

RESOLUTION INTEGREE D'ORDONNANCEMENT ET D'AFFECTION : ADAPTATION D'UNE METHODE DE RECHERCHE A DIVERGENCE LIMITEE A UNE APPROCHE FONDEE SUR LES RESEAUX DE PETRI ET LA LOGIQUE LINEAIRE

C. MANCEL^{1, 2}, M.-J. HUGUET¹, P. LOPEZ¹, M. MEDJOUJ¹, R. VALETTE¹

¹LAAS-CNRS, 7 Avenue du Colonel Roche, 31077 Toulouse FRANCE

Tél. (+33) (0)-561 336 409, E-mail : cmancel@laas.fr

²GFI Consulting, 2-3 Passage de l'Europe, F-31400 Toulouse

RÉSUMÉ : Cet article montre l'application d'heuristiques développées dans le cadre de la recherche opérationnelle à des problèmes modélisés par réseaux de Petri. Le problème d'affectation et d'ordonnancement est transformé en un problème d'accessibilité, et la recherche arborescente est fondée sur la construction d'un ensemble d'arbres de preuve dans le cadre du calcul des séquents. Des indicateurs peuvent être introduits pour éviter l'exploration de tout le voisinage de la première solution trouvée.

MOTS-CLÉS : Réseaux de Petri, Heuristiques, Gestion de ressources

1. INTRODUCTION

Les techniques classiques de résolution de problèmes d'ordonnancement et d'affectation s'appuient sur une décomposition en deux sous-problèmes. Souvent, on résout dans un premier temps un problème d'affectation, puis on recherche un ordonnancement pour l'affectation trouvée. Les tests ont montré qu'une bonne affectation associée à un bon ordonnancement ne produisaient pas nécessairement une bonne solution globale car les deux problèmes sont fortement liés.

D'autre part, résoudre les deux problèmes simultanément conduit à une explosion combinatoire. C'est pourquoi, dans la pratique, on s'oriente souvent vers l'utilisation d'heuristiques guidant une recherche arborescente. Après l'obtention d'une première solution, on peut rechercher un certain nombre de solutions proches, c'est-à-dire ne s'éloignant que très ponctuellement des choix proposés par l'heuristique, mais améliorant la solution. C'est le principe de la technique appelée *recherche à divergence limitée* (LDS : Limited Discrepancy Search) [Harvey et Ginsberg, 1997, Medjoudj, 2002].

L'approche que nous avons retenue consiste à exploiter les mécanismes de propagation de contraintes pour détecter parmi les décisions proposées par l'heuristique celles qui sont susceptibles d'être remises en cause car localement mauvaises. Le fait de travailler sur un modèle fondé sur les réseaux de Petri, nous permet d'une part de mieux maîtriser les relations de causalité existant entre les décisions (amélioration de la propagation de contraintes qui s'effectue dans le cadre d'un ordre partiel) et d'autre part de mettre à jour des indicateurs sur la qualité locale des

décisions proposées par l'heuristique.

Le réseau de Petri modélisant le système est en fait vu comme un ensemble de formules en logique linéaire [Girard, 1987], et la construction d'une solution (affectation et ordonnancement simultanés) comme une preuve d'un séquent en logique linéaire. L'heuristique est utilisée pour déterminer à chaque étape quelle règle du calcul des séquents (et surtout avec quelles formules et quels atomes on va instancier la règle). La technique LDS nous permet de ré-écrire la première preuve obtenue en d'autres preuves, proches de la précédente et correspondant à des solutions potentiellement meilleures.

2. MODELISATION

2.1. Représentation par réseaux de Petri

La modélisation classique des problèmes d'ordonnancement consiste à associer une place à chaque ressource dans l'état disponible. Si le problème d'affectation se réduit au choix d'une ressource dans un pool de n ressources toutes identiques, il suffit de mettre initialement n jetons dans la place correspondante. Un tel schéma est représenté sur la figure 1 dans le cas de 3 demandes de ressources (places dem_1 , dem_2 et dem_3) et de 2 ressources identiques res_1 disponibles.

