Vers la coopération des métaheuristiques

Vincent Bachelet *
Zouhir HAfidi **
Philippe Preux ***
El-Ghazali Talbi ****

* *ENS Mines de Douai, 941, rue Charles Bourseul, BP 838, 59508 Douai cedex*
bachelet@ensm-douai.fr
** *LIFL, URA CNRS 369, Université de Lille 1, 59655 Villeneuve d'Ascq cedex*
hafidi@lifl.fr
*** *LIL, Université du Littoral, BP 719, 62228 Calais cedex*
preux@lil.univ-littoral.fr
**** *LIFL, URA CNRS 369, Université de Lille 1, 59655 Villeneuve d'Ascq Cedex*
talbi@lifl.fr

RÉSUMÉ. *Nous nous proposons ici de revisiter la notion de parallélisme. Le parallélisme n'est pas qu'une source de puissance de calcul. L'activité concurrente de plusieurs algorithmes peut accélérer des traitements par le biais d'une coopération entre ces algorithmes. Dans cet article, nous discutons ce point de vue que nous appliquons à un problème d'optimisation combinatoire, le problème d'affectation quadratique. Ce travail fait suite à des études jusqu'alors cantonnées à la résolution de problèmes purement artificiels, sans intérêt pratique. Nous montrons ici que cette idée est également valable pour des problèmes du monde réel, en vraie grandeur. Bien entendu, la coopération entre des algorithmes ne transforme pas un problème NP en problème polynomial. Cependant, il s'avère que des événements statistiquement rares pour un algorithme stochastique (par exemple, découvrir un optimum local d'une certaine qualité) ont une probabilité d'occurrence beaucoup moins négligeable quand plusieurs algorithmes coopèrent.*

ABSTRACT. *The notion of parallelism is revisited. Parallelism is not solely a source of computational power. Concurrent activity of many algorithms can speed-up processing via the cooperation between algorithms. In this paper, we discuss this point of view and apply it to combinatorial optimization, the quadratic assignment problem. This work is a follow-up to earlier work dedicated to purely artificial problems, without any practical interest. We show here that this idea can also be applied to real size problems of the real world. Obviously, cooperation between algorithms does not change an untractable problem into a tractable one. However, rare events for a stochastic algorithm (such as finding a local optimum of a certain quality) can become far less rare when some algorithms are cooperating.*

MOTS-CLÉS : *coopération, métaheuristique, optimisation combinatoire.*

KEY WORDS: *cooperation, metaheuristic, combinatorial optimisation.*

1. Introduction

Classiquement, le parallélisme a été considéré comme un moyen d'augmenter la puissance de calcul en exécutant des tâches simultanément. Dans ce but, ont été développées des plates-formes SIMD, MIMD, réseaux ou fermes de stations de travail avec le sempiternel objectif de tirer parti du fait qu'exécuter des tâches en parallèle est plus rapide que les exécuter l'une après l'autre.

Une perspective complètement différente est cependant apparue depuis quelques années, beaucoup moins technique, beaucoup plus conceptuelle, que l'on peut notamment relier à l'intelligence artificielle distribuée (IAD). Ici, l'objectif n'est plus d'accélérer les traitements ; l'intérêt majeur est d'avoir des activités concurrentes. De plus, l'implantation est reléguée au second ordre, elle peut être aussi bien parallèle que séquentielle. L'IAD simule des systèmes complexes composés de nombreux agents en interaction tels des colonies d'insectes dont s'inspirent les algorithmes « en essaim » (modèle des fourmis par exemple [DG97]). Les réseaux d'automates cellulaires rentrent dans ce cadre de pensée également. Dans la même veine conceptuelle, quelques rares travaux en optimisation ont montré expérimentalement que la coopération d'activités permet d'obtenir des optima de meilleure qualité. Ainsi, [MSK95] montre comment plusieurs activités concurrentes trouvent des états d'énergie plus bas dans des modèles de verre de spin ; [FRP97b] montre que des algorithmes génétiques coopérant par échange de solutions résolvent facilement un problème (artificiel) alors que chacun des algorithmes génétiques pris séparément en est incapable (la probabilité qu'il trouve l'optimum global est négligeable en pratique). Nous souhaitons ici poursuivre sur cette voie en nous attaquant cette fois-ci à un problème d'optimisation combinatoire standard. [VA96] présente les quelques travaux réalisés selon cette approche qui consiste essentiellement à l'exécution parallèle de plusieurs tabous ou algorithmes génétiques. Notons que l'idée de ces travaux repose sur la constatation expérimentale que ce type d'algorithmes fournit de meilleurs optima locaux. En ce qui nous concerne, nous sommes guidés par des travaux en cours sur la structure des espaces de recherche [FRP97a, FRPT97] qui éclairent les raisons pour lesquelles cette approche fonctionne, mais aussi sur quels types d'instances elle peut fonctionner et sur lesquelles elle ne peut vraisemblablement rien apporter. Nous n'avons pas la place ici de décrire ces recherches, mais il est important que le lecteur sache que les idées que nous avançons ici sont fondées sur des études de la structure des espaces de recherche.

