Stratégies de négociation entre agents dans le domaine du transport

Mourad Sassi, Brahim Chaib-draa

Département d'Informatique, Faculté des Sciences et de Génie,
Université Laval, Ste-Foy, PQ, Canada, G1K 7P4
E-mails : {sassi, chaib}@iad.ift.ulaval.ca

Résumé

Le problème de planification et de négociation dans les systèmes de transport de marchandises par camions est un problème difficile. Dans ce papier, nous proposons une approche multi-agents pour aider à le résoudre. Dans cette approche, les camions-conducteurs considérés comme des agents intelligents, autonomes et rationnels négocient leurs propositions, via leur compagnie, en effectuant des ventes et des achats de commandes, en vue d'atteindre un compromis qui minimise le coût du plan commun. Pour améliorer la performance de notre système multi-agent, nous avons associé à ce "commerce simulé" des heuristiques de recherche (IDA*, tabou et recuit simulé) permettant d'améliorer, entre autres, la qualité et le temps d'obtention de la solution. Les performances obtenues par les algorithmes associés à ces différentes heuristiques sont discutées en détail, et les conclusions quant à l'adéquation de l'utilisation de chaque algorithme sont finalement dégagées.

Mots-Clés : Négociation entre agents, commerce simulé, planification dans le domaine du transport, heuristiques de recherche.

Abstract

It is generally established that negotiation and planning problem in the transportation domain is a complex problem. To contribute to this problem, we propose in this paper a multi-agent approach. In this approach, trucks considered as rational and autonomous "intelligent agents" negotiate their tasks by selling and buying their tasks. This sort of simulated trading allows them to reach a compromise which minimise their global plan. We then associate to the simulated trading some heuristics (IDA*, Tabu and Simulated Annealing) to improve it. Results of negotiation between trucks using this "new" simulated trading are discussed in details.

Key-Words: Negotiation between agents, simulated trading, planning in the transportation domain.

1 Introduction

Le processus de négociation est un processus de la vie de tous les jours que les humains emploient généralement pour promettre, s'engager, traiter ou trouver un compromis avec autrui. Il pourrait sembler que ce processus prend uniquement place dans les sociétés

humaines et que les interactions entre machines (informatiques) pourraient se passer de ce processus fort difficile à comprendre et à mettre en œuvre. C'est actuellement le cas, dans la mesure où les interactions entre les machines ont lieu dans des environnements restreints et que les règles gouvernant ces interactions sont relativement faciles. Or, les machines deviennent de plus en plus autonomes et commencent de plus en plus à prendre des décisions. Un grand nombre de ces décisions sont effectuées de concert avec d'autres machines. Dès lors, il est important de se pencher sur les protocoles de négociation entre machines de manière à rendre efficace ce type de coopération. Une bonne conception de ces protocoles rendrait également les interactions personne(s)-machine(s) plus efficaces, ce qui aurait une influence positive sur les utilisateurs. Pour contribuer à cela, on vise dans ce papier à concevoir des protocoles de négociation dédiés aux systèmes informatiques dans les systèmes de transport.

Généralement, la conception de protocoles pour la négociation entre agents, nécessite l'utilisation de techniques analytiques et de méthodes de modélisation issues du monde de la théorie des jeux, de l'IA distribuée, des systèmes multi-agents, de l'analyse de décision, et de bien d'autres disciplines. Par *agent*, on entend généralement, la conception d'une entité informatique qui soit capable de raisonner et qui soit la plus *autonome* possible. Une telle entité est également capable de communiquer, d'échanger des points de vue, de négocier et de collaborer avec les autres entités de son environnement.

La "négociation" intéresse un grand nombre de chercheurs appartenant à des domaines aussi variés que l'économie et les sciences politiques [16]. Ce mot a été utilisé de diverses façons mais, en général, il fait référence aux processus de communication devant mener les parties à résoudre leurs conflits et/ou à coordonner leurs activités [12, 13, 17, 19]. La communication d'information pourrait servir, dans ce cadre, soit à altérer les buts des autres agents comme dans les travaux de Sycara [23, 22], soit à permettre à des agents de s'échanger des plans partiaux, comme dans les travaux de Durfee [3]. La négociation pourrait également être considérée comme un processus permettant la redistribution des tâches sur un ensemble d'agents intelligents, autonomes et rationnels. C'est sur quoi ont travaillé Zlotkin et Rosenschein [18], qui ont élaboré des protocoles de négociation basés sur des aspects formels, empruntés à la théorie des jeux.

C'est dans le contexte de cette dernière approache que se situe notre travail dans la mesure où il voit la négociation dans le transport, comme un processus permettant la redistribution des tâches sur un ensemble de camions-conducteurs considérés comme des agents intelligents, autonomes et rationnels. Dans notre approche précisément, les camions-conducteurs négocient leurs propositions, via leur compagnie, en effectuant des ventes et des achats de commandes, en vue d'atteindre un compromis qui minimise le coût du plan commun. Ce mécanisme basé sur le "commerce simulé" permet : 1) de minimiser la distance à parcourir lors de la planification, 2) de minimiser le nombre d'agents et, 3) de maximiser l'utilité globale du système.

Pour améliorer la performance gobale de notre système multi-agent, nous avons associé à ce "commerce simulé" des heuristiques de recherche permettant d'améliorer, entre autres, la qualité et le temps d'obtention de la solution. Les performances obtenues par les algorithmes associés à ces différentes heuristiques sont discutées en détail, et les conclusions quant à l'adéquation de l'utilisation de chaque algorithme sont finalement dégagées.

2 Le domaine du transport considéré comme un système multi-agent

Le domaine de transport comporte deux entités physiques distinctes : les *compagnies de transports* et les *camions* (cf. figure 1). Les compagnies de transport disposent d'une flotte de camions et sont amenées à allouer à leur flotte de véhicules, un ensemble de commandes leur arrivant d'une manière asynchrone et dynamique, tout en respectant un ensemble de contraintes. Dans ce domaine, les camions (associés à leur conducteur) peuvent être considérés comme des agents transporteurs, capables d'effectuer des livraisons de commandes dans différentes localités, générer des plans locaux et négocier entre eux afin de minimiser le coût du plan commun. Chaque véhicule reçoit des commandes sous la forme : "transporter a_1 unités de la marchandise g_1 de la localité l_1 à la localité l_2". Après avoir livré toutes les commandes, les agents doivent retourner au point de départ (c'est à dire au dépôt de la compagnie de transport).