S'il n'y a qu'une ressource de type res_1 (impossibilité d'avoir plus d'un jeton dans la place correspondante), nous sommes en présence d'un problème purement d'ordonnancement. En termes de réseau de Petri, il s'agit de trouver la meilleure séquence de franchissement (l'ordre

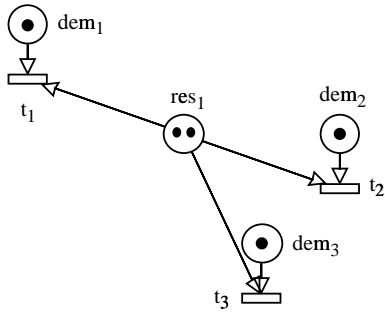


FIG. 1: Problème d'ordonnancement et d'affectation avec des ressources identiques

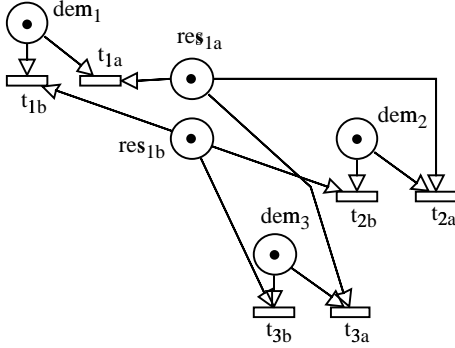


FIG. 2: Problème général d'ordonnancement et d'affectation

optimal) permettant de franchir une fois chacune des transitions t_1 , t_2 et t_3 . Si l'on met deux jetons dans la place res_1 on va devoir résoudre à la fois un problème d'affectation (pour chaque transition on doit choisir entre l'un des deux jetons contenus dans res_1) et d'ordonnancement.

Pour que le modèle ne contienne aucune ambiguïté, nous avons choisi de travailler avec un modèle déplié, c'est-à-dire tel que chaque place ressource est dupliquée en autant de places qu'elle peut contenir de jetons. Dans le cas des deux ressources, nous obtenons alors le réseau de la figure 2. Le problème d'affectation consiste à choisir entre les transitions t_{ia} et t_{ib} (pour $i = 1 \dots 3$) laquelle sera franchie. Le problème d'ordonnancement consiste à ordonner les transitions restant en conflit une fois l'affectation effectuée (les transitions non choisies sont effacées).

2.2. Traduction en logique linéaire

La traduction en logique linéaire se fait en suivant les mêmes principes que dans [Pradin-Chézalviel *et al.*, 1999]. L'accessibilité entre deux marquages M_1 et M_2 est équivalente à la prouvabilité d'un séquent de la forme $M_1, \mathcal{S} \vdash M_2$, où M_1 et M_2 sont des monômes en \otimes et \mathcal{S} est la liste des transitions à franchir. Dans nos travaux antérieurs, nous supposons que l'identité des transitions à franchir est

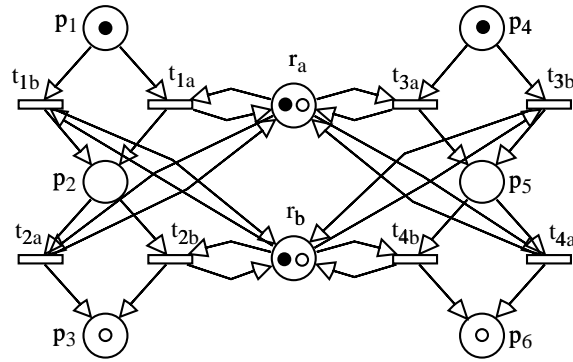


FIG. 3: Exemple illustratif

	r_a	r_b
t_1	3	4
t_2	4	3
t_3	3	4
t_4	4	3

FIG. 4: Durées des opérations en fonction des ressources

exactement connue et donc chaque élément de \mathcal{S} (chaque franchissement d'une transition t) est une expression de la forme $Pre(t) \multimap Post(t)$ ($Pre(t)$ et $Post(t)$ sont des monômes en \otimes). Ici, puisque nous avons une modélisation du type de celle de la figure 2, si on sait que l'on doit franchir une (et une seule) des transitions de chaque groupe (t_{ia} ou t_{ib} pour $i = 1 \dots 3$), on ne sait pas laquelle sera effectivement franchie.

