Optimisation de déploiement et localisation de cible dans les réseaux de capteurs

Matthieu Le Berre

Thèse
de doctorat
de l'UTT

Matthieu LE BERRE

Optimisation de déploiement et localisation de cible dans les réseaux de capteurs

Spécialité :
Optimisation et Sécurité des Systèmes

2014TROY0021

Année 2014

utt
université de technologie
Troyes

THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Matthieu LE BERRE

le 5 juin 2014

Optimisation de déploiement et localisation de cible dans les réseaux de capteurs

JURY

M. G. GELLE	PROFESSEUR DES UNIVERSITES	Président
M. F. HNAIEN	MAITRE DE CONFERENCES	Directeur de thèse
M. A. ROSSI	MAITRE DE CONFERENCS - HDR	Rapporteur
M. P. SIARRY	PROFESSEUR DES UNIVERSITES	Rapporteur
M. H. SNOUSSI	PROFESSEUR DES UNIVERSITES	Directeur de thèse

Remerciements

J'adresse mes remerciements aux personnes qui m'ont aidé dans la réalisation de ce travail. Tout d'abord, je remercie mes deux directeurs de thèse, Faicel Hnaien et Hichem Snoussi, pour m'avoir encadré au cours de ces trois années, pour leurs conseils et leur soutien dans les moments les plus difficiles, mais également pour leur présence à tout instant. Je tiens également à remercier Maher Rebai, qui a participé activement à la réalisation de cette thèse, et m'a apporté une aide précieuse au cours de ces trois années.

J'adresse également mes remerciements aux rapporteurs, M. André Rossi et M. Patrick Siarry, pour leur intérêt vis-à-vis de mon travail, et pour avoir minutieusement étudié ce manuscrit, ainsi qu'à tous les membres du jury.

Je remercie également tous mes amis doctorants, post-doctorants et permanents, pour leur soutien et leur bonne humeur, qui m'ont permis de tenir et de finaliser ce travail, et tout particulièrement Elyn Solano.

Ma reconnaissance va à ceux qui ont plus particulièrement assuré le soutien affectif de ce travail doctoral : ma famille ainsi que ma compagne Marion Levalet.

Optimisation de déploiement et localisation de cibles dans les réseaux de capteurs

Résumé

Au cours de cette thèse, nous avons abordé des problématiques liées à l'optimisation de déploiement et la localisation de cible dans les réseaux de capteurs. Nous avons tout d'abord proposé un premier modèle pour l'optimisation de deux objectifs contradictoires : le nombre de capteurs déployés ainsi que la précision de la localisation. Quatre algorithmes multiobjectif classiques ont été implémentés, et des versions hybrides ont également été proposées. Une variante du précédent problème est également étudiée, dédiée aux applications de localisation *indoor*. Les algorithmes proposés pour le premier problème n'ont montré qu'une efficacité relative au cours des premières expérimentations. Une nouvelle heuristique est alors développée, et les résultats ont montré de très bonnes performances sur les instances de taille réduite, ainsi que de bien meilleures performances que les autres algorithmes implémentés sur des instances de grande taille. Enfin, la notion de connectivité et de couverture est également traitée et intégrée dans un modèle linéaire de déploiement. Un algorithme de *Branch and Bound* a été développé afin de traiter ce problème, puis des tests ont été effectués afin de le comparer aux solveurs linéaires actuels.

Mots clefs : Optimisation combinatoire, Réseaux de capteurs (technologie), Algorithmes d'approximation, Algorithmes optimaux

Deployment optimization and target tracking in sensor networks

Abstract

In this thesis, a joint approach for deployment optimization and target tracking in sensor networks is developed. First, we have proposed a linear model to minimize the number of deployed sensors and maximize the accuracy of the localization. We have also implemented several multiobjective methods and proposed hybridization for some of them. We have also proposed a modification of the previous model, taking into account the indoor localization constraints. Two methods of the previous problem have been used, and a specific heuristic has been developed. Finally, two linear models taking into account coverage and connectivity have been proposed. A Branch and Bound algorithm has also been developed, considering a geometric lower bound and two properties to reduce the number of fathomed nodes.

