N°d'ordre: ...../2021

République Algerienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la Recherche scientifique
Université M'hamed Bougara Boumerdes

Faculté des Sciences

Thèse de Doctorat

Présentée par :

Mr Ali CHAOUCHE

Filière : Informatique
Option : Informatique

Approches évolutionnaires pour le problème de partitionnement de graphes

---

Devant le jury :

M.	MAOUCHE Amine Riad	Professeur UMBB	Président du jury
M.	AIT ZAI Abdelhakim	Professeur USTHB	Examinateur
M.	BOUKRA Abdelmadjid	Professeur USTHB	Examinateur
M.	ATIF Karim	MCA USTHB	Examinateur
M.	GACEB Djamel	MCA UMBB	Examinateur
M.	BOULIF Menouar	Professeur UMBB	Directeur de thèse
M.	HADDAD Mohamed	MC Université Lyon 1 (France)	Invité

Année Universitaire : 2020 - 2021

A mes chers parents
A mes enfants Mohamed Amine, et Nada Assile
A mes frère et soeurs et leurs enfants
A mon beau père et belle mère
A mes beaux frère et belles soeurs
A toute personne qui m'est chere
Je dédie ce manuscrit
Ali.

Remerciements

Au nom d'Allah, le Tout Miséricordieux, le Très Miséricordieux, et prières et paix sur le plus noble des prophètes et sur sa famille et ses compagnons. Après un travail d'une longue halène, grâce à Allah le tout puissant, on est arrivé à terme de ce travail de recherche qui s'inscrit dans le cadre de ma thèse de doctorat louange à Allah.

Je voudrais tout d'abord exprimer mes plus profonds remerciements à mon Directeur de thèse M.Menouar BOULIF qui m'a encadré, orienté, formé, ... , en plus de sa patience et de son caractère qui étaient parfois mon seul soutient face aux divers obstacles qu'un doctorant puisse rencontrer durant son travail. Je trouvais en lui le frère qui n'a jamais cessé de m'encourager et de me pousser vers l'avant pour arriver à terme de ma thèse. Merci.

Ensuite, Je tiens à adresser mes sincères remerciements à M. MAOUCHE Riad Amine d'avoir accepté de présider le jury.

Je tiens aussi à remercier vivement les membres du jury Monsieur Abdelhakim AIT ZAI, Monsieur BOUKRA Abdelmadjid et Monsieur ATIF Karim pour avoir pris le temps de lire et d'évaluer mon travail.

Monsieur GACEB Djamel et HADDAD Mohamed, je dis merci pour votre disponibilité et votre présence fraternelle et permanente et pour tout le soutien que vous m'avez apporté.

Mes remerciements vont aussi à notre Directeur du laboratoire LIMOSE Monsieur MEZGHICHE Mohamed, au chef de département Monsieur CHAABANI Mohamed qui m'avait facilité toutes les tâches administratives et à tous les enseignants du département d'informatique et particulièrement Monsieur BERICHI Ali.

J'adresse aussi mes vifes remerciements à ma femme qui était pour moi un véritable soutien moral avec sa patience, ses encouragements et surtout sa disponibilité... **Merci**.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Les graphes sont souvent utilisés pour la modélisation des problèmes de la vie réelle. Cet outil de modélisation permet une réduction de complexité qui rend les graphes plus appropriés pour la détermination de la solution indépendamment de l'application finale qu'on cherche à résoudre en bénéficiant d'un socle théorique solide et diversifié. Le problème de partitionnement de graphes (PPG) est l'un des problèmes de la théorie des graphes, ou plus généralement, d'optimisation combinatoire dont le spectre d'applicabilité s'étend sur plusieurs domaines. De manière simpliste, ce problème consiste à décomposer l'ensemble des sommets d'un graphe pondéré en sous-ensembles ou clusters disjoints de façon à minimiser le coût des arêtes inter-clusters, en respectant un certain nombre de contraintes telles que la taille des clusters.

Le PPG est classé parmi les problèmes NP-difficiles, et par conséquent, l'utilisation de méthodes exactes pour le résoudre ne peut être envisagée pour les instances de grandes tailles. De ce fait, l'utilisation de méthodes approchées telles que les algorithmes génétiques (AG) semble être la piste la plus prometteuse.

L'AG est une métaheuristique inspirée du phénomène d'évolution et de survie des êtres vivants. Sa forme simple consiste à utiliser un codage pour représenter par des chromosomes un ensemble de solutions choisies aléatoirement pour former la population initiale. L'aptitude de survie de chaque chromosome est mesurée par une fonction dite fitness. La population génétique subit ensuite de manière itérative un ensemble d'opérateurs génétiques qui assurent son évolution. Ces opérateurs favorisent par sélection la survie des chromosomes de bonne fitness qui vont échanger une partie de leurs codes génétiques par croisement et/ou voir leurs codes génétiques altéré par mutation. L'évolution est arrêtée lorsqu'un critère d'arrêt prédéterminé est atteint.

Les AG ont été utilisés pour résoudre une grande gamme de problèmes avec beaucoup de succès et ont fait et font encore l'objet de conférences internationales dédiées de renommée telles que GECCO [2020], FOGA [2019] et PPSN [2020].

Les performances des AGs lorsqu'ils sont utilisés pour la résolution de problèmes complexes, et le PPG n'en fait pas exception, repose sur plusieurs aspects. Parmi ces aspects, le codage des individus de la population revêt une importance capitale et sans égale. En effet, l'adoption d'une mauvaise représentation des

solutions peut compromettre le procédé de résolution de manière structurelle Boulif [2019].

Dans la littérature, plusieurs représentations de solutions (codages) pour l'AG sont proposées. Nous avons par exemple, le codage basé sur les sommets Venugopal and Narendran [1992], le codage fractionnaire Goncalves and Resende [2002], la représentation binaire basée sur les arêtes Boulif and Atif [2006], le codage par co-cycles Boulif [2016] pour ne citer que quelques uns. Cette abondance de codages engendre un problème de choix pour distinguer la représentation qui permet de booster au mieux le rendement de l'AG. Apporter une solution à cette problématique fera l'objet de cette thèse. Plus précisément, notre travail porte sur l'étude et la comparaison des différents codages existants. Aussi, nous essayons de proposer par la suite de nouveaux codages dans la perspective de faire avancer l'état de l'art. Notre étude comparative utilise un ensemble d'instances de différentes tailles tirées de la littérature.

Notre travail est organisé comme suit :

Le *chapitre 1* présente les notions théoriques liées au problème de partitionnement de graphes, avec un survol sur quelques domaines d'application.

Le *chapitre 2*, traite les différentes méthodes de résolution du PPG, avec leurs deux grands axes, à savoir, les méthodes exactes et les méthodes approchées, en mettant l'accent sur les approches évolutionnaires.

Le *chapitre 3*, sera entièrement consacré à l'étude des AGs. Nous présentons leurs principes de bases, avec une description détaillée de chacun de leur concept notamment la génération de la population initiale, les opérateurs génétiques, le paramétrages, etc.

Le *chapitre 4*, présente une liste non exhaustive des propriétés théoriques des représentations génétiques. Une description des espaces génotypique et phénotypes avec leurs particularités est également présentée.

Le *chapitre 5*, présente les codages génétiques étudiés dans le cadre de cette thèse. Nous donnons une description détaillée de chacun de ces codages avec leurs modes de fonctionnement et leurs particularités.

Notre étude comparative est présentée dans le *chapitre 6* avec explication des méthodes et stratégies utilisées pour les différentes approches.

Enfin, nous concluons notre thèse avec un retracement des résultats clés présentés dans cette thèse avec quelques perspectives pour des travaux futurs.

1 INTRODUCTION AU PROBLÈME DE PARTITIONNEMENT DE GRAPHES

1.1 Introduction

Dans ce premier chapitre, nous donnons quelques notions de bases pour situer le problème de partitionnement de graphes. Dans un second temps, nous nous intéressons aux différents domaines d'application de PPG dans le but d'illustrer son importance et ses apports à des domaines très variés. À la fin de ce chapitre nous présentons un survol sur la NP-Complétude de certaines variantes de PPG.

1.2 Optimisation

L'optimisation est omniprésente dans différents domaines tels que les sciences de l'ingénieur, la médecine, l'astronomie, etc. Du et al. [2009]. Il s'agit d'une branche des mathématiques, ou plus précisément, de la recherche opérationnelle et des mathématiques appliquées. Elle consiste à positionner, modéliser et résoudre des problèmes. La position du problème permet de bien le décrire. Ensuite, la modélisation associe au problème posé un modèle mathématique qui est une abstraction du problème originel précisant les variables de décision, l'objectif et le cas échéant les contraintes devant être respectées par toute solution pour qu'elle soit admissible. L'étape de résolution cherche parmi l'ensemble de solutions celle qui permet la minimisation (ou la maximisation) de la fonction objectif définie sur l'ensemble des variables de décision.

La formulation mathématique des problèmes d'optimisation est donnée comme suit : Soit $f\colon A \to \mathbb{R}$ une fonction de l'ensemble de solutions admissibles A vers l'ensemble des nombres réels $\mathbb{R}$. L'objectif est de trouver $x^* \in A$ tel que : $f(x^*) \leq f(x), \forall x \in A$ ($f(x^*) \geq f(x)$ pour les problèmes de maximisation) Papadimitriou and Steiglitz [1998].

1.3 Optimisation combinatoire

Selon la nature des variables de décision, on distingue deux types de problèmes, à savoir, les problèmes d'optimisations continues où les variables de décision prennent leurs valeurs dans $\mathbb{R}$ et les problèmes d'optimisation combinatoires ou discrètes.
L'optimisation combinatoire se distingue par le fait que les variables de décision sont définies dans des ensembles dénombrables en général finis. Un problème d'optimisation combinatoire se définit à partir d'un triplet $(E; p; f)$ où :

- E : un ensemble discret appelé espace de solutions (ou espace de recherche) ;
- p : un prédicat sur E, i.e. une fonction de E dans $\{vrai, faux\}$;
- f : une fonction de E dans $\mathbb{R}$ ($f : E \to \mathbb{R}$) qui fait associer à tout élément $x \in E$ un coût $f(x)$. f est appelée *fonction objectif* ou *fonction de coût*.

p permet de créer un ensemble $E_a = \{x \in E$ tel que $p(x)$ est vrai$\}$. L'ensemble E_a est appelé l'ensemble des solutions admissibles du problème.
Il s'agit de trouver l'élément $\tilde{x} \in E_a$ qui minimise f :

$$f(\tilde{x}) = min_{x \in E_a} f(x). \tag{1.1}$$

Remarque : Lorsqu'il s'agit de maximiser, i.e. de chercher l'élément x de E_a qui maximise la fonction objectif, il suffit pour revenir à la forme de minimisation de réaliser la transformation suivante :

$$max_{x \in E_a} f(x) = -min_{x \in E_a}(-f(x)). \tag{1.2}$$

1.4 Partitionnement de graphes

Le partitionnement de graphes est l'un des problèmes génériques d'optimisation combinatoire. Un graphe $G = (S, A)$ est un couple d'ensembles. Le premier, S, est dit l'ensemble de sommets de G. Le deuxième, A, dit ensemble d'arêtes, est un ensemble de couples non ordonnés de S. Nous considérons des graphes pondérés. C-à-d, chaque arête est munie d'une valeur réelle dite poids de l'arête.
De manière non formelle, partitionner un graphe consiste à regrouper ses sommets en sous-ensembles disjoints. La partition doit respecter un ensemble de contraintes pour qu'elle soit admissible. L'objectif est de trouver une partition qui optimise une fonction qui dépend des poids des arêtes.

1.4.1 Formulation mathématique

Soit $G = (S, A)$ un graphe simple non orienté, où $S = \{s_1, s_2, \ldots, s_n\}$ représente l'ensemble de ses sommets et $A = \{a_1, a_2, \ldots, a_m\}$ est l'ensemble de ses arêtes. L'ensemble $W = \{w_1, w_2, \ldots, w_m\}$ définit sur $A \to \mathbb{R}$ associe à chaque arête dans l'ensemble A un poids positif ou nul. Partitionner le graphe G consiste à décomposer l'ensemble de ses sommets S en k sous-ensembles $(k > 1)$. On Appelle l'ensemble $P = \{C_1, C_2, \ldots, C_k\}$ la partition du graphe G vérifiant les conditions suivantes :

$$\forall i \in \{1, 2, \ldots, k\} : C_i \neq \emptyset \tag{1.3}$$

$$\cup_{i=1}^{k} C_i = S \tag{1.4}$$

$$C_i \cap C_j = \emptyset, \forall i, j \in \{1, 2, \ldots, k\}, i \neq j \tag{1.5}$$

Les éléments de P sont appelés des clusters Schulz [2016].
La fonction objectif à minimiser est la somme des poids des arêtes inter-clusters A_P (appelée aussi la coupe de la partition, i.e. les deux extrémité des arêtes de A_P se trouvent dans deux clusters différents).

$$min(\sum_{i/a_i \in A_P} w_i) \tag{1.6}$$

Bichot and Siarry [2011]

Dans cette formulation le nombre de clusters k peut être fixé au préalable ou considéré variable. En outre, en ajoutant une restriction sur la taille (nombre de sommets) des clusters, on retrouve la variante appelée k-way partition qui représente le cas général du problème de partitionnement de graphes. En modifiant cette restriction et/ou en la changeant par d'autres, on obtient d'autres variantes. Dans ce qui suit, on présente certaines parmi les plus utilisées dans la littérature.

1.4.2 Partitionnement équilibré

Le partitionnement équilibré est similaire au k-way-partition sauf que la contrainte de taille des clusters est remplacée par une contrainte d'équilibrage imposant que cette taille soit la même pour tous les clusters à un ϵ près, i.e. chaque sous-ensemble contient au plus $(1+\epsilon) \times \frac{|S|}{k}$ Andreev and Racke [2006] avec $0 \geq \epsilon \leq 1$

1.4.3 Bi-partitionnement

Si pour la variante partitionnement équilibré on fixe le nombre de clusters à deux, on obtient la variante bi-partitionnement de graphes. Dans ce cas, la taille des deux clusters doit être la même (plus ou moins un, pour les graphes d'ordre impair). Même avec cette restriction du nombre de clusters, le problème est toujours NP-difficile.

1.5 Applications

GPP a été appliqué dans plusieurs domaines depuis fort longtemps tels que l'industrie, l'informatique, l'électronique, etc. Mais avec l'évolution de la technologie, de nouveaux domaines d'application (et sous-domaines) sont apparus.

Dans ce qui suit nous présentons quelques uns.

1.5.1 Résolution des systèmes d'équations linéaires à matrices creuses

Un système d'équations linéaires est constitué d'un ensemble d'équations linéaires qui portent sur les mêmes inconnues. L'objectif est de trouver les valeurs des inconnus qui satisfassent toutes les équations simultanément. L'ensemble des coefficients de ce système forment généralement une matrice creuse. La multiplication Matrice-Vecteur est utilisée pour la résolution d'un tel système. Par ailleurs, La taille de la matrice a un impact direct sur le temps d'exécution. Á cet effet, le PPG est utilisé pour décomposer la matrice creuse en sous-matrice dans le but de paralléliser les calculs et réduire ainsi le temps d'exécution.

Le PPG classique s'applique sur les matrices carrées et symétriques, chose qui n'est pas toujours respectée par les systèmes d'équations linéaires. Les chercheurs ont opté pour les hypergraphes pour la modélisation et la résolution de ce calcul. Les hypergraphes permettent la connexion de plusieurs sommets via une seule arête appelée *hyper-arête* ou *net* (réseau). Pour calculer $A \times X$ de taille $(m \times n)$ et $(n \times 1)$ respectivement, deux modèles peuvent s'appliquer à savoir *column-line* ou *row-line*.

Pour le premier modèle, la matrice A est représentée par un hypergraphe $\mathcal{H}_R = (S_R, N_C)$ où chacune de ses lignes est représentée par un sous-ensemble de sommets de S_R tandis que ses colonnes sont représentées par un sous-ensemble de nets dans N_C. Dans le modèle *row-line* l'hypergraphe est défini par $\mathcal{H}_C = (S_C, N_R)$ où les lignes sont représentées par des sommets dans S_C. et les colonnes par des nets dans N_R. Ainsi, on a un seul sommet s_i et un seul net n_j pour chaque ligne i et colonne j pour les modèles *row-line* et *column-line* respectivement Murni et al. [2017]. La matrice A est convertit ensuite en bloc de sous-matrices de taille $K \times K$ en utilisant des techniques de partitionnement des hypergraphes Lin and Hsiao [2004].

1.5.2 Traitements parallèles

L'utilisation efficace de la mémoire partagée pour les processeurs parallèles nécessite l'équilibrage des charges de calculs entre ces derniers de sorte à minimiser les communication inter-processeurs. Le partitionnement de graphe est au centre des opérations de mappage (tâches/processeurs). On présente un calcul par un graphe non orienté et pondéré $G = (S, A)$. Chaque sommet $s_i \in S$ correspond à une tâche de calcul à exécuter sur un seul processeur. Les poids des sommets $p_s(s_i)$ représentent le temps d'exécution des tâches. P_s est la somme des poids de tout les sommets du graphe. Une arête $a_{ij} \in A$ connectant les deux sommets s_i et s_j si l'une des deux tâches nécessite un input de la deuxième. Les arêtes possèdent un poids positif $p_a(a_{ij})$ représentant la quantité de données à acheminer entre les deux sommets formant ses extrémités. Si chacune des deux tâches nécessite des données de l'autre alors le poids de l'arête sera la somme des deux quantités de données. Partitionner les tâches de calcul sur l'ensemble des processeurs signifie l'affectation de chaque sommet du graphe à un processeur de sorte à réduire la quantité de données véhiculées entre les processeurs Hendrickson and Leland [1995].

1.5.3 Gestion de mémoire par pagination

Le PPG est aussi présent dans le domaine d'organisation de la mémoire. Il est utilisé pour améliorer l'affectation des composantes d'un programme aux différentes pages d'une mémoire paginées. Un programme peut être vu comme un ensemble d'entités inter-connectées. Les entités sont des routines, procédures ou même des simples instructions avec leurs données, ceci dépend du niveau de détail requis. Les connexions entre entités peuvent représenter des flux de données, des flux de contrôles, ou des références d'une entité à une autre. Le PPG est ainsi utilisé pour assurer la meilleure affectation possible des entités d'un programme aux différentes pages de la mémoire. La capacité de stockage des pages mémoires est limitée et doit être prise en considération lors de l'affectation. L'objectif de cette opération est de minimiser le nombre de références entre les objets lier aux différentes pages Kernighan and Lin [1970]

1.5.4 Réseaux complexes

Les différents types de réseaux complexes (informatiques, sociaux, routiers, énergétiques, etc.) représentent un domaine fertile pour l'application du PPG. Ces réseaux sont considérés comme étant des graphes pondérés avec une structure parfois non-triviale créée à partir de la vie réelle ou par des processus de modélisation. Ceci rend la conception d'algorithmes pour ces types d'applications très difficile et présente un certain nombre de challenges Bulucc et al. [2016].

a.Réseaux énergétiques

Les perturbations et la propagation des effondrement électriques (blackout) dans les systèmes de réseaux énergétiques peuvent causer des catastrophes. Afin de prévenir les dégâts et/ou de mieux les gérer une approche consistant à décomposer le réseau énergétique en sous-réseaux autonomes est utilisée Li et al. [2005]. Souvent, la maximisation du degré d'autonomie combiné avec celle de la minimisation du coût de recours au délestage (Shedding) améliore nettement la robustesse du système et réduit l'impact des effets de propagation des pannes électriques Li and Liu [2009].

b.Réseaux géographiquement intégrés

La prolifération des appareils de géolocalisation par GPS génère un accroissement rapide de la diffusion des données à travers les réseaux. Ces données doivent être analysées par des algorithmes très rapides. Les nœuds de ces réseaux représentent des emplacements géographiques et les liens entre nœuds sont les flux telles que la migration, les trajectoires des véhicules, et les activités des peuples.
Dans les problèmes en relation avec les données spatiales et les réseaux géographiques, des contraintes liées à la contiguïté spatiale sont souvent considérées. Le PPG est ainsi appliqué afin de réduire les dimensions des ensembles de données en regroupant les items de données similaires dans un seul cluster tout en préservant la distribution globale des données Guo [2009].

c.Réseaux biologiques

Plusieurs systèmes biologiques complexes telles que l'interaction entre les protéines et la co-expression entre les gènes, peuvent être représentés en utilisant des graphes. Dans ce type de réseaux les nœuds sont les entités biologiques telles que les gènes ou les protéines et les arêtes correspondent à l'existence de propriétés communes dans le processus biologique visé Junker and Schreiber [2011], Mondaini [2010].

d.Réseaux sociaux

L'identification des structures communautaires est un sujet d'actualité dans l'étude des réseaux sociaux. Dans ce problème, on spécifie rarement le nombre de clusters à priori. Malgré quelques différence liées à la considération d'indices de groupement (Modularité, edge betweeness, ...) comme fonction objectif, les méthodes de partitionnement de graphes sont souvent au cœur de la plupart des techniques de détection de communautés et sont souvent utilisées comme premières approximations. Les entités du réseaux représentent les sommets du graphe, et les arêtes sont modélisées par l'existence de relation entre les entités (amitié, appartenance au même group, ...) Newman [2013] Fortunato [2010].

1.5.5 Réseaux routiers

Le partitionnement de graphes est utilisé pour simplifier la résolution des problèmes de transport Lauther [2004] Mohring et al. [2007] Kieritz et al. [2010] Delling et al. [2011] Luxen and Schieferdecker [2012] Delling and Werneck [2013]. Les sommets du graphe représentent les intersections des routes tandis que les arêtes représentent les segments des routes délimités par deux intersections successives.

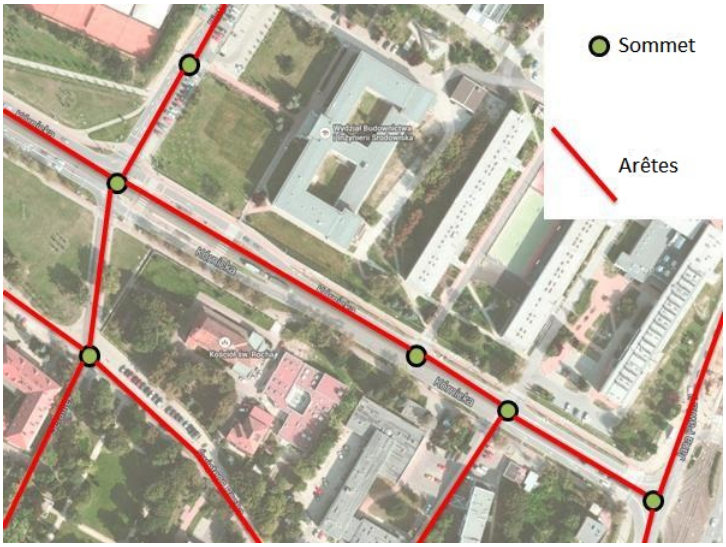


FIGURE 1.1 – *Optimisation des réseaux routiers en utilisant le PPG*

La méthode multi-niveaux proposée par Schulz et al. [2002] pour les problèmes de routage a été l'une des premières techniques d'accélération des algorithmes de recherche de plus court chemin par des routines de pré-traitement utilisant un partitionnement de graphes. Elle a été améliorée ensuite dans Delling et al. [2011] Delling and Werneck [2013]. L'algorithme présenté dans Lauther [2004] utilise une approche de partitionnement géométrique pour réduire l'espace de recherche pour l'algorithme de plus court chemin. Cette méthode a été améliorée par Möhring Mohring et al. [2007].

1.5.6 Traitement d'images

Pendant les deux dernières décennies la représentation d'image basée sur les graphes est devenue très populaire. Dans cette représentation, chaque pixel de l'image (dans certains cas un groupe de pixels) correspond à un nœud du graphe. Deux nœuds sont connectés par une arête pondérée par le degré de similitudes entre eux s'ils sont associés à des pixels adjacents. Basé sur cette formulation, le PPG est utilisé pour réaliser une segmentation d'image qui est une tâche fondamentale de traitement d'images. Son objectif est de partitionner l'image en sous-ensemble de pixels afin d'identifier les objets qui la composent. Parmi les algorithmes de partitionnement de graphes les plus réussis pour la

segmentation d'image on cite les approches spectrales et multiniveaux Basavaprasad and Hegadi [2012], Peng et al. [2013].

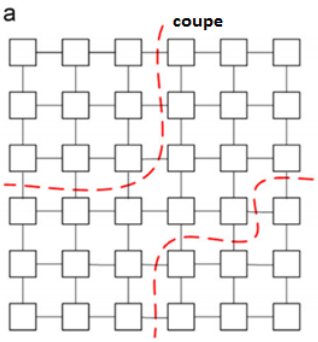
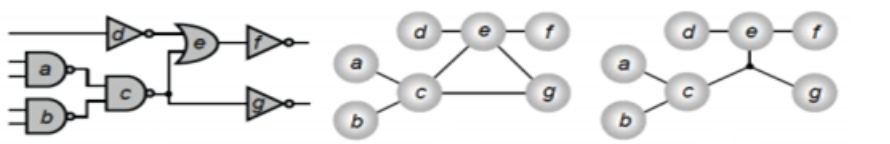


FIGURE 1.2 – *PPG pour la segmentation d'images.*

La Figure 1.2, illustre un exemple de coupe d'un graphe. L'algorithme de segmentation définit les frontières entre les régions de l'image par la mesure d'un indice utilisant : la différence d'intensité à travers les limites entre les régions (frontières) ou la différence d'intensité entre les pixels voisins d'une même région Basavaprasad and Hegadi [2012]

1.5.7 Conception de circuits VLSI

La présence de millions de composants dans les circuits VLSI (Very Large Scale Integration) a rendu leur conception une tâche très difficile. Afin de réduire le coût de cette conception, les composants du circuit sont regroupés en blocs. Dans cette application, les sommets sont les portes logiques tandis que les fils électriques représentent les arêtes. Puisqu'un fil peut relier plusieurs portes logiques, les hypergraphes sont souvent utilisés Kahng et al. [2011].



a) Circuit électronique

b) Représentation du circuit par un graphe non orienté

c) Représentation du circuit pas un hypergraphe

FIGURE 1.3 – *PPG pour la conception de circuits VLSI.*

1.5.8 Découpage d'espace aérien

L'espace aérien est décomposé en secteurs géographiques élémentaires nommés secteurs de contrôle. Ces secteurs sont regroupés en zones de qualification. Le problème de découpage de l'espace aérien s'appuie sur la configuration actuelle des secteurs de contrôle et des routes aériennes. Le but de ce découpage est d'augmenter la sûreté ainsi que la fluidité du contrôle aérien. La modélisation de ce problème se fait en considérant chaque secteur de contrôle comme étant un sommet de graphe. Ensuite, à chaque flux entre deux secteurs est associée une arête pondérée par le nombre d'avions de ce flux. La fonction objectif est une combinaison du flux entre les zones de qualification à minimise et du flux entre les secteurs d'une même zone à maximiser Bichot [2007].

1.6 COMPLEXITÉ

La théorie de la complexité et la conception d'algorithmes efficaces sont deux domaines de l'informatique qui tracent les limites entre ce qui peut et ce qui ne peut pas être mis en œuvre de manière utile comme programme résolvant un problème donné Wegener [1987]. Nous distinguons deux notions souvent confondues par les non spécialistes, à savoir, la complexité d'un algorithme et la complexité d'un problème. La complexité d'un algorithme est mesurée par le nombre d'opérations élémentaires et le besoin de stockage requis par l'exécution de l'algorithme dans le pire des cas. Un algorithme efficace pour la résolution d'un problème est lui même une preuve de la complexité polynomiale de ce

problème. La détermination de la complexité d'un problème permet d'éviter d'essayer de fournir des efforts pour la recherche d'un algorithme efficace car un tel algorithme risque d'être inconcevable. C'est notamment le cas des problèmes NP-difficiles et il s'avère que le problème de partitionnement de graphes en fait partie. Pour déterminer la complexité d'un problème, on utilise souvent la restriction ou bien la réduction polynomiale à partir d'un problème de satisfaisabilité.
Le PPG avec sa variante supervisée est l'un des problèmes NP-complets cités dans le livre de référence Garey and Johnson [1979]. Dans ce qui suit nous présentant la complexité de certaines variantes du PPG.

a. Bipartition équilibrée

Le problème de bipartition équilibré appelé aussi "Minimum Balanced Graph Partitioning" est NP-Complet Garey and Johnson [1979] Andreev and Racke [2006]. Les auteurs ont utilisé une réduction polynomial à partir du problème *Coupe Maximale Simple* qui est une forme simplifiée du problème de *coupe maximale* qui sont tous des problèmes NP-Complet Garey et al. [1974].

b. k-partition équilibrées

Pour une $(k, 1)$-partition équilibrées, la taille des clusters est égale $\frac{|V|}{k}$. Andreev and Racke [2006] a utilisé une réduction à partir du problème de 3-partitions pour prouver l'appartenance du problème de $(k, 1)$-partition équilibrées à la classe NP-Complet.

c. k-partition équilibrée avec la contraint d'équilibrage relaxée

En utilisant le facteur ϵ, la contrainte sur la taille des clusters de k-partition est relaxée. En effet, la taille maximale permise est égale à $(1 + \epsilon) \times \frac{|V|}{k}$. Garey and Johnson [1979] Even et al. [1999] ont utilisé une réduction à partir d'une instance du même problème avec $(\epsilon > 1)$ pour prouver la NP-Complétude pour $(\epsilon \leq 1)$. Pour $(\epsilon \geq 1)$ Even et al. [1999] propose un algorithme en $\mathcal{O}(\log n)$.

d. k-partition non supervisé

Pour le partitionnement non supervisé, la valeur de k n'est pas connue à priori, par conséquent l'espace de recherche augmente considérablement ce qui rend les problèmes précédents encore plus difficiles Andreev and Racke [2006].

1.7 Conclusion

Dans ce chapitre, nous avons présenté le problème de partitionnement de graphes, ses variantes, ainsi que quelques domaines d'application. La complexité est abordée en fin de chapitre pour expliquer la démarche de résolution approximative que nous adoptons dans notre thèse. Dans le chapitre suivant, nous présentons des méthodes de résolution tirées de la littérature, en mettant l'accent sur les méthodes approchées, et en particulier sur les approches évolutionnaires.