Chaque élément de la liste \mathcal{S} doit alors avoir la forme $(Pre(t_{ia}) \multimap Post(t_{ia})) \& (Pre(t_{ib}) \multimap Post(t_{ib}))$ (1) où l'opérateur $\&$ peut être compris comme un ou exclusif avec un choix libre.

Lors de la construction d'un arbre de preuve du séquent, l'application de la règle du calcul des séquents permettant de ne conserver que l'un des termes du monôme en $\&$ (règle d'élimination à gauche du connecteur additif $\&$) correspond exactement à une décision d'affectation. Par contre, lorsque l'on choisit le terme de \mathcal{S} à éliminer (règle d'élimination à gauche de l'implication linéaire \multimap), cela correspond à une décision d'ordonnancement.

2.3. Exemple

Nous allons illustrer la démarche sur l'exemple très simple de la figure 3. On suppose, pour simplifier, que les opérations sont associées aux transitions. On a deux produits à faire. Le premier doit subir deux opérations représentées par les transitions t_1 et t_2 . Le second doit également subir deux opérations représentées par les transitions t_3 et t_4 . Comme deux ressources sont disponibles pour ces opérations (r_a et r_b), chaque transition est éclatée en deux (par

exemple t_1 est éclatée en t_{1a} et t_{1b}). Les ressources n'étant pas équivalentes, le tableau 4 donne les durées des opérations en fonction des ressources utilisées. L'état initial est représenté par les jetons noirs et l'état final par les jetons blancs. Le séquent à prouver est donc le suivant :

$$p_1 \otimes p_4 \otimes r_a \otimes r_b, t_1, t_2, t_3, t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (2)$$

avec :

$$t_1 : (p_1 \otimes r_a \multimap p_2 \otimes r_a) \& (p_1 \otimes r_b \multimap p_2 \otimes r_b)$$

$$t_2 : (p_2 \otimes r_a \multimap p_3 \otimes r_a) \& (p_2 \otimes r_b \multimap p_3 \otimes r_b)$$

$$t_3 : (p_4 \otimes r_a \multimap p_5 \otimes r_a) \& (p_4 \otimes r_b \multimap p_5 \otimes r_b)$$

$$t_4 : (p_5 \otimes r_a \multimap p_6 \otimes r_a) \& (p_5 \otimes r_b \multimap p_6 \otimes r_b)$$

Vis-à-vis du séquent $M_1, \mathcal{S} \vdash M_2$, nous avons :

$$M_1 = p_1 \otimes p_4 \otimes r_a \otimes r_b \quad (3)$$

$$M_2 = p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (4)$$

$$\mathcal{S} = t_1, t_2, t_3, t_4 \quad (5)$$

3. HEURISTIQUE POUR LA CONSTRUCTION D'UNE SOLUTION

3.1. Principe

Il faut une heuristique pour l'ordonnancement et une heuristique pour l'affectation. La première, dans la construction de la preuve en logique linéaire, donne l'ordre dans lequel les éléments de la liste \mathcal{S} sont pris en considération, la seconde donne pour chaque élément de \mathcal{S} , le terme devant être conservé (par exemple $(p_1 \otimes r_b \multimap p_2 \otimes r_b)$ pour t_1).

L'heuristique d'affectation consiste donc à établir une priorité au sein des listes de ressources associées à chaque tâche (par exemple en fonction des valeurs croissantes de la durée opératoire). Ainsi dans le tableau de la figure 4 on choisira r_a pour t_1 et r_b pour t_2 . Lors de la construction de l'arbre de preuve, cela va consister à éliminer les connecteurs $\&$ en choisissant $(p_1 \otimes r_a \multimap p_2 \otimes r_a)$ pour t_1 et $(p_2 \otimes r_b \multimap p_3 \otimes r_b)$ pour t_2 .

