CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

Une heuristique de recherche avec tabous pour la conception de réseaux de distribution de contenu électronique

Ghislain Dubuc
Tolga Bektas
Jean-François Cordeau
Gilbert Laporte

Septembre 2007

CIRRELT-2007-34

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

Université Laval · UQÀM · HEC Montréal · École Polytechnique Montréal · Université de Montréal

Une heuristique de recherche avec tabous pour la conception de réseaux de distribution de contenu électronique

Ghislain Dubuc[1], Tolga Bektas[2], Jean-François Cordeau[1,*], Gilbert Laporte[1]

[1] Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT), Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7 et HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

[2] School of Management, University of Southampton, Highfield Southampton, Royaume Uni, SO17 1BJ

Résumé. Cet article décrit un algorithme de résolution heuristique pour le problème de conception de réseaux de distribution de contenu électronique. Pour chaque serveur-mandataire préalablement localisé, le contenu à reproduire doit être sélectionné pour permettre de le rapprocher le plus possible des utilisateurs. Chacun des clients doit également être affecté à un seul de ces serveurs-mandataires afin de minimiser les coûts de communication dans le réseau. L'algorithme proposé utilise la méthode de recherche avec tabous. Cette méthode, qui utilise la notion de mémoire, offre l'avantage de poursuivre la recherche au-delà des optima locaux en permettant une détérioration momentanée de la solution. Dans le but de tester l'algorithme, une série de nouvelles instances sont générées. Celles-ci nous permettent d'ajuster les nombreux paramètres qui composent notre méthode afin de la rendre plus robuste. Une quinzaine d'instances extraites de la littérature sont ensuite utilisées pour permettre la comparaison de la performance de l'algorithme. Les résultats obtenus montrent que la méthode proposée permet d'améliorer les meilleures solutions connues pour la majorité des instances testées, et ce, sans grande variabilité d'un essai à l'autre.

Mots-clés. Contenu électronique, serveurs-mandataires, conception de réseaux, heuristiques, recherche avec tabous.

Remerciements. Ce travail a bénéficié d'un appui financier du Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), subventions 227837-04 et 39682-05.

---

* Auteur correspondant: jean-francois.cordeau@hec.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2007

1 Introduction

L'utilisation du réseau Internet est maintenant devenue chose courante pour une majorité de personnes dans les pays développés. De plus, rien ne semble indiquer une diminution dans le nombre d'utilisateurs puisque la croissance industrielle des pays qualifiés de plus pauvres permet à un nombre toujours plus grand de personnes d'avoir accès à cette technologie. Cependant, à mesure que les gens se familiarisent avec ce nouveau moyen de communication, leurs exigences et leurs attentes deviennent plus difficiles à combler, en particulier en ce qui concerne le temps d'accès. En effet, selon une étude menée par la compagnie Zona Research, le délai maximal nécessaire pour charger une page web ne devrait pas dépasser huit secondes, sans quoi l'utilisateur normal risque de perdre patience et de quitter le site en question (Zona Research, 1999). Le temps d'accès a encore plus d'importance dans le domaine des achats en ligne où le client peut quitter la page d'une entreprise en un seul clic.

Bien que la rapidité des connexions offertes par les fournisseurs d'accès Internet augmente constamment année après année, il en va de même pour la taille des objets utilisés sur les différents sites. Ainsi, non seulement retrouvons-nous encore des objets textes, d'une taille moyenne de 4 à 8 Ko, et des images d'environ 14 Ko (Saroiu, 2004), mais de plus en plus d'éléments vidéos sont maintenant intégrés dans les pages web. Ces objets multimédias ont une taille médiane d'environ 1,1 Mo, mais peuvent souvent atteindre plusieurs mégaoctets (Acharya et Smith, 1998). Il devient donc primordial pour les entreprises de trouver une façon d'accélérer le transfert des données des différents serveurs jusqu'aux clients pour éviter des temps d'attente inacceptables.

C'est dans cette optique que les réseaux de distribution de contenu électronique (RDCÉ) ont commencé à faire leur apparition à la fin des années 90. Comme l'expliquent Bektaş et al. (2006a), « l'objectif d'un RDCÉ est de copier le contenu à partir d'un serveur d'origine vers des serveurs-mandataires géographiquement dispersés à partir desquels les clients reçoivent le contenu demandé ». Cette exigence a donc pour effet de diminuer la distance entre le contenu et ses utilisateurs, ce qui permet à la fois d'en accélérer l'accès et d'en diminuer le coût de distribution.

Dans le reste de cette section, nous décrivons le problème étudié. Nous passons en revue les différentes méthodes qui ont été proposées pour le résoudre à la section 2. Ensuite, la modélisation du problème est expliquée dans la section 3. À la section 4, l'algorithme que nous avons développé est présenté alors que les résultats obtenus sont résumés et analysés à la section 5.

1.1 Énoncé du problème

Le problème précis que nous cherchons à résoudre est celui présenté par Bektaş et al. (2006a,b), qui supposent que les serveurs-mandataires ont déjà été localisés à différents endroits sur le réseau. Soit $G = (V, E)$ un graphe complet non orienté où V est l'ensemble des sommets et $E = \{\{i, j\} : i, j \in V\}$ est l'ensemble des arêtes. L'ensemble des sommets V est divisé en trois sous-ensembles non vides I, J et $S = \{0\}$ où I représente l'ensemble des clients ou utilisateurs, J représente l'ensemble des sommets sur lesquels les serveurs-mandataires sont localisés et S représente le serveur d'origine. Enfin, l'ensemble K est composé de tous les objets qui se retrouvent sur le serveur d'origine.

Le problème à résoudre consiste à déterminer quels objets copier, sur quels serveurs-mandataires les placer et comment assigner les clients à ceux-ci dans le but de minimiser le coût total de transfert dans le réseau. De plus, la taille cumulative des objets copiés sur un serveur-mandataire ne doit en

aucun cas en dépasser la capacité et le délai subi par un client pour accéder à un objet doit être inférieur à une borne prédéfinie, en accord avec les termes de service. Finalement, il est à noter que chaque client doit être assigné à un seul serveur-mandataire duquel il recevra la totalité de ses objets. Ceci implique donc que les clients n'ont jamais directement accès au serveur d'origine et que, dans le cas où un objet demandé ne se trouve pas sur le serveur-mandataire, ce dernier contacte le serveur d'origine et transfère l'objet au client. Cette contrainte se justifie par des considérations de sécurité puisque les clients ne peuvent ainsi modifier le contenu original. Enfin, il est à noter que nous ne remettons pas en question la localisation des serveurs-mandataires sur le réseau. Le problème qui nous intéresse peut donc être qualifié d'opérationnel et doit par le fait même être résolu à une fréquence plus élevée.

2 Revue de la littérature

Il existe de multiples recherches sur le sujet de l'optimisation des RDCÉ. Cependant, bon nombre d'entre elles ne tiennent compte que d'un ou deux des sous-problèmes qui composent le problème général. Nous allons donc passer en revue quelques-unes des différentes recherches qui ont été effectuées sur chacune des parties du sujet.