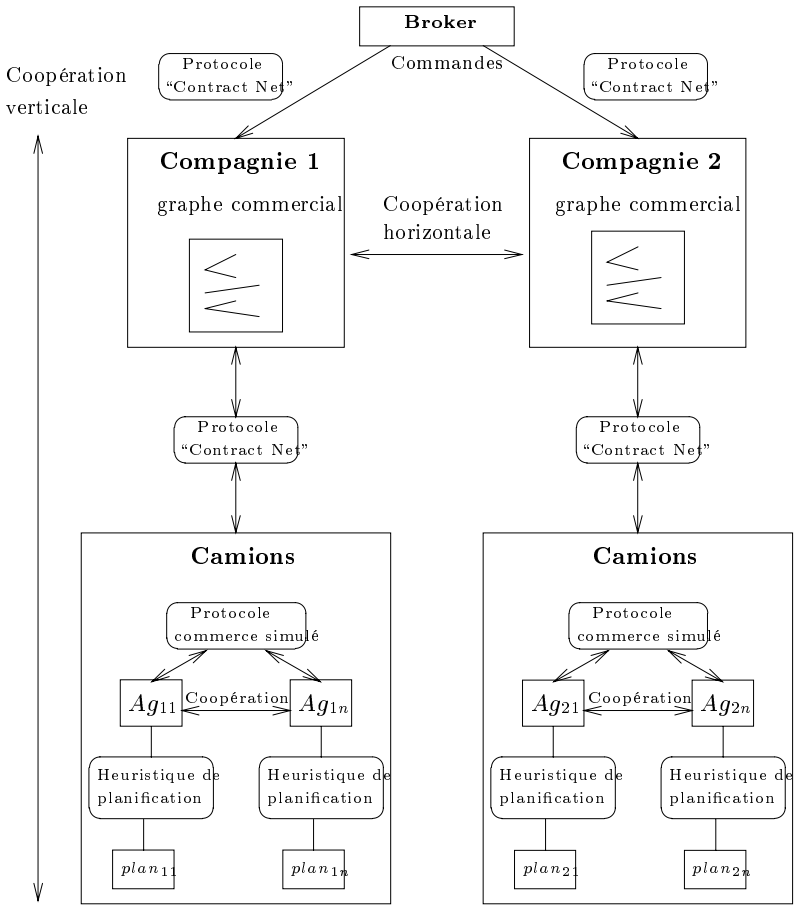

Figure 1: Organisation hiérarchique du domaine de transport

2.1 Négociation basée sur les contrats (le "Contract Net")

Dans le domaine du transport (cf. figure 1), les commandes du *broker* doivent d'abord être allouées aux compagnies. Ces dernières reçoivent des commandes à livrer et construisent leur appel d'offre afin de l'annoncer aux agents de transport. C'est ainsi qu'on a un processus de *distribution de tâches*. La négociation entre la compagnie et ses camions se fait à l'aide du protocole "Contract Net". Toutefois, si les commandes dépassent les capacités d'une compagnie ou d'un véhicule, le "contract-net" doit être combiné à un processus de *décomposition de tâches* : c'est le "Contract Net étendu".

2.1.1 Le protocole "Contract Net"

Le Protocole "Contract Net" [20] est un mécanisme de négociation entre deux types d'agents : *contractant* et *gestionnaire*. Le "Contract Net" permet à un gestionnaire, suite à quelques échanges avec un groupe d'agents, de retenir les services d'un agent appelé contractant pour l'exécution d'une tâche (contrat). Ce protocole est qualifié de type "sélection mutuelle" puisque, pour "signer un contrat", l'agent choisi doit s'engager envers le gestionnaire pour l'exécution de la tâche et le gestionnaire ne sélectionne que l'agent ayant fourni la proposition la plus avantageuse. La version originale du protocole comporte trois étapes principales : l'appel d'offre, la soumission de proposition et l'attribution de contrat.

L'algorithme 2.1 présente le protocole "Contract Net". Sa description détaillée en sept étapes, regroupe les traitements du gestionnaire et des contractants, ainsi que les échanges entre eux. Pour appliquer ce protocole, le concepteur doit détailler le contenu des messages échangés, le temps d'expiration ainsi que les fonctions "évalue-annonce" et "évalue-soumission".

2.1.2 Le "Contract Net" étendu

Le modèle "Contract Net" [5] étendu est basé sur une décomposition décentralisée de tâches. L'agent gestionnaire fractionne une tâche en plusieurs sous-tâches et les annonce à un agent ou à un groupe d'agents soumissionnaires. Mais contrairement au "Contract Net" original où une tâche ne peut être qu'*accordée* ou *rejetée*, dans le "Contract Net" étendu, une tâche peut être *accordée temporairement*, *rejetée temporairement*, accordée définitivement et *rejetée définitivement* par les agents.

Dans le "Contract Net" étendu, L'agent gestionnaire annonce une commande un groupe d'agents. Il sectionne le meilleur agent soumissionnaire à qui il envoie un accord temporaire. Tous les autres agents reçoivent un rejet temporaire. Si la meilleure soumission ne couvre pas la totalité de la commande, la partie restante de la commande est réannoncée par le gestionnaire jusqu'à couvrir sa totalité. Au terme de ce processus les accords et des rejets temporaires sont transformés par des accords et des rejets définitifs. Quand un agent reçoit un accord temporaire, il crée une copie de son ancien plan, ainsi il sera en mesure de le récuperer en cas de rejet définitif.

Étant donné une tâche, un gestionnaire, un groupe de $(n-1)$ soumissionnaires :

1. Le gestionnaire envoie un message de type "annonce-tâche" à un groupe d'agents (ou fait un
   "broadcast").
2. Chaque agent évalue la tâche annoncée à l'aide d'une fonction locale "évalue-annonce".
3. L'évaluation précédente permet à certains agents de soumettre une proposition à l'aide d'une "soumission-tâche" au gestionnaire.
4. Si une proposition est jugée satisfaisante (à l'aide de la fonction "évalue-soumission"), alors
   le gestionnaire envoie un message de type "acceptation" à celui dont la proposition
   est retenue. Il envoie également un message de type "refus" aux autres agents dont les
   propositions n'ont pas été retenues.
5. Le gestionnaire peut mettre fin à la période d'acceptation de proposition si le temps d'expiration
   est dépassé.
6. Si aucune proposition n'a été retenue, alors le gestionnaire fait parvenir à tous les agents
   non retenus un message de type "refus" pour indiquer le rejet de chacune des propositions.
   Il peut alors se retirer de la négociation, retenir la proposition la plus acceptable, redémarrer un
   nouvel appel d'offre (nouveau "annonce-tâche") ou prolonger le temps d'expiration de la période
   d'acceptation de proposition.
7. L'agent ayant obtenu le contrat, remet un rapport d'exécution lorsque la tâche est complétée.

Algorithme 2.1: Algorithme du "Contract Net"

2.2 Le commerce simulé

La négociation dans le modèle du commerce simulé [1] est aussi basée sur le protocole "Contract Net". Les agents cherchent à vendre et à acheter des tâches (ou commandes), en vue d'améliorer leur plan local dont l'union constitue un plan commun. L'idée principale ici est d'appliquer les mécanismes du commerce habituel dans le but d'optimiser le partionnement des commandes. Les échanges de commandes définissent un graphe bi-partie à niveaux, appelé aussi *graphe commercial*. Les nœuds correspondent à une insertion (achat) d'une commande dans le plan ou à une suppression (vente). Les arêtes représentent les échanges possibles, et possèdent un poids positif ou négatif correspondant au gain obtenu. Chaque relation commerciale entre agents correspond à un nombre d'échanges de commandes. Si la valeur de la relation (la somme de tous les poids des arêtes) est positive, alors le plan commun a été amélioré. Ce protocole favorise l'échange de commandes entre agents et permet de réaliser une amélioration globale du plan.

2.2.1 Description de la fonction coût

La procédure principale du commerce simulé effectue des opérations d'insertions et de suppressions de tâches ainsi qu'une recherche de relations d'appariement dépendemment d'une fonction coût. Nous allons maintenant préciser cette fonction de coût.