MÉTHODES DE RÉSOLUTION 2

2.1 Introduction

Dans ce chapitre nous nous intéressons aux différentes méthodes de résolution du problème de partitionnement de graphes. Nous présentons les classifications proposées dans la littérature, ensuite nous discutons des avantages et des inconvénients de ces méthodes. Enfin, nous mettons l'accent sur les approches évolutionnaires qui représentent la piste d'investigation adoptée dans cette thèse.

2.2 Classification des approches de résolution

Suivant l'optimalité de la solution produite, les algorithmes de partitionnement sont divisés en deux catégories. La première appelée **exacte**, regroupe l'ensemble des méthodes fournissant la solution optimale. Le principe de ces algorithme consiste à énumérer toutes les solutions de l'espace de recherche. Pour les petites instances ces méthodes arrivent à la solution optimale dans un temps acceptable. Néanmoins, avec l'augmentation de la taille des instances, ces dernières vont se retrouver avec des espaces de recherche immenses. Ceci conduit à une augmentation considérable du temps de calcul, ce qui les rend inadaptées.
La deuxième catégorie à savoir, les méthodes **approchées** vient donner une alternative des algorithmes exactes pour traiter les grandes instances. Cette catégorie englobe les méthode heuristiques et métaheuristiques. De par la nature de ces

méthodes, l'optimalité n'est pas assurée, néanmoins, la solution produite est généralement de bonne qualité. Dans ce qui suit, nous allons présenter un état de l'art non exhaustif pour chacune des deux catégories.

2.3 Méthodes exactes

La littérature révèle plusieurs méthodes exactes pour le PPG. La majorité de ces dernières sont basées sur la méthode de « séparation et évaluation » (branch and bound) Land and Doig [2010]. Les bornes mises sur les solutions à exclure ou à maintenir sont calculées en utilisant différentes approches telle que la programmation semi-définie Karisch et al. [2000], Armbruster [2007]. Plusieurs algorithmes exactes ont été proposés pour le problèmes de bi-partition. On cite en guise d'exemple Brunetta et al. [1997], Sellmann et al. [2003], Feldmann and Widmayer [2011], Delling et al. [2012], Hager et al. [2013]. Pour le problème de k-partitionnement, nous avons Ferreira et al. [1998], Sensen [2001].
Par ailleurs, toutes ces méthodes ne peuvent résoudre le problème qu'avec un temps de calcul prohibitif ; exception faite de certains cas particuliers. Par exemple, le problème de bi-partitionnement de grille bidimensionnelle peut être résolue de manière optimale en un temps polynomial Feldmann and Widmayer [2011], Delling et al. [2011],Delling and Werneck [2013]. Pour le k-partitionnement, le temps de calcul de ces méthodes augmente d'une façon exponentielle avec la taille des graphes.

2.4 Méthodes approchées

Pour cette section, nous avons adopté la classification suivante : premièrement les méthodes approchées en leur globalité sont décomposées en deux grandes catégories notamment les *heuristiques* et les *métaheuristiques*. Suivant la façon dont les algorithmes abordent le PPG, la première catégorie est décomposée en trois sous-classes à savoir les algorithmes constructif, les algorithmes de raffinages et la stratégie mutliniveau. Les métaheuristiques sont à leur tour décomposées en méthodes à solution unique utilisant une stratégie de recherche

locale, et méthodes à population. Cette dernière classe contient les approches évolutionnaire en plus d'autres méthodes.

2.4.1 Heuristiques

Les heuristiques appliquées au PPG sont généralement basées sur le déplacement des sommets d'un cluster à un autre. Ces déplacements se font d'une manière successive tout en préservant l'équilibrage des parties. Les déplacements des sommets sont décidés sur la base d'une fonction appelée gain. Le gain (i.e. variation du coût de coupe) de déplacement d'un sommet s à partir d'un cluster à un autre est donné par la formule :

$$g(s) = ext(s) - int(s) \text{ avec } \begin{cases} int(s) = \sum\{\omega(\{s,s'\})|s' \in C(s)\} \\ ext(s) = \sum\{\omega(\{s,s'\})|C(s) \neq C(s')\} \end{cases}$$

Avec $\omega(\{s,s'\})$ est le poids de l'arête $\{s,s'\}$ et $C(s)$ le cluster du sommet s.
Le calcul du coût de coupe d'une partition peut être implémenté d'une manière trivial avec une complexité de $O(|A|)$. Cependant, le partitionnement nécessite plusieurs déplacements, ce qui augmente considérablement cette complexité. Ceci peut être géré par l'utilisation d'une technique plus subtile qui ne considère que les sommets déplacés. Ainsi, la complexité est réduite au degré du sommet déplacé.

a. Les Algorithme constructif (Algorithme général)

Un algorithme requière en générale une solution initiale pour commencer l'optimisation. La façon la plus simple pour construire une telle solution est de la générer aléatoirement. Pour ce faire, on scrute l'ensemble des sommets pour les affecter à des clusters choisis aléatoirement. Cette affectation doit s'assurer du respect des contraintes notamment l'équilibre des poids des parties. Les solutions générées aléatoirement sont dans la plus part des cas d'une mauvaise qualité. Néanmoins, la complexité de la technique aléatoire qui est de l'ordre de $O(|V|)$ avec sa facilité d'implémentation la rend favorable. En plus, la qualité de la solution peut être améliorée à travers une méthode de raffinage (voir § 2.4.1

). Nous présentons dans les sections suivante certains heuristiques qui utilisent des techniques plus élaborées.

i. Algorithme de Bipartition spectrale

Cet algorithme est utilisé uniquement pour le bi-partitionnement de graphe i.e. une partition de deux clusters. Sa technique est basée sur l'utilisation du vecteur propre calculé en utilisant l'algorithme de Lanczos Cullum and Willoughby [1986] qui est un algorithme itératif dont le nombre des itérations dépend de la précision désirée. Les informations contenues dans le vecteur propre sont ensuite utilisées pour générer une partition initiale Karypis and Kumar [1998]. Le principe de cet algorithme est donné comme suit :
Soit $(S_1, S_2 = S \setminus S_1)$ une bipartition du graphe $G = (S, A)$. On définie le vecteur $x \in \{1, -1\}^n$ par l'équation suivante :

$$(x_i) = \begin{cases} 1, \ i \in V_1 \\ -1, \ i \in V_2 \end{cases}$$

Ainsi, minimiser le coût de coupe de la bipartition revient à trouver le vecteur x qui minimise $x^T Lx$ avec L la matrice Laplacian du graphe G Schulz [2016].

ii. Algorithme glouton

La plus part des algorithmes gloutons proposés pour le PPG possèdent pratiquement la même structure donnée par l'algorithme 1 Brandfass et al. [2013]. Ces derniers se diffèrent juste par leur fonction gloutonne. L'algorithme commence par la création de k parties contenant chacune un sommet appelé grain choisi aléatoirement. Ensuite, à tour de rôle, dans l'ordre de la sélection gloutonne, les sommets libres sont affectés aux k parties précédemment crées. Cependant, la qualité de la solution fournie par ces méthodes est fortement liée aux choix des gains. Pour remédier à ce problème, l'algorithme glouton peut être lancé plusieurs fois pour ensuite prendre le meilleur résultat.

Algorithme 1 : Construction gloutonne.

Input : Un graph $G = (S, A)$, k : le nombre de parties désirées.
Output : Une k-partition de S.
$P \leftarrow \{P_0, ..., P_k\}$
$C' \leftarrow S$
Pour $i \in [0, ..., k-1]$ **faire**
    $u \leftarrow$ un sommet aléatoire de C'
    $P_i \leftarrow \{u\}$.
    $C' \leftarrow C' \setminus \{u\}$.
$p \leftarrow 0$.
While $|C'| > 0$ **faire**
    $b \leftarrow fonction\_gloutonne(C', P, i, G)$.
    $P_i \leftarrow P_i \cup \{b\}$.
    $C' \leftarrow C' \setminus \{b\}$.
    $i \leftarrow i + 1 \bmod k$.
Retourne P.

iii. Expansion de région

Cet algorithme a été introduit par Karypis and Kumar [1998]. Il est basé sur une fonction gloutonne pour la construction de partition. Pour chacune de ses k itérations (k : le nombre de parties) l'algorithme choisit aléatoirement un sommet libre (non encore affecté) comme noyau de la nouvelle partie qu'il commence à élargir à partir de ce noyau. L'expansion de la région autour du sommet choisi est réalisée en utilisant un algorithme DFS (depth first search). Ainsi la partition initiale est formée.

iv. Expansion de région gloutonne

Pour cette méthode Karypis and Kumar [1998] remplace le DFS par une fonction qui minimise le gain total à chaque affectation de sommet. Contrairement à l'expansion de région, à chaque itération, toutes les parties sont scrutées pour en choisir celle qui minimise le gain total. La structure de la fonction gloutonne est donnée par l'algorithme 2. On introduit ici la notion du gain total de l'affectation d'un sommet s à une partie i par la formule :

$$t(s, i') = \sum_{i \neq i'} g_i(s, i) \tag{2.1}$$

Avec $t(s, i')$ le gain total de l'affectation du sommet s à la partie i' et $g_i(s, i)$ le gain d'affectation du sommet s à la partie i.

Algorithme 2 : Fonction gloutonne.

Input :

. une liste de taille S' de sommets non affectés,

. la partition courante P,

. i l'indice de la partie destinataire du sommet,

. un graphe $G = (S, A)$

Output :

. Un sommet qui doit être affecté à la partie P_i.

$m \leftarrow min_{s \in S'}(t_i(s, i))$
$C \leftarrow \{s \in S' | t(s, i) = m\}$
retourne un sommet aléatoire de C ;.

En plus de ce qu'on vient d'avancer, il existe d'autres variantes d'algorithmes de construction glouton tel que *l'algorithme glouton min-max* de Battiti and Bertossi [1999] et *l'algorithme différentiel glouton* de Battiti and Bertossi [1997].

b. Algorithmes de raffinage

Une bipartition parfaitement équilibrée $\{A, B\}$ peut être améliorée juste par la permutation de leurs sous-ensembles $X \subset A$ et $Y \subset B$ avec $(|X| = |Y|)$ de sorte à réduire son coût de coupe. La détermination de X et Y est une tâche très difficile qui nécessite l'utilisation d'une heuristique Schloegel et al. [2000]. Dans les sections qui suivent, nous allons présenté certains des algorithmes de raffinage les plus populaires.

i. Kernighan-Lin (KL)

Kernighan and Lin [1970] avaient proposé l'une des premières méthodes utilisées pour le raffinage des bipartitions. La majorité des algorithmes d'amélioration locale sont basées sur le principe de cette dernière. L'algorithme KL réalise des permutations successives de sommets des deux parties améliorant la qualité de la partition. La permutation permet de préserver l'équilibrage des parties

(nombre de sommets sortants égale aux nombre de sommets entrants). Le choix des sommets à permuter est basé sur une fonction de gain. Ainsi les sommets qui fournissent le plus de gain après leur permutation sont donc choisis. Le calcul du gain d'une permutation d'une paire de sommets est donné par la formule suivante :

$$g(s,s') = g_s + g_{s'} - 2\delta(s,s') \text{ avec } \delta(s,s') = \begin{cases} \omega(\{s,s'\}) \text{ si } \{s,s'\} \in A \\ 0 \text{ sinon} \end{cases} \tag{2.2}$$

La comparaison des gains est faite en valeur absolue. Les sommets choisis sont ensuite verrouillés pour empêcher leur permutation dans les prochaines itérations. L'algorithme s'arrête lorsque tous les sommets sont verrouillés. Un autre tour peut être joué en utilisant la partition finale précédente comme input. Le processus est arrêté lorsque aucune amélioration n'est possible.

ii. Fiduccia-Mattheyes (FM)

La méthode FM Fiduccia and Mattheyses [1982] représente une amélioration de celle de Kernighan-Lin. En terme du temps de calcul, cette heuristique présente une complexité de l'ordre de $O(|A|)$ au lieu de $O(|S|^2)$ pour la méthode KL. Un temps qui augmente linéairement avec la taille du graphe. Cette amélioration du temps de calcul est du à l'utilisation d'une structure de donnée appelée *bucket list*. En plus, les auteurs traitent le cas des sommets pondérés par le déplacement d'un seul sommet à la fois. L'équilibrage des parties est maintenu en se basant sur la taille des parties obtenus par la somme des poids de leurs sommets. Une variante de cette technique pour le k-partitionnement a été proposé par Osipov and Sanders [2010]. Leur technique consiste à utiliser une liste dans laquelle les sommets et leurs gains correspondants sont stockés et ordonnés.

c. Méthode multiniveau

Partitionner les graphes de petite taille est une tâche relativement facile pour la majorité des méthodes de partitionnement. Une technique subtile pour faciliter le partitionnement c'est de réduire la taille des graphes. Hendrickson and Leland [1995] Bui and Jones [1993] indépendamment ont présenté des techniques

pour la réduction de la complexité des graphes. Ainsi, la méthode de partitionnement pourra facilement trouver une bonne solution. La stratégie multiniveau combine en plus de la réduction de la taille du graphe, un algorithme constructif et un algorithme de raffinage. Elle comporte quatre phases consécutives qu'on décrit comme suit :

— **Contraction :** c'est la phase pendant laquelle la taille du graphe est suffisamment réduite.
— **Initialisation :** durant cette phase, un algorithme constructif est utilisé pour fournir une partition initiale du graphe contracté.
— **Décontraction :** retour sur la version initiale du graphe on décontractant les sommets agrégés.
— **Raffinage :** Un algorithme de raffinage est utilisé pour améliorer la qualité de la solution produit durant les phases précédentes.

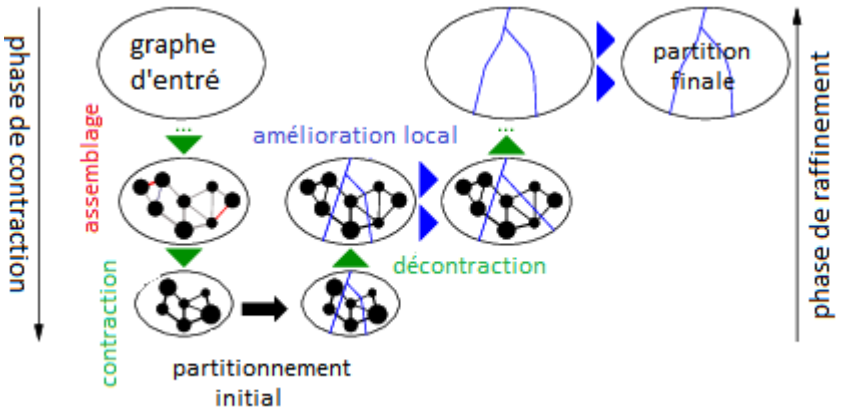


FIGURE 2.1 – *Principe de la méthode multi-niveaux*

L'algorithme 3 de Walshaw [2008] décrit d'une manière formelle la méthode multi-niveaux dans sa version générale.

2.4.2 Métaheuristiques

Ceux sont des méthodes souvent inspirées d'un processus naturel. Elles sont applicables sur un large spectre de problèmes. Leur principe est basé sur l'exploration de l'espace de solutions à la recherche d'un optimum global en utilisant des mécanismes leur permettant de s'échapper des optima locaux. Elles partagent les caractéristiques suivantes :

Algorithme 3 : La méthode multi-niveaux.

Input Une instance P_0 du problème.
Input Une solution pour le problème.
$l \leftarrow 0$
Tant que P_l très grand **Faire**
$P_{l+1} \leftarrow contracter(P_l)$
$l \leftarrow l + 1$.
$C_l \leftarrow initialise(P_l)$
Tant que $l > 0$ **Faire**
$l \leftarrow l - 1$
$C_l^0 \leftarrow decontracter(C_{l+1}, P_l)$
$C_l \leftarrow raffiner(C_l^0, P_l)$
Retourne C_0.

— Leur aspect stochastique facilite l'exploration des espaces de recherche.
— Elles sont facilement adaptables à de nouveaux problèmes.
— La plupart de ces méthodes ont un principe de fonctionnement bio-inspirés.
— Elles requièrent un paramétrage qui est souvent une tâche fastidieuse.

Dans ce qui suit nous présentons les deux classes des métaheuristiques, quelques unes des plus utilisées.

a. Métaheuristiques à solution unique

Les méthodes à solution unique, plus connues sous le nom de méthodes de recherche locale ou encore méthodes de trajectoire, sont basées sur l'évolution d'une seule solution dans l'espace de recherche. La recherche tabou et le récuit simulé deux fameuses métaheuristiques à solutions unique connues de leur efficacité.

i. Recherche tabou (RT)

La méthode de recherche tabou a été proposée par Glover [1989] Glover [1990]. Par la suite plusieurs travaux de recherche l'ont adoptée seule ou combinée avec d'autres heuristiques pour la résolution de PPG Rolland et al. [1996] Benlic and Hao [2011a] Benlic and Hao [2010] Benlic and Hao [2011b] Galinier

et al. [2011]. Le processus de la RT est décrit comme suit : une solution initiale est générée soit aléatoirement ou en utilisant une heuristique. La technique utilisée par Rolland et al. [1996] pour générer la solution initiale consiste à créer une bipartition ou les sommets sont affectés à l'une des deux parties suivant une procédure probabiliste. Si à la fin de cette procédure les deux parties n'ont pas une taille similaire, une procédure d'équilibrage gloutonne est utilisée pour les rendre ainsi. Ensuite, suivant un processus itératif, des sommets sont choisis de l'une des deux parties pour être déplacés vers l'autre. Les sommets récemment déplacés sont interdis au déplacement pendant un certain nombre d'itérations et ce pour éviter de revenir sur des solutions déjà visitées. Afin d'éviter d'être coincé dans un optimum locale, la méthode utilise un facteur de déséquilibre qui autorise une certaine marge de déséquilibre entre les clusters. Ce paramètre doit être réinitialiser périodiquement. L'algorithme 4 de Rolland et al. [1996] résume les étapes principales de la méthode RT pour le PPG.

Algorithme 4 : La recherche Tabou.

Étape 1. $MeilleureSolution \leftarrow$ une solution réalisable générée aléatoirement.
Étape 2. $Iteration \leftarrow 0$;
$MaxIteration \leftarrow Max\{100, 5 \times |S|\}$;
$SansAmelioration \leftarrow 0$;
$MaxSansAmelioration \leftarrow 20$;
$FacteurDeDesiquiblibre \leftarrow 0$;
Étape 3. Tant que $Iteration < MaxIteration$ **alors**
(a) Evaluer $MeilleureSolution$
(b) Choisir un sommet s à déplacer, initialiser $Tabou\_Temps(s)$
(c) Calculer le coût de coupe de la $NouvelleSolution$
(d) **Si** $(NouvelleSolution < MeilleureSolution)$ Et $(|S_1| = |S_2|)$ **alors**
$MeilleureSolution \leftarrow NouvelleSolution$
Sinon
$SansAmelioration \leftarrow SansAmelioration + 1$
(e) **Si** $(SansAmelioration > MaxSansAmelioration)$ **alors**
$FacteurDeDesequiblibre \leftarrow FacteurDeDesequiblibre + 1$
$SansAmelioration \leftarrow 0$
Si $FacteurDeDesequiblibre > 4$ **alors**
$FacteurDeDesiquiblibre \leftarrow 0$
(f) $Iteration \leftarrow Iteration + 1$

ii. Recuit simulé (RS)

Le recuit simulé est apparue au début des années 1980 Vecchi and Kirkpatrick [1983]. C'est une approche basée sur le comportement des systèmes physiques en présence d'un bain de chaleur. Le résultat de cette technique donne une structure cristalline au matériau avec une résistance bien meilleur. Le recuit simulé peut être vu comme une amélioration de l'algorithme de la recherche locale. Le principe de cette dernière se base l'exploration du voisinage directe de la solution en cours. Cette technique présente un fort risque de convergence vers un optimum locale. Afin d'échapper aux minima locaux, le recuit simulé accepte avec une certaine probabilité les solutions dégradées. Ceci permet à cette méthode d'atteindre des optima globaux.
Le recuit simulé peut être utilisé comme méthode principale pour le PPG Johnson et al. [1989] Williams [1991], ou comme outil de raffinage d'une partition Martin and Otto [1996], Diekmann et al. [1996], Baños et al. [2003] Gil et al. [2006]. L'application du RS pour le PPG nécessite la définition des notions suivantes :

— *Solution* : elle peut être n'importe quelle partition du graphe.
— *Voisin* : deux partitions sont voisines si l'une peut être obtenue à partir de l'autre par le déplacement d'un seul sommet d'un cluster à un autre. Par exemple, si $P = \{C_1, C_2\}$ une solution de deux clusters (bipartition) et $s \in C_1$ alors la solution $P' = (C_1 - \{s\}, C_2 \cup \{s\})$ est dite voisine de P.
— *Coût* : c'est la valeur de la fonction objective.

La méthode de recuit simulé présentée par l'algorithme 5 applique d'une manière itérative l'algorithme de *Métropolis* Metropolis et al. [1953]. Les paramètres de la méthode sont donnés comme suit :

— e_{init} : l'état initial du système ou la solution de départ.
— T_{max} : la température initiale du système.
— T_{min} : la température finale du système (système dans son état figé).

Le choix de e_{init} peut se faire soit d'une manière aléatoire, soit en utilisant une heuristique. La température T représente le paramètre de contrôle pour l'acceptation des dégradations. Si T est grand, les solutions dégradées sont acceptées avec une probabilité plus importante. Inversement, pour $T = 0$, aucune dégradation n'est acceptée. La diminution de la température est assurée par une fonction

Algorithme 5 : Recuit simulé.

Procédure $Recuit(e_{init}, T_{max}, T_{min})$
$T \leftarrow T_{max}$
$e \leftarrow e_{init}$
$e^* \leftarrow e_{init}$
Tant que $T > T_{min}$ **faire** {Boucle de décroissance de la température}
  Répéter {Boucle de la stabilisation}
    $e' \leftarrow$ modification élémentaire de e
    $\Delta e \leftarrow energie(e') - energie(e)$
    Si $\Delta e > 0$ **Alors** {Règle d'acceptation de Métropolis}
      $e \leftarrow e'$
    Sinon Si $proba - Boltzman(\Delta e) \leq$ nombre aléatoire $x \in [0, 1]$ **Alors**
      $e \leftarrow e'$
    Fin Si
    Si $energie(e^*) > energie(e)$ **Alors**
      $e^* \leftarrow e$
    Fin Si
  Jusqu'à ce que le système soit figé.
  $T \leftarrow abaisser(T)$
Fin Tant que
Retourner e^*
Fin Procédure.

appelée schéma de refroidissement Johnson et al. [1989]. Dans l'algorithme 5 elle est représentée par la fonction abaisser. Cette dernière peut être définie de différentes manières dont la plus simple est :

$$abaisser(T) = x \times T_{max} \text{ ou } x \in [0.9, 0.99] \tag{2.3}$$

Le système est considéré stable, si pour un certain nombre d'itérations les modifications élémentaires successives sont rejetées par la règle d'acceptation de la Métropolis. Le calcul de cette probabilité est donné par l'équation 2.4.

$$proba - Boltzman(\Delta e) = e^{(-\Delta e/T)} \tag{2.4}$$

b. Métaheuristiques à population

La métaheuristique à population manipule simultanément un ensemble d'individus (solutions) dans l'espace de recherche, où chacun profite de l'expérience du groupe, de manière directe ou indirecte. On peut distinguer deux catégories de métaheuristiques à population : les algorithmes d'intelligence en essaim et les algorithmes évolutionnaires. Dans ce qui suit nous présentons l'algorithme de colonie de fourmis et les algorithmes génétiques comme prototype de chacune des deux catégories cités ci-dessus.

i. Colonies de fourmis

Cette métaheuristique a été proposé par Colorni et al. [1992] pour résoudre le problème de voyageur de commerce. L'adaptation de cette algorithme pour le problème k-partitionnement de graphe est présenté par Bichot [2007]. Cette approche consiste à mettre en concurrence plusieurs colonies de fourmis. La description de cette adaptation est donnée comme suit :
Un graphe $G = (S, A)$ est assimilé au terrain sur lequel plusieurs colonies de fourmis sont dispersées. La nourriture des fourmis est répartie sur les arêtes de graphes (champs) dont la quantité correspond au poids de l'arête. L'ensemble des arêtes de graphe représentent le territoire des fourmis, tandis que chaque sommet représente un carrefour entre plusieurs chemins menant aux champs. Les fourmis ne peuvent pas rester sur les champs qui peuvent par contre les tra-

verser pour récupérer la nourriture. Les fourmis peuvent s'arrêter à n'importe quel sommet du graphe (carrefour). Le but de chaque colonie est de conquérir une région riche en nourriture. Les directions des fourmis sont choisies par rapport à la quantité de phéromone déposée par ses congénères sur les champs entourant le carrefour sur lequel elle se trouve. Une colonie peut détenir un carrefour si la quantité de phéromone signalant les chemins qui l'entoure dépasse celles de toutes les autres colonies. Si les deux carrefours menant à un champ appartiennent à une même colonie, alors on considère que celle-ci possède le champ. La région possédée par une colonie est constituée de l'ensemble des carrefours appartenant à cette colonie. Par conséquent, en suivant ce processus on arrive à former les clusters de la partition qui sont représentés par les régions des colonies. Le nombre de clusters de la partition correspond au nombre de colonies utilisées lors du lancement de l'algorithme. Initialement une quantité fictive de phéromone propre à chaque colonie est déposée sur les chemins pour autoriser la circulation des fourmis. La méthode de colonie de fourmis poposé par Bichot [2007] est présenté dans l'algorithme 6.

La probabilité d'acceptation des solutions dégradées est donnée par la formule suivante :

$$h(P'_k|P_k) = e^{\frac{f(P'_k)-f(P_k)}{T}} \tag{2.5}$$

où T est une constante qui permet d'ajuster cette fonction. La fonction simulant l'évaporation du phéromone est donnée par la formule suivante :

$$phromones(A \rightarrow B) = \sum_{t=1}^{t_{max}} \frac{1}{t} \times pose(t) \tag{2.6}$$

avec $pose(t)$ est la quantité de la phéromone déposé au tour t et t_{max} la duré de vie maximale du phéromone.

ii. Algorithmes génétique

C'est dans les années 1860s que le naturaliste Charle Darwin avait publié son livre intitulé "L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature", dans lequel il fonde sa théorie par rapport à

l'évolution des espèces naturelles en rejetant totalement l'idée qui régnait à cette époque et qui dit que les espèces naturelles sont figées et elles sont déjà adaptées à leurs environnements, en mettant au point sa nouvelle théorie de l'évolution des espèces naturelles par le biais de la reproduction pour s'adapter graduellement à leurs milieux naturels. Peu après, et plus exactement dans les années 1866s, Mandel publie le résultat de dix années d'hybridation chez les végétaux (combinaison des gènes) et l'adresse à la communauté scientifique des quatre coins du monde, mais cette dernière n'était pas encore prête pour reconnaitre de tel succès, et ce n'est qu'en 1900 qu'un groupe de chercheurs publiaient les mêmes résultats que ceux de Mandel, ce qui a valorisé les travaux de ce dernier. Ces systèmes évolutifs représentèrent par la suite la matière grasse pour John Holland qui commença à les étudier dans les années 1960, pour finir par publier son livre "Adaptation in naturel and artificial systems" Holland [1975] dans lequel il introduit le premier modèle formel pour les algorithmes génétiques "the canonical genetic algorithm", et expliqua l'impact des opérateurs génétiques telles que le croisement et la mutation sur les programmes informatiques, et comment ces derniers leur rajoutent de l'intelligence. Ce travail, devenu par la suite une référence de taille, a été reprit par David Goldberg, qu'il publia en 1989 son livre dans lequel il vulgarisa la théorie des AG Goldberg [1989], et lui rajouta les notions suivantes :

1. Un individu est lié à son environnement par son ADN.
2. Une solution est liée à un problème par son indice de qualité.

Et depuis, les algorithmes génétiques ont été étudier et améliorer par pas mal de chercheurs notamment Z.Michalewicz avec son livre de "Genetic Algorithms + Data Structures = Evolution Programs" Michalewicz [2013]. Les détails de cette méthode ainsi que ses différents principes et mécanismes de fonctionnement seront présentés dans le chapitre suivant.

2.5 Conclusion

Tout au long de ce chapitre, nous avons retracé quelques méthodes utilisées pour la résolution des problèmes d'optimisations notamment le PPG. Une classification de ces méthodes par rapport à leur mécanisme de fonctionnement

et par rapport à la nature de la solution produite est présentée. Les méthodes heuristiques sont plutôt dépendantes des problèmes à résoudre. Par ailleurs les approches évolutionnaires notamment les AGs ont montré leur généricité, simplicité et efficacité à résoudre des problèmes complexes. Durant cette dernière décennie, plusieurs travaux de recherches ont opté pour ces méthodes pour résoudre le PPG tels que Yoon and Kim [2014], Wang et al. [2015],Biedermann et al. [2018], etc.

Dans le chapitre suivant, nous allons aborder en détail les algorithmes génétiques que nous allons utiliser comme méthode de résolution du problème de partitionnement de graphe.

Algorithme 6 : Colonies de fourmis pour le problème de partitionnement de graphe.

Procedure $ColonieDeFourmi(G = (S, A), k)$
$P_k \leftarrow$ initialisation de la partition en k parties.
Répète Boucle externe
  Pour toute colonie de fourmis c_i **faire**
    Pour toute fourmi f_j^i de la colonie c_i **faire**
      choisir un chemin partant du carrefour où se trouve la fourmi f_j^i
      f_j^i dépose une quantité de phéromone égale à celle de nourriture du champ
      Si au moins un sommet change de partie à cause de la quantité de phéromone **Alors**
        $P_k \leftarrow$ mettre à jour la partition en fonction des sommets déplacés.
        Si $f(P'_k) < f(P_k)$ ou $h(P'_k|P_k) > random(1)$ **Alors**
          conserver le nouveau déplacement
          $P_k \leftarrow P'_k$
        Sinon
          rejeter le déplacement et considérer que la fourmi f_j^i est restée immobile.
        Fin Si.
      Fin Si.
    Fin Pour.
  Fin Pour.
  évaporation de la phéromone.
jusqu'à critère d'arrêt.
Retourne P_k.
Fin Procédure.