Même si elle n'a été que rarement exprimée auparavant sous cette forme, l'idée de faire coopérer des algorithmes de recherche est classique en optimisation combinatoire. On parle alors d'algorithmes hybrides dont on sait qu'ils sont supérieurs aux algorithmes de recherche « purs » en ce qui concerne la qualité des optima trouvés et le temps mis pour les trouver [FF95, PT97]. Cette coopération entre des algorithmes prend généralement l'une des formes suivantes[1] :

– *hybride séquentiel* où deux algorithmes sont impliqués l'un après l'autre ; les résultats fournis par le premier étant les solutions initiales du second à la manière d'un pipeline ;

1. Nous reprenons la terminologie utilisée dans [PT97].

– *hybride parallèle synchrone* où un algorithme de recherche est utilisé à la place d'un opérateur (exemple : utiliser un tabou à la place d'un opérateur de mutation dans un algorithme évolutif) ;

– *hybride parallèle asynchrone* où plusieurs algorithmes de recherche travaillent concurremment et s'échangent des informations.

Si les deux premières catégories d'hybrides ont été largement étudiées, peu d'étude concerne la dernière, notamment dans le cadre de l'optimisation de problèmes $\mathcal{NP}$-durs. Dans cette dernière catégorie, nous proposons une classification en trois types. Un hybride pouvant, indépendamment de son type, impliquer des méthodes de recherche identiques ou des méthodes différentes, est qualifié respectivement d'hybride homogène ou d'hybride hétérogène :

– pour le premier type, tous les algorithmes coopérant optimisent *le même problème*. L'échange d'information entre les algorithmes permet par exemple de faire sortir un algorithme d'un optimum local ou d'une zone de l'espace de recherche sans intérêt ; c'est ce que nous avons réalisé dans [FRP97b] ;

– dans le deuxième type d'hybride asynchrone, *le problème est décomposé*. L'espace de recherche est partagé et chaque partie est optimisée indépendamment. Généralement, les sous-méthodes de recherche communiquent entre elles pour résoudre les contraintes induites par la décomposition du problème et construire une solution globale ;

– pour le dernier type, plusieurs algorithmes de recherche traitent *des problèmes différents* mais coopèrent pour la résolution d'un même problème. Cette dernière approche, encore peu explorée à notre connaissance, est abordée dans l'étude présentée dans cet article.

En ce qui concerne l'hybride que nous avons réalisé, un algorithme génétique et un algorithme de recherche tabou collaborent alors qu'ils traitent deux problèmes d'optimisation distincts. Le tabou résout le problème propre pendant que l'algorithme génétique lui fournit des solutions de départ dans des zones de l'espace de recherche encore peu explorées. Ils coopèrent à travers une zone d'échange, *la mémoire adaptative*, à laquelle la méthode tabou communique les zones déjà explorées et l'algorithme génétique communique de nouveaux points dans des zones non encore visitées.

Dans la suite de cet article, nous rappelons tout d'abord la définition du problème d'affectation quadratique (QAP) et nous donnons quelques définitions préalables. Nous présentons l'algorithme hybride et des résultats expérimentaux issus de son application au QAP. Enfin, nous terminons par une discussion de ces résultats et des perspectives ouvertes par ce travail.

2. Définitions

2.1. *Le problème d'affectation quadratique*

Nous utilisons le problème d'affectation quadratique (QAP) comme problème test. Cependant, la classe d'algorithmes que nous discutons ici peut s'appliquer à tout autre problème d'optimisation.

Le QAP est un problème important en optimisation combinatoire qui possède de nombreuses applications (placement, ordonnancement, synthèse d'images, etc.). Le QAP est un problème $\mathcal{NP}$-dur [GJ79] et des algorithmes exacts (tel que le *branch-and-bound*) ne peuvent traiter que des instances de petite taille ($n < 25$). Aussi, ont été proposées dans la littérature, de nombreuses heuristiques ; plutôt que d'en faire un traitement superficiel, nous préférons indiquer au lecteur un état de l'art récent [PRW94].

