N° d'ordre : 72/2019-C/MT

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et Technologie Houari Boumediene

Faculté des Mathématiques

Thèse de Doctorat

Présentée pour l'obtention du grade de Docteur

En : Mathématiques

Spécialité : Recherche Opérationnelle

Par : Youcef DJEDDI

Sujet

Algorithmes du Problème de la Clique Maximum de Graphes et Applications aux réseaux biologiques

Soutenue publiquement le 25/09/2019, devant le jury composé de :

M. A. Berrachedi	Professeur	à l'U.S.T.H.B d'Alger	Président
M. H. Ait Haddadene	Professeur	à l'U.S.T.H.B d'Alger	Directeur de thèse
M. N. Belacel	Professeur	au C.N.R.C. & Univ. Moncton de Canada	Co-Directeur de thèse
M. M. A. Boutiche	Maître de conférences/A	à l'U.S.T.H.B d'Alger	Examinateur
M. B. Sadi	Maître de conférences/A	à l'U.M.M.T.O de Tizi Ouzou	Examinateur

Table des matières

Liste des Algorithmes

Liste des tableaux

Table des figures

Remerciements

En premier lieu, j'exprime mes profonds remerciements à monsieur **Hacène Ait Haddadene** professeur à l'USTHB et à monsieur **Nabil Belacel** chercheur principal au conseil national de recherches Canada (CNRC) et professeur à l'université de Moncton de Canada, mes directeurs de recherche pour la confiance qu'ils m'ont faite en acceptant de diriger mon travail et pour le temps précieux qu'ils m'ont consacré malgré leurs nombreuses occupations et pour leurs précieux conseils, soutiens et orientations tout au long de ce travail.

Je remercie monsieur **Abdelhafid Berrachedi** professeur à l'USTHB pour m'avoir fait l'honneur en acceptant de présider le jury.

Je remercie également monsieur **Mohamed Amine Boutiche** maître de conférences à l'USTHB et monsieur **Bachir Sadi** maître de conférences à l'UMMTO de Tizi Ouzou pour avoir acceptés d'examiner ce travail et qui ont pris la peine de lire avec soin cette thèse pour évaluer son contenu.

Je remercie mes parents et frères pour m'avoir soutenu, et ma fiancée pour avoir su m'écouter et me redonner confiance.

Je remercie monsieur **Madani Bezoui** enseignant à l'UMBB de Boumerdes et monsieur **Salah Eddine Lakhdari** pour leur aide et soutien.

Enfin, Merci à tous ceux qui m'ont aidé de près ou de loin et à tous ceux qui m'ont incité à faire mieux, veuillez trouver ici le témoignage de ma très profonde gratitude.

Dédicaces

A mes parents,

A mes frères ,

A ma fiancée,

A toute ma famille,

A tous mes amis.

Introduction générale

La recherche opérationnelle avec toutes ses branches ont connu un développement énorme dans les dernières décennies. L'optimisation est parmi les branches de recherche opérationnelle qui ont été développées avec un rythme étonnant. L'optimisation combinatoire est domaine d'optimisation très vaste où l'ensemble des solutions est fini mais généralement très grand. Les problèmes d'optimisation combinatoire sont considérés comme un domaine de recherche fondamentale très important dans nos jours. Ils ont montré un grand succès dans la résolution des problèmes de la vie réelle pratiquement dans tous les domaines comprenant finance, industrie, télécommunications, transport, gestion de l'énergie, biochimie, biomédecine...

Le problème de la clique maximum est un problème d'optimisation combinatoire NP-complet dans le cas général, considéré comme un sujet de recherche actuel en théorie des graphes. Avec une large application dans de nombreux domaines comprenant les réseaux biologiques, génie biomédical, bio-informatique et chimio-informatique, la théorie de la classification, les réseaux sociaux et l'économie. Les réseaux biologiques représentent les systèmes biologiques complexes sous la forme d'un graphe, ils sont utilisés pour comprendre les fonctions moléculaires et des processus biologiques.

Un grand nombre de travaux de recherche sont dédiés à la résolution du problème de la clique avec des algorithmes exacts ou des heuristiques. Cette thèse est consacrée à présenter l'état actuel des méthodes de résolution du problème de la clique maximum, ses généralisations et relaxations, et au développement d'approches efficaces exactes ou heuristiques pour le problème de la clique maximum, ses généralisations et relaxations, et à l'application des approches heuristiques développées aux réseaux biologiques.

Objectifs

- ◇ Le premier objectif de cette thèse est de faire une étude de l'état de l'art des méthodes de résolution du problème de la clique maximum, ses généralisations et relaxations ; et de proposer des approches (exactes ou heuristiques) très efficaces pour le problème de la clique maximum, ses généralisations et relaxations : le problème de la clique de poids maximum des sommets, le problème de la clique de poids maximum des arêtes, le problème de la quasi-clique maximum et le problème du sous-graphe de poids maximum des arêtes.
- ◇ Le deuxième objectif de cette thèse est l'application des approches développées aux réseaux biologiques en général et dans les réseaux d'interactions protéine-protéine en particulier.

Contributions

Les contributions principales de cette thèse sont :

Une extension de la recherche tabou adaptative multistart pour le problème de la quasi-clique maximum : nous avons proposé un algorithme de recherche tabou appelé TSQC, pour la résolution du problème de la quasi-clique maximum. TSQC est une extension de l'algorithme de recherche tabou adaptative multistart (AMTS) proposé par Wu & Hao (2013) [124] pour la résolution du problème de la clique maximum. Notre algorithme TSQC est le premier algorithme de recherche tabou utilisé pour résoudre le problème de la quasi-clique maximum. Nous avons comparé les résultats de TSQC avec les méthodes exactes et les heuristiques les plus récentes. Les résultats des calculs révèlent que l'algorithme TSQC atteint les plus grandes quasi-cliques connues dans un temps raisonnable sur les réseaux réels et qu'il surperforme les méthodes existantes sur les instances DIMACS et BHOSLIB. De plus, notre algorithme donne de nouveaux résultats avec une qualité améliorée pour le MQCP. Un papier décrivant cet algorithme est publié dans la revue Computers & Industrial Engineering [30].

Une heuristique hybride et sa version parallèle pour le problème de la clique maximum : nous avons proposé une heuristique hybride (HHa) et sa version parallèle (une heuristique hybride parallèle (PHa)) pour le problème de la clique maximum. Ces heuristiques sont obtenues par une hybridation entre un algorithme exact et l'algorithme de recherche tabou adaptative multistart (AMTS). Des expériences numériques ont montré que nos algorithmes atteignent les meilleurs résultats connus pour les instances de référence DIMACS avec un temps d'exécution compétitif lorsque nous les comparons avec les résultats de l'algorithme AMTS. Une première version de ce travail a été acceptée dans la conférence : The 2018 International

Conference of Applied and Engineering Mathematics (WCE 2018).

Un algorithme de recherche tabou multistart et multivoisinage pour le problème de la clique de poids maximum des arêtes : nous avons proposé une heuristique de recherche tabou multistart et multivoisinage (MRNTS) pour le problème de la clique de poids maximum des arêtes. Notre algorithme MRNTS utilise quatre voisinages pour explorer l'espace de recherche et il utilise une stratégie de redémarrage (restart) pour éviter les optimums locaux. La méthode proposée est évaluée sur les instances de référence DIMACS et BHOSLIB et sur quelques réseaux biologiques. Les résultats comparatifs ont montré que MRNTS surperforme l'heuristique la plus récente du problème de la clique de poids maximum des arêtes sur 26 instances DIMACS parmi 32, sur 30 instances BHOSLIB parmi 30 et sur 12 réseaux biologiques parmi 12.

Résolution du problème du sous-graphe de poids maximum des arêtes dans des réseaux biologiques : nous avons proposé un algorithme de recherche tabou multistart pour le problème du sous-graphe de poids maximum des arêtes (MEWS). L'algorithme proposé utilise un voisinage contraint et deux listes tabous et une stratégie de vérification de la configuration qui est très utilisée dans les algorithmes de recherche locale pour éviter les optimums locaux. L'algorithme proposé est utilisé pour résoudre le MEWS dans des réseaux biologique, qui ont des vrais poids sur les arêtes. Les résultats ont montré que notre algorithme surperforme les autres heuristiques de l'état de l'art. Un papier décrivant l'algorithme proposé et ses applications est accepté au Colloque sur l'Optimisation et les Systèmes d'Information (COSI 2019).

Extraction de la plus grande Quasi-Clique à partir d'un réseau d'interactions protéine-protéine humaine : nous utilisons l'algorithme recherche tabou (TSQC) pour détecter la plus grande quasi-clique dans un réseau d'interactions protéine-protéine humaine. Une quasi-clique peut représenter un module dans un réseau d'interactions protéine-protéine, un module est défini comme un ensemble des protéines qui sont connectées pour réaliser une fonction biologique. Les résultats ont montré que l'algorithme TSQC peut être utilisé pour la détection des modules fonctionnels avec la plus grande taille et que TSQC donne des résultats prometteurs.

Organisation de la thèse

Chapitre 1 : Dans ce chapitre, nous commençons par définir le problème de la clique maximum et les problèmes équivalents. Puis nous donnons un aperçu de l'état de l'art des méthodes de résolution du problème de la clique. Et enfin nous définissons les généralisations

et relaxations du problème de la clique et nous donnons un aperçu de l'état de l'art de leurs méthodes de résolution.

Chapitre 2 : Il est consacré pour présenter la première contribution majeure de cette thèse i.e., une extension de la recherche tabou adaptative multistart pour le problème de la quasi-clique maximum. Nous avons aussi donné une évaluation de la méthode proposée et une étude comparative.

Chapitre 3 : Dans ce chapitre, nous avons présenté les deux méthodes proposées pour le problème de la clique maximum, l'heuristique hybride et l'heuristique hybride parallèle. À la fin de ce chapitre, nous avons donné une évaluation numérique de la performance des heuristiques proposées.

Chapitre 4 : Dans ce chapitre, nous nous sommes intéressés au problème de la clique de poids maximum des arêtes, nous avons proposé un algorithme de recherche tabou multistart et multivoisinage. Nous avons décrit l'algorithme proposé et nous avons donné une étude comparative de l'algorithme proposé et l'heuristique la plus récente de l'état de l'art.

Chapitre 5 : Il est dédié au problème du sous-graphe de poids maximum des arêtes. Dans l'introduction, nous avons défini le problème et montré l'importance du problème dans les réseaux biologiques. Après, une nouvelle méthode de recherche tabou multistart est proposée. Enfin, la méthode proposée est utilisée pour résoudre le problème du sous-graphe de poids maximum des arêtes dans des réseaux biologiques avec des poids originaux sur les arêtes. Et une comparaison avec les méthodes de la littérature est donnée.

Chapitre 6 : Il est dévoué aux réseaux d'interactions protéine-protéine. Nous avons donné une introduction aux réseaux d'interactions protéine-protéine (PPI). Puis, nous avons utilisé l'algorithme TSQC proposé dans le chapitre 2 pour chercher la plus grande quasi-clique et d'autres quasi-cliques de différentes tailles dans le réseau PPI de la base de données de référence sur les protéines humaines (HPRD, 2009). Et nous avons présenté et analysé les résultats.

Chapitre 1

Problème de la clique maximum

Le problème de la clique maximum (the maximum clique problem (MCP) en anglais) est l'un des problèmes d'optimisation combinatoire les plus connus. Le MCP est un des 21 problèmes NP-complet présenté par Karp [55]. Le MCP a des nombreuses applications pratiques dans des différents domaines tel que l'économie, l'infrastructure de transmission sans fil, l'analyse de la transmission des signaux, l'ordonnancement, génie biomédical, bio-informatique et chimio-informatique et les réseaux sociaux. Dans ce chapitre, nous présentons un état de l'art du problème de la clique maximum, ses généralisations et relaxations.

Sommaire

1.1 Quelques notions de base de la théorie des graphes

Définition 1.1.1. (Définition d'un graphe) :

Un graphe est composé d'un couple (V, E) de deux ensembles disjoints :

- L'ensemble V est un ensemble fini de points, les éléments de l'ensemble V sont appelés les sommets (vertices en anglais) du graphe G.
- L'ensemble E ($E \subseteq V \times V$) est un ensemble fini de lignes, les éléments de l'ensemble E sont appelés les arêtes (edges en anglais) du graphe G. Chaque arête $e = \{u, v\}$ (notée aussi par uv) relie deux sommets u et v appelés ses extrémités. Si les deux extrémités d'une arête sont égales, on dit que l'arête est une boucle.

On dit qu'un graphe est simple (graphe simple) si c'est un graphe sans boucle (chaque deux sommets sont reliés par au plus une arête). Un exemple d'un graphe est donné dans la figure 1.1.

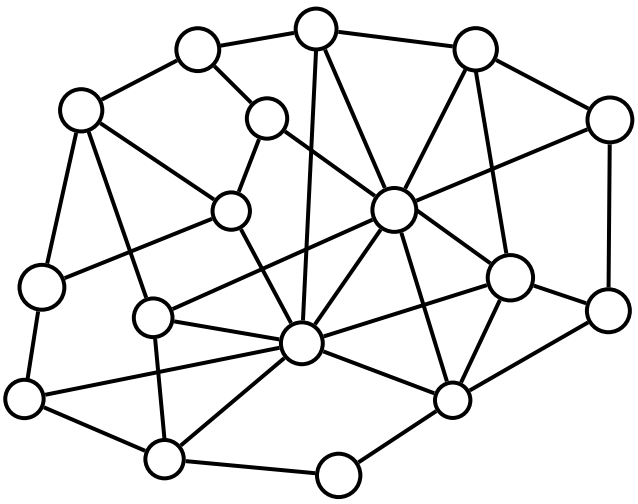


FIGURE 1.1 – Exemple d'un graphe

Définition 1.1.2. :

Si u est une extrémité d'une arête e ($e = uv$), on dit que u et e sont incidents. On dit que deux sommets u et v sont adjacents s'il sont reliés par une arête et on dit que u est voisin de v (v est voisin de u). L'ensemble des voisins d'un sommet u est appelé le voisinage (neighbourhood en anglais) de u, noté par $N(v)$. Le nombre de voisins d'un sommet v représente le degré $d(v)$ de sommet v i.e. $d(v) = |N(v)|$.

Définition 1.1.3. (Définition d'un sous-graphe) :

Soit $G(V, E)$ et $G'(V', E')$ deux graphes. G' est un sous-graphe (sub-graph en anglais) de G si les sommets de G' sont des sommets de G ($V' \subseteq V$) et les arêtes de G' sont des arêtes de G ($E' \subseteq E$). On dit que G' est un sous-graphe induit (induced sub-graph en anglais) de G si

$V' \subseteq V$ et pour tous les couples de sommets $(u, v) \in V' \times V'$ il existe une arêtes $uv \in E'$ si et seulement si $uv \in E$ (On dit aussi que G' est le sous-graphe de G induit par V', et on le note $G[V']$).

Définition 1.1.4. (Définition d'un graphe complémentaire) :

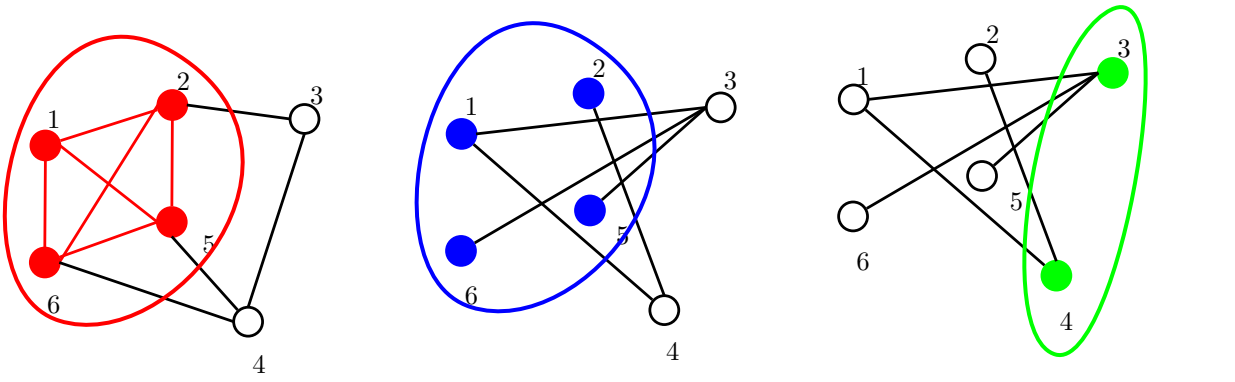
Soit $G(V, E)$ et $G'(V', E')$ deux graphes simples. G' est un graphe complémentaire (appelé aussi graphe inversé) du graphe G si G et G' ont les mêmes sommets ($V = V'$) et chaque deux sommets distincts de G' sont adjacents si et seulement s'ils ne sont pas adjacents dans G. Le graphe complémentaire du graphe G est noté par $\overline{G}$.

1.2 Définitions et problèmes équivalents au problème de la clique maximum

Soit $G = (V, E)$ un graphe avec un ensemble des sommets $V = \{1, ..., n\}$ et un ensemble des arêtes $E \subseteq V \times V$.

- ◇ Une clique C dans G est un sous-ensemble de V tel que tous les sommets de C sont adjacents i.e., chaque couple de sommets représente une arête ($\forall u, v \in C, uv \in E$). Le problème de la clique maximum vise à trouver la plus grande clique dans G, la taille de la clique maximum dans G est appelée le nombre de la clique maximum noté $\omega(G)$. Si une clique ne peut pas être contenue dans une autre clique, on dit que c'est une clique maximale.
- ◇ Un stable (appelé aussi l'ensemble indépendant) S dans G est un sous-ensemble de V tel que tous les sommets de S ne sont pas adjacents i.e., $\forall u, v \in S, uv \notin E$ (le stable est le complémentaire de la clique). Le problème du stable maximum (the maximum independent set problem (MIS) en anglais) cherche à trouver le plus grand stable dans G, la taille du stable maximum dans G est appelée le nombre de stabilité noté $\alpha(G)$.
- ◇ Le problème du transversal minimum connu aussi avec problème de couverture minimum par sommets (the minimum vertex cover problem (MVC) en anglais) vis à trouver un sous-ensemble T de V tel que toutes les arêtes du graphe G ont une extrémité dans T i.e., $\forall uv \in E, u \in T$ *ou* $v \in T$.

Le problème de la clique maximum est équivalent au problème du stable maximum et le problème de transversal minimum, qui sont parmi les problèmes d'optimisation combinatoire les plus connus.



Graphe G avec clique max C en rouge

Graphe complémentaire de G ($\overline{G}$) avec stable max S en bleu

$\overline{G}$ avec transversal min T en vert

FIGURE 1.2 – Illustration de la relation entre la clique maximum (rouge), le stable maximum (bleu) et le transversal minimum (vert).

Soit G un graphe et $\overline{G}$ son graphe complémentaire, la Figure 1.2 représente un exemple qui montre la relation entre la clique maximum, le stable maximum et le transversal minimum. Il est facile de voir que $C = \{1, 2, 5, 6\}$ est une clique maximum dans G si seulement si $S = C = \{1, 2, 5, 6\}$ est un stable maximum dans $\overline{G}$ et seulement si $T = \{3, 4\}$ est un transversal minimum dans $\overline{G}$. Vu la relation forte entre le problème de la clique maximum et le problème du stable maximum, nous nous intéressons aux algorithmes des deux problèmes.

1.3 Formulations et complexité

Dans la littérature, plusieurs travaux et études sont faits pour la formulation (modélisation) du problème de la clique maximum. Dans cette section, nous donnons les formules les plus utilisées et les plus générales, pour une étude complète et détaillée sur les formulations du problème de la clique maximum voir Pardalos & Xue (1994) [86], Bomze *et al.* (1999) [14], Butenko (2003) [23] et Wu & Hoa (2015) [126].

La formulation la plus simple est donnée dans le modèle mathématique avec variables binaires suivant :

$$\begin{cases} Max & \sum_{i=1}^{n} x_i \\ S.c & x_i + x_j \leq 1, \forall \{i, j\} \in \overline{E} \\ & x_i \in \{0, 1\}, i = 1, ..., n. \end{cases}$$

La variable x_i représente la situation du sommet i i.e., $x_i = 1$ si le sommet i est dans la clique C et $x_i = 0$ si le sommet i n'est pas dans la clique C où C est une clique définie par une solution réalisable.

Une deuxième formulation basée sur l'ensemble de tous les stables maximaux de G est donnée dans le modèle mathématique ci-dessous. Cette formulation exige que tout sommet d'une clique C de G ne peut être contenu que dans un seul stable maximal de G :

$$\begin{cases} Max & \sum_{i=1}^{n} x_i \\ S.c & \sum_{i \in s} x_i \leq 1, \forall s \in S \\ & x_i \in \{0,1\}, i = 1, ..., n. \end{cases}$$

Où S est l'ensemble de tous les stables maximaux du graphe G (On dit qu'un stable est maximal s'il n'est pas contenu dans un autre stable).

Le problème de la clique maximum est un problème NP-difficile Gary & Johnson (1979) [38] et sa version de décision est un des 21 premiers problèmes NP-Complets présentés par Karp (1972) [55]. Dans la littérature, nombreux travaux sont traités pour étudier la complexité du problème de la clique maximum et ses relaxations et généralisations. Ces études incluent des analyses de la complexité des algorithmes exponentiels, la complexité paramétrique et des études des approximations du problème de la clique maximum. Pour une examination détaillée des résultats de la complexité du problème de la clique et ces généralisations voir Pardalos & Xue (1994) [86], Bomze *et al.* (1999) [14] et Wu & Hoa (2015) [126].

1.4 Les méthodes de résolution

Vu l'importance du problème de la clique maximum et le problème du stable maximum, on trouve de nombreux travaux sur le MCP et MIS dans la littérature. On cite trois travaux sur l'état de l'art du MCP Pardalos & Xue (1994) [86], Bomze *et al.* (1999) [14] et Wu & Hoa (2015) [126]. Dans cette section, nous donnons un aperçu sur les méthodes de résolutions exactes et heuristiques (Section 1.4.1 et 1.4.2 respectivement) les plus connues pour le problème de la clique maximum et le problème du stable maximum.

1.4.1 Les méthodes exactes

Dans cette section, nous donnons un aperçu sur l'état de l'art des méthodes exactes du problème de la clique maximum. La plupart des méthodes exactes trouvées dans la littérature sont basées sur le principe général de l'algorithme Branch and Bound (B&B). Les différences entre eux se trouvent dans les techniques de détermination des bornes (supérieures et inférieures) et dans les techniques de branchement.

Comme est noté dans Pardalos & Xue (1994) [86], Bomze *et al.* (1999) [14], le premier algorithme d'énumération de toutes les cliques dans un graphe arbitraire est probablement proposé par Harary & Ross (1957) [45] en 1957. Dans les années soixante-dix (1970), Desler

& Hakimi (1970) [28], Houck (1974) [49] and Tarjan (1972) [114] ont proposé les premiers algorithmes d'énumération implicite pour le MCP et MIS. Ces algorithmes ont été améliorés par Tarjan & Trojanowski (1977) [115] et Houck & Vemuganti (1977) [50] en 1977, Tarjan & Trojanowski (1977) [115] ont proposé un algorithme récursif pour le problème du stable maximum avec une complexité $O(n^{\frac{n}{3}})$.

En 1986, Robson [97] a proposé une version améliorée de l'algorithme récursif proposé par Tarjan & Trojanowski (1977) [115] avec une complexité $O(n^{0.276n})$. Dans la même année, Balas & Yu (1986) [5] ont aussi proposé un algorithme d'énumération implicite avec une nouvelle manière, cet algorithme est l'un des plus importants algorithmes d'énumération implicite du problème de la clique maximum, il peut trouver la solution optimale dans des graphes avec jusqu'à 400 sommets et 30 000 arêtes.

En 1990, Carraghan & Pardalos [26] ont proposé un algorithme Branch and Bound (appelé CP), CP est l'algorithme le plus important et connu. CP est un algorithme récursif très simple, qui est la base de plusieurs algorithmes B&B améliorés proposés par la suite. L'algorithme CP utilise deux ensembles critiques : l'ensemble C appelé la solution courante ou clique (la clique en construction) et l'ensemble P appelé l'ensemble des sommets candidats ou l'ensemble des sommets candidatures, P est un sous-ensemble de $V \setminus C$ tel que $P = \{v \in V \setminus C, \forall u \in C, uv \in E\}$. Dans la même année, Babel Tinhofer (1990) [4] ont aussi proposé un algorithme Branch and Bound (appelé l'algorithme BT). L'algorithme BT utilise une heuristique de coloration appelée DSATUR (proposé par Brélaz (1979) [19]) pour trouver une borne supérieure chaque étape.

En 2002, Östergård [82] a proposé un algorithme amélioré de l'algorithme CP appelé Cliquer ; Cliquer utilise un ordre de sommets défini par la coloration des sommets et une nouvelle stratégie d'élagage. Östergård a comparé son algorithme avec plusieurs autres algorithmes sur des instances DIMACS (DIMACS benchmark) et sur des graphes aléatoires, il a affirmé que la performance de son algorithme surperforme celles des autres. Dans la même année, Fahle (2002) [31] a aussi proposé un algorithme amélioré de l'algorithme CP appelé χ+DF, Fahle a introduit des techniques de filtrage de l'ensemble candidat P (supprime certains sommets qui ne peuvent pas élargir la solution (clique) courante), il a présenté aussi des bornes basées sur la coloration.

En 2003, Régin [96] a proposé une approche de programmation par contraintes pour déterminer une borne supérieure de la taille de la clique maximum. Régin [96] a proposé une nouvelle stratégie pour classer les sommets qui peuvent élargir la solution courante, son algorithme a pu résoudre sept instances DIMACS pour la première fois. Aussi en 2003, Tomita & Seki (2003) [117] ont proposé un algorithme Branch and Bound appelé MCQ, qui a été amélioré par plusieurs algorithmes par la suite. MCQ utilise une coloration des sommets basée sur un ordre

prédéterminé des sommets pour trouver une borne supérieure de la taille de la clique maximum ($\omega(G)$). Les auteurs ont montré que leur algorithme (MCQ) peut résoudre le MCP dans des graphes aléatoires et DIMACS instances de tailles jusqu'à 15 000 sommets.

En 2007, Konc & Janežič [60] et Tomita & Kameda (2007) [116] ont proposé deux algorithmes Branch and Bound appelés MaxCliqueDyn et MCR respectivement, ces deux algorithmes sont des versions améliorés de l'algorithme MCQ. Konc & Janežič (2007) [60] ont décrit un algorithme approximative de coloration et utilisé cet algorithme pour trouver une borne de $\omega(G)$ dans l'algorithme MaxCliqueDyn. Les auteurs de l'algorithme MCR (Tomita & Kameda (2007) [116]) ont amélioré MCQ en introduisant un meilleur ordre initial des sommets. L'algorithme MaxCliqueDyn est plus rapide que MCQ dans les graphes denses.

En 2010, Li & Quan [65] ont proposé un nouveau codage du problème de la clique maximum dans le problème de satisfiabilité maximum (MaxSAT) et utilisé la technologie de MaxSAT pour trouver une borne supérieure basée sur une partition de P. Tomita *et al.*(2010) [118] ont aussi proposé un algorithme B&B appelé MCS, MCS est une amélioration de l'algorithme MCR (encore une amélioration de l'algorithme MCQ) en améliorant la procédure de coloration par une stratégie de recoloration.

Une année après (2011), San Segundo *et al.* [104] ont proposé un algorithme Branch and Bound bit-parallèle appelé BB-MaxClique. BB-MaxClique utilise une nouvelle règle de branchement global basée sur une approche de coloration amélioré pour trouver des bornes supérieures plus serrées à l'ensemble candidat P.

En 2013, Li *et al.* [64] ont proposé une combinaison de raisonnement de MaxSAT et une borne supérieure incrémentale dans un algorithme B&B appelé IncMaxCLQ. La borne est calculée en temps linéaire à partir des bornes supérieures pour les sous-graphes organisés en dégénérescence statique. Aussi en 2013, San Segundo *et al.* [103] ont effectué des améliorations sur l'algorithme BB-MaxClique et proposé un algorithme appelé BBMCI, ils ont utilisé la stratégie de recoloration proposée par Tomita *et al.*(2010) [118] ainsi que différentes stratégies d'optimisation pour un balayage rapide des bits.

En 2014, Maslov *et al.* [75] ont proposé une amélioration des deux algorithmes Branch and Bound MCS (Tomita *et al.*(2010) [118]) et MaxSAT (Li & Quan (2010) [65]), ils ont utilisé une heuristique appelée ILS (proposé par Andrade *et al.*(2010) [2]) pour trouver une solution initiale, ensuite utilisée cette solution comme borne inférieure dans les deux algorithmes B&B. Les algorithmes nouveaux améliorés sont plus rapides aux algorithmes basiques (MCS et MaxSAT). Batsyn *et al.*(2014) [8] ont aussi proposé une amélioration de l'algorithme MCS dans la même année (2014), ils ont aussi ajouté l'heuristique ILS pour trouver une solution initiale et utilisé cette solution comme une borne inférieure, mais ils ont ajouté une technique basée sur la coloration pour une détection rapide des sommets de la clique dans l'ensemble candidat

P.

En 2016, San Segundo *et al.* [102] ont proposé un algorithme Branch and Bound appelé BBMCSP pour les graphes creux (Sparse graph en anglais) de grande taille. BBMCSP est basé sur un solveur non-creux bit-parallèle et utilise un nouveau code parcimonieux par la matrice d'adjacente. Dans la même année, Tomita *et al.*(2016) [119] ont proposé une amélioration de l'algorithme MCS (Tomita *et al.*(2010) [118]), le nouvel algorithme est appelé MCT. Ils ont utilisé un autre algorithme (appelé $k-optlocalsearch$ proposé par Katayama *et al.*(2005) [56]) plus rapide pour trouver une borne inférieure initiale. Leurs tests numériques ont monté que MCT est plus rapide que MCS.

Une année après (en 2017), San Segundo *et al.* [101] ont proposé un algorithme Branch and Bound appelé BBMCW pour les graphes creux de grande taille, BBMCW peut être codé directement en mémoire principal (RAM) sans pénaliser la performance. BBMCW est basé sur l'algorithme BB-MaxClique.

Pour une étude détaillée sur les méthodes exactes proposées avant 1999 voir Pardalos & Xue (1994) [86] et Bomze *et al.* (1999) [14], et pour les méthodes proposées avant 2015 voir Wu & Hoa (2015) [126]. Vous trouvez aussi dans Wu & Hoa (2015) [126] une étude comparative sur les méthodes exactes avec des tests numériques.

1.4.2 Les heuristiques

Rappelons que le problème de la clique maximum est un problème NP-Complet, d'où la recherche de la solution optimale avec les méthodes exactes prend beaucoup de temps sur les graphes de grande taille (plus la taille s'agrandit plus l'exécution des méthodes exactes prendra plus de temps). Si pour cela plusieurs chercheurs ont dirigé leurs travaux ver les heuristiques. Les heuristiques sont plus rapides que les méthodes exactes et ils ont montré une grande efficacité pour le problème de la clique maximum et le stable maximum. Dans cette section, nous donnons un aperçu sur les heuristiques du problème de la clique maximum et du stable maximum. Dans la littérature, on trouve de nombreuses heuristiques proposées pour le MCP et le MIS, ces heuristiques comprenant des algorithmes gloutons, recherche locale, recherche tabou, recherche à voisinage variable, des algorithmes évolutionnaires, ... etc.

En 1989, Friden *et al.* [35] ont proposé un algorithme de recherche tabou appelé STABULUS pour le problème du stable maximum, qui est la première heuristique de recherche tabou. STABULUS est basé sur la stratégie de pénalité k-fixé et cherche à trouver un stable légal de taille k i.e., il commence avec un sous ensemble S de taille k (stable illégal) et cherche à minimiser le nombre des arêtes dans S jusqu'à zeros (S devenir légal stable ou stable). Et cela, se fait en changent un sommet de S avec un sommet de $V \setminus S$. Friden *et al.* (1989) [35] ont noté que le problème de la clique maximum peut être approximativé par trouver une série des

k-clique (clique de taille k) en incrémentant k par 1 chaque fois. STABULUS a été amélioré en 1996 par Fleurent & Ferland (1989) [34].

En 1993, Gendreau *et al.* [39] ont proposé deux variantes de la méthode de recherche tabou pour le problème de la clique maximum. La première est déterministe : Soit C la solution courante (la clique) et EC l'ensemble candidat, $EC = \{v \in V \setminus C, uv \in E, \forall u \in C\}$ (EC a la même définition avec P dans la Section 1.4.1), s'il y a plusieurs sommets de EC qui peuvent être ajoutés à la solution courante (les sommets qui ne sont pas interdit par la liste tabou), le sommet de EC qui a le plus de voisin dans EC va être sélectionné. La deuxième variante est probabiliste : s'il y a plusieurs sommets de EC qui peuvent être ajoutés à la solution courante, un sommet va être choisi aléatoirement de EC.