ALGORITHMES GÉNÉTIQUES 3

3.1 Introduction

Les algorithmes génétiques (AG) font partie des méthodes de résolution évolutionnaires les plus utilisées. Ils ont été proposés dans leur forme moderne par Holland [1975] puis vulgarisés par son étudiant David Goldberg Goldberg [1989]. Par la suite, plusieurs variantes ont été proposées par modification des opérateurs ou par hybridation. Dans ce chapitre nous décrivons leur mécanisme de fonctionnement afin de bien situer notre contribution qui vise à améliorer leur rendement dans la résolution du problème de partitionnement de graphes.

3.2 Principe des AG

Les AG sont inspirés de l'évolution des espèces vivantes. Ce dernier favorise la survie des individus les mieux adaptés à leur environnement. Les survivants transmettent leur code génétique à leur progéniture via l'appariement ce qui améliore leurs capacités d'adaptation.
Les AG sont connus pour leur efficacité à résoudre des problèmes complexes. Pour se faire, on choisit une population initiale d'individus où chacun représente une solution potentielle du problème posé. Le choix de la population initiale peut se faire d'une manière aléatoire ou en utilisant une heuristique. Ensuite, des opérateurs génétiques (sélection , croisement, mutation) sont appliqués sur les individus de cette population pour en produire une nouvelle. Ce processus est itéré jusqu'à la satisfaction d'un certain critère d'arrêt. La forme générale

d'un AG est donnée dans l'algorithme 7 Michalewicz [2013].

Algorithme 7 : Forme canonique de l'AG.

Procédure *Algorithme Genetique*
Début
$t \leftarrow 0$ {la première génération}
$Initialiser(P(t))$ {la population initiale }
$Evaluer(P(t))$ {évalutation de la population initiale }
Tant que (critère d'arrêt non satisfait) **faire**
Debut
$t \leftarrow t + 1$ {la prochaine génération}
sélectionner $P(t)$ à partir de $P(t-1)$ {opérateur de sélection}
$AltrerP(t))$ {opérateur génétique}
$Evaluer(P(t))$ {évaluer la nouvelle génération}
Fin tant que
Fin Procédure.

La figure 3.1 présente une illustration graphique du principe des AGs, ou des individus de la génération k sont sélectionnés pour subissent les opérateurs génétiques afin de produire la génération suivante.

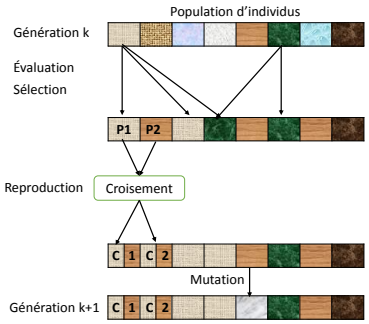


FIGURE 3.1 – *Forme simple des Algorithmes génétiques*

3.3 Particularité des AG

Les AG se distinguent des méthodes traditionnelles par un ou plusieurs points parmi les suivants Goldberg [1989] :

— L'AG utilise un codage des variables de décision du problème, au lieu des variables elles même.
— Le codage utilisé a une grande influence sur le procédé d'optimisation.
— L'AG opère sur une population de solutions et les traitements qu'il réalise sur ces individus sont hautement parallélisables.
— L'AG utilise la valeur de la fonction objectif sans requérir que celle-ci présente des caractéristiques spéciales comme la dérivabilité ou la continuité.
— Le passage à une nouvelle solution se fait de façon probabiliste en héritant d'informations émanant de plus d'une solution.

3.4 Terminologie et analogie avec la biologie

La terminologie utilisée dans le domaine des AG est similaire à celle de la biologie. Par conséquent, il est judicieux de définir quelques termes que nous allons utiliser dans la suite de ce manuscrit. La composante élémentaire des espèces naturelles est la *cellule*. Le noyaux de cette dernière comportent des *chromosomes* qui sont des chaînes d'*ADN*. Les chromosomes sont à leur tour constitués d'une suite de *gènes*.

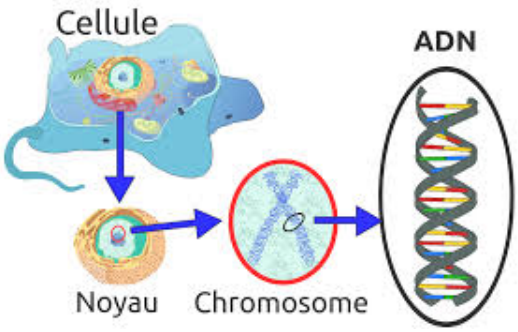


Figure 3.2 – *Analogie des AG avec la biologie*

L'emplacement du gène sur le chromosome est appelé *locus*. L'ensemble des gènes d'un individu représente son *génotype* et l'ensemble du patrimoine génétique d'une espèce est le *génome*. Les différentes versions d'un même gène sont

appelées *allèles*. Par analogie avec le processus de l'évolution, le vocabulaire utilisé par les AG est donné comme suit :

- *chromosome* : une chaine de caractères représentant une solution candidate ;
- *gène* : un emplacement dans le chromosome ;
- *allèle* : une valeur potentielle pour le gène ;
- *locus* : un point de coupure sur le chromosome ;
- *génotype* : La représentation génétique d'une solution en utilisant un codage prédéfinie ;
- *phénotype* : La solution finale après décodage du génotype.

Par ailleurs des propriétés génétiques définissant le comportement des gènes et leur réaction envers les différents opérateurs génétiques (croisement, mutation) trouvent aussi leur place dans le domaine algorithmes génétique. En effet, la causalité, l'épistasie et bien d'autre sont détaillées dans le chapitre 4.

3.5 Codage

La conception d'un AG efficace repose sur plusieurs critères parmi lesquels le codage des solutions joue un rôle vitale. Avec son large spectre d'applicabilité en plus de sa compatibilité avec le théorème des schémas, le codage binaire proposé par Holland [1975] était le premier à être utilisé. Néanmoins il demeure non souhaitable pour certains problèmes. Les solutions illégales (voir §4.4.3) fournies par ce codage nécessite l'invocation d'une routine de réajustement qui consomme un temps d'exécution additionnel ralentissant ainsi l'exécution de l'AG. Par la suite, plusieurs codages ont été présentés tels que le codage entier qui utilise comme alphabet l'ensemble des entiers naturels $\mathbb{N}$, et le codage réel utilisant un alphabet basé sur l'ensemble $\mathbb{R}$.

3.6 Génération de population initiale

La première génération de l'AG contient l'ensemble des solutions de départ qui forment la base du processus de l'évolution. La génération de cette population peut se faire de façon aléatoire ou en utilisant une heuristique. L'utilisation

de cette dernière a pour objectif de faciliter le convergence de l'AG vers une solution optimale. La manière avec laquelle cette population est générée peut influencer le processus de recherche de l'AG. En effet si la population initiale est générée de manière non diversifiée ceci peut conduire à une convergence prématurée ou l'AG sera coincé dans un optimum local.

3.7 Fonction d'évaluation

La formule mathématique permettant le calcule de la fonction objectif donnée par l'équation 1.6 évalue le niveau d'adaptabilité de chaque individu avec son environnement. Le calcule de la fonction objective se fait de manière itérative et précède la sélection des individus de la génération courante. Un choix judicieux de cette dernière est primordial pour éviter de ralentir l'AG. En générale la fonction objectif et la fitness sont les même comme il s'agit d'une simple maximisation (minimisation). Par ailleurs, pour les problème complexes multi-objectifs avec des contraintes il serait nécessaire de choisir d'une manière minutieuse la fonction fitness.

3.8 Pression de sélection

Agissant ensemble la mutation et le croisement permettent à l'AG de mieux explorer l'espace de recherche (diversification de la population), tandis que, la sélection fournit un mécanisme favorisant l'exploitation des informations présentent au sein de la population. Un AG efficace doit assuré le meilleur compromis qu'il soit entre ces deux concepts (diversification/exploitation). D'une manière informelle on définie une forte/faible pression de sélection, le niveau de focalisation de cette opérateur sur les meilleurs individus Back [1994]. Une formule permettant la quantification de la pression de sélection est donnée par l'équation 3.1 Reynès [2007].

$$prssionSlection = \frac{\text{Probabilité(sélectionner le meilleur individu)}}{\text{Probabilité(sélectionner l'individu moyen)}} \tag{3.1}$$

Dans cette même optique, une autre mesure a été présentée par Schlierkamp-Voosen and Muhlenbein [1994] qu'ils ont baptisé *intensité de sélection*. Ils s'appuient pour cela sur une notion de la génétique des populations introduites dans le cadre de l'élevage.

$$IntensitéSélection = \frac{\bar{f}_s(t) - \bar{f}(t)}{\sigma(f(t))} \tag{3.2}$$

— $\bar{f}_s(t)$:est la fitness moyenne des individus de la génération t, sélectionner pour construire la population suivante.
— $f(t)$: est la fitness moyenne de la population à la génération t.
— $\sigma(f(t))$: représente l'écart-type de la fitness au sein de la population.

Pour assurer le meilleur compromis (exploration/exploitation) décrit précédemment, des techniques ont été mis en place telle que la mise à l'échelle des fitness.

3.9 Mise à l'échelle des fitness " Scaling "

La diversification de la population et la pression de sélection sont les facteurs principaux (voir les seuls) qui affectent le processus de recherche de l'AG. Ces facteurs sont fortement liées : l'augmentation de la pression de sélection réduit le diversité de la population et vise versa. Autrement dit, une forte pression de sélection conduit à une convergence prématurée de l'AG. Par ailleurs, une faible pression de sélection rend le processus de recherche inefficace. Ainsi, pour assurer une bonne exploration de l'espace de recherche avec une convergence vers une solution optimale, un compromis entre ces deux facteurs doit être assuré Goldberg [1989]. La mise à l'échelle de la fonction d'adaptation appelée aussi *fitness scaling* a été largement étudiée et utilisée. Elle consiste à modifier la fitness des individus afin de réduire ou d'amplifier les écarts entre les valeurs d'adaptations des individus. La littérature révèle trois types de fitness scaling, à savoir, le scaling linéaire, le scaling exponentiel, et la sigma scaling Goldberg [1989].

3.9.1 Mise à l'échelle linéaire

Cette méthode a été introduite par Goldberg [1989]. La formule générale du scaling linéaire est donnée comme suit :

$$f_s = a \times f_r + b \tag{3.3}$$

tel que : $a = \frac{max' - min'}{max - min}$; $b = \frac{(min' \times max) - (min \times max')}{max - min}$

La signification des variables est donnée comme suit :

— f_s : la fitness modifiée par le scaling (utilisée par la sélection).

— f_r : la fitness réelle de l'individu.

— a, b : ces deux coefficients de l'équation gèrent les écarts entre les fitness des individus de la population. Plus la valeur du coefficient a est inférieure à 1 plus les écarts entre les fitness des individus diminuent assurant ainsi une meilleure exploration de l'espace de recherche.

Bien qu'elle assure une certaine diversification au sein de la population, la nature statique de la mise à l'échelle linéaire freine la convergence vers des solutions dominantes à la fin du processus de recherche.

3.9.2 Mise à l'échelle exponentielle

La méthode exponentielle de la mise à l'échelle vient pallier le problème posé par la méthode linéaire vis-à-vis la convergence vers une meilleure solution. Elle a été proposée par Goldberg [1989]. La formule de cette méthode est donnée comme suit :

$$f_s = (f_r)^{k(n)} \tag{3.4}$$

Où :

— n : est le numéro de la génération courante

— k : Pour une valeur de k proche de zéro, les écarts des fitness sont réduits. Par conséquent, aucun individu n'est favorisé. Pour $k = 1$, le scaling est inopérant. Pour $k > 1$, les écarts entre les fitness sont exagérés, et les meilleurs individus sont favorisés.

— N : le nombre de générations total prévu.

Dans la pratique, on fait généralement varier k des faibles valeurs vers les fortes valeurs au cours des générations. Pour cela on peut utiliser la formule suivante :

$$k = (tan[(\frac{n}{N+1}) \times \frac{\pi}{2}])^p \tag{3.5}$$

3.9.3 Sigma Scaling

Cette méthode a été introduite par Forrest [1985] et modifié par la suite par Goldberg [1989], qu'il avait baptisé *sigma truncation*. Sigma scaling utilise la variance des valeurs de la fonction d'adaptation pour la mise à l'échelle des fitness. Les formules utilisés par cette technique sont comme suit :

$$moyenneFitness = \frac{1}{N} \times \sum_{i=1}^{N} f(x_i) \tag{3.6}$$

tel que $f(x_i)$ est la fitness de l'individu x_i

$$varianceFitness = \frac{1}{N} \times \sum_{i=1}^{N} (f(x_i) - moyenneFitness)^2 \tag{3.7}$$

La variable *varianceFitness* représente la variance des fitness de la population courante qui va servir à calculer l'écart-type des fitness.

$$\sigma = \sqrt{varianceFitness} \tag{3.8}$$

A la fin, la nouvelle valeur de la fitness $g(f)$ des individus de la population courante est calculée à la base de σ en utilisant la formule suivante :

$$g(f) = \frac{(f(x_i) - moyenne(t))}{2\sigma} \tag{3.9}$$

3.9.4 Sigma truncature

Cette technique a été proposée par David Goldberg Goldberg [1989]. Elle représente une amélioration de la technique sigma scaling proposée par Forest en évitant les valeurs négatives produites par cette dernière. La formule permet-

tant de calculer la nouvelle fintess en utilisant cette technique est donnée par l'équation suivante :

$$g(f) = \frac{(f(x_i) - moyenne(t))}{2\sigma} \tag{3.10}$$

Où $g(f(x_i))$ représente la nouvelle fintess de l'individu x_i.
L'intérêt de cette technique est de donner une chance de sélection aux individus les moins adaptés car l'expérience a montré que ces derniers peuvent produire des solutions de bonne qualité.

3.10 Opérateurs génétiques

Les opérateurs génétiques sont fondamentaux pour implanter le processus reproductif caractéristique d'un AG. De nombreux opérateurs génétiques existent ainsi que différentes stratégies d'implantation. Dans les paragraphes qui suivent nous abordons les opérateurs les plus utilisés à savoir : l'opérateur de sélection, de croisement et de mutation.

3.10.1 Sélection

L'opérateur de sélection choisit parmi les individus de la population, ceux qui auront le droit de se reproduire pour donner une nouvelle génération. Cette sélection est faite en se basant sur la valeur de fitness des individus. En conséquence, les individus ayant une meilleure adaptabilité auront plus de chance d'être sélectionner, par contre ceux les moins adaptés auront tendance à disparaitre au fil des générations Holland [1975] Goldberg [1989] Goldberg and Deb [1991].

La littérature présente plusieurs types de sélection. Dans ce qui suit nous citons les plus connus, en mettant l'accent sur "*la sélection par la roue de loterie*" donner par Goldberg Goldberg [1989], que nous utilisons dans nos expériences.

a. Roue de loterie

Pour ce type de sélection, une roue de loterie est créée où chaque individu de la population aura un secteur dont l'angle est proportionnel à sa fitness. La simulation de la roue de loterie suit les étapes suivantes :

1. Calculer la somme des fitness de tous les individus de la population.
2. La probabilité de sélection $ps(a_i)$ d'un individu $a_i \in P = \{a_1, a_2, .., a_n\}$ avec une fitness $f(a_i)$ est donnée par la formule 3.11 Zhong et al. [2005]
   .
   $$ps(a_i) = \frac{f(a_i)}{\sum_{j=1}^{n} f(a_j)}, i = 1, \ldots, n \tag{3.11}$$
3. Partitionner la roue de loterie en secteur correspondants aux probabilités de sélection des individus.
4. On fait tourner la roue et quand cette dernière s'arrête, le secteur pointé par l'aiguille indique l'individu sélectionné.
5. L'étape 4 est répéter autant de fois qu'on veut avoir d'individus pour l'appariement.

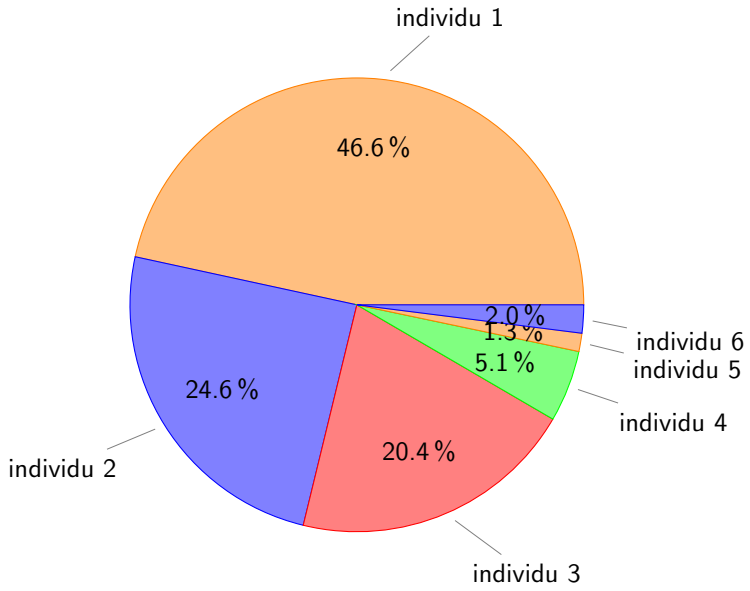


FIGURE 3.3 – *Méthode de sélection par la roue de loterie*

La figure 3.3 est une illustration graphique de la roue de loterie où les individus $\{1, 2, 3, 4, 5, 6\}$ possèdent chacun un secteur proportionnel à sa fitness.

b. Elitisme

La sélection par élitiste, consiste à trier les individus de la population P en ordre décroissant par rapport à leur valeur de fitness. Ensuite les n premiers individus seront choisis pour prendre leurs places dans la nouvelle génération P' Ahn and Ramakrishna [2003], Mashohor et al. [2005], Yang [2007].

c. Tournois

Cette méthode tire aléatoirement deux individus de la population P et les fait combattre, le vainqueur sera celui ayant la meilleur performance. Ce dernier ne sera accepté qu'avec une probabilité comprise entre 0.5 et 1. C'est ce qu'on appel la tournois binaire. La compétition peut s'élargir à un sous-ensemble de N individus. Ceci donne plus de chance d'avoir des individus avec une meilleure adaptation Miller et al. [1995].
Une variante appelée Boltzmann a été proposée par Goldberg et al. [1990]. Elle rappelle certains aspects du recuit simulé car elle fait intervenir la température dans la détermination du vainqueur du tournoi.

d. Sélection par rang (ranking selection)

Proposée par Baker [1985] cette technique consiste à classer les individus de la population par rapport à leur fitness en ordre décroissant. Ensuite, on se basant sur le rang des individus, on leurs affecte le nombre de descendants qui vont avoir à travers l'opérateur de croisement. Le processus de calcul du nombre d'enfants pour chaque individu est donné comme suit :

$$enfant_i = enfant_{max} - (enfant_{max} - enfant_{min}) \times \frac{(rang_i - 1)}{(N - 1)} \tag{3.12}$$

où :

— $enfant_{max}$: Le nombre maximum d'enfants pour le meilleur individu.
— $enfant_i$: Le nombre d'enfants de l'individu i.
— $rang_i$: Le rang de l'individu i.
— N : La taille de la population.

3.10.2 Croisement

Le rôle du croisement est de choisir aléatoirement deux individus parmi ceux sélectionnés, les faire apparier afin d'en produire de nouvelles. La banque génétique des enfants est composée d'un mélange du patrimoine génétique de leurs parents. Cette opération est répétée autant que nécessaire pour produire un nombre d'individus correspondant au taux de croisement $P_c \in [0,1]$ préalablement établi au début du processus évolutif Goldberg [1989]. Dans ce qui suit, nous présentons certains types croisement en mettant l'accent sur celui que nous avons utilisé à savoir l'utilisation d'un seul point de coupure (locus).

a. Croisement simple avec un seul point de coupure

Pour ce type, un point de coupure est choisi de façon aléatoire entre 1 et la taille du $(chromosome - 1)$. Les parents vont se retrouver fragmentés en deux segments, où chacun des segments est échangé avec son homologue de l'autre parent pour en avoir deux descendent avec un nouveau patrimoine génétique comportant une partie des caractéristiques de leurs parents. La figure 3.4 explique le procédé de croisement.

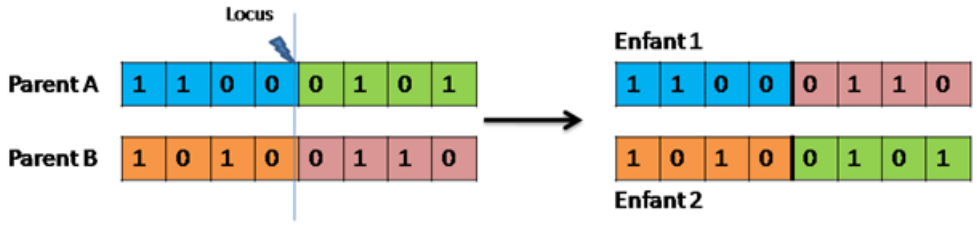


FIGURE 3.4 – *Opérateur de croisement appliqué avec un seul point de coupure*

b. Croisement multipoints

Cet opérateur est une généralisation du croisement à un point. Les points choisis aléatoirement coupent chaque parent en $n+1$ parties (n : nombre de coupures). La construction des chromosomes enfants se fait par le collage alternatif des parties des chromosomes des parents délimités par les points de coupure. La figure 3.5 montre le principe de fonctionnement du croisement multipoint.

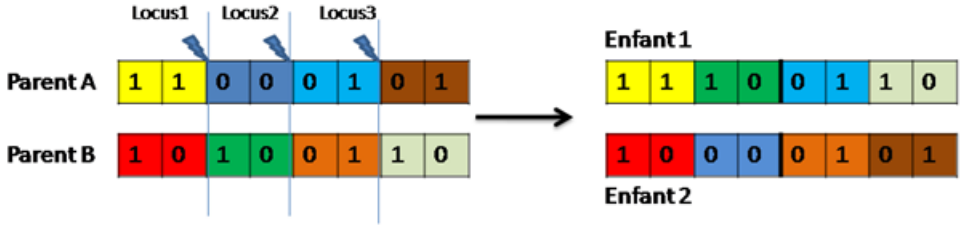


FIGURE 3.5 – *Croisement multipoints*

c. Croisement uniforme

Le croisement uniforme utilise un masque binaire généré aléatoirement avec une taille égale à celle des chromosomes. La valeur du masque décide lequel des parents fournira son gène à leur premier fils. Le deuxième enfant sera construit d'une manière symétrique. Plus concrètement, si la valeur du masque pour un gène donné est égale à 0 alors le premier fils prend la valeur du gène de son premier parent sinon il recevra celui du deuxième. Le second fils prendra toujours l'opposé du premier et ainsi de suite. La figure 3.6 explique davantage ce processus.

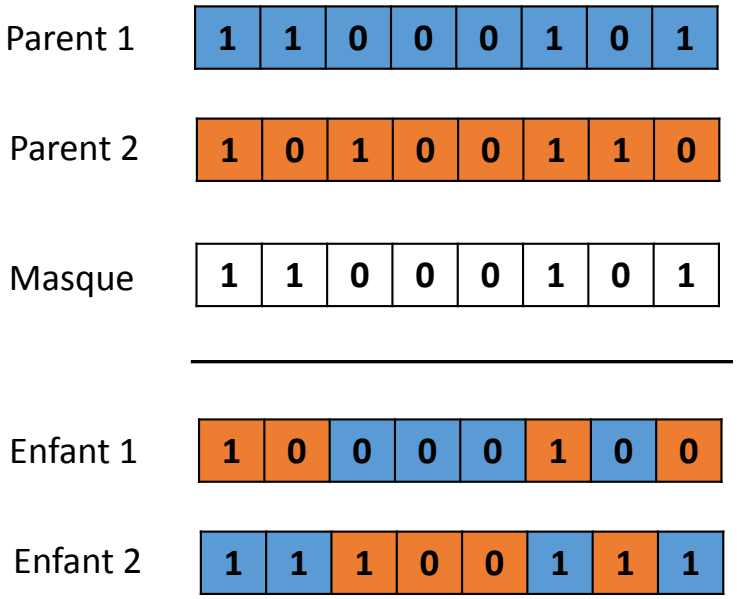


FIGURE 3.6 – *Croisement uniforme*

3.10.3 Mutation

Le principe de la mutation consiste faire un tirage sur les individus de la population avec une très faible probabilité (de l'ordre de 1/1000). Ensuite pour les individus retenus, on choisi aléatoirement un gène de leurs chromosomes puis le modifier. La nouvelle valeur du gène est choisi de façon arbitraire dans l'intervalle de l'alphabet du codage. Par exemple, pour le codage binaire il suffit d'inverser la valeur du gène. La figure 3.7 illustre l'application de l'opérateur de mutation sur un chromosome utilisant un codage binaire.

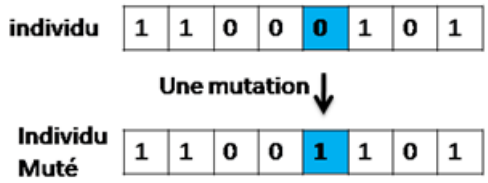


FIGURE 3.7 – *Application de l'opérateur de mutation*

3.11 PARAMÈTRES DE L'AG

La conception d'un AG efficace est étroitement lié au choix de ses paramètre. Ces derniers représentent les probabilité d'application des opérateurs génétiques et la taille de la population. Un bon paramétrage de l'AG garantie la qualité de la solution finale.

3.11.1 Taille de la population

La taille de la population est un paramètre très important dont l'impact est considérable sur les performances de l'AG. En effet, une population avec un nombre d'individus réduit augmente la pression de sélection conduisant l'AG à une convergence prématurée. Par contre, une taille importante peut faire noyer l'AG et augmente considérablement le temps d'exécution. Finalement, Un choix judicieux de cette taille doit prendre en considération le problème posé, le codage utilisé, et la nature des opérateurs génétiques adoptés Arabas et al. [1994] . D'après De Jong [1975] la taille idéale doit être comprise dans l'intervalle

$[50, 100]$. Pour Grefenstette [1986] un nombre d'individus entre 20 et 40 est suffisant. En général cette taille est déterminé d'une manière empirique, néanmoins ceci nécessite un temps de traitement important Hassanat et al. [2019].

3.11.2 Taux de sélection

Pour assurer la convergence de l'AG, le survie des meilleurs individus à travers les génération est nécessaire. De Jong [1975] propose un taux de 0.1% de la population à reproduire à travers les génération.

3.11.3 Taux de croisement

Ce paramètre représente le nombre d'individus à sélectionner pour s'apparier. La valeur 100% veut dire que la nouvelle génération sera formée uniquement par les nouveaux individus crées, tandis que 0% siginifie que l'évolution de l'AG est lié au taux de mutation puisque aucun nouveau individus n' est créé Hassanat et al. [2019]. Les résultats de De Jong [1975] suggère un taux de 60% pour une taille comprise entre 50 et 100 individus. Grefenstette [1986] propose 0.95 pour un nombre d'individus compris entre 20 et 40.

3.11.4 Taux de mutation

L'analyse des interactions entre le taux d'appariement et la mutation révèle une relation entre cette dernière et la taille de la population. Schlierkamp-Voosen [1993] affirme qu'un taux de $1/|P|$ donne de bons résultats. Pour la configuration de De Jong [1975] : taille de la population $\in [50, 100]$, taux de croisement égale à 60% le taux de mutation est de l'ordre de 0.001. D'autre part, celle de Grefenstette [1986] : taille de la population $\in [20, 40]$, taux de croisement est de 95% la mutation n'excède pas les 0.01.

3.11.5 AG sans paramètres

La méthode classique pour l'ajustement des paramètres de l'AG consiste à réaliser des testes. Plusieurs combinaisons de paramètres sont utilisés pour en

choisir la meilleure. Or ceci prend beaucoup du temps, par exemple pour 4 paramètres avec 5 valeurs on aura $5^4 = 625$ combinaisons. Réalisant 100 exécutions pour chacune ça fait 625.000 exécutions Eiben et al. [1999]. Harik and Lobo [1999] ont proposé une solution à cette situation basée sur l'utilisation d'un AG sans paramètre. Contrairement à ce qu'on peut comprendre, un AG sans paramètres vise à dispenser l'utilisateur de la tâche qui consiste à ajuster ses paramètres. Pour ce faire, on dispose de deux possibilités :

— Une formule permettant de fixer les valeurs des paramètres en fonction des données du problème.
— Automatiser le réglage via l'AG à travers les générations.

La première alternative n'est pas évidente à mettre en œuvre à cause des fortes interactions entre les paramètres d'une part, en plus des spécificités liées à chaque problème Harik and Lobo [1999]. Pour la second variante, Harik and Lobo [1999] propose une solution à base de croisement pour l'auto détermination de la taille optimale de la population. Le principe de sa technique consiste à lancer la processus évolutif avec une population d'une petite taille et la laisser converger sans utilisation de l'opérateur de mutation et en fixant le taux de croisement et celui de la sélection. Ensuite on fait doubler la taille de la population et la laisser converger à chaque fois jusqu'à satisfaction de la qualité finale de la solution. Ceci prend le double de la durée nécessaire pour un AG commençant par une taille optimale de la population. Une deuxième solution consiste à mettre en concurrence plusieurs populations de différentes tailles et procéder par élimination. Si par exemple la moyenne des fitness d'une population est plus grande que celle ayant une taille plus petite, ces dernières seront détruites. Parmi les travaux de recherches qui ont utilisé ces AGs, on cite à titre d'example Nadi and Khader [2011], Doerr and Zheng [2020].

3.12 Conclusion

A travers ce chapitre nous avons présenté les algorithmes génétiques, leurs principes, leur mode de fonctionnement et enfin leurs particularités. Un accent est mis aussi sur le paramétrage de ces méthodes et l'utilisation des AG sans

paramètres. Le chapitre suivant, sera consacré à l'études des codages et leurs propriétés génétiques.