2.1.1. *Définition*

Une instance du QAP peut être définie par :

- n, la taille de l'instance ;
- un ensemble de n objets $O = \{O_1, O_2, ..., O_n\}$;
- un ensemble de n positions $P = \{P_1, P_2, ..., P_n\}$;
- une matrice de flux C, dans laquelle chaque élément (c_{ij}) indique le flux entre les objets O_i et O_j ;
- une matrice de distance D dans laquelle chaque élément (d_{kl}) donne la distance entre les positions P_k et P_l.

Ces éléments étant donnés, trouver un positionnement des objets, c'est-à-dire une bijection $U : O \longrightarrow P$, qui minimise la fonction Φ :

$$\Phi(U) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} . d_{U(i)U(j)}$$

2.1.2. *Représentation des solutions*

Une solution au QAP peut être représentée de multiples manières. Dans cet article, nous utilisons la même représentation dans tous les algorithmes qui est une permutation de n entiers.

$$u = (u_1, u_2, ..., u_n)$$

dans laquelle $u_i = U(i)$ est la position de l'objet O_i. Ainsi, la fonction objectif φ est définie sur les permutations par :

$$\forall u, \varphi(u) = \Phi(U)$$

2.2. *Métrique dans l'espace des solutions*

Nous munissons l'ensemble des $n!$ points de l'espace de recherche d'une instance du QAP d'une métrique qui nous sera utile plus loin. Au préalable, cet ensemble de points doit être structuré. Cette structure est donnée par l'opérateur utilisé pour transformer un point en un autre ; ce même opérateur qui construit de nouveaux points dans notre algorithme hybride. Cet opérateur, $swap$, effectue un échange de deux objets dans une permutation. On peut le définir comme suit :

- soit V l'ensemble des $n!$ points de l'espace de recherche ;
- $(i,j) \in \{1,...,n\}^2, i \neq j$

$$swap_{ij} : \quad V \longrightarrow V$$
$$u \mapsto v = \begin{cases} v(i) = u(j) \\ v(j) = u(i) \\ v(k) = u(k), \forall k, k \neq i, k \neq j \end{cases}$$

L'ensemble des permutations muni de cet opérateur engendre un graphe $G(V,E)$ où E est l'ensemble des couples $(u,v) \in V^2$ tels que $\exists (i,j) \in \{1,...,n\}^2,\ i \neq j$, $v = swap_{ij}(u)$. Ce graphe G est l'espace de recherche associé à l'opérateur $swap$.

2.2.1. *Distance entre permutations*

On définit la distance $dist(u,v)$ entre u et v comme la longueur du plus court chemin entre u et v dans le graphe G. Cette distance est calculable en $O(n)$ opérations.

2.2.2. *Entropie d'une population de permutations*

L'algorithme que nous proposons travaillant sur une population de solutions [2], il importe de pouvoir quantifier la diversité de cette population, la diversité de la population d'un algorithme génétique étant un facteur essentiel pour son comportement. Nous suivons [FF94] en définissant une notion d'entropie Γ qui mesure cette diversité. L'entropie d'un objet O_i est donnée par :

$$\Gamma_i = \frac{\sum_{j=1}^{n} \frac{n_{ij}}{m} . \log \frac{n_{ij}}{m}}{-\log n}$$

L'entropie de la population est l'entropie moyenne de ses objets, soit :

$$\Gamma = \frac{\sum_{i=1}^{n} \Gamma_i}{n}$$

où :

- n_{ij} est le nombre d'affectations de l'objet O_i à la position P_j ;
- m est la cardinalité de la population de permutations.

On a $0 \leq \Gamma \leq 1$. Si $\Gamma = 0$, la population contient m fois la même permutation. Si $\Gamma = 1$, la diversité de la population est maximale.

2. Et non un ensemble, le même point pouvant apparaître plusieurs fois.

3. Algorithme hybride

Notre algorithme hybride utilise deux briques de base, d'une part une recherche tabou, d'autre part un algorithme génétique. Ces deux algorithmes sont des variantes des algorithmes itératifs locaux (AIL) [AL97]. Contrairement aux algorithmes constructifs, les AIL démarrent avec un point de l'espace de recherche (éventuellement pris au hasard) qu'ils tentent d'améliorer en utilisant un opérateur. Cet opérateur produit des voisins au point courant. L'algorithme sélectionne alors l'un des points voisins pour devenir le point courant. Nous donnons ici, très brièvement, les caractéristiques générales de ces deux algorithmes.