En 2001, Jagota & Sanchis [51] ont proposé trois heuristiques gloutonnes aléatoires adaptatives multistart pour le problème de la clique maximum (appelées AI, AW et NA). Ces heuristiques utilisent une règle probabiliste gloutonne pour la sélection des sommets avec une stratégie multistart. Dans la même année (2001), Battiti & Protasi [9] ont proposé un algorithme de recherche locale Réactif pour le MCP appelé RLS. RLS commence par un sous-ensemble vide (clique vide) et il exploite l'espace de recherche en ajoutant un sommet à C ou supprimant un sommet de C à chaque itération. La section des sommets à ajouter ou à supprimer est déterministe (choisir le sommet qui a le plus de voisin dans EC). Les auteurs se sont inspirés de la recherche tabou et ils ont ajouté une liste tabou à RLS. RLS utilise aussi une stratégie multistart.

Une année après (en 2002), Marchiori [74] a proposé trois heuristiques, génétique recherche locale, recherche locale itérative et de recherche locale multistart pour le MCP appelées GENE, ITER et MULT respectivement. GENE utilise un croisement uniforme, une mutation d'échange et une méthode de descente. ITER utilise plusieurs reprises sur une seule solution candidate et elle renvoie la meilleure solution trouvée. MULT applique la recherche locale sur un ensemble de solution candidat de grande taille et elle prend la meilleure solution de cet ensemble.

En 2004, Grosso *et al.* [43] ont proposé une heuristique de recherche gloutonne adaptative profonde pour le problème de la clique maximum appelée DAGS. Les auteurs ont introduit deux stratégies basées sur l'opération de changement des sommets (swap en anglais) et le poids des sommets, pour sélectionner les sommets dans une heuristique à deux phases pour éviter les pièges habituels des heuristiques gloutonnes. Dans la première phase, DAGS applique une heuristique gloutonne à partir de chaque sommet de graphe. Dans la deuxième phase, GAGS utilise un sous-ensemble réduit basé sur les résultats de la première phase et une heuristique gloutonne adaptative basée sur un mécanisme multistart. Dans la même année, Hansen *et al.* (2004) [44] ont proposé une heuristique recherche à voisinage variable pour le MCP appelée VNS. VNS utilise la différence de cardinalité entre deux solutions T et T' comme une métrique

de distance pour définir le voisinage d'une solution $N_k(T)$ ($N_k(T)$ est composé de toutes les solutions à distance k de T). VNS utilise trois opérations de déplacement, de changement des sommets (swap en anglais), ajouter un sommet (add en anglais) et supprimer un sommet (drop en anglais) pour les voisinages sélectionnés $N_k(T)$ ($k = 1, ..., k_{max}$).

En 2005, Zhang *et al.* [131] ont proposé un algorithme évolutionnaire hybride avec une mutation guidée pour le problème de la clique maximum appelé EA/G. EA/G améliore les résultats de GENE (Marchiori (2002) [74]). EA/G utilise la mutation guidée pour générer des nouveaux solutions (progénitures), ensuite elle utilise la recherche locale de GENE pour améliorer ces solutions. La mutation guidée combine les informations statistiques globales et les informations de localisation des solutions trouvées auparavant. Les résultats expérimentaux de Zhang *et al.* (2005) [131] ont montré que EA/G superforme GENE sur les instances de référence DIMACS. Aussi en 2005, Katayama *et al.* [56] ont proposé un algorithme basé sur la recherche en profondeur variable (variable depth search based algorithm) appelé $k - optlocalsearch$ (KLS) pour le MCP. KLS généralise le principe basique de la recherche locale à 1-opt ($1 - optlocalsearch$) en déplaçant plusieurs sommets à chaque itération au lieu d'un seul sommet (en déplaçant k sommets avec des opérations successives d'ajouter ou de supprimer de la clique courante), la sélection des sommets se fait avec une manière déterministe comme celle de Gendreau *et al.* (1993) [39]. KLS utilise aussi une stratégie multistart.

En 2006, Pullan & Hoos [94] ont proposé un algorithme de recherche locale stochastique pour le problème de la clique maximum appelé DLS-MC. DLS-MC alterne entre une phase d'amélioration itérative, pendant laquelle des sommets souhaitables sont ajoutés à la solution courante, et la recherche de plateau, pendant laquelle les sommets de la solution courante (C) sont échangés avec des sommets de $V \setminus C$. La sélection des sommets dans DLS-MC est uniquement basée sur des pénalités de sommets qui sont ajustées dynamiquement pendant la recherche, DLS-MC utilise aussi un mécanisme de perturbation pour surmonter la stagnation de la recherche. Dans la même année, Singh & Gupta (2006) [110] ont proposé une heuristique génétique hybride basée sur l'état stationnaire pour le MCP appelé HSSGA. HSSGA combine trois algorithmes, un algorithme génétique hybrid basé sur l'état stationnaire, un algorithme glouton aléatoire séquentiel et l'algorithme Branch and Bound CP (proposé par Carraghan & Pardalos (1990) [26]). Les résultats expérimentaux de Singh & Gupta (2006) [110] ont montré que HSSGA superforme les autres algorithmes évolutionnaires proposés auparavant. Aussi en 2006, Pullan [92] a proposé un algorithme de recherche locale par phases (Phased Local Search) pour le MCP appelé PLS. PLS intercale 3 sous-algorithmes qui alternent entre des séquences d'amélioration itérative, au cours desquelles des sommets appropriés sont ajoutés à la solution courante, et la recherche de plateau, où les sommets de la solution courante (C) sont échangés avec des sommets de $V \setminus C$. Les sous-algorithmes se diffèrent dans la sélection des sommets

et dans la technique de perturbation (diversification). Encore en 2006, Busygin [22] a proposé un algorithme glouton déterministe itératif pour le MCP appelé QUALEX-MS. L'auteur a attribué à chaque sommet un poids qui représente son importance vers l'inclusion dans la solution courante.

En 2011, Pullan *et al.* [95] ont proposé un algorithme de recherche locale coopérant pour le problème de la clique maximum appelé CLS. CLS est un algorithme parallèle hyper-heuristique Burke *et al.* (2003) [21]. L'hyper-heuristique est définie comme une heuristique qui cherche à automatiser le processus de sélection, de combinaison, de génération ou d'adaptation d'un certain nombre d'heuristiques simples pour résoudre efficacement des problèmes de recherche computationnelle. CLS utilise quatre heuristiques simples qui peuvent résoudre différents types d'instances efficacement. La différence entre ces heuristiques réside dans la technique de sélection des sommets et dans la façon de traitement les plateaux.

Une année après en 2012, Andrade *et al.* [2] ont proposé deux heuristiques recherches locales rapides pour le problème du stable maximum appelées ILS. Les auteurs ont introduit deux types de déplacement, un $(1,2)-swap$, qui change un seul sommet de la solution courante S par deux sommets de $V \setminus S$. Et un $(2,3)-swap$, qui change deux sommets de la solution courante S par trois sommets de $V \setminus S$. L'étude des auteurs a montré aussi que, pour toute solution, l'opération $(1,2)-swap$ peut être implémenté en un temps linéaire, tandis que l'opération $(2,3)-swap$ peut être implémenté en $O(m\Delta)$, où m est le nombre d'arêtes et Δ est le plus grand degré dans le graphe. Les résultats expérimentaux ont montré l'algorithme ILS fonctionne très bien sur les instances de référence DIMACS et BHOSLIB.

En 2013, Wu & Hoa [124] ont proposé une heuristique de recherche tabou adaptative multistart pour le problème de clique maximum appelé AMTS. AMTS recherche une clique de taille fixée k en explorant efficacement un espace de recherche composé par tous les sous-ensembles de V de taille k (légale et illégale clique de taille k). AMTS combine une procédure de recherche tabou avec une stratégie multistart guidé. AMTS utilise un voisinage contraint pour limiter le déplacement des sommets (l'opération swap) à deux sous-ensembles critique $A \subseteq C$ et $B \subseteq V \setminus C$, le voisinage résultant contient toujours les meilleures solutions du voisinage. L'ensemble A contient les sommets de la solution (clique) courante, qui ne sont pas interdits par la liste tabou et qui ont le plus petit nombre de voisins dans C, et l'ensemble B contient les sommets de $B \subseteq V \setminus C$ qui ne sont pas interdits par la liste tabou et qui ont le plus grand nombre de voisins dans C. Une solution voisine C' de la solution C est alors obtenue en changeant un sommet de A avec un sommet de B. L'algorithme AMTS a montré une excellente performance sur l'ensemble de référence DIMACS. Aussi en 2013, Benlic & Hao (2013) [10] ont proposé un algorithme de recherche locale breakout pour le MCP (et ça version de poids des sommets) appelé BLS. BLS alterne entre une phase de recherche locale et une phase de

diversification par perturbation, la première est pour trouver un optimum local et la deuxième pour passer d'un optimum local à un autre optimum local. La deuxième phase (diversification) a une grande importance sur la performance de BLS. La diversification s'effectue en alternant entre des perturbations aléatoires ou dirigées et des perturbations faibles ou fortes en fonction de l'état actuel de la recherche. L'algorithme BLS a montré une excellente performance sur les deux ensembles de référence DIMACS et BHOSLIB.

En 2015, Jin & Hao [53] ont proposé un algorithme de recherche tabou basé sur un swap général pour le problème du stable maximum appelé SBTS. SBTS alterne dynamiquement entre une étape d'intensification et une étape de diversification à chaque itération de recherche. Et cela, se fait en appliquant une opération de déplacement générale $(k,1)-swap$ ($1 \geq k \geq 0$ pour l'intensification et $k \geq 2$). Soit S un stable (une solution courante), l'opération de déplacement $(k,1)-swap$ change un seul sommet de $V \setminus S$ avec k sommets de S. Les résultats de calcul numérique ont montré que l'algorithme SBTS compète favorablement avec 5 algorithmes de l'état de l'art de la littérature.

En 2019, Smith *et al.* [112] ont proposé un algorithme hybride pour le problème de la clique maximum appelé HTS. HTS est une nouvelle heuristique de départ efficace qui permet à la recherche tabou de se concentrer sur les domaines prometteurs de l'espace de recherche. HTS utilise une heuristique principale qui génère des bonnes solutions initiales, certains nombres d'optimisations simples et une heuristique de recherche tabou. L'heuristique de recherche tabou est différente aux autres, elle utilise un algorithme exact ou pseudo-exact.

Vous trouvez aussi une étude détaillée sur les heuristiques proposées dans 1999 voir Pardalos & Xue (1994) [86] et Bomze *et al.* (1999) [14], la plupart de ces heuristiques sont des heuristiques gloutonnes appelées aussi des heuristiques gloutonnes séquentielles. Dans Wu & Hoa (2015) [126] vous trouvez une étude détaillée et une étude comparative sur les heuristiques publiées avant 2015. La plupart des travaux publiés après 2015 sont dédiés pour le cas de la clique de poids maximum i.e., pour le problème de la clique de poids maximum (maximum weighted clique problem en anglais). Dans la Section 1.5.1 et 1.5.2 nous définissons le problème de la clique de poids maximum avec ces deux versions (poids des sommets et poids des arêtes respectivement) et nous donnons un aperçu sur leurs méthodes de résolution.

1.5 Généralisations et relaxations du problème de la clique maximum

Le problème de la clique maximum a connu quelque généralisation et relaxation, qui sont très utiles pour la modélisation d'un grand nombre de problèmes où le MCP standard n'est

pas approprié. Les généralisations du MCP les plus connues sont le problème de la clique de poids maximum des sommets (MVWCP) et le problème de la clique de poids maximum des arêtes (MEWCP), ces problèmes ont connu de nombreuses applications dans le monde réel, on cite Li & Latecki (2012) [66], Brendel *et al.* (2011) [16], Brendel & Todorovic (2010) [17]. On trouve dans la littérature trois types de relaxation. La première est la relaxation basée sur la densité ou quasi-clique [1]. La deuxième est la relaxation basée sur le degré connu sous le nom de k-plexes [105] et la troisième est le chemin basé sur les relaxations connues sous le nom de k-club [78] et k-cliques [70]. Le problème de la quasi-clique maximum (MCQP) est le plus connu. Le MCQP a une large application dans nombreux réseaux réels tel que les réseaux interactions protéine-protéine, les réseaux sociaux et les réseaux écologiques.

1.5.1 Problème de la clique de poids maximum des sommets (MVWCP)

Soit $G(V, E, W_v)$ un graphe avec des poids sur les sommets (connu aussi sous graphe pondéré), où V l'ensemble des sommets, E l'ensemble des arêtes et $W_v : V \to R^+$ une fonction de pondération qui associe à chaque sommet $u \in V$ un poids positif w_u.

Pour toute clique C de G, le poids des sommets de C est défini par $W_v(C) = \sum_{u \in C} w_u$. Le problème de la clique de poids maximum des sommets (the maximum vertex weight clique problem en anglais) vise à trouver la clique de poids maximum des sommets C^* i.e., Soit Ω l'ensemble de tous les cliques de G, $\forall C \in \Omega, W_v(C^*) \geq W_v(C)$. $\omega_{w_v}(G)$ (noté aussi par $\omega_w(G)$) est le poids de la clique de poids maximum des sommets (appelé le nombre de la clique de poids maximum des sommets). On note que la clique de poids maximum des sommets n'est pas forcément la clique maximum, sauf si tous les poids des sommets égaux à 1, dans ce cas la clique de poids maximum des sommets est elle-même la clique maximum. Il est facile de voir que le problème de la clique maximum est un cas particulier du problème de la clique de poids maximum des sommets où tous les poids des sommets sont égaux à 1. Il est aussi facile de voir que le MVWCP est un problème NP-Complet, car le MCP est un cas particulier de MVWCP.

Le problème de la clique de poids maximum des sommets a connu plusieurs algorithmes exacts : Branch and Bound Östergård (1999)[83], Warren & Hicks (2006) [123], Yamaguchi & Masuda (2008) [129], Jiang *et al.* (2017) [52] ; Algorithme exact basé sur une heuristique de coloration des sommets et une recherche en retour sur trace Kumlander (2004) [61], Shimizu *et al.* (2012) [109] ; Algorithme exact basé sur le raisonnement de MaxSAT Fang *et al.* (2016) [33] ; algorithme branch and reduce qui est aussi un algorithme B&B Lamm *et al.* (2019) [62]. On trouve aussi dans la littérature plusieurs heuristiques proposées pour la résolution de MVWCP : Algorithme d'accroissement (Augmentation algorithm en anglais) Mannino & Stefanutti (1999) [73] ; algorithme parallèle, distribué, basé sur un réplicateur dynamique Bomze *et al.* (2000) [15] ; algorithme à pivotement complémentaire (complementary pivoting algorithm en anglais)

Massaro *et al.* (2002) [76] ; algorithme de recherche locale, dynamique, réactive, stochastique Pullan (2008) [93] ; un algorithme de recherche tabou multi-voisinage Wu *et al.* (2012) [127] ; algorithme qui intercale la construction de la clique et la réduction de graphe Cai & Lin (2016) [24] ; algorithme de recherche tabou avec un opérateur généralisé (PUSH) Zhou *et al.* (2017) [133] ; algorithme de recherche locale itérative hybride Nogueira *et al.* (2018) [80] ; algorithme de recherche locale CPU-GPU Nogueiraa & Pinheiroa (2018) [79] ; algorithme de recherche locale avec deux nouvelle stratégie tabou et multistart Fan *et al.* (2018) [32].

1.5.2 Problème de la clique de poids maximum des arêtes (MEWCP)

Soit $G(V, E, W_e)$ un graphe avec des poids sur les arêtes (connu aussi sous graphe pondéré), où $W_e : E \rightarrow R^+$ une fonction de pondération qui associe à chaque arête $uv \in E$ un poids positif w_{uv}.

Le poids des arêtes d'une clique C est défini par $W_e(C) = \sum_{u,v \in C} w_{uv}$. Le problème de la clique de poids maximum des arêtes (the maximum edge weighted clique problem en anglais) vise à trouver la clique de poids maximum des arêtes C^* i.e., $\forall C \in \Omega, W_e(C^*) \geq W_e(C)$. Comme le MCP et le MVWCP, le MEWCP est problème NP-Complet Ausiello *et al.* (2012) [3]. Dans la littérature, on trouve plusieurs cas particuliers et relaxations de MEWCP connus sous différents noms. Par exemple, le problème de sous-graphe de poids maximum des arêtes (the maximum edge-weighted subgraph problem en anglais) Macambira (2002) [71], le problème de remote-clique Birnbaum & Goldman (2009) [13] et le problème de la diversité maximum (the maximum diversity problem en anglais) Glover *et al.* (1998) [40].

Le problème de la clique de poids maximum des arêtes n'a pas connu un grand nombre de méthodes de résolution dans la littérature comme le MCP et le MVWCP. Les méthodes exactes : Dijkhuizen & Faigle (1993) [29] ont proposé un algorithme des plans sécants (cutting-plane algorithm en anglais), Park *et al.* (1996) [87] ont proposé une approche de formulation élargie, Gouveia & Martins (2015) [42] ont étudié le comportement des formulations compactes pour la résolution de MEWCP, Shimizu *et al.* (2018) [107] et Shimizu *et al.* (2019) [108] ont proposé des algorithmes Branch and Bound, Hosseinian *et al.* (2018) [47] ont proposé un algorithme Branch and Bound basé une formulation d'optimisation quadratique non convexe pour le MEWCP. On trouve aussi quelque heuristiques dans la littérature : recherche tabou Macambira (2002) [71] ; recherche locale Pullan (2008) [93], Li *et al.* (2018) [67] ; une heuristique basée sur l'optimisation d'une fonction quadratique sur une sphère Hosseinian *et al.* (2016) [46].

1.5.3 Problème de la quasi-clique maximum

Soit $G = (V, E)$ un graphe et $\gamma \in (0, 1]$ un paramètre constant réel. Un sous-ensemble S de V est une $\gamma - quasi - clique$ ($\gamma - clique$) si le nombre d'arêtes dans le sous-graphe induit par S est au moins égal à $\gamma * \begin{pmatrix} |S| \\ 2 \end{pmatrix}$ arêtes ($\gamma * \begin{pmatrix} |S| \\ 2 \end{pmatrix} = \gamma * |S| * (|S| - 1)/2$), (Abello *et al.* (2002) [1]). Le problème de la $\gamma - quasi - clique$ maximum (MQCP) vise à déterminer la $\gamma - quasi - clique$ de plus grande taille dans G, il est montré que le MQCP est un problème NP-difficile Pattillo *et al.* (2013) [89], $\omega_\gamma(G)$ est la taille de la $\gamma - quasi - clique$ maximum. Il est facile de voir que si $\gamma = 1$, la $\gamma - quasi - clique$ devient une clique et le problème de la $\gamma - quasi - clique$ maximum devient le problème de la clique maximum ($\omega_1(G) = \omega(G)$). Une deuxième définition du MQCP est aussi trouvée dans la littérature, un sous-graphe S de V est une $\gamma - quasi - clique$ si tout sommet de S à au moins $\gamma * (|S| - 1)$ voisins dans le sous-graphe induit par S (Matsuda *et al.* (1999) [77]), le MQCP avec cette définition est aussi connu sous le problème de la $\gamma - quasi - clique$ maximum basé sur le degré (the maximum degree-based γ-quasi-clique problem en anglais) Pastukhov *et al.* (2018) [88]. Dans cette thèse, nous nous intéressons au problème de la quasi-clique selon la première définition.

Soit G un graphe et P une propriété de G, P est dite héréditaire si tout sous-graphe de G vérifié la propriété P. La clique est héréditaire par définition (tout sous-ensemble d'une clique est une clique), tandis que la $\gamma - quasi - clique$ n'est pas héréditaire (la condition que tout sous-ensemble d'une $\gamma - quasi - clique$ est une $\gamma - quasi - clique$ n'est pas vrai). La Figure 1.3 montre un exemple d'une $\gamma - quasi - clique$ dans un graphe de densité 0.62 avec illustration de l'absence de l'hérédité, à gauche en rouge une $0.8 - quasi - clique$ ($S = \{2, 3, 4, 5, 6\}$) et à droit en vert un sous-ensemble de cette $0.8 - quasi - clique$ ($\{3, 4, 5, 6\} \subset S$) qui n'est pas une $0.8 - quasi - clique$. Patillo *et al.* (2013) [89] ont introduit la notion quasi-héréditaire et ils ont prouvé que la $\gamma - quasi - clique$ est quasi-héréditaire, une propriété P est quasi-héréditaire veux dire que dans un graphe G qui vérifie la propriété P, il existe un sommet v tel que $G - v$ vérifie P. Le plus grand défi dans les algorithmes du problème de la quasi-clique maximum est comment gérer l'absence de l'hérédité.

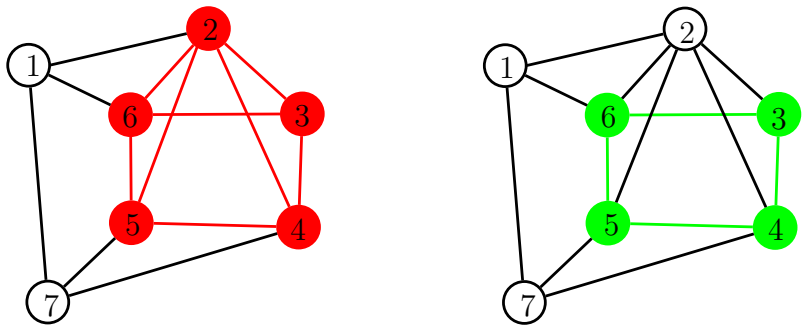


FIGURE 1.3 – Exemple d'une $\gamma - quasi - clique$ avec illustration de l'absence de l'hérédité, à gauche en rouge une $0.8 - quasi - clique$ S et à droit en vert un sous-ensemble de S qui n'est pas une $0.8 - quasi - clique$.

Dans la littérature, nombreuses méthodes exactes et heuristiques ont été proposées par divers chercheurs pour résoudre le problème de la quasi-clique maximum. Parmi les méthodes exactes, nous mentionnons, la méthode basée sur le principe général Branch and Bound proposée par Pajouh *et al.* en 2014 [84]. Deux méthodes exactes basées sur la programmation en nombres entiers mixte proposées par Pattillo *et al.* (2013) [89] et Veremyev *et al.* (2016) [120]. En 2018, une autre méthode Branch and Bound est proposée par Pastukhov *et al.* [88] pour la résolution de MQCP selon la deuxième définition. Plusieurs heuristiques sont proposées pour trouver des solutions approchées, la première est proposée par Matsuda *et al.* [77] en 1999, elle est été utilisée pour trouver la plus grande $\gamma - quasi - clique$ pour classer un grand nombre de séquences de protéines non caractérisées. En 2002, Abello *et al.* [1] ont proposé un algorithme glouton adaptatif aléatoire (greedy randomized adaptive search procedure (GRASP) en anglais) pour trouver la plus grande $\gamma - quasi - clique$ dans les réseaux de télécommunications. Brunato *et al.* (2007)[20] ont introduit une nouvelle définition (une troisième définition) pour la quasi-clique (le sous-graphe $G[S]$ induit par le sous-ensemble S de V est une (λ, γ)-quasi-clique avec $0 \leq \lambda \leq \gamma \leq 1$ si seulement si tout sommet de S à au moins $\lambda(|S| - 1)$ voisins dans $G[S]$ et le nombre d'arêtes dans $G[S]$ est au moins égal à $\gamma \begin{pmatrix} |S| \\ 2 \end{pmatrix}$), ils ont aussi proposé des extensions de deux algorithmes de MCP pour trouver la (λ, γ)-quasi-clique maximum dans des instances (réseaux) artificiels et du monde réel. Une heuristique gloutonne est développée par Bhattacharyya & Bandyopadhyay en 2009 [11] pour trouver la plus grande $\gamma - quasi - clique$ dans un réseau d'interaction protéine-protéine humain. En 2013, Oliveira *et al.* [81] ont proposé une heuristique constructive. En 2017, Sriwastava *et al.* [113] ont aussi proposé une heuristique appelée qCLiP pour trouver la plus grande $\gamma - quasi - clique$ dans un réseau d'interaction protéine-protéine humain. Pastukhov *et al.* (2018) [88] ont proposé une heuristique appelée l'algorithme de décomposition des degrés (the degree decomposition algorithm (DDA) en anglais) pour résoudre le MQCP selon la deuxième définition. En 2018, Pinto *et al.* [91] ont proposé deux variantes d'un algorithme génétique à clé aléatoire biaisé (biased random-key

genetic algorithm (BRKGA) en anglais), une version préliminaire de l'algorithme BRKGA est proposée par Pinto *et al. (2015)* [90].

Chapitre 2

Une extension de la recherche tabou adaptative multistart pour le problème de la quasi-clique maximum

Dans ce chapitre, nous proposons un algorithme de recherche tabou appelé TSQC, pour la résolution du problème de la quasi-clique maximum. TSQC est une extension de l'algorithme de recherche tabou adaptative multistart (AMTS) proposé par Wu & Hao (2013) [124] pour la résolution du problème de la clique maximum. Nous avons évalué notre algorithme TSQC sur les instances de référence DIMACS et BHOSLIB du problème de la clique maximum, et aussi sur des réseaux réels, nous avons comparé les résultats de TSQC avec les méthodes exactes et les heuristiques les plus récentes. Les résultats des calculs révèlent que l'algorithme TSQC atteint les plus grandes quasi-cliques connues dans un temps raisonnable sur les réseaux réels et qu'il surperforme les méthodes existantes sur les instances DIMACS et BHOSLIB. De plus, notre algorithme donne des nouveaux résultats avec une qualité améliorée pour le MQCP. Le contenu de ce chapitre est publié dans Djeddi, Ait Haddadene & Belacel [30].

Sommaire

2.1 Introduction

Le principe de l'algorithme TSQC est de trouver une k-γ-quasi-clique i.e. une γ-quasi-clique de taille k (un sous-ensemble $S^* \subseteq V$ tel que le nombre d'arêtes dans le sous-graphe induit par S est au moins égal à $\gamma * \begin{pmatrix} |S| \\ 2 \end{pmatrix}$). Elle commence par un sous-ensemble aléatoire de taille k (k-γ-quasi-clique illégale, solution initiale) et elle essaye d'améliorer le nombre d'arêtes de ce sous-ensemble. Pour améliorer la solution initiale, TSQC utilise une série d'intensification et de diversification. Dans l'intensification, l'algorithme TSQC échange un sommet d'un sous-ensemble critique $A \subset S$ avec un sommet d'un sous-ensemble critique $B \subset V \setminus S$ pour améliorer la solution actuelle (voir Section 2.2.4). Pour échapper à l'optimum local, TSQC applique une diversification et déplace la recherche vers une nouvelle zone de recherche. De plus, l'algorithme TSQC utilise deux listes tabous pour empêcher la recherche à revenir aux solutions déjà visitées. Nous avons amélioré quelques règles dans le critère d'inspiration de la liste tabou pour le rendre plus efficace sur le problème de la quasi-clique maximum (voir Section 2.2.4). Pour une exploration plus efficace de l'espace de recherche et visiter des nouvelles régions, TSQC applique la même stratégie de redémarrage (restart) utilisée par l'algorithme AMTS (voir Section 2.2.5). Notre algorithme TSQC est le premier algorithme de recherche tabou utilisé pour résoudre le MQCP et aussi la première application du k-γ-quasi-clique pour la résolution du MQSP.

2.2 Méthode de recherche tabou pour la résolution du problème de la quasi-clique maximum

AMTS utilise ce qui est noté dans Friden *et al.* (1989)[35], le problème de la clique maximum peut être approché par la recherche d'une série de k-cliques, chaque fois qu'une k-clique est trouvée, k est incrémenté par un, jusqu'à ce qu'aucune k-clique ne puisse être trouvée [124]. Dans notre algorithme, nous avons appliqué le même principe dans le problème de la γ-quasi-clique

maximum. Le problème de la γ-quasi-clique maximum peut être approché par la recherche d'une série de k-γ-quasi-clique (k-γ-quasi-clique est une γ-quasi-clique de taille k), ainsi, chaque fois qu'une k-γ-quasi-clique est trouvée, nous incrémentons k par 1 et nous répétons cette opération jusqu'à ce que nous ne puissions pas trouver une k-γ-quasi-clique. La dernière k-γ-quasi-clique trouvée représente une approximation de la γ-quasi-clique maximum. Notre algorithme TSQC est conçu pour le problème de la recherche d'une k-γ-quasi-clique dans un graphe $G = (V, E)$. Notre algorithme TSQC sera présenté dans la section 2.2.1 suivante.

2.2.1 Procédure générale

TSQC comme AMTS commence par une solution initiale S (un sous-ensemble de taille k) générée par une heuristique gloutonne, à partir de cette solution initiale TSQC cherche à trouver une k-γ-quasi-clique par une série d'intensification et de diversification, ceci est réalisé par l'algorithme TSQ. À chaque étape TSQ change un sommet de S avec un sommet de $V \setminus S$, le changement est limité aux sommets de deux sous-ensembles A et B (A et B sont des sous-ensembles contraints de S et $V \setminus S$ respectivement). Comme dans AMTS, l'algorithme TSQC s'arrête lorsqu'une k-γ-quasi-clique légale est trouvée et retourne cette k-γ-quasi-clique ou lorsqu'il atteint le nombre maximum d'itérations préalablement fixé (It_{mx}) sans trouver la k-γ-quasi-clique légale (dans ce cas, une erreur est détectée). La procédure générale de TSQC est résumée dans l'algorithme 1 et une vue générale de l'algorithme TSQC est présentée sous forme d'organigramme à la Figure 2.1.

Algorithme 1 L'algorithme TSQC pour le problème de la γ-quasi-clique maximum

Entres: Un graphe G, $\gamma \in (0, 1]$, Un entier k (la taille de la γ-quasi-clique), Un entier L (profondeur de recherche), Un entier It_{mx} (le nombre maximum d'itérations autorisées)
Sorties: k-γ-quasi-clique si elle est trouvée

```
Début
S ← solution initiale(k) /* Solution initiale Section 2.2.3 */
It ← 0 /* Initialisation du compteur d'itération */
Tantque It < It_mx Faire
  S* ← TSQ(G, γ, S, k, L, It) /* Appeler la procédure TSQ pour améliorer la solution initiale S Section 2.2.4 */
  Si S* est une k-γ-quasi-clique légale Alors
    Retourner(S*) et Arrêter
  Sinon
    S ← nouvelle solution initiale(k) /* Générer une nouvelle solution initiale S Section 2.2.5 */
  Finsi
Fin tantque
Fin
Retourner(Défaut)
```

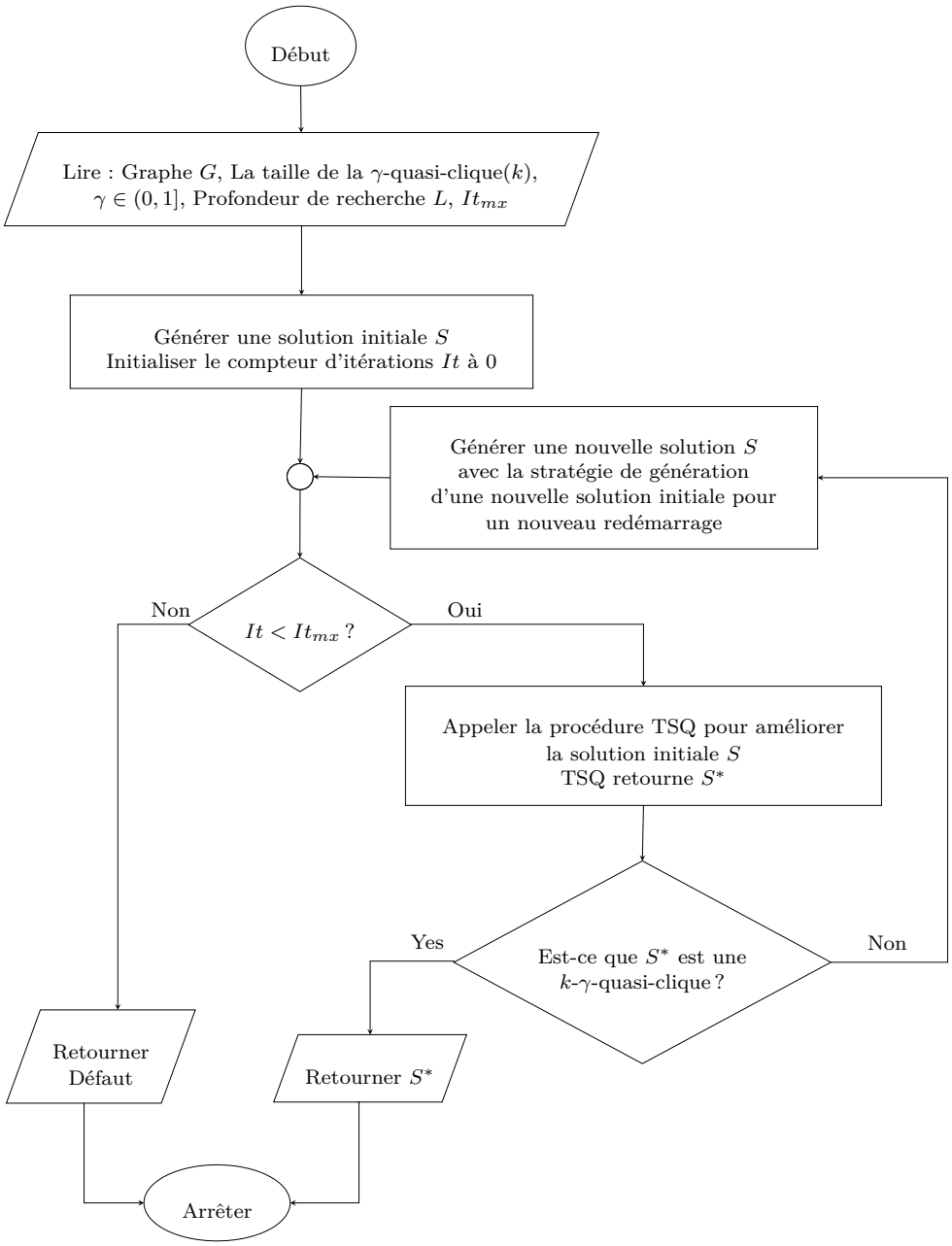


FIGURE 2.1 – Organigramme illustrant une vue générale de l'algorithme TSQC

2.2.2 Espace de recherche et fonction d'évaluation

Les algorithmes TSQC et AMTS explorent le même espace de recherche. Soit $G = (V, E)$ un graphe, l'espace de recherche Ω de G est l'ensemble de tous les sous-ensembles S de taille k (k-γ-quasi-clique réalisable et non-réalisable) i.e., $\Omega = \{S \subset V : |S| = k\}$.