Soit un ensemble de commandes $V = \{1, \ldots, n\}$. Dans ce cas, un *plan* est un sous-ensemble $P \subseteq V$. Le *coût* d'un plan $c(P)$ est donné par le poids de la fonction $c : 2^V \rightarrow \mathbb{R} \cup \{\infty\}$. Un *plan commun* $\mathcal{P} = (P_1, \ldots, P_t)$ est une partition de V. Le coût du plan commun est donné par :

$$c(\mathcal{P}) = \sum_{i=1}^{t} c(P_i).$$

Le coût de suppression d'une tâche t_i d'un plan P est donné par :

$$c^-(P, t_i) = c(P) - c(P - \{t_i\}).$$

Le coût d'insertion d'une tâche t_i dans un plan P est donné par :

$$c^+(P, t_i) = c(P + \{t_i\}) - c(P).$$

Le poids d'une arête est défini comme étant l'amélioration de la fonction de coût due à l'échange de tâches :

$$c^-(P_1, t_i) - c^+(P_2, t_i).$$

Dans le graphe, les nœuds correspondent à une insertion ou à une suppression d'une tâche dans un plan. Le poids des arêtes représente le gain obtenu. Si le gain est positif, le plan commun est généralement amélioré.

2.2.2 Description de l'algorithme du commerce simulé

L'algorithme 2.2 présente la description générale de l'algorithme du commerce simulé. À noter que la phase d'appariement consiste à trouver le meilleur graphe commercial, s'il existe. Cette phase est présentée dans la section 2.2.4. La présentation de l'algorithme 2.2 est donnée dans le cadre général; sa version détaillée est décrite dans l'algorithme

2.3. Dans cet algorithme, le critère d'arrêt peut être la convergence, un nombre fixe d'itérations, une liste vide de tâches en vente ou un critère spécifique au problème.

```
répéter
   phase de vente et d'achat de tâches;
   phase de recherche de relations d'appariement;
   si appariement trouvé alors
      mise à jour du plan commun;
   fin-si
jusqu'à coût du plan commun soit minimal
```

Algorithme 2.2: Description générale de l'algorithme du commerce simulé

```
commerce_simule
/* Soit {a_1, ..., a_n} un ensemble d'agents */
/* Soit P_i = (t_1, ..., t_n) le plan de l'agent a_i */
/* Soit S = la liste des tâches en vente */
tant-que critère d'arrêt non satisfait faire
   pour k = 1 jusqu'à n faire
      choisir vente ou achat;
      si vente alors
         choisir une tâche i ∈ P_k tel que c^-(P_k, t_i) est maximum;
         P_k = P_k - {t_i}; /* supprime la tâche t_i du plan P_k */
         S = S + {t_i}; /* ajoute la tâche t_i dans la liste des tâches vendues */
         s(i) = c^-(P_k, t_i); /* le prix de vente de la tâche t_i dans P_k */
         ajouter un nœud de suppression au graphe;
      sinon
         si une tâche t_i ∈ S tel que c^-(P_k, t_i) - c^+(P_k, t_i) est maximum alors
            P_k = P_k + {t_i}; /* insère la tâche t_i dans le plan P_k */
            ajouter un nœud d'insertion au graphe avec les arêtes appropriées;
            replanifier qP_k tel que c(P_k) est minimum;
         fin-si
      fin-si
   fin-pour
fin-tant-que
```

Algorithme 2.3: Procédure d'achat/vente et de recherche de relations d'appariement

2.2.3 Phase d'achat et de vente de tâches

Durant cette phase, on a le choix de faire une vente, un achat ou rien du tout. Dans la phase de vente de chaque plan, les tâches qui causent de grandes pertes sont supprimées

des plans des agents. Lors d'un achat, les tâches qui procurent de grands gains sont préférées et sont insérées dans les plans des agents.

Durant la phase d'achat et de vente de tâches, les plans P_k sont sauvegardés. Un graphe commercial est construit à l'aide des classes nœuds de ventes V_v et nœuds d'achats V_a. Les nœuds de ventes et d'achats sont adjacents s'ils réfèrent à la même tâche. Le poids d'une arête représente l'amélioration de la fonction coût provoquée par l'inter-échange de tâches entre les agents.

- Le choix d'une tâche t_i à vendre est déterminé à l'aide de son coût dans le plan P_k de l'agent A_k. C'est en fait la tâche qui cause le coût maximal dans P_k.
- Le choix d'une tâche à acheter est déterminé à l'aide de son coût d'insertion dans le plan P_k de l'agent A_k. C'est la tâche qui procure le gain maximal car son coût d'insertion dans le plan est minimal.
- Pour chaque décision de vente ou d'achat, un nœud est ajouté au graphe. Ce nœud contient l'information suivante : identification du plan et identification de l'agent. Une tâche à vendre doit être ajoutée dans une liste S des tâches en vente.

  $$S = S + \{t_i\},\ s(i) = c^-(P, t_i).$$

  Les économies réalisées en supprimant la tâche t_i représentent le poids de l'arête e. Le poids est donné par :

  $$s(i) - c^+(P, t_i).$$

- Soit k un agent et t_i une tâche, P_k est défini de la manière suivante :

$$P_k = \begin{cases} P_k - \{t_i\} & \text{l'agent } k \text{ supprime la tâche } t_i \text{ de son plan } P. \\ P_k + \{t_i\} & \text{l'agent } k \text{ insère la tâche } t_i \text{ à son plan } P. \end{cases}$$

Dans l'algorithme 2.3, les plans P_k sont sauvegardés et à l'aide d'algorithmes de recherche comme le recuit simulé ou la métaheuristique de recherche tabou, on cherche à minimiser la fonction $c(P_k)$ pour chaque agent.

2.2.4 Phase de recherche de relations d'appariement

Durant la phase d'appariement, un ensemble de paires de nœuds doit être recherché. Chaque paire doit contenir un nœud d'achat et son nœud de vente approprié, avec la condition que chaque nœud ne doit être inclus que dans une paire au plus. C'est-à-dire que , si plusieurs arêtes pointent vers un même nœud appartenant à la classe V_a, on ne doit retenir que celui qui donne un gain maximum et éliminer les autres du graphe commercial. La *valeur* de l'appariement M est défini de la manière suivante :

$$w(M) = \sum_{e \in M} w(e).$$

2.3 Résultats obtenus à date avec le commerce simulé

L'équipe multi-agent du centre de recherche allemand en IA, le fameux DFKI, a développé une plateforme de simulation et de négociation entre agents, qu'elle a appliqué aux compagnies de transport [4]. C'est ainsi que les compagnies sont simulées comme des entités autonomes (i.e., agents) devant gérer des ordres de commandes leur arrivant d'une manière asynchrone et dynamique. Ces compagnies mettent à la disposition de leurs camions des commandes qu'ils doivent négocier avec leur compagnie respective suivant le protocole "contract-net". La coopération entre camions est en revanche assurée par le commerce simulé.

3 Stratégie associant Contract Net, recuit simulé et heuristiques de recherche

Nous présentons, dans cette section, une nouvelle approche associant commerce simulé et heuristiques de recherche, ce qui devrait nous permettre d'améliorer le temps de recherche d'une solution et la qualité de celle-ci. Les heuristiques de recherche sont : le recuit simulé, les tabous et IDA*. Suite à cela, nous présenterons une approche de planification de commandes basée sur les algorithmes génétiques (AG), de manière à situer l'association commerce simulé-heuristique par rapport à ces algorithmes généralement considérés comme très performants.