Keywords : Combinatorial optimization, Sensor networks, Approximation algorithms, Exact methods

Table des matières

Chapitre 1

Introduction générale

Ces dernières années, les innovations technologiques dans la miniaturisation, la gestion de l'énergie et la communication sans fil ont permis l'avènement des réseaux de capteurs sans fil. Ceux-ci présentent un intérêt croissant dans un grand nombre d'applications et de domaines (Tseng et al., 2006) (Burrell et al., 2004), comme par exemple la surveillance de zones pour prévenir les intrusions et les vols, la surveillance de la faune dans un environnement naturel (Tseng et al., 2005), la prévention d'incendies dans les bâtiments, ... Les réseaux de capteurs sans fil présentent un grand nombre d'avantages, le principal étant la possibilité de déploiement dans des zones ne possédant aucun réseau d'électricité ni de communication. Ces réseaux sont composés d'un grand nombre de dispositifs communicants appelés capteurs. Chaque capteur comprend les parties suivantes : (i) le dispositif d'acquisition, c'est-à-dire un capteur adapté à l'application du réseau, (ii) le dispositif de communication permettant d'envoyer les données recueillies, (iii) un processeur permettant d'effectuer des traitements tels que la compression de données avant l'envoi et (iv) une batterie permettant le déploiement de capteurs dans des zones diverses. Cette composition met en évidence un grand nombre de problématiques quant à la mise en place et l'utilisation des réseaux de capteurs sans fil. Le principal objectif d'un réseau de capteurs est la conséquence directe de son utilisation, généralement la surveillance d'une zone. La surveillance implique la notion de couverture, qui correspond à la quantification de l'efficacité du réseau de capteurs. Dans la plupart des cas, la couverture est représentée par la proportion de la zone surveillée, ou couverte, par un ou plusieurs capteurs. Les problématiques liées à la couverture sont nombreuses, notamment les problématiques de déploiement mais également les problématiques d'ordonnancement d'activation. A la couverture vient parfois s'ajouter la contrainte de connectivité. En effet, selon la technologie choisie pour le dispositif de communication, l'envoi des données ne peut être garanti si la distance entre l'émetteur et le récepteur est trop grande. Cette contrainte supplémentaire sera surtout présente dans les applications de surveillance des zones de grande taille, ainsi que dans les déploiements de très faible densité. Enfin, l'énergie est une ressource importante dans les réseaux de capteurs sans fil. L'utilisation de batterie est souvent inévitable, les alternatives telles que les cellules photovoltaïques sont

généralement trop coûteuses ou inutiles dans certaines applications. La gestion de l'énergie entre généralement en conflit avec la couverture, et l'optimisation de ces deux objectifs contradictoires peut être réalisée via des algorithmes multiobjectif.

Dans cette thèse, nous nous intéressons à des problèmes d'optimisation mono et multiobjectif dans les réseaux de capteurs. Nous nous concentrons sur les problématiques de déploiement et de couverture, notamment pour des applications de localisation et de suivi de cible. La plupart des problèmes de déploiement peuvent se formaliser de la manière suivante : soit une zone à couvrir, le but est de minimiser le nombre de capteurs déployés sous la contrainte que chaque position soit couverte par un certain nombre de capteurs. Le nombre de capteurs nécessaires à la couverture d'une position diffère selon les exigences de l'application. Les positions sont généralement obtenues par discrétisation de la zone, de manière uniforme ou non.

Si la couverture d'une zone est une contrainte primordiale dans la plupart des applications des réseaux de capteurs, la précision de la localisation est un objectif critique dans un grand nombre de domaines tels que la détection d'évènements et d'intrusions. Si l'on prend l'exemple de la détection d'intrusion, une couverture totale de la zone permet de détecter une présence, mais maximiser la précision permet de minimiser les zones de recherche et ainsi diminuer le temps d'interception de l'intrus. L'objectif de cette thèse est de fournir des modèles mono et multiobjectif permettant de proposer des déploiements prenant en compte les coûts de déploiement ainsi que l'optimisation de la couverture et de la localisation. La localisation de cible fait généralement appel à la technique de triangulation. Étant donnée une zone en deux dimensions, la position d'une cible peut être connue en utilisant trois capteurs, en considérant qu'il est possible de définir la distance en fonction de la puissance du signal. Nous nous positionnons dans les cas où l'usage de signaux stables n'est pas possible, où l'utilisation de la technique de triangulation n'est pas possible.

1.1 Contributions

Dans cette thèse, nous proposons plusieurs modélisations multiobjectif prenant en compte les besoins de localisation dans les applications de suivi de cibles. Nous adoptons le long de cette thèse une approche intégrée originale qui consiste à résoudre le problème de déploiement (conception) en prenant en compte les contraintes de localisation (application). Les contributions majeures, classées par chapitre, sont énoncées ci-dessous :

— Dans le chapitre "Déploiement des capteurs et localisation de cibles", nous proposons un modèle mathématique multiobjectif prenant en compte la minimisation du coût de déploiement ainsi que la minimisation de l'imprécision lors de la localisation de cible. Quatre méthodes multiobjectif sont implémentées, et nous proposons différents opérateurs d'initialisation et de réparation pour traiter le problème. Les expérimentations sont faites sur quatre ensembles d'instances différents. Nous proposons également une hybridation des algorithmes NSGA-II, SPEA2 et MOPSO afin d'accroître leur compétitivité sur ce problème.