Le problème de localisation de serveurs-mandataires semble avoir été étudié pour la première fois par Li et al. (1999). Ce problème consiste à placer de façon optimale un certain nombre de serveurs-mandataires sur un réseau et à y assigner les différents clients dans le but de minimiser la latence ou les coûts de communication. Plusieurs auteurs se sont attardés à ce problème. Notons, entre autres, Qiu et al. (2001), Yang et Fei (2003) ainsi que Tang et Xu (2005). Ces derniers introduisent d'ailleurs une contrainte supplémentaire au problème généralement étudié, soit les *conditions de service*. Cette contrainte assure aux clients l'accès au contenu demandé en deçà d'une certaine limite de temps prédéfinie, ce qui constitue une condition de plus en plus souvent imposée dans la réalité.

Le problème de reproduction de contenu consiste, quant à lui, à déterminer sur quels serveurs-mandataires placer des copies des objets à distribuer et ce, en tenant compte de la capacité disponible. Kangasharju et al. (2002) ainsi que Cidon et al. (2002) s'y sont attardés. De façon similaire, la localisation de bases de données sur un réseau d'ordinateurs a été étudiée bien avant l'apparition des RDCÉ. Les recherches effectuées dans ce domaine par Fisher et Hochbaum (1980), Pirkul (1986), Gavish et Suh (1992), Chari (1996) et Hakimi et Schmeichel (1997) sont de bonnes références et offrent des pistes intéressantes pour l'optimisation des RDCÉ.

Différentes combinaisons de sous-problèmes des RDCÉ ont été étudiées. Ainsi, Ryoo et Panwar (2001) ont considéré le problème d'optimisation d'un réseau pour le service de distribution sur demande de contenu multimédia. Leur travail consiste en l'élaboration d'un modèle de résolution déterminant à la fois la capacité des liens de communication, la capacité de chaque serveur et la distribution des différents contenus parmi eux sur un réseau en arbre. De leur côté, Xu et al. (2002) tentent de minimiser les coûts d'écriture et de lecture en faisant la localisation d'au plus M serveurs-mandataires sur un réseau en arbre et en y reproduisant un objet unique. Le problème conjoint de localisation de serveurs et de reproduction d'objets dans un réseau de distribution est également abordé par Xuanping et al. (2003). Ces auteurs proposent une méthode permettant de minimiser le coût total de communication, tout en gardant les coûts de reproduction du contenu sur les serveurs inférieurs à un budget prédéterminé. Dans une autre recherche, Erçetin et Tassiulas (2003) utilisent

la théorie des jeux pour résoudre le problème conjoint de reproduction des objets et d'affectation des clients aux serveurs dans un réseau de distribution. Ils supposent ici que tous les objets possèdent la même taille.

Le problème d'allocation des ressources dans un graphe général pour l'optimisation simultanée de la taille des serveurs-mandataires et du placement des objets a été étudié par Laoutaris et al. (2004). Dans un autre article, les mêmes auteurs tentent à nouveau de résoudre ce problème, mais en supposant cette fois que le réseau possède une topologie hiérarchique où l'assignation des clients est déterminée à l'avance (Laoutaris et al., 2005). Almeida et al. (2004) considèrent le problème conjoint de routage et de localisation des serveurs-mandataires sur une topologie Internet dans le cas de transmission multimédia en continu. Les auteurs se basent sur des études antérieures voulant que les algorithmes gloutons produisent des résultats près de l'optimum pour tenter de minimiser les coûts totaux de livraison du contenu. Une approche nouvelle, dans laquelle le contenu Internet est distribué en utilisant un réseau et des serveurs partagés par plusieurs fournisseurs de réseaux de distribution, fut proposée par Nguyen et al. (2005). Le problème à résoudre consiste à maximiser le profit engendré en déterminant conjointement la localisation des serveurs, leur taille, la reproduction du contenu et le routage.

De leur côté, Bektaş et al. (2007) présentent une méthode exacte pour résoudre le problème conjoint de localisation des serveurs, de reproduction du contenu et de routage des demandes. Par ailleurs, Bektaş et al. (2006a) tentent de résoudre de façon exacte le problème plus opérationnel de reproduction du contenu et de routage des demandes. Ils supposent ici que les serveurs ont déjà été localisés lors de la mise en place du réseau de distribution. Enfin, les mêmes auteurs proposent une heuristique basée sur le recuit simulé à deux niveaux pour résoudre ce même problème (Bektaş et al., 2006b). C'est d'ailleurs ce dernier article qui est à la base de nos recherches. En effet, bien que les résultats obtenus à l'aide de cette heuristique présentent des coûts en majorité inférieurs à ceux obtenus avec un algorithme exact tronqué, il subsite un écart relativement grand par rapport aux bornes inférieures calculées par Bektaş et al. (2006a). La recherche avec tabous étant généralement supérieure au recuit simulé, tant au chapitre de la qualité des résultats que du temps de calcul nécessaire (Soriano et Gendreau, 1997), nous nous proposons donc de reprendre le problème formulé par Bektaş et al. (2006b) pour tenter d'en améliorer les solutions. De plus, la taille de ce problème et sa complextié constituent selon nous des raisons suffisantes de croire qu'une méthode de recherche heuristique représente l'option la plus susceptible d'offir les meilleurs résultats.

3 Modélisation du problème

Pour mieux situer le problème, nous en présentons une formulation mathématique. Un résumé de la notation utilisée est reproduit dans le tableau 1. Le coût unitaire pour transférer un objet entre le serveur-mandataire $j \in J$ et le client $i \in I$ est noté c_{ij}. Il représente le nombre de bonds (nombre de routeurs, passerelles, répéteurs, etc.) qu'un objet doit franchir à l'intérieur d'un lien pour atteindre sa destination. Selon Qiu et al. (2001), il s'agit d'un indicateur approprié de la proximité, de la fiabilité et de la stabilité du lien. Ensuite, nous supposons que tous les serveurs-mandataires possèdent la même capacité notée s_j. La taille de chaque objet $k \in K$ est représentée par b_k et la probabilité pour qu'un client $i \in I$ demande l'objet $k \in K$ est notée λ_{ik}. Le délai encouru pour transférer des données entre les sommets i et j est symbolisé par d_{ij} et, selon Radoslavov et al. (2002), cette valeur est proportionnelle au nombre de bonds qui les séparent. Le temps nécessaire pour un client

i pour recevoir un objet k à partir du serveur-mandataire j est représenté par $t(b_k, d_{ij})$ puisqu'il est fonction de la taille des objets (Huang et Abdelzaher, 2005) et du délai unitaire de transfert. Comme dans Bektaş et al. (2006b), le temps requis par un client i pour accéder à un objet k à partir du serveur d'origine par le serveur-mandataire j est représenté par $t'(b_k, d_{ij}) > t(b_k, d_{ij})$. Enfin, Δ_{ik} symbolise la latence maximale tolérée pour qu'un client i reçoive un objet k, en conformité avec les conditions de service. Deux types de variables de décision binaires sont utilisés dans la formulation du problème. La variable x_{ij} prend la valeur 1 si le client $i \in I$ est affecté au serveur-mandataire $j \in J$, 0 sinon. De plus, $z_{jk} = 1$ si l'objet $k \in K$ est copié sur le serveur $j \in J$, 0 sinon. Nous pouvons donc écrire le problème comme suit :

$$\text{Minimiser} \qquad \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} (b_k \lambda_{ik} c_{ij} x_{ij} z_{jk} + b_k \lambda_{ik} (c_{ij} + c_{j0}) x_{ij} (1 - z_{jk})) \tag{1}$$

sous les contraintes

$$\sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{k \in K} b_k z_{jk} \leq s_j \qquad \forall j \in J \tag{3}$$