4 PROPRIÉTÉS DES REPRÉSENTATIONS GÉNÉTIQUES

4.1 Introduction

La représentation des solutions par des chromosomes est un point très sensible dans la mise en œuvre des AGs. Dans ce chapitre, nous abordons les différents concepts liées aux codages génétiques. Dans un premier lieu, nous commençons par donner une description des espaces génotypique et phénotypique et le lien entre eux. Par la suite, nous mettons l'accent sur le concept de solutions irréalisables et illégales. Les propriétés génétique des solutions sont mise en relief afin de fournir une base d'évaluation des représentations et leur adaptabilité par rapport au problème d'optimisation traité.

4.2 Espaces génotypique et phénotypique

L'algorithme génétique travaille alternativement sur les deux espaces génotypique et phénotypique. Suivant la nature du codage utilisé, le passage du premier espace vers le deuxième nécessite une opération de décodage. Les opérateurs génétiques sont appliqués sur les individus de l'espace génotypique, tandis que l'évaluation des solutions est réalisée par rapport aux phénotypes. L'opérateur de la sélection est le lien entre les deux espaces Gen and Cheng [1996a].

4.3 Solutions irréalisables et solutions illégales

Le décodage des chromosomes pour aller sur l'espace phénotypique peut révéler certaines situations qui rendent l'utilisation du codage inapproprié pour le problème traité. En effet, cette opération peut aboutir à des solutions non réalisables. Une solution est dite non réalisable si elle viole au moins une des contraintes imposées par le problème. Afin d'éviter la prolifération de ces solutions dans la population, l'utilisation d'une technique de pénalisation est nécessaire pour défavoriser ces dernières Gen and Cheng [1996b], Michalewicz [1995].

De l'autre côté, on a la notion des solutions illégales. En effet, le décodage de certains chromosomes produit des solutions qui n'ont pas de représentant dans l'espace phénotypique. Les techniques de pénalisation des solutions ne permettent pas de gérer ce problème. À la place, des techniques de réajustement (réparation) des chromosomes sont utilisées. Dans la figure 4.1 nous illustrons le concept des solutions irréalisables et celui des solutions illégales Orvosh and Davis [1994].

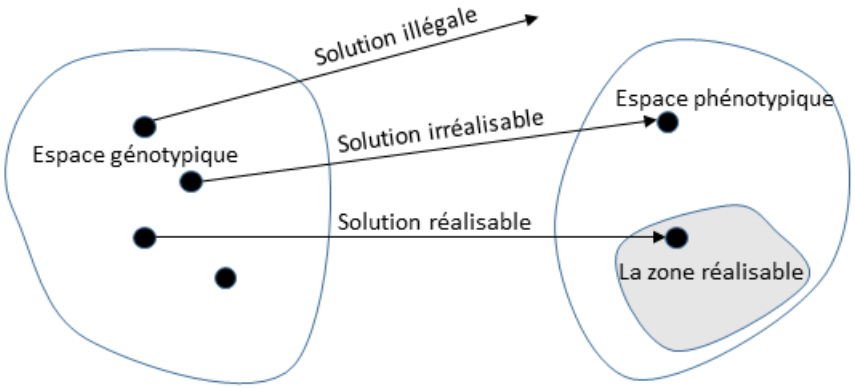


Figure 4.1 – *Les solutions irréalisables et les solutions illégales*

4.4 Propriétés des codages

L'élaboration d'une nouvelle représentation génétique pour un problème d'optimisation doit respecter un certain nombre de critères. La qualité du codage dépend du nombre et de la nature des critères respectés. Dans ce qui suit

nous citons une liste non exhaustive de critères Gen and Cheng [1996a] telles que :

— La non redondance.
— La cécité (Complétude).
— La légalité.
— La propriété de Lamarckian.
— La causalité.
— L'épistasie.

4.4.1 La non-redondance

On définie la redondance des représentations génétique comme suit :
Soit Φ_p l'espace phénotypique et Φ_g son espace génotypique. Une représentation génétique du problème est définie par la fonction $f_g : \Phi_g \rightarrow \Phi_p$. Elle détermine quel phénotype $x^p \in \Phi_p$ est représenté par quel génotype $x^g \in \Phi_g$. Ainsi la représentation f_g est dite redondante si pour chaque phénotype $x^p \in \Phi_p$ est associé un sous-ensemble $\varphi_g \subset \Phi_g$ avec $|\varphi_g| > 1$. Autrement dit, $|\Phi_g| > |\Phi_p|$. Une redondance est dite synonyme si tous les chromosomes représentant la même solution sont similaires. Dans le cas contraire, la redondance est dite non-synonyme Rothlauf [2006]. Dans la figure 6.12.1, les quatre carrés noire représentent tous un triangle dans l'espace phénotypique (redondance symétrique). De l'autre côté, on trouve l'étoile et le cercle qui correspondent au carré (redondance non-symétrique).

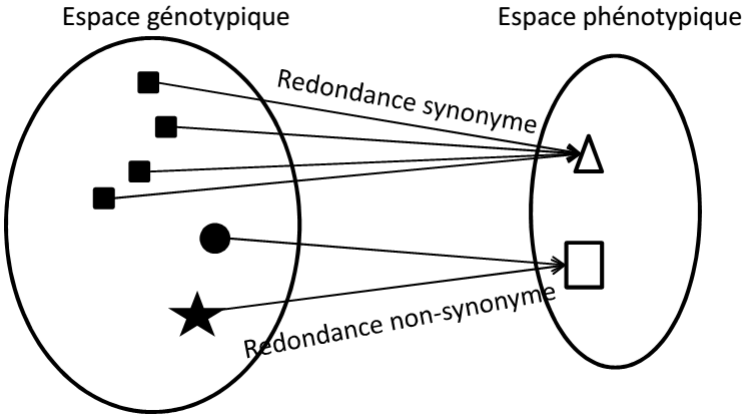


FIGURE 4.2 – *Les différents types de redondances*

4.4.2 Complétude

Cette propriété traduit l'aptitude d'un codage à représenter toutes les solutions dans l'espace de recherche. Autrement dit, si $\forall x^p \in \Phi_p, \exists x^g \in \Phi_g$ tel que $f_g : x^g \rightarrow x^p$ alors le codage est dit complet, sinon on dit que ce codage souffre de la cécité. Ceci veut dire que pour toute solution prise de l'espace phénotypique, cette dernière possède une représentation par le codage utilisé. L'impact de la complétude devient très pesant si le codage est incapable de représenter la solution optimale du problème Rothlauf [2006] Gen and Cheng [1996a].

4.4.3 Légalité

Tout individu de l'espace génotypique doit impérativement avoir une image dans l'ensemble des solutions (espace phénotypique). Autrement dit, toutes les combinaisons possibles des caractères de l'alphabet du codage correspondent à une solution. Cette propriété garantit l'utilisation de n'importe quel opérateur génétique sans risque d'avoir des individus qui produisent des solutions illégales Gen and Cheng [1996a].

4.4.4 Lamarckianisme

Cette propriété concerne la partie de l'hérédité des algorithmes génétiques. Les propriétés génétiques des parents sont transmises à leurs enfants à travers l'appariement de leurs chromosomes. Ceci aide en principe à améliorer l'adaptabilité des enfants par rapport à leur environnement. La propriété de lamarckian se base sur l'interprétation des allèles des chromosomes. En fait, si les parties du chromosome transmises du père à ses fils gardent leur sens et leur interprétation, on dira alors que le codage utilisé préserve la propriété de lamarckian. Le cas contraire décrit l'absence de cette propriété. Aussi, on dit que la représentation est demi-lamarkian si une partie seulement des gènes qui gardent son interprétation. La propriété de lamarckian contribue d'une manière efficace à l'amélioration du patrimoine génétique des descendants. Par contre, l'absence de cette propriété peut avoir des conséquences sévères sur l'adaptabilité des individus par rapport à leur environnement Kennedy [1993] Gen and Cheng [1996a].

Prenons un exemple illustratif de l'absence de la propriété de Lamarckian : Dans son article Bean [1994], l'auteur utilise un codage fractionnel pour le problème du voyageur de commerce à neuf villes. Un vecteur est utilisé pour représenter les solutions, où les indices représentent les villes et les allèles sont des valeur dans l'intervalle $[0, 1]^n$, Le vecteur est ensuite trié en ordre croissant tout en gardant la correspondance entre les valeurs des allèles et les indices comme illustrer dans la figure 4.3.

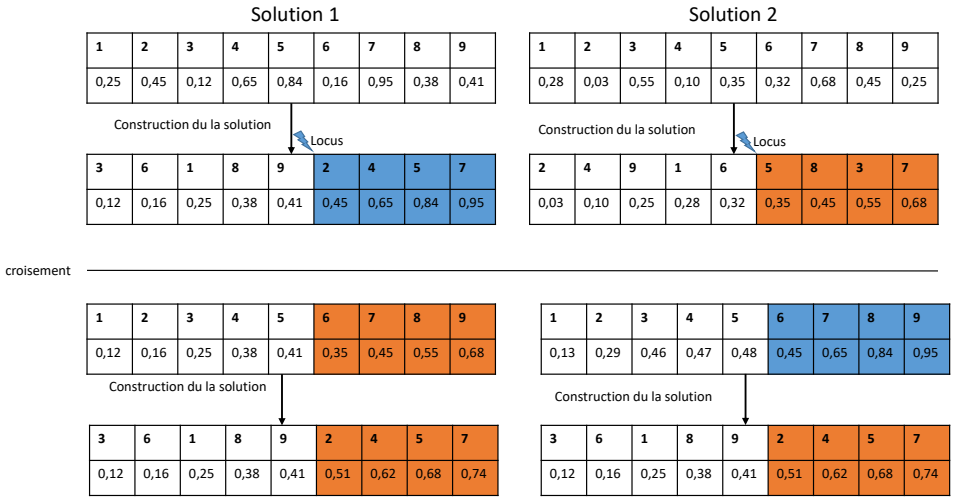


FIGURE 4.3 – *La propriété Lamarckian*

Dans la figure 4.3, nous voyons clairement que les propriétés des parents ne sont pas conservées chez leurs fils. Par conséquent, la propriété Lamarckian n'est pas préservée par cette représentation.

4.4.5 Causalité

Des travaux de recherche tels que Eshelman [1997] ont montré que les bonnes solutions se trouvent toujours à proximité. L'application de la mutation permet de réaliser des petites modification au niveau du code génétique. Si ces modifications engendre des petites modifications au niveau phénotypique on dit alors que la localité est préservée. Par conséquent le codage est dit d'une forte causalité. En revanche, si les petites altérations du code génétique génèrent des solutions très éloignées dans l'espace phénotypique, alors la localité n'est pas

préservée (falaise de hamming). Ce type de codage est qualifié de faible causalité Rosca and Ballard [1995] Rechenberg [1994].

4.4.6 Epistasie

Quoi que l'importance des représentations a été reconnu, l'effet de l'interdépendance entre les éléments de la représentation n'a encore pas reçu suffisamment d'importance. L'épistasie est un terme biologique qui qualifie l'ampleur des interactions intrachromosome des gènes. La discussion de l'épistasie en GA révèles plusieurs caractéristiques. Les interactions entre les gènes sont une question centrale dans la génétique naturel. En plus de l'expression des caractéristiques phénotypiques, certains gènes peuvent aussi MASQUER ou ACTIVER l'expression d'autre gènes. L'épistasie est utilisée pour décrire la situation où plusieurs gènes masquent ou modifient l'effet d'autres gènes. Une forte épistasie signifie que plusieurs gènes sont en étroite interaction avec d'autres gènes.
La recherche d'épistasie est une préoccupation insaisissable parce que la présence des éléments épistatiques ne peut être découverte qu'au niveau phénotypique loin de leur niveau d'interaction (génotypique). L'effet de l'épistasie réside dans la capacité de prédire la valeur d'un tout à partir de la valeur de ses parties Davidor [1990].

4.5 CONCLUSION

A travers ce chapitre nous avons étudié l'importance de la représentation des solutions pour les algorithmes génétiques et leurs impacts sur le rendement de ces derniers. Une liste de propriétés théoriques est dressée afin de faire une comparaison équitable entre les codages. Dans le chapitre suivant, nous présentons l'ensemble des codages étudié dans ce manuscrit. Cette liste est composé d'un certain nombre de codage déjà présent dans la littérature en plus de ceux que nous avons proposés.

REPRÉSENTATIONS GÉNÉTIQUES 5

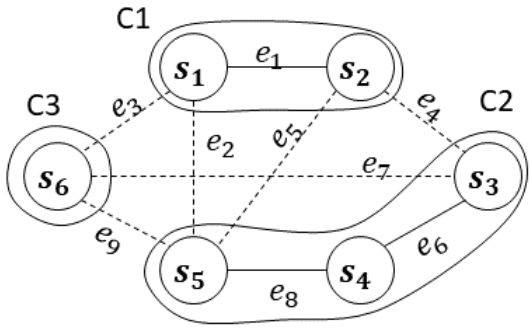
5.1 Introduction

La résolution de PPG en utilisant les AGs nécessite la définition d'un codage pour la représentation des solutions. Dans ce chapitre nous présentons un recensement assez suffisant (non exhaustif) des différentes représentations proposées dans la littérature. En plus, nous mettons l'accent sur les représentations que nous avons proposées. Pour chacun des codages, nous décrivons la procédure du codage et décodage des solutions.

5.2 Représentations existantes

5.2.1 Représentation entière à base de sommets (DVTC)

Ce codage a été proposé par Venugopal and Narendran [1992] pour la résolution du problème de composition des cellules de production. Grâce à sa simplicité, ce codage direct (ne nécessite pas une procédure de décodage) a été adopté par la majorité des travaux adoptant des approches évolutionnaires.
Le principe de ce codage repose sur l'utilisation d'un vecteur de taille $|S|$. Les indices du vecteur correspondent aux sommets, tandis que ses valeurs correspondent aux numéros des clusters de la partition. Les valeurs utilisées sont comprises dans l'intervalle $[1, |S|]$. Nous illustrons cette représentation à travers l'exemple de la figue 5.1.
Dans la figure 5.1, la partition du graphe comportant trois clusters C_1,C_2 et C_3 est représentée par un vecteur de taille $|S| = 6$. La valeur de chaque élément du tableau correspond au numéro de cluster auquel le sommet représenté par



a) Exemple d'une partition

s_1	s_2	s_3	s_4	s_5	s_6
1	2	2	3	3	3

b) La représentation DVTC correspondante

FIGURE 5.1 – *Le représentation DVTC d'une partition de trois clusters.*

l'indice de cet élément appartient. Ainsi les sommets $\{s_1, s_2\}$ appartiennent au cluster C_1, $C_2 = \{s_3, s_4, s_5\}$ et $C_3 = \{s_6\}$.

5.2.2 Représentation fractionnel à base de sommets (FVTC)

Le codage fractionnel a été utilisé par Goncalves and Resende [2002] pour la résolution du problème de composition de cellules de production. Le FVTC utilise des chromosomes de taille $(|S| + 1)$ dont les valeurs de leurs éléments sont des nombres fractionnels avec une précision donnée. Les $|S|$ premiers allèles des chromosomes sont dédiés à l'affectation des sommets du graphe aux différents clusters, tandis que la $(|S| + 1)$ ième case est réservée pour le nombre de clusters de la partition.
Le FVTC est un codage indirect nécessitant l'utilisation d'une routine de décodage pour passer au domaine phénotypique. La composition des clusters est déduite en suivant les étapes suivantes :

— La valeur du dernier allèle, une fois multiplié par le nombre maximum de clusters donne le nombre de clusters (*nbrClusters*) de la partition. Ceci en prenant la partie entière du résultat de la multiplication plus un.
— Le *nbrClusters* trouvé dans l'étape précédente est à son tour, combiné avec chacune des $|S|$ première valeurs du chromosome pour obtenir l'indice du cluster hôte pour chaque sommet, en prenant toujours la partie entière de la multiplication plus un.

Par exemple, pour m_{max} (le nombre maximum de clusters) est égal à 4, la parti-

tion de la figure 5.1 peut être codé avec la chaîne suivante :

0.32	0.23	0.47	0.35	0.65	0.81	**0.59**

Le nombre de clusters est donné par $\lceil 0.56 \times m_{max} \rceil = 3$ (où $\lceil . \rceil$ est la fonction ceil). Ensuite, le premier sommet est affecté au cluster numéro $\lceil 0.32 \times 3 \rceil = 1$ et ainsi de suite.

5.2.3 Représentation binaire à base de liaison (EA)

Le codage EA a été proposé par Boulif and Atif [2006]. Les auteurs ont utilisé un vecteur binaire de taille $|A|$ ($|A|$ nombre d'arêtes du graphe). Le graphe traité est supposé connexe, dans le cas contraire des arêtes fictives de poids nul sont rajoutées afin de le rendre ainsi. Les éléments du vecteur binaire correspondent chacun à une arête du graphe. Ces dernières sont triées dans un ordre bien défini. La valeur 0 de l'allèle correspond à une arête *intra*-cluster i.e. les deux extrémités de l'arête appartiennent au même cluster. Tandis que la valeur 1 correspond à une arête *inter*-cluster i.e. les extrémités de l'arête se trouvent dans des clusters distincts. Le vecteur ci-après montre la représentation EA correspondante à la partition de la figure 5.1.

e_9	e_8	e_7	e_6	e_5	e_4	e_3	e_2	e_1
1	0	1	0	1	1	1	1	0

Le passage au domaine phénotypique nécessite l'invocation d'une procédure de décodage. Cette dernière rassemble les composantes connexes du graphe formées par les arêtes intra-cluster. Chacune des composantes connexes représente un clusters de la partition. Dans le vecteur ci-dessus, les arêtes du graphe de la figure 5.1 représentées par e_1, e_6 forment une seule composantes connexe qui correspond au cluster C_1, tandis que l'arête représentée par e_8 regroupant les sommets s_4 et s_5 forme le deuxième cluster. Le dernier sommet forme à lui seule le dernier cluster.

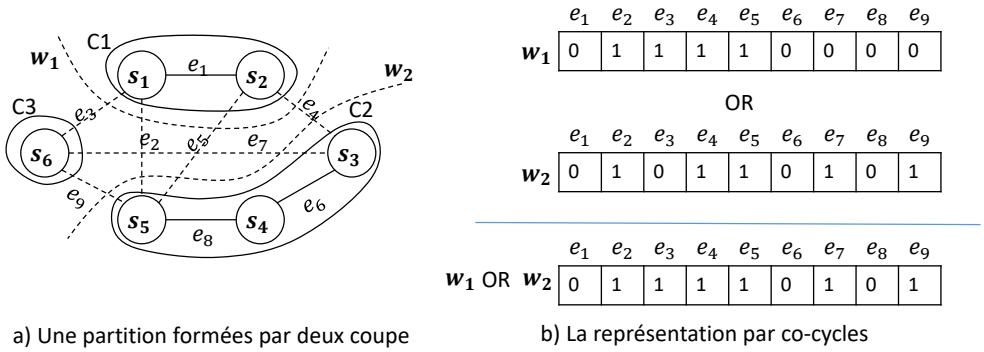


FIGURE 5.2 – *Le codage DC de la partition du graphe.*

5.2.4 Représentation binaire à base de co-cycles (DC)

La représentation DC a été proposé par Boulif [2016] pour la résolution du problème de composition de cellules de production. La représentation des solutions en utilisant ce codage est faite en suivant les démarches ci-après.
Par définition, une partition peut être représentée par la somme *OR* d'un certains nombre de coupes appliquées sur un graphe connexe. Ces coupes sont représentées par des vecteurs binaires de taille $|A|$. Les arêtes touchées par une coupe sont considérées *inter*-cluster. Le reste sont considérées *intra*-cluster. Par exemple la partition de la figure 5.1 peut être définie par la somme des coupes w_1 et w_2 illustrées dans la figure 5.2, où $w_1 = (0,1,1,1,0,0,1,0)$, $w_2 = (0,1,1,0,1,1,1,0)$. La somme $w_1 \oplus w_2 = (0,1,1,1,1,0,1,0,1)$ correspond à la représentation EA qui nécessite le recouvrement de ses composantes connexes pour déterminer les clusters (voir la figure 5.2).

Le codage DC souffre d'un handicap major. En effet, une chaine binaire ne correspond pas forcément une coupe du graphe. Pour remédier à ce problème, nous avons utilisé une technique basée sur les propriétés des coupes de la théorie des graphes. L'ensemble des coupes d'un graphe définissent un espace vectoriel. La base de cet espace est un sous-ensemble formé par les coupes correspondant au $|S|-1$ premiers sommets. Une coupe est alors calculée par la somme $\oplus$ de ses coupes de bases. Par exemple, les coupes de bases du graphe de la figure 5.2 sont données par $w(\{s_1\}) = (1,1,1,0,0,0,0,0,0)$, $w(\{s_2\}) = (1,0,0,1,1,0,0,0,0)$, $w(\{s_3\}) = (0,0,0,1,0,1,1,0,0)$, $w(\{s_4\}) =$

$(0,0,0,0,0,1,0,1,0)$, $w(\{s_5\}) = (0,1,0,0,1,0,0,1,1)$. Ainsi la coupe w_1 de la figure 5.2 est calculée par $w_1 = w(\{s_1\}) \oplus w(\{s_2\})$.

Formation des chromosomes

Le codage CBBC utilise des chaine binaire de taille $k \times (|S| - 1)$ avec $k = (\lceil \frac{|S|}{N} \rceil - 1)$ (N est la taille maximale des clusters, $\lceil . \rceil$ la fonction ceil). Pour chacune des k parties, les $(|S| - 1)$ allèles auront des valeurs binaires. La valeur 1 signifie que la coupe de base du sommet correspondant est sélectionnée, 0 sinon. La combinaison des coupes de bases choisies durant la première phase produit k chaines binaire de taille $|A|$, chacune représente une coupe du graphe. La formule permettant le calcul de k prend en considération la taille maximale des cluster. Par conséquent le nombre de cluster qui est égale à $k + 1$ permet d'avoir un partitionnement équilibré. L'étape suivante consiste à trouver la solution finale par l'addition des coupes précédentes en utilisant l'opérateur OR.
Dans ce qui suit, un exemple illustratif de l'utilisation du codage CBBC. Prenant $k = 2$, la chaine de $(2 \times (6 - 1)) = 10$ bits suivante représente la partition de la figure 5.1.

Partie 1					Partie 2				
$w(\{s_1\})$	$w(\{s_2\})$	$w(\{s_3\})$	$w(\{s_4\})$	$w(\{s_5\})$	$w(\{s_1\})$	$w(\{s_2\})$	$w(\{s_3\})$	$w(\{s_4\})$	$w(\{s_5\})$
1	1	0	0	0	0	0	1	1	1

Les traitements séparés de chacune des parties donnent les coupes w_1 et w_2. Ces dernières sont additionnées en utilisant l'opérateur OR pour produire la chaine binaire $(0,1,1,1,1,0,1,0,1)$. Les composantes connexes formées par les arêtes intra-cluster détermine la partition du graphe.

$$w_1 = w(\{s_1\}) \oplus w(\{s_2\}) = (1,1,1,0,0,0,0,0,0) \oplus (1,0,0,1,1,0,0,0,0) = (0,1,1,1,1,0,0,0,0)$$

$$w_2 = w(\{s_3\}) \oplus w(\{s_4\}) \oplus w(\{s_5\}) =$$

$$(0,0,0,1,0,1,1,0,0) \oplus (0,0,0,0,0,1,0,1,0) \oplus (0,1,0,0,1,0,0,1,1) = (0,1,0,1,1,0,1,\ 0,1)$$

$$w_1 OR\ w_2 = (0,1,1,1,1,0,0,0,0)\ \text{OR}\ (0,1,0,1,1,0,1,\ 0,1) = (0,1,1,1,1,0,1,0,1)$$

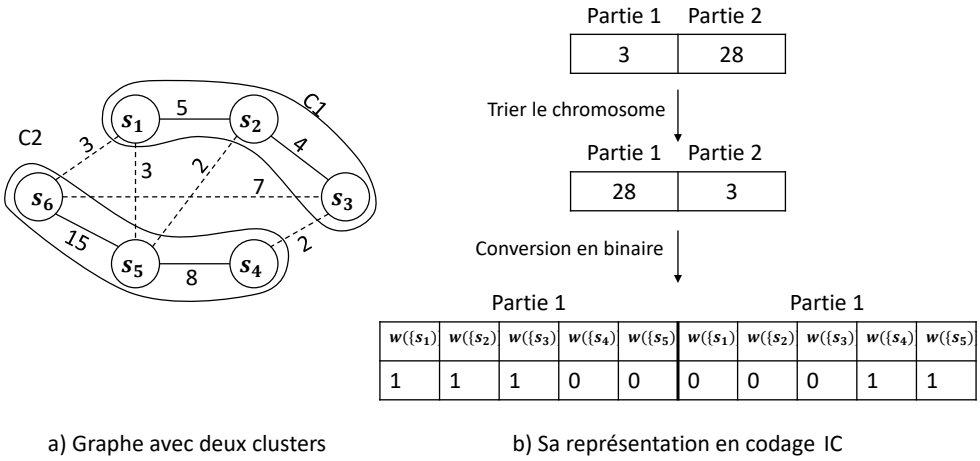


FIGURE 5.3 – *La représentation IC d'une partition de deux clauster.*

5.2.5 Représentation entière à base de co-cylces (IC)

Le codage IC représente une amélioration du codage DC. Les allèles des chromosomes sont des entiers dans l'intervalle $[0,\ 2^{(|S|-1)}-1]$. La valeur $(2^{(|S|-1)}-1)$ correspond au nombre maximum de coupes de base qu'on peut avoir pour un graphe d'ordre $|S|$. La taille des chromosomes est égale à k calculé suivant le même processus décrit dans la section 5.2.4. Après la génération du chromosome, les valeurs de ses gènes sont ensuite triées dans un ordre décroissant. Le tri des chromosomes est réalisé dans le but de réduire la redondance induite par la permutation des allèles. Le passage de l'espace génotypique à l'espace phénotypique pour l'évaluation des solutions nécessite l'application d'un processus de décodage. Le recouvrement des partitions consiste à générer des chaines de k parties de taille $(|S|-1)$. Ensuite, chacune de ces k parties recevra la conversion binaire de l'entier correspondant dans le chromosome. Le résultat de cette opération est une chaine DC. Par conséquent, le calcul de la partition suit le même processus décrit dans la section 5.2.4.
La figure 5.3 illustre le processus de génération des chromosomes et le recouvrement de la partition correspondante. En effet, le premier vecteur de la figure 5.3-b représente la solution brute (le vecteur avant le trie). Le vecteur initial est ensuite trié dans un ordre décroissant. L'étape suivante consiste à convertir les nombres entiers du vecteur trié en une séquence binaire sur une longueur de $(|S|-1)$. La combinaison XOR des co-cycles de bases de chaque partie donne

k chaines binaires de taille $|A|$. Ces chaines sont additionnées en utilisant l'opérateur logique OR. La solution finale (partition) est trouvée par le recouvrement des composantes connexes correspondantes aux arêtes intra-clusters représenté par des 0 dans le vecteur final.

5.3 Représentation modifiées

5.3.1 Représentation entière shifté à base de sommets (SVTC)

Une amélioration du codage DVTC est donné par Chaouche and Boulif [2019]. L'utilisation de l'opérateur d'appariement avec le codage DVTC crée des écarts entre les valeurs utilisées au sein des chromosomes enfants. Ceci est clairement illustré par la figure 5.4. Chaouche and Boulif [2019] ont proposé d'éliminer ces creux. Ceci nécessite l'utilisation d'une routine permettant de limiter les valeurs utilisées au nombre de clusters de chaque solution.

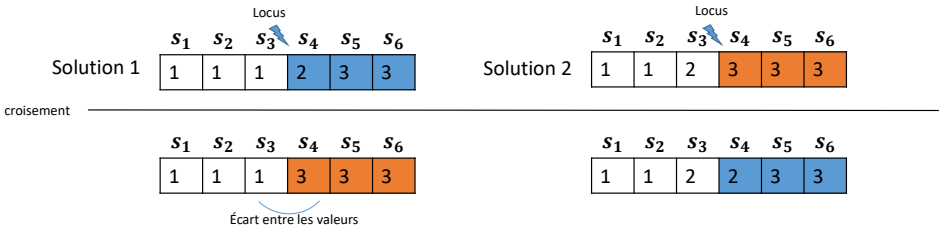


FIGURE 5.4 – *Les écarts entre les valeurs des gènes induits par l'application de l'opérateur de croisement.*

Après chaque itération du processus de l'évolution, une routine de réajustement des chromosomes est invoquée pour éliminer les écarts entre les valeurs utilisées par les allèles. Les clusters seront numérotés de 1 à $|P|$ ($|P|$ la cardinalité de la partition). Les allèles auront leurs nouvelles valeurs suivant leur ordre dans le chromosome original. Dans l'exemple de la figure 5.5, le premier sommet gardera sa valeur, ensuite, les prochains allèles dont la valeur suit directement celle qu'on vient d'attribuer auront comme valeur le successeur direct de la valeur précédente à savoir 2, et ainsi de suite.