L'évaluation de tout sous-ensemble S est basée sur le nombre d'arêtes induites par S i.e., la fonction d'évaluation de tout sous-ensemble S égale au nombre d'arêtes induites par S :

$$f(S) = |\{uv \in E : u \in S, v \in S\}|$$

La fonction $f(S)$ est à maximiser i.e., on dit que S' est mieux que S si seulement si

$f(S') > f(S)$. Si $f(S) \geq \gamma * (k * (k-1)/2)$, S est une k-γ-quasi-clique sinon S n'est pas une k-γ-quasi-clique (S est une k-γ-quasi-clique illégale).

2.2.3 Solution initiale

La solution initiale utilisée par TSQC est générée de la même manière avec celle de AMTS, elle est générée par l'heuristique gloutonne aléatoire suivante :

1. Initialiser l'ensemble S à vide.
2. Sélectionner un sommet $v \in V$ aléatoirement et ajouter v à S.
3. Ajouter à S le sommet qui a le plus grand nombre de voisins dans S, s'il y a plusieurs sommets, sélectionnez un aléatoirement.
4. Répéter l'étape (3) tantque $|S| < k$ et retourner S.

2.2.4 L'algorithme TSQ

L'algorithme TSQ est une adaptation de l'algorithme TS^0 de la méthode AMTS dans le problème de la recherche d'une k-γ-quasi-clique (TS^0 est un algorithme de recherche tabou, il est utilisé dans la méthode AMTS comme sous algorithme). Comme TS^0, l'algorithme TSQ est basé sur la méthode de recherche tabou donnée par Glover & Laguna (1997)[41]. La procédure TSQ est résumée dans l'algorithme 2. La solution actuelle est représentée par S et la meilleure solution trouvée jusqu'à présent est représentée par S^*. I et It sont deux compteurs d'itérations utilisés dans les tests d'arrêt, I est utilisé pour le test d'arrêt de TSQ et It est utilisé pour le test d'arrêt global de TSQC. TSQ explore l'espace de recherche d'une façon itérative en passant de la solution actuelle à une nouvelle solution (la boucle tantque, les lignes 5-22). Dans chaque étape, TSQ applique l'intensification i.e., elle remplace la solution actuelle S par une nouvelle solution améliorée S' ($f(S') \geq f(S)$) (les lines 6-7, voir Section 2.2.4) ; ou bien elle applique une diversification en remplaçant la solution actuelle S par une nouvelle solution S' tel que $f(S') < f(S)$, lorsque la solution actuelle ne peut être améliorée à l'étape d'intensification (lorsque la recherche est tombée dans un optimum local) ; dans l'étape de diversification, TSQ s'échappe de l'optimum local et déplace la recherche vers une nouvelle zone de recherche (les lines 8-9, voir Section 2.2.4). TSQ utilise deux listes tabous pour ne pas revenir temporairement aux solutions déjà visitées, ces listes tabous sont mises à jour après chaque étape d'intensification ou de diversification (la line 11, voir Section 2.2.4). Si une k-γ-quasi-clique est trouvée i.e., une solution S avec $f(S) \geq \gamma * (k * (k-1))$, la procédure TSQ s'arrête et retourne cette solution (les lignes 12-14). Si l'algorithme TSQ trouve une nouvelle solution S tel que S est mieux que la meilleure solution trouvée jusqu'à présent S^*, TSQ remplace S^* par S (les lignes 15-21). Si la

solution actuelle S n'est pas améliorée après L itérations consécutives (L est appelé profondeur de recherche), la boucle tantque s'arrêtera et TSQ retourne S^* comme solution finale. Dans ce cas, TSQC est supposé être tombé dans un optimum local, pour lequel TSQC génère une nouvelle solution initiale et redémarre TSQ avec cette solution (voir Section 2.2.5). Une vue générale de l'algorithme TSQ est présentée sous forme d'organigramme dans la Figure 2.2.

Algorithme 2 La procédure TSQ pour rechercher une k-γ-quasi-Clique

Entres: Un graphe G, $\gamma \in (0, 1]$, S (Solution initiale), Un entier k (la taille de la γ-quasi-clique), Un entier L (profondeur de recherche), Un entier It (Compteur d'itérations)
Sorties: S^* (La meilleure solution trouvée)

Début
$S^* \leftarrow S$ /* Initialiser la meilleure solution trouver jusqu'à présent S^* à la solution initiale S */
$f^* \leftarrow f(S^*)$ /* Enregistrer la valeur de la fonction d'évaluation de la meilleure solution S^* */
$I \leftarrow 0$ /* Initialiser le compteur d'itérations consécutives I à 0 */
Tantque $I < L$ **Faire**
Si il existe une solution d'amélioration **Alors**
$S \leftarrow Intensification(S)$ /* Appliquer une intensification Section 2.2.4 */
Sinon
$S \leftarrow Diversification(S)$ /* Appliquer une Diversification Section 2.2.4 */
Finsi
Mis à jour *tabu_list_u* and *tabu_list_v* /* voir Section 2.2.4 */
Si S is a legal k-γ-quasi-clique **Alors**
Retourner(S) and **Arrêter**
Finsi
Si $f(S) > f^*$ **Alors**
$S^* \leftarrow S$
$I \leftarrow 0$
$f^* \leftarrow f(S^*)$
Sinon
$I \leftarrow I + 1$
Finsi
$It \leftarrow It + 1$
Fin tantque
Fin
Retourner(S^*)

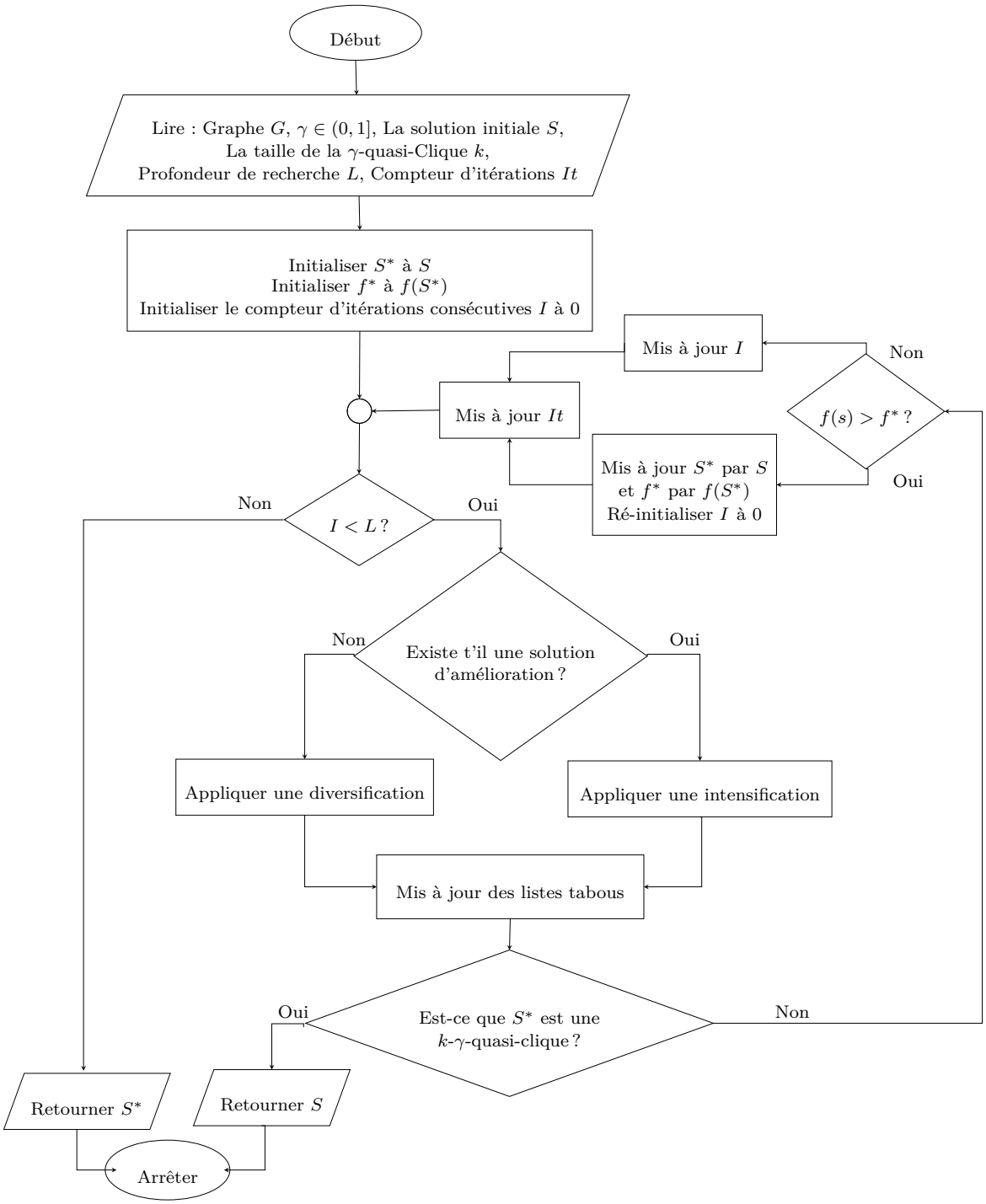


FIGURE 2.2 – Organigramme illustrant une vue générale de l'algorithme TSQ

Intensification

Avant d'expliquer l'étape d'intensification, nous définissons d'abord le voisinage d'une solution S utilisée par TSQ. Soit S un sous-ensemble de Ω ($S \subset \Omega$), le degré $d(v)$ relatif au sous-ensemble S de tout sommet v est défini comme suit :

$$d(v) = |\{u \in S : uv \in E\}|$$

Le voisinage contraint est défini par deux ensembles critiques A et B comme dans [124] :
Soit $tabu\_list\_u$ et $tabu\_list\_v$ les deux listes tabous utilisées par TSQ (voir Section 2.2.4).
Soit $MinInS = min\{d(u) : u \in S, u \notin tabu\_list\_u\}$ et
Soit $MaxOutS = max\{d(v) : v \in V \backslash S, v \notin tabu\_list\_v\}$
A et B sont définis comme suit :

$$A = \{u \in S : u \notin tabu\_list\_u, d(u) = MinInS\}$$

$$B = \{v \in V \backslash S : v \notin tabu\_list\_v, d(v) = MaxOutS\}$$

On dit que S' est une solution voisine de S, si elle est obtenue en échangeant un sommet $u \in A$ avec un sommet $v \in B$. La formule de transformation de S à S' est donnée comme suit : $S' = S \bigoplus swap(u, v)$ ou $S' = S \backslash \{u\} \cup \{v\}$. Le voisinage contraint $CN(S)$ est l'ensemble de toutes les solutions induites par tous les déplacements possibles entre les sommets de A et B i.e.,

$$CN(S) = \{S' : S' = S \backslash \{u\} \cup \{v\}, u \in A, v \in B\}$$

Wu & Hao (2013)[124] ont défini le gain du $swap(u, v)$ par la variation de la fonction objectif f comme suit :

$$\Delta_{uv} = f(S') - f(S) = d(v) - d(u) - e_{uv}$$

Où $e_{uv} = 1$ si $uv \in E$ sinon $e_{uv} = 0$.

Nous en déduisons que pour tout sommet $u \in A$ et tout sommet $v \in B$:

$$\Delta_{uv} = \begin{cases} MaxOutS - MinInS - 1, & si\ uv \in E \\ MaxOutS - MinInS, & sinon. \end{cases}$$

Il est facile de voir que le meilleur $swap(u, v)$ est le $swap$ qui a $\Delta_{uv} = MaxOutS - MinInS$. Soit T l'ensemble des meilleurs échanges (swaps) i.e., $T = \{(u, v) : u \in A, v \in B, uv \notin E, \Delta_{uv} = MaxOutS - MinInS\}$.

L'objectif de l'intensification de TSQ est de trouver une nouvelle solution S' mieux que la solution actuelle S ou une nouvelle solution S' sans détériorer la solution S ($f(S') \geq f(S)$). Il est facile de voir que pour faire une intensification, le Δ_{uv} du $swap(u, v)$ doit être supérieur ou égal à zéro ($\Delta_{uv} \geq 0$). L'étape d'intensification est faite comme suit :

- Si $MaxOutS - MinInS \geq 0$ et T n'est pas vide, TSQ sélectionne une paire de sommets (u, v) aléatoirement à partir T et échange u et v (applique le $swap(u, v)$).
- Sinon si $MaxOutS - MinInS - 1 \geq 0$, TSQ sélectionne un sommet u de A aléatoirement et un sommet v de B aléatoirement et échange u et v (applique le $swap(u, v)$).

- Sinon il n'existe pas de $swap(u, v)$ tel que $\Delta_{uv} \geq 0$. D'où, TSQ poursuit sa recherche avec l'étape de diversification que nous expliquerons dans la section suivante.

Après chaque échange ($swap$) d, $MinInS$, $MaxOutS$, A et B sont mises à jour pour préparer la prochaine itération.

Diversification

Lorsqu'il n'y a pas de solution d'amélioration dans le voisinage de la solution actuelle (TSQ est tombé dans un optimum local), TSQ applique la règle de sélection des mouvements de diversification probabiliste pour éviter cet optimum local et encourager la recherche à visiter une nouvelle zone.

- ◇ Appliqué une grande perturbation avec une petite probabilité $P = min\{\frac{l+2}{k}, 0.1\}$ où $l = \gamma * (k * (k-1)/2) - f(S)$, sélectionner un sommet u de S ($u \in S$) aléatoirement et sélectionner un sommet v de $V \backslash S$ ($v \in V \backslash S$) tel que $d(v) < h$ (s'il y a plusieurs sommets, choisir un aléatoirement) et échange u et v (applique le $swap(u, v)$). Où $h = \lfloor 0.85 * \gamma * k \rfloor$ si $Dn \leq 0.5$, $h = \lfloor \gamma * k \rfloor$ sinon. Dn est la densité de graphe.
- ◇ Avec une probabilité $1 - P$, sélectionner aléatoirement une paire de sommets (u, v) de T si T n'est pas vide, sinon sélectionner deux sommets u et v aléatoirement de A et B respectivement et échanger u avec v.

La définition de l dans TSQ n'est pas la même avec celle de TS^0, aussi la sélection du sommet $v \in V \setminus S$ dans le premier cas ($v \in V \backslash S$ tel que $d(v) < h$) n'est pas la même avec celle de TS^0. Dans TS^0, v a été sélectionné parmi les sommets qui ont $d(v) < \lfloor k * Dn \rfloor$. Ce type de sélection ne fonctionne pas dans le problème de la recherche d'une k-γ-quasi-clique, parce que le problème de la γ-quasi-clique maximum a une grande application dans les réseaux réels, ces réseaux sont des graphes sans échelle (scale-free graphs en anglais). Les graphes sans échelle ont une densité très faible, dans de nombreux cas, TSQ tombe dans $d(v) < \lfloor k * Dn \rfloor$, où $\lfloor k * Dn \rfloor = 0$ ce qui est impossible. Par exemple dans le cas de réseau Geom avec $k = 23$ et $\gamma = 0.9$ (voir Section 2.3), TSQ s'arrête et reporte une erreur après 18 itérations, car la densité Geom est égale à 0.00044 ($Dn = 0.00044$) d'où $d(v) < \lfloor 23 * 0.0004 \rfloor \Rightarrow d(v) < 0$ et cela est impossible parce qu'il n'y a pas de degré ($d(v)$) négatif.

Après chaque échange ($swap$) d, $MinInS$, $MaxOutS$, A et B sont mises à jour pour préparer la prochaine itération.

Listes tabous et critère d'aspiration

La procédure TSQ est comme TS^0, elle utilise deux listes tabous pour éviter le retour rapide aux solutions déjà visitées. Après chaque $swap(u, v)$ le sommet u est ajouté à une liste tabou (appelée $tabu\_list\_u$) donc il ne peut pas revenir à la solution actuelle S qu'après Tu itérations

(Tu est appelé tabou tenure, voir Glover & Laguna (1997) [41]). Et le sommet v est ajouté à une liste tabou appelée $tabu\_list\_v$ pour qu'il ne peut pas sortir de la solution actuelle S avant Tv itérations, Tv est aussi appelée tabou tenure.

Dans notre procédure TSQ, nous avons utilisé le même mécanisme proposé par Wu & Hao (2013)[124], qui est inspiré de Galinier & Hao (1999)[36]. Les tabous tenures sont adaptées par une formule selon la fonction d'évaluation $f(S)$. Précisément, en fonction de l'écart entre la fonction d'évaluation $f(S)$ et le nombre minimum d'arêtes dans la k-γ-quasi-clique (la valeur souhaitée à atteindre par $f(S)$). Le nombre d'arêtes de k-clique est supérieur au nombre d'arêtes de k-γ-quasi-clique d'où l'adaptation des tabous tenures en TSQ est différente à celle de TS^0 (Si nous prenons les mêmes adaptations de TS^0, nous obtenons des grandes valeur de Tu et Tv et cela nous amène à certains cas où tous les sommets de S sont interdits de quitter S, donc A sera vide), Tu et Tv sont définis par les formules suivantes :

$$Tu = l + Random(C - 1)$$

$$Tv = 0.6 * l + Random(0.6 * C - 1)$$

Où :
$l = min\{(\gamma * k * (k-1)/2) - f(S), 10\}$
$C = max\{\lfloor k/40 \rfloor, 6\}$
$Random(Y)$ génère aléatoirement un entier en $\{0, ..., Y\}$.

2.2.5 La stratégie pour générer une nouvelle solution initiale pour un nouveau redémarrage

Wu & Hao (2013) [124] ont proposé un mécanisme pour encourager AMTS à visiter une nouvelle région de recherche lorsque TS^0 retourne une k-clique illégale (TS^0 ne peut pas trouver une solution d'amélioration après L itérations), ce mécanisme est également utilisé dans TSQC i.e., lorsque TSQ retourne une k-γ-quasi-clique illégale, elle redémarre la procédure TSQ avec une nouvelle solution initiale générée selon la règle suivante. Wu & Hao (2013) [124] ont attaché à chaque sommet $v \in V$ une fonction g_v appelée mémoire de fréquence à long terme, g_v enregistre le nombre de mouvements du sommet v, la fonction g_v est maintenue comme suit :

- Initialement, pour tout sommet v de V, $g_v = 0$.
- Chaque fois que le sommet v est entré ou sorti de la solution actuelle S, g_v est incrémenté par un ($g_v = g_v + 1$).
- Si pour tout sommet v de V, $g_v > k$, alors pour tout $v \in V$, réinitialise g_v à zéro ($g_v = 0$).

La nouvelle solution est générée comme suit :

◇ Initialiser un ensemble S à vider.

◇ Ajouter à S le sommet v qui a la plus petite fréquence g_v dans V et répéter l'étape(3) jusqu'à ce que la taille de S devient égal à k.

◇ Sélectionner un sommet $v \in V \backslash S$ qui a le plus grand degré $d(v)$, s'il y a plusieurs sommets qui vérifient cette condition, sélectionner le sommet qui a la plus petite fréquence g_v. S'il y en a encore plusieurs, sélectionner un aléatoirement.

2.3 Résultats expérimentaux

Dans cette section, nous donnons une évaluation de notre méthode TSQC en termes de la qualité de solution (la plus grande γ-quasi-clique trouvée) et en termes de temps d'exécution. Nous avons comparé nos résultats avec les résultats obtenus par les méthodes exactes et heuristiques les plus récentes. Pour l'évaluation du TSQC, nous avons utilisé deux groupes d'instances. Le premier groupe est composé de 25 instances de référence DIMACS et de 28 instances de référence BHOSLIB du problème de la clique maximum (DIMACS 1996 [54], BHOSLIB [128]), la densité de ces graphes est comprise entre 0.24 et 0.91. Nous avons comparé les résultats du TSQC sur certaines instances DIMACS avec les résultats des méthodes exactes données dans Pajouh *et al. (2014)* [84] avec $\gamma \in \{0.85, 0.95, 1\}$, Pajouh *et al. (2014)* [84] ont proposé une méthode Branch and Bound et ils ont comparé leur méthode avec les résultats de deux formulations exactes en nombres entiers mixtes (F1 et F2) proposées par Pattillo *et al. (2013)* [89]. Nous avons comparé nos résultats avec les résultats obtenus par Pattillo *et al. (2013)* [89]. Nous avons également comparé les résultats du TSQC sur les instances DIMACS et BHOSLIB avec les résultats obtenus récemment par Pinto *et al. (2018)* [91], avec leur récente heuristique génétique à clé aléatoire biaisé BRKGA-IG* avec $\gamma \in \{0.5, 0.8, 0.95, 0.999\}$.

Le deuxième groupe d'instances est composé de quelques réseaux réels (ces réseaux sont des graphes sans échelle, ils ont une densité < 0.04), Réseaux biologiques (C.Elegans), Réseaux d'Internet et de communication (AS-735, California, Email, Harvard500), Réseaux sociaux (Geom, ca-GRQC, SmallW, netscience), Réseaux de transport (USAir97), ces réseaux sont trouvés dans Balasundaram *et al. (2005)* [6], les bases de données Pajek (2006) [7] et la collection de matrices de la Suite Sparse Matrix [27] (Florida Sparse Matrix Collection). Nous avons comparé les résultats obtenus sur le deuxième groupe d'instances avec les résultats obtenus dans Veremyev *et al. (2016)* [120] avec $\gamma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Veremyev *et al. (2016)* ont proposé dans leur papier [120] quatre formulations exactes en nombres entiers mixtes (F3, F3log, F4 et F4log) et deux algorithmes (AlgF3 et AlgF4). Ils ont comparé leurs résultats avec les deux formulations exactes en nombres entiers mixtes (F1 et F2) proposées par Pattillo *et al. (2013)* [89].

Nous avons implémenté notre algorithme TSQC sous Matlab et nous avons compilé TSQC

dans une machine HP Workstation Z600 avec Intel Xeon X5550, 2.66GHz et 12 GB RAM. Pajouh *et al. (2014)* [84] ont implémenté leur méthode (B&B) en C++, ils ont implémenté les deux formulations F1 et F2 sous C++ et ils ont résolu F1 et F2 en utilisant IBM ILOG CPLEX OPTIMIZER 12.2, ils ont utilisé un DELL POWEREDGE 1950 III compute node, avec dual quad-core Xeon CPU E5430, 2.66 GHz et 16 GB RAM. Veremyev *et al. (2016)* [120] ont utilisé un ordinateur portable Dell avec un processeur Intel Core i7 940XM, 2.13 GHz, L2 8MB et 8 GB RAM, ils ont résolu toutes les formulations en utilisant FICO Xpress Optimizer. Pinto *et al. (2018)* [91]. Ils ont également implémenté leur méthode BRKGA-IG* en C++ avec le compilateur GNU GCC compiler C/C++ version 5.4.0 et ils ont utilisé un ordinateur Lenovo avec un processeur i7-6500U, 2.50 GHz (avec fréquence turbo maxi 3.10 GHz), 8 GB RAM et un système d'exploitation Linux Ubuntu 16.04 LTS avec les fonctions de traitement parallèle désactivées.

Nous avons exécuté TSQC avec les paramètres suivants (10^8 est largement utilisé comme un nombre maximum d'itérations dans les tests des méthodes de résolution du problème de la clique maximum). Nous avons fixé $L = 1000$ sur la base de l'analyse donnée dans la Section 2.4.2, l'étude de Wu & Hao (2013) [124] et selon la nature des instances utilisées.

Les résultats de notre heuristique proposée et leur comparaison avec d'autres méthodes sont résumés dans les tableaux 2.1-2.10. Les tableaux 2.1-2.3 présentent une comparaison entre les résultats obtenus par TSQC et (B&B) sur les instances DIMACS, les tableaux 2.4 et 2.5 comparent les performances de TSQC et BRKGA-IG* sur les instances DIMACS et BHOSLIB respectivement, les tableaux 2.6-2.10 présentent les résultats obtenus par TSQC, F3, F3log, F4, F4log, AlgF3, AlgF4 et BRKGA-IG* sur le second groupe d'instances (réseaux réels). Le temps d'exécution est limité à 3600 secondes (1h, temps limité noté LT) pour toutes les méthodes. Nous notons qu'il y a des instances qui ne sont pas résolues par les autres méthodes, c'est pour cela que nous avons mis (_). Ainsi, le symbole (_) dans les tableaux signifie que cette valeur n'est pas disponible.

TABLE 2.1 – Résumé des résultats sur les instances DIMACS pour $\gamma = 0.85$: la meilleure taille connue de la γ-quasi-clique maximum (ω_γ^*), la taille de la plus grande γ-quasi-clique trouvée (ω_γ), le nombre d'arêtes dans le plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Instances						ω_γ				Temps(s)			
Nom	$\|V\|$	$\|E\|$	Densité	ω_γ^*	TSQC f^*	TSQC	B&B	F1	F2	TSQC	B&B	F1	F2
p_hat300-1	300	10 933	0.244	12	57	**12**	**12**	0	11	0.4170	LT	LT	LT
p_hat300-2	300	21 928	0.489	85	3036	**85**	**85**	49	**85**	0.4698	LT	LT	LT
brock200_2	200	9 876	0.496	19	146	**19**	18	0	16	1.9043	LT	LT	LT
hamming8-4	256	20 864	0.639	35	506	**35**	**35**	1	33	0.2596	LT	LT	LT
keller4	171	9 435	0.649	31	396	**31**	26	0	28	0.2670	LT	LT	LT
brock200_4	200	13 089	0.658	39	630	**39**	34	0	33	0.3610	LT	LT	LT
p_hat300-3	300	33 390	0.744	180	13695	**180**	**180**	169	179	0.2204	LT	LT	LT
brock400_2	400	59 786	0.749	100	4208	**100**	93	0	64	5.3654	LT	LT	LT
brock400_4	400	59 765	0.749	102	4379	**102**	94	0	89	15.7867	LT	LT	LT
p_hat700-1	700	60999	0.2493	19	146	19	_	_	_	6.1647	_	_	_
p_hat700-2	700	121728	0.4976	223	21041	223	_	_	_	0.6611	_	_	_
p_hat700-3	700	183010	0.7480	430	78434	430	_	_	_	0.9280	_	_	_
p_hat1500-2	1500	568960	0.5061	487	100593	487	_	_	_	3.2909	_	_	_
p_hat1500-3	1500	847244	0.7536	943	377532	943	_	_	_	5.5452	_	_	_
keller5	776	225990	0.7515	286	286	286	_	_	_	183.6144	_	_	_

TABLE 2.2 – Résumé des résultats sur les instances DIMACS pour $\gamma = 0.95$: la meilleure taille connue de la γ-quasi-clique maximum (ω_γ^*), la taille de la plus grande γ-quasi-clique trouvée (ω_γ), le nombre d'arêtes dans le plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Instances						ω_γ				Temps(s)			
Nom	$\|V\|$	$\|E\|$	Densité	ω_γ^*	TSQC f^*	TSQC	B&B	F1	F2	TSQC	B&B	F1	F2
p_hat300-1	300	10 933	0.244	9	35	**9**	**9**	2	8	0.6222	LT	LT	LT
p_hat300-2	300	21 928	0.489	41	779	**41**	39	26	39	0.837	LT	LT	LT
brock200_2	200	9 876	0.496	13	75	**13**	12	3	10	2.9906	LT	LT	LT
hamming8-4	256	20 864	0.639	17	130	**17**	**17**	1	0	0.2071	LT	LT	LT
keller4	171	9 435	0.649	15	100	**15**	12	7	12	0.7377	LT	LT	LT
brock200_4	200	13 089	0.658	21	200	**21**	18	0	17	0.9247	LT	LT	LT
p_hat300-3	300	33 390	0.744	71	2371	**71**	**71**	48	69	0.5563	LT	LT	LT
brock400_2	400	59 786	0.749	40	741	**40**	34	0	10	7.3518	LT	LT	LT
brock400_4	400	59 765	0.749	39	704	**39**	34	0	10	3.1727	LT	LT	LT
p_hat700-1	700	60999	0.2493	13	75	13	_	_	_	8.85	_	_	_
p_hat700-2	700	121728	0.4976	96	4332	96	_	_	_	0.6677	_	_	_
p_hat700-3	700	183010	0.7480	176	14630	176	_	_	_	1.157	_	_	_
p_hat1500-2	1500	568960	0.5061	193	17602	193	_	_	_	3.2652	_	_	_
p_hat1500-3	1500	847244	0.7536	351	58354	351	_	_	_	5.036	_	_	_
keller5	776	225990	0.7515	47	1027	47	_	_	_	137.3117	_	_	_

TABLE 2.3 – Résumé des résultats sur les instances DIMACS pour $\gamma = 1$: la meilleure taille connue de la γ-quasi-clique maximum (ω_γ^*), la taille de la plus grande γ-quasi-clique trouvée (ω_γ), le nombre d'arêtes dans le plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Instances						ω_γ				Temps(s)			
Nom	$\|V\|$	$\|E\|$	Densité	ω_γ^*	TSQC f^*	TSQC	B&B	F1	F2	TSQC	B&B	F1	F2
p_hat300-1	300	10 933	0.244	8	28	**8**	**8**	**8**	**8**	0.6434	1772.37	LT	214.84
p_hat300-2	300	21 928	0.489	25	300	**25**	**25**	16	**25**	0.4111	LT	LT	1845.39
brock200_2	200	9 876	0.496	12	66	**12**	11	**12**	**12**	2960.20	LT	3356.52	81.77
hamming8-4	256	20 864	0.639	16	120	**16**	**16**	**16**	**16**	0.2087	LT	51.94	23.23
keller4	171	9 435	0.649	11	55	**11**	9	**11**	**11**	0.4596	LT	2580.04	26.09
brock200_4	200	13 089	0.658	17	120	16(**17**)	16	14	**17**	1.6792 (LT)	LT	LT	341.09
p_hat300-3	300	33 390	0.744	36	630	**36**	35	31	33	1.0056	LT	LT	LT
brock400_2	400	59 786	0.749	29	300	**28(29)**	23	19	23	450.660(LT)	LT	LT	LT
brock400_4	400	59 765	0.749	33	300	**33**	22	16	23	216.6818	LT	LT	LT
p_hat700-1	700	60999	0.2493	11	55	11	_	_	_	7.7848	_	_	_
p_hat700-2	700	121728	0.4976	44	946	44	_	_	_	1.1092	_	_	_
p_hat700-3	700	183010	0.7480	62	1891	62	_	_	_	2.4923	_	_	_
p_hat1500-2	1500	568960	0.5061	65	2080	65	_	_	_	14.9737	_	_	_
p_hat1500-3	1500	847244	0.7536	94	4371	94	_	_	_	307.5155	_	_	_
keller5	776	225990	0.7515	27	351	27	_	_	_	89.8147	_	_	_

TABLE 2.4 – Résumé des résultats de TSQC et BRKGA-IG* sur les instances DIMACS : les meilleures et les moyennes de taille de solution des 10 exécutions et le temps d'exécution.