$$\sum_{j \in J} t(b_k, d_{ij}) x_{ij} z_{jk} + \sum_{j \in J} t'(b_k, d_{ij}) x_{ij} (1 - z_{jk}) \leq \Delta_{ik} \qquad \forall i \in I, k \in K \tag{4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in I, j \in J \tag{5}$$

$$z_{jk} \in \{0, 1\} \qquad \forall i \in I, j \in J. \tag{6}$$

TAB. 1 – Résumé de la notation utilisée

Ensembles	
I	ensemble des clients
J	ensemble des serveurs-mandataires
K	ensemble des objets
S	ensemble contenant le serveur d'origine
Paramètres	
b_k	taille de l'objet $k \in K$
s_j	capacité du serveur $j \in V$
λ_{ik}	probabilité pour qu'un client $i \in I$ demande l'objet $k \in K$
c_{ij}	coût unitaire de transférer un objet par une arête $\{i, j\} \in E$
d_{ij}	délai unitaire causé par le transfert d'un objet par une arête $\{i, j\} \in E$
$t(b_k, d_{ij})$	temps requis pour transférer un objet k à partir d'un serveur-mandataire j jusqu'à un client i
$t'(b_k, d_{ij})$	temps requis pour transférer un objet k à partir du serveur d'origine jusqu'à un client i via un serveur-mandataire j
Δ_{ik}	borne supérieure sur la latence pour qu'un client i reçoive un objet k
x_{ij}	variable binaire pour l'affectation du client i au serveur-mandataire j
z_{jk}	variable binaire pour la reproduction de l'objet k sur le serveur-mandataire j

Les contraintes (2) garantissent que chaque client sera assigné à un et un seul serveur-mandataire. Les contraintes (3) indiquent que la taille totale des objets copiés sur un serveur-mandataire ne doit pas en dépasser la capacité. Enfin, les contraintes (4) assurent que la latence observée par un client i pour obtenir un objet k est inférieure à une borne supérieure Δ_{ik} préalablement fixée. Nous les retrouvons de plus en plus dans les réseaux de distribution de contenu électronique alors qu'un certain niveau de service est exigé. Ces contraintes (4), tout comme la fonction objectif (1), se divisent en deux parties. Ainsi, le premier segment est utilisé lorsque l'objet demandé par un client est reçu du serveur-mandataire auquel il est assigné. Par contre, lorsque le client doit obtenir l'objet en question du serveur d'origine, le deuxième segment est imposé. Il est à noter que le problème que nous avons à résoudre n'est pas linéaire. En effet, deux variables de décisions sont utilisées et elles s'influencent mutuellement tant dans la fonction objectif (1) que dans les contraintes de latence (4). Ceci a pour effet de compliquer grandement la résolution du problème.

4 Méthodologie

Bektaş et al. (2006a,b) ont prouvé que le RDCÉ est NP-difficile. Pour cette raison et parce que le problème est résolu à un niveau opérationnel, donc à une fréquence élevée, nous avons cru opportun de développer une méthode de résolution heuristique. Nous avons donc mis au point une méthode de recherche avec tabous.

4.1 Solution initiale

L'algorithme suivant permet de construire une solution initiale.

1. Trier les objets en ordre décroissant de popularité globale.
2. Pour tous les serveurs-mandataires $j \in J$, faire
   (a) Poser $n := 1$.
   (b) Si la taille du $n^{\text{ième}}$ objet le plus populaire est plus petite que la capacité résiduelle du serveur-mandataire
      i. Copier cet objet sur le serveur-mandataire.
      ii. Soustraire la taille de l'objet ajouté de la capacité résiduelle du serveur-mandataire.
   (c) $n := n + 1$
3. Pour tous les clients $i \in I$
   (a) Assigner le client au serveur-mandataire qui lui permet de minimiser ses coûts de communication.

La popularité globale, utilisée ci-dessus à l'étape 1 pour trier les objets, est définie par Bektaş et al. (2006b) comme étant la somme des demandes de chaque client i pour un objet k. Toujours selon ces auteurs, c'est ce critère qui réussit le mieux lorsque vient le temps de copier les objets sur les serveurs-mandataires.

4.2 Fonction de pénalité

La solution initiale ainsi obtenue n'est cependant pas nécessairement admissible. En effet, pour permettre à notre algorithme de passer plus facilement d'une solution à une autre lors de la recherche, nous avons autorisé que les solutions violent parfois les différentes contraintes de capacité et de latence à l'aide d'une fonction de pénalité. Par contre, dans la solution initiale, seules les contraintes de latence peuvent être violées puisqu'on ajoute un objet sur un serveur-mandataire seulement si la capacité résiduelle est suffisante. Pour inclure ces différentes pénalités, notre heuristique de recherche avec tabous utilise une fonction de coût $f(s) = c(s) + \alpha q(s) + \beta d(s)$ où

$$
\begin{aligned}
c(s) &= \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}(b_k\lambda_{ik}c_{ij}x_{ij}z_{jk} + b_k\lambda_{ik}(c_{ij}+c_{j0})x_{ij}(1-z_{jk})),\\
q(s) &= \sum_{j\in J}\left[\left(\sum_{k\in K} b_k z_{jk}\right) - s_j\right]^+,\\
d(s) &= \sum_{i\in I}\sum_{k\in K}\left[\left(\sum_{j\in J} t(b_k,d_{ij})x_{ij}z_{jk} + \sum_{j\in J} t'(b_k,d_{ij})x_{ij}(1-z_{jk})\right) - \Delta_{ik}\right]^+.
\end{aligned}
$$

La fonction $q(s)$ représente le nombre d'unités qui excèdent la capacité des serveurs-mandataires alors que la fonction $d(s)$ impose une pénalité proportionnelle au dépassement de la latence permise. Ainsi, lorsqu'aucune contrainte n'est violée, la solution est admissible et $f(s)$ correspond à $c(s)$. Les coefficients α et β, qui prennent une valeur initiale égale à 1, sont de plus ajustés de façon dynamique, après chaque itération. En effet, si une contrainte n'est pas respectée, le coefficient correspondant est multiplié par $(1+\delta)$, alors qu'il est divisé par le même nombre dans le cas contraire. La valeur du paramètre δ est quant à elle fixée préalablement lors de l'analyse de sensibilité qui sera expliquée à la section suivante.

4.3 Espace des solutions et voisinage

L'utilisation des pénalités permet d'explorer un espace des solutions, noté X, composé de toutes les solutions possibles, réalisables ou non. Le voisinage étudié à chaque itération est dénoté $N(s)$ et est composé de toutes les opérations consistant à ajouter ou à retirer un objet sur un serveur-mandataire à partir de la solution courante. Par contre, vu la taille du problème et la fréquence à laquelle il doit être résolu, nous ajoutons également les paramètres π_k et π_j servant à déterminer la taille de l'échantillon aléatoire des objets et des serveurs-mandataires respectivement. Ainsi, à chaque itération, un certain nombre d'objets est choisi au hasard parmi l'ensemble K. Ces objets sélectionnés constituent ensuite l'ensemble B. Le même procédé est enfin utilisé pour déterminer quels serveurs-mandataires considérer. L'ensemble A est ainsi créé. On ne considère donc qu'un sous-ensemble $N'(s) \subset N(s)$ choisi aléatoirement, ce qui allège la recherche et introduit un autre mécanisme permettant d'éviter le cyclage.