3.1 Planification des commandes associant Contract Net, recuite simulé et heuristiques de recherche

Dans cette nouvelle forme de planification, on combine le Contract Net et le commerce simulé de la manière suivante :

1. *Annonce de tâche :* Le gestionnaire lance un appel d'offre pour une commande. Celle-ci pourrait être soit une nouvelle commande, soit une commande qu'un des agents veut vendre.
2. *Évaluation de l'annonce :* chacun des éventuels contractants détermine le coût d'insertion de la nouvelle commande dans son plan local et ce à l'aide de la fonction "évalue-annonce". Cette fonction utilise l'une des trois heuristiques suivantes : l'algorithme de recherche arborescent A*, le recuit simulé ou l'heuristique de recherche avec tabous.
3. *Soumission du message :* les agents intéressés soumettent une proposition indiquant leur intention d'inclure la nouvelle commande dans leur plan ainsi que le coût d'insertion de la commande.
4. *Évaluation de soumissions Acceptation/Refus :* si l'une des propositions d'insertion d'une commande est jugée satisfaisante (à l'aide de la fonction "évalue-soumission"), alors le gestionnaire envoie un message de type "acceptation" au contractant dont la proposition est retenue et un message de type "refus" aux autres proposants.

3.1.1 Les heuristiques de recherche

A. Algorithme IDA*

Le problème de planification des commandes est un problème de recherche dans un espace d'états pour lequel l'un des algorithmes les plus utilisés, est l'algorithme de recherche avec la stratégie de sélection du *meilleur d'abord* [14]. Il consiste à construire un arbre dont les nœuds sont les états, et à privilégier un déplacement dans cet arbre qui nous assure de trouver un *but* ou *nœud final*.

Le problème de la tournée de véhicules peut être mis sous la forme d'un problème de recherche de solutions dans un espace d'états. Cet espace est un arbre dont les nœuds sont des états intermédiaires (ou des solutions partiellement développées). Ces derniers représentent les localités à visiter. À chaque localité correspond une commande à livrer. La racine est le dépôt contenant toutes les commandes. Les branches de l'arbre représentent toutes les solutions finales possibles. Cet arbre est développé à partir des spécifications suivantes :

- *la racine :* c'est un état initial, pour lequel aucune localité n'est encore visitée, $P = \emptyset$ (plan de l'agent) et $Q = \{V_i\}$ (ensemble des commandes à effectuer ou des localités à visiter);
- *un nœud :* c'est un état i désignant une étape de l'algorithme, où une commande a déjà été assignée (ou une localité visitée soit V_i). Cet état est donc décrit par le paramètre P tel que $P \leftarrow P + \{V_i\}$, où $P \subseteq Q$;
- *génération des successeurs :* c'est la procédure de développement d'un nœud, qui consiste à sélectionner une localité parmi celles non encore visitées, puis à créer un nœud fils pour chaque placement possible.

Dans notre problème du transport, nous avons utilisé IDA* [11] (une variante de A*) pour les raisons suivantes : IDA* ne gère pas (comme le fait A*) la liste des nœuds non développés, d'où un gain en espace mémoire. Cet algorithme fonctionne comme suit : à chaque itération de l'algorithme, on fait une recherche en profondeur d'abord, avec élagage des branches de l'arbre lorsque le coût d'un nœud dépasse un seuil. Ce seuil est pris comme étant le minimum des coûts des branches obtenus à chaque itération. Le coût d'un nœud est calculé en utilisant la fonction monotone croissante $f() = g() + h()$. Les fonctions g et h peuvent être formulées comme suit. Soit un nœud quelconque n désignant une localité à visiter. $g(n)$ est le coût du trajet défini comme étant la distance allant de la racine à n et $h(n)$ le coût estimé du reste du parcours, c'est la distance allant de la localité n jusqu'à la feuille de l'arbre, en passant par les localités intermédiaires de la branche. Finalement, $f(n) = g(n) + h(n)$ représente le coût de la solution passant par n.

La description de IDA* adaptée à notre problème est donné dans l'Algorithme 3.1.

B. Le recuit simulé

Le recuit simulé [10] est un algorithme basé sur une technique connue en thermodynamique. Il fait partie de la famille des algorithmes utilisant la stratégie de l'*escalade*,

```
IDA*;
/* Initialisation */
Soit n_0 la 1^ère feuille rencontrée lors de la visite de l'arbre en "profondeur d'abord";
seuil ← f(n_0);
répéter
  Développer l'arbre en "profondeur d'abord";
  si n est une feuille alors
    si f(n) < seuil alors
      seuil ← f(n)
    fin-si
  sinon
    Élaguer la branche de l'arbre à partir de n;
  fin-si
jusqu'à ce qu'il ne reste qu'une branche dans l'arbre
Retourner cette branche;
```

Algorithme 3.1: Algorithme IDA*

avec une minimisation de la fonction objectif au lieu d'une maximisation. Une description pseudo-algorithmique du recuit simulé, tel qu'utilisé pour notre problème de transport, est élaborée dans l'algorithme 3.2.

La fonction de coût prend en paramètre l'ensemble des commandes à effectuer $Q = \{c_i\}$ et retourne le coût du plan P. Les paramètres T_{init}, k, N_c et N_{it} sont des nombres définis expérimentalement, avec $0 < k < 1$.

Deux remarques s'imposent ici: 1) Pour la solution initiale, l'heuristique du plus proche voisin est utilisée : on commence par un nœud au hasard et on visite son plus proche voisin. 2) Le recuit simulé améliore la valeur de la fonction objective, et ce même si la longueur du plan semble augmenter, la perturbation est quand même acceptée avec une certaine probabilité.

C. La méthode de recherche avec tabous

La recherche tabou [7, 8] exploite la notion de mémoire. Elle consiste à garder la trace du cheminement passé effectué dans le processus de recherche dans une ou des mémoires et à se servir de cette information afin d'en orienter le déroulement futur. Ceci permet donc de diriger son exploration vers des régions non encore visitées. La méthode la plus fréquemment utilisée consiste à définir les tabous en fonction des "transformations" qui permettent de se déplacer d'une solution à une autre. On garde alors une liste des T dernières transformations effectuées ou de certaines de leurs caractéristiques, et on interdit à la procédure de les inverser. La plupart du temps, on note directement la liste des transformations inverses qu'on veut proscrire.

Dans le domaine du transport, la méthode de recherche tabou est utilisée pour les raisons suivantes : 1) elle permet de rechercher un plan optimal lors de l'élaboration des plans, 2) elle permet d'éviter de rentrer dans des boucles infinies, 3) elle interdit les

Recuit_Simule;
/* Étape d'initialisation */
Soit P_0 un plan initial; /* P_0 configuration initiale */
$E_0 \longleftarrow coût(P_0)$;
$T_0 \longleftarrow T_{init}$; /* Température initiale */
Soit T_{min} la température minimale;
/* Le système est dans la configuration P_i */
/* à la température T_i, et possède l'énergie E_i */
tant-que $T_i \geq T_{min}$ **faire**
  répéter
    Générer aléatoirement une nouvelle configuration P_{i+1} voisine de P_i;
    $E_{i+1} \longleftarrow coût(P_{i+1})$;
    $\Delta E \longleftarrow E_{i+1} - E_i$;
    Soit $p(T_i, \Delta E)$ la probabilité d'accepter la nouvelle configuration P_{i+1};
    $p(T_i, \Delta E) \longleftarrow min(1, e^{-\Delta E/T_i})$;
    Tirer un nombre aléatoire *nb_rand* dans $[0, 1]$;
    si $nb\_rand \leq p(T_i, \Delta E)$ **alors**
      Accepter la nouvelle configuration et retenir P_{i+1} comme solution courante;
    sinon
      Rejeter la nouvelle configuration et retenir P_i comme solution courante;
    fin-si
  jusqu'à N_c configurations soient acceptées **ou** N_{it} itérations soient exécutées;
  /* Diminuer la température d'un facteur k */
  $T_{i+1} \longleftarrow kT_i$;
fin-tant-que

Algorithme 3.2: Algorithme du recuit simulé

solutions qui détériorent le plan commun, 4) elle fixe le nombre d'itérations maximal, 5) elle effectue la recherche de la solution tout en évitant les optimums locaux.