Instances					BRKGA-IG*			TSQC		
Nom	$\|V\|$	$\|E\|$	Densité	γ	Meilleur	Moyenne	Temps(s)	Meilleur	Moyenne	Temps(s)
brock200_2	200	9876	0.4963	0.800	**24**	**24.00**	1.44	**24**	**24.00**	2.6779
brock400_2	400	59786	0.7492	0.800	186	185.90	32.45	**187**	**187.00**	4.7675
brock800_2	800	208166	0.6513	0.800	93	91.70	31.75	**96**	**96.00**	1193.9
C125.9	125	6963	0.8985	0.999	**34**	**34.00**	1.82	**34**	**34.00**	0.3468
C250.9	250	27984	0.8991	0.999	**44**	**44.00**	5.29	**44**	**44.00**	2.4934
C500.9	500	112332	0.9005	0.999	57	56.70	12.09	**58**	**58.00**	356.3441
C1000.9	1000	450079	0.9011	0.999	67	65.80	43.61	**68**	**68.00**	183.2987
C2000.9	2000	1799532	0.9002	0.999	74	72.20	95.67	**76**	**76.00**	411.4479
DSJC500.5	500	62624	0.5020	0.800	**34**	32.80	5.93	**34**	**34.00**	34.7178
DSJC1000.5	1000	249826	0.5002	0.800	37	36.20	13.34	**41**	**41.00**	1712.3
gen200_p0.9_44	200	17910	0.9000	0.999	42	40.40	3.91	**44**	**44.00**	4.7675
gen400_p0.9_65	400	71820	0.9000	0.999	**66**	62.00	9.24	**66**	**66.00**	16.4846
hamming8-4	256	20864	0.6392	0.800	**71**	**71.00**	4.89	**71**	**71.00**	0.2342
hamming10-4	1024	434176	0.8289	0.950	82	81.20	38.48	**86**	**86.00**	1482.3
keller4	171	9435	0.6491	0.800	**54**	**54.00**	3.86	**54**	**54.00**	2.2055
keller5	776	225990	0.7515	0.800	**486**	**486.00**	110.42	**486**	**486.00**	7.9418
p_hat300-1	300	10933	0.2438	0.500	**64**	**64.00**	8.05	**64**	**64.00**	0.4723
p_hat300-2	300	21928	0.4889	0.800	**114**	**114.00**	9.72	**114**	**114.00**	0.1802
p_hat700-1	700	60999	0.2493	0.500	**119**	118.90	36.10	**119**	**119.00**	1.0874
p_hat700-2	700	121728	0.4976	0.800	**288**	**288.00**	55.17	**288**	**288.00**	0.6674
p_hat1500-2	1500	568960	0.5061	0.800	**642**	**642.00**	274.25	**642**	**642.00**	3.6236

TABLE 2.5 – Résumé des résultats de TSQC et BRKGA-IG* sur les instances BHOSLIB : les meilleures et les moyennes de taille de solution des 10 exécutions et le temps d'exécution.

Instances					BRKGA-IG*			TSQC		
Name	$\|V\|$	$\|E\|$	Densité	γ	Meilleur	Moyenne	Temps(s)	Meilleur	Moyenne	Temps(s)
frb30-15-1	450	83198	0.8235	0.950	59	58.40	11.39	**60**	**60.00**	12.0682
frb30-15-2	450	83151	0.8231	0.950	**58**	56.90	10.82	**58**	**58.00**	8.5024
frb30-15-4	450	83194	0.8235	0.950	60	57.20	11.13	**61**	**61.00**	42.0043
frb30-15-5	450	83231	0.8239	0.950	59	58.60	12.32	**60**	**60.00**	11.8621
frb35-17-1	595	148859	0.8424	0.950	78	76.60	23.07	**79**	**79.00**	470.1865
frb35-17-2	595	148868	0.8424	0.950	74	73.40	20.95	**76**	**76.00**	97.2269
frb35-17-4	595	148873	0.8424	0.950	80	78.60	23.21	**81**	**81.00**	218.7344
frb35-17-5	595	148572	0.8407	0.950	79	77.60	24.84	**80**	**80.00**	13.1226
frb40-19-1	760	247106	0.8568	0.950	111	109.70	44.72	**114**	**114.00**	205.9782
frb40-19-2	760	247157	0.8569	0.950	102	100.40	41.34	**105**	**105.00**	156.6117
frb40-19-4	760	246815	0.8557	0.950	95	94.00	43.53	**97**	**97.00**	687.2537
frb40-19-5	760	246801	0.8557	0.950	99	97.50	43.92	**101**	**101.00**	158.1604
frb45-21-1	945	386854	0.8673	0.950	121	120.00	64.34	**126**	**126.00**	1708.7
frb45-21-2	945	387416	0.8686	0.950	120	118.50	56.12	**124**	**124.00**	421.2255
frb45-21-4	945	387491	0.8687	0.950	128	126.00	74.75	**132**	**132.00**	642.7572
frb45-21-5	945	387461	0.8687	0.950	121	118.40	70.98	**124**	**124.00**	513.6349
frb50-23-1	1150	580603	0.8788	0.950	158	154.80	123.42	**161**	**161.00**	516.6889
frb50-23-2	1150	579824	0.8776	0.950	154	153.10	107.99	**161**	**161.00**	3126.6
frb50-23-4	1150	580417	0.8785	0.950	155	152.80	103.64	**158**	**158.00**	151.5861
frb50-23-5	1150	580640	0.8789	0.950	158	155.20	117.98	**164**	**164.00**	2576
frb53-24-1	1272	714129	0.8834	0.950	199	196.50	159.12	**203**	**203.00**	1043.2
frb53-24-2	1272	714067	0.8834	0.950	169	167.10	136.73	**174**	**174.00**	197.859
frb53-24-4	1272	714048	0.8833	0.950	178	176.50	143.50	**184**	**183.70**	2848.2
frb53-24-5	1272	714130	0.8834	0.950	167	164.50	120.71	**173**	**173.00**	912.0794
frb59-26-1	1534	1049256	0.8924	0.950	239	237.40	264.45	**246**	**246.00**	2574.2
frb59-26-2	1534	1049648	0.8927	0.950	236	232.70	234.40	**243**	**243.00**	666.8573
frb59-26-4	1534	1048800	0.8920	0.950	227	225.60	251.09	**234**	**234.00**	932.1968
frb59-26-5	1534	1049829	0.8929	0.950	224	219.70	198.77	**227**	**227.00**	671.925

TABLE 2.6 – Résumé des résultats sur les réseaux réels pour $\gamma = 0.9$: la meilleure taille connue de la γ-quasi-clique maximum et la taille de la plus grande γ-quasi-clique trouvée par toutes les méthodes (ω^*_γ), le nombre d'arêtes dans la plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Réseaux						Temps(s)									
Nom	$\|V\|$	$\|E\|$	Densité	ω^*_γ	TSQC f^*	TSQC	F1	F2	F3	F3log	F4	F4log	AlgF3	AlgF4	BRKGA-IG*
Geom	7343	11898	0.00044	23	238	9.0058	_	_	16.6	19.7	20.7	115.8	176.4	722.9	_
ca-GRQC	5242	14496	0.00106	49	1062	5.7978	_	_	18.9	21	16.3	45.3	266.8	248.8	2.34
AS-735	7716	13895	0.00047	15	98	2.9892	_	_	19.5	102.1	228.4	2231.5	126	731	_
California	9664	15969	0.00034	12	61	2514.7	_	_	1540.1	LT	LT	LT	LT	2110	_
Email	1133	5451	0.00850	13	74	13.7677	LT	LT	1443.4	LT	2854	LT	LT	531.4	0.17
USAir97	332	2126	0.03869	35	537	15.8223	LT	LT	5.9	4.5	4.5	22.6	20.4	6.8	0.22
Harvard500	500	2043	0.01638	23	231	9.5131	LT	LT	1	2	0.3	0.4	13.6	4.4	0.07
SmallW	396	994	0.01271	11	50	0.1842	LT	2929.5	1.6	7.4	3.5	6.8	7.3	2.9	0.06
netscience	1589	2742	0.00217	21	193	0.447	LT	LT	0.5	0.5	0.3	0.7	9.2	6.5	_
C.Elegans	453	2025	0.01977964	12	60	2.4917	LT	LT	3.4	15.5	5.6	26.9	13.6	6.9	_

TABLE 2.7 – Résumé des résultats sur les réseaux réels pour $\gamma = 0.8$: la meilleure taille connue de la γ-quasi-clique maximum et la taille de la plus grande γ-quasi-clique trouvée par toutes les méthodes (ω^*_γ), le nombre d'arêtes dans la plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Réseaux						Temps(s)									
Nom	$\|V\|$	$\|E\|$	Densité	ω^*_γ	TSQC f^*	TSQC	F1	F2	F3	F3log	F4	F4log	AlgF3	AlgF4	BRKGA-IG*
Geom	7343	11898	0.00044	26	260	4.9076	_	_	35.56	45.63	10.1	11.4	18.4	63.5	_
ca-GRQC	5242	14496	0.00106	52	1085	6.4461	_	_	55.55	72.72	61.8	128.3	41	107.9	_
AS-735	7716	13895	0.00047	19	137	2.3506	_	_	34.36	42.69	15.8	38.4	151.6	479.8	_
California	9664	15969	0.00034	13	69	LT	_	_	3501.2	LT	LT	LT	LT	LT	_
Email	1133	5451	0.00850	15	86	14.7898	_	_	_	_	_	_	_	_	_
USAir97	332	2126	0.03869	43	723	0.1224	_	_	_	_	_	_	_	_	_
Harvard500	500	2043	0.01638	24	234	20.3368	_	_	_	_	_	_	_	_	_
SmallW	396	994	0.01271	14	73	4.2406	_	_	_	_	_	_	_	_	_
netscience	1589	2742	0.00217	22	191	18.9707	_	_	_	_	_	_	_	_	_
C.Elegans	453	2025	0.01978	15	84	0.9971	_	_	_	_	_	_	_	_	_

TABLE 2.8 – Résumé des résultats sur les réseaux réels pour $\gamma = 0.7$: la meilleure taille connue de la γ-quasi-clique maximum et la taille de la plus grande γ-quasi-clique trouvée par toutes les méthodes (ω_γ^*), le nombre d'arêtes dans la plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Réseaux						Temps(s)									
Nom	$\|V\|$	$\|E\|$	Densité	ω_γ^*	TSQC f^*	TSQC	F1	F2	F3	F3log	F4	F4log	AlgF3	AlgF4	BRKGA-IG*
Geom	7343	11898	0.00044	30	306	2.3166	_	_	16.8	16.1	135.5	361.6	167.2	LT	_
ca-GRQC	5242	14496	0.00106	57	1118	6.7603	_	_	342.4	323.8	175.2	475.2	1893.2	400.5	_
AS-735	7716	13895	0.00047	24	197	3.278	_	_	15.6	10.5	13.4	61.7	150.4	LT	_
California	9664	15969	0.00034	14	71	LT	_	_	LT	LT	LT	LT	LT	LT	_
Email	1133	5451	0.00850	17	97	5.6918	_	_	_	_	_	_	_	_	_
USAir97	332	2126	0.03869	49	824	0.122	_	_	_	_	_	_	_	_	_
Harvard500	500	2043	0.01638	26	239	13.8702	_	_	_	_	_	_	_	_	_
SmallW	396	994	0.01271	17	96	1.7762	_	_	_	_	_	_	_	_	_
netscience	1589	2742	0.00217	24	196	15.0881	_	_	_	_	_	_	_	_	_
C.Elegans	453	2025	0.01978	18	108	0.1533	_	_	_	_	_	_	_	_	_

TABLE 2.9 – Résumé des résultats sur les réseaux réels pour $\gamma = 0.6$: la meilleure taille connue de la γ-quasi-clique maximum et la taille de la plus grande γ-quasi-clique trouvée par toutes les méthodes (ω^*_γ), le nombre d'arêtes dans la plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Réseaux						Temps(s)									
Nom	$\|V\|$	$\|E\|$	Densité	ω^*_γ	TSQC f^*	TSQC	F1	F2	F3	F3log	F4	F4log	AlgF3	AlgF4	BRKGA-IG*
Geom	7343	11898	0.00044	36	379	6.8793	_	_	7.4	23	8.4	30	184	LT	_
ca-GRQC	5242	14496	0.00106	63	1173	2.1793	_	_	LT	LT	LT	2082.3	LT	LT	_
AS-735	7716	13895	0.00047	29	247	2.467	_	_	11.6	6.4	7.3	16.5	157.4	LT	_
California	9664	15969	0.00034	14	55	2082.3	_	_	LT	LT	LT	LT	LT	LT	_
Email	1133	5451	0.00850	20	117	0.3255	_	_	_	_	_	_	_	_	_
USAir97	332	2126	0.03869	57	965	0.1253	_	_	_	_	_	_	_	_	_
Harvard500	500	2043	0.01638	29	250	22.6601	_	_	_	_	_	_	_	_	_
SmallW	396	994	0.01271	22	140	0.1561	_	_	_	_	_	_	_	_	_
netscience	1589	2742	0.00217	27	211	4.7596	_	_	_	_	_	_	_	_	_
C.Elegans	453	2025	0.01978	24	166	0.1871	_	_	_	_	_	_	_	_	_

TABLE 2.10 – Résumé des résultats sur les réseaux réels pour $\gamma = 0.5$: la meilleure taille connue de la γ-quasi-clique maximum et la taille de la plus grande γ-quasi-clique trouvée par toutes les méthodes (ω^*_γ), le nombre d'arêtes dans la plus grande γ-quasi-clique trouvée par TSQC (TSQC f^*) et le temps d'exécution.

Réseaux						Temps(s)									
Nom	$\|V\|$	$\|E\|$	Densité	ω^*_γ	TSQC f^*	TSQC	F1	F2	F3	F3log	F4	F4log	AlgF3	AlgF4	BRKGA-IG*
Geom	7343	11898	0.00044	44	479	3.6635	_	_	4.1	10.5	5.1	5.1	255.8	152.1	_
ca-GRQC	5242	14496	0.00106	81	1625	499.1641	_	_	182.1	45.6	119.8	57.6	364.7	676.8	LT
AS-735	7716	13895	0.00047	36	318	2.2245	_	_	4.8	4.2	7.1	6.7	168.4	100.8	_
California	9664	15969	0.00034	26	163	5.0426	_	_	901.7	LT	LT	LT	LT	LT	_
Email	1133	5451	0.00850	25	150	3.4613	LT	LT	3002.9	3543.1	LT	LT	1245	LT	1.48
USAir97	332	2126	0.03869	67	1117	0.1161	LT	LT	0.2	0.7	0.1	0.9	16	2.9	0.53
Harvard500	500	2043	0.01638	37	333	LT	LT	LT	15.9	21	7.7	13.6	86.3	17.7	1.21
SmallW	396	994	0.01271	28	190	0.1303	LT	LT	1.2	1	0.5	1.1	5.7	2.7	0.28
netscience	1589	2742	0.00217	31	233	4.3186	LT	LT	0.4	1.8	0.3	1.5	12	9.7	_
C.Elegans	453	2025	0.01978	30	219	0.1527	LT	LT	1.2	1.7	0.7	1.5	10.3	2.9	_

2.3.1 Résultats et discussion

Vu la nature stochastique de l'algorithme BRKGA-IG* de Pinto *et al. (2018)* [91], ils l'exécutent indépendamment 10 fois pour chaque instance. Nous avons suivi la même idée pour exécuter TSQC en raison de sa nature stochastique également. Pour les instances où nous connaissons la taille de la γ-quasi-clique maximum (la taille de la solution exacte), nous fixons k à la taille de la γ-quasi-clique maximum ; si TSQC ne trouve pas la solution, nous répéterons l'exécution avec $k-1$. Pour les instances où nous ne connaissons pas la plus grande taille de la γ-quasi-clique, nous avons recherché la γ-quasi-clique maximum en incrémentant k chaque fois qu'une k-γ-quasi-clique est trouvée (i.e., nous exécutons TSQC avec un k fixe si une k-γ-quasi-clique est trouvée nous incrémentons k à $k+1$) et on le répète jusqu'à ce que TSQC ne trouve pas une k-γ-quasi-clique. Le temps d'exécution de nos expériences est limité à 1h. Ainsi, si le temps d'exécution atteint 1h, l'exécution s'arrête.

Les tableaux 2.1-2.10 comparent les performances de TSQC avec les autres méthodes, dans les quatre premières colonnes nous avons résumé les informations des instances. Dans les tableaux 2.1-2.3, la cinquième colonne montre la taille du plus grande γ-quasi-clique connue (ω^*_γ), la sixième colonne indique le nombre d'arêtes dans la plus grande γ-quasi-clique trouvée par TSQC i.e., la valeur de la fonction objectif (TSQC f^*), les quatres colonnes suivantes présentent la taille de la γ-quasi-clique (ω_γ) trouvée par la méthode TSQC et les méthodes B&B, F1 et F2 et les quatre dernières colonnes montrent le temps d'exécution de chaque méthode. La cinquième colonne des tableaux 2.4-2.5 indique le seuil γ utilisé par chaque méthode et chaque instance, les trois colonnes suivantes affichent la meilleure et la moyenne des tailles des γ-quasi-cliques trouvées par BRKGA-IG* et les temps d'exécution de la méthode BRKGA-IG* et ceux du TSQC sont donnés dans les trois dernières colonnes. Dans les tableaux 2.6-2.10, les quatre premières colonnes résument les informations des réseaux et la cinquième colonne indique la taille de la plus grande γ-quasi-clique connue et la taille de la γ-quasi-clique trouvée par toutes les méthodes (ω^*_γ) (toutes les méthodes ont trouvé la meilleure solution connue), les colonnes suivantes montrent le nombre d'arêtes dans la plus grande γ-quasi-clique trouvée par TSQC et les dix dernières colonnes montrent les temps d'exécution.

Le temps d'exécution TSQC est le temps moyen de ses 10 exécutions. Nous comparons les résultats de la méthode TSQC proposée avec les résultats des autres méthodes pour prouver la qualité des résultats de TSQC. Nous comparons également leur temps d'exécution comme information complémentaire.

Les tableaux 2.1-2.3 résument les résultats sur les instances DIMACS pour $\gamma = 0.85$, $\gamma = 0.95$ et $\gamma = 1$ respectivement, les résultats de ces tableaux montrent que TSQC peut trouver une γ-quasi-clique avec la même taille que la γ-quasi-clique trouvée par les méthodes exactes ou mieux (des solutions améliorées par rapport à celles des méthodes exactes, il faut noter que

le temps d'exécution est limité à 3600s), TSQC a obtenu des solutions améliorées sur 14 cas. TSQC est parvenu à la solution exacte dans les 10 exécutions, sauf dans deux cas *brock*200_4 et *brock*400_2 pour $\gamma = 1$, où il n'a pas trouvé la solution exacte dans le temps limité, c'est pourquoi nous avons mis dans le tableau le symbole (LT).

Les tableaux 2.4-2.5 résument les résultats sur les instances DIMACS et BHOSLIB respectivement, les résultats révèlent que la qualité des solutions trouvées par notre algorithme TSQC est meilleure ou égale à celle des solutions trouvées par l'algorithme BRKGA-IG*. De plus, TSQC a obtenu la meilleure solution dans 35 cas sur 49 et les meilleures solutions moyennes dans les autres cas.

Les tableaux 2.6-2.10 résument les résultats sur l'ensemble les réseaux réels (deuxième groupe d'instances) pour $\gamma = 0.5$, $\gamma = 0.6$, $\gamma = 0.7$, $\gamma = 0.8$ et $\gamma = 0.9$ respectivement, les résultats montrent que TSQC trouve la solution exacte pour tous les instances et dans tous les 10 exécutions, sauf dans deux cas Californie pour $\gamma = 0.7$ et $\gamma = 0.8$, il n'a pas trouvé la solution exacte dans le temps limité pour les 10 exécutions. Dans ces cas, nous avons mis le symbole (LT).

2.4 Analyse de TSQC

2.4.1 Analyse de tailles des listes tabous et des tabous tenures

Rappelons qu'après chaque $swap(u, v)$ (chaque itération), les sommets u et v sont ajoutés aux listes tabous appelés *tabu_list_u* et *tabu_list_v* respectivement, donc ils ne peuvent pas être ajoutés et supprimés de la solution actuelle S pendant Tu et Tv itérations respectivement (Tu et Tv sont appelés tabous tenures). Ces listes tabous n'ont pas de tailles fixes (c'est-à-dire que l'ajout et le retrait des sommets de ces listes ne dépendent que des tabous tenures). Dans cette section, nous effectuons deux analyses, la première est sur la taille des listes tabous, pour cela nous considérons les cinq cas suivants :

Case 1 : Les tailles des listes tabous *tabu_list_u* et *tabu_list_v* sont non bornées (ce que nous avons utilisé dans notre algorithme).

Case 2 : La taille de *tabu_list_u* égale à 5 et la taille de *tabu_list_v* est non bornée.

Case 3 : La taille de *tabu_list_u* est non bornée et la taille de *tabu_list_v* égale à 5.

Case 4 : La taille de *tabu_list_u* égale à 10 et la taille de *tabu_list_v* est non bornée.

Case 5 : La taille de *tabu_list_u* est non bornée et la taille de *tabu_list_v* égale à 10.

La deuxième analyse se concentre sur la valeur des tabous tenures (Tu et Tv), pour cela nous considérons les trois cas suivants :

Case 1 : $Tu = 10$ et $Tv = 6$.

Case 2 : $Tu = 20$ et $Tv = 12$.

Case 3 : $Tu = l + Random(C-1)$ et $Tv = 0.6 * l + Random(0.6 * C - 1)$, où $l = min\{(\gamma * k * (k-1)/2) - f(S), 10\}$, $C = max\{\lfloor k/40 \rfloor, 6\}$ et $Random(Y)$ génère aléatoirement un entier en $\{0, ..., Y\}$. Dans notre algorithme TSQC, nous avons utilisé ces valeurs de Tu et Tv (voir Section 2.2.4).

Nous étudions l'influence de ces variations sur la performance de l'algorithme basique TSQ, nous nous concentrons sur la fonction d'évaluation f (f est défini à la Section 2.2.2) en termes de nombre d'itérations. Pour cela, nous avons choisi une instance BHOSLIB (frb59-26-1) et un réseau réel (Harvard500). Nous exécutons TSQ 10 fois sur frb59-26-1 pour chaque nombre d'itération avec $k = 246$ et $\gamma = 0.95$, et nous avons pris la moyenne de ces 10 exécutions. Nous exécutons TSQ 50 fois sur Harvard500 pour chaque nombre d'itération avec $k = 29$ et $\gamma = 0.6$, et nous avons pris la moyenne de ces 50 exécutions. Les résultats sont résumés dans les figures. 2.3,2.4,2.5 et 2.6.

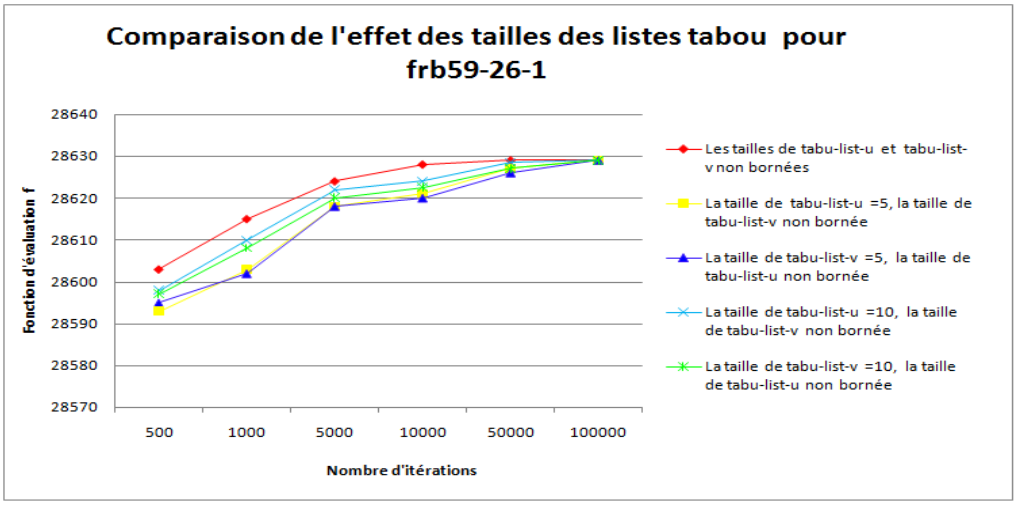


FIGURE 2.3 – Profil d'exécution de TSQ avec différentes tailles de listes tabous sur frb59-26-1($K = 246$ et $\gamma = 0.95$)

Les figures 2.3 et 2.4 montrent les profils d'exécution avec différentes tailles de listes tabous sur frb59-26-1 et Harvard500 respectivement. À partir de ces figures, nous pouvons remarquer que le cas (cas 1) utilisé par notre algorithme TSQC domine les autres cas (2,3,4 et 5).

Les figures 2.5 et 2.5 montrent les profils d'exécution avec différentes valeurs de tabous tenures sur frb59-26-1 et Harvard500 respectivement. À partir de ces figures, nous pouvons remarquer que le cas (cas 3) utilisé par notre algorithme TSQC domine les autres cas (1 et 2).

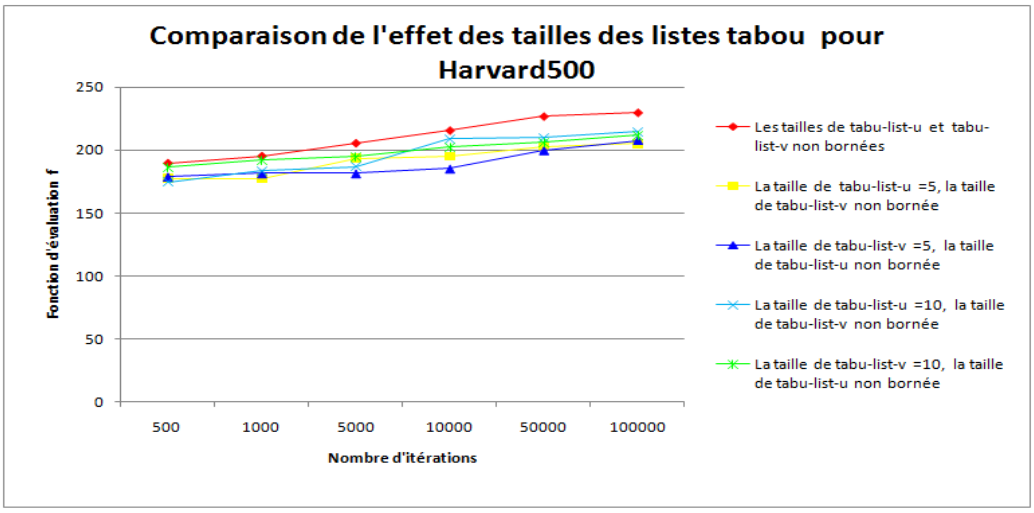


FIGURE 2.4 – Profil d'exécution de TSQ avec différentes tailles de listes tabous sur Harvard500 ($K = 29$ et $\gamma = 0.6$)

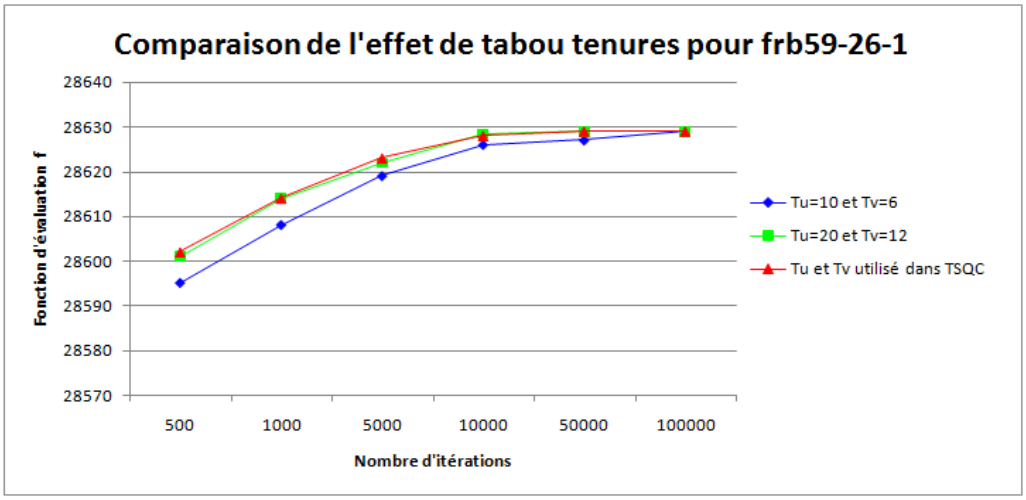


FIGURE 2.5 – Profil d'exécution de TSQ avec différentes valeurs de tabous tenures (Tu et Tv) sur frb59-26-1 ($K = 246$ et $\gamma = 0.95$)

2.4.2 Profondeur de recherche L

Rappelons que si la solution actuelle ne peut être améliorée pendant L itérations consécutives, TSQC redémarre (restart) avec une nouvelle solution initiale générée selon la stratégie présentée à la Section 2.2.5. Il est facile de voir que, le nombre de redémarrages dépend de la valeur de L, ce qui signifie que si L prend une valeur plus grande (resp. plus petite), le nombre de redémarrages sera plus petit (resp. plus grand). Dans cette section, nous effectuons une expérience pour étudier l'effet de la valeur de L sur la convergence de notre algorithme TSQC. Pour ce faire, nous adoptons trois valeurs de L ($L = 500$, $L = 1000$ et $L = 5000$) et 30 instances (DIMACS, BHOSLIB et réseaux réels). Et puis nous exécutons TSQC 10 fois pour chaque valeur de L et pour chaque instance. Nous fixons le nombre maximum d'itérations à

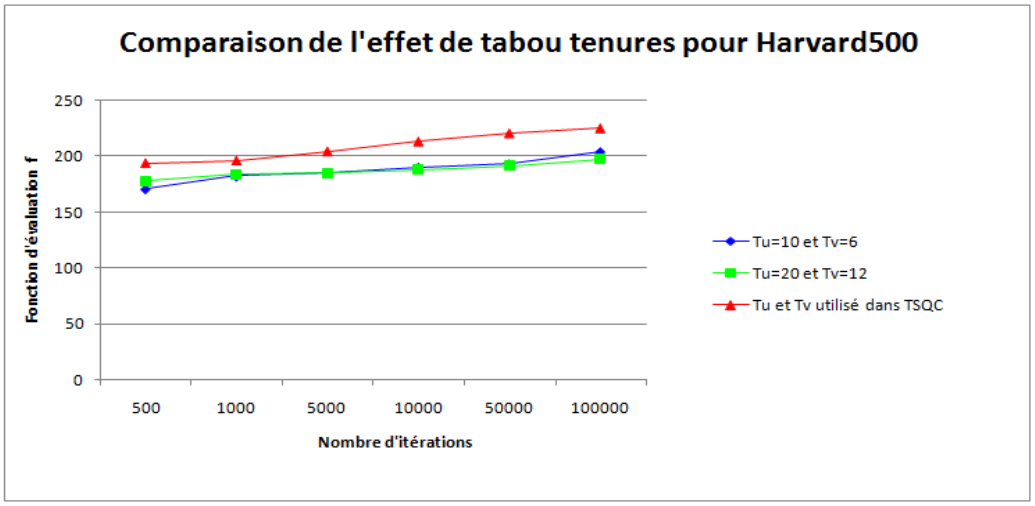


FIGURE 2.6 – Profil d'exécution de TSQ avec différentes valeurs de tabous tenures (Tu et Tv) sur Harvard500 ($K = 29$ et $\gamma = 0.6$)

10^8 (i.e. $It_{mx} = 10^8$).

Le tableau 2.11 présente les résultats comparatifs du TSQC (sur le temps d'exécution) pour $L = 500$, $L = 1000$ et $L = 5000$ (colonnes 4 ,5 et 6 respectivement), colonnes 2 et 3 montre γ et ω^*_γ (nous fixons $k = \omega^*_\gamma$). Nous observons que le temps d'exécution de TSQC avec $L = 1000$ domine les autres dans 20 graphes (instances) et qu'il est dominé par l'un des autres ($L = 500$ ou $L = 5000$) dans les autres graphes sauf dans un seul graphe (frb35-17-2) est dominé par les deux. Cela explique notre choix du paramètre L dans la Section 2.3. Comme il est noté par Wu & Hao (2013) dans [124], L est très sensible à la structure du graphe et il doit être ajusté soigneusement.

2.5 Conclusion

Dans ce chapitre, nous avons présenté l'algorithme TSQC, algorithme de recherche tabou pour le problème de la quasi-clique maximum (Tabu Search for the maximum Quasi-Clique problem en anglais). TSQC est une extension de l'algorithme de recherche tabou adaptative multistart (Adaptive Multistart Tabu Search ,AMTS) proposé par Wu & Hao (2013) [124] pour le problème de la clique maximum. TSQC explore l'espace de recherche composé de tous les sous-ensembles de taille k pour rechercher une quasi-clique de taille k pour un k préalablement fixé. TSQC utilise un voisinage contraint et une liste tabou double. Pour explorer plus efficacement l'espace de recherche, TSQC utilise une stratégie de démarrage multiple.