4.4 Mécanismes de tabous

Le mécanisme de tabous qui est utilisé consiste à interdire l'inverse d'une transformation effectuée dans les θ dernières itérations. Ainsi, lorsqu'un objet k est ajouté ou enlevé d'un serveur-mandataire j, l'itération à laquelle toute opération utilisant le couple (j,k) n'est plus jugée taboue est conservée en mémoire et est notée τ_{jk}. De cette façon, une solution nécessitant l'utilisation du couple (j,k) ne peut être acceptée si l'étiquette associée à ce couple montre un nombre inférieur à l'itération courante. Ce procédé peut cependant être révoqué temporairement si une solution jugée taboue respecte le critère d'aspiration, c'est-à-dire si elle est réalisable et possède un coût inférieur à celui de la meilleure solution connue. Ainsi, à chaque itération, un sous-ensemble $M(s) \subseteq N(s)$ est créé en y ajoutant toutes les solutions $\bar{s} \in N(s)$ telles que la modification à apporter à s pour obtenir $\bar{s}$ n'est pas taboue ou satisfait le critère d'aspiration.

4.5 Diversification

Pour éviter que la recherche ne s'enlise dans une portion limitée de l'espace des solutions, nous ajoutons un mécanisme de diversification permettant d'encourager l'inclusion d'attributs peu présents dans les solutions déjà visitées. Soit ρ_{jk}, le nombre d'itérations au cours desquelles l'attribut (j,k) a été modifié dans la solution courante et γ, une constante. À chaque itération, on ajoute à la valeur de la solution une pénalité déterminée en multipliant la fréquence de modification de l'attribut ρ_{jk} avec la constante γ et le coût de la solution courante, le tout divisé par le nombre d'itérations effectuées. Ainsi, si une modification est effectuée fréquemment, le coût de la solution qui en résulte sera plus fortement pénalisé par rapport à une modification rarement rencontrée. Enfin, l'importance de la diversification croît proportionnellement avec la valeur de la constante γ. Donc, plus la pénalité est élevée, plus la recherche explore une grande partie de l'espace des solutions.

4.6 Intensification

Pour compléter le mécanisme de diversification et surtout à cause du fait que nous utilisons des échantillons aléatoires pour effectuer notre recherche, un processus d'intensification est ajouté à l'algorithme. Pour réduire le risque que de bonnes solutions soient oubliées, la recherche est reprise ponctuellement à partir de la meilleure solution trouvée en utilisant cette fois l'ensemble du voisinage, et ce, pendant une certaine durée limitée. Au-delà de cette période, l'intensification prend fin et la recherche continue en réduisant de nouveau la taille des échantillons d'objets et de serveurs-mandataires utilisées. Le paramètre μ représente le nombre d'itérations pendant lesquels l'intensification a lieu à chaque période. L'intervalle entre deux périodes d'intensification est désigné ν. Dans la pratique, μ est beaucoup plus petit que ν. Enfin, le moment à partir duquel le mécanisme peut commencer à être utilisé est défini par φ qui indique le pourcentage du temps alloué à la recherche qui a été utilisé avant de pouvoir lancer l'intensification.

4.7 Critère d'arrêt

Le critère d'arrêt utilisé est l'atteinte d'un temps de traitement prédéfini. Ceci permet l'obtention d'un nombre suffisant d'itérations dans un temps acceptable pour toutes les instances du problème

à résoudre.

4.8 Description détaillée de notre algorithme

Les différents concepts ayant été expliqués et détaillés, nous pouvons maintenant présenter notre algorithme pour le RDCÉ. Notons d'abord κ, un compteur d'itérations, ψ, un indicateur du temps utilisé en secondes et η, le temps maximal alloué à l'heuristique de recherche. L'algorithme est initialisé pour les valeurs de η, γ, δ, π_k, π_j, φ, ν, μ et θ (voir section 5) et retourne la meilleure solution admissible identifiée une fois la recherche terminée. Si par contre aucune solution admissible n'est trouvée, un message en ce sens est retourné, de même que la valeur de départ. Afin de contrôler l'application des phases d'intensification, on utilise une variable binaire ω égale à 1 si et seulement si la recherche est en mode d'intensification. Si $\psi \geq \varphi\eta$ et que κ est un multiple de $\mu+\nu$, l'intensification commence à l'itération $\sigma = \kappa$ et se termine à l'itération $\sigma + \mu - 1$.

1. Poser $\kappa = 1$, $\psi = 0$, $\omega = 0$, $\alpha := 1$ et $\beta := 1$. Initialiser η, γ, δ, π_k, π_j, φ, ν, μ et θ. Si s_0 est admissible, poser $s^* := s_0$.
2. Pour tout (j, k), faire
   (a) Poser $\rho_{jk} := 0$ et $\tau_{jk} := 0$
3. Tant que $0 \leq \psi \leq \eta$, faire
   (a) Si $\psi \geq \varphi\eta$ et que κ est un multiple de $\mu + \nu$, alors $\omega = 1$, $s := s^*$, $\sigma = \kappa$, $\pi_j := \pi_k := 1$.
   (b) Choisir aléatoirement $\pi_k|K|$ objets et $\pi_j|J|$ serveurs-mandataires
   (c) Pour tout $j \in A$ et pour tout $k \in B$, faire
      i. Ajouter l'objet k sur le serveur-mandataire j s'il ne si trouve pas déjà. L'enlever sinon.
      ii. Faire l'affectation des clients aux serveurs-mandataires de façon exacte.
   (d) Pour chaque solution $\bar{s} \in N(s)$, faire
      i. S'il existe une opération utilisant le couple (j, k) telle que $\tau_{jk} < \kappa$ ou que $\bar{s}$ est admissible et que $c(\bar{s}) < s^*$, poser $M(s) := M(s) \cup \{\bar{s}\}$.
   (e) Pour toute solution $\bar{s} \in M(s)$ obtenue en modifiant le couple (j, k), faire
      i. Poser $g(\bar{s}) := f(\bar{s}) + \gamma c(s^*)\rho_{jk}/\kappa$.
   (f) Identifier la solution $s' \in M(s)$ minimisant $g(\bar{s})$.
   (g) Pour le couple (j, k) utilisé par la transformation permettant d'obtenir la solution s', faire
      i. Ajouter l'objet k sur le serveur-mandataire j s'il ne si trouve pas déjà. L'enlever sinon.
      ii. Faire l'affectation des clients aux serveurs-mandataires de façon exacte.
      iii. Poser $\tau_{jk} := \kappa + \theta$.
      iv. Poser $\rho_{jk} := \rho_{jk} + 1$.
   (h) Si s' est admissible, faire
      i. Si $c(s') < c(s^*)$, poser $s^* := s'$.
   (i) Si $q(s') = 0$, poser $\alpha := \alpha/(1 + \delta)$; sinon, poser $\alpha := \alpha(1 + \delta)$.

(j) Si $d(s') = 0$, poser $\beta := \beta/(1+\delta)$; sinon, poser $\beta := \beta(1+\delta)$.

(k) Mettre à jour la valeur de ψ.

(l) Si $\omega = 1$ et que $\kappa \geq \sigma + \mu - 1$, alors réinitialiser π_j et π_k et poser $\omega = 0$.

(m) $\kappa = \kappa + 1$.