L'application de cette heuristique au domaine du transport a déjà été faite par l'équipe de Gendreau [6]. Dans notre cas, il nous faut considérer les points suivants. Soit un plan initial P, les transformations qu'il est possible d'y apporter par l'entremise de l'heuristique interne définissent un voisinage de P identifié par $N(P)$. Le voisinage $N(P)$ d'une solution P est défini comme l'ensemble des solutions P' dans un domaine $\mathcal{X}$ et pouvant être obtenues suite à l'application d'une transformation locale à P. Le choix d'une nouvelle solution P' est déterminé par l'évaluation d'une fonction de coût.

Cependant, l'évaluation complète du voisinage de la solution courante $N(P)$ s'avère souvent extrêmement coûteuse et ne constitue pas toujours une stratégie d'évaluation raisonnable. Dans ces situations, on a recours à des techniques d'échantillonage aléatoire pour générer un sous-ensemble à partir de $N(P)$ et on restreint le choix de la prochaine solution à celle appartenant à $N(P)$. On peut ainsi réduire considérablement les coûts d'évaluation.

Soit le problème d'optimisation qui consiste à minimiser la fonction $coût(P)$ sur un domaine $\mathcal{X}$ et soit P^* la meilleure solution rencontrée jusqu'à présent et c^* la valeur de

celle-ci, $c^* = coût(P^*)$, $N(P,k)$ est l'ensemble des solutions admissibles à partir de P à l'itération k. L'algorithme 3.3 schématise l'algorithme générique de la recherche tabou.

Ce processus d'exploration rend la RT insensible aux optimums locaux, mais introduit du même coup le risque de cycler dès que l'on sort de l'un des optimums, en y retournant aussitôt. Pour régler ce problème, la RT mémorise alors les dernières informations sur le chemin effectué à travers $\mathcal{X}$ pour interdire certaines transformations de la solution courante qui pourraient ramener la procédure vers des solutions déjà rencontrées. Ces transformations sont alors déclarées "tabous". Ce caractère tabou permet d'orienter l'exploration de $\mathcal{X}$ au fur et à mesure qu'elle se déroule. Cependant, ceci ne doit pas être maintenu en permanence.

La recherche tabou a été appliquée au problème du transport et de tournées de véhicules. Osman [15] a développé plusieurs variantes de la RT et du recuit simulé pour la version du PVT avec contraintes de capacité sur les véhicules, améliorant à l'époque les durées sur les tournées. Dans les recherches d'Osman, la RT s'est avérée supérieure au recuit simulé et a donné de très bonnes solutions.

Tabou;
/ étape d'initialisation */*;
choisir une solution initiale P_0 dans $\mathcal{X}$;
$P^* \leftarrow P_0$;
$c^* \leftarrow coût(P^*)$;
$k \leftarrow 0$;
tant-que critère d'arrêt non satisfait **faire**
  $k \leftarrow k+1$;
  choisir P tel que $coût(P)$ minimum;
  si $coût(P) < c^*$ **alors**
    $P^* \leftarrow P$;
    $c^* \leftarrow coût(P^*)$;
  fin-si
  mettre à jour les tabous;
fin-tant-que

Algorithme 3.3: Algorithme de la recherche tabou

3.2 Planification dans le transport utilisant les algorithmes génétiques

Les algorithmes génétiques (AG) conçus initialement dans un cadre général, constituent actuellement une classe d'algorithmes stochastiques utilisés avec succès dans les problèmes d'optimisation numérique. Ce sont des algorithmes de recherche basés sur la sélection naturelle. Ils combinent le principe de la survie du plus fort avec un échange d'information aléatoire biaisé entre les structures. Les AGs ont été développés par John Holland [9], à l'Université du Michigan.

Un algorithme génétique travaille à modifier une population initiale d'individus. Soit P une population d'individus, $P(0)$ l'ensemble initial des individus, et $P(t)$ la population au temps t. Un algorithme génétique consiste à générer aléatoirement au

temps $t+1$ une population $P(t+1)$, à partir de la précédente. Pour réaliser cette phase de reproduction, des "opérateurs génétiques" sont appliqués sur les individus de $P(t)$. Le choix des individus sur lesquels sont appliqués ces opérateurs est guidé par le principe de la sélection naturelle : *Les individus offrant les meilleurs coûts sont sélectionnés pour générer la prochaine génération d'individus.* Afin de mesurer la qualité des individus obtenus, on définit une fonction d'évaluation de coûts applicable sur ces individus. Les deux plus efficaces opérateurs génétiques, sont "l'hybridation" et la "mutation".

Un AG n'est en fait qu'un algorithme d'optimisation qui permet de résoudre une plus grande variété de problèmes que les techniques d'optimisation traditionnelle. Cependant, il n'exige aucune connaissance préalable du problème pour bien fonctionner. Ceci est un avantage non négligeable sur la plupart des techniques d'optimisation connues.

Une population est constituée d'un ensemble d'individus contenant des solutions valides pour le problème. La sélection, l'hybridation et la mutation se font à des probabilités P_s, P_r et P_m définies par l'utilisateur. Le critère de fin peut être la convergence, un nombre fixe d'itérations ou un critère quelconque spécifique au problème.

Pour implanter ce type d'algorithme, il convient de résoudre quatre problèmes : 1) définir l'encodage et le décodage d'une solution, ainsi que la valeur d'une solution; 2) définir une bonne méthode de sélection des individus (chromosomes); 3) définir de bons opérateurs d'hybridation, 4) définir de bons opérateurs de mutation.

3.2.1 Application au domaine du transport

Dans ce domaine, on crée un mécanisme de recherche coopératif qui introduit les algorithmes génétiques comme mécanisme de négociation. Dans l'algorithme 3.4, l'espace de recherche est résolue par plusieurs agents, chacun d'eux exécute l'algorithme génétique périodiquement produisant ainsi des descendants de manière stochastique. La génération des descendants est basée sur leur valeur d'adaptation définie par une heuristique qui nous donne une estimation de la valeur de son état courant. Ainsi, on préfère le plan d'un agent si la distance globale qu'il aura à parcourir est minimale. Tandis que les plans coûteux seront exterminés. Les résultats expérimentaux montrent que cet algorithme est très efficace pour les problèmes divisibles en sous buts (ex, routage de véhicules), car l'agent de transport n'a aucune connaissance sur les buts des autres agents. En particulier, cet algorithme peut résoudre d'énormes plans de parcours, qui ne peuvent être résolues dans un temps raisonnable par les méthodes de résolution classiques, à moins de donner explicitement la connaissance à tous les agents.