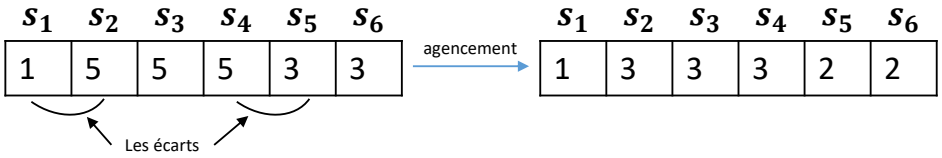
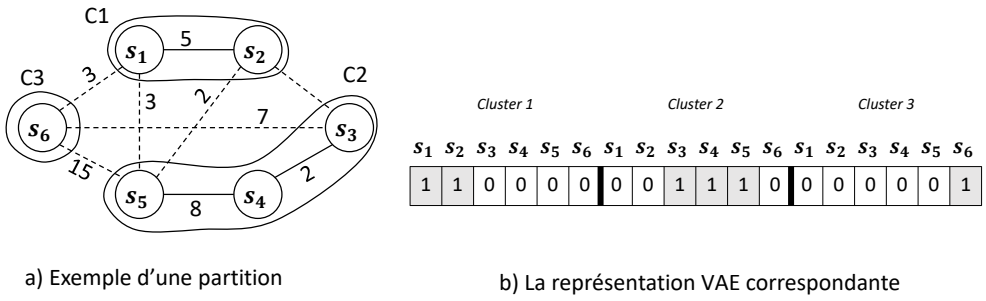


FIGURE 5.5 – *Agencement des gènes pour le codage SVTC.*

5.3.2 Représentation binaire à base de sommets/clusters (VAE)

Dans sa version originale proposée par Wicks and Reasor [1999] ce codage autorisait l'affectation simultanée d'un sommet à plus d'un seul cluster. Une nouvelle version est proposée par Chaouche and Boulif [2019]. Dans cette version, le sommet est affecté à un et un seul cluster à la fois. Le codage utilise des chromosomes de $|S|$ parties de taille $|S|$. Les allèles du chromosome sont soit 1 soit 0. La valeur 1 signifié la présence du sommet dans le cluster, 0 sinon. En plus, un tri lexicographique est utilisé afin de réduire l'ampleur de la redondance. A travers l'exemple de la figure 5.6 nous allons essayer de mieux éclaircir le principe de fonctionnement de ce codage. Le vecteur de la figure 5.6.b est composé de trois partie de taille $|S| = 6$. La partie dénommée *cluster*1 est réservée premier cluster. Elle contient deux allèles à 1 correspondant aux sommets s_1 et s_2, ce qui veut dire que ces deux sommets sont assignés au premier cluster, et ainsi de suite. Ainsi la partition sera formée par les clusters $C1 = \{s_1, s_2\}$, $C2 = \{s_3, s_4, s_5\}$ et $C3 = \{s_6\}$.



a) Exemple d'une partition

b) La représentation VAE correspondante

FIGURE 5.6 – *La représentation VAE d'une partition de 3 clusters.*

5.4 Représentations proposées

5.4.1 Représentation entière à base des écarts de co-cycles (CGE)

Le principe du codage par écarts consiste à générer des chaines d'entiers de taille k avec $k = (\lceil \frac{|S|}{N} \rceil - 1)$, ($\lceil . \rceil$: la fonction ceil, et N : la taille maximale des clusters). Les allèles des chromosomes auront leurs valeurs dans l'intervalle $[0,\ 2^{(|S|-1)} - 1]$. Une nouvelle chaine est ensuite calculée, avec comme valeur de chacun de ses éléments, la somme de l'allèle en cours de la première chaine avec tous ceux qui le précèdent. Le sens d'affectation des valeurs est de gauche à droite, le premier allèle reste inchangé. À partir de l'allèle ou la somme dépasse la borne supérieure, la valeur 0 est affectée. Une troisième chaine construite à partir de celle des écarts cumulés par leur conversion binaire, chacun sur une chaine de $(|S| - 1)$ bits. Le résultat de cette opération est une chaine binaire de taille $(k \times (|S| - 1))$ avec 1 correspond au choix de la coupe de base correspondante, 0 sinon. La procédure du calcul de la partition à partir de cette étape est similaire à celui du codage IC (voir §5.2.5).

Dans l'exemple de la figure 5.7, la partition à quatre clusters à gauche est représentée par le codage CGE illustré dans la figure 5.7.b. La première chaine contient les écarts générés aléatoirement dans l'intervalle $[0,\ 31]$. Les éléments de la deuxième chaine sont calculés comme suit :

— Partie 1, reçoit la valeur du premier écart de la permière chaine 24.

— Partie 2, recoit la somme du premier et second écart $24 + 6 = 30$.

— Partie 3, recoit la somme de tous les éléments précédent $24 + 6 + 1 = 31$.

Le passage à la représentation binaire est illustré par la troisième chaine à 15 bits. Les coupes de base choisis dans chacune des parties sont cumulés en utilisant l'opérateur $\oplus$. Cette étape produit trois chaines binaire de taille $|A|$. Ces dernières sont additionnées par l'opérateur OR pour en produire une seule. Les bits à 1 de cette dernière chaine correspondent aux arêtes inter-cluster tandis que les 0 représentent les arêtes intra-cluster. Finalement, les clusters sont formés par les composantes connexes formées par les arêtres intra-cluster.

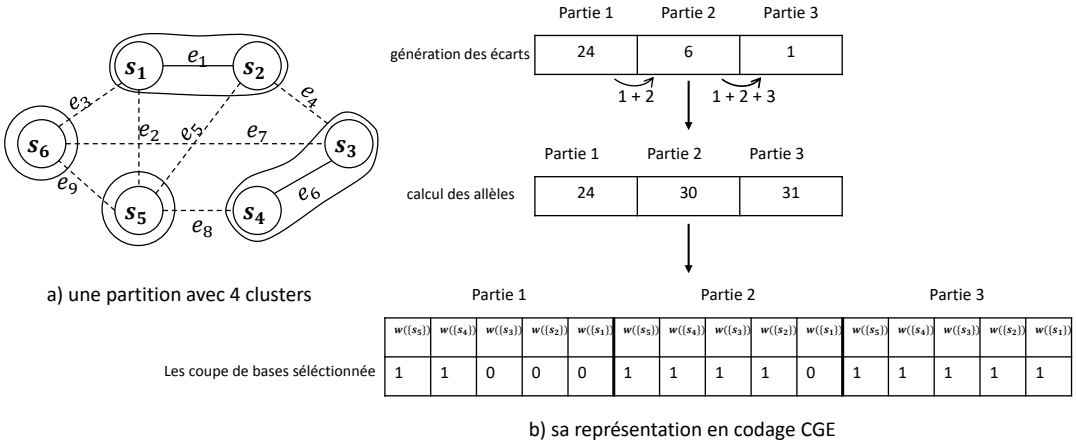


FIGURE 5.7 – *La représentation CGE d'une partition à trois clusters.*

5.4.2 Codages P-médian

Le problème de P-médian a été évoqué pour la première fois par Fermat au 17^{ieme} siècle. Son principe est décrit comme suit : pour un triangle donné, minimiser la somme des distances maximales entre un point de ce triangle et ses têtes. De nos jours, ce problème a connu plusieurs variantes dont l'objectif est quasiment le même à savoir la minimisation de la somme des distances maximales séparant un ensemble de points de leurs médians. À la fin des années 1960 Hakimi Hakimi [1964], Hakimi [1965] a appliqué le même problème dans les réseaux et les graphes.

5.4.3 Formalisation du problème de P-médian

Pour un $G = (S, A)$, l'objectif est de trouver $(S_p \subset S)$ de sommets appelés médians maximisant la somme des poids des plus courts chemins reliant les sommets non médians $\{S \setminus S_p\}$ à leurs plus proches de S_p. L'application du problème de P-médian au PPG nécessite une légère adaptation. De ce fait, les sommets non-médians sont affectés aux clusters formés par chacun des sommets de S_p. Cette affectation est basée sur la maximisation des liaisons entre $\{S \setminus S_p\}$ et S_p. Lorena and Furtado [2001] ont utilisé ce modèle pour proposer un algorithme génétique constructif pour le problème de clustering. Les auteurs

utilisaient une chaine de longueur $|S|$ où les allèles sont des valeurs binaires (1 : correspond à un sommet médian, 0 sinon). En plus, les auteurs définissaient une distance entre tous les paires de sommets (graphe complet) et supposent que le nombre de cluster est connu à priori. Pour notre approche non supervisée, le nombre de clusters n'est pas connu à priori, en plus le graphe n'est pas nécessairement complet. Par conséquent, un sommet non-médians n'est pas toujours lié aux sommets médians. Ceci pose un problème pour la procédure d'affectation. Les deux variantes de ce codages sont détaillés dans les deux sous-sections qui suivent.

5.4.4 Représentation P-médian à base de liaison (PMEB)

La procédure d'affectation basée sur les arêtes du graphe est présentée dans l'algorithme 8. La notation suivante est utilisée :

— $f_{ij} = \omega(a_{ij})$ si $a_{ij} \in A$, 0 sinon ;
— $\hat{f}_{ik} = \sum_{s_j \in C_k} f_{ij}$;
— M : L'ensemble des médians ;
— NM : L'ensemble des non médians.

Algorithme 8 : Procédure d'affectation du codage PMEB

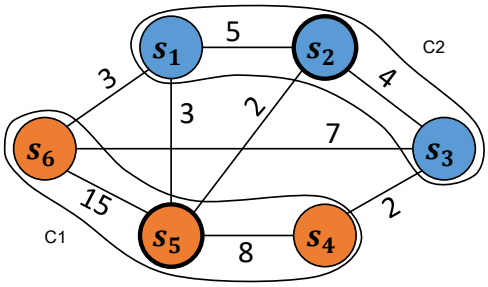
Etape 1 : Pour chaque sommet $s_i \in M, i = \{1, \ldots, M\}$ créer un cluster $C_i = \{s_i\}$.
Etape 2 : Affecter chaque sommet non médian s_i ayant des arêtes avec un ou plusieurs sommets médians, on choisissant celui dont l'affectation maximise f_{ij}.
Etape 3 : Affecter chaque sommet non médian non affectés durant l'étape 2 à un cluster C_k dont l'affectation maximise $\hat{f}_{ik}$

La première étape consiste à créer un cluster pour chaque sommet médian. Dans un second lieu, nous procédons à l'affectation des sommets non-médians possédant des liens directs avec un ou plusieurs médians. Le cluster C_j est choisi pour abriter le sommet s_i si f_{ij} est maximale. L'étape 3 traite l'affectation des sommets non-médians ne possédant pas de liens direct avec les médians. Cette affectation se fait par rapport aux arêtes reliant les non-médians avec tous les sommets des clusters. Ainsi le sommet s_i est affecté au cluster C_k si $\hat{f}_{ik}$ est maximale.

Remarque :

— Si les distances séparant le sommet non médian aux médians sont égaux, ce dernier est naturellement affecté au cluster dont l'indice du médian est inférieur respectant les contraintes pré-imposées telles que la taille maximale des clusters.

— Le traitement des sommets se fait en ordre par rapport à leur indice (ordre croissant).



a) Graphe avec deux partition

s_1	s_2	s_3	s_4	s_5	s_6
0	1	0	0	1	0

b) sa représentation PMEB

FIGURE 5.8 – *La représentation PMEB d'une partition de deux clusters.*

La figure 5.8 montre une partition de deux clusters et sa représentation PMEB. L'ensemble des sommets médians et celle des non médians sont données respectivement par $M = \{s_2, s_5\}$, $NM = \{s_1, s_3, s_4, s_6\}$. L'application de l'algorithme d'affectation donne la partition $P = \{C_1 = \{s_1, s_2, s_3\}, C_2 = \{s_4, s_5, s_6\}\}$. En effet, la première étape permet de créer le noyau de la partition $C_1 = \{s_2\}$,$C_2 = \{s_5\}$ où les clusters sont formés uniquement par les sommets médians. Pendant l'étape 2, les sommets s_1,s_4 sont affectés à C_1 et C_2 maximisant f_{ij}. La dernière étape affecte les sommets s_3,s_6 de sorte à maximiser la fonction $\hat{f}_{ik}$.

5.4.5 Représentation P-médian à base de contraintes (PMCA)

Le PMCA est la deuxième variante des codages basés sur le problème de P-médians. Cette représentation est basée sur le respect des contraintes imposées par le problème pour réaliser l'affectation des sommets non médians aux sommets médians. L'algorithme 9 présente les différentes étapes à suivre pour le recouvrement de la solution à partir d'une représentation PMCA.

Algorithme 9 : P-médian Affectation basée sur les contraintes

1: **Etape 1 :** Pour chaque sommet $s_i \in M, i = \{1, \ldots, M\}$ créer un cluster $C_i = \{s_i\}$.
2: **Etape 2 :** Procédons à cette affectation si c'est possible : allouer chaque sommet non médian s_i au cluster C_k avec qui il possède un arête maximisant f_ij à condition que cette affectation ne viole aucune des contraintes du problème. Sinon allez à l'étape 5.
3: **Etape 3 :** Affecter chaque sommet non médian non affectés durant l'étape 2 à un cluster C_k dont l'affectation maximise $\hat{f}_{ik}$.
4: **Etape 4 :** Affecter chaque sommet non médian restant s_i au premier cluster C_k (s'il existe) sans violer aucune des contraintes.
5: **Etape 5 :** Si tous les sommets non médians sont affectés, l'algorithme se termine.
6: **Etape 6 :** Affecter tous les sommets restants au dernier cluster (ce qui produit une solution irréalisable).

Prenons l'exemple de la figure 5.9 avec les contraintes suivantes :

— les sommet s_1 et s_2 ne doivent pas cohabiter le même cluster.
— la taille maximale des clusters est égale à 3.

L'étape 1 consiste à créer un cluster pour chaque sommet médian ce qui donne $P = \{C_1 = \{s_1\}, C_2 = \{s_5\}\}$. L'étape 2 affecte le sommet s_2 à C_1 car $\omega(s_2, s_1) > \omega(s_2, s_5)$ par conséquent $P = \{C_1 = \{s_1, s_2\}, C_2 = \{s_5\}\}$. Le sommet s_4 est ensuite affecté à $C2$ parce qu'il est lié uniquement avec le sommet médian s_5. Le sommet s_6 est lié avec les deux sommets médians avec $\omega(s_1, s_6) < \omega(s_5, s_6)$ or celui-ci est affecté à $C1$ pour assurer la contrainte de cohabitation. À la fin, le sommet s_3 est affecté au cluster $C2$ car son affectation au cluster $C1$ enfreint la contrainte de la taille maximale des clusters.

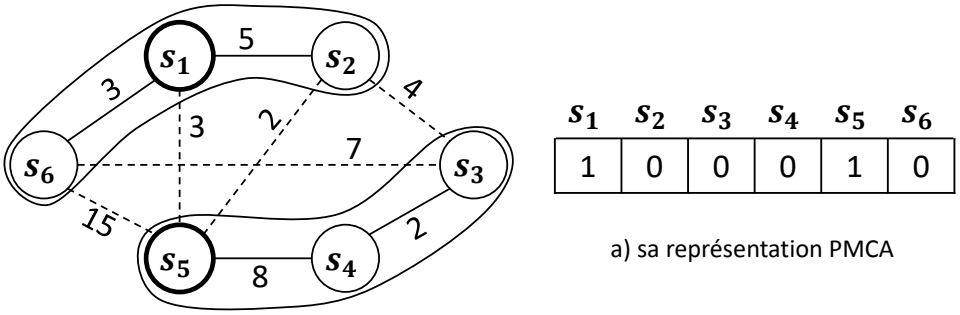


FIGURE 5.9 – *La représentation PMCA d'une partition de deux clusters.*

5.5 Classification des codages

Par rapport aux caractéristiques des codages vus dans ce chapitre et leurs principes de représentation des solutions, nous avons élaboré la classification illustrée dans le tableau 5.1. En effet, trois familles de codages existent. La première est celle basée sur les sommets du graphe telles que DVTC, SVTC, FVTC et VAE. La deuxième famille comporte les codages basés sur les arêtes où nous trouvons les codages EA, DC, IC et CGE ; la dernière famille combine les deux approches précédentes que nous appelons codages hybrides, c'est-à-dire qu'elle est basée à la fois sur les sommets et les arêtes pour représenter les solutions, elle comporte les codage PMEB, PMCA.

codages à base de sommets	codages à base des liaisons	codages hybrides
DVTC	EA	PMEB
SVTC	DC	PMCA
FVTC	IC	
VAE	CGE	

TABLE 5.1 – *Classification des codages.*

5.6 propriétés génétiques des codages

Le tableau 5.2 donne un récapitulatif des propriétés génétiques des différentes représentations présentées dans ce manuscrit.

Propriétés génétiques des codages						
Codage	**Non redondance**	**Complétude**	**Légalité**	**Lamarckianisme**	**Causalité**	**Epistasie**
DVTC	✗	✓	✓	✓	✗	✗
SVTC	✗	✓	✓	✓	✗	✗
FVTC	✗	✓	✓	✗	✗	✓
VAE	✓	✓	✗	✓	✗	✗
EA	✗	✗	✓	✓	✗	✓
DC	✗	✗	✓	✓	✗	✓
IC	✗	✗	✓	✓	✗	✓
CGE	✗	✗	✓	✓	✗	✓
PMEB	✗	✗	✓	✓	✗	✓
PMCA	✗	✗	✓	✓	✗	✓

TABLE 5.2 – *Propriétés génétiques des codages*

5.7 CONCLUSION

Ce chapitre a fait l'objet d'une étude détaillée des principaux codages présents dans la littérature. Nous avons aussi amélioré certains codages existants telles que le codage SVTC qui représente une amélioration du codage DVTC, le codage VAE représente quant à lui une amélioration du codage présenté par Wicks et Reasor. Pour la famille des codages basée sur les arêtes, les codages IC et CGE sont deux nouveaux codages que nous avons proposés et qui sont basés sur le même principe que celui de DC. En plus, notre étude nous a permis de proposer deux variantes de codages basés sur le problème de P-médian notamment PMEB et PMCA. À la fin du chapitre, nous avons proposé une classification de ces codages par rapport à leurs principes. Dans le chapitre suivant, nous exposons les différentes propriétés génétiques des codages vus dans ce chapitre.

ÉTUDE COMPARATIVE 6

6.1 Introduction

Après avoir réalisé l'étude théorique qui nous a permis de bien comprendre le comportement des différentes représentations vues dans ce manuscrit, ce chapitre sera consacré à une étude empirique afin d'assurer une comparaison équitable entre les codages et voir l'aptitude de chacun à résoudre le PPG avec efficacité. À cet effet, nous avons opté pour une stratégie de deux étapes appliquée sur un ensemble de graphes pris de la littérature de différentes caractéristiques telles que, l'ordre du graphe donné par $|S|$, la taille du graphe $|A|$, la densité du graphe ...). La première phase de notre stratégie consiste à utiliser le même paramétrage de l'AG pour tous les codages. Les résultats de cette première phase seront comparés à ceux de la deuxième phase qui consiste quant à elle à trouver le meilleur paramétrage de l'AG pour chacun des codages, ensuite l'appliquer sur l'ensemble des graphes de notre jeu de données. Les résultats produites par l'AG seront ensuite comparés à ceux d'une méthode exacte à savoir la SCIP dans le but d'avoir un point de repère pour les résultats donnés par nos différents codages. Une comparaison plus détaillée est faite à la fin de ce chapitre en utilisant le test de Friedman.

6.2 Paramétrage de l'algorithme génétique

6.2.1 Taille de la population

Une population initiale est générée d'une façon aléatoire. La taille de la population est de 100 individus et ne change pas au cours des itérations. Le nombre

d'individus de la population est choisi suite à une série d'expérience sur l'ensemble des graphes de notre jeu de données.

6.2.2 Reprodution

Afin de préserver les meilleurs individus et d'assurer la convergence de l'AG vers un optimum, nous sauvegardons $r_1\%$ des meilleures solutions de la population courante avant de passer à l'étape de sélection naturelle, pour les réinsérer dans la nouvelle génération.

6.2.3 Sélection

Parmi les opérateurs de sélections existant dans la littérature, nous avons choisi celui proposé par David Goldberg Goldberg [1989] à savoir la roue de loterie. En effet, à chaque itération $100 - (r_1\% + r_2\%)$ individus sont sélectionnés pour l'appariement afin de produire les individus de la nouvelle génération. Les parents sélectionnés seront remplacés par leurs enfants dans la nouvelle population.

6.2.4 Croisement

Un croisement simple avec un seul point de coupure est alors appliqué sur les $100 - (r_1\% + r_2\%)$ individus choisis durant la phase de sélection. Les deux parents à apparier sont choisis d'une manière aléatoire ainsi que le point de coupure (locus). À l'issue de cette opération, deux enfants sont alors créés où chacun possède des gènes de ses deux parents.

6.2.5 Mutation

Après avoir appliqué les deux opérateurs génétiques précédents, nous procédons ensuite à l'application de l'opérateur de mutation avec un taux égal à $r_3\%$. Cet opérateur est donc appliqué sur les $(100 - r_1\%)$ individus. Par conséquent, un gène est choisi aléatoirement pour lui modifier ensuite sa valeur par une nouvelle valeur aléatoire, ou juste inversion de la valeur de bit pour les codages utilisant un alphabet binaire.

6.2.6 Critère d'arrêt

Le critère d'arrêt décide du moment d'arrêter les itérations de l'AG pour récupérer ensuite la meilleure solution obtenue. Il peut être fixé par le nombre de générations, un seuil de la fonction d'adaptation, etc. Le critère d'arrêt que nous avons adopté consiste à achever le processus itératif après un certain nombre de générations sans amélioration de la fonction d'adaptation. Après plusieurs nombres d'expériences, nous avons fixé ce nombre à 30.

6.2.7 Nombre d'exécutions

Etant une méthode métaheuristique, le comportement de l'AG peut être différent d'une exécution à une autre. En plus, la qualité de la solution finale ainsi que la durée écoulée pour en arriver peut aussi être différent. À cet effet, l'utilisation de plusieurs exécutions au lieu d'une seule va nous permettre de mieux comprendre le comportement de chacun de nos codages vis-à-vis chaque instance de jeu de données utilisé.

6.2.8 Mise à l'échelle des fitnesses

Nous avons mis en place deux techniques permettant un bon compromis entre la diversification de la population et l'exploitation des meilleurs individus. La première consiste à regénérer aléatoirement $r_2\%$ nouveaux individus à chaque itération. L'insertion de ces individus dans la population est effectuée en faisant une compétition un par un avec ceux des $(100 - r_1\%)$ individus tirés aléatoirement. Bien que les individus générés aléatoirement peuvent assurer une certaine diversification au sein de la population, ceci reste tout de même insuffisant pour éviter la convergence prématurée de l'AG. Pour palier ce problème nous avons opté pour la sigma troncature de Goldberg [1989]. L'intérêt de cette technique est de donner la chance à tous le individus pour se reproduire.

6.3 Description de jeu de données de graphes

Les tests menés dans ce travail visent à mesurer les performances des différentes représentations génétiques vues précédemment. Pour arriver à cette fin,

nous avons utilisé un jeu de données composé de 45 graphes tirés de la littérature. Les graphes sont choisis avec des caractéristiques nuancées. L'ordre des graphes varie entre 10 et 1000 tandis que le nombre d'arêtes appartient à l'intervalle $[10, 307350]$, ceci nous a permet d'avoir des graphes avec différentes densités. L'ensemble des graphes est divisé par rapport à leurs ordres en trois classes différentes, petite $[10, 100]$, moyenne $[121, 256]$ et grande $[300, 1000]$. De chaque classe, nous avons pris une instance représentative pour expliquer les résultats détaillés de chaque représentation. La table 6.1 résume tout le jeu de données avec les contraintes imposées. Chaque couple de sommet dans les colonnes cohabitation, non cohabitation représentent respectivement les couples de sommets à cohabiter et à non cohabiter. En plus, les contraintes sur la taille maximale des clusters sont données par la colonne 8. Les instances représentatives des trois classes sont mises en gras. Les trois classes sont séparées par des lignes horizontales.

Graph Number	Reference	Tag	$\|V\|$	$\|E\|$	Density	Edge weight sum	Constraints		
							Max cluster size	Cohabitation	Non cohabitation
1	David A. et al. [2016]	DEBR4	8	10	0.36	10	4	(2,5)(4,7)	-
2	Merchichi and Boulif [2015]	C13	10	15	0.33	15	5	-	(3,4)
3	Merchichi and Boulif [2015]	C8	12	22	0.33	102	4	(4,7)(8,9)	(2,11)
4	David A. et al. [2016]	cage_3_5	14	21	0.23	21	5	(2,3)	-
5	Merchichi and Boulif [2015]	C10	15	55	0.52	184	5	(6,15)	(8,11) (5,15)
6	David A. et al. [2016]	BFLY	24	36	0.13	36	8	(6,7)	(15,17)
7	Walshaw [2016]	Myciel4	30	45	0.10	45	6	(6,8)	(9,11)
8	Walshaw [2016]	Queen7_7	36	290	0.46	290	12	(11,16)	(4,7) (6,10)
9	**Walshaw [2016]**	**Queen8_8**	**49**	**476**	**0.40**	**476**	**16**	**(6,7)(9,12)**	**(3,4)**
10	Walshaw [2016]	Queen6_6	74	301	0.11	301	12	(5,7)(6,10)	(24,32)
11	Walshaw [2016]	Queen8_12	96	1368	0.30	1368	16	(36,43)	(2,4) (8,10)
12	Walshaw [2016]	Queen10_10	100	1470	0.30	1470	20	(53,65)(59,79)	(17,69) (86,93)
13	Walshaw [2016]	Queen11_11	121	1980	0.27	1980	16	(14,16)(46,47)	(8,90) (9,100) (11,121)
14	David A. et al. [2016]	DSJC125.5	125	3891	0.50	3891	25	(116,117)(11,12)	(10,110)
15	David A. et al. [2016]	SE7	128	190	0.02	190	25	(16,18)(25,26)(43,46)(57,59)	(19,44) (26,41) (16,97)
16	Walshaw [2016]	Queen12_12	144	2596	0.25	2596	30	(17,24)(45,57)(80,86)	(14,27)
17	David A. et al. [2016]	Bcsstk05	153	1135	0.10	1135	50	(19,23)(26,33)(96,104)	(16,90) (25,36) (43,103)
18	**Walshaw [2016]**	**Queen13_13**	**169**	**3328**	**0.23**	**3328**	**30**	**(5,7)(6,10)(53,65)(59,79)**	**(11,22) (16,20) (17,26) (19,33)**
19	Walshaw [2016]	Queen14_14	196	4147	0.22	4166	35	(122,126)(133,146)(157,161)	(5,7) (6,10) (53,65) (59,79)
20	Walshaw [2016]	Queen15_15	225	5180	0.21	5180	40	(26,63)(144,157)(175,183)	(122,126) (133,146) (157,161)
21	David A. et al. [2016]	G250.04	250	613	0.02	613	30	(3,4)(114,154)(151,211)(241,244)	(144,183) (214,246)
22	David A. et al. [2016]	DSJC250.1	250	3218	0.10	3218	50	(25,26)	(3,103) (5,105) (6,106)
23	David A. et al. [2016]	DSJC250.5	250	15668	0.50	15668	50	(125,126)(145,146)	(12,112) (16,116)
24	David A. et al. [2016]	DSJC250.9	250	27897	0.90	27897	50	(97,98)(208,209)(22,23)	(3,103)
25	David A. et al. [2016]	R250.5.col	250	14849	0.48	14849	50	(44,45)(182,183)	(17,117) (19,119)
26	Walshaw [2016]	Queen16_16	256	381	0.01	381	30	(24,216)	(114,154) (151,211) (241,246)
27	David A. et al. [2016]	SE8	256	6320	0.19	6320	40	(76,83)(88,93)(114,121)(176,183)(231,234)	(144,157) (175,201) (3,4)
28	David A. et al. [2016]	flat300_26_0.col	300	21633	0.48	21633	50	(66,67)(199,200)(121,122)(177,178)	(13,113) (8,108)
29	David A. et al. [2016]	flat300_28_0.col	300	21695	0.48	21695	50	(292,293)(28,29)(299,300)(191,192)	(14,114) (14,114) (5,105)
30	David A. et al. [2016]	school1_nsh.col	352	14612	0.24	14612	50	(341,342)(174,175)	(8,108)
31	David A. et al. [2016]	school1.col	385	19095	0.26	19095	50	(23,24)	(18,118) (20,120)
32	**David A. et al. [2016]**	**Bcsstk06**	**420**	**3720**	**0.04**	**3720**	**60**	**(12,26)(37,46)(352,406)**	**(143,214) (181,294)**
33	David A. et al. [2016]	FFT6	448	768	0.01	768	60	(131,133)(343,344)	(57,58) (26,127) (13,414)
34	David A. et al. [2016]	le450_5d.col	450	9757	0.10	9757	50	(313,314)(141,142)(330,331)(194,195)	(11,111) (3,103) (9,109) (7,107)
35	David A. et al. [2016]	le450_15a.col	450	8168	0.08	8168	50	(412,413)	(9,109)
36	David A. et al. [2016]	le450_15b.col	450	8169	0.08	8169	50	(9,10)(176,177)(444,445)	(11,111) (7,107) (2,102)
37	David A. et al. [2016]	le450_15c.col	450	16680	0.17	16680	50	(406,407)(323,324)	(5,105) (15,115) (17,117) (1,101)
38	David A. et al. [2016]	le450_25c.col	450	17343	0.17	17343	50	(193,194)(13,14)(267,268)	(20,120)
39	David A. et al. [2016]	le450_25d.col	450	17425	0.17	17425	50	(99,100)(150,151)(436,437)	(3,103) (10,110) (2,102)
40	David A. et al. [2016]	G500.04	500	2355	0.02	2355	80	(15,16)(46,48)(79,81)(151,153)	(214,414) (324,424)
41	David A. et al. [2016]	DSJC500.1	500	12458	0.10	12458	50	(178,179)(149,150)	(5,105) (11,111) (18,118)
42	David A. et al. [2016]	DSJR500.1c	500	121275	0.97	121275	50	(237,238)(431,432)(32,33)	(5,105) (13,113)
43	David A. et al. [2016]	SE9	512	766	0.01	766	80	(14,16)(57,58)(127,129)(322,324)	(126,301)
44	David A. et al. [2016]	latin_square_10.col	900	307350	0.76	307350	100	(436,437)(707,708)(291,292)	(13,113) (7,107)
45	David A. et al. [2016]	G1000.02	1000	10107	0.02	10107	100	(57,58)(136,137)(451,452)(531,533)	(622,736) (681,721) (757,847) (824,931)

TABLE 6.1 – *Les instances de graphes.*

6.4 DESCRIPTION DES MÉTRIQUES UTILISÉES POUR LA COMPARAISON

La définition des métriques représente le point angulaire pour la mesure des performances de chaque codage. Cette étape est très sensible car si les métriques utilisées ne reflètent pas les vraies performances des codages, les résultats finaux de la comparaison seront erronés et par conséquent l'étude perd de sa fiabilité. Notre stratégie repose sur la comparaison de trois types différents de métrique mais étroitement liés. Le premier type contient deux métriques à savoir la moyenne des temps d'exécution ART (Average Run Time) correspondant

aux exécutions de la meilleure solution. Bien que la ART est une métrique très importante utilisée pratiquement dans tous les travaux de recherche, elle reste insuffisante pour montrer le vrai effort fourni par la méthode pour atteindre son meilleur score. En effet, la ART est une moyenne des meilleurs temps d'exécution y compris les valeurs correspondantes à des solutions non réalisables, ce qui signifie que les résultats seront biaisés. Les différences entre les codages en termes de ART sont due principalement à la complexité algorithmique de la routine permettant la transition entre le domaine génotypique et le domaine phénotypique. A cet effet, nous avons utilisé une deuxième métrique nommée AES (Average Evaluation to Solution) permettant le calcul du nombre de solutions visitées dans l'espace de recherche avant d'arriver à la meilleure solution. Cette métrique reflète mieux les efforts déployés par les différents codages pour arriver à leurs meilleures solutions. Cependant, elle souffre d'un problème important qui empêche de faire une bonne comparaison des performances des codages. En effet, pour les exécutions fournissant des solutions non réalisables, la valeur de AES peut fausser le calcul surtout pour les exécutions où l'AG eut une convergence prématurée vers une solution non réalisable ce qui correspond à une faible valeur de AES.