5 Résultats numériques

5.1 Génération des instances

Dans leurs articles, Bektaş et al. (2006a,b) génèrent aléatoirement les différentes données utilisées pour composer les problèmes à résoudre. Ainsi, pour permettre la comparaison entre les résultats que nous avons obtenus et ceux obtenus par les auteurs précédents, nous devons nous assurer de procéder de la même manière en ce qui concerne la génération des données. Tout d'abord, nous avons utilisé un générateur de topologie Internet (GT-ITM) pour reproduire avec le plus de précision possible les propriétés d'une topologie Internet réelle. Cette façon de faire permet d'obtenir des résultats qui sont proches de la réalité. En effet, comme le mentionnent Qiu et al. (2001), le changement observé dans le niveau de performance de différents algorithmes est négligeable lorsque du bruit est incorporé dans la topologie utilisée. Le fait d'utiliser un générateur de topologie Internet pour simuler un réseau réel n'affecte donc pas significativement les solutions qui seront produites. Enfin, comme nous l'avons mentionné à la section 3, le coût de transfert unitaire entre deux sommets est constitué du nombre de bonds qui les séparent.

La génération de la demande pour les différents objets a été effectuée à l'aide d'une loi de type Zipf (Zipf, 1949). Si nous supposons que les objets sont classés en ordre décroissant de popularité, l'objet i étant le $i^{\text{ième}}$ plus populaire dans la liste, cette loi stipule que la probabilité pour qu'une requête soit destinée à l'objet i est inversement proportionnelle à $1/i^{\alpha}$, où $\alpha \leq 1$. Cette loi a été utilisée par plusieurs auteurs pour générer un modèle de requêtes d'objets sur Internet (Breslau et al., 1999 ; Kangasharju et al., 2002 ; Saroiu et al., 2002 ; Yang et Fei, 2003 ; Laoutaris et al., 2005). De façon plus précise, la probabilité conditionnelle pour qu'une requête soit destinée à l'objet i est déterminée par Bektaş et al. (2006a,b) comme étant

$$P_K(i) = \Omega i^{-\alpha},$$

où

$$\Omega = \left(\sum_{j=1}^{|K|} j^{-\alpha} \right)^{-1}$$

est une constante de normalisation. La loi de Zipf correspond au cas $\alpha = 1$. Selon Breslau et al. (1999), la valeur du paramètre α varie dans la plupart des cas entre 0,64 et 0,83 pour des requêtes d'objets sur des pages Internet. Nous utilisons $\alpha = 0{,}733$ qui est, selon Yang et Fei (2003), la valeur qui correspond le mieux à la loi de Zipf pour des objets multimédias.

En ce qui a trait à la génération des objets constituant l'ensemble K, elle est faite selon une loi uniforme continue sur l'intervalle $[1; 100]$. Ces bornes sont utilisées puisque selon Huang et Abdelzaher (2005), la majorité des objets Internet ont une taille se situant dans cet intervalle (en Ko) et la latence moyenne pour transférer un tel objet est proportionnel à sa taille. L'ensemble K est

ensuite divisé en quatre groupes par Bektaş et al. (2006a,b), chacun d'eux ayant des paramètres de latence différents selon la taille des objets. Ainsi, comme le montre le tableau 2, à chaque objet est associé un délai unitaire u_l^k et une borne de latence Δ_d, calculés en millisecondes. Les auteurs ont ensuite déterminé que le délai nécessaire au transfert d'un objet k sur un lien (i, j) pouvait être calculé comme étant $t(b_k, d_{ij}) = u_l^k d_{ij}$, où $d_{ij} = vc_{ij}$ et v est une constante. Il est à noter que les valeurs associées aux délais unitaires sont tirées de l'article de Johnson et al. (2001), qui étudient les performances, en terme de latence, de Savvis (autrefois Digital Island) et Akamai, deux des joueurs les plus importants dans le domaine des réseaux de distribution de contenu électronique. Enfin, la capacité individuelle des serveurs-mandataires a été fixée à 40% de la somme de la taille des objets de l'ensemble K.

TAB. 2 – Paramètres de latence associés aux objets

Groupe	Taille (Ko)	u_l^k (ms)	Δ_d(ms)
1	$1 \leq b_k \leq 15$	20	100
2	$16 \leq b_k \leq 31$	30	200
3	$32 \leq b_k \leq 63$	40	400
4	$64 \leq b_k \leq 100$	50	600

5.2 Instances de test

Pour que la recherche avec tabous produise de bons résultats, les nombreux paramètres que nous avons décrits à la section précédente doivent être calibrés et ajustés. Il n'existe cependant pas, dans la littérature, de problèmes semblables à ceux que nous étudions sur lesquels nous pourrions effectuer les tests nécessaires. Ainsi, pour nous assurer que notre algorithme soit robuste et pour être en mesure d'en évaluer adéquatement la performance, nous avons dû créer un certain nombre d'instances de problèmes respectant les caractéristiques propres de notre recherche.

Nous avons donc généré dix nouvelles instances qui sont présentées dans le tableau 3. Ces instances ont été créées en utilisant le même procédé que pour les instances originelles. Ainsi, la taille de l'ensemble J a été générée suivant une loi uniforme continue sur l'intervalle $[15; 115]$. L'ensemble K a quant à lui été obtenu de la même manière, mais possède une taille comprise dans l'intervalle $[40; 140]$. Notons que nous avons gardé fixe à 200 la taille de l'ensemble I puisqu'elle influence peu le temps de calcul nécessaire à la résolution des problèmes. De plus, les instances proposées par Bektaş et al. (2006a,b) contiennent toutes ce nombre de clients. Enfin, de nouvelles demandes et de nouveaux ensembles d'objets ont été générés pour chaque instance, selon la procédure décrite précédemment.

5.3 Analyse de sensibilité

Afin de déterminer quelles valeurs donner aux différents paramètres qui gouvernent notre heuristique, nous avons effectué plusieurs tests à l'aide des instances nouvellement générées. En nous inspirant des travaux effectués par Cordeau et al. (1997), les paramètres ont d'abord été fixés à une valeur initiale et nos recherches nous ont ensuite permis d'en déterminer la valeur optimale. Ainsi, nous avons initialement fixé $\delta = 0{,}1$ ce qui a pour effet de conserver des pénalités relativement stables.

TAB. 3 – Nouvelles instances générées

Instances	$\|J\|$	$\|I\|$	$\|K\|$
a	35	200	132
b	78	200	60
c	94	200	84
d	110	200	119
e	38	200	52
f	54	200	115
g	88	200	47
h	45	200	42
i	48	200	106
j	21	200	135

Les pourcentages de la taille des échantillons à considérer (π_k et π_j) ont été initialisés à 0,3. Cette valeur permet d'obtenir un bon compromis entre le nombre d'itérations effectuées et le temps de calcul. Le paramètre θ a lui été fixé à 7 alors que le critère d'arrêt utilisé était $\eta = 3\ 000$ itérations. Enfin, aucune intensification n'est effectuée dans la résolution des instances servant à calibrer les paramètres, les valeurs de μ, ν et φ étant déterminées les dernières.

Il est à noter que dans la version définitive de notre heuristique, le critère d'arrêt est déterminé en fonction du temps de calcul et non pas en fonction du nombre d'itérations. Puisque les premiers paramètres étudiés n'ont aucune influence sur le temps de calcul, nous avons toutefois opté pour un critère d'arrêt basé sur le nombre d'itérations pour faciliter la comparaison des résultats obtenus pour chaque instance. Enfin, les instances ont été testées sur un ordinateur Pentium 4 à 3 GHz fonctionnant à l'aide d'un système d'opération de type Linux. L'algorithme a été programmé en langage C++.