3.1. *Les briques de base*

La recherche tabou [Glo95] considère un ensemble de points voisins du point courant ; puis le meilleur voisin, dans cet ensemble, est choisi pour être le point courant suivant, même s'il est de qualité inférieure au point courant précédent. Cependant, cette stratégie a tendance à revenir sur des points déjà visités ; c'est pourquoi la méthode tabou utilise une mémoire à court terme : *la liste tabou*. Cette liste conserve une information sur les dernières itérations de l'algorithme qui permet de se prémunir contre les cycles courts. De plus, la recherche tabou intègre un mécanisme, appelé *aspiration*, qui sélectionne un voisin que la liste tabou rend inaccessible. Généralement, seuls les points d'une qualité inégalée depuis le début de la recherche, sont sélectionnés par l'aspiration. Souvent, la recherche se poursuit jusqu'à ce qu'un nombre donné d'itérations soit effectué ; néanmoins d'autres critères d'arrêt peuvent être appliqués. Par ailleurs, afin de mieux exploiter l'espace de recherche, certaines recherches tabous plus élaborées recourent à des techniques de diversification fondées sur le concept de mémoire à moyen terme.

Quant à l'algorithme génétique, inspiré de l'évolution naturelle des êtres vivants, [Mit96, Pre95], il ne travaille pas avec un unique point courant, comme la recherche tabou, mais il fait évoluer une population de points. L'algorithme génétique classique, fonctionne avec deux opérateurs, la recombinaison (ou *crossing over*) et la mutation. A partir d'une population de points (dénommés les parents), cet algorithme construit une nouvelle population (les enfants) en combinant plusieurs parents (recombinaison) et en appliquant des modifications aléatoires (mutation). La phase de sélection produit la population pour l'itération suivante en choisissant les meilleurs points parmi les parents et les enfants. Habituellement, l'algorithme génétique converge, c'est-à-dire que la population tend à perdre sa diversité, il perd alors de son efficacité ; c'est pourquoi la convergence est souvent utilisée comme critère d'arrêt. Toutefois, il est fréquent de stopper la recherche après un nombre donné d'itérations (encore dénommées générations).

3.2. *La mémoire adaptative*

L'algorithme hybride que nous proposons, combine des algorithmes tabous fonctionnant concurremment et un algorithme génétique, lui aussi fonctionnant concurremment avec les tabous (voir la figure 1). Le principe de cet hybride est de favoriser

une exploration aussi complète que possible de l'espace de recherche. L'algorithme tabou a plutôt tendance à n'explorer qu'une partie de l'espace de recherche. L'algorithme génétique est chargé de fournir des points de départ pour la recherche effectuée par les tabous. Aussi, cet algorithme doit construire des points initiaux éloignés des points déjà visités pour alimenter les tabous. La mémoire adaptative est une structure de données qui aide l'algorithme génétique à fournir des points de départ dans des zones non encore explorées de l'espace. Cette mémoire comporte deux éléments : une mémoire de fréquence et une mémoire de points de départ.

La mémoire de fréquence stocke des informations relatives à tous les points visités par les tabous durant leurs recherches. Elle mémorise la fréquence d'occurrence de certains événements durant les itérations des tabous. Pour le QAP, cet événement est ici « l'objet O_i est placé sur la position P_j ». La mémoire de points de départ stocke tous les points fournis par l'algorithme génétique pour démarrer les tabous. En utilisant ces informations, l'algorithme génétique peut fournir des solutions initiales non encore utilisées et se trouvant dans des zones qui pourraient se révéler fructueuses *a posteriori*.