— Dans le chapitre "Localisation par zonage et déploiement de capteurs", nous proposons une modification du modèle précédent afin de traiter une problématique de localisation à l'intérieur des bâtiments. Les méthodes précédentes sont également adaptées à la nouvelle formulation du problème, et nous proposons également une heuristique se basant sur les méthodes spécialisées du *Set Covering Problem*.

— Dans le chapitre "Localisation outdoor et connectivité", nous présentons un modèle linéaire pour satisfaire à la fois couverture et connectivité, et nous définissons une borne inférieure en nous basant sur les propriétés géométriques du problème. Celle-ci est utilisée dans un algorithme par séparation et évaluation (*Branch and Bound*), qui sera comparé à la programmation linéaire.

1.2 Organisation

Le deuxième chapitre constitue un état de l'art des problématiques de couverture dans les réseaux de capteurs, dans lequel nous ferons également une présentation des divers travaux sur un problème NP-complet classique : le *Set Covering Problem*. Ce dernier est en effet étroitement lié au problème de minimisation du nombre de capteurs déployés sous la contrainte de couverture. Nous introduirons ensuite les notions d'optimisation multiobjectif et nous présenterons les méthodes utilisées par la suite dans nos travaux. Nous recenserons également les travaux traitant de l'optimisation multiobjectif pour les problèmes de déploiement. Le chapitre 3 présente une première problématique basée sur la localisation et le déploiement de réseaux de capteurs. Le chapitre suivant introduit une nouvelle modélisation dédiée à la localisation à l'intérieur de bâtiments. Le chapitre 5 présente un comparatif de deux approches pour la résolution exacte d'un problème de couverture et de connectivité. Enfin, le dernier chapitre conclut cette thèse et présente les perspectives faisant suite à nos travaux.

Chapitre 2

État de l'art de l'optimisation dans les réseaux de capteurs

2.1 Introduction

L'optimisation dans les réseaux de capteurs a été abordée dans de nombreux travaux, traitant de problématiques variées. Nous commencerons donc par détailler les différentes thématiques d'optimisation mono-objectif relatives aux réseaux de capteurs, ainsi que la modélisation de la couverture. Puis, nous effectuerons un état de l'art concernant les travaux traitant du *Set Covering Problem*, justifié par les similitudes avec la modélisation de la couverture choisie dans cette thèse. Ensuite, nous présenterons un aperçu de l'optimisation multiobjectif ainsi que les méthodes implémentées pour traiter les modèles des chapitres suivants. Enfin, nous détaillerons les différents travaux traitant de l'optimisation multiobjectif dans les réseaux de capteurs sans fil.

2.2 Problématiques d'optimisation dans les réseaux de capteurs

L'optimisation intervient dans de nombreuses problématiques des réseaux de capteurs sans fil : la gestion de la couverture de la zone à surveiller, la gestion de l'énergie dans le cas de capteurs autonomes, les contraintes de connectivité dans le cas d'applications extérieures, le *clustering* et la hiérarchisation du réseau pour les problématiques d'envoi de données, ... Nous commencerons par présenter les différents modèles de couverture utilisés dans la littérature, puis nous aborderons successivement les différents axes énoncés ci-dessus.

2.2.1 Modèles de couverture

La couverture assurée par un réseau de capteurs peut être assimilée à la qualité de service : c'est la fonction et le but principal du réseau. Il existe plusieurs manières de définir cette couverture, en

fonction de l'application attribuée au réseau de capteurs. En effet, on peut souhaiter une couverture globale d'une zone à observer ou encore n'exiger que l'observation d'un nombre défini de cibles. Dans tous les cas, la couverture est un objectif primordial dans les réseaux de capteurs et est souvent soit à maximiser, soit exprimée sous la forme de contraintes afin d'assurer une couverture totale.