5.3.1 Paramètre γ

Nous avons en premier lieu fait varier la valeur du paramètre γ dans l'intervalle [0,001; 0,1] en résolvant nos 10 instances. Une concentration des meilleures solutions vers la borne supérieure de l'intervalle, c'est-à-dire, pour une valeur de γ égale à 0,1 a cependant été observée. Il semble donc qu'une constante trop petite ne permette pas de pénaliser suffisamment les solutions fréquentes et crée un effet de diversification trop limité. Nous avons donc repris les expérimentations avec de nouvelles bornes et l'intervalle [0,2; 0,6] s'est avéré être plus approprié. Nous remarquons cependant que les résultats obtenus avec les différentes valeurs sont relativement homogènes et peu dispersés. Par contre, nos observations nous ont permis de voir que lorsque γ prend une valeur comprise entre 0,3 et 0,4, les résultats sont légèrement plus intéressants qu'avec les autres valeurs de l'intervalle. Nous avons donc fixé $\gamma = 0{,}35$ dans notre heuristique.

5.3.2 Paramètre δ

En gardant fixes les valeurs des paramètres γ, θ, π_k, π_j ainsi que les trois paramètres d'intensification, nous avons fait varier le paramètre δ dans l'intervalle [0,05; 25,0]. Les résultats ainsi obtenus ne

donnent cependant pas l'impression d'être significativement influencés par les différentes valeurs que peut prendre le paramètre. Nous avons cependant remarqué qu'une valeur de δ inférieure à 0,25 n'offrait généralement pas de solutions intéressantes. Ceci s'explique par le fait qu'une valeur trop faible ne modifie pas suffisamment les pénalités appliquées lorsqu'une contrainte est violée. Ainsi, même après une séquence de plusieurs solutions admissibles consécutives, le coût associé au non-respect d'une contrainte n'est toujours pas assez faible pour inciter la recherche à emprunter ce chemin. Par contre, les résultats obtenus en fixant $\delta = 5$ semblent être ceux qui sont le plus intéressants. Suite à une étude plus approfondie des différentes instances et après avoir testé plusieurs autres valeurs situées dans un petit intervalle concentré autour de 5, nous posons $\delta = 5{,}5$ puisqu'il s'agit de la valeur qui semble être la mieux adaptée pour le paramètre.

5.3.3 Paramètre θ

Les différentes expérimentations que nous avons effectuées nous démontrent que la longueur appropriée de la liste de tabous est influencée par la taille des échantillons utilisés et, par le fait même, par la taille des instances à résoudre. Nous avons donc dû utiliser différentes combinaisons de tailles d'échantillons pour être en mesure de déterminer une valeur θ pouvant être suffisamment robuste pour offrir de bonnes performances dans chaque circonstance. Ainsi, nous n'avons observé que peu de variations dans les résultats obtenus avec une valeur de $1 \leq \theta \leq 15$. Nous avons cependant remarqué que l'utilisation d'une formule tenant compte de la taille des échantillons utilisés génère une longueur de liste qui est mieux adaptée aux différentes instances. De plus, la moyenne des solutions obtenues de cette manière se situe généralement parmi les meilleures rencontrées lors de nos essais. La valeur de la liste de tabous a donc été déterminée comme étant $\theta = \sqrt{\pi_j|J| + \pi_k|K|}$ après différentes expérimentations.

5.3.4 Paramètres π_k et π_j

Pour évaluer la taille des échantillons à utiliser, nous avons délaissé le critère d'arrêt basé sur le nombre d'itérations pour plutôt utiliser $\eta = 3\,600$ secondes. La raison qui nous a poussés à effectuer ce changement de procédure est que les valeurs que prennent π_k et π_j influencent directement le nombre d'itérations qu'il est possible d'effectuer pendant un temps donné. Ainsi, il est nécessaire de déterminer quelles tailles d'échantillons permettent d'obtenir le meilleur compromis entre le nombre d'itérations effectuées et la taille du voisinage disponible à l'exploration à chaque itération. Nous avons donc entrepris une recherche exhaustive en faisant varier successivement chacun des deux paramètres dans l'intervalle [0,1; 1,0]. La moyenne des résultats ainsi obtenus pour chaque instance et pour chaque combinaison (π_k, π_j) nous permet de déterminer que l'utilisation d'ensembles formés de 80% des objets et de 10% des serveurs-mandataires à chaque itération fournit en moyenne les meilleures solutions.

5.3.5 Paramètres d'intensification φ, ν et μ

Les derniers paramètres à déterminer sont ceux reliés au processus d'intensification de la recherche. Nous avons tout d'abord tenté de trouver quand débuter ce processus, c'est-à-dire, quelle valeur donner à φ. Pour déterminer la valeur à lui assigner, nous avons étudié l'évolution des solutions des

différentes instances en fonction du temps. Ainsi, selon nos observations, la majorité des solutions trouvées après 1 800 secondes, soient dans le cas de sept des 10 instances, se situent en deçà de 1% de la solution finale qui est obtenue après 3 600 secondes. L'écart maximal rencontré est quant à lui égal à 2,1%. Après avoir étudié ces résultats, nous en sommes venus à la conclusion que φ serait fixé à 0,5. La recherche n'inclut donc aucune intensification pendant les premières 30 minutes.

La détermination des valeurs de ν et μ a quant à elle été effectuée d'une façon similaire à l'analyse des paramètres π_k et π_j. Ainsi, en gardant fixe la valeur de la durée μ, nous avons résolu les différentes instances en faisant varier ν dans l'intervalle [5; 300]. Nous avons ensuite repris le même procédé en faisant varier μ dans l'intervalle [5; 100]. La moyenne des résultats obtenus pour chaque combinaison (ν, μ) nous permet de remarquer qu'une intensification de 5 itérations à toutes les 200 itérations à partir de la $30^{\text{ième}}$ minute de la recherche offre les meilleures performances en général. Il est cependant à noter que l'amélioration apportée par le processus d'intensification est très marginale, les résultats obtenus avec la version définitive de notre heuristique étant en moyenne inférieurs de 0,02% à ceux obtenus lorsqu'aucune intensification n'est faite. Cela s'explique par le fait que l'avantage gagné en étudiant la totalité du voisinage pendant certaines itérations est contrebalancé par la diminution du nombre total d'itérations effectuées lors de la recherche.

Les résultats obtenus lors des différentes expérimentations faites sur les instances définies précédemment sont résumés dans le tableau 4. L'algorithme étant ajusté adéquatement, nous avons résolu les quelques instances proposées par Bektaş et al. (2006a,b).

TAB. 4 – Valeurs assignées aux paramètres

Paramètres	γ	δ	π_k	π_j	φ	ν	μ	θ
Valeurs	0,35	5,5	0,8	0,1	0,5	200	5	$\sqrt{\pi_j \lvert J\rvert + \pi_k \lvert K\rvert}$

5.4 Application sur les instances de la littérature

Nous avons exécuté notre programme à 20 reprises et ce, pour chacune des instances à résoudre qui sont représentées dans le tableau 5. Nous pouvons remarquer qu'il est possible de diviser l'ensemble de ces instances en trois grandes classes distinctes. En effet, la taille de l'ensemble des serveurs-mandataires varie dans l'intervalle [20; 60] par sauts de 10 alors que le nombre d'objets est fixé successivement à 60, 80 et 100. Cette situation est intéressante puisqu'elle nous permet d'évaluer plus facilement le comportement de notre heuristique lorsqu'un de ces deux paramètres est modifié.