Afin de combler les insuffisances présentées par le premier type de métriques, nous avons utilisé un deuxième type qui va nous permettre de mesurer la qualité de la solution produite par les différents codages. Une fois de plus, nous avons utilisé deux types de métriques. La première est la moyenne des meilleures fitness (MBF : Mean Best Fitness) à travers les trente exécutions. Ceci va mettre en avant les efforts fournis par chaque codage et mettre le point sur les faux positifs des métriques ART et AES. Par exemple, pour un codage qui présente une valeur faible de AES avec une solution irréalisable ceci représente un faux positif pour la métrique AES. Pour les solutions non réalisables, la valeur de sa fonction objectif est remplacé par la somme totale des poids des arêtes de graphes. La deuxième métrique est la variance des meilleurs fitness calculée sur les trente exécutions (SDBF : Standard Deviation Best Fitness). Cette métrique nous permet de mesurer la stabilité de chaque codage et ses aptitudes à produire des solutions réalisables.

Le dernier type de métrique est la meilleure solution donnée par chaque codage à travers les trente exécutions (BF : Best Fitness). Bien que, la métrique MBF me-

sure la qualité des solutions, le fait d'avoir des solutions non réalisables dont la fitness sera remplacé par la somme totale des poids des arêtes, ceci peut ne pas refléter clairement l'aptitude du codage à atteindre des solutions de bonne qualité voire optimales dans certains cas Eiben and Smith [2015].

6.5 Gestion des contraintes

Afin d'éviter que les solutions irréalisables ne prennent l'assaut sur les réalisables lors de l'application de l'opérateur de sélection, nous avons appliqué le processus de pénalisation décrit dans Boulif and Atif [2006]. En effet, la fitness de chaque solution est calculée en fonction de son coût de coupe et le nombre de contraintes qu'elle viole en utilisant le processus suivant : le problème de minimisation du coût de coupe devient un problème de maximisation des poids des arêtes intra-clusters. Ensuite, la valeur de la fitness B de la solution est calculée en utilisant la formule suivante : $B = W - \sum_{e \in E_P} \omega(e)$ où W représente la somme totale des poids du graphe. Soit C le nombre de contraintes imposées par le problème PPG et c le nombre de contraintes violées par une solution donnée. La nouvelle valeur de la fitness de cette solution est donnée par $Z = B + (C - c) \times W$. Par conséquent, plus le nombre de contraintes violées augmente plus la valeur de Z diminue ce qui donne l'avantage aux solutions réalisables.

6.6 Analyse des résultats par échantillon

Afin de réaliser une bonne comparaison entre les représentations génétiques, nous avons réalisé une analyse incrémentale. En effet, les résultats de la première métrique à savoir ART sont traités en premier ensuite nous passons à la métrique AES, MBF, SDBF et à la fin la meilleure fitness BF. Cette première comparaison est basée sur le résultat d'utilisation des mêmes paramètres de l'AG vus dans le §6.2.

6.6.1 Analyse des résultats de la métrique ART

Les résultats fournis par les échantillons pris de chaque classe de graphe pour la métrique ART sont exposés dans la figure 6.1. Les valeurs obtenues montrent la supériorité des codages DVTC, SVTC et FVTC. De l'autre côté, les codages basés sur les arêtes, en particulier IC et CGE se trouvent les derniers pour toutes les instances de graphes. Les autres codages présentent des valeurs modérées de ART pour toutes les instances à l'exception de PMEB qui commence à donner des scores significatifs avec l'augmentation de l'ordre des instances $|S|$ à l'opposé de PMCA qui garde toujours des valeurs modérées. Cette différence en terme de temps d'exécution est due principalement à la complexité des structures de données représentant le codage et surtout à la complexité algorithmique des routines permettant le passage entre les domaines génotypique et phénotypique.

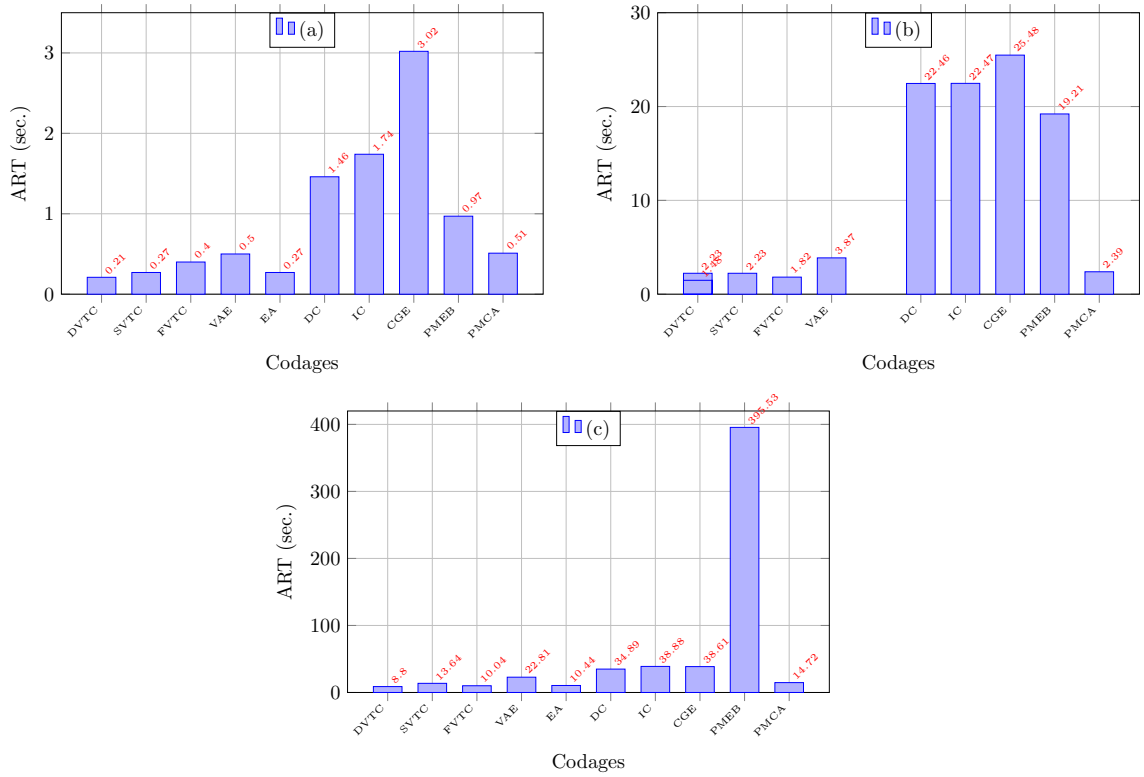


FIGURE 6.1 – *Comparaison des performaces par la métrique ART.*

6.6.2 Analyse des résultats de la métrique AES

Afin de mieux mesurer les efforts déployés par chaque codage pour arriver à sa meilleure solution, nous avons utilisé la métrique AES. Les résultats sont présentés dans la figure 6.2.

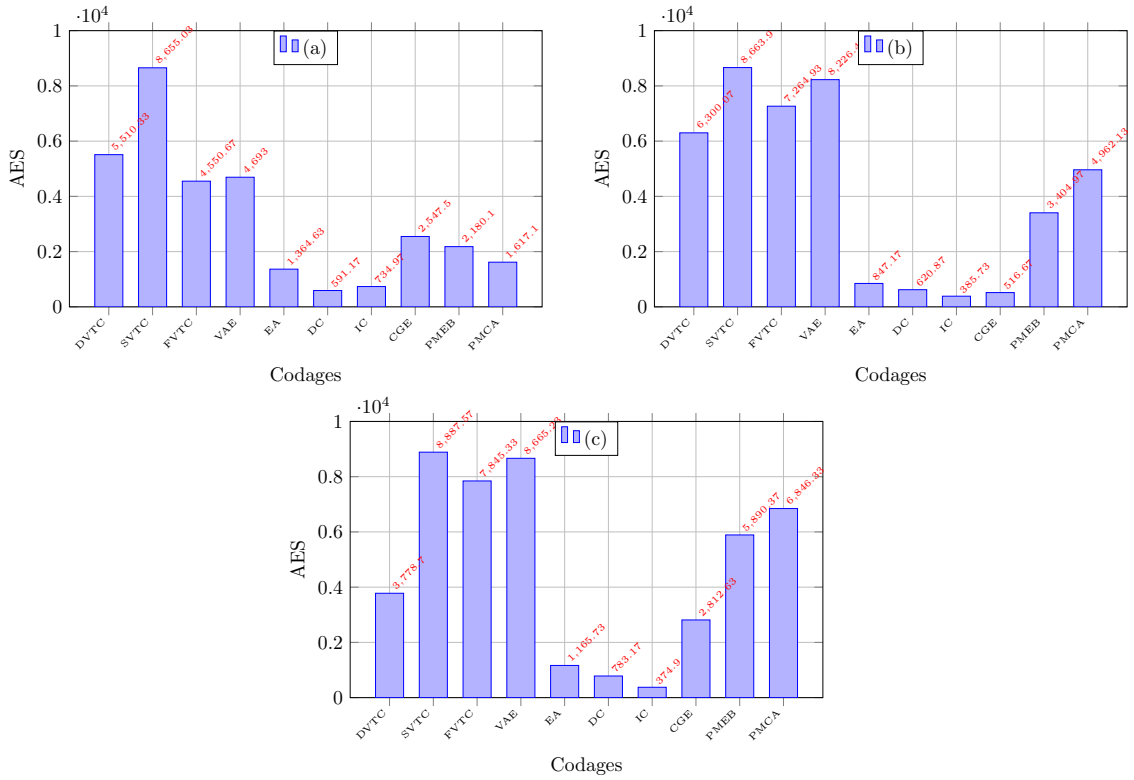


FIGURE 6.2 – *Comparaison des performaces par la métrique AES.*

Suivant les valeurs fournies par la métrique AES, les codages basés sur les sommets donnent les mauvais résultats. Tandis que ceux basés sur les arêtes présentent les meilleurs scores. Cet écart entre les valeurs de ART et celles de AES, c'est-à-dire la situation où le codage présente des valeurs élevées de AES avec un faible ART ou le contraire est dû à la complexité algorithmique des codages. En plus de la routine de décodage (génotype/phénotype) qui nécessite un temps de calcul additionnel. Par exemple, les codages DVTC et SVTC n'ont pas besoin de faire ce passage car les chromosomes représentent directement les solutions. Par conséquent le temps d'exécution est réduit. À l'opposé des codages DC, IC et CGE qui nécessite plus de traitements.

6.6.3 Analyse des résultats de la métrique MBF

Comme vu précédemment dans le paragraphe §6.4, l'utilisation des deux métriques précédentes n'est pas suffisante pour faire une comparaison équitable des performances des représentations. Pour la métrique MBF, le codage PMEB donne le meilleur score à travers toutes les instances de graphes à l'exception de la deuxième. Le codage PMCA est pour la majorité des instances dans la deuxième ou la troisième position. Ces résultats inattendus du PMCA prouvent que la prise en considération des contraintes n'a pas aussi d'impact qu'il le semble être. Par conséquent, nous pouvons supposer que les meilleures solutions peuvent être atteintes à partir des régions contenant des solutions non réalisables. Ensuite, nous avons le codage FVTC qui figure toujours parmi les quatre premières places. Le codage FVTC est suivi par le codage SVTC, VAE et à la fin le codage DVTC. Pour les codages basés sur les arêtes, le codage EA présente les meilleurs scores, suivis par IC, DC et à la fin le codage CGE.
La comparaison des performances des codages par famille en utilisant la métrique MBF, nous a permet de constater la supériorité de la famille des codage hybride suivi par celle basée sur les sommets et à la fin les codages basés sur les arêtes. Cependant, ce classement n'est pas absolu, car à l'exception des codages PMEB et PMCA, le restant des codages montrent une instabilité considérable comme le montre la métrique SDBF.

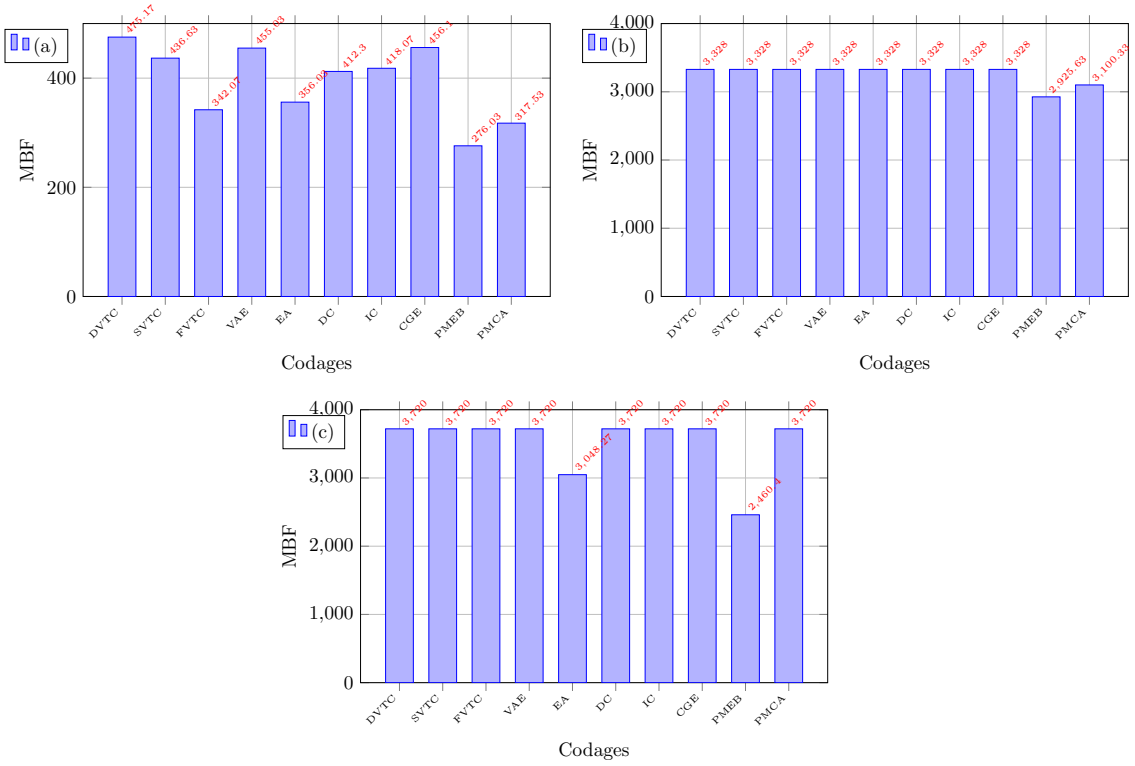


FIGURE 6.3 – *Comparaison des performaces par la métrique MBF.*

6.6.4 Analyse des résultats de la métrique SDBF

L'instabilité des résultats fournis par certains codages est due principalement à leur incapacité de produire des solutions réalisables pour toutes les exécutions. Dans ce cas, la valeur utilisée pour le calcul du MBF est remplacé par la somme totale des poids du graphe ce qui peut avoir un impact significatif sur la valeur de SDBF. Dans ce contexte, il est important de noter que les codages basés sur les sommets notamment SVTC, VAE et DVTC trouvent des difficultés pour satisfaire les contraintes de cohabitations, surtout avec l'augmentation de l'ordre du graphe et le nombre de couples de sommets à cohabiter. Ceci demeure vrai pour le codage FVTC qui présente une certaine instabilité malgré la qualité de ses solutions. En plus de la contrainte de cohabitation, ces codages ont du mal à assurer la contrainte de la taille maximale des clusters.

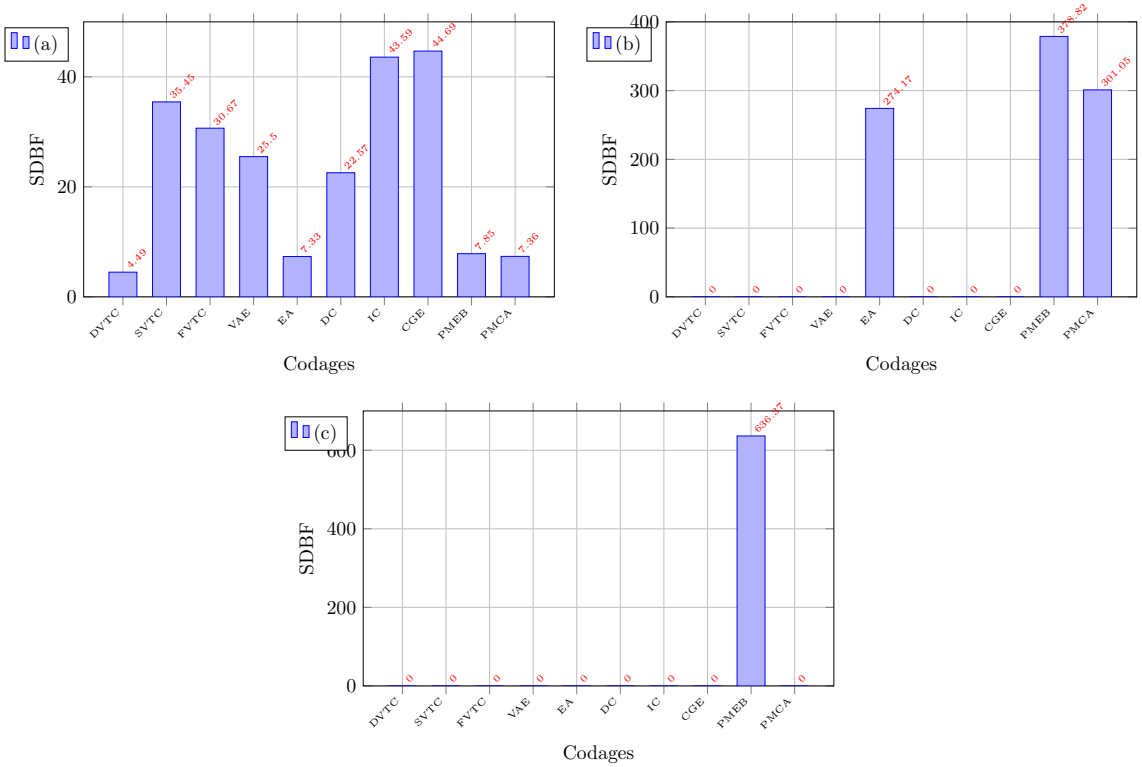


FIGURE 6.4 – *Comparaison des résultat par la métrique SDBF.*

Pour la famille des codages basés sur les arêtes, on peut noter que le codage EA devient incapable de converger vers une solution réalisable pour les instances d'une grande densité. En fait, avec l'augmentation de la densité des graphes, le codage EA rencontre des difficultés pour assurer la contrainte de la taille maximale des clusters. Dans la même famille, les codages DC et IC ont du mal à satisfaire les contraintes de cohabitation surtout pour les graphes à faible densité. De plus, le nombre de couples de sommets à cohabiter représente un autre obstacle pour ces représentations. Le codage CGE quant à lui n'arrive pas à assurer les contraintes de cohabitation et celle de la taille maximale des clusters. Les valeurs nulles de la métrique SDBF présentes pour certains codages reflètent apriori leur stabilité parfaite, or les valeurs MBF de ces codages correspondant à la même instance égale à la somme totale des poids de l'instance. Par conséquent, la valeur zéro de SDBF ce n'est en vérité qu'une convergence prématurée à une solution non réalisable pour les 30 exécutions comme le montrent les figures 6.3, 6.4.

6.7 Analyse globale des résultats

Dans le but d'avoir une vision plus large sur les performances de nos représentations génétique, nous avons opté pour une deuxième comparaison portant sur l'ensemble globale de jeu de données. A cet effet, nous avons choisi les boîtes à moustaches pour la représentation graphique des résultats. Les figures 6.5 et 6.6 nous offrent une comparaison des performances des codages par rapport aux métrique MBF et AES respectivement.

Afin de faciliter l'interprétation des graphiques, nous avons divisé la valeur MBF de chaque instance sur la somme totale de ses poids (voir l'entrée somme totale des poids dans la table 6.1). Ainsi les valeurs des MBF seront toutes comprises entre 0 et 1 . De cette façon, les médians proches de la valeur 1, correspondent aux codages qui ont abouti à une solution non réalisable. Ce comportement commence à partir de la dixième instance. D'autre part, la présence de moustaches près de la valeur zéro présume l'existence de bonnes solutions pour certains codages.

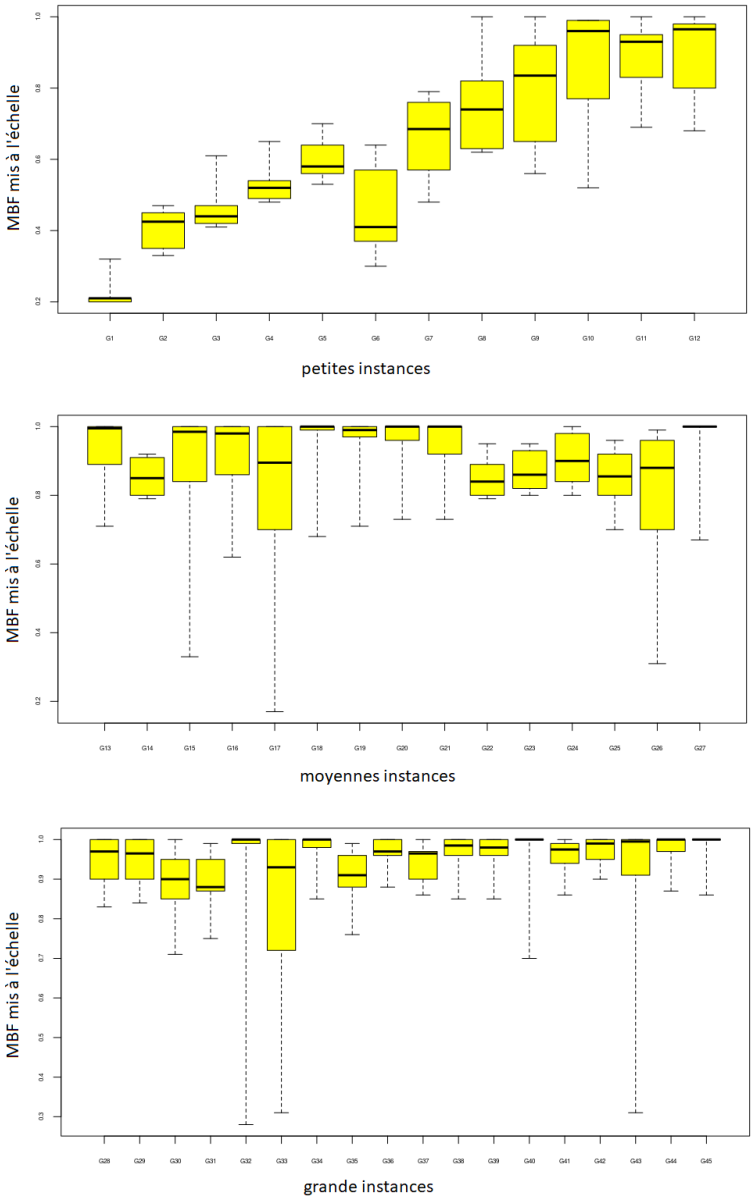


FIGURE 6.5 – *Les boites à moustaches des performances des codages pour la métrique MBF.*

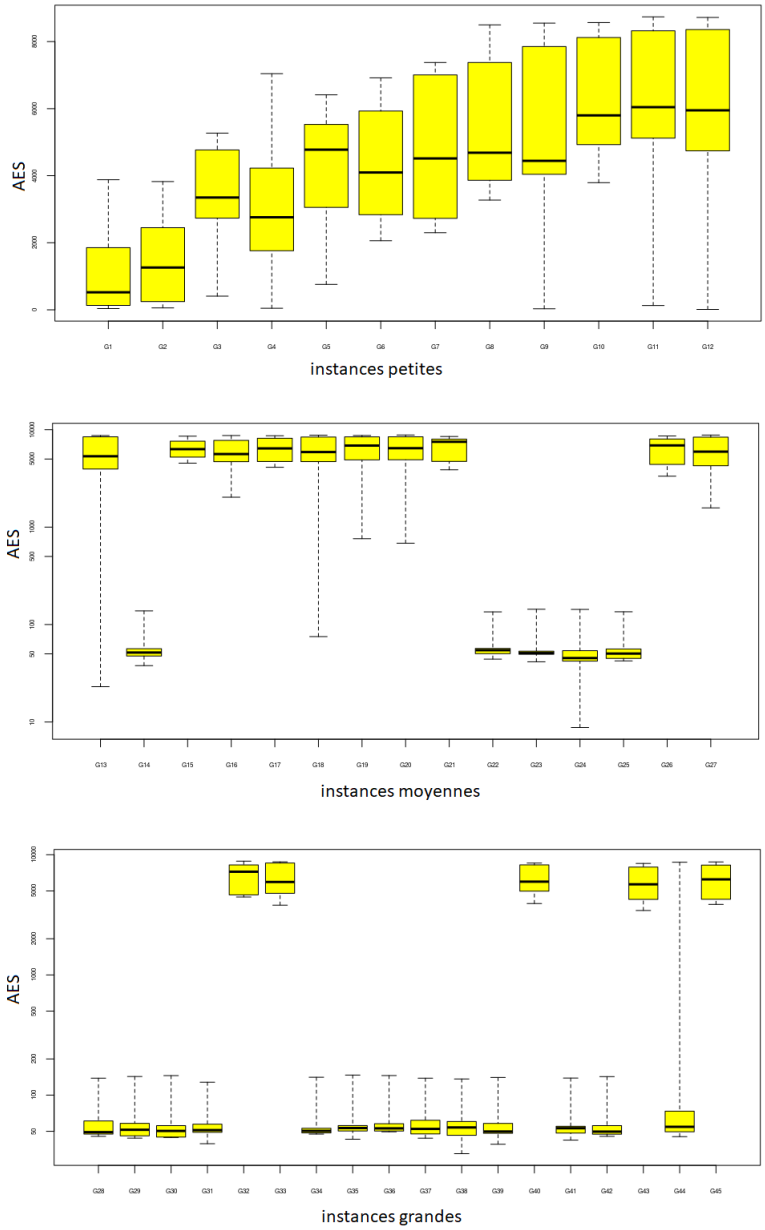


FIGURE 6.6 – *Les boites à moustaches des performances des codages pour la métrique AES.*

Dans le but d'identifier ces codages et voir la position de chacun dans les boîtes à moustaches, nous avons résumé dans la table 6.2 le nombre d'apparition de chaque codage dans le premier ($\in \#Q1$) ou le dernier quartile ($\in \#Q4$) pour les trois métriques MBF, AES et ART.
Selon la ligne correspondante à la métrique MBF, le codage le plus intuitive et le plus utilisé dans la littérature à savoir le DVTC figure uniquement six fois dans

le premier quartile $Q1$ tandis que les codages hybrides y figurent pratiquement pour toutes les instances sauf une seule.

	Quartiles	DVTC	SVTC	FVTC	VAE	EA	DC	IC	CGE	PMEB	PMCA
ART	$\# \in Q1$	45	23	41	1	22	12	0	0	13	27
	$\# \in Q4$	0	1	0	0	0	0	43	45	0	1
AES	$\# \in Q1$	0	0	0	45	45	2	7	33	34	14
	$\# \in Q4$	15	43	30	0	0	0	0	1	0	1
MBF	$\# \in Q1$	2	5	34	3	7	43	2	14	45	37
	$\# \in Q4$	25	22	2	33	32	0	17	17	0	2

TABLE 6.2 – *La présences des codages dans le premier et le dernier quartiles pour les métriques ART, AES et MBF.*

6.8 Comparaision des performances avec une méthode exacte (SCIP)

Bien que les résultats présentés jusqu'ici nous donnent une manière de comparer les performances des codages entre eux, ceci demeure insuffisant pour assurer une évaluation absolue. Afin d'atteindre cet objectif, nous avons besoin d'une approche exacte comme point de référence et sur laquelle nous allons nous baser pour décider de la qualité des solutions fournies par chaque codage. À cet effet, nous utilisons un modèle mathématique classique du problème de partitionnement de graphe présenté dans Merchichi and Boulif [2015]. Le partitionnement de graphes se fait par la suite en utilisant la méthode SCIP Berthold et al. [2012].

La table 6.3 présente une comparaison entre les meilleures valeurs de la fonction objectif de chaque codage pour chacune des 45 instances à travers les trente exécutions avec les valeurs obtenues par la méthode SCIP. Etant une méthode exacte, la SCIP énumère toutes les solutions de l'espace de recherche pour arriver la solution optimale. Or, comme il s'agit d'un problème NP-complet, le temps d'exécution de la méthode devient très important. À cet effet, nous avons fixé la durée d'exécution de la SCIP à deux heures. Une fois la durée fixée écoulée,

nous récupérons la solution obtenue pour la comparer ensuite à celles de nos codages.