Le rôle de l'heuristique d'ordonnancement, dans le cadre de notre modélisation, est de choisir un ordre pour l'élimination des éléments de la liste \mathcal{S} . Diverses informations peuvent être utilisées pour trouver un ordre intéressant. Par exemple dans [Sellami, 2001] l'auteur propose un ensemble de règles portant sur les dates au plus tôt et au plus tard de début d'exécution qui permettrait de trouver l'ordre $\mathcal{O} = t_1; t_3; t_2; t_4$

3.2. Calcul des dates

L'heuristique d'ordonnancement donne l'ordre dans lequel les variables de type "date de début de tâche" se-

ront calculées (et non l'ordre de ces dates). Quand une variable de type "date de début de tâche" est considérée, on choisit d'abord la ressource (heuristique d'affectation), et, comme l'heuristique d'ordonnancement respecte les relations de précédence, nous avons normalement toutes les informations pour calculer la date (au plus tôt) de début par simple propagation de contraintes.

Lorsque l'on travaille sur l'arbre de preuve (cf [Pradin-Chézalviel *et al.*, 1999]), la procédure est exactement la même. La propagation des contraintes de précédence et des contraintes temporelles quantitatives se fait automatiquement par le calcul des dates de production des jetons. A chaque jeton est associée sa date de production (c'est la parenthèse à droite du nom du jeton).

Supposons dans l'exemple de la figure 3 que les demandes de fabrication (jetons dans les places p_1 et p_4) sont disponibles à la date 0 et qu'à cette même date les ressources sont disponibles. Le premier séquent s'écrit (après élimination des connecteurs \otimes pour transformer le marquage M_1 en une liste d'hypothèses indépendantes) :

$$p_1(0), p_4(0), r_a(0), r_b(0), t_1, t_2, t_3, t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b(6) \quad (6)$$

L'heuristique d'ordonnancement (la séquence \mathcal{O}) spécifie l'élimination (règle du calcul des séquents pour l'élimination à gauche de l'implication) de t_1 d'abord et celle d'affectation l'utilisation de la ressource r_a (franchissement de t_{1a}). Comme la durée opératoire est égale à 3 nous obtenons le séquent suivant :

$$p_2(3), p_4(0), r_a(3), r_b(0), t_2, t_3, t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (7)$$

Puis nous éliminons t_3 avec r_a . Cette opération ne peut se faire que lorsque la ressource r_a devient disponible à la date 3. Nous obtenons :

$$p_2(3), p_5(6), r_a(6), r_b(0), t_2, t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (8)$$

Puis nous éliminons t_2 avec la ressource r_b , également à la date 3 :

$$p_3(6), p_5(6), r_a(6), r_b(6), t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (9)$$

Puis nous éliminons t_4 avec la ressource r_b à la date 6 :

$$p_3(6), p_6(9), r_a(6), r_b(9) \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (10)$$

Le séquent est prouvé, on a donc une solution admissible donnant une durée égale à 9.

3.3. Indicateurs

Le fait de travailler sur un modèle fondé sur les réseaux de Petri permet d'associer une interprétation systématique des manipulations effectuées sur les variables. Les places représentent en effet soit des états des ressources, soit des états des produits (demandeurs de ressource) entre deux opérations. Toute attente d'un jeton dans une place du premier type correspond donc à une ressource non utilisée pendant l'intervalle de temps correspondant. Toute attente dans une place du second type correspond à une attente pour un produit.

Lors de chaque élimination d'une transition dans l'arbre de preuve, le travail de propagation des contraintes temporelles consiste d'abord à déterminer la première date de franchissement possible. Cette date est la plus grande valeur des estampilles temporelles (date d'arrivée des jetons dans les places) des jetons consommés par la transition. L'écart entre cette date et l'estampille de chaque jeton donne l'attente du jeton dans la place.