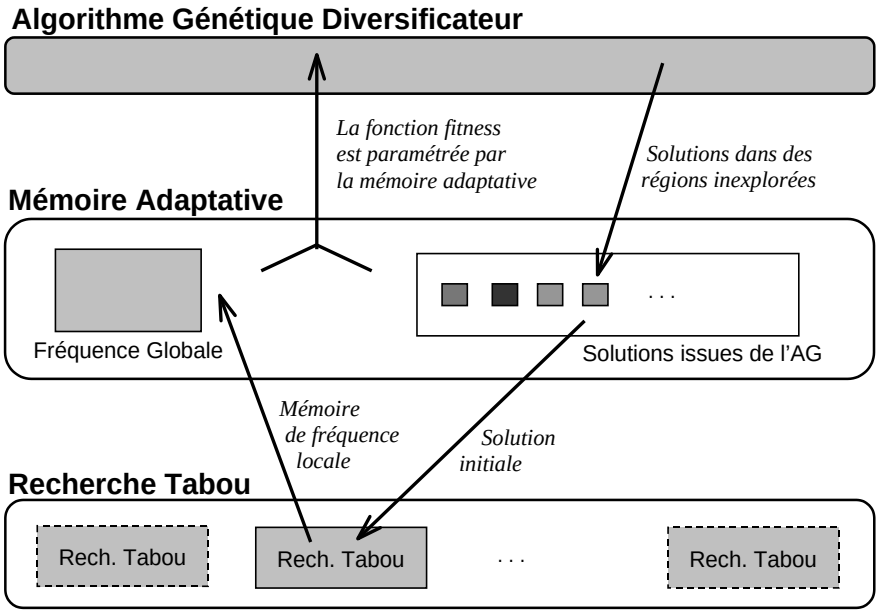

FIG. 1 – *Principe de notre algorithme hybride. Des tabous et un algorithme génétique s'exécutent concurremment. La mémoire adaptative est le médium au travers duquel les algorithmes s'échangent l'information*

Plus précisément, l'algorithme génétique minimise la fonction objectif :

$$\varphi(u) = \alpha f(u) - g(u)$$

où α est un coefficient permettant d'équilibrer les influences de f et g. f est relative à la mémoire de fréquence, g à la mémoire de solutions initiales. u est un point de la population de l'algorithme génétique. f se définit de la manière suivante :

$$f(u) = \sum_{e \in H} 1(e, u) F_e$$

où H est l'ensemble des événements, F_e est la fréquence de l'événement e, $1(e, u)$ vaut 1 si l'événement e apparaît dans u, 0 sinon. g vaut :

$$g(u) = \min_{v \in S}(\text{dist}(u, v))$$

où S est la mémoire de points de départ, dist(u, v) la distance (dissimilarité) entre deux points u et v. Rappelons que cette distance doit être définie par rapport à l'opérateur utilisé dans les tabous, la distance entre deux points étant le nombre d'applications de l'opérateur qui sont nécessaires pour passer de u à v. Pour le QAP, nous utilisons donc la distance définie à la section 2.2.1. Cependant, le rôle de la fonction g étant d'évaluer la diversité d'une population, nous pouvons utiliser la notion d'entropie rencontrée plus haut et réécrire g sous la forme :

$$g(u) = \Gamma(S \cup \{u\})$$

4. Résultats expérimentaux

Cette partie présente les caractéristiques et les résultats de l'application de notre heuristique hybride au problème d'affectation quadratique. Afin d'évaluer le rôle de l'algorithme génétique, nous comparons l'hybride et le tabou multiple seul et discutons les résultats obtenus.

4.1. *Protocole expérimental*

Nous décrivons, ci-après, les caractéristiques des quatre algorithmes que nous avons utilisés pour notre étude. Sont détaillés, les deux briques de bases, (recherche tabou et algorithme génétique) et les algorithmes composés (méthode hybride et tabou multiple).

En ce qui concerne, la *méthode tabou*, l'algorithme que nous avons mis en place est un tabou simple, il n'intègre pas de technique de diversification ou d'intensification sophistiquées. Au démarrage, la longueur de la liste tabou est choisie aléatoirement entre $0,5\ n$ et $1,5\ n$ et demeure inchangée tout au long de la recherche. La fonction d'aspiration est classique, elle s'applique lorsqu'un mouvement conduit à une solution supérieure à la meilleure solution rencontrée depuis le début de la recherche. La recherche est arrêtée après un nombre donné d'itérations.

Pour *l'algorithme génétique* que nous avons réalisé, la population comprend n points qui sont générés uniformément au départ. Les opérateurs de mutation et de recombinaison sont, respectivement, la fonction $swap$ et l'opérateur PMX qui garantit

que le produit du croisement reste une permutation [Gre87]. A chaque génération, le $swap$ et la recombinaison sont appliqués en compétition sur le meilleur point de la population, le cross-over utilisant un autre point tiré au hasard dans la population. Si un point, supérieur en qualité au meilleur point, est trouvé, alors il remplace le plus mauvais point de la population. Cet algorithme génétique, où au plus un point de la population est remplacé à chaque génération, suit le modèle *steady-state* défini par Whitley [Whi88]. Pour produire une nouvelle solution, l'AG effectue n itérations pour optimiser la fonction φ suivante :

$$\varphi(u) = \sum_{e \in H} 1(e,u) F_e \; - \; \min_{v \in S}(\text{dist}(u,v))$$

Le *tabou multiple* implique 100 recherches tabous qui s'exécutent indépendamment. Chaque recherche tabou fait $50\ n$ itérations. Bien que les tabous fonctionnent simultanément sur notre plate-forme parallèle, le tabou multiple rendrait le même résultat si les tabous s'exécutaient en séquence, puisque ces derniers ne coopèrent pas entre eux.