Notre algorithme proposé TSQC est évalué sur deux groupes d'instances, les instances de référence DIMACS et BHOSLIB du problème de la clique maximum (graphes de différentes densités) et des réseaux réels (graphes sans échelle). Nous avons comparé nos résultats avec les

TABLE 2.11 – Comparaisons du temps d'exécution de TSQC avec $L = 500$, $L = 1000$ et $L = 5000$

Instances			Temps(s)		
Nom	γ	ω^*_γ	TSQC avec L=500	TSQC avec L=1000	TSQC avec L=5000
brock200_2	0.800	24	3.6887	2.8375	**2.7250**
brock400_2	0.800	187	6.3136	5.8673	**5.4461**
C500.9	0.999	58	712.2529	**356.3441**	626.6337
C1000.9	0.999	68	291.2529	**183.2987**	219.2264
C2000.9	0.999	76	512.7008	411.4479	**228.8391**
DSJC500.5	0.800	34	**23.2828**	34.7178	59.7137
gen200_p0.9_44	0.999	44	7.5790	**4.7675**	5.8259
gen400_p0.9_65	0.999	68	37.7320	**16.4846**	52.9674
keller5	0.800	486	9.5891	**7.9418**	72.5436
p_hat1500-2	0.800	642	3.6375	3.6236	**3.5422**
frb30-15-1	0.950	60	26.7970	**12.0682**	19.884
frb30-15-2	0.950	58	9.2861	**8.5024**	24.0486
frb30-15-4	0.950	61	**27.5116**	42.0043	72.2848
frb30-15-5	0.950	60	18.6017	**11.8621**	33.8313
frb35-17-2	0.950	76	**24.4431**	97.2269	82.1867
frb35-17-5	0.950	80	104.9988	**13.1226**	82.1867
frb40-19-1	0.950	114	**70.8805**	205.9782	206.9799
frb40-19-5	0.950	101	454.7533	**158.1604**	302.1339
frb50-23-4	0.950	158	424.2112	**151.5861**	258.1821
frb53-24-2	0.950	174	331.0154	**197.8590**	251.0564
Email	0.9	13	**7.5978**	13.7677	24.2150
Harvard500	0.9	23	12.9295	**9.5131**	44.4002
Email	0.8	15	16.6138	**14.7898**	34.9984
netscience	0.8	22	38.8841	**18.9707**	55.3145
Harvard500	0.7	26	51.1427	**13.8702**	51.9171
netscience	0.7	24	25.8395	**15.0881**	107.2291
Geom	0.6	36	12.3472	**6.8793**	8.2760
AS-735	0.6	29	3.3133	2.4670	**2.0834**
Geom	0.5	44	5.1385	**3.6635**	4.2214
netscience	0.5	31	9.8838	**4.3186**	25.3338

résultats obtenus par les méthodes exactes et heuristiques les plus récentes. Pour les réseaux réels, notre algorithme TSQC est capable de trouver la solution exacte en un temps raisonnable et superforme les méthodes existantes sur les instances DIMACS et BHOSLIB. Notre algorithme TSQC trouve des nouvelles solutions améliorées dans 49 instances (22 instances de DIMACS et 27 instances de BHOSLIB). De plus, notre algorithme TSQC montre une très bonne performance sur ces deux groupes d'instances, ainsi TSQC montre d'excellents résultats sur les graphes de différentes densités et sur les graphes sans échelle (graphes creux). Dans le Chapitre 6, nous utilisons l'algorithme TSQC pour chercher des modules fonctionnels dans un réseau d'interactions protéine-protéine humaine.

Chapitre 3

Une heuristique hybride et sa version parallèle pour le problème de la clique maximum

Dans ce chapitre, nous proposons une heuristique hybride (HHa) et sa version parallèle (une heuristique hybride parallèle (PHa)) pour le problème de la clique maximum. Ces heuristiques sont obtenues par une hybridation entre un algorithme exact dérivé du résultats qui sont donnés par Kilkusts [58] et l'algorithme de recherche tabou adaptative multistart (AMTS) proposé par Wu & Hao (2013) [124]. Des expériences numériques ont montré que nos algorithmes atteignent les meilleurs résultats connus pour les instances de référence DIMACS avec un temps d'exécution compétitif lorsque nous les comparons avec les résultats de l'algorithme AMTS. Une première version de ce chapitre a été acceptée dans la conférence : The 2018 International Conference of Applied and Engineering Mathematics (WCE 2018).

Sommaire

3.1 Principe général des algorithmes HHa et PHa

Comme nous l'avons dit, notre nouvelle heuristique hybride (HHa) est une hybridation entre l'algorithme A^* (l'algorithme 3) et l'algorithme de recherche tabou adaptative multistart (AMTS) proposé par Wu & Hao (2013) [124], l'heuristique hybride parallèle (PHa) est la version parallèle de HHa. Nous notons que la méthode AMTS a donné de très bonnes performances et de très bons résultats concurrentiels par rapport aux autres méthodes du problème de la clique maximum et aux méthodes du problème du stable maximum dans les études comparatives données dans Wu & Hao (2013) [124] et Wu & Hao (2015) [126]. Jin & Hao [53] ont également noté que la méthode AMTS est l'une des heuristiques les plus performantes de la littérature.

L'algorithme A^* est basé sur la propriété 3.1.1. La propriété 3.1.1 est connue et facile à prouver, elle est dérivée du résultats donnés par Kilkusts [58] pour le problème de stabilité. L'algorithme A^* est basé sur deux étapes. La première étape consiste à trouver un ordre de sommets (un ordre basé sur une propriété du graphe ou un ordre quelconque). La deuxième étape de A^* est basée sur la formule de la propriété 3.1.1, le calcul de $\omega(G-v)$ ($\omega_w(G-v)$) se fait séquentiellement du dernier sommet au premier sommet et le calcul de $\omega(G(N(v)))$ ($\omega_w(G(N(v)))$) se fait par un autre algorithme exact.

Propriété 3.1.1. Soit G un graphe et v un sommet de G. Alors, $\omega(G) = Max\{\omega(G(N(v)))+1, \omega(G-v)\}$ et la clique maximum se trouve dans $G(N(v)\cup\{v\})$ ou dans $G-v$. Où $N(v)$ est le voisinage du sommet v dans le graphe G, $G(N(v)\cup\{v\})$ est un sous-graphe de G induit par $N(v)\cup\{v\}$ et $\omega(G)$ la taille de la plus grande clique dans G (le nombre de la clique maximum).

Algorithme 3 L'algorithme A^* pour le problème du nombre de la clique maximum

Entres: Un graphe $G = (V, E)$.
Sorties: $\omega(G)$ /*Le nombre de la clique maximum*/
Début

1. Rechercher un ordre de sommets : l'ordre $[v_1, v_2, ..., v_n]$;
2. Calculer la valeur $\omega(G_i)$ de $i = n-1$ à $i = 1$ par la formule F_ω **:** $\omega(G_i) = Max\{\omega(G(N(v_i))) + 1, \omega(G_i - v_i)\}$./*$\omega(G_i - v_i) = \omega(G_{i+1})$*/

Fin

Remarque 3.1.1. Il est très facile de voir que si on applique une heuristique pour calculer $\omega(G(N(v)))$ (pour chercher la clique maximum dans $G(N(v))$) dans l'algorithme A^*, A^* devient une heuristique (On peut construire une heuristique pour chercher la clique maximum dans G). Sur cette base, nous avons proposé les heuristiques HHa et PHa pour le problème de clique maximum.

La méthode de recherche tabou adaptative multistart (AMTS) est proposée par Wu & Hao (2013) [124] pour le problème de la clique maximum. AMTS est conçue pour approximer le problème de la clique maximum par trouver une série de k-clique dans un graphe $G = (E, V)$ [124] (k-clique est une clique de taille k, voir Wu & Hao (2013) [124]). AMTS est une méthode de recherche tabou, elle commence avec une solution initiale aléatoire S (un sous-ensemble $S \in V$ de taille k, k-clique illégale) et elle essaie de trouver une solution qui a plus d'arêtes (essaie de l'améliorer). À chaque étape AMTS échange (swap) un sommet $u \in S$ avec un sommet $v \in S/V$. AMTS s'arrête après un nombre d'itération préfixé ou lorsque le nombre d'arêtes du sous-ensemble S atteint $(k * (k-1)/2)$ (S devient une clique, une k-clique légale est trouvée).

L'espace de recherche Ω utilisé par AMTS est l'ensemble de tous les sous-ensembles $S \in V$ de taille k. (i.e. $\Omega = \{S \subset V : |S| = k\}$), la fonction d'évaluation de chaque solution $S \in \Omega$ est définie comme suit $f(S) = \sum_{u,v \in S} e_{uv}$ où $e_{uv} = 1$ if $uv \in E$ sinon $e_{uv} = 0$. Pour explorer l'espace de recherche Ω, AMTS utilise une procédure de recherche tabou appelée TS^0. AMTS appelle TS^0 pour améliorer la solution initiale S en maximisant sa fonction d'évaluation $f(S)$. Si TS^0 ne trouve pas une solution améliorée après L itérations (L appelé profondeur de recherche [124]), AMTS construit une nouvelle solution initiale et redémarre (restart) TS^0 avec cette nouvelle solution initiale. Si TS^0 trouve une k-clique, AMTS s'arrête et retourne cette solution. AMTS s'arrête aussi si elle ne trouve pas une k-clique après $Iter_{max}$ itérations ($Iter_{max}$ est le nombre maximum d'itérations autorisé). Pour plus de détails sur la construction d'une nouvelle solution et la technique de redémarrage (voir Wu & Hao (2013) [124]).

Nos algorithmes HHa et PHa comme AMTS sont désignés pour le problème de trouver une k-clique dans un graphe $G = (E, V)$. Le principe général des algorithmes HHa et PHa est défini comme suit : dans la première étape, nous ordonnons les sommets par ordre décroissant de leur degré dans G $(v_1, v_2, ..., v_n)$. Dans la deuxième étape, nous utilisons une généralisation de la propriété 3.1.1 (voir la propriété 3.1.2) et l'algorithme TS^0 pour trouver une $(k-1)$-clique dans les voisinages des sommets v_i $(i = 1...n)$, nous recherchons une $(k-1)$-clique parce que si nous trouvons une $(k-1)$-clique S dans le voisinage $N(v_i)$ alors nous avons trouvé une k-clique $S \cup \{v_i\}$ dans G.

Propriété 3.1.2. Soit $G = (V, E)$ un graphe, v un sommet de G et $(v_1, v_2, ..., v_n)$ un ordre de P-élimination ou un ordre de sommets. Alors, $\omega(G) = Max_{i=1...n}\{\omega(G_i(N(v_i)) + 1\}$ et la clique maximum se trouve dans l'un des sous-graphes $G_i(N(v_i) \cup v_i)$.

Preuve de la propriété 3.1.2 : Soit $G = (V, E)$ un graphe, v un sommet de G et $(v_1, v_2, ..., v_n)$ un ordre de P-élimination ou un ordre de sommets. De la propriété 3.2.1 on a $\omega(G) = Max\{\omega(G(N(v)) + 1, \omega(G - v)\}$

$$\Longleftrightarrow \omega(G) = Max\{\omega(G_1 - v_1), \omega(G_1(N(v_1)) + 1\}$$

$$\Longleftrightarrow \omega(G) = Max\{Max\{\omega(G_2(N(v_2)) + 1, \omega(G_2 - v_2)\}, \omega(G_1(N(v_1)) + 1\}$$

$$\Longleftrightarrow \omega(G) = Max\{Max\{\omega(G_i(N(v_i)) + 1, \omega(G_i - v_i)\}, ..., \omega(G_2(N(v_2)) + 1, \omega(G_1(N(v_1)) + 1\}$$

$$\Longleftrightarrow \omega(G) = Max\{Max\{\omega(G_n(N(v_n)) + 1, \omega(G_n - v_n)\}, ..., \omega(G_i(N(v_i)) + 1, ...,$$

$$, \omega(G_2(N(v_2)) + 1, \omega(G_1(N(v_1)) + 1\}$$

$$\Longleftrightarrow \omega(G) = Max\{\omega(G_n(N(v_n)) + 1, ..., \omega(G_i(N(v_i)) + 1, ..., \omega(G_2(N(v_2)) + 1, \omega(G_1(N(v_1)) + 1\}$$

$$\Longleftrightarrow \omega(G) = Max_{i=1..n}\{\omega(G_i(N(v_i)) + 1\}$$

HHa et PHa explorent l'espace de recherche Ω_G, qui est l'ensemble de tous les sous-ensembles de taille k (comme AMTS), pour trouver une k-clique dans G. Pour explorer Ω_G HHa et PHa explorent un sous-espace de recherche Ω_H de Ω_G, Ω_H est l'ensemble des sous-ensembles de taille $(k-1)$ dans le sous-graphe $H = (V_H, E_H)$ induit par le voisinage du sommet v_i $(N(v_i))$ pour trouver une $(k-1)$-clique ($\Omega_H = \{S \subset V_H : |S| = k-1\}$). La fonction d'évaluation $f(S)$ utilisée par HHa et PHa est la même avec AMTS (le nombre d'arêtes dans le sous-graphe induit par S), $f(S) = \sum_{u,v \in S} e_{uv}$ où $e_{uv} = 1$ if $uv \in E$ sinon $e_{uv} = 0$). Dans la Section 3.2 (Section 3.3 respectivement), nous donnons une description détaillée de l'algorithme HHa (l'algorithme PHa respectivement).

3.2 L'algorithme HHa pour le problème de la clique

La procédure générale de l'algorithme HHa est donnée dans l'Algorithme 4. L'algorithme HHa commence par trier les sommets de G par ordre décroissant de leur degré ($d(v_1) \geq (v_2) \geq (v_2) \geq ... \geq d(v_n)$) où $d(v_i)$ correspond au degré du sommet v_i dans G (ligne 2). Puis HHa recherche une $(k-1)$-clique dans le voisinage de chaque sommet v_i selon l'ordre trouvé à la ligne 2 (lignes 5-18), elle construit un graphe H qui est un sous-graphe de G induit par le voisinage du sommet v_i (ligne 6) et elle recherche une $(k-1)$-clique dans H comme suit : HHa génère une solution initiale S ($S \in \Omega_H$, un sous-ensemble de taille $(k-1)$) à la manière suivante, à partir d'un ensemble vide S et dans $(k-1)$ étapes, elle ajoute le sommet $v \in V_H \backslash S$ qui a le plus grand nombre de voisins (sommets adjacents) dans S, s'il y a plusieurs sommets elle choisit un sommet aléatoirement. Après HHa appelle la procédure TS^0 pour améliorer cette solution initiale S (ligne 8). TS^0 s'arrête lorsqu'elle trouve une $(k-1)$-clique ou si elle ne trouve pas une solution améliorée après L itérations consécutives (L s'appelle la profondeur de recherche [124]).

Si TS^0 trouve une $(k-1)$-clique dans H (le voisinage du sommet v_i), HHa ajoute le sommet v_i à la $(k-1)$-clique trouvée par TS^0 pour la rendre une k-clique, puis elle retourne cette k-clique

et elle s'arrête (lignes 9-11). Si TS^0 ne trouve pas une $(k-1)$-clique dans H, HHa incrémente le i par un pour passer au voisinage du sommet suivant $(N(v_{i+1}))$ dans l'ordre (ligne 13). Si i est arrivé au dernier sommet (le sommet v_n), HHa réinitialise i à 1 pour revenir au premier sommet (v_1) et refaire la recherche (lignes 14-16). Si HHa ne trouve pas une k-clique après It_{mx} itérations (le compteur global d'itérations It atteint le nombre maximum d'itérations autorisées It_{mx}), elle s'arrête et déclare un défaut.

Algorithme 4 L'algorithme HHa pour le problème de la clique maximum

Entres: Un graphe $G = (V, E)$, Un entier k (la taille de la clique), Un entier L (profondeur de recherche), Un entier It_{mx} (le nombre maximum d'itérations autorisées)
Sorties: k-clique si elle est trouvée

1: **Début**
2: Trier les sommets de G par ordre décroissant de leurs degrés /* $d(v_1) \geq (v_2) \geq ... \geq d(v_n)$ où $d(v_i)$ est le degré de v_i dans G*/
3: $It \leftarrow 0$ /* Initialisation du compteur d'itérations */
4: $i \leftarrow 1$ /*Initialiser le sommet v_i à v_1 (i à 1)*/
5: **Tantque** $It < It_{mx}$ **Faire**
6: $H \leftarrow sousgraphe(G, v_i)$ /* Construction du sous-graphe H induite par le voisinage du sommet v_i $(N(v_i))$*/
7: $S \leftarrow$ *solution initiale*$(k-1)$ /* Solution initiale dans H */
8: $S^* \leftarrow TS^0(H, S, k-1, L, It)$ /* Appeler la procédure TS^0 pour améliorer la solution initiale S dans le sous-graphe H */
9: **Si** S^* est une $(k-1)$-clique légale **Alors**
10: $S^* \leftarrow S^* \cup \{v_i\}$
11: **Retourner**(S^*) et **Arrêter**
12: **Sinon**
13: $i \leftarrow i + 1$ /*Incrémenter le compteur des sommets i par un pour aller au voisinage du prochain sommet*/
14: **Si** $i > n$ **Alors**
15: $i \leftarrow 1$ /*Si le compteur des sommets arrive au dernier sommet HHa revient au premier sommet*/
16: **Finsi**
17: **Finsi**
18: **Fin tantque**
19: **Fin**
20: **Retourner**$(Défaut)$

3.3 L'algorithme PHa pour le problème de la clique maximum

La procédure générale de la méthode PHa est donnée dans l'Algorithme 5. Comme nous l'avons dit l'algorithme PHa est la version parallèle de l'algorithme HHa. Il est facile de voir qu'on peut rechercher une $(k-1)$-clique dans les voisinages des sommets $[v_1, ..., v_n]$ $(N(v_i), i =$

$\overline{1...n}$) indépendamment (la recherche d'une $(k-1)$-clique dans le voisinage d'un sommet v ne dépend pas de la recherche d'une $(k-1)$-clique dans les voisinages des autres sommets). Donc nous pouvons rechercher une $(k-1)$-clique dans les voisinages des sommets $[v_1, ..., v_n]$ en parallèle. Sur la base de ces informations, nous sommes arrivés à proposer l'algorithme PHa. La différence entre PHa et HHa est dans la recherche de la $(k-1)$-clique dans les voisinages des sommets $[v_1, ..., v_n]$. HHa recherche une $(k-1)$-clique dans les voisinages des sommets $[v_1, ..., v_n]$ un par un dans l'ordre décroissant de leur degré. PHa recherche une $(k-1)$-clique dans les voisinages des sommets $[v_1, ..., v_n]$ par bloc (le bloc est un ensemble de NW voisinages de sommet), selon l'ordre décroissant de leur degrés i.e., elle essaye de trouver une $(k-1)$-clique dans les NW voisinages de sommet en parallèle, puis elle teste si la $(k-1)$-clique est trouvée, elle s'arrête, sinon elle passe aux prochains NW voisinages de sommet (NW est le nombre de core dans l'ordinateur utilisé i.e., le nombre possible de travailleurs (workers) en parallèle, NW diffère entre les ordinateurs).

Comme HHa, PHa commence par trier les sommets de G (ligne 2) et recherche une $(k-1)$-clique dans les voisinages $[N(v_1), ..., N(v_n)]$ des sommets $[v_1, ..., v_n]$ (lignes 5-23). PHa construit un sous-graphe H pour chaque voisinage de sommet (ligne 8). Pour chaque sous-graphe H, PHa génère une solution initiale et appelle TS^0 pour améliorer cette solution, cette opération se fait en parallèle par bloc de NW sous-graphes (i.e., elle génère une solution initiale et appelle TS^0 pour les NW sous-graphes en parallèle par une boucle parallèle, lignes 7-11).

Après chaque boucle parallèle, PHa prend la meilleure solution trouvée dans la boucle parallèle (dans le bloc de NW sous-graphes) (ligne 12). Si cette solution est une $(k-1)$-clique, PHa ajoute à cette $(k-1)$-clique le sommet qui correspond à son voisinage pour qu'elle devienne une k-clique, puis elle retourne cette k-clique et elle s'arrête (lignes 13-15). Si PHa ne trouve pas une $(k-1)$-clique dans la boucle parallèle (dans le bloc de NW voisinages de sommet (sous-graphiques)) par la procédure TS^0, elle augmente i par NW pour aller au prochain bloc de voisinages de sommet (ligne 17). Si PHa ne trouve pas une $(k-1)$-clique dans tous les voisinages $[N(v_1), ..., N(v_n)]$, elle réinitialise i à 0 pour revenir au premier bloc.

PHa a mis à jour le nombre global d'itérations après chaque boucle parallèle (ligne 21). PHa est comme HHa, si elle ne trouve pas une k-clique après It_{mx} itérations, elle s'arrête et déclare un défaut.

3.4 La procédure TS^0

TS^0 est une adaptation de la méthode de recherche tabou pour le problème de la recherche d'une k'-clique Wu & Hao [124] (nous utilisons la notation k'-clique au lieu de k-clique juste pour faire une différence entre la k-clique recherchée par HHa et PHa et la k-clique recherché

Algorithme 5 L'algorithme PHa pour le problème de la clique maximum

Entres: Un graphe $G = (V, E)$, Un entier k (la taille de la clique), Un entier L (profondeur de recherche), Un entier It_{mx} (le nombre maximum d'itérations autorisées)

Sorties: k-clique si elle est trouvée

1: **Début**
2: Trier les sommets de G par ordre décroissant de leurs degrés /* $d(v_1) \geq (v_2) \geq ... \geq d(v_n)$ où $d(v_i)$ est le degré de v_i dans G*/
3: $It \leftarrow 0$ //* Initialisation du compteur d'itérations */
4: $i \leftarrow 0$ /*Initialiser le compteur des sommets i à 0*/
5: **Tantque** $It < It_{mx}$ **Faire**
6: $Itt(j) \leftarrow 0, j = \overline{1...NW}$ /* Initialiser le compteur d'itérations dans les voisinages à 0*/
7: **Parpour** $j \leftarrow 1$ **à** NW /*Boucle parallèle, NW est le nombre de workers (core) dans la machine (PC)*/
8: $H \leftarrow sousgraphe(G, v_{i+j})$ /* Construction du sous-graphe H induit par le voisinage du sommet v_{i+j} $(N(v_{i+j}))$*/
9: $S \leftarrow$ $solution\ initiale(k-1)$ /* Solution initiale dans H */
10: $S_j \leftarrow \mathrm{TS}^0(H, S, k-1, L, Itt(j))$ /* Appeler la procédure TS^0 pour améliorer la solution initiale S dans le sous-graphe H */
11: **Finparpour**
12: $S^* \leftarrow S^*_j$ /*S^*_j est la solution avec la plus grande valeur de la fonction d'évaluation parmi les NW solutions*/
13: **Si** S^* est une $(k-1)$-clique légale **Alors**
14: $S^* \leftarrow S^* \cup \{v*_j\}$ /* $v*_j$ est le sommet qui correspond au voisinage qui nous donne la $((K-1))$-clique*/
15: **Retourner**(S^*) et **Arrêter**
16: **Sinon**
17: $i \leftarrow i + NW$ /*Incrémentation du compteur ded sommets par NW pour aller au prochain bloc de voisinages du sommet */
18: **Si** $i \geq n$ **Alors**
19: $i \leftarrow 0$ /*Si le compteur des sommets arrive au dernier sommet HHa revient au premier sommet*/
20: **Finsi**
21: $It \leftarrow It + \sum_{j=1}^{NW} Itt(j)$ /*Mise à jour du compteur d'itérations */
22: **Finsi**
23: **Fin tantque**
24: **Fin**
25: **Retourner**$(Défaut)$

par TS^0). TS^0 utilise deux solutions, la solution actuelle S (un sous-ensemble de taille k') et la meilleure solution trouvée jusqu'à présent S^* (chaque fois que TS^0 trouve une solution mieux que S^*, S^* est mis à jour).

Pour améliorer la solution S et explorer plus efficacement l'espace de recherche, TS^0 utilise un voisinage contraint i.e., TS^0 échange (swap) un sommet u d'un sous-ensemble A de S avec un sommet v d'un sous-ensemble B de $V \backslash S$ (A est le sous-ensemble des sommets de S qui ont le minimum de voisins en S et B est le sous-ensemble des sommets de $V \backslash S$ qui ont le maximum de voisins en S). Si TS^0 ne trouve pas une solution améliorée (la recherche est tombée dans un optimum local), TS^0 applique deux règles de sélection déterministes et probabilistes pour échapper à cet optimum local (avec une petite probabilité $P \leq 0.1$, TS^0 accepte une solution avec une grande détérioration et avec une grande probabilité $1 - P$ elle accepte une petite détérioration).

Après chaque opération de swap ($swap(u, v)$, l'opération $swap(u, v)$ échange u avec v), TS^0 ajoute les deux sommets u et v à une liste appelée *tabu_list* (u et v sont appelés tabou), le sommet u ne sort pas de la liste tabou qu'après un nombre d'itérations fixé (Tu itérations), de même le sommet v ne sort pas de la liste tabou qu'après Tv itérations (Tu et Tv sont appelés tabous tenures, voir Glover & Laguna [41]). TS^0 s'arrête si elle trouve une k'-clique et elle retourne cette k'-clique, sinon elle s'arrête si elle ne trouve pas une solution améliorée après L itérations consécutives et elle retourne la meilleure solution trouvée S^* comme résultat (L est un paramètre d'entrée appelé profondeur de recherche [124]). Pour plus de détails sur la procédure TS^0, nous renvoyons le lecteur à voir Wu & Hao [124].

3.5 Résultats expérimentaux

Dans cette section, nous présentons une évaluation de nos méthodes HHa et PHa, cette évaluation se fait sur quelques instances de référence DIMACS (Deuxième défi d'implémentation de DIMACS [54]), nous avons comparé les résultats obtenus par HHa et PHa avec les résultats obtenus par la méthode AMTS. Nous avons implémenté et exécuté les trois méthodes HHa, PHa et AMTS sous MATLAB sur une station HP workstation Z600 avec Windows 7 64 bits, 2,67 GHz CPU, 8 core et 12G RAM.

Nous avons exécuté les méthodes HHa, PHa et AMTS avec les paramètres suivants, pour les trois méthodes, nous avons fixé le nombre maximum d'itérations autorisées à $It_{mx} = Iter_{max} = 10^8$ comme dans Wu & Hao [124] (10^8 est utilisé dans plusieurs articles récents du problème de la clique maximum). Le paramètre L pour la méthode AMTS est défini comme dans Wu & Hao [124], $L = V * K$ pour toutes les instances sauf les instances brock $L = 4 * K$. Dans l'exécution de HHa et PHa L est fixé à 100 sur la base de l'étude des valeurs de L données par Wu &

Hao [124] (puisque HHa et PHa appellent TS^0 avec un sous-graphe induit par un voisinage de sommet donc sa taille est petite, pour cela, nous avons fixé $L = 100$). Pour le paramètre K est fixé à taille de la plus grande clique connue dans les trois méthodes (HHa, PHa et AMTS) comme dans Wu & Hao [124]. Nous avons exécuté PHa avec $NW = 8$ le nombre de core dans notre ordinateur (station).

Le tableau 3.1 résume les résultats obtenus, les trois premières colonnes représentent les informations des instances : le nom, le nombre de sommets ($|V|$) et le nombre d'arêtes ($|E|$), la quatrième colonne représente la taille de la plus grande clique connue (ω^*). Dans les autres colonnes, chaque deux colonnes représente les résultats d'une méthode, la taille de la clique trouvée (ω) et le temps d'exécution en seconde (Time(s)). Vu la nature stochastique des trois méthodes, nous avons exécuté chaque méthode 100 fois et nous avons pris le temps d'exécution moyen.

Les résultats du tableau 3.1 montrent que nos méthodes HHa et PHa obtiennent les meilleurs résultats connus (la plus grande clique connue) dans un temps compétitif par rapport aux résultats de AMTS. Le temps d'exécution de PHa domine le temps d'exécution des méthodes HHa et AMTS dans tous les cas sauf dans trois instances (hamming8-4, p_hat300-1 et p_hat300-2) où le temps d'exécution de HHa domine le temps d'exécution de PHa. Le temps d'exécution de HHa domine le temps d'exécution de AMTS dans 20 instances et le temps d'exécution de AMTS domine le temps d'exécution de HHa dans les autres instances (6 instances).

TABLE 3.1 – Résumé des résultats trouvés par PHa, HHa et AMTS sur certaines instances DIMACS : La taille de la meilleure clique maximum connue (ω^*), la taille de la k-clique trouvée par PHa, HHa et AMTS (ω) et le temps d'exécution de chaque méthode (Temps(s)).

				PHa		HHa		AMTS	
Instances	$\|V\|$	$\|E\|$	ω^*	ω	Temps(s)	ω	Temps(s)	ω	Temps(s)
brock200_2	200	9876	12	12	**82,3874**	12	1348,7	12	394,1068
brock200_4	200	13089	17	17	**42,8446**	17	2330,4	17	3769,4
brock400_2	400	59786	29	29	**350,5512**	29	1533,3717	29	2126,2411
C125.9	125	6963	34*	34	**0,2559**	34	0,4173	34	0,4877
C250.9	250	27984	44*	44	**1,9984**	44	3,1434	44	3,8057
C500.9	500	112332	≥ 57	57	**209,8102**	57	1095,6	57	802,7752
C2000.5	2000,00	999836,00	16*	16	**469,7907**	16	767,4583	16	1594,3923
DSJC500_5	500	125248	13	13	**3,8026**	13	20,4122	13	25,8707
DSJC1000_5	1000	499652	15	15	**132,744**	15	842,6482	15	983,9132
gen200_p0.9_44	200	17910	44	44	**2,2199**	44	3,0129	44	5,1101
gen200_p0.9_55	200	17910	55	55	**2,1187**	55	6,5362	55	12,6776
gen400_p0.9_65	400	71820	65	65	**104,167**	65	300,159	65	141,4964
gen400_p0.9_75	400	71820	75	75	**366,3869**	75	1109,6	75	930,1966
hamming8-4	256	20864	16	16	0,2223	16	**0,2125**	16	0,214
keller4	171	9435	11	11	**0,2043**	11	0,2463	11	0,275
keller5	776	225990	27	27	**43,6932**	27	213,4616	27	233,6089
MANN_a27	378	70551	126	126	**745,6612**	126	3048,3	126	3518,6
p_hat300-1	300	10933	8	8	0,2444	8	**0,1435**	8	0,4656
p_hat300-2	300	21928	25	25	0,3364	25	**0,2908**	25	0,3698
p_hat300-3	300	33390	36	36	**1,4868**	36	2,4858	36	2,2649
p_hat700-1	700	60999	11	11	**6,7408**	11	26,9342	11	15,0993
p_hat700-2	700	121728	44	44	**1,0533**	44	1,1736	44	1,7809
p_hat700-3	700	183010	62*	62	**4,4534**	62	5,6193	62	6,6524
p_hat1500-1	1500	284923	12	12	**656,7791**	12	1262,6066	12	1534,3
p_hat1500-2	1500	568960	65*	65	**11,8139**	65	35,9126	65	41,1728
p_hat1500-3	1500	847244	94*	94	**56,941**	94	255,1121	94	282,3911

3.6 Conclusion

Dans ce chapitre, nous avons présenté deux nouvelles heuristiques pour une approximation du problème de la clique maximum, une heuristique hybride (HHa) et une heuristique hybride parallèle (PHa). HHa et PHa sont désignés pour la recherche d'une k-clique (une clique de taille fixe k) dans un graphe G. HHa et PHa sont une hybridation entre un algorithme exact dérivé du résultats donnés par Kilkusts [58] (l'algorithme (A^*)) et la méthode de recherche tabou adaptative multistart (AMTS) proposé par Wu & Hao [124]. HHa et PHa utilisent la procédure TS^0 utilisée dans la méthode AMTS pour trouver une $(k-1)$-clique dans le voisinage de chaque sommet, ce qui équivaut à trouver une k-clique dans le graphe entier. HHa et PHa donnent de très bons résultats sur certaines instances de référence DIMACS, elles obtiennent le meilleur résultat connu pour le problème de la clique maximum en temps compétitif par rapport à la méthode AMTS (le temps d'exécution de HHa et PHa surpasse favorablement le temps d'exécution de l'AMTS).

Chapitre 4

Un algorithme de recherche tabou multistart et multivoisinage pour le problème de la clique de poids maximum des arêtes

Dans ce chapitre, nous proposons une heuristique de recherche tabou multistart et multivoisinage (MRNTS) pour le problème de la clique de poids maximum des arêtes. Rappelons que le problème de la clique de poids maximum des arêtes (MEWCP) est de déterminer une clique avec le plus grand poids dans un graphe avec des poids sur les arêtes. Notre algorithme MRNTS utilise quatre voisinages pour explorer l'espace de recherche et utilise une stratégie de redémarrage (restart) pour éviter les optimums locaux. La méthode proposée est évaluée sur des instances de référence DIMACS et BHOSLIB et sur quelques réseaux biologiques. Les résultats comparatifs ont montré que MRNTS surperforme l'heuristique la plus récente de MEWCP (LSMR proposé par Li *et al.* (2018) [67]) sur 26 instances DIMACS parmi 32, sur 30 instances BHOSLIB parmi 30 et sur 12 réseaux biologiques parmi 12.

Sommaire

4.1 La méthode de recherche tabou multistart et multivoisinage pour MEWCP

L'algorithme de recherche tabou multistart et multivoisinage (MRNTS) proposé suit la structure de recherche locale itérative présentée par Lourenço *et al.* (2003) [69]. MRNTS commence par construire une solution initiale ou une première solution réalisable appelée S (une clique de poids maximal des arêtes), en utilisant une procédure gloutonne aléatoire (Section 4.1.2). MRNTS explore l'espace de recherche par des étapes d'intensifications et de diversifications (Sections 4.1.4 et 4.1.5) en utilisant quatre voisinages et un opérateur de changement général ($(k,1)-swap$) (Section 4.1.3), les voisinages et l'opérateur de changement partageaient une similarité avec ceux de l'algorithme SBTS proposé par Jin & Hoa (2015) [53] pour le problème du stable maximum. Pour éviter les optimums locaux, MRNTS utilise une liste tabou, une stratégie de vérification de la configuration et une stratégie de redémarrage (restart) (Section 4.1.6 et 4.1.7). MRNTS utilise aussi une nouvelle technique pour réduire l'espace de recherche (voir Section 4.1.1). MRNTS enregistre toujours la meilleure solution dans une variable appelée S^*. Elle s'arrête quand le nombre global d'itérations atteint un nombre fixé (It_{mx}). La procédure générale de l'algorithme MRNTS est résumée dans Algorithme 6.