TAB. 5 – Caractéristiques des instances de la littérature

Instance	$\lvert J\rvert$	$\lvert I\rvert$	$\lvert K\rvert$	Instance	$\lvert J\rvert$	$\lvert I\rvert$	$\lvert K\rvert$	Instance	$\lvert J\rvert$	$\lvert I\rvert$	$\lvert K\rvert$
1	20	200	60	6	20	200	80	11	20	200	100
2	30	200	60	7	30	200	80	12	30	200	100
3	40	200	60	8	40	200	80	13	40	200	100
4	50	200	60	9	50	200	80	14	50	200	100
5	60	200	60	10	60	200	80	15	60	200	100

Les résultats obtenus sont résumés dans le tableau 6. Les meilleures solutions trouvées par Bektaş

et al., que ce soit à l'aide de méthodes exactes tronquées ou en utilisant une heuristique par recuit simulé, sont retranscrites dans la deuxième colonne. Les valeurs de $c(s^*)$ correspondent aux meilleurs résultats auxquels nous sommes parvenus pour chacune des 15 instances à résoudre. Notons qu'un terme en gras signifie que la valeur trouvée est la meilleure connue. De leur côté, $c(s_{moy})$ et $c(s_{max})$ représentent respectivement la moyenne des résultats ainsi que la valeur la plus élevée obtenues lors des 20 exécutions de notre algorithme. Enfin, les colonnes notées % donnent le pourcentage d'amélioration obtenu avec nos différentes solutions par rapport à celles qu'ont trouvées Bektaş et al. dans leurs recherches. Une valeur négative signifie une détérioration dans la qualité de notre résultat.

TAB. 6 – Résultats obtenus sur 20 exécutions de notre algorithme pour les instances de la littérature

Instance	Bektaş et al.	$c(s^*)$	%	$c(s_{moy})$	%	$c(s_{max})$	%
1	27 207,29	**25 780,62**	5,24	25 885,11	4,86	26 044,27	4,27
2	26 691,37	**24 067,43**	9,83	24 167,10	9,46	24 305,27	8,94
3	27 419,94	**23 946,39**	12,67	24 103,26	12,10	24 219,62	11,67
4	26 588,05	**23 213,08**	12,69	23 327,64	12,26	23 457,85	11,77
5	26 052,30	**22 910,47**	12,06	22 986,37	11,77	23 069,76	11,45
6	31 413,67	**28 127,19**	10,46	28 271,77	10,00	28 464,67	9,39
7	27 906,90	**26 364,39**	5,53	26 452,53	5,21	26 554,92	4,84
8	29 556,16	**26 163,57**	11,48	26 275,63	11,10	26 422,82	10,60
9	30 188,04	**25 462,78**	15,65	25 522,69	15,45	25 613,48	15,15
10	29 323,40	**24 951,84**	14,91	25 026,65	14,65	25 153,59	14,22
11	32 618,58	33 294,13	−2,07	33 452,12	−2,56	33 634,52	−3,11
12	31 450,86	**31 124,88**	1,04	31 325,38	0,40	31 525,45	−0,24
13	30 322,23	30 952,55	−2,08	31 129,68	−2,66	31 329,45	−3,32
14	32 376,74	**30 173,48**	6,81	30 322,43	6,35	30 508,39	5,77
15	32 702,77	**29 707,41**	9,16	29 875,60	8,65	30 086,58	8,00

Nous pouvons donc remarquer que notre heuristique réussit relativement bien, améliorant les solutions des auteurs précédents dans 13 des 15 instances disponibles, et ce, pour les meilleures valeurs trouvées et pour la moyenne. Même lorsque les moins bonnes valeurs sont utilisées, seulement une instance de plus n'est pas améliorée par notre méthode. La faible différence rencontrée entre $c(s^*)$, $c(s_{moy})$ et $c(s_{max})$ pour chaque instance s'explique par le fait que notre algorithme est relativement robuste, c'est à dire qu'il offre de bonnes performances pour une large gamme d'instances ayant des caractéristiques hétérogènes. Ainsi, l'amélioration moyenne rencontrée pour $c(s^*)$ et $c(s_{max})$ est respectivement de 8,23% et de 7,29%, une différence de seulement 0,93%. La méthode de recherche avec tabous que nous avons développée, bien que n'offrant pas toujours la meilleure solution possible, permet tout de même d'obtenir un résultat acceptable à chaque tentative pour les instances à résoudre.

Une autre donnée nous permettant de constater la constance des solutions trouvées est le coefficient de variation CV (écart type/moyenne) obtenu pour l'ensemble des 20 essais de chaque instance. En effet, les faibles valeurs que nous retrouvons dans la colonne CV du tableau 7 signifient que les résultats obtenus sont très peu dispersés autour de la moyenne. Ainsi, même si les résultats obtenus varient à chaque essai, nous pouvons être confiants qu'une solution donnée sera semblable aux autres obtenues précédemment en matière de qualité.

TAB. 7 – Autres caractéristiques des solutions obtenues

Instance	CV	$\bar{\kappa}$	κ^*	ψ^*
1	0,0028	60 197	60 071	3 587,47
2	0,0028	44 768	44 691	3 592,42
3	0,0031	34 385	34 249	3 586,93
4	0,0028	26 213	23 370	3 157,56
5	0,0020	20 399	20 315	3 586,66
6	0,0032	45 501	44 701	3 532,35
7	0,0021	22 758	22 152	3 488,21
8	0,0026	21 490	20 933	3 500,08
9	0,0016	15 135	15 066	3 581,30
10	0,0020	12 540	12 511	3 593,49
11	0,0033	32 270	31 170	3 464,02
12	0,0032	20 153	20 107	3 592,94
13	0,0034	14 819	14 762	3 582,76
14	0,0030	11 399	11 277	3 560,12
15	0,0031	9 326	9 141	3 518,16

Les autres colonnes du tableau 7 permettent de clarifier quelques aspects du comportement de notre heuristique. En effet, pour le meilleur essai de chaque instance, la colonne $\bar{\kappa}$ représente le nombre total d'itérations effectuées pendant la recherche alors que la colonne κ^* représente l'itération à laquelle la meilleure solution a été trouvée. Enfin, le temps en secondes auquel l'heuristique est arrivée à cette itération est noté ψ^*. Le temps total de traitement n'est pas reproduit dans le tableau puisqu'il a été fixé à 3 600 secondes et qu'il ne change pas d'une instance à l'autre. Nous remarquons donc que le nombre d'itérations effectuées dans le temps alloué est généralement faible et qu'il diminue à mesure que la taille des instances augmente. De plus, à l'exception de la 4$^{\text{ième}}$ instance, la meilleure solution est trouvée vers la toute fin de la recherche. Cela signifie donc que l'heuristique utilise tout le temps disponible et que des résultats encore plus intéressants pourraient être obtenus en allongeant la durée du travail effectué.