Notre *méthode hybride* fédère 100 recherches tabous et un algorithme génétique qui coopèrent via la mémoire adaptative ; toutes les recherches étant exécutées en mode asynchrone. Au départ, chaque tabou démarre d'une solution générée aléatoirement selon une distribution uniforme. Quand un tabou se termine, il communique à la mémoire adaptative sa mémoire de fréquence locale et effectue une requête à l'algorithme génétique qui lui fournit une nouvelle solution à partir de laquelle il redémarre. Les autres tabous continue leur exécution pendant ce temps là. Chacun des 100 tabous démarre 10 fois pour $5\ n$ itérations.

4.2. *La plate-forme d'exécution*

La recherche tabou parallèle et l'algorithme hybride ont été développés sur l'environnement MARS (*Multi-user Adaptive Resource Scheduler*) [THG96, THG97]. MARS est un environnement de programmation parallèle sur architectures hétérogènes multi-utilisateurs. Il permet, d'une part, d'exploiter la sous-utilisation des machines (en tenant compte du caractère personnel des stations de travail), et d'autre part, de supporter le parallélisme adaptatif pour reconfigurer dynamiquement l'ensemble des processeurs supportant l'application parallèle en fonction de l'état de charge du système. L'application parallèle est calquée sur un modèle *maître/esclaves.* Le processus *maître* assure la gestion des esclaves et la coordination. Il contrôle la mémoire adaptative et l'algorithme génétique séquentiel. Les processus *esclaves* font une recherche tabou. Durant l'exécution, MARS *déplie* (crée de nouveaux processus esclaves) et *replie* (retire des processus esclaves) l'application selon que les machines sont disponibles ou réquisitionnées. Un mécanisme de sauvegarde/restauration a été mis en place pour se prémunir contre les pannes matérielles et logicielles, très fréquentes lors de longues exécutions sur de nombreuses machines. Pour cette étude, nous avons utilisé des machines hétérogènes : 126 stations de travail (PC-Linux, Sun4-Sunos, Alpha-OSF, Sun-Solaris) et une machine parallèle composée de 16 processeurs Alpha connectés entre eux par un réseau optique à haut débit.

4.3. *Les instances tests*

Les instances du QAP peuvent être classées en fonction de leur nature. On distingue les instances générées aléatoirement des instances du monde réel, les instances uniformes des instances plus structurées (non uniformes). L'indicateur de dominance, défini par Vollman et Buffa [VB66], bien qu'imparfait, permet d'aider à la classification des instances ; une dominance élevée correspond généralement à une instance structurée. Etant donné que les performances d'une méthode heuristique dépendent de l'instance à résoudre [Tai95, BPT96], nous avons évalué notre hybride coopératif sur des instances des différentes classes. Elles sont recensées dans le tableau 1.

Instance	Taille	Distances	Flux	Dominance
Tai25a	25	uniforme	uniforme	64
Tai100a	100	uniforme	uniforme	60
Nug30	30	grille	aléatoire	83
Sko64	64	grille	aléatoire	108
Els19	19	réel	réel	530
Bur26d	26	réel	réel	228
Tai35b	35	pseudo-réel	pseudo-réel	309
Tai100b	100	pseudo-réel	pseudo-réel	321
Ste36c	36	réel	réel	400
Tai64c	64	réel	réel	127
T256c	256	réel	réel	217

TAB. 1 – *Les instances tests que nous utilisons sont issues de la bibliothèque standard QAPLib [BKR91]. Pour chacune, nous indiquons sa taille et le type de distribution des distances et des flux et la valeur de la dominance*

4.4. *Les résultats*

Nous présentons, ici, le résultat des expérimentations réalisées au cours de notre étude (voir tableau 2). Pour ce faire, nous distinguons les instances de petite taille des instances de grande taille. Pour les instances de petite taille ($n \leq 50$), on ne constate pas de différence entre le tabou multiple et l'hybride diversificateur en ce qui concerne la qualité du meilleur point trouvé. En effet, les deux méthodes trouvent des points de qualité semblable, optima ou pseudo-optima pour la plupart des instances tests[3]. En revanche, pour les instances structurées, il apparaît nettement que l'hybride coopératif trouve, plus souvent que le multi-tabous un (pseudo-) optimum. En ce qui concerne les instances de grande taille, il faut en outre différencier les instances peu structurées des autres. Sur les instances faiblement structurées, le multi-tabous s'impose alors que l'hybride diversificateur fournit des points de meilleure qualité sur les instances structurées.