Dans la table 6.3, les valeurs avec * en exposant représentent des valeurs optimales pour les instances en question. Ceci veut dire que la SCIP a pu obtenir la solution optimale en une durée inférieure ou égale deux heures sinon la solution n'est pas optimale. Les résultats obtenus montrent à nouveau la supériorité des codages hybrides. PMEB donne des solutions qui sont généralement meilleures que celles de la SCIP quand cette dernière n'arrive pas à trouver la solution optimale dans la durée allouée. Nous pouvons aussi constater que le codage FVTC donne de meilleures résultats comparés à ceux de sa famille, particulièrement quand l'ordre du graphe augmente. De même, le codage FVTC surclasse les codages de la famille basée sur les arêtes pour les graphes ayant une forte densité. Dans le cas contraire, c'est-à-dire pour les graphes à faible densité, le codage EA fonctionne beaucoup mieux. Chose qui demeure vrai pour le codage CGE.

	Encoding schemes										
Graphs	DVTC	SVTC	FVTC	VAE	EA	DC	IC	CGE	PMEB	PMCA	SCIP
1	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2***
2	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	7	7	**5***
3	**42**	**42**	43	**42**	**42**	**42**	**42**	46	**42**	43	**42***
4	**10**	**10**	**10**	**10**	**10**	**10**	**10**	12	**10**	**10**	**10***
5	99	**93**	**93**	101	95	95	105	112	96	97	**93***
6	17	15	11	20	12	14	12	17	10	11	**9***
7	29	27	25	31	24	32	32	27	21	24	**20**
8	228	198	177	219	-	191	204	201	174	170	**159**
9	414	369	296	404	-	376	343	351	**262**	303	279
10	259	254	214	263	211	287	279	247	149	**121**	184
11	1298	1234	1111	1233	-	1286	1210	1178	**910**	925	944
12	1416	1353	1155	1350	-	1325	1278	1270	**921**	1046	941
13	1904	1872	1693	1850	-	1948	1957	1845	**1402**	1444	1421
14	3173	3132	3022	3078	3485	3198	3542	3303	**3044**	3079	3086
15	-	-	125	-	115	-	185	102	**54**	104	112
16	2541	-	2008	2507	-	2393	2235	-	**1574**	1862	1674
17	-	-	704	-	-	939	828	825	**181**	561	210
18	-	-	2729	-	-	3216	3105	-	**2211**	2464	2286
19	4147	4147	3373	4043	4147	3989	4001	4147	2825	3323	**2803**
20	-	-	4295	-	-	5027	4995	-	**3406**	3898	3410
21	-	-	497	-	434	-	-	460	**402**	452	531
22	**2266**	2987	2453	2503	2788	2561	2931	2726	2298	2673	2462
23	12857	13325	12530	12487	14258	12907	14438	13490	**12429**	12643	12663
24	23056	-	22701	22387	-	23307	25734	24112	22395	**22361**	22422
25	10572	12947	11154	11822	13211	12148	13552	12752	11026	**9916**	11695
26	359	354	306	351	258	374	373	178	**112**	220	233
27	-	-	5454	-	-	-	-	-	**4164**	4939	4274
28	-	-	18260	-	20055	18946	-	18837	**17965**	18140	18157
29	-	-	18340	18049	20261	19119	-	18909	**18041**	18210	18317
30	11982	13531	11873	12639	13599	13111	14431	12673	10672	**9929**	12244
31	16531	17746	16427	16606	17689	17520	18864	16621	14685	**14099**	16355
32	-	-	3137	-	-	-	-	3208	**971**	2333	1113
33	-	746	613	766	536	757	755	384	**219**	484	253
34	-	-	8626	-	9147	9289	-	-	**8058**	8685	8614
35	6912	7694	6984	7209	7504	7813	8099	7540	**5968**	6106	6865
36	-	-	7032	7240	7494	7906	-	7666	**5929**	6241	6898
37	14865	16004	14663	14791	15822	15774	16572	15559	14097	**13926**	14651
38	15916	-	15377	15425	16407	16468	-	16125	14133	**13927**	15393
39	-	-	15336	15501	16488	16193	-	16257	14533	**13998**	15040
40	-	-	2019	-	-	-	-	-	**1636**	1928	1978
41	10875	12036	11111	11182	11795	12141	-	11598	**10663**	11243	11143
42	-	-	110728	109354	119808	115293	-	-	109342	**109327**	109359
43	-	-	-	-	549	-	-	373	**207**	451	435
44	-	-	282611	-	303303	305824	-	-	266250	**265258**	-
45	-	-	-	-	-	-	-	-	**8213**	-	8731

TABLE 6.3 – *Comparaison des meilleures fitness des codages*

6.9 APPLICATION DU TEST DE FRIEDMAN

Dans le but d'avoir une vérification statistique des résultats obtenus, nous avons réalisé un test de Friedman Friedman [1937]. Le test de Friedman est un test non paramétrique utilisé lorsque nous sommes en présence de k $(k > 2)$ échantillons appariés correspondant à k traitements, afin de mettre en évidence

les différences entre ces traitements. L'application du test de Friedman suppose la non normalité des échantillons traités (ne sont pas normalement distribués), ce qui est le cas pour les résultats des expériences basées sur des algorithmes stochastiques Derrac et al. [2011].

6.9.1 Principe de test de Friedman

Afin de faciliter l'interprétation des résultats du test de Friedman, dans ce qui suit nous allons donner quelques définitions.

— Les n « blocs » : représentent la variable d'appariement du test.
— Les K « échantillons » : représentent les éléments que nous souhaitons tester.

Le couple d'hypothèses est défini comme suit :

— Hypothèse nulle Ho : cette hypothèse suppose la non existence de différences statistiques significatives entre les échantillons.
— Hypothèse non nulle H1 : existence de différences statistique significatives entre les échantillons.

6.9.2 La table des rangs

La table 6.4 représente les rangs obtenus par l'application de test de Friedman. Les lignes de la table représentent les échantillons (les graphes), et les colonnes représentent les blocs (les codages).

Graph #	DVTC	SVTC	FVTC	VAE	EA	DC	IC	CGE	PMEB	PMCA
1	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5	5.5
2	4.5	4.5	4.5	4.5	4.5	4.5	4.5	4.5	9.5	9.5
3	4.0	4.0	8.5	4.0	4.0	4.0	4.0	10.0	4.0	8.5
4	5.0	5.0	5.0	5.0	5.0	5.0	5.0	10.0	5.0	5.0
5	7.0	1.5	1.5	8.0	3.5	3.5	9.0	10.0	5.0	6.0
6	8.5	7.0	2.5	10.0	4.5	6.0	4.5	8.5	1.0	2.5
7	7.0	5.5	4.0	8.0	2.5	9.5	9.5	5.5	1.0	2.5
8	9.0	5.0	3.0	8.0	10.0	4.0	7.0	6.0	2.0	1.0
9	9.0	6.0	2.0	8.0	10.0	7.0	4.0	5.0	1.0	3.0
10	7.0	6.0	4.0	8.0	3.0	10.0	9.0	5.0	2.0	1.0
11	9.0	7.0	3.0	6.0	10.0	8.0	5.0	4.0	1.0	2.0
12	9.0	8.0	3.0	7.0	10.0	6.0	5.0	4.0	1.0	2.0
13	7.0	6.0	3.0	5.0	10.0	8.0	9.0	4.0	1.0	2.0
14	6.0	5.0	1.0	3.0	9.0	7.0	10.0	8.0	2.0	4.0
15	8.5	8.5	5.0	8.5	4.0	8.5	6.0	2.0	1.0	3.0
16	7.0	9.0	3.0	6.0	9.0	5.0	4.0	9.0	1.0	2.0
17	8.5	8.5	3.0	8.5	8.5	6.0	5.0	4.0	1.0	2.0
18	8.0	8.0	3.0	8.0	8.0	5.0	4.0	8.0	1.0	2.0
19	8.5	8.5	3.0	6.0	8.5	4.0	5.0	8.5	1.0	2.0
20	8.0	8.0	3.0	8.0	8.0	5.0	4.0	8.0	1.0	2.0
21	8.0	8.0	5.0	8.0	2.0	8.0	8.0	4.0	1.0	3.0
22	1.0	10.0	3.0	4.0	8.0	5.0	9.0	7.0	2.0	6.0
23	5.0	7.0	3.0	2.0	9.0	6.0	10.0	8.0	1.0	4.0
24	5.0	9.5	4.0	2.0	9.5	6.0	8.0	7.0	3.0	1.0
25	2.0	8.0	4.0	5.0	9.0	6.0	10.0	7.0	3.0	1.0
26	8.0	7.0	5.0	6.0	4.0	10.0	9.0	2.0	1.0	3.0
27	7.0	7.0	3.0	7.0	7.0	7.0	7.0	7.0	1.0	2.0
28	8.5	8.5	3.0	8.5	6.0	5.0	8.5	4.0	1.0	2.0
29	9.0	9.0	4.0	2.0	7.0	6.0	9.0	5.0	1.0	3.0
30	4.0	8.0	3.0	5.0	9.0	7.0	10.0	6.0	2.0	1.0
31	4.0	9.0	3.0	5.0	8.0	7.0	10.0	6.0	2.0	1.0
32	7.5	7.5	3.0	7.5	7.5	7.5	7.5	4.0	1.0	2.0
33	10.0	6.0	5.0	9.0	4.0	8.0	7.0	2.0	1.0	3.0
34	8.0	8.0	2.0	8.0	4.0	5.0	8.0	8.0	1.0	3.0
35	3.0	8.0	4.0	5.0	6.0	9.0	10.0	7.0	1.0	2.0
36	9.0	9.0	3.0	4.0	5.0	7.0	9.0	6.0	1.0	2.0
37	5.0	9.0	3.0	4.0	8.0	7.0	10.0	6.0	2.0	1.0
38	5.0	9.5	3.0	4.0	7.0	8.0	9.5	6.0	2.0	1.0
39	9.0	9.0	3.0	4.0	7.0	5.0	9.0	6.0	2.0	1.0
40	7.0	7.0	3.0	7.0	7.0	7.0	7.0	7.0	1.0	2.0
41	2.0	8.0	3.0	4.0	7.0	9.0	10.0	6.0	1.0	5.0
42	8.5	8.5	4.0	3.0	6.0	5.0	8.5	8.5	2.0	1.0
43	7.5	7.5	7.5	7.5	4.0	7.5	7.5	2.0	1.0	3.0
44	8.0	8.0	3.0	8.0	4.0	5.0	8.0	8.0	2.0	1.0
45	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0	1.0	6.0
Average	6.733	7.300	3.622	6.011	6.633	6.456	7.433	6.111	1.867	2.833

TABLE 6.4 – *Tables des rangs du test de Friedman*

La figure 6.7 nous donne une représentation visuelle des résultats de l'application de test de Friedman pour une meilleure interprétation.

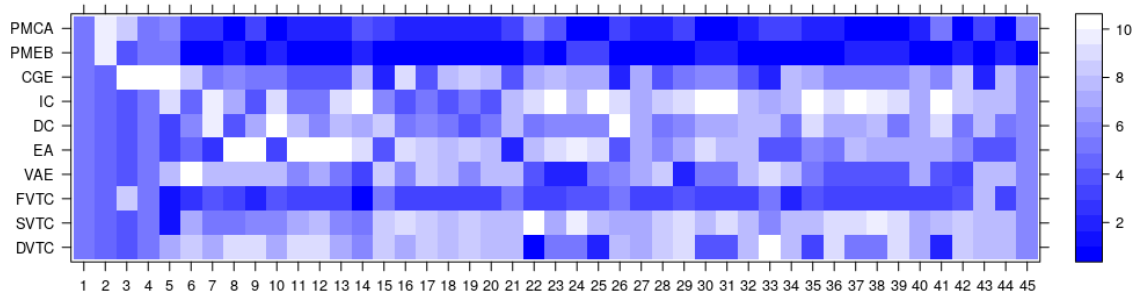


FIGURE 6.7 – *Représentation visuelle des rangs du test de Friedman.*

6.9.3 La valeur de p-value

La p-value est utilisée pour quantifier la significativité statistique d'un résultat dans le cadre d'une hypothèse nulle. Elle représente la probabilité d'erreur en rejetant l'hypothèse nulle si elle est vraie. Plus la valeur de p-value est petite, plus la probabilité d'erreur en rejetant l'hypothèse nulle est faible. Une valeur limite de $0,05$ est souvent utilisée.
La p-value calculé par la procédure de test de Friedman est inférieure à $2.2e-16$. L'hypothèse nulle dans ce cas-là est rejetée. Par conséquent, l'hypothèse H1 est retenue, ce qui signifie l'existence de différences statistiquement significatives entre les résultats obtenus par chaque codage. Ces différences proviennent probablement des performances de PMEB, PMCA et FVTC comme le suggère la ligne sombre dans la figure 6.7.

6.10 Application du test de Nemenyi (post-hoc)

Le rejet de l'hypothèse nulle par le test de Friedman suppose l'existence de différences statistiquement significatives. Afin de déterminer les couple d'échantillon à l'origine de ses différences nous allons procéder à un test post-hoc à savoir le test de Nemenyi. En statistique, le test de Nemenyi est un test post-hoc utilisé pour trouver les groupes de données qui diffèrent après application d'un test statistique de multi-comparaisons avec (p-value $<$ 0.05) c'est-à-dire, hypothèse nulle rejetée Nemenyi [1963]. Le principe de ce test repose sur des

comparaisons deux à deux entre tous les échantillons pour enfin déterminer les pairs à l'origine des différences.
La table 6.5 montre les résultats obtenus par l'utilisation de package PMCMR de R. Le test retourne la matrice triangulaire inférieure qui contient les valeurs p-value des comparaisons deux-à-deux des performances des codages.

vs	DVTC	SVTC	FVTC	VAE	EA	DC	IC	CGE	PMEB
SVTC	0.99686	-	-	-	-	-	-	-	-
FVTC	4.8e-05	3.7e-07	-	-	-	-	-	-	-
VAE	0.98156	0.58524	0.00700	-	-	-	-	-	-
EA	1.00000	0.98950	0.00010	0.99365	-	-	-	-	-
DC	0.99999	0.94863	0.00038	0.99954	1.00000	-	-	-	-
IC	0.98516	1.00000	1.1e-07	0.43715	0.96351	0.87960	-	-	-
CGE	0.99365	0.69452	0.00383	1.00000	0.99833	0.99995	0.54782	-	-
PMEB	1.2e-12	9.4e-14	0.15385	3.8e-09	3.7e-12	2.9e-11	8.2e-14	1.3e-09	-
PMCA	4.5e-08	1.2e-10	0.96667	2.8e-05	1.2e-07	6.2e-07	2.6e-11	1.2e-05	0.88684

TABLE 6.5 – *Les résultats du test Friedman-Nemenyi post-hoc*

La figure 6.8 présente les mêmes résultats que de la table 6.5 sous forme graphique.

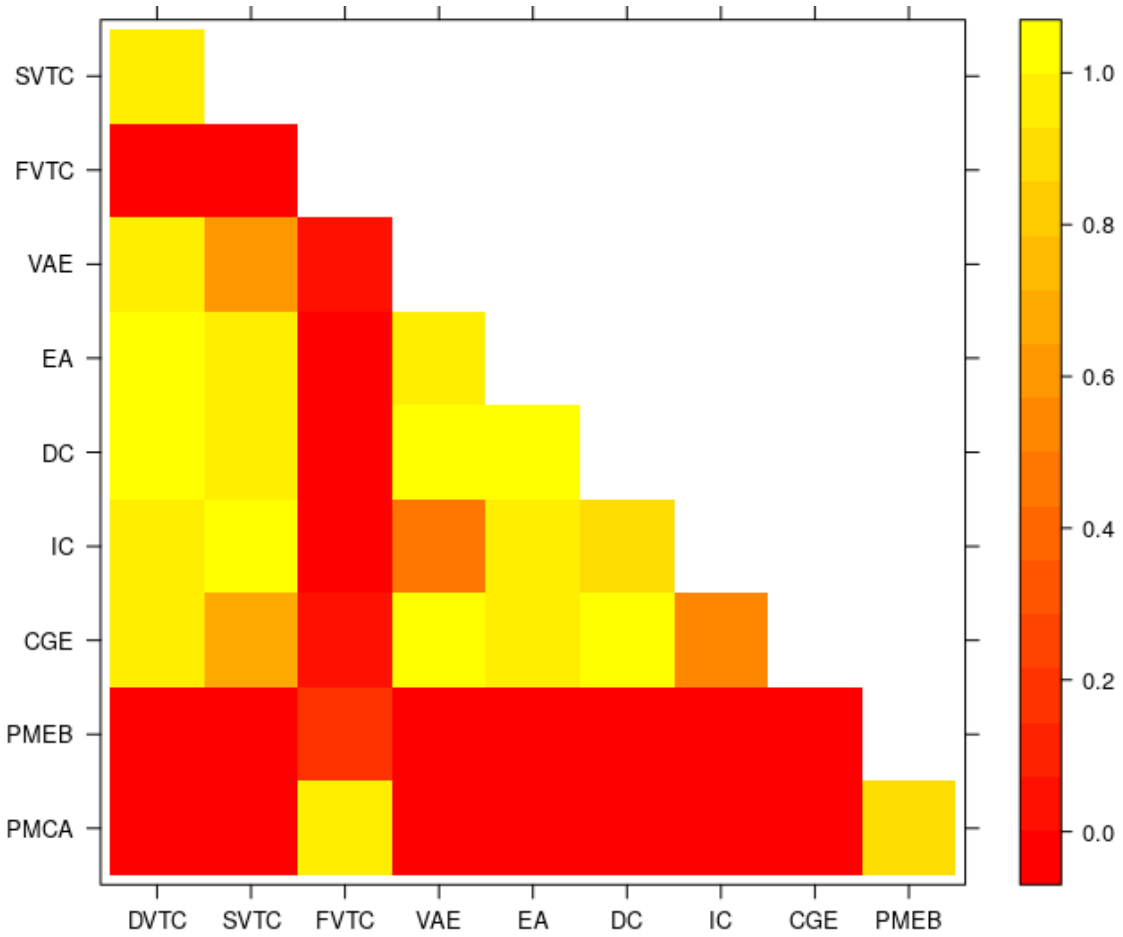


FIGURE 6.8 – *Représentation visuelle des résultats de test de Friedman-Nemenyi post hoc.*

Les résultats présentés dans la table 6.5 et comme on peut le voir sur la figure 6.8, les codages PMEB, PMCA et FVTC dans cet ordre représentent les meilleurs codages par rapport aux résultats de leurs performances. Ceci correspond parfaitement à notre analyse faite au début de ce chapitre.

6.11 Comparaison des performances avec le meilleur paramétrage de l'AG pour chacune des représentation

L'analyse des performances présentée jusqu'à présent se base sur l'utilisation de la même plateforme génétique avec le même paramétrage dans le but d'avoir

une comparaison équitable entre les performances des différents codages. En plus de cette comparaison, nous avons procédé à une deuxième stratégie qui consiste à mettre chaque codage dans son environnement favorable en terme de paramétrage de l'AG. Ceci va nous permettre de voir si les performances des variantes gagnantes ont été boostées par le paramétrage de l'AG en détriment des autres. À cet effet, nous avons réalisé une deuxième série de tests en utilisant le meilleur paramétrage de l'AG pour chaque codage. Les valeurs des paramètres sont fixées après une série de tests sur chaque instance. La table 6.6 résume les intervalles des meilleurs des paramètres pour chaque codage trié par famille. Le taux de reproduction ainsi que le nombre de générations sans amélioration ne sont pas modifiés, ils sont respectivement 10 et 30 pour toutes les instances. Les résultats de cette deuxième série de tests sont présentés dans la table 6.7.

Codages	petit			moyen			grand		
	taille. Pop	r_1(% Elit.)	r_3(% Mut.)	Pop. size	r_1(% Elit.)	r_3(% Mut.)	Pop. size	r_1(% Elit.)	r_3(% Mut.)
DVTC	[100,150]	10	3	[150,250]	[10,20]	3	[250,300]	[20,30]	3
SVTC	[100,150]	10	1	[150,250]	[10,20]	1	[250,300]	[20,30]	1
FVTC	[100,150]	10	2	[150,250]	[10,20]	2	[250,300]	[20,30]	2
VAE	[50,100]	10	2	[100,150]	[10,20]	2	[150,200]	[20,30]	2
EA	[50,100]	10	2	[100,150]	[10,20]	2	[150,200]	[20,30]	2
DC	[50,100]	10	1	[100,150]	[10,20]	1	[150,200]	[20,30]	1
IC	[50,100]	10	1	[100,150]	[10,20]	1	[150,200]	[20,30]	1
CGE	[50,100]	10	3	[100,150]	[10,20]	3	[150,200]	[20,30]	3
PMEB	[100,150]	10	2	[150,250]	[10,20]	2	[250,300]	[20,30]	2
PMCA	[100,150]	10	1	[150,250]	[10,20]	1	[250,300]	[20,30]	1

TABLE 6.6 – *Les valeurs des meilleurs paramètres de l'AG pour les codages.*

Graphes	Les représentaions génétiques																			
	DVTC		SVTC		FVTC		VAE		EA		DC		IC		CGE		PMEB		PMCA	
#	BF	BRT	BF	BRT	BF	BRT	BF	BRT	BF	BRT	BF	BRT	BF	BRT	BF	BRT	BF	BRT	BF	BRT
1	**2**	**0.06**	**2**	0.23	**2**	0.40	**2**	4.45	**2**	2.54	**2**	4.60	**2**	3.27	**2**	3.80	**2**	1.16	**2**	0.18
2	**5**	**0.07**	**5**	0.24	**5**	0.42	**5**	4.29	**5**	2.55	**5**	4.64	**5**	3.23	**5**	3.73	**5**	1.15	7	0.29
3	**42**	0.13	**42**	0.27	**42**	0.57	**42**	4.29	**42**	4.20	**42**	5.80	**42**	6.24	**42**	6.93	**42**	1.21	43	0.43
4	**10**	0.13	**10**	0.13	**10**	0.36	**10**	4.33	**10**	2.48	**10**	5.12	**10**	4.83	**10**	8.13	**10**	1.18	**10**	0.29
5	**93**	0.20	**93**	0.19	**93**	0.54	96	5.56	95	4.04	**93**	9.36	97	9.05	97	11.52	96	1.47	97	0.35
6	11	0.22	10	0.70	**9**	**0.83**	16	7.32	12	2.50	**9**	7.49	12	10.29	11	5.74	**9**	1.57	11	0.66
7	23	0.29	22	0.49	22	1.66	30	5.26	23	2.55	25	11.67	29	15.74	23	5.74	**20**	1.42	23	0.86
8	166	0.87	168	1.01	163	1.19	176	5.42	235	0.45	**158**	3.73	184	2.78	191	4.14	168	4.26	170	0.41
9	276	2.16	287	1.93	270	3.62	328	7.18	386	2.89	292	15.19	343	17.08	328	18.60	**262**	4.33	303	1.06
10	162	4.92	174	4.17	142	10.93	238	1.88	204	3.20	233	35.21	206	37.54	206	22.73	**109**	4.93	121	2.12
11	1093	6.36	1046	19.34	997	12.18	1086	6.26	1216	3.87	1138	20.97	1210	51.22	1155	34.49	**910**	16.05	917	7.66
12	1143	6.29	1143	5.66	1053	10.44	1137	2.68	1293	3.98	1140	23.61	1245	49.57	1239	20.91	**921**	7.70	1034	2.34
13	1640	9.65	1613	33.65	1579	21.16	1694	8.68	1828	4.58	1828	38.07	1935	76.98	1813	23.72	**1391**	15.10	1428	1.64
14	**2987**	26.22	3104	40.81	3003	25.66	3077	10.39	3467	5.65	3148	74.51	3301	25.68	3259	136.34	3037	9.05	3067	2.04
15	-	-	-	-	85	16.92	139	7.34	73	4.71	137	19.77	133	30.74	85	18.25	**48**	10.66	99	8.42
16	1800	41.86	2108	22.75	1836	37.17	2016	4.73	2216	5.98	2024	72.42	2233	67.40	2191	24.80	**1574**	8.86	1855	3.82
17	231	31.08	675	93.53	315	44.35	800	5.21	638	5.55	559	61.49	828	11.82	784	11.65	**181**	12.21	542	13.73
18	2620	31.88	2796	34.76	2495	57.30	2718	9.56	2943	9.44	2802	121.91	3105	36.15	2883	27.22	**2191**	27.34	2430	9.46
19	3337	49.46	3566	80.23	3159	61.61	3372	7.55	3701	8.17	3492	62.56	3663	47.42	3564	20.16	**2792**	39.24	3301	44.17
20	4112	94.21	4554	71.99	3959	79.95	4233	9.67	4775	1.38	4353	300.77	4952	118.03	4456	37.24	**3406**	64.35	3900	13.87
21	-	-	-	-	414	127.64	523	14.60	432	10.42	566	13.04	-	-	447	51.26	**364**	42.27	431	7.17
22	**2138**	248.37	2773	189.76	2447	83.53	2503	4.17	2788	2.92	2552	461.61	2700	32.85	2680	75.17	2237	52.42	2671	34.41
23	**12154**	196.47	13135	224.42	12344	131.20	12487	4.46	14185	13.34	12638	320.67	13421	34.63	13430	25.41	12358	37.77	12639	34.10
24	**22255**	136.38	24002	169.61	22336	77.71	22387	3.97	-	-	23257	1094.75	24095	77.56	23956	27.58	22381	147.13	22351	57.83
25	**9688**	226.52	11567	337.76	10781	242.82	11804	8.83	13115	13.25	11827	834.29	12704	31.19	12614	21.96	10680	106.86	9758	127.06
26	167	204.47	289	148.63	232	133.42	311	12.32	220	10.23	353	39.62	366	82.79	170	10.31	**100**	55.59	204	11.70
27	-	-	-	-	5194	43.07	5310	3.70	5739	11.98	5692	103.22	-	-	5470	28.86	**4139**	32.24	4866	5.19
28	**17471**	380.45	-	-	18259	115.75	18048	11.57	20041	19.36	18799	2332.52	-	-	18810	29.97	17896	97.32	18105	67.31
29	-	-	-	-	18257	216.37	18048	11.59	20206	18.27	19084	1748.00	-	-	18830	29.60	**17947**	98.73	18199	37.05
30	10549	628.19	12719	484.52	11818	308.38	12639	13.79	13408	2.80	11782	937.15	13222	84.72	12654	137.11	10413	450.09	**9817**	56.91
31	15894	432.15	16860	683.62	16418	208.08	16606	15.82	17662	39.63	17520	195.27	17371	80.84	16614	292.33	14569	307.89	**13994**	245.82
32	1192	800.68	-	-	2632	322.52	3055	5.72	2561	25.23	3257	56.42	3127	164.05	3183	49.02	**818**	63.28	1885	251.44
33	-	-	629	563.12	501	245.24	636	30.49	338	38.74	699	550.51	744	158.76	352	35.94	**174**	70.64	434	51.49
34	8460	592.41	-	-	8590	192.16	-	-	9145	29.03	8577	667.46	-	-	9410	40.68	**8048**	125.83	8685	19.14
35	6617	681.32	7360	1275.07	6979	343.58	7169	19.77	7477	33.71	7813	107.84	7597	64.88	7540	66.58	**5771**	358.11	6077	201.08
36	-	-	-	-	7006	369.37	7240	6.12	7488	32.37	6992	617.57	-	-	7630	38.02	**5841**	252.64	6207	136.81
37	14528	722.11	15530	647.92	14638	465.12	14791	6.30	15806	32.79	15774	623.47	15640	102.63	15433	45.44	14072	257.25	**13860**	295.94
38	15325	493.66	15550	1516.10	15277	308.52	15425	6.35	16361	32.27	16468	480.07	16329	118.37	16125	56.86	14133	255.74	**13927**	25.00
39	15466	551.40	-	-	15004	573.15	15480	31.26	16488	4.65	15436	1224.94	-	-	16177	44.95	14512	250.14	**13892**	381.93
40	-	-	-	-	1790	359.75	1955	7.16	1929	36.37	2126	1624.87	-	-	2026	31.03	**1473**	254.39	1926	30.89
41	10868	1014.78	12031	249.83	10974	367.39	11166	25.06	11789	40.31	11044	1334.99	11773	132.55	11564	55.72	**10624**	318.73	11192	207.00
42	-	-	-	-	110531	205.91	109345	22.67	118842	69.90	115293	4237.38	-	-	-	-	109327	314.72	**109320**	244.53
43	-	-	-	-	510	359.91	611	7.69	311	35.72	667	669.45	-	-	345	37.92	**178**	144.99	410	39.76
44	-	-	-	-	279505	548.68	-	-	303256	193.94	305824	285.90	-	-	-	-	266229	802.93	**264861**	452.02
45	-	-	-	-	8992	2480.74	9143	19.03	9359	15.88	9565	254.82	-	-	-	-	**8125**	780.53	-	-

TABLE 6.7 – *La meilleur fitness avec le meilleur temps d'exécution correspondant aux meilleurs paramètres.*

En adoptant le même test statistique vu précédemment notamment le test de Friedman, nous obtenons les valeurs moyennes des rangs représentés dans la table 6.8.

Graph #	DVTC	SVTC	FVTC	VAE	EA	DC	IC	CGE	PMEB	PMCA
Average	4.700	6.933	3.433	5.778	7.144	6.044	8.456	6.678	2.000	3.833

TABLE 6.8 – *La moyenne des rangs de la deuxième série de tests*

Le test de Friedman révèle à nouveau une différence statistique significative entre

les performances des codages. À cet effet, nous procédons au test post-hoc de nemenyi pour déterminer les paires à l'origine des différences. Les résultats de l'application du test de nemenyi sont résumés dans la table 6.9.

vs	DVTC	SVTC	FVTC	VAE	EA	DC	IC	CGE	PMEB
SVTC	0.01688	-	-	-	-	-	-	-	-
FVTC	0.61006	1.9e-06	-	-	-	-	-	-	-
VAE	0.80233	0.72883	0.00907	-	-	-	-	-	-
EA	0.00502	1.00000	2.7e-07	0.49807	-	-	-	-	-
DC	0.52288	0.92987	0.00176	0.99999	0.78240	-	-	-	-
IC	1.8e-07	0.33480	3.1e-13	0.00113	0.56030	0.00614	-	-	-
CGE	0.06066	1.00000	1.6e-05	0.92454	0.99932	0.99276	0.14131	-	-
PMEB	0.00097	5.5e-13	0.42524	1.5e-07	1.6e-13	1.1e-08	1.0e-13	1.1e-11	-
PMCA	0.93976	5.2e-05	0.99981	0.07054	9.4e-06	0.01902	2.0e-11	0.00036	0.11336

TABLE 6.9 – *Les résultat du Nemenyi post hoc sur les performances correspondant aux meilleurs paramètres.*

Les résultats obtenus durant cette deuxième phase de test révèlent une amélioration des performances du codage DVTC (voir la table 6.7 où les meilleures performances sont en gras). En revanche, malgré cette amélioration, l'ordre des meilleurs scores n'est pas changé en comparaison avec celui obtenu dans la première série de tests. Ceci peut clairement apparaitre dans les p-values qui sont très significatives entre les codages PMEB, PMCA et FVTC, contrairement avec les autres codages qui ne sont pas très significatives.

