network Bandwidth Sharing*, Technical Report CU/CTR/TR 487-98-21, Center for Telecommunications Reseach, Columbia University, New York, April 1998.

[MD 88] M. S. Miller et K. E. Drexler. *Markets and computations : Agorics open systems*. In Bernardo Huberman, editor, The Ecology of Computations. Elsevier Science Publishers, North Holland, 1988.

[MWG 95] A. MAS-Collel, M. D. Winston et J. R. Green. *Microeconomic*. New York Oxford. Oxford University. Press 1995.

[Mou 81] H. Moulin, *Théorie des jeux pour l'économie et la politique*. Hermann, Paris 1981.

[Mil 87] P. R. Milgrom. *Advances in Economic Theory: Fifth World Congress*, chapter Auction Theory, pages 1-31. Number 12 in Econometric Society Monographs. Cambridge university Press, 1987.

[Mye 81] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58-73, February 1981.

[ORS 93] A. Orda, R. Rom, et N. Shimkin, *Competitive routing in multiuser communication networks*, IEEE/ACM Trans. Networking, vol.1, October 1993.

[SCEH 96] S. Shenker, D. Clark, D. Estrin, et S. Herzog.N. *Pricing in computer networks: Reshaping the research agenda.* ACM Comput. Commun. Review, 26(2):19-43, 1996.

[Sem 96] N. Semret, *T-REX, The Ressource EXchange: A distributed networking game on the WWW.* http://argo.ctr.columbia.edu:1024/~nemo/Trex/trex.html.

[She 94] S. Shenker. *Making greed work in networks : A game-theoretic analysis of switch service diciplines.* In Proc. ACM SIGCOMM, 1994.

[SM 89] H. M. Salkin et K. Mathur, *Foundations of Integer Programming*, Elsevier Science Publishing Co., INC., New York, 1989.

[Tan 96] A. Tanenbaum, *Réseaux*, InterEdition, Paris, 1996

[Vick 61] W. Vickrey, *Counterspeculation, auctions and competitive sealed tenders.* Journal of Finance, 16, 1961.

[Wins 94] W.L. Winston, *Operations Research, Applications and Algorithms*, Wadsworth Publishing Company, California, 1994.

[WHS 92] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, et W. S. Stornetta. Spawn : A distributed computational economy. IEEE Trans. on Software Engineering, 18(2):103-117, February 1992.