3. Un point qu'on croit optimal, sans que cela ait été montré

Instance	Solution QapLib	Hybride		Tabou Multiple	
		Ecart	Taux	Ecart	Taux
Tai25a	1 167 256	7.40 e-1	3.0	7.40 e-1	4.0
Tai100a	21 125 314	9.89 e-3	.33	6.09 e-3	.33
Nug30	6 124	0.00	2.3	0.00	4.7
Sko64	48 498	2.23 e-3	.33	3.71 e-4	.33
Els19	17 212 548	0.00	18.	0.00	6.3
Bur26d	3 821 255	0.00	7.0	0.00	.33
Tai35b	283 315 445	0.00	1.0	0.00	2.3
Tai100b	1 185 996 137	4.47 e-3	.33	5.45 e-3	.33
Ste36c	8 239 110	0.00	.60	0.00	.60
Tai64c	1 855 928	0.00	51.	0.00	7.3
Tai256c	44 759 294	2.28 e-3	.33	2.36 e-3	.33

TAB. 2 – *Comparaison entre l'hybride diversificateur et le tabou multiple sur un panel d'instances standard. Pour chacune des deux heuristiques sont donnés, pour trois « runs », l'écart relatif de la meilleure solution trouvée par rapport à la meilleure solution connue et le pourcentage de tabous qui l'ont trouvée. Les paramètres des deux métaheuristiques sont positionnés de manière à rendre les temps d'exécution semblables (sur des plate-forme matérielles identiques)*

Ainsi, l'analyse des résultats des évaluations du multi-tabous et de l'hybride diversificateur sur le QAP, montre la supériorité de l'algorithme hybride coopératif dans la résolution d'instances structurées. En effet, l'hybride parvient à exploiter la structure des instances et s'impose face à la recherche tabou. Par ailleurs, les résultats obtenus mettent en évidence la faiblesse de l'hybride vis à vis du multi-tabous pour les autres instances, plus uniformes. Toutefois, cette supériorité de la recherche tabou n'est pas surprenante. En effet, les instances uniformes sont peu structurées et, par là même, tout indicateur heuristique demeure quasiment identique, quelle que soit la région considérée dans l'espace de recherche. Ainsi, l'application d'un algorithme, basé sur la recherche de régions inexplorées, ne peut pas être performant sur des instances de nature uniforme.

5. Conclusion et perspectives

Dans cette étude, nous avons illustré un aspect souvent méconnu de la puissance du paraléllisme. Certes nous avons profité de la « force brutale » de notre plate-forme d'éxécution qui est fortement parallèle ; mais surtout, la méthode que nous proposons tient sa supériorité de la collaboration (parallélisme) de plusieurs méthodes. Ainsi, pour notre hybride, la coopération de plusieurs algorithmes de recherche augmente la probabilité de trouver un optimum de bonne qualité, au moins sur des instances structurées, autrement dit, sur les instances qui présentent le plus grand intérêt pratique du fait des applications réelles sous-jacentes. En ce qui concerne la validation de notre

heuristique hybride comme méthode de résolution performante, nous poursuivons, aujourd'hui, notre expérimentation sur de nouvelles instances réelles de grande taille pour affiner le paramétrage de notre algorithme.

Cependant, malgré ses performances satisfaisantes, nous sommes loin de clamer que notre algorithme est une panacée. Plutôt, nous insistons sur le point qu'il faut clairement expliciter les problèmes sur lesquels l'hybride coopératif s'impose face à des algorithmes plus simples. En effet, l'idée de diversifier les points de départ n'a d'intérêt que si le paysage de l'instance à résoudre est composé de régions disparates. Si ce n'est pas le cas, un multi-tabous est suffisant car il n'apporte rien d'ajouter un organe de communications entre les tabous qui cherchent dans différentes régions. Aussi, c'est l'occasion de souligner l'importance des travaux sur les paysages adaptatifs et l'intérêt de leur apport dans la compréhension du fonctionnement des heuristiques en optimisation combinatoire.