Par exemple, lors du premier pas de l'élaboration de la solution de l'exemple de la figure 3, on élimine t_1 du séquent 6. Les estampilles des jetons dans p_1 et r_a sont toutes les deux égales à 0. Il n'y a donc pas d'attente. Au pas suivant (séquent 7), il y a une attente de 3 dans la place p_4 (le produit 2 attend que la ressource r_a soit disponible). Au pas suivant (séquent 8), il y a une attente de 3 pour la ressource r_b . Au pas 9 il n'y a pas d'attente. Nous voyons donc que la solution élaborée à partir des heuristiques comprend à la fois une attente pour une tâche (t_3) et une ressource sous-utilisée (r_b) qui pourrait être utilisée par t_3 . Il peut donc être intéressant de chercher à l'améliorer.

Cette recherche peut être une recherche locale en limitant la divergence par rapport à la solution initiale et en utilisant des heuristiques pour n'explorer que les points de divergence intéressants.

4. RECHERCHE A DIVERGENCE LIMITEE

4.1. Principe

En présence d'une très importante explosion combinatoire, il n'est pas possible de rechercher systématiquement toute les solutions par une recherche arborescente pour savoir quelle est la meilleure. On se contente en pratique de solutions approchées obtenues rapidement et fondées sur des méta-heuristiques et des heuristiques. Toutefois, même de bonnes heuristiques peuvent être ponctuellement mauvaises, c'est pourquoi il peut être intéressant d'explorer le voisinage immédiat de la première solution trouvée pour voir s'il n'est pas possible de l'améliorer. Le principe de la méthode à divergence limitée (LDS) est de ne se mettre en discordance avec les heuristiques qu'un certain nombre de fois, défini à l'avance. Une recherche de divergence 2 consiste à travailler dans un espace de solutions qui ne sont en discordance avec les heuristiques que pour deux décisions.

Dans notre contexte, les discordances que l'on va évaluer sont celles qui concernent l'heuristique d'affectation. C'est-à-dire qu'au lieu de prendre la première ressource de la liste on va prendre la deuxième (ou la troisième pour une discordance de 2) et cela pour un nombre limité de tâches.

Pour être efficace, nous allons utiliser les indicateurs définis ci-dessus (retard de début de tâche et intervalle de non

utilisation de ressource) pour savoir quelles décisions vont être remise en cause. C'est-à-dire que nous allons en fait explorer un sous-espace de celui défini par le degré de divergence. Il sera ainsi possible de considérer un degré de divergence un peu plus élevé tout en limitant l'explosion combinatoire. Seul le voisinage supposé intéressant sera exploré.

L'idée est de ne remettre en cause que les décisions qui localement ne sont pas optimales car elles entraînent des attentes ou des mauvaises utilisations des ressources. La règle de priorité est la suivante :

- Ne considérer que les décisions ayant entraîné un retard du à une attente de ressource, et seulement si la ressource suivante dans la liste de l'heuristique d'affectation est apparue dans les indicateurs avec un intervalle de non utilisation,
- Remettre en cause les décisions dans l'ordre proposé par l'heuristique d'ordonnement.

Cela revient à combiner une approche gloutonne (recherche de l'optimalité locale) avec l'heuristique d'ordonnement. Le respect de l'ordre fixé par l'heuristique d'ordonnement a pour but d'éviter qu'une nouvelle divergence ne remette en cause une optimisation locale résultant d'une divergence effectuée auparavant.

4.2. Retour sur l'exemple

Nous avons eu une attente de 3 au pas 7, et une ressource non utilisée de 3 au pas 8. Donc nous allons remettre d'abord en cause la décision d'affectation du pas 7 (sans rien modifier concernant l'heuristique d'ordonnement, c'est-à-dire l'ordre dans lequel les tâches sont examinées). Donc à partir du séquent 7 nous éliminons t_3 avec r_b ce qui donne (il n'y a plus d'attente) :

$$p_2(3), p_5(4), r_a(3), r_b(4), t_2, t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (11)$$

Maintenant il faut éliminer t_2 avec r_b comme l'indique l'heuristique d'affectation. Il y a une attente de 1 pour la tâche.

$$p_3(7), p_5(4), r_a(3), r_b(7), t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (12)$$