Le critère d'évolution est conditionné par l'abaissement des coûts des individus nouvellement créés. Si au bout d'un certain nombre d'itérations, il n'y a aucune évolution, l'algorithme s'arrête. Un individu créé est accepté si son coût est intéressant, c'est-à-dire si *coût(X)*, le coût du plan X, n'est pas supérieur à ceux de tous les autres plans composant la population. La procédure de génération d'individus $hybridation(P_i, P_j)$ consiste à générer un seul individu, dont chaque composant de sa représentation (code génétique) provient aléatoirement de P_i ou de P_j. L'opérateur de mutation est utilisé de manière aléatoire, et ce, conjointement à l'opérateur d'hybridation dans le but de générer de petites perturbations dans le plan des agents concernés. Ceci permet de surmonter les optimums locaux.

Algo_Genetique;
Générer une population aléatoire d'individus;
tant-que évolution **faire**
  Sélectionner deux individus P_i et P_j aléatoirement;
  $X \longleftarrow$ *hybridation*(P_i, P_j)
  si *coût(X)* intéressant **alors**
    Remplacer l'individu de plus fort coût par X;
  sinon
    Rejeter X;
    Muter certains individus;
  fin-si
fin-tant-que

Algorithme 3.4: Algorithme génétique

a) *L'encodage et le décodage* : Dans le domaine spécifique du transport de marchandises par camions, la population est constituée de plans locaux de commandes. Un plan représente la liste des localités que doit visiter l'agent. Il faut ensuite trouver une méthode permettant de trouver une valeur d'adaptation à partir de l'encodage. La valeur d'adaptation est une indication de la qualité de la solution, qui peut être calculée en fonction de la distance à parcourir ou à partir du nombre de véhicules utilisé.

b) *La sélection* : Durant cette phase, les solutions parentes sont sélectionnées à partir de la population locale. Le processus de sélection est stochastique et biaisé envers les meilleures solutions. La valeur d'adaptation nous donne une indication sur la qualité de la solution. Une solution A est meilleure qu'une solution B si $distance(A) < distance(B)$.

c) *L'hybridation* : Cet opérateur crée un descendant en remplaçant une portion de plan de la solution parent 1 par une portion de plan de la solution parent 2. Cette phase peut laisser certaines localités du plan isolées ou dupliquées. Dans la figure 2, le plan avec les localités noires remplace le plan correspondant dans la solution parent 2. On remarque qu'il y a des localités qui appartiennent à la fois à deux plans, tandis que certaines localités restent isolées. Un opérateur de réparation est appliqué aux descendants dans le but de générer un plan réalisable.

d) *La mutation* : L'opérateur de mutation sélectionne un plan et déplace une localité à visiter à une position aléatoire du plan d'un agent (cf. figure 3).

4 Résultats expérimentaux

Dans le but d'évaluer l'influence des différentes stratégies de négociation dans la recherche d'une solution au problème de planification, un jeu de test extrait de la littérature [2, 21] a été utilisé à cet effet. Ce jeu de test comprend les coordonnés des localités à visiter, les commandes à transférer et la capacité maximale des camions. Le problème consiste à visiter 100 localités et d'y effectuer des livraisons avec un coût de parcours

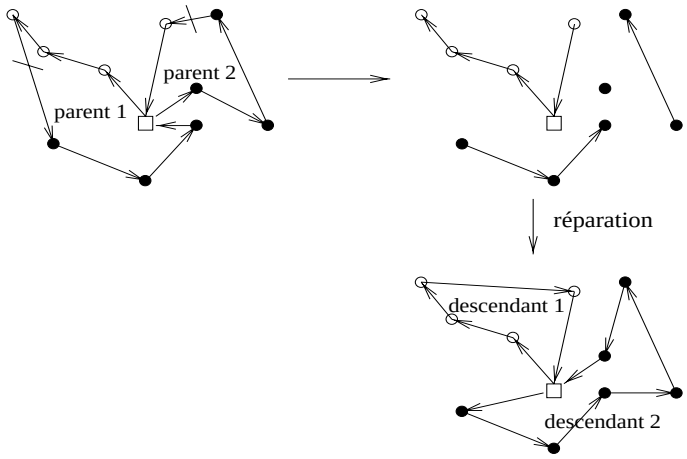

Figure 2: L'opérateur d'hybridation

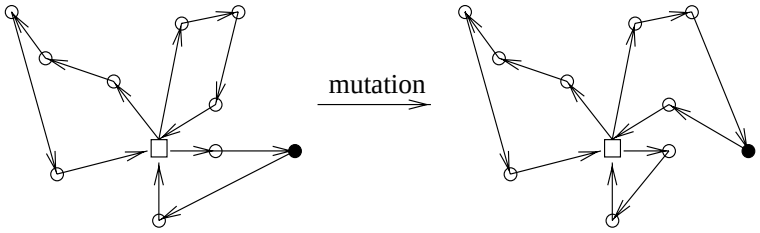

Figure 3: L'opérateur de mutation

minimal. Les commandes ont la structure suivante : "transférer k unités de la localité l_1 à la localité l_2". Les commandes sont présentées aléatoirement aux agents. Ce choix est très important car ceci permet de simuler un système dynamique. Dans un système dynamique, les agents n'ont aucune connaissance sur l'ordre d'arrivée des commandes.

Ce jeu de test impose toutefois certaines restrictions : il n'y a qu'un dépôt à partir duquel les agents peuvent s'approvisionner en marchandises. Chaque exemple contient 100 commandes destinées à 100 localités et aucune localité n'apparaît plus d'une fois. Il n'y a qu'une seule compagnie de transport. Il est également supposé qu'il n'y a qu'un chemin rectiligne séparant deux localités différentes. Toutefois, en dépit de ces restrictions, la solution optimale n'est connue que pour un petit nombre d'exemples.

Les paramètres observés sont la distance parcourue par les camions (premier critère dans l'évaluation des performances), le nombre de camions requis dans la solution (critère important du point de vue économique), le temps mis par chaque algorithme dans la recherche d'une solution, la qualité de celle-ci et enfin, la complexité en terme d'espace mémoire.

4.1 Analyse des résultats

La figure 4 compare, à l'aide du critère coût représenté par la distance à parcourir, les performances obtenues initialement lors de l'application du protocole "Contract Net" (PCN) classique et ultérieurement après l'ajout de la stratégie du commerce simulé

(C.S) et du recuit simulé. L'axe des abscisses représente 21 configurations d'une liste de 100 commandes. Ces configurations sont générées aléatoirement pour simuler un système dynamique. La courbe du haut représente le plan initial des agents; elle est obtenue à l'issue de la phase de négociation à l'aide du protocole "Contract Net" et de l'heuristique du *plus proche voisin*: on commence le plan par une localité et on visite son plus proche voisin. On obtient ainsi, un plan initial réalisable. Après introduction de la stratégie du commerce simulé et du recuit simulé, le coût de la solution initiale se voit considérablement amélioré d'environ 20%. Ceci prouve bien que le commerce simulé est une bonne stratégie dans le traitement des problèmes dynamiques.