Références

[AL97] E. AARTS, J.K. LENSTRA. *Local search in combinatorial optimization.* John Wiley and Sons editor, 1997.

[BKR91] R.E. BURKARD, S. KARISCH, F. RENDL. Qaplib: A quadratic assignment problem library. *European Journal of Operational Research*, 55:115–119, 1991.

[BPT96] V. BACHELET, P. PREUX, E-G. TALBI. Parallel hybrid meta-heuristics : application to the quadratic assignment problem. pages 233–242, Versailles, France, March 1996. Parallel Optimization Colloquium POC96.

[DG97] M. DORIGO, L. GAMBARDELLA. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. on Evolutionary Computation*, 1(1), 1997.

[FF94] C. FLEURENT, J. FERLAND. Genetic hybrids for the quadratic assignment problem. *DIMACS Series in discrete Mathematics and Theoretical Computer Science*, 16:173–188, 1994.

[FF95] C. FLEURENT, J. FERLAND. Algorithmes génétiques hybrides pour l'optimisation combinatoire. *RAIRO*, 1995.

[FRP97a] C. FONLUPT, D. ROBILLIARD, P. PREUX. A comparison of the 2-opt-move and the city-swap operators for the TSP. In *Proc. Evolution Artificielle*, Nimes, France, October 1997.

[FRP97b] C. FONLUPT, D. ROBILLIARD, P. PREUX. Preventing premature convergence via cooperating genetic algorithms. In *Proc. Mandel 97*, pages 50–55, Brnö, Tchéquie, June 1997.

[FRPT97] C. FONLUPT, D. ROBILLIARD, P. PREUX, E-G. TALBI. Fitness landscape and performance of meta-heuristics. In *Proc. Meta-Heuristics'97 (MIC'97)*, Sophia-Antipolis, France, July 1997.

[GJ79] M. GAREY, D. JOHNSON. *Computers and Intractability: A guide to the theory on NP-completeness*. W.H. Freeman and Co. Publishers, New York, 1979.

[Glo95] F. GLOVER. Tabu search. In [Ree95], chapter 3, pages 70–150. 1995.

[Gre87] J.J. GREFENSTETTE. Incorporating problem specific knowledge into genetic algorithms. In L. Davis, editor, *Genetic algorithms and Simulated annealing*, Research Notes in Artificial Intelligence, pages 42–60, San Mateo, CA, USA, 1987. Morgan Kaufmann.

[Mit96] M. MITCHELL. *An Introduction to Genetic Algorithms*. MIT Press, A Bradford Book, 1996.

[MSK95] W.G. MACREADY, A.G. SIAPAS, S.A. KAUFFMAN. Criticality and parallelism in combinatorial optimization. *Science*, 1995.

[Pre95] P. PREUX. Les algorithmes évolutifs. Technical Report LIL-94-1, Laboratoire d'Informatique du Littoral, Calais, France, Septembre 1995.

[PRW94] P.M. PARDALOS, F. RENDL, H. WOLKOWICZ. The quadratic assignment problem: A survey and recent developments. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 16:1–42, 1994.

[PT97] P. PREUX, E-G. TALBI. Towards hybrid evolutionary algorithms. Technical Report LIL-97-15, Laboratoire d'Informatique du Littoral, 1997. (soumis).

[Ree95] COLIN R. REEVES, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Advanced Topics in Computer Science. Mc Graw-Hill, 1995.

[Tai95] E. TAILLARD. Comparison of iterative searches for the quadratic assignment problem. *Location Science*, 3:87–133, 1995.

[THG96] E-G. TALBI, Z. HAFIDI, J-M. GEIB. Mars : Adaptive scheduling of parallel applications in a multi-user heterogeneous environment. pages 119–122, Alpe d'Huez, France, April 1996. 2nd European School on Computer Science ESPPE'96.

[THG97] E-G. TALBI, Z. HAFIDI, J-M GEIB. Parallel adaptative tabu search for large scale optimization problems. Sophia Antipolis, France, july 1997. 2nd Metaheuristics International Conference MIC'97.

[VA96] M.G.A. VERHOEVEN AND E.H.L. AARTS. Parallel local search. *Journal of Heuristics*, 1(1):43–66, 1996.

[VB66] T.E. VOLLMAN, E.S. BUFFA. *The Facilities Layout Problem in Perspective*, volume 12, pages 450–468. Management Science, June 1966.

[Whi88] D. WHITLEY. GENITOR: A different genetic algorithm. In *Proc. of the Rocky Mountain Conference on Artificial Intelligence*, Denver, CO, USA, 1988.