Maintenant nous éliminons t_4 avec la ressource r_b . Il y a une attente de 3 pour la tâche t_4 .

$$p_3(7), p_6(10), r_a(3), r_b(10) \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (13)$$

Globalement la durée est maintenant de 10 (donc elle est moins bonne) et la ressource r_a est très mal utilisée puisque son intervalle de non utilisation finale est d'une durée de 7 (l'estampille finale de r_a est 3). Néanmoins, il y a eu une attente pour t_2 et la ressource alternative proposée par l'heuristique d'affectation est r_a qui est sous-utilisée. Donc nous pouvons aller à une divergence de 2 en éliminant t_2 avec r_a dans le séquent 11. Nous obtenons :

$$p_3(7), p_5(4), r_a(7), r_b(4), t_4 \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (14)$$

Maintenant nous éliminons t_4 avec la ressource r_b . Il n'y a pas d'attente.

$$p_3(7), p_6(7), r_a(7), r_b(7) \vdash p_3 \otimes p_6 \otimes r_a \otimes r_b \quad (15)$$

Nous avons construit une solution pour laquelle il n'y a plus aucune attente, donc nous pouvons stopper la recherche à divergence limitée. La remise en cause de deux affectations nous a permis de passer d'une durée de 9 à une durée de 7. Une analyse plus poussée montrerait que la solution obtenue est optimale.

5. DISCUSSION

On peut avoir l'impression que nous n'avons remis en cause que l'affectation car il y a en effet une différence entre l'heuristique d'affectation et l'heuristique d'ordonnement. La première donne un ordre de priorité (c'est-à-dire une relation de préférence) pour l'affectation des ressources aux opérations. Il est alors facile de définir une divergence : au lieu de choisir la ressource la meilleure, nous choisissons la deuxième (ou la troisième) de la liste. Par contre, l'heuristique d'ordonnement fixe la séquence des prises de décision concernant l'ordre de passage des opérations sur les machines et le calcul des dates de début de ces opérations. Cet ordre est le résultat d'une évaluation des dates de début au plus tôt et au plus tard des opérations valable pour toute affectation et tout ordonnancement. Cette heuristique n'explique aucune relation de préférence et il est donc plus difficile de la remettre en cause. C'est pourquoi la divergence ne se fait que vis-à-vis de l'heuristique d'affectation.

Cela ne veut pas dire que l'ordonnement n'est pas modifié. En effet, l'heuristique d'ordonnement ne fixe que l'ordre des prises de décision et non l'ordonnement lui-même. Une décision d'ordonnement implique le calcul d'une date de début d'opération par propagation des contraintes d'affectation et des contraintes liées aux décisions d'ordonnement antérieures. Ce n'est pas parce que le calcul de la date de franchissement de la transition t_i est effectué avant celui de la transition t_j que t_i sera nécessairement franchie avant t_j . Cela n'est le cas que lorsque t_i et t_j concernent le même produit ou utilisent la même ressource.

Dans l'exemple ci-dessus, considérons la ressource r_b . Pour la première solution, l'ordonnement des opérations sur cette ressource consiste à faire l'opération t_2 au temps 3 puis l'opération t_4 au temps 6. L'ordonnement de cette même ressource correspondant à la solution obtenue pour la divergence de deux consiste à faire l'opération t_3 au temps 0 puis l'opération t_4 au temps 4. Les deux ordonnements sur cette ressource sont donc bien différents même en l'absence de divergence vis-à-vis de l'heuristique d'ordonnement.

Dans les cas complexes (plusieurs ressources affectées à une opération avec, de plus, la possibilité de conserver des ressources pendant plus d'une opération), l'heuristique d'ordonnement peut se trouver en contradiction avec la propagation des contraintes définies par les décisions antérieures d'affectation et d'ordonnement. Cela

se traduira par le fait que l'heuristique d'ordonnement va demander de calculer la date de franchissement d'une transition non franchissable. Il faudra donc prendre la séquence de décisions fixée par l'heuristique d'ordonnement comme une relation de préférence et ne prendre à chaque instant que les transitions franchissables dans cette liste. Dans cette situation nous aurons donc également des divergences vis-à-vis de l'heuristique d'ordonnement et non pas seulement des divergences vis-à-vis de l'heuristique d'affectation. Il subsistera toutefois une différence : les divergences vis-à-vis de l'heuristique d'ordonnement seront imposées par la propagation des contraintes et non librement décidées.