4.1.1 Espace de recherche et fonction d'évaluation

La méthode MRNTS explore l'espace de recherche Ω qui est composé de toutes les cliques du graphe $G = (V, E, W_e)$ i.e.,

$$\Omega = \{S \subseteq V : u, v \in S, uv \in E\}$$

La fonction d'évaluation de toute solution réalisable (clique) S de Ω est défini par son poids i.e.,

$$f(S) = \sum\nolimits_{u,v \in S, uv \in E} w_{uv}$$

L'objectif est de maximiser la fonction $f(S)$, soit S et S' deux solutions réalisables, on dit que S' est mieux que S si $f(S') > f(S)$.

Algorithme 6 L'algorithme MRNTS pour le problème de la clique de poids maximum des arêtes

Entres: Un graphe $G = (G, E, W_e)$ avec des poids sur les arêtes, Un entier L (profondeur de recherche), Un entier It_{mx} (le nombre maximum d'itérations).

Sorties: La clique du plus grand poids touvée S^*.

Début
$S^i \leftarrow$ *solution initiale* /*Générer une solution initiale Section 4.1.2*/
$It \leftarrow 0$ /* Initialisation du conteur global d'itérations*/
$S^* \leftarrow S^i$ /*Enregistrer la meilleure solution trouvée jusqu'à présent*/
$f^* \leftarrow f(S^*)$ /*Enregistrer le poids de la meilleure solution trouvée jusqu'à présent*/
Tantque $It < It_{mx}$ **Faire**
$R\acute{e}duire(\Omega)$ /*Réduire l'espace de recherche selon S^i Section 4.1.1 */
$S \leftarrow S^i$
$I \leftarrow 0$ /*Initialisation du compteur d'itérations consécutives*/
Tantque $I < L$ **Faire**
Si il existe une solution d'amélioration **Alors**
$S \leftarrow Intensification(S)$ /*Section 4.1.4*/
Sinon
$S \leftarrow Diversification(S)$ /*Section 4.1.5*/
Finsi
Mis à jour de $Tlist$ et $Cnfg$ /*Section 4.1.6*/
Si $f(S) > f^*$ **Alors**
$S^* \leftarrow S$
$I \leftarrow 0$
$f^* \leftarrow f(S^*)$
Sinon
$I \leftarrow I + 1$
Finsi
$It \leftarrow It + 1$
Fin tantque
$S^i \leftarrow$ *nouvelle solution initiale* /* Générer une nouvelle solution initiale suivant la stratégie de construction d'une nouvelle solution initiale, Section 4.1.7*/
Fin tantque
Fin
Retourner(S^*)

Pour une recherche plus efficace, MRNTS réduit l'espace de recherche Ω i.e., après chaque construction de la solution initiale S^i, MRNTS réduit l'espace de recherche Ω selon S^i et elle continue sa recherche dans l'espace de recherche réduit Ω_R. L'espace de recherche réduit Ω_R est défini comme suit :

$$\Omega_R = \{S \subseteq V : u, v \in S, uv \in E, u, v \in SV\}$$

Où $SV = \{u' \in V : v' \in S^i, u'v' \in E\}$, il est facile de voir que $\Omega_R \subseteq \Omega$.

4.1.2 Solution initiale

La solution initiale S^i utilisée par MRNTS est générée de la manière aléatoire suivante :

1. Initialiser un ensemble S à vide.
2. Sélectionner un sommet $v \in V \setminus S$ aléatoirement et ajouter v à S.
3. Supprimer de V tous les sommets qui ne sont pas adjacents à v.
4. Ajouter à S un sommet $v \in V \setminus S$ tel que $\sum_{u \in S, uv \in E} w_{uv}$ est la plus grande somme, s'il y a plusieurs sommets choisir un aléatoirement.
5. Répéter (2) et (3) jusqu'à ce que V devient vide.

4.1.3 Voisinages et opération d'exploration de voisinages

Dans cette section, nous définissons les paramètres et les voisinages utilisés dans notre algorithme MRNTS. Soit S la solution (clique) actuelle et SV l'ensemble des sommets relatif à l'espace de recherche réduit Ω_R (i.e., $VS = \{v' \in V : v' \in S^i, uv \in E\}$). Pour tout sommet $v \in SV$ nous avons désigné un degré $d(v)$ et un degré pondéré $dp(v)$ relatifs à S qui sont définis comme suit :

$$d(v) = |\{u \in S : uv \in E\}|$$

$$dp(v) = \sum_{u \in S, uv \in E} w_{uv}$$

Nous avons aussi désigné un paramètre appelé $dm(u)$ pour tous sommet u de S défini comme suit :

$$dm(u) = |\{v \in SV \setminus S : uv \in E, |S| - d(v) = 1\}|$$

Comme nous l'avons dit MRNTS utilise quatre voisinages qui sont définis comme suit :

$$V0 = \{v \in SV \setminus S : |S| - d(v) = 0\}$$

$$V1 = \{v \in SV \setminus S : |S| - d(v) = 1\}$$

$$V2 = \{v \in SV \setminus S : |S| - d(v) = 2\}$$

$$V3 = \{v \in SV \setminus S : |S| - d(v) \geq 3\}$$

Pour explorer ces voisinages, notre algorithme MRNTS utilise un opérateur de changement général appelé $(k,1) - swap$. Dans chaque étape d'intensification ou de diversification, l'opérateur $(k,1) - swap$ échange (swap en anglais) k sommets de S ($k \geq 0$) avec un seul de $SV \setminus S$ pour obtenir une nouvelle solution S' i.e., $S' = S \bigoplus swap(k,1)$ ou $S' = S \backslash \{u_1, ..., u_k\} \cup \{v\}$ si $k \geq 1$, $S' = S \cup \{v\}$ si $k = 0$.

4.1.4 Intensification

Dans cette section, nous présentons l'étape d'intensification, l'étape d'intensification vise à trouver une nouvelle solution S' mieux que la solution actuelle S ou une solution qui ne détruit pas la solution S. Dans chaque étape d'intensification, MRNTS utilise l'opérateur $(k,1) - swap$ ($0 \leq k \leq 2$) pour échanger k sommets de S avec un seul sommet du voisinage Vk. Pour trouver le meilleur $(k,1) - swap$, nous désignons un gain Δ_v pour tout sommet v de voisinages Vk ($0 \leq k \leq 2$), défini par la différence entre la valeur de la fonction objective de S et celle de la solution S' obtenue par l'échange du sommet $v \in Vk$ avec les k sommets u_i de S qui ne sont pas adjacents avec lui. Δ_v est défini par la formule suivante :

$$\Delta_v = f(S') - f(S) = \begin{cases} dp(v) & si\ k = 0\,; \\ dp(v) - \sum_{i=1}^{k} dp(u_i) & sinon. \end{cases}$$

Comme nous l'avons dit dans l'étape d'intensification MRNTS cherche à trouver une nouvelle solution S' mieux que S ou une solution avec la même valeur de la fonction objective, et cela ne saura pas possible si et seulement s'il existe un sommet v avec $\Delta_v \geq 0$. Donc s'il existe un ou plusieurs sommets qui ont $\Delta_v \geq 0$, l'étape d'intensification est faite comme suit :

- ◇ Exclure de $V0$, $V1$ et $V2$ les sommets v_i qui ont $Cnfg(v_i) = 0$;
- ◇ S'il existe un seul sommet v de $SV \setminus S$ (de $V0 \cup V1 \cup V2$) qui a $\Delta_v \geq 0$, MRNTS échange le sommet v avec les k sommets de S qui ne sont pas adjacents avec lui, en utilisant l'opération $(k,1) - swap$;
- ◇ Sinon, s'il existe plusieurs sommets qui ont $\Delta_{v_i} \geq 0$, calculer Δ_v^* le maximum des Δ_{v_i} et appliquer l'opération $(k,1) - swap$ comme suit :
  - S'il existe un ou plusieurs de $V0$ qui ont $\Delta_{v_i} = \Delta_v^*$, alors choisir un sommet aléatoirement de $V0$ et ajouter ce sommet à S (appliquer le $(0,1) - swap$) ;
  - Sinon, s'il existe un ou plusieurs de $V1$ qui ont $\Delta_{v_i} = \Delta_v^*$, alors choisir un sommet v^* tel que le sommet u^* de S qui n'est pas adjacent avec v^* a le plus grand $dm(u)$

(s'il y a plusieurs choisir un aléatoirement), et échanger v^* avec u^* (appliquer le $(1,1) - swap$) ;

- Sinon choisir un sommet v aléatoirement de $V2$ et échanger le sommet v avec les 2 sommets de S qui ne sont pas adjacents avec lui (appliquer le $(2,1) - swap$).

◇ Sinon il n'existe pas un sommet v qui a $\Delta_v \geq 0$, alors MRNTS contenue sa recherche avec l'étape de diversification, qui est détaillée dans la section suivante.

Après chaque étape d'intensification d, dp, dm, $V0$, $V1$, $V2$ et $V3$ sont mises à jour pour préparer la prochaine itération.

4.1.5 Diversification

Lorsque la solution actuelle ne peut pas être améliorée i.e., il n'existe pas un sommet v qui a $\Delta_v \geq 0$, la recherche est tombée dans un optimum local. Pour sortir de cet optimum local, MRNTS applique l'opération $(k,1) - swap$ pour perturber la solution actuelle, cette étape définit l'objectif de la diversification. Il est facile de voir que s'il n'existe pas un sommet v tel que $\Delta_v \geq 0$ alors $V0 = \emptyset$, donc dans l'étape de diversification, MRNTS utilise les trois voisinages $V1$, $V2$ et $V3$. L'étape de diversification est faite comme suit :

◇ Exclure de $V1$, $V2$ et $V3$ les sommets v_i qui sont marqué tabou i.e $v_i \in Tlist$.

◇ Si $V1$ n'est pas vide, alors choisir un sommet v tel que le sommet u de S qui n'est pas adjacent avec v a le plus grand $dm(u)$, s'il y a plusieurs sommets choisir le sommet qui a le plus grand Δ_v, s'il reste plusieurs choisir un aléatoirement ; puis échanger v avec u (appliquer le $(1,1) - swap$) ;

◇ Sinon, si $V2$ n'est pas vide, alors :

- Avec une probabilité $P = 0.5$ choisir un sommet v de $V2$ qui a le plus grand Δ_v, s'il y a plusieurs choisir un aléatoirement et échanger le sommet v avec les 2 sommets de S qui ne sont pas adjacents avec lui (appliquer le $(2,1) - swap$) ;
- Avec une probabilité $1 - P$, si $V3$ n'est pas, choisir un sommet v aléatoirement de $V3$ et échanger le sommet v avec les k sommets de S qui ne sont pas adjacents avec lui (appliquer le $(k,1) - swap$).

◇ Sinon, si $V3$ n'est pas vide, alors choisir un sommet v aléatoirement de $V3$ et échanger le sommet v avec les k sommets de S qui ne sont pas adjacents avec lui ;

◇ Sinon choisir un sommet v aléatoirement de $SV \setminus S$ et échanger le sommet v avec les sommets de S qui ne sont pas adjacents avec lui.

4.1.6 La liste tabou, critère d'aspiration et la stratégie de vérification de la configuration

L'algorithme proposé MRNTS utilise une liste tabou et une stratégie de vérification de la configuration pour éviter le retour aux solutions visitées précédemment. La stratégie de vérification de la configuration est utilisée dans l'étape d'intensification pour que le sommet qui a sorti de la solution actuelle S ne revient qu'après un sommet v de ses voisins entre ou sorte de S. La stratégie de vérification de la configuration est représentée par un vecteur de 0 et 1 appelé $cnfg$ de taille $|V|$. Pour tout sommet v de V, $cnfg(v) = 1$ si le sommet v peut entrer à la solution actuelle, $cnfg(v) = 0$ si le sommet v ne peut pas entrer à la solution actuelle. Le vecteur $cnfg$ est maintenu comme suit :

1. Initialement, $cnfg(v) = 1$ pour tout sommet $v \in V$.
2. Après chaque itération, MRNTS change la valeur de la configuration du sommet u (le sommet qui a sorti de solution actuelle) de 1 à 0 ($cnfg(u) = 0$) et $cnfg(v')$ de 0 à 1 ($cnfg(v) = 1$) pour tout sommet $v' \in N(u)$ ($N(u)$ est le voisinage du sommet u).

La liste tabou est utilisée dans l'étape de diversification, après chaque itération, MRNTS ajoute les sommets u_i qui ont sorti de la solution actuelle S à une liste tabou appelé $Tlist$ pour qu'ils ne peuvent pas entrer à S pendant T itérations (T est appelé tabou tenure). La valeur de tabou tenure T est adaptée comme dans l'algorithme SBTS proposé par Jin & Hoa (2015) [53] pour le problème du stable maximum. Après l'opération $(k, 1) - swap$, la valeur de T des sommets u_i qui ont sorti de S est ajustée selon le sommet v qui a entré à S comme suit :

- $T = 10 + Random(|V1|)$ si v appartient au voisinage $V1$ où $Random(X)$ génère aléatoirement un entier en $\{0, ..., X\}$.
- $T = 10 + Random(|V2|)$ si v appartient au voisinage $V2$ où $Random(X)$.
- $T = 7$ sinon.

4.1.7 Construction de la nouvelle solution pour un nouveau restart

Pour sortir de l'optimum local et encourager la recherche à visiter des nouvelles zones de recherche, MRNTS utilise le mécanisme proposé par Wu & Hao (2013) [124] (il est aussi utilisé par l'algorithme TSQC proposé dans le Chapitre 2). Wu & Hao ont désigné à chaque sommet v de V une fonction g_v appelée mémoire de fréquence à long terme (voir Wu & Hao (2013) [124]). la valeur de g_v représente le nombre de mouvements du sommet v, g_v est mis à jour comme suit :

1. Initialement, $g_v = 0$, $\forall v \in V$.

2. Chaque fois que le sommet v entre ou sorte de la solution actuelle S, MRNTS mis à jour g_v par $g_v + 1$ ($g_v = g_v + 1$).
3. Si $g_v > |S^*|$ pour tout sommet $v \in V$, MRNTS réinitialise $g_v = 0$ pour tout sommet v de V.

La construction de la nouvelle solution est faite comme suit :

1. Initialiser un ensemble S à vide.
2. Sélectionner le sommet v de V qui a la plus petite valeur de g_v.
3. Supprimer de V tous les sommets qui ne sont pas adjacents à v.
4. Ajouter à S un sommet $v \in V \setminus S$ tel que $\sum_{u \in S, uv \in E} w_{uv}$ est la plus grande somme, s'il y a plusieurs sommets choisir un aléatoirement.
5. Répéter (3) et (4) jusqu'à ce que V devient vide.

4.2 Résultats expérimentaux

Dans cette section, une étude comparative est présentée, nous avons comparé l'algorithme proposé (MRNTS) avec l'heuristique la plus récente (LSMR) proposé par Li *et al.* (2018) [67]. Nous notons que l'algorithme LSMR a montré de bons résultats dans l'étude comparative donnée par Li *et al.* (2018) [67]. Nous avons comparé notre algorithme MRNTS et LSMR sur les instances de référence de DIMACS et BHOSLIB du problème de la clique maximum (DIMACS 1996 [54], BHOSLIB [128]), et aussi sur quelques réseaux biologiques (nous avons téléchargé ces réseaux biologiques à partir du référentiel des données des réseaux [98]). Les réseaux biologiques sont à l'origine avec poids des arêtes et les instances de DIMACS et BHOSLIB sont des graphes sans poids, et pour les rendre avec poids des arêtes, nous utilisons la règle utilisée par Zhang *et al.* (2018) [132]. Soit G un graphe, pour toutes arêtes uv de G :

$$w_{u,v} = \frac{|N(v) \cap N(u)|}{|N(v) \cup N(u)|}$$

Où $N(v)$ est le voisinage du sommet v dans le graphe G i.e. l'ensemble des sommets adjacent à V. Cette règle est utilisée par Zhang *et al.* (2018) [132] dans leur algorithme de détection des complexes dans les réseaux interactions protéine-protéine, ce qui est pratiquement significatif.

Nous avons implémenté les algorithmes MRNTS et LSMR sous MATLAB, et nous les avons compilés sur la même station, HP Workstation Z600 avec Intel Xeon X5550, 2.66GHz et 12 GB RAM. Nous avons exécuté MRNTS avec $It_{mx} = 10^8$ et $L = 1000$ pour les instances DIMACS, $L = 100$ pour les instances BHOSLIB et $L = 100$ les réseaux biologiques. Vu que les deux algorithmes sont de nature stochastique, nous avons exécuté chaque algorithme 10 fois pour

chaque instance, et avons pris la moyenne des valeurs des fonctions objectives des 10 solutions (noté par Moyenne) et la valeur de la fonction objective de la meilleure solution (noté par Meilleur). Le temps exécution de chaque algorithme est fixé à $1000s$. Les résultats sont résumés dans les Tableaux 4.1, 4.2 et 4.3.

Les Tableaux 4.1, 4.2 et 4.3 résume les résultats comparatifs de notre algorithme MRNTS et l'algorithme LSMR sur les instances DIMACS, BHOSLIB et les réseaux biologiques respectivement. Les trois premières colonnes présentent les informations des instances (nom, nombre des sommets $|V|$ et nombre des arêtes $|E|$). La quatrième colonne montre les moyennes des valeurs des fonctions objectives des 10 solutions trouvées par MRNTS (la sixième colonne montre celles de LSMR) et la cinquième colonne montre les valeurs des fonctions objectives des meilleures solutions trouvées par MRNTS (la septième colonne montre celles de LSMR).

Le Tableau 4.1 montre que notre algorithme MRNTS domine l'algorithme LSMR sur 26 instances DIMACS parmi 32 dans les moyennes et sur 23 instances parmi 32 dans les meilleures solutions, et l'algorithme LSMR domine MRNTS sur 5 instances dans les moyennes et sur 8 dans les meilleures solutions. Le Tableau 4.2 montre que MRNTS domine LSMR sur toutes les instances BHOSLIB dans les moyennes et que LSMR domine MRNTS sur 2 instances dans les meilleures solutions. Le Tableau 4.3 montre que l'algorithme MRNTS domine l'algorithme LSMR sur tous les réseaux biologiques dans les moyennes et que LSMR domine MRNTS sur un seul réseaux dans les meilleures solutions. Les résultats ont prouvé que notre algorithme MRNTS donne de bons résultats par rapport à l'algorithme LSMR dans les instances DIMACS, BHOSLIB et les réseaux biologiques.

TABLE 4.1 – Résultats comparatifs des algorithmes MRNTS et LSMR sur les instances DIMACS

Les intances			MRNTS		LSMR	
Nom	$\|V\|$	$\|E\|$	Moyenne	Meilleur	Moyenne	Meilleur
brock200_2	200	9876	**17.1348**	**20.0306**	14.5967	16.7343
brock200_4	200	13089	**54.0293**	**63.4505**	44.1427	48.9277
brock400_2	400	59786	**133.7931**	**146.2657**	112.2918	132.0438
brock400_4	400	59765	**136.7848**	**158.6679**	120.2794	133.2252
brock800_2	800	208166	**70.4243**	**78.1446**	53.8605	62.0515
brock800_4	800	207643	**66.5402**	**77.6566**	57.4122	61.5526
C125.9	125	6963	423.0334	445.5492	**439.0521**	**473.4341**
C250.9	250	27984	**618.1367**	659.3436	572.6013	**659.4692**
C500.9	500	112332	849.1070	**913.5866**	**874.1904**	752.5419
C1000.9	1000	450079	**1152.0**	**1245.9**	1023.2	1158.0
C2000.5	2000	999836	**25.1229**	**31.8098**	21.7468	26.9537
DSJC500.5	500	62624	**20.0265**	**24.0191**	16.9897	20.3082
DSJC1000.5	1000	249826	**22.3497**	**27.8547**	20.1887	23.5348
gen200_p0.9_44	200	17910	463.9191	480.5006	**504.5517**	**563.5358**
gen200_p0.9_55	200	17910	578.9546	600.0260	**1180.2**	**1214.5**
gen400_p0.9_55	400	71820	**570.6520**	**598.0639**	552.2514	567.5331
gen400_p0.9_65	400	71820	**751.9313**	**834.6351**	691.5445	735.6542
gen400_p0.9_75	400	71820	**884.1155**	**966.1539**	797.2580	874.0316
hamming8-4	256	20864	**57.6737**	**57.6737**	**57.6737**	**57.6737**
MANN_a27	378	70551	**7562.7**	**7681.5**	7480.0	7562.6
MANN_a45	1035	533115	**56963.0**	**57458.0**	25375.5	25375.5
keller4	171	9435	18.1520	18.2352	**19.7299**	**26.9135**
keller5	776	225990	**84.1948**	98.8417	83.7508	**99.4450**
p_hat300-1	300	10933	**6.4925**	**6.5278**	4.8968	5.1258
p_hat300-2	300	21928	**160.5093**	**169.5462**	113.6551	134.9935
p_hat300-3	300	33390	**361.8385**	387.9860	343.3679	**430.0194**
p_hat700-1	700	60999	**9.5104**	**12.8857**	5.9723	8.5296
p_hat700-2	700	121728	**464.4277**	501.6735	368.2757	**512.2509**
p_hat700-3	700	183010	**1211.9**	**1269.0**	1012.2	1144.4
p_hat1500-1	1500	284923	**9.5928**	**10.8754**	7.9631	10.5531
p_hat1500-2	1500	568960	**1014.1**	**1141.1**	802.0704	1067.4
p_hat1500-3	1500	847244	**2394.6**	**2638.4**	1197.4	1208.0

TABLE 4.2 – Résultats comparatifs des algorithmes MRNTS et LSMR sur les instances BHOSLIB

Les intances			MRNTS		LSMR	
Nom	$\|V\|$	$\|E\|$	Moyenne	Meilleur	Moyenne	Meilleur
frb30-15-1	450	83198	**971.0719**	**4363.4**	105.227	556.5081
frb30-15-2	450	83151	**193.74**	**217.5092**	176.9884	202.6954
frb30-15-3	450	83216	**180.9510**	**202.6022**	178.4362	202.5866
frb30-15-4	450	83194	**198.9213**	**219.1934**	171.8745	203.2924
frb30-15-5	450	83231	**199.8893**	**218.8003**	176.2978	180.1404
frb35-17-1	595	148859	**279.6836**	**305.4183**	252.3287	286.6027
frb35-17-2	595	148868	**264.1881**	**328.5704**	234.6710	266.4511
frb35-17-3	595	148784	**267.5293**	**289.9552**	250.0593	289.0359
frb35-17-4	595	148873	**270.9269**	**288.0147**	238.2269	249.1349
frb35-17-5	595	148572	**263.6488**	285.5262	251.3110	**286.6027**
frb40-19-1	760	247106	**348.7145**	**388.3007**	306.3172	342.7161
frb40-19-2	760	247157	**358.0299**	**364.1087**	341.5633	283.9155
frb40-19-4	760	246815	**345.1118**	**409.3629**	344.1165	386.7281
frb40-19-5	760	246801	**327.2346**	**385.1560**	314.1872	362.6799
frb45-21-1	945	386854	**442.3280**	**496.8870**	424.9814	470.5797
frb45-21-2	945	387416	**458.3078**	**498.3265**	411.2046	473.2772
frb45-21-4	945	387491	**461.3880**	**501.2157**	426.6021	476.1399
frb45-21-5	945	387461	**458.3110**	**553.5688**	427.9758	470.7328
frb50-23-1	1150	580603	**543.4984**	**598.8275**	500.5817	434.5690
frb50-23-2	1150	579824	**567.9089**	598.4274	546.1950	**659.7924**
frb50-23-4	1150	580417	**555.5609**	**626.3957**	511.8017	513.9366
frb50-23-5	1150	580640	**599.7871**	**691.5066**	513.3363	570.5948
frb53-24-1	1272	714129	**595.7643**	**636.6124**	576.1421	636.1641
frb53-24-2	1272	714067	**605.5037**	**668.0490**	561.6744	568.9681
frb53-24-4	1272	714048	**613.2495**	**666.7884**	586.4964	636.9198
frb53-24-5	1272	714130	**631.7201**	**699.9190**	598.6707	637.4351
frb59-26-1	1534	1049256	**760.3076**	**819.7272**	718.9118	667.8001
frb59-26-2	1534	1049648	**760.5581**	**857.1522**	730.2092	786.1414
frb59-26-4	1534	1048800	**743.1556**	**820.1371**	732.3131	820.1245
frb59-26-5	1534	1049829	**731.6634**	**818.0293**	728.1595	817.2347

TABLE 4.3 – Résultats comparatifs des algorithmes MRNTS et LSMR sur les réseaux biologiques

Les instances			MRNTS		LSMR	
Nom	$\|V\|$	$\|E\|$	Moyenne	Meilleur	Moyenne	Meilleur
bio-CE-CX	15229	245952	**2597.0**	**10598.0**	88.3975	635.5119
bio-CE-GN	2220	53683	**298.9874**	**398.0299**	112.5773	359.6499
bio-CE-GT	924	3239	**65.2686**	**69.3283**	7.2198	13.9429
bio-CE-HT	2617	2985	**12.7095**	**13.9429**	7.2198	8.4567
bio-DM-CX	4040	76717	**3038.6**	**7013.3**	140.9016	683.9763
bio-DR-CX	3289	84940	**9455.5**	**13068.0**	1338.6	**13068.0**
bio-HS-CX	4413	108818	**9237.5**	**12768.0**	83.7948	329.4734
bio-HS-HT	2570	13691	**2896.6**	**2896.6**	295.5976	1151.2
bio-HS-LC	4227	39484	**3291.0**	**5196.3**	81.8574	320.3115
bio-SC-GT	1716	33987	**1414.9**	2125.4	782.6367	**2166.4**
bio-SC-HT	2084	63027	**4615.0**	**6308.5**	1601.0	5621.3
bio-SC-LC	2004	20452	**372.6931**	**496.9940**	27.4620	77.5886

4.3 Conclusion

Dans ce chapitre, nous avons présenté un algorithme de recherche tabou multistart et multivoisinage (MRNTS) pour le problème de la clique de poids maximum des arêtes (Miltistart and multineighborhood tabu search for the maximum edge weighted clique problem en anglais). MRNTS utilise quatre voisinages pour explorer l'espace de recherche et une liste tabou et une stratégie de vérification de la configuration pour éviter les cycles de recherche. Une stratégie adaptative de redémarrage (restart) est aussi utilisée dans MRNTS pour encourager MRNTS à visiter des nouvelles zones de recherche. Nous avons testé notre algorithme sur trois ensembles d'instances (32 instances DIMACS, 30 instances BHOSLIB et 12 réseaux biologiques). Les résultats ont montré que notre heuristique (MRNTS) donne de bons résultats par rapport à l'heuristique la plus récente (LSMR) de MEWCP sur les instances des trois ensembles instances (DIMACS, BHOSLIB et les réseaux biologiques). En outre, MRNTS domine LSMR sur 26 instances DIMACS parmi 32, sur toutes les instances BHOSLIB et sur tous les réseaux biologiques.

Chapitre 5

Contribution à la résolution du problème du sous-graphe de poids maximum des arêtes dans des réseaux biologiques

Soit $G = (V, E, W_e)$ un graphe avec des poids sur les arêtes, avec V l'ensemble des sommets, E l'ensemble des arêtes et W_e une fonction de pondération qui associe à chaque arête $e \in E$ un poids positif $w_e \geq 0$. Le problème du sous-graphe de poids maximum des arêtes (MEWS) vise à trouver un sous-graphe $H = (S, F)$ de G tel que la somme des poids des arêtes de F est la plus grande et le nombre des sommets de S égal à k, où k un entier ($k \geq 3$), il est prouvé que le MEWS est un problème NP-Complet. Dans ce chapitre, nous proposons un algorithme de recherche tabou multistart pour le MEWS. Nous avons utilisé notre algorithme pour résoudre le MEWS dans des réseaux biologiques, qui ont des vrais poids sur les arêtes. Les résultats ont montré que notre algorithme surperforme les autres heuristiques de l'état de l'art [Le contenu de ce chapitre est accepté au Colloque sur l'Optimisation et les Systèmes d'Information (COSI 2019)].

Sommaire

5.1 Introduction

Soit $G = (V, E, W_e)$ un graphe avec des poids sur les arêtes où $V = \{1, 2, ..., n\}$ l'ensemble des sommets, $E \subseteq V \times V$ l'ensemble des arêtes et $W_e : E \to Z^+$ une fonction de pondération qui associe à chaque arête $uv \in E$ un poids positif w_{uv}, et soit k un entier tel que $k \geq 3$, le problème du sous-graphe de poids maximum des arêtes (the Maximum Edge-Weighted Subgraph problem (MEWS) en anglais) cherche à trouver un sous-graphe $H = (S, F)$ de G tel que la somme des poids des arêtes de F est la plus grande somme possible et le nombre des sommets de S égal à k, le MEWS est connu aussi sous le nom du problème du k-sous-graphe le plus lourd (the Heaviest k-Subgraph Problem en anglais). Si tous les poids des arêtes sont égaux à 1, le MEWS devient équivalent au problème du k-sous-graphe le plus dense. La version de décision de MEWS est NP-Complet [3]. Le modèle mathématique de MEWS est similaire au problème de la diversité maximum (MDP) [18], Brimberg *et al. (2009)* [18] ont noté que le MEWS est plus général que le MDP puisque les poids des arêtes n'ont pas besoin de représenter les distances et que le graphe n'a pas besoin d'être entièrement connecté. Le problème du sous-graphe de poids maximum des arêtes a plusieurs applications dans des différents domaines tel que les réseaux de télécommunications, les réseaux sociaux et les réseaux biologiques. Le MEWS est un cas particulier du problème de la clique de poids maximum des arêtes [48].

Vu l'importance du problème de sous-graphe de poids maximum des arêtes, un grand nombre de travaux ont été publiés sur ce problème. Macambira & de Meneses (1998) [72] ont proposé une heuristique gloutonne adaptative aléatoire (GRASP) pour le MEWS, Macambira (2002)[71] a proposé un algorithme de recherche tabou pour résoudre le MEWS, Billionnet (2005) [12] a proposé différentes formulations pour résoudre le MEWS, Brimberg *et al.* (2009) [18] ont proposé une heuristique recherche à voisinage variable (VNS) pour le MEWS, Letsios *et al.* (2016) [63] ont proposé un algorithme exact pour le MEWS dans les médias sociaux, Singh *et al.* (2019) [111] ont proposé une approximation de la solution de MEWS, ils ont utilisé une heuristique gloutonne pour réduire la taille de graphe puis ils ont utilisé ce graphe comme une entrée dans un algorithme Branch and Bound. Plusieurs travaux ont aussi proposé pour le problème équivalent, le problème de la diversité maximum Palubeckis (2007) [85], Gallego *et al.* (2009) [37], Wu & Hao (2013) [125], Wang *et al.* (2014) [122].

Dans ce chapitre, nous proposons un algorithme de recherche tabou multistart (MTSEWS) pour le MEWS, nous avons utilisé notre algorithme pour résoudre le MEWS dans 12 réseaux biologiques avec des poids sur les arêtes, puisque l'identification des complexes protéiques et des modules fonctionnels a été formulée comme un problème du sous-graphe de poids maximum des arêtes [68]. Nous avons comparé les résultats de notre algorithme avec deux heuristiques, l'algorithme de recherche tabou proposé par Macambira [71] et l'algorithme de recherche à

voisinage variable (VNS) proposé par Brimberg *et al.* [18]. Les résultats ont montré que notre algorithme surperforme les autres heuristiques dans la qualité des solutions. Dans le reste du papier, nous donnons une présentation détaillée de notre algorithme Section 5.2 et une étude comparative Section 5.3.

5.2 L'algorithme proposé MTSEWS

Le principe général de notre algorithme MTSEWS est résumé dans l'Algorithme 7. MTSEWS commence par une solution initiale générée par une procédure aléatoire (un sous-ensemble $S \subset V$ de taille k, Section 5.2.1). À partir de cette solution initiale MTSEWS utilise une procédure appelée BTS pour chercher à trouver une solution améliorée (Section 5.2.2). BTS utilise deux sous-ensembles critiques $A \subset S$ et $B \subset V \setminus S$, chaque itération BTS change un sommet de A avec un sommet de B (A et B sont des sous-ensembles contraints). Si BTS ne peut pas améliorer la solutions actuelle S pendant L itérations consécutives, MTSEWS redémarre (restart) BTS avec une nouvelle solution initiale générée suivant une stratégie de construction d'une nouvelle solution initiale pour un nouveau restart (voir Section 5.2.3). Après chaque appel de BTS, MTSEWS compare la solution S trouvée par BTS avec la meilleure solution trouvée jusqu'a présent S^* et elle fait le mis à jour de S^*. L'algorithme MTSEWS s'arrête quand le nombre d'itérations atteint It_{mx} (It_{mx} est le nombre maximum d'itérations autorisées).