Dans le cas où le problème de couverture consiste à minimiser le nombre de capteurs déployés tout en garantissant une couverture totale de la zone, plusieurs approches sont exploitées dans la littérature. La complexité du problème change selon la représentation de la zone. En effet, si la zone est représentée par une surface continue en deux dimensions, le problème se résout de manière géométrique, en exploitant les travaux de (Rourke, 1987) pour le problème de la galerie d'art. Par contre, dans le cas d'une représentation par un ensemble fini de positions à couvrir, le problème a été démontré comme étant NP-complet (Ke et al., 2011). Nous nous focaliserons sur cette dernière représentation du problème ("Critical Square Grid Coverage"), permettant une modélisation plus aisée des contraintes liées au déploiement de réseaux de capteurs. De nombreuses méthodes d'optimisation ont été adoptées pour résoudre des variations de ce problème (Ghosh and Das, 2008), (Zhu et al., 2011). On peut notamment citer (Andersen and Tirthapura, 2009), où les auteurs ont défini le problème de déploiement de capteurs dans une zone en trois dimensions comme un *Set Covering Problem*, en procédant à une discrétisation de la zone. On peut retrouver ce modèle dans un grand nombre de travaux, notamment (Altinel et al., 2008). Le cas d'une zone en trois dimensions est également abordé dans (Alam and Haas, 2006), où les auteurs utilisent des propriétés géométriques pour effectuer le déploiement des capteurs. Ces modèles ont été étendus au *K-coverage*, garantissant que chaque position soit couverte par au moins K capteurs ((Rebai et al., 2013), (Fusco and Gupta, 2009), (Wu et al., 2008), (Ammari and Das, 2006), (Mini et al., 2011), (Ammari and Das, 2006), (Mo et al., 2006), (Yun et al., 2010)). Le *K-coverage* a plusieurs finalités : l'application de la triangulation avec le *3-coverage*, l'extension de la durée de vie du réseau, etc.

La couverture d'une zone est elle-même exprimable sous la forme de plusieurs modèles (Ghosh and Das, 2008) : (1) le modèle binaire et (2) le modèle probabiliste. Le premier modèle est le plus simple à appliquer dans le domaine continu ou combinatoire. En considérant une aire à couvrir, tout intervenant à une distance inférieure ou égale à la portée d'acquisition d'un capteur sera alors détecté par celui-ci. Ce modèle est notamment utilisé dans (Liao et al., 2011), où l'auteur traite un problème de redéploiement des capteurs. Étant donné un ensemble de capteurs mobiles déployés aléatoirement sur la zone à couvrir, l'auteur utilise un algorithme GSO pour permettre la maximisation de la couverture de la zone tout en limitant les mouvements des capteurs (afin d'éviter les pertes énergétiques inutiles). Le second modèle est un dérivé du modèle binaire et intervient lui aussi dans le domaine continu. L'idée principale de ce modèle est qu'à partir d'une certaine distance un événement ne sera pas forcément détecté par le capteur. Ce modèle prend en compte la diminution du signal et de la précision des mesures avec la distance, se rapprochant ainsi de la réalité en prenant en compte les limites technologiques des composants des capteurs. On retrouve ce modèle dans (Wang and Wang, 2011), où l'auteur traite un problème de compromis

entre la couverture et la consommation d'énergie. Le cadre est similaire à celui utilisé dans (Liao et al., 2011), c'est-à-dire le cas où l'on procède à un déploiement aléatoire de capteurs mobiles. Le modèle énergétique utilisé prend uniquement en compte l'énergie utilisée lors des déplacements. L'optimisation de l'objectif se fera par une variante de l'algorithme PSO, prenant en compte des forces virtuelles dans le calcul des équations de trajectoire : les capteurs se repoussent mutuellement, et les obstacles repoussent les capteurs. En ce qui concerne la couverture, l'optimisation se fait en priorité sur les zones dites " intéressantes ".

2.2.2 Couverture et connectivité

De nombreuses applications nécessitent l'ajout de contraintes de connectivité aux modèles de couverture. En effet, couvrir une zone est inutile s'il n'est pas possible de transmettre les informations recueillies à l'utilisateur. Pour remédier à cela, une station est déployée dans la zone, permettant la connexion à un réseau global (Internet, satellites, ...), chaque capteur ayant l'obligation de transmettre les informations à cette station (Ammari and Das, 2012). Shakkottai et al. (2005) ont présenté les conditions nécessaires et suffisantes pour la couverture et la connectivité dans une grille, tout en développant un panel de méthodes afin de maintenir à la fois la connectivité et la couverture dans le réseau déployé. Bai et al. (2006) ont proposé une stratégie de déploiement optimal pour assurer à la fois la couverture et la 2-connectivité, en proposant également des schémas de déploiement améliorant la prise de décision. Liu et al. (2006) présentent un problème de planification d'activation de capteurs pré-déployés, pour assurer connectivité et couverture. Tian and Georganas (2005) ont démontré que si le réseau original est connecté, et que les nœuds choisis pour l'activation sont capables de couvrir la même région que les nœuds originaux, alors le réseau formé par ces nouveaux capteurs est connecté, à condition que le rayon de communication soit au moins deux fois plus grand que celui de couverture. Zhao and Gurusamy (2008) ont développé un algorithme glouton permettant de proposer une solution assurant à la fois couverture et connectivité. Ils ont également proposé une approche pour la planification du temps d'activation des capteurs. Dans (Gupta et al., 2006), les auteurs présentent la notion de couverture de capteurs connectés, qui est définie par l'ensemble des capteurs apportant une couverture totale de la zone, cet ensemble étant un graphe connecté. Les auteurs ont également prouvé que la minimisation du nombre de capteurs à déployer sous la contrainte de connectivité était un problème NP-complet. Zhou et al. (2004) ont également intégré la notion de connectivité dans un problème de *K-coverage*.