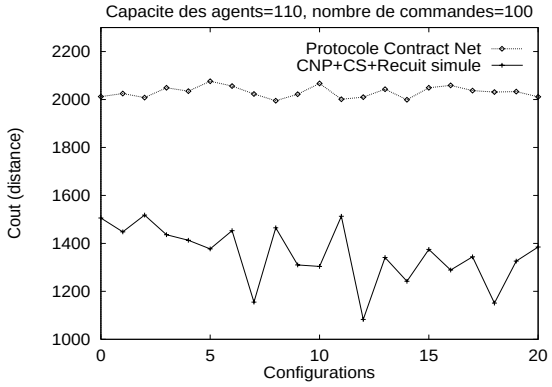

Figure 4: Apport du C.S sur le PCN

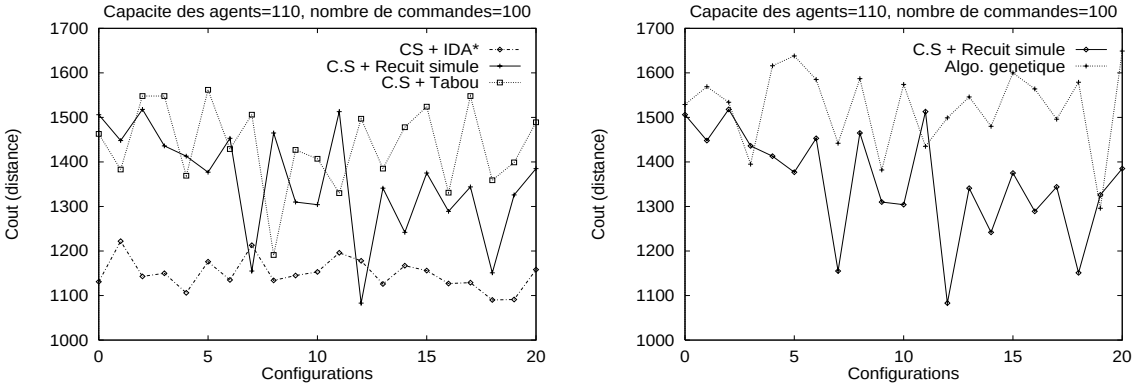

Figure 5: Performance des différentes stratégies

La figure 5-a compare, quant à elle les performances obtenues par le C.S après l'application des différentes heuristiques d'optimisation. Si l'on prend la courbe représentative de IDA* comme référence, cette dernière est en moyenne de 17% à 22% meilleure que ses homologues le recuit simulé et tabou. Il arrive cependant que le recuit simulé trouve la solution optimale dépassant ainsi le résultat de référence fourni par IDA*. Par contre, si l'on se situe par rapport au plan initial (obtenu grâce au "Contract Net" classique), les stratégies tabou, recuit simulé et IDA* améliorent le coût de la solution initiale de respectivement 28%, 32% et 44%.

Les solutions trouvées par l'algorithme IDA* sont voisines de la solution optimale et

sont utilisées comme référence. En général, la solution optimale ne peut être trouvée que si le problème est traité comme fermé, ce qui dans notre cas correspond à la connaissance de *toutes* les informations et ce dès le début de la planification. Une telle contrainte est difficile voire impossible à réaliser dans le mesure où le problème de transport est en général un problème ouvert.

La figure 5-b compare les performances obtenues par la stratégie basée sur les algorithmes génétiques, avec celle du commerce simulé en association avec l'heuristique du recuit simulé. Les algorithmes génétiques nous donnent comparativement au recuit simulé une solution de 11% moins bonne mais par contre, le temps de traitement est le plus bas de tous comme l'indique la figure 7-a. Ceci est intéressant dans la mesure où l'on pourrait utiliser les AG comme première approximation dans la recherche d'un plan initial.

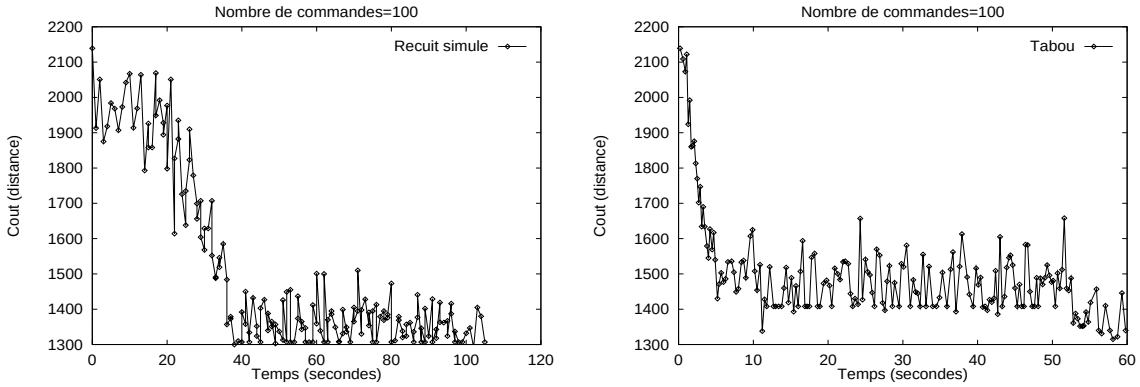

Figure 6: Vitesse de convergence des stratégie en fonction du temps

Les figures 6 montrent l'allure typique de la courbe *coût du plan* en fonction du temps de calcul. Elles représentent l'amélioration du plan commun en fonction du temps total nécessaire à son calcul. Le commerce simulé associé au recuit simulé atteint 90% de son amélioration dans les 40 premières secondes. En revanche, la stratégie basée sur le commerce simulé associée à la méthode tabou converge encore plus vite, elle atteint 90% de son amélioration dans seulement 10 secondes de temps de calcul.

4.2 Interprétation des résultats

Au vu de ces résultats, plusieurs constatations s'imposent. En premier lieu, et comme nous le supposions, le recuit simulé, la métaheuristique de recherche tabou et l'algorithme génétique sont nettement plus rapides que l'algorithme de recherche arborescente IDA*. Cependant, la comparaison au niveau de la qualité de la solution profite à l'algorithme IDA*, étant donné que ce dernier se rapproche le plus de la solution optimale.

En second lieu, ces tests nous confirment l'inadéquation de l'utilisation de l'algorithme IDA*. En effet, la résolution de ce type de problème, la complexité en terme de temps et d'espace mémoire de cet algorithme est trop importante pour que cela soit réalisable. Néanmoins, pour de petits jeux d'essai, l'utilisation de IDA* est intéressante (cf. figure 7-a). Le temps d'exécution de IDA* croît d'une manière exponentielle. Il est

de beaucoup supérieur à celui des autres algorithmes sous-optimaux pour des grands jeux d'essai.

Les temps mis par les différentes stratégies pour trouver la solution sont à l'avantage de l'algorithme génétique, car ce dernier utilise des opérateurs génétiques simples : l'"hybridation" et la "mutation". Le premier regénère les plans en prenant deux d'entre eux, les coupant à une position aléatoire de la représentation de chacun d'eux, et les recombinant, de manière à ce que chacun ait une partie du code génétique de l'autre. La mutation modifie un plan sur des positions (de sa représentation) de manière aléatoire.