6. CONCLUSION

Le travail que nous venons de présenter illustre qu'il est tout à fait possible d'utiliser des heuristiques développées dans le cadre de la recherche opérationnelle pour construire un arbre de preuve intéressant (ou un ensemble d'arbres de preuve intéressants) pour prouver des séquents décrivant des problèmes d'accessibilité sur des réseaux de Petri.

Plus précisément, dans le cadre des problèmes mixtes d'ordonnement et d'affectation, les heuristiques classiques peuvent parfaitement être utilisées. On peut ensuite essayer d'améliorer la première solution trouvée en utilisant une recherche à divergence limitée. Il est clair que plus la divergence acceptée sera grande, plus on explorera de solutions et plus on se rapprochera de la solution optimale. L'idée de la divergence limitée est de donner des solutions qui correspondent à des heuristiques qui sont bonnes pour la grande majorité des décisions, mais qui sont mauvaises dans quelques rares situations. Si la solution est améliorée progressivement et de façon continue lorsque la divergence augmente, c'est que l'heuristique n'est pas bonne. Dans le cas contraire, nous devons arriver rapidement à un minimum (éventuellement local) en sachant quels sont les choix de l'heuristique qui ont été remis en cause. Cette approche sert donc également à évaluer la qualité d'une heuristique et son adéquation à un cas particulier.

Nous avons également proposé de nous appuyer sur la modélisation par réseaux de Petri pour associer des indicateurs à chaque décision. En s'appuyant sur ces indicateurs, on peut éviter la recherche exhaustive de toutes les solutions divergeant d'un degré donné, en se limitant à celles qui semblent les plus intéressantes.

Ce travail est encore préliminaire, car l'approche présentée doit être validée. Pour cela il faut développer un logiciel pour la mettre en œuvre et la tester sur des "benchmarks". S'il existe des "benchmarks" pour des problèmes types de recherche opérationnelle, il n'en existe pas, à notre connaissance, dans le cadre des

approches fondées sur les réseaux de Petri, il faut donc également développer un outil pour les générer.

REFERENCES

- [Girard, 1987] J.Y. Girard : Linear Logic, *Theoretical Computer Science*, n°50, 1987.
- [Gunter et Gehlot, 1989] C. Gunter, V. Gehlot: Nets as tensor theories, In: Proceedings of the 10th International Conference on Application and Theory of Petri Nets, 1989, Bonn, Germany, pages 174-191. 1989.
- [Harvey et Ginsberg, 1997] W. D. Harvey, M.L. Ginsberg. Limited discrepancy search, Technical report, CIRL University of Oregon, Oregon 97403, USA
- [Mancel *et al.*, 2002] C. Mancel, P. Lopez, N. Rivière, R. Valette : Relationships between Petri nets and constraint graphs: application to manufacturing, 15th IFAC world congress, Barcelona, Spain, 21-26 July 2002, n. 634.
- [Medjoudj, 2002] M. Medjoudj. Résolution intégrée de problèmes d'ordonnancement et d'affectation : adaptation d'une méthode de recherche à divergence limitée (LDS), rapport de DEA, Ecole doctorale systèmes, Toulouse Juin 2002.
- [Pradin-Chézalviel *et al.*, 1999] B. Pradin-Chézalviel, R. Valette, L.A. Künzle: Scenario duration characterization of t-timed Petri nets using linear logic, IEEE PNPM'99, 8th International Workshop on Petri Nets and Performance Models, Zaragoza, Spain, September 6-10, 1999, p.208-217.
- [Sellami, 2001] I. Sellami. Problèmes mixtes d'ordonnancement et d'affectation. Mémoire d'ingénieur, ENAC juin 2001.