L'espace de recherche Ω exploré par notre algorithme MTSEWS est l'ensemble de tous les sous-ensembles de taille k (l'ensemble de toutes les solutions réalisables), soit $G = (V, E, W_e)$ un graphe avec des poids sur les arêtes, $\Omega = \{S \subset V : |S| = k\}$. L'évaluation de chaque solution S est basée sur la somme des poids des arêtes du sous-graphe induit par S, i.e., la fonction d'évaluation est définie comme suit :

$$f(S) = \sum\nolimits_{u,v \in S, uv \in E} w_{uv}$$

La fonction $f(S)$ est à maximisée, on dit qu'une solution S' est mieux que S si $f(S') > f(S)$.

Algorithme 7 L'algorithme MTSEWS

Entres: Un graphe G avec des poids sur les arêtes, Un entier k (la taille du sous-graphe), Un entier L (profondeur de recherche), Un entier It_{mx} (le nombre maximum d'itérations).
Sorties: Le k-sous-graphe de poids maximum des arêtes.

```
Début
S ← solution initiale /*Générer une solution initiale, S est la solution actuelle*/
It ← 0 /* Initialisation du conteur global d'itérations*/
S* ← S /*Enregistrer la meilleure solution trouvée jusqu'à présent S**/
Tantque It < It_mx Faire
  S ← BTS(G, S, k, L, It)
  Si f(S) > f(S*) Alors
    S* ← S
  Finsi
  S ← nouvelle solution initiale /* Générer une nouvelle solution initiale suivant la stratégie de construction d'une nouvelle solution initiale*/
Fin tantque
Fin
Retourner(S*)
```

5.2.1 Solution initiale

La solution initiale utilisée par notre algorithme MTSEWS est générée de la manière suivante :

1. Initialiser un ensemble S à vide.
2. Sélectionner un sommet $v \in V \setminus S$ aléatoirement et ajouter v à S.
3. Ajouter à S un sommet $v \in V \setminus S$ tel que $\sum_{u \in S, uv \in E} w_{uv}$ est la plus grande somme, s'il y a plusieurs sommets choisir un aléatoirement.
4. Tant que $|S| < k$ répéter (3) et retourner S.

5.2.2 L'algorithme BTS

L'algorithme BTS est un algorithme de recherche tabou, le principe général de l'algorithme BTS est résumé dans l'Algorithme 8. BTS commence par enregistrer la meilleure solution trouvée jusqu'à présent S^* et sa valeur de la fonction d'évaluation f^*, et initialise le conteur d'itérations consécutives I à 0. S représente la solution actuelle. Pour améliorer la solution actuelle BST applique une série d'intensification et de diversification (Voir Section 5.2.2). BTS utilise deux listes tabous pour que les sommets qui sont entrés à (sorti de) la solution actuelle ne puissent pas sortir de (revenir à) la solution actuelle rapidement, nous avons aussi ajouté une stratégie de vérification de la configuration à BTS, cette stratégie est utilisée par les algorithmes de recherche locale, nous avons utilisé cette stratégie pour que les sommets qui ont amélioré la solution actuelle ne reviennent pas directement (Voir Section 5.2.2). Si BTS trouve une nouvelle

solution S mieux que S^*, BTS met à jour S^* par S. Si BTS ne trouve pas une solution mieux que S^* pendant L itérations consécutives, BTS s'arrête et elle retourne S^*.

Algorithme 8 L'algorithme BTS

Entres: Un graphe G avec des poids sur les arêtes, Un entier k (la taille de sous-graphe), Un entier L (profondeur de recherche), Un entier It (Conteur des itération).
Sorties: S^*

```
Début
S* ← S /*Enregistrer la meilleure solution trouvée jusqu'à présent*/
f* ← f(S*) /*f* enregistre la valeur de la fonction d'valuation de S**/
I ← 0 /*Initialisation du compteur d'itérations consécutives à 0*/
Tantque I < L Faire
  Si il existe une solution d'amélioration Alors
    S ← Intensification(S)
  Sinon
    S ← Diversification(S)
  Finsi
  Mis à jour de Tlist_u, Tlist_v et cnfg
  Si f(S) > f* Alors
    S* ← S
    I ← 0
    f* ← f(S*)
  Sinon
    I ← I + 1
  Finsi
  It ← It + 1
Fin tantque
Fin
Retourner(S*)
```

Intensification et Diversification

Dans l'étape d'intensification, BTS met à jour (remplace) la solution actuelle S par une nouvelle solution S' tel que $f(S') \geq f(S)$. Dans l'étape de diversification, BTS remplace la solution actuelle S par une nouvelle solution S' tel que $f(S') < f(S)$. Dans cette section, nous expliquons l'étape d'intensification et de diversification, pour cela, nous commençons par définir quelques paramètres nécessaires. Soit S la solution actuelle ($S \in \Omega$), nous avons attribué à chaque sommet $v \in V$ un poids relatif à S noté par $wd(v)$ et un degré relatif à S noté $d(v)$, $wd(v)$ et $d(v)$ sont définis comme suit :

$$wd(v) = \sum_{u \in S, uv \in E} w_{uv}$$

$$d(v) = |\{u \in S : uv \in E\}|$$

Comme nous avons dit, BTS utilise deux sous-ensembles critiques $A \subset S$ et $B \subset V \setminus S$ pour

définir le voisinage contraint. A et B sont définis comme suit :

Soit $Tlist\_u$ et $Tlist\_v$ les deux listes tabous utilisées par BTS et $cnfg$ le vecteur qui représente la stratégie de vérification de la configuration utilisée par BTS (Voir Section 5.2.2).

Soit $PMinInS = min\{wd(u) : u \in S, u \notin Tlist\_u, cnfg(u) = 1\}$ et

Soit $PMaxOutS = max\{wd(v) : v \in V \backslash S, v \notin Tlist\_v, cnfg(v) = 1\}$

$$A = \{u \in S : u \notin Tlist\_u, cnfg(u) = 1, wd(u) \leq PMinInS + Pmax\}$$

$$B = \{v \in V \backslash S : v \notin Tlist\_v, cnfg(v) = 1, wd(v) \geq PMaxOutS - Pmax\}$$

Où $Pmax = max\{w_{uv} : u \in S, v \in V \backslash S, v \notin Tlist\_v, cnfg(v) = 1, wd(v) = PMaxOutS\}$

Pour obtenir une solution voisine S' de la solution actuelle S, nous changeons un sommet de $u \in A$ avec un sommet de $v \in B$ i.e., $S' = S \bigoplus swap(u, v)$ ou $S' = S \backslash \{u\} \cup \{v\}$. Le voisinage contraint est composé de toutes les solutions obtenues par le changement d'un sommets de A avec un sommet B :

$$VC(S) = \{S' : S' = S \backslash \{u\} \cup \{v\}, u \in A, v \in B\}$$

On note que Wu & Hao (2013) [125] ont aussi utilisé le même principe de voisinage contraint dans leur algorithme hybride pour le problème de la diversité maximum, mais le voisinage utilisé dans notre algorithme est plus serré que le voisinage utilisé par leur algorithme, puisque $Pmax$ dans notre algorithme est plus petit que celui de leur algorithme.

Nous désignons un gain $\Delta_{u,v}$ pour chaque $swap(u, v)$, ce gain est défini par la variation de la fonction d'évaluation :

$$\Delta_{uv} = f(S') - f(S) = wd(v) - wd(u) - we_{uv}$$

avec $e_{uv} = w_{uv}$ si $uv \in E$ sinon $we_{uv} = 0$.

Comme nous avons dit, dans l'intensification BTS cherche à trouver une solution mieux que la solution actuelle S ou une solution qui ne détruit pas la solution S ($f(S') \geq f(S)$), il est facile de voir que nous ne pouvons pas faire une intensification si et seulement s'il existe un $swap(u, v)$ tel que $\Delta_{u,v} \geq 0$. L'étape d'intensification est faite comme suit :

1. S'il existe un ou plusieurs couples de sommets (u, v) tel que $\Delta_{uv} \geq 0$, BTS choisir le couple de sommets (u, v) qui a le maximum de Δ_{uv} et il applique le changement $swap(u, v)$, s'il y a plusieurs couples choisir un couple aléatoirement.
2. S'il n'existe pas un couple de sommets (u, v) tel que $\Delta_{uv} \geq 0$, BTS continuer sa recherche par une étape de diversification.

Si BTS n'a pas pu appliquer une intensification donc elle est tombée dans un optimum local, pour sortir de cet optimum, BTS applique une diversification. L'étape de diversification est faite comme suit :

1. Avec une grande probabilité ($P = 0.9$), BTS applique une petite diversification. Choisir un couple de sommets (u, v) ($u \in A$ *et* $v \in B$) qui a le grand Δ_{uv} et appliquer $swap(u, v)$, s'il y a plusieurs, choisir le couple qui a le minimum de Δd_{uv} où $\Delta d_{uv} = d(v) - d(u) - e_{uv}$ et $e_{uv} = 1$ si $uv \in E$ sinon $e_{uv} = 0$. S'il y a encore plusieurs choisir un couple aléatoirement.
2. Avec une petite probabilité $(1 - P)$, BTS applique une grande diversification. Choisir un sommet $u \in S$ aléatoirement et un sommet $v \in V \setminus S$ tel que $d(v) < \lfloor 0.4 \times k \rfloor$, s'il y a plusieurs choisir un aléatoirement, et appliquer le changement $swap(u, v)$

Après l'opération de changement ($swap(u, v)$), BTS met à jour les paramètres wd, d, $PMinInS$, $PMaxOutS$, A et B, pour préparer l'itération suivante.

La liste tabou et la stratégie de vérification de la configuration

BTS utilise deux listes tabous pour éviter le retour rapide des sommets changés dans l'étape de diversification et une stratégie de vérification de la configuration pour éviter le retour direct des sommets changés dans l'étape d'intensification (la stratégie de vérification de la configuration est aussi utilisée dans l'algorithme proposé dans le Chapitre 4). Après chaque étape de diversification, BTS ajoute le sommet u qui est sorti de la solution actuelle à une liste tabou appelé $Tlist_u$ pour qu'il ne peut pas revenir la solution actuelle qu'a après Tu itérations. BTS ajoute aussi le sommet v qui est entré à la solution actuelle à une liste tabou appelé $Tlist_v$ pour qu'il ne puisse pas sortir de la solution actuelle qu'a après Tv itérations (Tu et Tv sont appelés tabous tenures, voir Glover & Laguna (1997) [41]). Nous avons fixé $Tu = 15$ et $Tv = 9$ si $k \geq 30$, sinon $Tu = 8$ et $Tv = 5$.

Pour l'application de la stratégie de vérification de la configuration, nous utilisons un vecteur de 0 et 1 appelé $cnfg$ de taille $|V|$, chaque élément $cnfg(v)$ représente l'état du sommet v. $cnfg(v) = 1$ veut dire que le sommet v peut sortir de (revenir à) la solution actuelle, et si $cnfg(v) = 0$ le sommet v ne peut pas sortir de (revenir à) la solution actuelle. Nous maintenons le vecteur $cnfg$ comme suit :

1. Initialement, $cnfg(v) = 1$ pour tous sommet $v \in V$.

2. Après chaque étape d'intensification, l'état des sommets u et v change i.e., BTS change les valeurs de $cnfg(u)$ et $cnfg(v)$ de 1 à 0 ($cnfg(u) = 0$ et $cnfg(v) = 0$). BTS réinitialise aussi $cnfg(v')$ à 1 ($cnfg(v') = 1$) pour tout sommet $v' \in N(u) \cup N(v)$.

5.2.3 La stratégie de construction d'une nouvelle solution pour un nouveau restart

Pour encourager l'algorithme MTSEWS à visiter des nouvelles régions de recherche, nous avons utilisé le mécanisme proposé par Wu & Hao (2013) [124] (il est aussi utilisé dans la algorithme proposé dans le Chapitre 2). Wu & Hao ont attaché à chaque sommet v de V une fonction g_v appelée mémoire de fréquence à long terme (voir Wu & Hao (2013) [124]). g_v enregistre le nombre de mouvements du sommet v, la fonction g_v est maintenue comme suit :

1. Initialement, $g_v = 0$, $\forall v \in V$.
2. Chaque fois qu'un sommet v est ajouté ou supprimé de la solution actuelle S, g_v est incrémenté par un ($g_v = g_v + 1$).
3. Si pour tous sommet $v \in V$, $g_v > k$ alors réinitialiser $g_v = 0$ pour tout $v \in V$.

La nouvelle solution est construite comme suit :

1. Initialiser un ensemble S à vide.
2. Ajouter à S le sommet v qui a la plus petite fréquence g_v.
3. Ajouter à S un sommet $v \in V \setminus S$ tel que $\sum_{u \in S, uv \in E} w_{uv}$ est la plus grande somme, s'il y a plusieurs sommets choisir le sommet qui a la plus petite fréquence g_v, s'il y en a encore plusieurs choisir un aléatoirement.
4. Tant que $|S| < k$ répéter (3) et retourner S.

5.3 Résultats expérimentaux

Dans cette section, nous utilisons notre algorithme MTSEWS pour résoudre le problème du sous-graphe de poids maximum des arêtes dans quelques réseaux biologiques avec des poids sur les arêtes, nous comparons aussi les résultats de notre algorithme avec les résultats de l'algorithme de recherche tabou (TS_MEwS) proposé par Macambira [71] et les résultats l'algorithme de recherche à voisinage variable (VNS) proposé par Brimberg *et al.* [18]. Nous avons téléchargé les réseaux biologiques à partir du référentiel des données des réseaux [98], ces réseaux sont à l'origine avec poids sur les arêtes.

Nous avons programmé notre algorithme (MTSEWS) et les autres algorithmes (TS_MEwS et VNS) sous MATLAB, et nous avons compilé les trois algorithmes sur la même machine,

TABLE 5.1 – Résumé des résultats des algorithmes MTSEWS, TS_Mews et VNS

Les instances					MTSEWS		TS_Mews		VNS	
Nom	$\|V\|$	$\|E\|$	k	L	Moyenne	Meilleur	Moyenne	Meilleur	Moyenne	Meilleur
bio-CE-CX	15229	245952	86	100	**6986.2**	**7940.0**	2570.4	2570.4	1126.1	1351.9
bio-CE-GN	2220	53683	32	100	**1095.5**	**1095.5**	192.4753	192.4753	99.0834	122.3925
bio-CE-GT	924	3239	16	100	**204.6654**	**205.1335**	126.9924	128.5918	161.9576	183.5881
bio-CE-HT	2617	2985	16	100	**59.5432**	**65.0661**	10.0054	12.1446	5.7250	18.6123
bio-DM-CX	4040	76717	64	100	**5899.5**	**7681.9**	3256.0	3256.0	1478.2	1834.4
bio-DR-CX	3289	84940	98	1000	**14667.1**	**15802.0**	14139.0	14139.0	1810.7	2831.8
bio-HS-CX	4413	108818	36	1000	1955.0	**2587.7**	**1975.6**	1975.6	440.9218	549.8944
bio-HS-HT	2570	13691	76	1000	**7324.6**	**7324.6**	5835.6	5835.6	2829.8	5766.1
bio-HS-LC	4227	39484	116	1000	**14421.0**	**14421.0**	11370.0	11370.0	4902.0	5226.0
bio-SC-GT	1716	33987	78	1000	**4345.2**	**5329.5**	3704.1	3707.3	4787.6	5266.4
bio-SC-HT	2084	63027	124	1000	**20047.0**	**20047.0**	19501.0	19501.0	4941.7	5347.6
bio-SC-LC	2004	20452	58	1000	**2699.8**	**3280.1**	1802.2	1822.9	950.8185	1422.3

HP Workstation Z600 avec Intel Xeon X5550, 2.66GHz et 12 GB RAM. Nous avons exécuté MTSEWS avec les paramètres suivants $It_{mx} = 10^8$ et $L = 100$ pour quelques réseaux et $L = 1000$ pour les autres (voir Tableau 5.1), et TS_MEwS avec un nombre d'itérations maximum égal à 10^8. Nous avons fixé k à deux fois la borne inférieure de la taille de la clique maximum ω_{lb} ($k = 2 * \omega_{lb}$) sauf dans une instance (bio-CE-HT) $k = 4 * \omega_{lb}$ car ω_{lb} est très petit, nous avons obtenu les bornes inférieures de la taille de la clique maximum à partir de référentiel des données des réseaux [98]. Vu la nature stochastique des trois algorithmes, nous avons exécuté chaque algorithme 10 fois pour chaque réseau, et avons pris la valeur de la fonction objective de la meilleure solution (notée par Meilleur) et la moyenne des valeurs des fonctions objectives des 10 solutions (notée par Moyenne). Le temps de chaque exécution et de chaque algorithme est fixé à $1000s$. Les résultats sont résumés dans le Tableau 5.1.

Le Tableau 5.1 résume les résultats comparatifs de notre algorithme MTSEWS et les autres algorithmes (TS_Mews et VNS). Les colonnes 1, 2 et 3 présentent les informations des réseaux (le nom, le nombre des sommets $|V|$ et le nombre des arêtes $|E|$ respectivement). La colonne 4 montre la valeur de k. La colonne 5 montre la valeur de L utilisée dans l'exécution de MTSEWS. Les colonnes 6-11 présentent les valeurs des fonctions objectives des meilleurs solutions trouvées et les moyennes des solutions trouvées par chaque algorithme. Les résultats montrent que notre algorithme surperforme les autres algorithmes dans tous les réseaux sauf dans un seul cas (la moyenne du réseau bio-HS-CX). En outre, MTSEWS a obtenu des solutions strictement supérieures aux autres dans 11 sur 12 dans les moyennes et 12 sur 12 dans les meilleures solutions.

5.4 Conclusion

Un algorithme de recherche tabou multistart (MTSEWS) est proposé dans ce chapitre pour la résolution du problème du sous-graphe de poids maximum des arêtes. L'algorithme proposé utilise deux sous-ensembles critiques (un sous-ensemble de la solution courante S et un sous-ensemble de $V \setminus S$) pour construire un voisinage contraint. Notre algorithme utilise deux listes tabous, et nous avons ajouté aussi une stratégie de vérification de la configuration qui est très utilisée dans les algorithmes de recherche locale pour éviter les optimums locaux. Pour une recherche efficace dans l'espace de recherche, MTSEWS utilise aussi une stratégie multistart pour générer des nouvelles solutions initiales pour des nouveaux redémarrages (restart).

Nous avons testé notre algorithme sur 12 réseaux biologiques avec des poids originaux sur les arêtes et comparé nos résultats avec deux algorithmes de l'état de l'art. Les résultats computationnels ont montré que notre algorithme donne une très bonne performance sur ces réseaux. De plus, notre algorithme domine largement les autres algorithmes dans presque tous les réseaux.

Chapitre 6

Extraction de la plus grande Quasi-Clique à partir d'un réseau d'interactions protéine-protéine humaine

La clique a connu plusieurs applications dans les réseaux réels. Aussi, dans plusieurs applications dans les réseaux réels, la clique n'est pas utile en raison de sa définition forte (sous-graphe compilé). Un module dans un réseau d'interactions protéine-protéine (PPI) est un ensemble des protéines qui sont connectées pour réaliser une fonction biologique, une perte des interactions ne détruira pas la structure du module, les modules sont utilisés pour prédire les fonctions des protéines. Une quasi-clique peut bien représenter un module dans un réseau PPI. Dans ce chapitre, nous utilisons l'algorithme TSQC proposé dans le Chapitre 2 pour détecter la plus grande quasi-clique dans un réseau d'interactions protéine-protéine humaine. Les résultats ont montré que l'algorithme TSQC peut être utilisé pour la détection des modules fonctionnels avec la plus grande taille et que TSQC donne des résultats prometteurs.

Sommaire

6.1 Introduction

Les protéines sont des composantes essentielles des activités cellulaires et sont essentielles pour comprendre les fonctions moléculaires et des processus biologiques [25]. Les protéines agissent en interaction entre eux (protéine avec protéine) pour accomplir des fonctions biologiques. Les réseaux d'interactions protéine-protéine (PPI) ont été largement utilisés pour comprendre la biologie au niveau du système [59] [99]. Un module dans un réseau d'interactions protéine-protéine (PPI) est un ensemble des protéines qui sont connectées (l'une à l'autre) pour réaliser une fonction biologique [121], une perte ou un gain d'une interaction ne détruira pas la structure du module [130], les modules sont utilisés pour prédire les fonctions des protéines [106]. Plusieurs méthodes ont été développées pour détecter les modules fonctionnels dans les réseaux PPI et les modules qui se chevauchent, c'est un domaine de recherche très important en biologie computationnelle.

Dans ce chapitre, nous utilisons l'algorithme de recherche tabou (noté TSQC) proposé dans le Chapitre 2 pour détecter la plus grande quasi-clique dans un réseau d'interactions protéine-protéine humaine. Nous avons appliqué l'algorithme TSQC sur le réseau PPI de la base de données de référence sur les protéines humaines (HPRD, 2009) [57]. Les résultats ont montré que l'algorithme TSQC peut être bien utilisé pour la détection de le plus grand module fonctionnel dans les réseaux PPI. Le plus grand module fonctionnel (quasi-clique) trouvé est de taille 22. Nous avons trouvé deux modules fonctionnels de taille 22, les activités biologiques des protéines de chaque module sont presque les mêmes. Nous avons aussi utilisé l'algorithme TSQC pour chercher une quasi-clique de taille 10 et une quasi-clique de taille 15, les résultats ont montré que ces deux quasi-cliques représentent aussi des modules fonctionnels. Dans la littérature, deux travaux sont dédiés à la détection de la plus grande quasi-clique dans les réseaux d'interactions protéine-protéine humaine, ces travaux sont proposés par Bhattacharyya & Bandyopadhyay (2009) [11] et Sriwastava *et al.* (2017) [113]. Bhattacharyya & Bandyopadhyay ont détecté la plus grande quasi-clique dans le réseau PPI de la base de données de référence sur les protéines humaines (HPRD) , ils ont trouvé une quasi-clique de taille 15. Sriwastava *et al.* ont détecté la plus grande quasi-clique dans un réseau PPI noté dip20100614 [100]. Dans les deux travaux, les auteurs ont utilisé la deuxième définition de la quasi-clique maximum (voir Section 1.5.3), dans ce chapitre nous utilisons la première définition de la quasi-clique ce qui est différent aux autres travaux.

6.2 Extraction de la plus grande quasi-clique à partir le réseau PPI de HPRD

Un réseau PPI est noté par $R = (P, I)$ où P l'ensemble des protéines et I l'ensemble des interactions entre les protéines, il est facile de voir le lien direct entre $R = (P, I)$ et $G = (V, E)$, les sommets de G représentent les protéines et les arêtes représentent les interactions entre les protéines, donc nous pouvons appeler TSQC avec un réseau PPI comme un graphe. Rappelons que la méthode TSQC cherche à trouver une γ-quasi-clique de taille k et que le problème de la quasi-clique maximum peut être approché par trouver une série des k-γ-quasi-clique (k-γ-quasi-clique quasi-clique de taille k) avec des valeurs croissantes de k. Dans cette section, nous appliquons TSQC dans les réseaux PPI avec le même principe pour trouver la plus grande quasi-clique dans un réseau PPI i.e., nous cherchons une k-γ-quasi-clique et si TSQC trouve une k-γ-quasi-clique nous agrémentons k par 1 et refait la recherche jusqu'à ce que TSQC ne trouve pas une k-γ-quasi-clique. La dernière k-γ-quasi-clique est la plus grande quasi-clique.

Nous avons utilisé le réseau d'interaction protéine-protéine de l'Homo Sapiens de la collection HPRD (2009) [57]. Ce réseau est composé de 37060 interactions entre 19599 protéines humaines (19599 sommets et 37060 arêtes), sa densité égale à 0.000193 donc c'est un réseau sans échelle. Comme nous avons dit TSQC est implémenté sous MATLAB. Nous avons exécuté TSQC avec le réseau de HPRD sur une station HP workstation Z600 avec Windows 7 64 bits, 2,67 GHz CPU et 12G RAM. Nous avons fixé γ à 0.7 comme dans Bhattacharyya & Bandyopadhyay (2009) [11], L à 1000 et le nombre maximum des itérations (It_{mx}) à 10^8. Nous avons excusé TSQC pour déterminer la taille de la plus grande quasi-clique dans le réseau de HPRD, la plus grande quasi-clique est de taille 22. Après nous avons re-exécuté TSQC avec $k = 22$ plusieurs fois pour trouver les différentes quasi-cliques de taille 22. Nous avons obtenu deux quasi-cliques (de taille 22), les informations des protéines de la première quasi-clique sont résumées dans les tableaux 6.1 et 6.2 et celles de la deuxième quasi-clique sont résumées dans les tableaux 6.3 et 6.4 (les informations des protéines sont téléchargées à partir de la base de données de référence sur les protéines humaines (HPRD, 2009) [57]).

Les tableaux 6.1, 6.2, 6.3 et 6.4 donnent les noms et les Gènes HPRD ID pour chaque protéine (colonne 1 et 2). La colonne 3 présente les activités biologiques des protéines : classe moléculaire, fonction moléculaire et processus biologique. En analysant ces résultats nous observons que les processus biologiques de la plupart des protéines de chaque quasi-clique sont significativement les mêmes, ce qui signifie que ces deux quasi-clique sont des modules fonctionnels significatifs.

Nous avons aussi utilisé TSQC pour chercher une quasi-clique de taille 10 et une autre de taille 15. Les résultats de la quasi-clique de taille 10 (15 respectivement) sont résumés dans le tableau 6.5 (les tableaux 6.6 et 6.7). L'analyse de ces résultats a montré que les processus

biologiques de la plupart des protéines de chaque quasi-clique (celles de taille 10 et 15) sont significativement les mêmes, ce qui signifie que ces deux quasi-clique sont aussi des modules fonctionnels significatifs. Donc notre algorithme TSQC peut être utilisé pour la détection des modules fonctionnels qui ont des tailles fixes et avec différentes tailles.

6.3 Conclusion

Dans ce chapitre, nous avons utilisé la méthode TSQC proposée dans le Chapitre 2 pour l'extraction de la plus grande quasi-clique à partir le réseau d'interactions protéine-protéine humaine de l'Homo Sapiens de la collection HPRD (2009). Le plus grand module fonctionnel (quasi-clique) obtenu est de taille 22. Nous avons aussi utilisé TSQC pour chercher des autres modules fonctionnels de taille 22, un module fonctionnel de taille 10 et un autre de taille 15. Les résultats obtenus sont deux modules fonctionnels de taille 22, un de taille 10 et un de taille 15. L'analyse des activités biologiques des protéines de ces modules fonctionnels a montré que les protéines de chaque module fonctionnel ont presque les mêmes processus biologiques, ce qui est biologiquement significatif. Donc notre méthode TSQC peut être utilisée pour la détection des modules fonctionnels de différentes tailles fixes. Les résultats de TSQC dans ce domaine sont prometteurs.

TABLE 6.1 – Informations biologiques des protéines trouvées dans la première quasi-clique de taille 22

Nom de protéine	Gène HPRD ID	Classe moléculaire, Fonction moléculaire et Processus biologique
Grb2	00150	Molécule d'adaptateur
		Récepteur signalant une activité d'échafaudage complexe
		Transduction du signal ; Régulation du cycle cellulaire
CBL	01320	Protéine du système protéasomique de l'ubiquitine
		Activité protéase spécifique de l'ubiquitine
		Communication cellulaire ; Transduction du signal ; Métabolisme des protéines
Fyn	00655	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
STAT5A	03301	Facteur de transcription
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
STAT5B	05037	Facteur de transcription
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
Phospholipase C, gamma 1	01398	Enzyme : Phospholipase
		Activité de la phospholipase
		Métabolisme
Erythropoietin receptor	00587	Récepteur de surface cellulaire
		Activité des récepteurs
		Communication cellulaire ; Transduction du signal
Protein tyrosine phosphatase, non-receptor type 11	01470	Tyrosine phosphatase
		Activité de la protéine tyrosine phosphatase
		Communication cellulaire ; Transduction du signal
c-Src	01819	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Transduction du signal
PDGF receptor, beta	01423	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
Phosphatidylinositol 3 kinase regulatory subunit, alpha	01381	Molécule d'adaptateur
		Récepteur signalant une activité d'échafaudage complexe
		Communication cellulaire ; Transduction du signal

TABLE 6.2 – Suite des informations biologiques des protéines trouvées dans la première quasi-clique de taille 22

Nom de protéine	Gène HPRD ID	Classe moléculaire, Fonction moléculaire et Processus biologique
VAV1	01284	Facteur d'échange nucléotidique guanine
		Activité du facteur d'échange guanyl-nucléotidique
		Communication cellulaire ; Transduction du signal
Insulin receptor	00975	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
Janus kinase 2	00993	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
Protein tyrosine phosphatase, non-receptor type 6	01475	Tyrosine phosphatase
		Activité de la protéine tyrosine phosphatase
		Communication cellulaire ; Transduction du signal
SHC (Src homology 2 domain containing) transforming protein 1	02780	Molécule d'adaptateur
		Liaison aux protéines
		Cell communication ; Signal transduction
KIT	01287	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
EGF receptor	00579	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
CRK	01267	Molécule d'adaptateur
		Récepteur signalant une activité d'échafaudage complexe
		Communication cellulaire ; Transduction du signal
PTK2B protein tyrosine kinase 2 beta	03131	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Transduction du signal
SYK	02514	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
FAK	02859	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal

TABLE 6.3 – Les informations biologiques des protéines trouvées dans la deuxième quasi-clique de taille 22

Nom de protéine	Gène HPRD ID	Classe moléculaire, Fonction moléculaire et Processus biologique
Grb2	00150	Molécule d'adaptateur
		Récepteur signalant une activité d'échafaudage complexe
		Transduction du signal ; Régulation du cycle cellulaire
CBL	01320	Protéine du système protéasomique de l'ubiquitine
		Activité protéase spécifique de l'ubiquitine
		Communication cellulaire ; Transduction du signal ; Métabolisme des protéines
Fyn	00655	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
STAT5A	03301	Facteur de transcription
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
Phospholipase C, gamma 1	01398	Enzyme : Phospholipase
		Activité de la phospholipase
		Métabolisme
Protein tyrosine phosphatase, non-receptor type 11	01470	Tyrosine phosphatase
		Activité de la protéine tyrosine phosphatase
		Communication cellulaire ; Transduction du signal
c-Src	01819	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Transduction du signal
PDGF receptor, beta	01423	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
Phosphatidylinositol 3 kinase regulatory subunit, alpha	01381	Molécule d'adaptateur
		Récepteur signalant une activité d'échafaudage complexe
		Communication cellulaire ; Transduction du signal
VAV1	01284	Facteur d'échange nucléotidique guanine
		Activité du facteur d'échange guanyl-nucléotidique
		Communication cellulaire ; Transduction du signal
Insulin receptor	00975	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal

TABLE 6.4 – Suite des informations biologiques des protéines trouvées dans la deuxième quasi-clique de taille 22

Nom de protéine	Gène HPRD ID	Classe moléculaire, Fonction moléculaire et Processus biologique
Janus kinase 2	00993	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
Protein tyrosine phosphatase, non-receptor type 6	01475	Tyrosine phosphatase
		Activité de la protéine tyrosine phosphatase
		Communication cellulaire ; Transduction du signal
SHC (Src homology 2 domain containing) transforming protein 1	02780	Molécule d'adaptateur
		Liaison aux protéines
		Cell communication ; Signal transduction
KIT	01287	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
EGF receptor	00579	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
CRK	01267	Molécule d'adaptateur
		Récepteur signalant une activité d'échafaudage complexe
		Communication cellulaire ; Transduction du signal
PTK2B protein tyrosine kinase 2 beta	03131	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Transduction du signal
SYK	02514	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
FAK	02859	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
Lck	01080	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
ERBB2	01281	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal

TABLE 6.5 – Les informations biologiques des protéines trouvées dans la quasi-clique de taille 10

Nom de protéine	Gène HPRD ID	Classe moléculaire, Fonction moléculaire et Processus biologique
Protein tyrosine phosphatase, non-receptor type 11	01470	Tyrosine phosphatase
		Activité de la protéine tyrosine phosphatase
		Communication cellulaire ; Transduction du signal
c-Src	01819	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Transduction du signal
Phosphatidylinositol 3 kinase regulatory subunit, alpha	01381	Molécule d'adaptateur
		Récepteur signalant une activité d'échafaudage complexe
		Communication cellulaire ; Transduction du signal
Protein tyrosine phosphatase, non-receptor type 6	01475	Tyrosine phosphatase
		Activité de la protéine tyrosine phosphatase
		Communication cellulaire ; Transduction du signal
SHC (Src homology 2 domain containing) transforming protein 1	02780	Molécule d'adaptateur
		Liaison aux protéines
		Cell communication ; Signal transduction
EGF receptor	00579	Récepteur tyrosine kinase
		Protéine du récepteur transmembranaire activité tyrosine kinase
		Communication cellulaire ; Transduction du signal
FAK	02859	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
Lck	01080	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
Csk	00496	Tyrosine kinase
		Activité protéine-tyrosine kinase
		Communication cellulaire ; Transduction du signal
Estrogen receptor alpha	00589	Facteur de transcription
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, nucléosides, nucléotides et acides nucléiques

TABLE 6.6 – Les informations biologiques des protéines trouvées dans la quasi-clique de taille 15

Nom de protéine	Gène HPRD ID	Classe moléculaire, Fonction moléculaire et Processus biologique
Transcription factor Sp1	1796	Transcription factor
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
c Jun	1302	Transcription factor
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
Androgen receptor	2437	Récepteur nucléaire
		Activité des récepteurs nucléaires dépendant des ligands
		Communication cellulaire ; Transduction du signal
SMAD family member 3	4380	Protéine régulatrice de la transcription
		Activité du régulateur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
NFKB3	1241	Transcription factor
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
BRCA1	218	Protéine régulatrice de la transcription
		Activité du régulateur de transcription
		Régulation de l'expression génique, épigénétique
p53	1859	Transcription factor
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques ; Apoptose
Estrogen receptor alpha	589	Transcription factor
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
CDC2	302	Sérine/thréonine kinase
		Activité de la protéine sérine/thréonine kinase
		Communication cellulaire ; Transduction du signal

TABLE 6.7 – Suite des informations biologiques des protéines trouvées dans la quasi-clique de taille 15

Nom de protéine	Gène HPRD ID	Classe moléculaire, Fonction moléculaire et Processus biologique
E1A binding protein p300	4078	Protéine régulatrice de la transcription
		Activité du régulateur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
p73	3587	Transcription factor
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques ; Apoptose
c-Myc	1818	Transcription factor
		Activité du facteur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques ; Apoptose ; Régulation de la prolifération cellulaire
SMAD2	3221	Protéine de liaison à l'ADN
		Activité du régulateur de transcription
		Transduction du signal ; Régulation de l'expression génique, épigénétique
CREBBP	2534	Protéine régulatrice de la transcription
		Activité du régulateur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques
PCAF	6780	Protéine régulatrice de la transcription
		Activité du régulateur de transcription
		Régulation du métabolisme des nucléobases, des nucléosides, des nucléotides et des acides nucléiques

Conclusion générale

Conclusion

Dans cette thèse, nous nous sommes intéressés au développement des algorithmes pour le problème de la clique maximum, ses généralisations et relaxations à savoir le problème de la quasi-clique maximum, le problème de la clique de poids maximum des arêtes, le problème du sous-graphe de poids maximum des arêtes et ses applications aux réseaux biologiques. Tous ces problèmes sont des problèmes d'optimisation combinatoire de théorie des graphes qui ont une large application dans des différents domaines de la vie réelle. Nous sommes arrivés à proposer un algorithme pour chacun de ces problèmes et à faire deux applications aux réseaux biologiques.