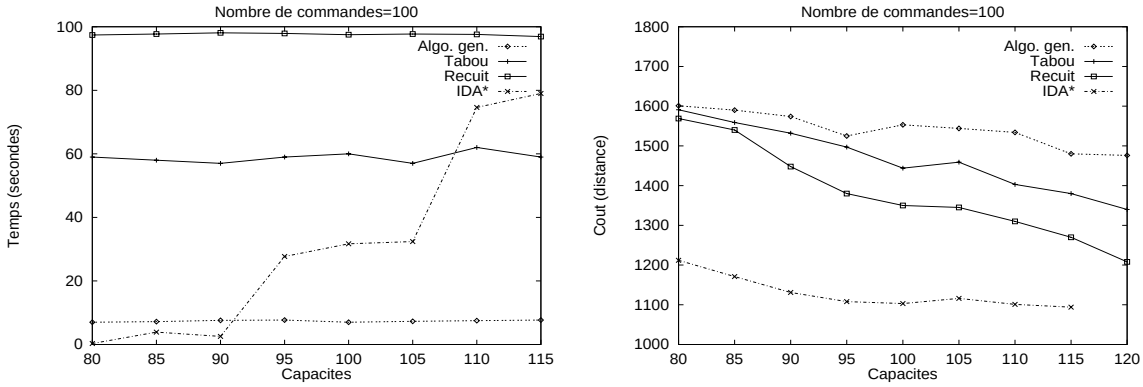

Figure 7: Comparaison des différentes stratégies en fonction : a - temps d'exécution, b - capacités

Enfin, lorsque les camions ont des capacités de chargement limitées, on n'obtient pas une amélioration considérable du coût du plan commun (cf. figure 7-b). Pour pallier à cet inconvénient, on peut subdiviser les commandes en unités plus petites en repassant par la phase de décomposition et de redistribution des tâches.

Ces premiers résultats nous ont permis de déterminer les algorithmes les plus adaptés pour la résolution du problème de planification; ce sont le recuit simulé, la métaheuristique de recherche tabou et l'algorithme génétique.

L'analyse des résultats montre bien que le CS + le recuit simulé et le CS + les tabous se comportent mieux que les algorithmes génétiques, pour toutes les configurations testées. On observe cependant un léger avantage pour le CS + recuit simulé. Cette dernière est d'une part rapide, et la solution trouvée est toujours plus intéressante. Nous pouvons extrapoler sur ces résultats et faire la supposition que cette supériorité restera toujours valide quel que soit le nombre de commandes. Cependant, la comparaison des solutions trouvées avec les solutions optimales n'a été réalisées que partiellement, étant donné que les algorithmes admissibles du type IDA* évoluent en temps et en espace mémoire, de manière exponentielle avec les tailles des problèmes traités. Le temps de résolution du jeu d'essai le plus important (100 commandes) avec le recuit simulé, est parfaitement raisonnable (97 secs.). L'exécution étant effectuée sur une machine SPARC 20 de SUN dotée de 32 Mo de RAM. Il est toutefois envisageable de résoudre des problèmes comptant quelques milliers de commandes.

En conclusion, nos résultats démontrent que l'ajout du recuit simulé comme de heuristique de planification au mécanisme de négociation basé sur le commerce simulé permet à ce dernier d'être plus performant sous les aspects de l'efficacité et de la capacité

de traitement. Cependant, ce résultat n'est valable que dans cette implantation adaptée au domaine du transport.

5 Conclusion

Tout d'abord, nous avons analyser le domaine du transport d'un point de vue systèmes multi-agents en vue d'identifier et d'utiliser des protocoles de négociation devant nous servir lors de la distribution et la décomposition de tâches entre les agents de transport. Suite à cela, nous avons développer une stratégie de négociation entre agents de transport basée sur le commerce simulé [1]. Dans cette stratégie, les agents sont amenés à vendre et à acheter des commandes en vue d'améliorer le plan global de leur entreprise. Des tests de simulation ont clairement montré qu'une telle approche de négociation permet d'augmenter la quantité de marchandises transportées tout en diminuant les coûts liés au transport de marchandises.

Le commerce simulé a été ensuite associé à des heuristiques permettant d'améliorer le temps de recherche d'une solution et la qualité de celle-ci : le recuit simulé, les tabous et IDA*. Les résultats obtenus par ces trois algorithmes ont été par la suite comparé aux algorithmes génétiques utilisés dans le cadre de la planification des commandes.

L'analyse des résultats montre bien que le CS + le recuit simulé et le CS + les tabous se comportent mieux que les algorithmes génétiques, pour toutes les configurations testées. On observe cependant un léger avantage pour le CS + recuit simulé. Cette dernière est d'une part rapide, et la solution trouvée est toujours plus intéressante.

Références

[1] A. Bachem, W. Hochstättler, and M. Malich. The simulated Trading Heuristic for Solving Vehicle Routing Problems. Technical Report 93.139, Mathematisches Institut, Universität zu Köln, Germany, April 1994.

[2] M. Desrochers, J. Desrosiers, and M. M. Solomon. A New Optimization Algorithm for Vehicle Routing Problem with Time Windows. *Operations Research*, 40(2), 1992.

[3] E. H. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers: Boston, MA, 1988.

[4] K. Fischer and J. P. Müller. A Decision-Theoretic Model for Cooperative Transportation Scheduling. In *Proc. of the Seventh European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'96)*, Eindhoven, The Netherlands, January 1996.

[5] K. Fischer, J. P. Müller, and M. Pischel. A Model for Cooperative Transportation Scheduling. In *Proceedings of the 1st International Conference on Multiagent Systems (ICMAS'95)*, San Francisco, June 1995.

[6] M. Gendreau, A. Herz, and G. Laporte. A tabu search heuristic for vehicle routing problem. *Management Science*, 40: 1276–1290, 1994.

[7] F. Glover. Tabu search part 1. *ORSA Journal on Computing*, 1: 190–206, 1989.

[8] F. Glover. Tabu search part 2. *ORSA Journal on Computing*, 2: 4–32, 1989.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Harbor, 1975.

[10] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimisation by simulated annealing. *Science*, 220(4598), 1983.

[11] R. E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27(1): 97–109, 1985.

[12] Sarit Kraus. Negotiation and cooperation in multi-agent environments. *Artificial Intelligence*, 94(1-2): 79–98, 1997.

[13] T. Kreifelts and F. von Martial. A negotiation framework for autonomous agents. In Y. Demazeau and J.-P. Müller, editors, *Decentralized AI 2 — Proceedings of the Second European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW-90)*, pages 71–88. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands, 1991.

[14] G. F. Luger and W.A. Stubblefield. *Artificial Intelligence, Structures and Strategies for Complex Problem solving*. The Benjamin/Cummings Publishing Company, Inc, 1993.

[15] I. H. Osman. Metastrategy simulated annealing and tabu search algorithms for vehicle routing problem. *Annals of Operational Research*, 41: 421–451, 1993.

[16] H. Raiffa. *The Art and Science of Negociation*. The Belknap Press of Harvard University Press, Cambridge, Massachusetts, 1982.

[17] J. Rosenschein and G. Zlotkin. Designing conventions for automated negotiation. *AI Magazine*, pages 29–46, 1994.

[18] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT press, 1994.

[19] Tuomas Sandholm and Victor Lesser. Coalitions among computationnally bounded agents. *Artificial Intelligence*, 94: 99–138, 1997.

[20] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12): 1104–1113, December 1980.

[21] M. M. Solomon. Algorithms for vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35: 254–265, 1987.

[22] K. Sycara. Utility theory in conflict resolution. *Annals of Operations Research*, 12: 65–84, 1988.

[23] K. Sycara. Negotiation planning: An ai approach. *European Journal of Operational Research*, 1989.