Nous pouvons cependant nous interroger sur la pertinence d'augmenter le temps de traitement pour accéder à des solutions d'une plus grande qualité. En effet, si nous reprenons le même exercice effectué plus tôt avec les instances tests, nous remarquons qu'aucune amélioration supérieure à 0,9% n'est obtenue dans les 20 dernières minutes de la recherche. Même après 35 minutes de travail, seulement une des instances donnait lieu à un résultat se situant à plus de 1% de la solution finale. Enfin, la valeur médiane de l'amélioration à la 30$^{\text{ième}}$ minute de travail démontre que la moitié des instances se trouvent au plus à 0,9% de la solution finale à ce temps précis. Il semble donc peu certain que l'accroissement de la qualité des résultats soit assez significatif pour augmenter de façon importante le temps alloué à la recherche. Les instances de plus grandes tailles seraient cependant les plus susceptibles d'en bénéficier vu le petit nombre d'itérations que notre heuristique peut exécuter à l'intérieur d'une heure.

6 Conclusion

Nous avons développé une méthode de recherche avec tabous permettant de résoudre le problème mixte de reproduction des objets et d'affectation des clients aux serveurs-mandataires. Partants des recherches effectuées par Bektaş et al., nous avons pu tester la performance de notre algorithme sur différentes instances que ces derniers ont mises au point. La méthode de recherche avec tabous que nous avons développée pour résoudre le problème de conception de RDCÉ offre indéniablement des résultats intéressants. Même la taille des différents problèmes à résoudre a peu d'impact sur la qualité des solutions obtenues puisque notre heuristique est relativement robuste.

7 Remerciements

Ce travail a bénéficié d'un appui financier du Conseil de recherche en sciences naturelles et en génie du Canada (subventions 227837-04 et 39682-05).

Références

Akamai Technologies, http ://www.akamai.com.

Savvis Inc., http ://www.savvis.net.

Acharya, S. et Smith, B. : 1998, An experiment to characterize videos stored on the web, dans *Proceedings of the ACM/SPIE Multimedia Computing and Networking*.

Almeida, J., Eager, D., Vernon, M., et Wright, S. : 2004, Minimizing delivery cost in scalable streaming content distribution systems, *IEEE Transactions on Multimedia* **6**, pp 356–365.

Bektaş, T., Cordeau, J.-F., Erkut, E., et Laporte, G. : 2006a, Exact algorithms for the joint object placement and request routing problem in content distribution networks, *Computers & Operations Research*, À paraître.

Bektaş, T., Cordeau, J.-F., Erkut, E., et Laporte, G. : 2006b, A two-level simulated annealing algorithm for efficient dissemination of electronic content, *Journal of the Operational Research Society*, À paraître.

Bektaş, T., Oğuz, O., et Ouveysi, I. : 2007, Designing cost-effective content distribution networks, *Computers & Operations Research*, À paraître.

Breslau, L., Cao, P., Fan, L., Phillips, G., et Shenker, S. : 1999, Web caching and Zipf-like distributions : Evidence and implications, dans *Proceedings of IEEE INFOCOM'99*, Vol. 1, pp 126–134, New York.

Chari, K. : 1996, Resource allocation and capacity assignment in distributed systems, *Computers & Operations Research* **23**, pp 1025–1041.

Cidon, I., Kutten, S., et Soffer, R. : 2002, Optimal allocation of electronic content, *Computer Networks* **40**, pp 205–218.

Cordeau, J.-F., Gendreau, M., et Laporte, G. : 1997, A tabu search heuristic for periodic and multi-depot vehicle routing problems, *Networks* **30**, pp 105–119.

Erçetin, Ö. et Tassiulas, L. : 2003, Market-based resource allocation for content delivery in the internet, *IEEE Transactions on Computers* **52**, pp 1573–1585.

Fisher, M. et Hochbaum, D. : 1980, Database location in computer networks, *Journal of the Association for Computing Machinery* **27**, pp 718–735.

Gavish, B. et Suh, M. : 1992, Configuration of fully replicated distributed database system over wide area networks, *Annals of Operations Research* **36**, pp 167–192.

GT-ITM, Georgia Tech Internetwork Topology Models, *http ://www.cc.gatech.edu/projects/gtitm*.

Hakimi, S. et Schmeichel, E. : 1997, Locating replicas of a database on a network, *Networks* **30**, pp 31–36.

Huang, C. et Abdelzaher, T. : 2005, Bounded-latency content distribution : Feasibility and evaluation, *IEEE Transactions on Computers* **54**, pp 1422–1437.

Johnson, K., Carr, J., Day, M., et Kaashoek, M. : 2001, The measured performance of content distribution networks, *Computer Communications* **24**, pp 202–206.

Kangasharju, J., Robert, J., et Ross, K. : 2002, Object replication strategies in content distribution networks, *Computer Communications* **25**, pp 376–383.

Laoutaris, N., Zissimopoulos, V., et Stavrakakis, I. : 2004, Joint object placement and node dimensioning for internet content distribution, *Information Processing Letters* **89**, pp 273–279.

Laoutaris, N., Zissimopoulos, V., et Stavrakakis, I. : 2005, On the optimization of storage capacity allocation for content distribution, *Computer Networks* **47**, pp 409–428.

Li, B., Golin, M., Italiano, G., et Deng, X. : 1999, On the optimal placement of web proxies in the internet, dans *Proceedings of IEEE INFOCOM'99*, Vol. 3, pp 1282–1290, New York.

Nguyen, T., Safaei, F., Boustead, P., et Chou, C. : 2005, Provisioning overlay distribution networks, *Computer Networks* **49**, pp 103–118.

Pirkul, H. : 1986, An integer programming model for the allocation of databases in a distributed computer system, *European Journal of Operational Research* **26**, pp 401–411.

Qiu, L., Padmanabhan, V., et Voelker, G. : 2001, On the placement of web server replicas, dans *Proceedings of IEEE INFOCOM'01*, Vol. 3, pp 1587–1596.

Radoslavov, P., Govindan, R., et Estrin, D. : 2002, Topology informed internet replica placement, *Computer Communications* **25**, pp 384–392.

Ryoo, J. et Panwar, S. : 2001, File distribution in networks with multimedia storage servers, *Networks* **38**, pp 140–149.

Saroiu, S. : 2004, *Measurement and Analysis of Internet Content Delivery Systems*, Thèse de doctorat, Université de Washington.

Saroiu, S., Gummadi, K., Dunn, R., Gribble, S., et Levy, H. : 2002, An analysis of internet content delivery systems, dans *Proceedings of 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA.

Soriano, P. et Gendreau, M. : 1997, Fondements et applications des méthodes de recherche avec tabous, *RAIRO (Recherche opérationnelle)* **31**, pp 133–159.

Tang, X. et Xu, J. : 2005, Qos-aware replica placement for content distribution, *IEEE Transactions on Parallel and Distributed Systems* **16**, pp 921–932.

Xu, J., Li, B., et Lee, D. : 2002, Placement problems for transparent data replication proxy services, *IEEE Journal on Selected Areas in Communications* **20**, pp 1383–1398.

Xuanping, Z., Weidong, W., Xiaopeng, T., et Yonghu, Z. : 2003, Data replication at web proxies in content distribution network, dans *Lecture Notes in Computer Science*, Vol. 2642, pp 560–569, Springer-Verlag, Berlin.

Yang, M. et Fei, Z. : 2003, A model for replica placement in content distribution networks for multimedia applications, dans *Proceedings of IEEE International Conference on Communications (ICC '03)*, Vol. 1, pp 557–561.

Zipf, G. : 1949, *Human Behavior and the Principle of Least-Effort*, Addison-Wesley, Cambridge, MA.

Zona Research : 1999, *The economic impacts of unacceptable web-site download speeds*, Rapport technique, Zona Research, Inc., Disponible au http ://also.co.uk/e-hosting.html (Accédé le 16 mai 2007).