Une définition et étude de l'état de l'art des méthodes de résolutions du problème de la clique maximum, ses généralisations et relaxations sont données dans le premier chapitre. Après, dans le deuxième chapitre, nous avons présenté un algorithme de recherche tabou (TSQC) pour le problème de la quasi-clique maximum, l'évaluation de cet algorithme a montré qu'il donne d'excellents résultats sur les graphes de différentes densités et sur les graphes sans échelle, et qu'il superforme les algorithmes existants de l'état de l'art. Ensuite, dans le troisième chapitre, nous avons présenté une heuristique hybride (HHa) et sa version parallèle une heuristique hybride parallèle (PHa) pour le problème de la clique maximum standard, ces heuristiques ont donné de très bons résultats sur les instances DIMACS. Puis dans le quantième chapitre, nous avons présenté une heuristique de recherche tabou multistart et multivoisinage pour le problème de la clique de poids maximum des arêtes, l'évaluation de cette heuristique a montré qu'elle superforme l'heuristique la plus récente sur les instances DIMACS, BHOSLIB et sur quelques réseaux biologiques. Ensuite dans le cinquième chapitre, un algorithme de recherche

tabou multistart (MTSEWS) est présenté pour résoudre le problème du sous-graphe de poids maximum des arêtes, MTSEWS est évalué sur quelques réseaux biologiques avec des poids originaux sur les arêtes et comparé avec deux algorithmes de l'état de l'art, les résultats ont montré que MTSEWS donne une très bonne performance sur les réseaux biologiques. Enfin dans le sixième chapitre, la méthode TSQC proposée dans le Chapitre 2 est utilisée pour l'extraction de la plus grande quasi-clique à partir le réseau d'interactions protéine-protéine humaine de l'Homo Sapiens de la collection HPRD (2009), nous avons aussi utilisé TSQC pour chercher des modules fonctionnels de différentes tailles, l'analyse des activités biologiques des protéines des modules fonctionnels trouvés a montré que les protéines de chaque module fonctionnel ont presque les mêmes processus biologiques, ce qui est biologiquement significatif, d'où les résultats de TSQC dans ce domaine sont prometteurs.

Perspectives

Dans les futurs travaux, nous aimerions :

- ◇ Utiliser notre algorithme TSQC pour l'identification de complexes protéiques et pour la détection de modules qui se chevauchent dans des réseaux d'interaction protéine-protéine ;
- ◇ Faire d'autres étude comparative entre nos algorithmes (HHa et PHa) et d'autre algorithme du problème de la clique maximum tel que l'algorithme TSQC [53] et HTS [112] et d'autres algorithme du problème de la clique de poids maximum des sommets ;
- ◇ Comparer notre algorithme MRNTS avec d'autres méthodes de résolution du problème de la clique de poids maximum des arêtes et utiliser MRNTS pour résoudre le problème de la clique de poids maximum des sommets ;
- ◇ Utiliser l'algorithme A^* pour le calcul de $\omega(G)$ dans quelques classes de graphes caractérisées par des propriétés de voisinage fermé (où un voisinage fermé d'un sommet v noté par $[N(v)]$ est l'ensemble $N(v) \cup \{v\}$ et $N(v)$ est le voisinage de v).

Bibliographie

[1] Abello, J., Resende, M. G. C., and Sudarsky, S. Massive Quasi-Clique Detection. In *LATIN 2002 : Theoretical Informatics* (Apr. 2002), Lecture Notes in Computer Science, Springer, Berlin, Heidelberg, pp. 598–612.

[2] Andrade, D. V., Resende, M. G., and Werneck, R. F. Fast local search for the maximum independent set problem. *Journal of Heuristics 18*, 4 (2012), 525–547.

[3] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. *Complexity and approximation : Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[4] Babel, L., and Tinhofer, G. A branch and bound algorithm for the maximum clique problem. *Zeitschrift für Operations Research 34*, 3 (1990), 207–217.

[5] Balas, E., and Yu, C. S. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing 15*, 4 (1986), 1054–1068.

[6] Balasundaram, B., Butenko, S., and Trukhanov, S. Novel Approaches for Analyzing Biological Networks. *Journal of Combinatorial Optimization 10*, 1 (Aug. 2005), 23–39.

[7] Batagelj, V., and Mrvar, A. *Pajek datasets*. 2006.

[8] Batsyn, M., Goldengorin, B., Maslov, E., and Pardalos, P. M. Improvements to MCS algorithm for the maximum clique problem. *Journal of Combinatorial Optimization 27*, 2 (2014), 397–416.

[9] Battiti, R., and Protasi, M. Reactive local search for the maximum clique problem 1. *Algorithmica 29*, 4 (2001), 610–637.

[10] Benlic, U., and Hao, J.-K. Breakout local search for maximum clique problems. *Computers & Operations Research 40*, 1 (2013), 192–206.

[11] Bhattacharyya, M., and Bandyopadhyay, S. Mining the largest quasi-clique in human protein interactome. In *Adaptive and Intelligent Systems, 2009. ICAIS'09. International Conference on* (2009), IEEE, pp. 194–199.

[12] Billionnet, A. Different Formulations for Solving the Heaviest K-Subgraph Problem. *INFOR : Information Systems and Operational Research 43*, 3 (Aug. 2005), 171–186.

[13] Birnbaum, B., and Goldman, K. J. An Improved Analysis for a Greedy Remote-Clique Algorithm Using Factor-Revealing LPs. *Algorithmica 55*, 1 (Sept. 2009), 42–59.

[14] Bomze, I. M., Budinich, M., Pardalos, P. M., and Pelillo, M. The maximum clique problem. In *Handbook of combinatorial optimization*. Springer, 1999, pp. 1–74.

[15] Bomze, I. R., Pelillo, M., and Stix, V. Approximating the maximum weight clique using replicator dynamics. *IEEE Transactions on Neural Networks 11*, 6 (Nov. 2000), 1228–1241.

[16] Brendel, W., Amer, M., and Todorovic, S. Multiobject tracking as maximum weight independent set. In *CVPR 2011* (2011), IEEE, pp. 1273–1280.

[17] Brendel, W., and Todorovic, S. Segmentation as Maximum-Weight Independent Set. In *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 307–315.

[18] Brimberg, J., Mladenović, N., Urošević, D., and Ngai, E. Variable neighborhood search for the heaviest k-subgraph. *Computers & Operations Research 36*, 11 (2009), 2885–2891.

[19] Brélaz, D. New methods to color the vertices of a graph. *Communications of the ACM 22*, 4 (1979), 251–256.

[20] Brunato, M., Hoos, H. H., and Battiti, R. On effectively finding maximal quasi-cliques in graphs. In *International conference on learning and intelligent optimization* (2007), Springer, pp. 41–55.

[21] Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. Hyper-heuristics : An emerging direction in modern search technology. In *Handbook of metaheuristics*. Springer, 2003, pp. 457–474.

[22] Busygin, S. A new trust region technique for the maximum weight clique problem. *Discrete Applied Mathematics 154*, 15 (2006), 2080–2096.

[23] Butenko, S. *Maximum independent set and related problems, with applications.* PhD thesis, University of Florida, 2003.

[24] Cai, S., and Lin, J. Fast Solving Maximum Weight Clique Problem in Massive Graphs. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), IJCAI'16, AAAI Press, pp. 568–574. event-place : New York, New York, USA.

[25] Cao, B., Luo, J., Liang, C., Wang, S., and Song, D. Moepga : A novel method to detect protein complexes in yeast proteinâ"protein interaction networks based on multiobjective evolutionary programming genetic algorithm. *Computational biology and chemistry 58* (2015), 173–181.

[26] Carraghan, R., and Pardalos, P. M. An exact algorithm for the maximum clique problem. *Operations Research Letters 9*, 6 (1990), 375–382.

[27] Davis, T. A., and Hu, Y. The University of Florida Sparse Matrix Collection. *ACM Trans. Math. Softw. 38*, 1 (Dec. 2011), 1 :1–1 :25.

[28] Desler, J. F., and Hakimi, S. L. On finding a maximum internally stable set of a graph. In *Proc. of Fourth Annual Princeton Conference on Information Sciences and Systems* (1970), vol. 4, pp. 459–462.

[29] Dijkhuizen, G., and Faigle, U. A cutting-plane approach to the edge-weighted maximal clique problem. *European Journal of Operational Research 69*, 1 (Aug. 1993), 121–130.

[30] Djeddi, Y., Ait Haddadene, H., and Belacel, N. An extension of adaptive multi-start tabu search for the maximum quasi-clique problem. *Computers & Industrial Engineering 132* (June 2019), 280–292.

[31] Fahle, T. Simple and fast : Improving a branch-and-bound algorithm for maximum clique. In *Algorithms—ESA 2002*. Springer, 2002, pp. 485–498.

[32] Fan, Y., Li, N., Li, C., Ma, Z., Latecki, L. J., and Su, K. Advancing Tabu and Restart in Local Search for Maximum Weight Cliques. *arXiv :1804.08187 [cs]* (Apr. 2018). arXiv : 1804.08187.

[33] Fang, Z., Li, C.-M., and Xu, K. An exact algorithm based on maxsat reasoning for the maximum weight clique problem. *Journal of Artificial Intelligence Research 55* (2016), 799–833.

[34] Fleurent, C., and Ferland, J. A. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research 63*, 3 (1996), 437–461.

[35] Friden, C., Hertz, A., and de Werra, D. Stabulus : A technique for finding stable sets in large graphs with tabu search. *Computing 42*, 1 (1989), 35–44.

[36] Galinier, P., and Hao, J.-K. Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization 3*, 4 (1999), 379–397.

[37] Gallego, M., Duarte, A., Laguna, M., and Martí, R. Hybrid heuristics for the maximum diversity problem. *Computational Optimization and Applications 44*, 3 (2009), 411.

[38] Gary, M. R., and Johnson, D. S. *Computers and Intractability : A Guide to the Theory of NP-completeness*. WH Freeman and Company, New York, 1979.

[39] Gendreau, M., Soriano, P., and Salvail, L. Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research 41*, 4 (1993), 385–403.

[40] Glover, F., Kuo, C.-C., and Dhir, K. S. Heuristic algorithms for the maximum diversity problem. *Journal of Information and Optimization Sciences 19*, 1 (Jan. 1998), 109–132.

[41] Glover, F., and Laguna, M. *Tabu search (Vol. 22)*. Boston : Kluwer academic publishers, 1997.

[42] Gouveia, L., and Martins, P. Solving the maximum edge-weight clique problem in sparse graphs with compact formulations. *EURO Journal on Computational Optimization 3*, 1 (Feb. 2015), 1–30.

[43] Grosso, A., Locatelli, M., and Della Croce, F. Combining swaps and node weights in an adaptive greedy approach for the maximum clique problem. *Journal of Heuristics 10*, 2 (2004), 135–152.

[44] Hansen, P., Mladenović, N., and Urošević, D. Variable neighborhood search for the maximum clique. *Discrete Applied Mathematics 145*, 1 (2004), 117–125.

[45] Harary, F., and Ross, I. C. A procedure for clique detection using the group matrix. *Sociometry 20*, 3 (1957), 205–215.

[46] Hosseinian, S., Fontes, D. B., and Butenko, S. A quadratic approach to the maximum edge weight clique problem. *XIII Global Optimization Workshop GOW* (2016), 125–128.

[47] Hosseinian, S., Fontes, D. B., and Butenko, S. A nonconvex quadratic optimization approach to the maximum edge weight clique problem. *Journal of Global Optimization 72*, 2 (Oct. 2018), 219–240.

[48] Hosseinian, S., Fontes, D. B. M. M., Butenko, S., Nardelli, M. B., Fornari, M., and Curtarolo, S. The Maximum Edge Weight Clique Problem : Formulations and Solution Approaches. In *Optimization Methods and Applications : In Honor of Ivan*

V. Sergienko's 80th Birthday, S. Butenko, P. M. Pardalos, and V. Shylo, Eds., Springer Optimization and Its Applications. Springer International Publishing, Cham, 2017, pp. 217–237.

[49] Houck, D. J. On the Vertex Packing Problem. *Ph.D. dissertation, The Johns Hopkins University, Baltimore.* (1974).

[50] Houck, D. J., and Vemuganti, R. R. An algorithm for the vertex packing problem. *Operations Research 25*, 5 (1977), 773–787.

[51] Jagota, A., and Sanchis, L. A. Adaptive, restart, randomized greedy heuristics for maximum clique. *Journal of Heuristics 7*, 6 (2001), 565–585.

[52] Jiang, H., Li, C.-M., and Manya, F. An exact algorithm for the maximum weight clique problem in large graphs. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[53] Jin, Y., and Hao, J.-K. General swap-based multiple neighborhood tabu search for the maximum independent set problem. *Engineering Applications of Artificial Intelligence 37* (2015), 20–33.

[54] Johnson, D. S., and Trick, M. A. Second DIMACS implementation challenge : cliques, coloring and satisfiability, volume 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science. *American Mathematical Society* (1996).

[55] Karp, R. M. Reducibility among combinatorial problems. In *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[56] Katayama, K., Hamamoto, A., and Narihisa, H. An effective local search for the maximum clique problem. *Information Processing Letters 95*, 5 (2005), 503–511.

[57] Keshava Prasad, T. S., Goel, R., Kandasamy, K., Keerthikumar, S., Kumar, S., Mathivanan, S., Telikicherla, D., Raju, R., Shafreen, B., and Venugopal, A. Human protein reference database−2009 update. *Nucleic acids research 37*, suppl_1 (2008), D767–D772.

[58] Kilkusts, P. Another algorithm determining the independence number of a graph. *Elektronische Informationsverarbeitung und Kybernetik 22*, 4 (1986), 157–166.

[59] Kitano, H. Systems biology : a brief overview. *Science 295*, 5560 (2002), 1662–1664.

[60] Konc, J., and Janezic, D. An improved branch and bound algorithm for the maximum clique problem. *proteins 4* (2007), 5.

[61] Kumlander, D. A new exact algorithm for the maximum-weight clique problem based on a heuristic vertex-coloring and a backtrack search. In *Proc. 5th Int'l Conf. on Modelling, Computation and Optimization in Information Systems and Management Sciences* (2004), Citeseer, pp. 202–208.

[62] Lamm, S., Schulz, C., Strash, D., Williger, R., and Zhang, H. Exactly Solving the Maximum Weight Independent Set Problem on Large Real-World Graphs. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, Proceedings. Society for Industrial and Applied Mathematics, Jan. 2019, pp. 144–158.

[63] Letsios, M., Balalau, O. D., Danisch, M., Orsini, E., and Sozio, M. Finding Heaviest k-Subgraphs and Events in Social Media. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)* (Dec. 2016), pp. 113–120.

[64] Li, C.-M., Fang, Z., and Xu, K. Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem. In *2013 IEEE 25th International Conference on Tools with Artificial Intelligence* (2013), IEEE, pp. 939–946.

[65] Li, C. M., and Quan, Z. An Efficient Branch-and-Bound Algorithm Based on MaxSAT for the Maximum Clique Problem. In *AAAI* (2010), vol. 10, pp. 128–133.

[66] Li, N., and Latecki, L. J. Clustering aggregation as maximum-weight independent set. In *Advances in neural information processing systems* (2012), pp. 782–790.

[67] Li, R., Wu, X., Liu, H., Wu, J., and Yin, M. An Efficient Local Search for the Maximum Edge Weighted Clique Problem. *IEEE Access 6* (2018), 10743–10753.

[68] Li, W., Liu, Y., Huang, H.-C., Peng, Y., Lin, Y., Ng, W.-K., and Ong, K.-L. Dynamical systems for discovering protein complexes and functional modules from biological networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 4*, 2 (2007), 233–250.

[69] Lourenço, H. R., Martin, O. C., and Stützle, T. Iterated local search. In *Handbook of metaheuristics*. Springer, 2003, pp. 320–353.

[70] Luce, R. D. Connectivity and generalized cliques in sociometric group structure. *Psychometrika 15*, 2 (June 1950), 169–190.

[71] Macambira, E. M. An Application of Tabu Search Heuristic for the Maximum Edge-Weighted Subgraph Problem. *Annals of Operations Research 117*, 1 (Nov. 2002), 175–190.

[72] Macambira, E. M., and de Meneses, C. N. A GRASP for the maximum edge-weighted subgraph problem. *IX Confresso Latino-Iberoamericano de Investigacion Operativa* (1998).

[73] Mannino, C., and Stefanutti, E. An augmentation algorithm for the maximum weighted stable set problem. *Computational Optimization and Applications 14*, 3 (1999), 367–381.

[74] Marchiori, E. Genetic, iterated and multistart local search for the maximum clique problem. In *Applications of Evolutionary Computing*. Springer, 2002, pp. 112–121.

[75] Maslov, E., Batsyn, M., and Pardalos, P. M. Speeding up branch and bound algorithms for solving the maximum clique problem. *Journal of Global Optimization 59*, 1 (2014), 1–21.

[76] Massaro, A., Pelillo, M., and Bomze, I. A Complementary Pivoting Approach to the Maximum Weight Clique Problem. *SIAM Journal on Optimization 12*, 4 (Jan. 2002), 928–948.

[77] Matsuda, H., Ishihara, T., and Hashimoto, A. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science 210*, 2 (1999), 305–325.

[78] Mokken, R. J. Cliques, clubs and clans. *Quality and Quantity 13*, 2 (Apr. 1979), 161–173.

[79] Nogueira, B., and Pinheiro, R. G. S. A CPU-GPU local search heuristic for the maximum weight clique problem on massive graphs. *Computers & Operations Research 90* (Feb. 2018), 232–248.

[80] Nogueira, B., Pinheiro, R. G. S., and Subramanian, A. A hybrid iterated local search heuristic for the maximum weight independent set problem. *Optimization Letters 12*, 3 (May 2018), 567–583.

[81] Oliveira, A. B., Plastino, A., and Ribeiro, C. C. Construction heuristics for the maximum cardinality quasi-clique problem. In *The 10th Metaheuristics International Conference* (Singapore, 2013), p. 84.

[82] Östergård, P. R. A fast algorithm for the maximum clique problem. *Discrete Applied Mathematics 120*, 1 (2002), 197–207.

[83] Östergård, P. R. J. A New Algorithm for the Maximum-Weight Clique Problem. *Electronic Notes in Discrete Mathematics 3* (May 1999), 153–156.

[84] Pajouh, F. M., Miao, Z., and Balasundaram, B. A branch-and-bound approach for maximum quasi-cliques. *Annals of Operations Research 216*, 1 (May 2014), 145–161.

[85] Palubeckis, G. Iterated tabu search for the maximum diversity problem. *Applied Mathematics and Computation 189*, 1 (2007), 371–383.

[86] Pardalos, P. M., and Xue, J. The maximum clique problem. *Journal of global Optimization 4*, 3 (1994), 301–328.

[87] Park, K., Lee, K., and Park, S. An extended formulation approach to the edge-weighted maximal clique problem. *European Journal of Operational Research 95*, 3 (Dec. 1996), 671–682.

[88] Pastukhov, G., Veremyev, A., Boginski, V., and Prokopyev, O. A. On maximum degree-based-quasi-clique problem : Complexity and exact approaches. *Networks 71*, 2 (2018), 136–152.

[89] Pattillo, J., Veremyev, A., Butenko, S., and Boginski, V. On the maximum quasi-clique problem. *Discrete Applied Mathematics 161*, 1-2 (2013), 244–257.

[90] Pinto, B. Q., Plastino, A., Ribeiro, C. C., and Rosseti, I. A biased random-key genetic algorithm to the maximum cardinality quasi-clique problem. In *Abstracts of the 11th Metaheuristics International Conference* (Agadir, 2015).

[91] Pinto, B. Q., Ribeiro, C. C., Rosseti, I., and Plastino, A. A biased random-key genetic algorithm for the maximum quasi-clique problem. *European Journal of Operational Research 271*, 3 (Dec. 2018), 849–865.

[92] Pullan, W. Phased local search for the maximum clique problem. *Journal of Combinatorial Optimization 12*, 3 (2006), 303–323.

[93] Pullan, W. Approximating the maximum vertex/edge weighted clique using local search. *Journal of Heuristics 14*, 2 (Apr. 2008), 117–134.

[94] Pullan, W., and Hoos, H. H. Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research* (2006), 159–185.

[95] Pullan, W., Mascia, F., and Brunato, M. Cooperating local search for the maximum clique problem. *Journal of Heuristics 17*, 2 (2011), 181–199.

[96] Regin, J. C. Solving the maximum clique problem with constraint programming. In *Proceedings of CPAIOR* (2003), vol. 3, Citeseer, pp. 634–648.

[97] Robson, J. M. Algorithms for maximum independent sets. *Journal of Algorithms 7*, 3 (Sept. 1986), 425–440.

[98] Rossi, R., and Ahmed, N. The network data repository with interactive graph analytics and visualization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence* (2015).

[99] Rual, J.-F., Venkatesan, K., Hao, T., Hirozane-Kishikawa, T., Dricot, A., Li, N., Berriz, G. F., Gibbons, F. D., Dreze, M., Ayivi-Guedehoussou, N., and others. Towards a proteome-scale map of the human protein-protein interaction network. *Nature 437*, 7062 (2005), 1173–1178.

[100] Salwinski, L., Miller, C. S., Smith, A. J., Pettit, F. K., Bowie, J. U., and Eisenberg, D. The database of interacting proteins : 2004 update. *Nucleic acids research 32*, suppl_1 (2004), D449–D451.

[101] San Segundo, P., Artieda, J., Batsyn, M., and Pardalos, P. M. An enhanced bitstring encoding for exact maximum clique search in sparse graphs. *Optimization Methods and Software 32*, 2 (Mar. 2017), 312–335.

[102] San Segundo, P., Lopez, A., and Pardalos, P. M. A new exact maximum clique algorithm for large and massive sparse graphs. *Computers & Operations Research 66* (Feb. 2016), 81–94.

[103] San Segundo, P., Matia, F., Rodriguez-Losada, D., and Hernando, M. An improved bit parallel exact maximum clique algorithm. *Optimization Letters 7*, 3 (2013), 467–479.

[104] San Segundo, P., Rodríguez-Losada, D., and Jiménez, A. An exact bit-parallel algorithm for the maximum clique problem. *Computers & Operations Research 38*, 2 (2011), 571–581.

[105] Seidman, S. B., and Foster, B. L. A graph-theoretic generalization of the clique concept. *Journal of Mathematical sociology 6*, 1 (1978), 139–154.

[106] Sharan, R., Ulitsky, I., and Shamir, R. Network-based prediction of protein function. *Molecular systems biology 3*, 1 (2007), 88.

[107] Shimizu, S., Yamaguchi, K., and Masuda, S. A Maximum Edge-Weight Clique Extraction Algorithm Based on Branch-and-Bound. *arXiv :1810.10258 [cs]* (Oct. 2018). arXiv : 1810.10258.

[108] Shimizu, S., Yamaguchi, K., and Masuda, S. A Branch-and-Bound Based Exact Algorithm for the Maximum Edge-Weight Clique Problem. In *Computational Science/Intelligence & Applied Informatics*, R. Lee, Ed., Studies in Computational Intelligence. Springer International Publishing, Cham, 2019, pp. 27–47.

[109] Shimizu, S., Yamaguchi, K., Saitoh, T., and Masuda, S. Some Improvements on Kumlander-s Maximum Weight Clique Extraction Algorithm. 5.

[110] Singh, A., and Gupta, A. K. A hybrid evolutionary approach to maximum weight clique problem. *International Journal of Computational Intelligence Research 2*, 4 (2006), 349–355.

[111] Singh, H., Kumar, M., and Aggarwal, P. Approximation of Heaviest k-Subgraph Problem by Size Reduction of Input Graph. In *Proceedings of 2nd International Conference on Communication, Computing and Networking* (2019), C. R. Krishna, M. Dutta, and R. Kumar, Eds., Lecture Notes in Networks and Systems, Springer Singapore, pp. 599–605.

[112] Smith, D. H., Perkins, S., and Montemanni, R. Solving the maximum clique problem with a hybrid algorithm. *International Journal of Metaheuristics 7*, 2 (Jan. 2019), 152–175.

[113] Sriwastava, B. K., Basu, S., and Maulik, U. A Quasi-Clique Mining Algorithm for Analysis of the Human Protein-Protein Interaction Network. In *Pattern Recognition and*

Machine Intelligence, Lecture Notes in Computer Science. Springer, Cham, Dec. 2017, pp. 411–417.

[114] Tarjan, R. E. Finding a maximum clique. *Technical Report , Computer Sci. Dept" Cornell University, Ithaca, NY* (1972), 72–123.

[115] Tarjan, R. E., and Trojanowski, A. E. Finding a maximum independent set. *SIAM Journal on Computing 6*, 3 (1977), 537–546.

[116] Tomita, E., and Kameda, T. An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *Journal of Global Optimization 37*, 1 (2007), 95–111.

[117] Tomita, E., and Seki, T. An efficient branch-and-bound algorithm for finding a maximum clique. In *Discrete mathematics and theoretical computer science*. Springer, 2003, pp. 278–289.

[118] Tomita, E., Sutani, Y., Higashi, T., Takahashi, S., and Wakatsuki, M. A simple and faster branch-and-bound algorithm for finding a maximum clique. In *WALCOM : Algorithms and computation*. Springer, 2010, pp. 191–203.

[119] Tomita, E., Yoshida, K., Hatta, T., Nagao, A., Ito, H., and Wakatsuki, M. A Much Faster Branch-and-Bound Algorithm for Finding a Maximum Clique. In *Frontiers in Algorithmics* (2016), D. Zhu and S. Bereg, Eds., Lecture Notes in Computer Science, Springer International Publishing, pp. 215–226.

[120] Veremyev, A., Prokopyev, O. A., Butenko, S., and Pasiliao, E. L. Exact MIP-based approaches for finding maximum quasi-cliques and dense subgraphs. *Computational Optimization and Applications 64*, 1 (May 2016), 177–214.

[121] Wagner, G. P., Pavlicev, M., and Cheverud, J. M. The road to modularity. *Nature Reviews Genetics 8*, 12 (2007), 921–931.

[122] Wang, Y., Hao, J.-K., Glover, F., and Lü, Z. A tabu search based memetic algorithm for the maximum diversity problem. *Engineering Applications of Artificial Intelligence 27* (2014), 103–114.

[123] Warren, J. S., and Hicks, I. V. Combinatorial branch-and-bound for the maximum weight independent set problem. *Relatório técnico, Texas A&M University, Citeseer* (2006).

[124] Wu, Q., and Hao, J.-K. An adaptive multistart tabu search approach to solve the maximum clique problem. *Journal of Combinatorial Optimization 26*, 1 (2013), 86–108.

[125] Wu, Q., and Hao, J.-K. A hybrid metaheuristic method for the maximum diversity problem. *European Journal of Operational Research 231*, 2 (2013), 452–464.

[126] Wu, Q., and Hao, J.-K. A review on algorithms for maximum clique problems. *European Journal of Operational Research 242*, 3 (2015), 693–709.

[127] Wu, Q., Hao, J.-K., and Glover, F. Multi-neighborhood tabu search for the maximum weight clique problem. *Annals of Operations Research 196*, 1 (2012), 611–634.

[128] Xu, K., Boussemart, F., Hemery, F., and Lecoutre, C. A simple model to generate hard satisfiable instances. *arXiv preprint cs/0509032* (2005).

[129] Yamaguchi, K., and Masuda, S. A new exact algorithm for the maximum weight clique problem. In *ITC-CSCC : International Technical Conference on Circuits Systems, Computers and Communications* (2008), pp. 317–320.

[130] Yu, Q., Li, G.-H., and Huang, J.-F. MOfinder : a novel algorithm for detecting overlapping modules from protein-protein interaction network. *BioMed Research International 2012* (2012).

[131] Zhang, Q., Sun, J., and Tsang, E. An evolutionary algorithm with guided mutation for the maximum clique problem. *Evolutionary Computation, IEEE Transactions on 9*, 2 (2005), 192–200.

[132] Zhang, Z., Song, J., Tang, J., Xu, X., and Guo, F. Detecting complexes from edge-weighted PPI networks via genes expression analysis. *BMC systems biology 12*, 4 (2018), 40.

[133] Zhou, Y., Hao, J.-K., and Goëffon, A. PUSH : A generalized operator for the Maximum Vertex Weight Clique Problem. *European Journal of Operational Research 257*, 1 (Feb. 2017), 41–54.

Résumé : Le problème de la clique maximum est un problème d'optimisation combinatoire NP-complet, considéré comme un sujet de recherche très actuel en théorie des graphes. Avec une large application dans de nombreux domaines tel que les réseaux biologiques, génie biomédical, bio-informatique et chimio-informatique, la théorie de la classification, les réseaux sociaux et l'économie. Dans cette thèse, nous nous sommes intéressés au développement des algorithmes pour le problème de la clique maximum, ses généralisations et relaxations, et ses applications aux réseaux biologiques. Pour atteindre ces objectifs, nous avons proposé : Une extension de la recherche tabou adaptative multistart pour le problème de la quasi-clique maximum, une heuristique hybride et sa version parallèle pour le problème de la clique maximum, un algorithme de recherche tabou multistart et multivoisinage pour le problème de la clique de poids maximum des arêtes et un algorithme recherche tabou pour le problème du sous-graphe de poids maximum des arêtes. En outre, nous avons fait deux applications aux réseaux biologiques : Résolution du problème du sous-graphe de poids maximum des arêtes dans des réseaux biologiques et extraction de la plus grande quasi-clique à partir d'un réseau d'interactions protéine-protéine humaine. Tous les algorithmes proposés sont testés sur des instances de références et comparés avec les algorithmes de l'état de l'art, les résultats sont compétitifs.

Mots clés : La clique maximum ; La clique de poids maximum des arêtes (sommets) ; La quasi-clique maximum ; Le sous-graphe de poids maximum des arêtes ; Les réseaux biologiques ; Les réseaux d'interactions protéine-protéine humaine.

Abstract : The maximum clique problem is a NP-complete combinatorial optimization problem, considered as a very current research topic in graph theory. With a wide application in many fields including biological networks, biomedical engineering, bioinformatics and chemoinformatics, classification theory, social networks and economics. In this thesis, we were interested in the development of algorithms for the maximum clique problem, its generalizations and relaxations, and its applications to biological networks. To achieve these objectives, we proposed : An extension of adaptive multi-start tabu search for the maximum quasi-clique problem, hybrid heuristic and parallel hybrid approaches for the maximum clique problem, multistart and multi-neighborhood tabu search for the problem of maximum edge weighted clique problem and A tabu search algorithm for the maximum edge weighted subgraph problem. In addition, we have made two applications to biological networks : Solving the maximum edge weighted subgraph problem in biological networks and mining the largest quasi-clique in human protein-protein interaction networks. All proposed algorithms are tested on benchmark instances and compared with state-of-the-art algorithms, the results are competitive.

Key Words : Maximum clique ; Maximum edge (vertex) weighted clique ; Maximum quasi-clique ; Maximum edge weighted subgraph ; Biological networks ; Human protein-protein interaction networks.