6.12 Impact des propriétés génétiques sur les résultats

Les différences entre les résultats expérimentales des représentations génétiques trouvent leurs justifications aussi par rapport aux propriétés génétique étudiées dans le chapitre 4. Dans ce qui suit nous allons mettre en évidence l'influence des propriétés génétiques notamment la redondance, la cécité, la légalité et l'épistasie sur nos résultats. La causalité quant à elle ne peut pas être utilisée pour justifier les résultats car tous les codages la présente de la même manière.

6.12.1 Redondance

Une forte redondance inonde l'espace de recherche par des solutions similaires qui freine le processus d'exploration et l'empêche d'atteindre des zones qui

peuvent être prometteuses. Ceci est expliqué par les score obtenu par les codages (DVTC, SVTC, EA) qui souffre d'une forte redondance. En outre, une très faible redondance voir son absence totale peut entraîner une baisse des performances de l'algorithme génétique en empêchant sa convergence vers un optimum globale, comme s'est fait le cas du codage VAE. En revanche une redondance modérée, comme celle présente chez les codages (FVTC, DC,IC,CGE, PMEB, PMCA) peut jouer un rôle très favorable pour l'exploration de l'espace de recherche et aboutir enfin à des solutions de qualité.

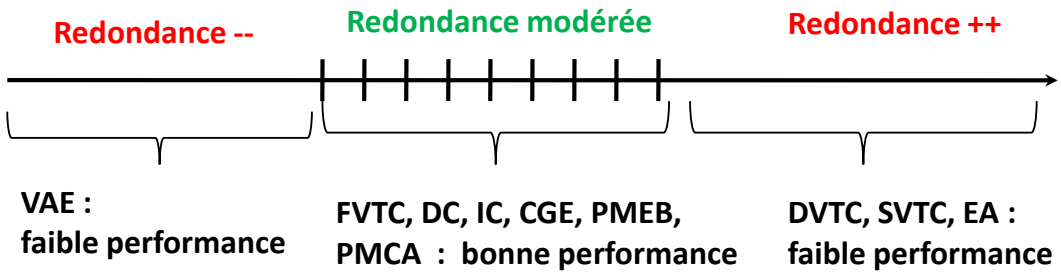


FIGURE 6.9 – *L'influence de la redondance sur les résultats des codages.*

6.12.2 Complétude

Définie par l'aptitude des codages à représenter (voir) l'ensemble de toutes les solutions de l'espace de recherche, ceci peut nous faire comprendre à tort que les codages complets sont les plus favorables. Or, une analyse minutieuse consolidée par les résultats de nos expériences nous montre autre chose. En effet, les scores présentés par les codages DVTC, SVTC, VAE qui sont considérés comme complets, sont nettement inférieurs par rapport à ceux de PMEB, PMCA qui souffrent du problème de la cécité. Ceci trouve sa justification par le fait que la cécité peut être considéré comme un facteur permettant la redirection du processus de recherche vers les zones prometteuses, en réduisant les efforts fournis par le codages (la métrique AES).

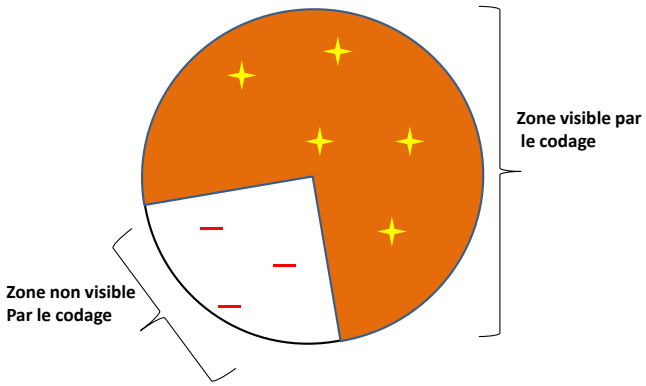


FIGURE 6.10 – *Le rôle de la cécité dans l'amélioration du processus de recherche.*

D'autre part, si cette dernière est redirigé vers les zones contenant de bonnes solutions, dans ce cas là l'AG sera pénalisé et ne pourra pas aboutir à des solutions de qualité comme le montre les résultats des codages DC, IC, CGE et même EA. La zone en orange de la figure 6.10 représente la zone visible par un codage, avec les étoiles représentent des solutions de bonne qualité. Tandis que la partie non visible qui contient les moins (solutions de qualité inférieures) représente la zone non visible.

6.12.3 Légalité

Parmi tous les codages, le VAE est le seul qui ne garantie pas la légalité de ses solutions. Ceci nécessite une routine de réajustement qui nécessite quant à elle un temps d'exécution supplémentaire.

6.12.4 Épistasie

Une faible épistasie (voir son absence totale) affecte négativement la qualité des solutions, ce qui est le cas pour les codages DVTC, SVTC, VAE. Aussi, un codage présentant une forte épistasie (le codage EA) produit des solutions médiocres. Entre ces deux extrémités, les codages PMEB, PMCA, FVTC, IC, DC, CGE possédant une épistasie modérée, présentent des solutions meilleurs.

6.13 Conclusion

Ce chapitre a fait l'objet d'une étude empirique comparative entre les codages présentés dans ce manuscrit. La comparaison des performances est basée sur une stratégie en deux parties. La première consiste à évaluer les performances de tous les codages en utilisant la même plateforme génétique avec les mêmes paramètres dans le but d'avoir la comparaison la plus équitable possible. Plusieurs métriques complémentaires ont été utiliser à savoir ART, AES, MBF, SDBF et à la fin la meilleur fitness (BF). Cette première étape de notre stratégie est quant à elle décomposer en deux étapes. La première partie est basée sur la comparaison des performances en prenant une instance représentative de chaque classe de graphe. Tandis que la deuxième partie, représente une comparaison globale des résultats. La deuxième phase de notre stratégie consiste à mettre chaque codage dans son environnement favorable avec le meilleur paramétrage qu'il soit. Les deux parties de notre stratégie est consolider par l'application d'un test statistique pour mieux mettre en avant les différences en terme de performances des différents codages. A cet effet, le test de Friedman et nemenyi ont été utilisé. L'analyse des différents résultats obtenu montre la supériorité des codages dites hybrides à savoir les codage PMEB et PMCA. Le codage FVTC montre aussi de bonnes performances.

Conclusion générale

Le champ d'étude de notre travail porte sur l'application des algorithmes génétiques pour la résolution du problème de partitionnement de graphe et l'impact de la représentation des solutions candidates sur les performances et le rendement de la méthode utilisée.
A des fins de lisibilité et de compréhension, nous avons décomposé notre travail en deux parties distinctes. La première représente un état de l'art dans lequel nous avons décrit en détail le domaine d'étude. Dans un premier temps nous avons décrit le problème de partitionnement de graphe avec ses différents domaines d'applications afin de montrer son importance pour la résolution des problèmes de la vie réelle. Par la suite, nous nous sommes penchés sur les méthodes de résolution de PPG. Deux grands axes ont été tracés pour définir ces méthodes. Le premier axe a fait l'objet d'une étude qui porte sur les méthodes fournissant des solutions optimales tels que les algorithmes gloutons, les méthodes spectrales, la croissance de région, etc. Ensuite les méthodes approchées ont eu leur part de description par la présentation des méthodes heuristiques, et les métaheuristiques. Un accent est mis sur les algorithmes génétiques qui font partie des méthodes évolutionnaires appartenant à leur tour à la catégorie des métaheuristiques en précisant leur mécanisme de fonctionnement et les différentes techniques utilisées pour améliorer leur rendement.
Dans le but de donner un aperçu théorique des codages, nous avons présenté une liste de propriétés génétiques. Cette dernière comporte la redondance, la cécité, la légalité, Lamarkian, la causalité et à la fin l'épistasie. Les concepts de solutions irréalisables et illégales sont également traités. Ensuite, nous avons décrit l'ensemble des codages que nous avons traités. Bien qu'elle ne soit pas une liste exhaustive, la liste des codages présents dans ce manuscrit énumère les codages les plus représentatifs et les plus utilisés dans le domaine de partitionnement de graphe. En plus de celles présentes dans la littérature, nous avons aussi proposé

trois nouvelles représentations notamment le PMEB, PMCA et CGE. D'autre codages représente des amélioration des versions existantes.

Les contributions proposées dans cette thèse peuvent être classifiées en deux parties : une partie théorique comportant la description des mécanismes des codages proposés. La deuxième partie porte sur l'implémentation des codages présentés avec une étude empirique comportant en soit une contribution au niveau des techniques utilisées et la philosophie de comparaison. Dans le but d'avoir une comparaison la plus équitable qu'elle soit nous avons procédé en deux étapes. Dans un premier temps, nous avons opté pour la même plateforme génétique c'est-à-dire les mêmes opérateurs génétiques et les mêmes paramètres. Pour cette première étape, nous avons procédé comme suit :
1. Classifier l'ensemble de jeu de données en trois classes distinctes par rapport à l'ordre des graphes, ensuite nous avons choisi une instance représentative pour chaque classe pour comparer les performances des codages par rapport à leurs résultats fournis pour l'instance en question.
2. Toujours avec les mêmes classes de graphes, mais cette fois-ci nous avons opté pour une analyse globale. En effet, nous avons traité les résultats de chaque classe en leur totalité avec une représentation graphique basée sur les boîtes à moustaches. Ensuite, nous avons étudié la distribution des résultats par rapport aux premier et au dernier quartile.

Pour les deux étapes précédentes nous avons utilisé cinq métriques différentes mais complémentaires à savoir ART, AES, MBF, SDBF et BF. En effet, la complexité du problème traité a fait en sorte que l'utilisation d'une seule métrique n'est pas suffisante pour mettre en relief les performances de chaque codage et accomplir ainsi une comparaison équitable.
En second lieu, nous avons utilisé le meilleur paramétrage de l'AG pour chaque codage afin de voir l'impact de ce paramétrage sur les performances et si les meilleurs codages de la première phase de comparaison ont eu cet avantage par rapport un paramétrage qui leur ait convenu ou par rapport à leurs mécanismes de représentations intrinsèques.

Pour les étapes précédentes nous avons utilisé le test de Friedman et Nemenyi pour faire surgir les différences entre les performances des codages. Les

résultats de ces tests statistiques ont approuvé les résultats de notre analyse. Ainsi les codages hybrides basés sur le problème de P-médians notamment les codages PMEB et PMCA ont montré une meilleure capacité à résoudre avec efficacité le problème de partitionnement de graphes. Le codage Fractionnel quant à lui a montré de bonnes performances. Les expériences que nous avons menées tout au long de ce travail nous ont permis d'étudier en détail les codages que nous avons présentés dans ce manuscrit et bien d'autres codages que nous avons implémentés et testés. En effet, pour commencer le codage binaire avec sa simplicité et son alphabet très restreint représente une variante très intéressante. La détermination du bon seuil des arêtes intra (inter) clusters peut conduire à des résultats impressionnants. Les expériences que nous avons réalisées nous ont montré que le seuil des arêtes intra clusters est en relation avec la densité, la distribution des poids des arêtes et le nombre d'arêtes de graphes. A cette effet, nous envisageant mettre en place une technique basée sur la logique floue pour la détermination de seuil. Une étude empirique de la combinaison des différents codages vus dans cette thèse à travers une approche évolutionnaire parallèle avec détermination des natures et périodes d'immigration entre les populations évoluant en parallèle en tenant en compte les différences structurelles entre ces représentations peut également aboutir à une amélioration importante des performances par rapport à la résolution du PPG. Ceci peut être accompagné par une heuristique pour la détermination des meilleurs paramètres pour chaque codage par rapport à chaque instance. Ensuite les AGs propres à chaque population peuvent être configurés séparément pour améliorer le rendement global de la méthode.

Bibliographie

Ahn, C. W. and Ramakrishna, R. S. (2003). Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 7(4) :367–385.

Andreev, K. and Racke, H. (2006). Balanced graph partitioning. *Theory of Computing Systems*, 39(6) :929–939.

Arabas, J., Michalewicz, Z., and Mulawka, J. (1994). Gavaps-a genetic algorithm with varying population size. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 73–78. IEEE.

Armbruster, M. (2007). Branch-and-cut for a semidefinite relaxation of large-scale minimum bisection problems. In *xxx*.

Back, T. (1994). Selective pressure in evolutionary algorithms : A characterization of selection mechanisms. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE World Congress on Computational Intelligence*, pages 57–62. IEEE.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications*, pages 101–111. Hillsdale, New Jersey.

Baños, R., Gil, C., Ortega, J., and Montoya, F. G. (2003). Multilevel heuristic algorithm for graph partitioning. In *Workshops on Applications of Evolutionary Computation*, pages 143–153. Springer.

Basavaprasad, B. and Hegadi, R. S. (2012). Graph theoretical approaches for image segmentation. *Aviskar–Solapur Univ Res J*, 2 :7–13.

Battiti, R. and Bertossi, A. A. (1997). Differential greedy for the 0-1 equicut problem. In *Network Design : Connectivity and Facilities Location*, pages 3–21.

Battiti, R. and Bertossi, A. A. (1999). Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE transactions on computers*, 48(4) :361–385.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2) :154–160.

Benlic, U. and Hao, J.-K. (2010). An effective multilevel memetic algorithm for balanced graph partitioning. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 121–128. IEEE.

Benlic, U. and Hao, J.-K. (2011a). An effective multilevel tabu search approach for balanced graph partitioning. *Computers & Operations Research*, 38(7) :1066–1075.

Benlic, U. and Hao, J.-K. (2011b). A multilevel memetic approach for improving graph k-partitions. *IEEE Transactions on Evolutionary Computation*, 15(5) :624–642.

Berthold, T., Gamrath, G., Gleixner, A. M., Heinz, S., Koch, T., and Shinano, Y. (2012). Solving mixed integer linear and nonlinear problems using the scip optimization suite. *ZIB-Report 12-27*, pages 1–23.

Bichot, C.-E. (2007). *Elaboration d'une nouvelle metaheuristique pour le partitionnement de graphe : la methode de fusion-fission. Application au decoupage de l'espace aerien*. PhD thesis, Institut National Polytechnique de Toulouse.

Bichot, C.-E. and Siarry, P. (2011). *Graph partitioning*. Wiley Online Library.

Biedermann, S., Henzinger, M., Schulz, C., and Schuster, B. (2018). Memetic graph clustering. *arXiv preprint arXiv :1802.07034*.

Boulif, M. (2016). A new cut-based genetic algorithm for graph partitioning applied to cell formation. *arXiv preprint arXiv :1612.05536*.

Boulif, M. (2019). Heterogeneous parallel genetic algorithm paradigm. *arXiv preprint arXiv :1905.06636*.

Boulif, M. and Atif, K. (2006). A new branch-&-bound-enhanced genetic algorithm for the manufacturing cell formation problem. *Computers & Operations Research*, 33(8) :2219–2245.

Brandfass, B., Alrutz, T., and Gerhold, T. (2013). Rank reordering for mpi communication optimization. *Computers & Fluids*, 80 :372–380.

Brunetta, L., Conforti, M., and Rinaldi, G. (1997). A branch-and-cut algorithm for the equicut problem. *Mathematical Programming*, 78(2) :243–263.

Bui, T. N. and Jones, C. (1993). A heuristic for reducing fill-in in sparse matrix factorization. Technical report, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA ....

Bulucc, A., Meyerhenke, H., Safro, I., Sanders, P., and Schulz, C. (2016). Recent advances in graph partitioning. *Algorithm Engineering Lecture Notes in Computer Science*, pages 117–158.

Chaouche, A. and Boulif, M. (2019). Solving the unsupervised graph partitioning problem with genetic algorithms : classical and new encoding representations. *Computers & Industrial Engineering*, page 106025.

Colorni, A., Dorigo, M., Maniezzo, V., et al. (1992). Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Cambridge, MA.

Cullum, J. and Willoughby, R. A. (1986). *Large scale eigenvalue problems*. Elsevier.

David A., B., Henning, M., Peter, S., and Dorothea, W. (2016). 10th dimacs implementation challenge - graph partitioning and graph clustering. http://www.cc.gatech.edu/dimacs10/index.shtml. Accessed April 4, 2016.

Davidor, Y. (1990). Epistasis variance : Suitability of a representation to genetic algorithms. *Complex Systems*, 4(4) :369–383.

De Jong, K. A. (1975). Analysis of the behavior of a class of genetic adaptive systems. Technical report.

Delling, D., Goldberg, A. V., Pajor, T., and Werneck, R. F. (2011). Customizable route planning. In *International Symposium on Experimental Algorithms*, pages 376–387. Springer.

Delling, D., Goldberg, A. V., Razenshteyn, I., and Werneck, R. F. (2012). Exact combinatorial branch-and-bound for graph bisection. In *2012 Proceedings of the Fourteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 30–44. SIAM.

Delling, D. and Werneck, R. F. (2013). Faster customization of road networks. In *International Symposium on Experimental Algorithms*, pages 30–42. Springer.

Derrac, J., Garcia, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(7) :3–18.

Diekmann, R., Luling, R., Monien, B., and Spraner, C. (1996). Combining helpful sets and parallel simulated annealing for the graph-partitioning problem. *International Journal of Parallel, Emergent and Distributed Systems*, 8(1) :61–84.

Doerr, B. and Zheng, W. (2020). From understanding genetic drift to a smart-restart parameter-less compact genetic algorithm. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 805–813.

Du, D.-Z., Pardalos, P. M., and Wu, W. (2009). *History of optimizationHistory of Optimization*. Springer US, Boston, MA.

Eiben, A. and Smith, J. (2015). *Introduction to evolutionary computing*. Springer, Berlin.

Eiben, Á. E., Hinterding, R., and Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on evolutionary computation*, 3(2) :124–141.

Eshelman, L. J. (1997). Crossover operator biases : Exploiting the population distribution. In *proc. of Int. Conf. on Genetic Algorithms, 1997*.

Even, G., Naor, J., Rao, S., and Schieber, B. (1999). Fast approximate graph partitioning algorithms. *SIAM Journal on Computing*, 28(6) :2187–2214.

Feldmann, A. E. and Widmayer, P. (2011). An o (n4̂) time algorithm to compute the bisection width of solid grid graphs. In *European Symposium on Algorithms*, pages 143–154. Springer.

Ferreira, C. E., Martin, A., de Souza, C. C., Weismantel, R., and Wolsey, L. A. (1998). The node capacitated graph partitioning problem : a computational study. *Mathematical programming*, 81(2) :229–256.

Fiduccia, C. M. and Mattheyses, R. M. (1982). A linear-time heuristic for improving network partitions. In *19th Design Automation Conference*, pages 175–181. IEEE.

FOGA (2019). Foundations of genetic algorithms. https://dl.acm.org/conference/foga. Accessed : 2020-11-07.

Forrest, S. (1985). Scaling fitnesses in the genetic algorithm. *Documentation for Prisoners Dilemma and Norms Programs That Use the Genetic Algorithm. Unpublished manuscript*.

Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3-5) :75–174.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32 :674–701.

Galinier, P., Boujbel, Z., and Fernandes, M. C. (2011). An efficient memetic algorithm for the graph partitioning problem. *Annals of Operations Research*, 191(1) :1–22.

Garey, M. R. and Johnson, D. S. (1979). *Computers and intractability : A Guide to the Theory of NP-Completeness*, volume 174. freeman San Francisco.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. (1974). Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63. ACM.

GECCO (2020). The genetic and evolutionary computation conference. https://gecco-2020.sigevo.org/index.html/HomePage. Accessed : 2020-11-07.

Gen, M. and Cheng, R. (1996a). *Genetic Algorithms and Manufacturing Systems Design*. John Wiley & Sons, Inc.

Gen, M. and Cheng, R. (1996b). A survey of penalty techniques in genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 804–809. IEEE.

Gil, C., Baños, R., Montoya, M., and Gómez, J. (2006). Performance of simulated annealing, tabu search, and evolutionary algorithms formulti-objective network partitioning. *Algorithmic Operations Research*, 1(1).

Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, 1(3) :190–206.

Glover, F. (1990). Tabu search—part ii. *ORSA Journal on computing*, 2(1) :4–32.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier.

Goldberg, D. E. et al. (1990). A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, 4(4) :445–460.

Goncalves, J. F. and Resende, M. G. (2002). A hybrid genetic algorithm for manufacturing cell formation. *AT&T Labs Research Technical Report TD-5FE6RN*, pages 1–30.

Grefenstette, J. J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on systems, man, and cybernetics*, 16(1) :122–128.

Guo, D. (2009). 12 multivariate spatial clustering and geovisualization. *Geographic Data Mining and Knowledge Discovery*, page 325.

Hager, W. W., Phan, D. T., and Zhang, H. (2013). An exact algorithm for graph partitioning. *Mathematical Programming*, 137(1-2) :531–556.

Hakimi, S. L. (1964). Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3) :450–459.

Hakimi, S. L. (1965). Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Operations Research*, 13(3) :462–475.

Harik, G. R. and Lobo, F. G. (1999). A parameter-less genetic algorithm. In *GECCO*, volume 99, pages 258–267.

Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., and Prasath, V. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10(12) :390.

Hendrickson, B. and Leland, R. (1995). An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2) :452–469.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems - An Introductory : Analysis with Applications to Biology, Control, and Artificial : Intelligence*. The University of Michigan Press.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. (1989). Optimization by simulated annealing : an experimental evaluation ; part i, graph partitioning. *Operations research*, 37(6) :865–892.

Junker, B. H. and Schreiber, F. (2011). *Analysis of biological networks*, volume 2. John Wiley & Sons.

Kahng, A. B., Lienig, J., Markov, I. L., and Hu, J. (2011). *VLSI physical design : from graph partitioning to timing closure*. Springer Science & Business Media.

Karisch, S. E., Rendl, F., and Clausen, J. (2000). Solving graph bisection problems with semidefinite programming. *INFORMS Journal on Computing*, 12(3) :177–191.

Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1) :359–392.

Kennedy, S. A. (1993). Five ways to a smarter genetic algorithm. *AI Expert*.

Kernighan, B. W. and Lin, S. (1970). An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2) :291–307.

Kieritz, T., Luxen, D., Sanders, P., and Vetter, C. (2010). Distributed time-dependent contraction hierarchies. In *International Symposium on Experimental Algorithms*, pages 83–93. Springer.

Land, A. H. and Doig, A. G. (2010). An automatic method for solving discrete programming problems. In *50 Years of Integer Programming 1958-2008*, pages 105–132. Springer.

Lauther, U. (2004). An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. *Geoinformation und Mobilitat-von der Forschung zur praktischen Anwendung*, 22 :219–230.

Li, H., Rosenwald, G. W., Jung, J., and Liu, C.-C. (2005). Strategic power infrastructure defense. *Proceedings of the IEEE*, 93(5) :918–933.

Li, J. and Liu, C.-C. (2009). Power system reconfiguration based on multilevel graph partitioning. In *PowerTech, 2009 IEEE Bucharest*, pages 1–5. IEEE.

Lin, Y.-C. and Hsiao, J.-W. (2004). A new approach to constructing optimal parallel prefix circuits with small depth. *Journal of Parallel and Distributed Computing*, 64(1) :97–107.

Lorena, L. A. N. and Furtado, J. C. (2001). Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, 9(3) :309–327.

Luxen, D. and Schieferdecker, D. (2012). Candidate sets for alternative routes in road networks. In *International Symposium on Experimental Algorithms*, pages 260–270. Springer.

Martin, O. C. and Otto, S. W. (1996). Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63(1) :57–75.

Mashohor, S., Evans, J. R., and Arslan, T. (2005). Elitist selection schemes for genetic algorithm based printed circuit board inspection system. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 974–978. IEEE.

Merchichi, S. and Boulif, M. (2015). Constraint-driven exact algorithm for the manufacturing cell formation problem. *European Journal of Industrial Engineering*, 9(6) :717–743.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6) :1087–1092.

Michalewicz, Z. (1995). A survey of constraint handling techniques in evolutionary computation methods. *Evolutionary programming*, 4 :135–155.

Michalewicz, Z. (2013). *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media.

Miller, B. L., Goldberg, D. E., et al. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3) :193–212.

Mohring, R. H., Schilling, H., Schutz, B., Wagner, D., and Willhalm, T. (2007). Partitioning graphs to speedup dijkstra's algorithm. *Journal of Experimental Algorithmics (JEA)*, 11 :2–8.

Mondaini, R. P. (2010). *Biomat 2009 : International Symposium on Mathematical and Computational Biology, Brasilia, Brazil, 1-6 August 2009*. World Scientific.

Murni, Bustamam, A., Ernastuti, Handhika, T., and Kerami, D. (2017). Hypergraph partitioning implementation for parallelizing matrix-vector multiplication using cuda gpu-based parallel computing. In *AIP Conference Proceedings*, volume 1862, page 030153. AIP Publishing LLC.

Nadi, F. and Khader, A. T. (2011). A parameter-less genetic algorithm with customized crossover and mutation operators. In *Proceedings of the 13th annual conference on genetic and evolutionary computation*, pages 901–908.

Nemenyi, P. (1963). Distribution-free multiple comparisons.

Newman, M. E. (2013). Community detection and graph partitioning. *EPL (Europhysics Letters)*, 103(2) :28003.

Orvosh, D. and Davis, L. (1994). Using a genetic algorithm to optimize problems with feasibility constraints. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 548–553. IEEE.

Osipov, V. and Sanders, P. (2010). n-level graph partitioning. In *European Symposium on Algorithms*, pages 278–289. Springer.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial optimization : algorithms and complexity*. Courier Corporation.

Peng, B., Zhang, L., and Zhang, D. (2013). A survey of graph theoretical approaches to image segmentation. *Pattern Recognition*, 46(3) :1020–1038.

PPSN (2020). International conference on parallel problem solving from nature. https://link.springer.com/conference/ppsn. Accessed : 2020-11-07.

Rechenberg, I. (1994). Evolution strategy. *Computational intelligence : Imitating life*.

Reynès, C. (2007). *Etude des Algorithmes genetiques et application aux donnees de proteomique*. PhD thesis, Universite Montpellier I.

Rolland, E., Pirkul, H., and Glover, F. (1996). Tabu search for graph partitioning. *Annals of operations research*, 63(2) :209–232.

Rosca, J. P. and Ballard, D. H. (1995). Causality in genetic programming. In *ICGA*, volume 95, pages 256–263.

Rothlauf, F. (2006). *Representations for genetic and evolutionary algorithms*. Springer-Verlag.

Schlierkamp-Voosen, D. (1993). Optimal interaction of mutation and crossover in the breeder genetic algorithm. In *International Conference on Genetic Algorithms ; Morgan Kaufmann Publishers Inc. : San Francisco, CA, USA*.

Schlierkamp-Voosen, D. and Muhlenbein, H. (1994). Strategy adaptation by competing subpopulations. In *International Conference on Parallel Problem Solving from Nature*, pages 199–208. Springer.

Schloegel, K., Karypis, G., and Kumar, V. (2000). *Graph partitioning for high performance scientific simulations*. Army High Performance Computing Research Center.

Schulz, C. (2016). Graph partitioning and graph clusteringin theory and practice. *Institute for Theoretical Informatics Karlsruhe Institute of Technology (KIT).–May*, 20 :24–187.

Schulz, F., Wagner, D., and Zaroliagis, C. (2002). Using multi-level graphs for timetable information in railway systems. In *Workshop on Algorithm Engineering and Experimentation*, pages 43–59. Springer.

Sellmann, M., Sensen, N., and Timajev, L. (2003). Multicommodity flow approximation used for exact graph partitioning. In *European Symposium on Algorithms*, pages 752–764. Springer.

Sensen, N. (2001). Lower bounds and exact algorithms for the graph partitioning problem using multicommodity flows. In *European Symposium on Algorithms*, pages 391–403. Springer.

Vecchi, M. P. and Kirkpatrick, S. (1983). Global wiring by simulated annealing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2(4) :215–222.

Venugopal, V. and Narendran, T. (1992). A genetic algorithm approach to the machine-component grouping problem with multiple objectives. *Computers & Industrial Engineering*, 22(4) :469–480.

Walshaw, C. (2008). Multilevel refinement for combinatorial optimisation : Boosting metaheuristic performance. In *Hybrid Metaheuristics*, pages 261–289. Springer.

Walshaw, C. (2016). The graph partitioning archive. http://chriswalshaw.co.uk/partition/. Accessed March 23, 2016.

Wang, L., Zhao, J., Wang, W., and Zhan, Z. (2015). Genetic algorithm for regionalization problem with adaptive equity constraint. In *2015 10th Asian Control Conference (ASCC)*, pages 1–6. IEEE.

Wegener, I. (1987). *The complexity of Boolean functions*. BG Teubner.

Wicks, E. M. and Reasor, R. J. (1999). Designing cellular manufacturing systems with dynamic part populations. *IIE Transactions*, 31(1) :11–20.

Williams, R. D. (1991). Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency : Practice and experience*, 3(5) :457–481.

Yang, S. (2007). Genetic algorithms with elitism-based immigrants for changing optimization problems. In *Workshops on Applications of Evolutionary Computation*, pages 627–636. Springer.

Yoon, Y. and Kim, Y.-H. (2014). Vertex ordering, clustering, and their application to graph partitioning. *Applied Mathematics & Information Sciences*, 8(1) :135–138.

Zhong, J., Hu, X., Zhang, J., and Gu, M. (2005). Comparison of performance between different selection strategies on simple genetic algorithms. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 2, pages 1115–1121. IEEE.