2.2.3 L'énergie dans les réseaux de capteurs

En ce qui concerne l'énergie, la consommation d'un capteur peut être influencée par différents facteurs : (1) la taille de la zone à couvrir par ce capteur, (2) la distance séparant ce capteur des autres pour l'envoi d'information et (3) le nombre de paquets de données transitant par ce capteur. Dans certains cas, le dispositif d'acquisition peut être réglé pour que sa portée soit plus ou moins

grande. Ce réglage aura un impact direct sur la consommation, l'énergie attribuée au dispositif étant différente. Pour ce qui est de la distance séparant le capteur du reste du réseau, l'envoi d'information nécessite une énergie non négligeable. Plus le récepteur est loin, plus l'énergie nécessaire pour lui envoyer des informations est importante. Enfin, le nombre et la taille des paquets de données influent également sur la consommation du capteur. De plus, dans le cas d'un déploiement aléatoire et dense, il n'est généralement pas nécessaire d'activer l'ensemble des capteurs en même temps (voir (Ting and Liao, 2010) et (Türkogullari and N. Aras, 2010)). Le réglage de dispositifs d'acquisition permet une utilisation plus ou moins intensive de la batterie d'un capteur (Cerulli et al., 2012). On retrouve notamment cette idée dans (Jia et al., 2009). Diminuer la portée du dispositif de communication permet d'accroître la durée de vie des capteurs, cependant elle exige généralement un déploiement plus dense afin de respecter la contrainte de connectivité tout en assurant la même couverture. Dans (Song et al., 2009), l'auteur se place dans le cas d'un déploiement aléatoire en couronnes concentriques, l'objectif étant de maximiser la durée de vie du réseau en ajustant la portée des dispositifs de communication. Les capteurs appartenant à une même couronne disposeront de la même portée de communication. La portée du dispositif de communication est prise en compte dans les modèles énergétiques d'un certain nombre de travaux. Dans (Rahman and Matin, 2011), un algorithme à essaim particulaire est utilisé pour trouver la position optimale du nœud statique afin d'accroître la durée de vie du réseau. Dans l'article (Lin et al., 2012), un algorithme de colonies de fourmis est utilisé pour diminuer la consommation d'énergie en agissant sur les chemins empruntés par les données dans le réseau. Le modèle énergétique prend en compte la taille des données ainsi que la distance de transmission. Rossi et al. (2012) ont proposé une génération de colonnes pour résoudre deux variantes d'un problème de conservation d'énergie dans la couverture de cibles. La première variante considère que les rayons de détection des capteurs peuvent varier au cours du temps, et sont ajustables en fonction de la situation. Les valeurs de ces rayons sont incluses dans un ensemble fini de valeurs. La deuxième variante change le domaine de définition de ces rayons, prenant alors leur valeur dans un ensemble continu. Le problème général est un problème d'activation et désactivation de capteurs, et est résolu de manière exacte par une génération de colonne couplée à un algorithme génétique, servant à l'optimisation du sous-problème. La même approche de résolution est utilisée dans (Rossi et al., 2013), mais pour deux variantes d'un problème sensiblement différent du précédent. Les auteurs se concentrent ici sur un problème d'activation et désactivation de capteurs non omnidirectionnels, tels que des caméras, la première variante considérant que l'orientation d'un capteur comme une variable entière alors que la deuxième variante autorise de fixer la valeur de l'orientation dans un ensemble continu de valeurs. La notion de *clustering* est extrêmement présente dans un grand nombre de travaux portant sur les réseaux de capteurs sans fil, notamment dans la problématique de conservation d'énergie. L'organisation en *clusters* a de multiples avantages. L'un d'eux est d'éviter la duplication de données dans le réseau. Par exemple, si deux capteurs détectent le même événement, ils enverront la même information qui se propagera en double dans le réseau et provoquera une utilisation de l'énergie accrue. L'organisation en *clusters* permet de réduire ce risque. En effet, la probabilité que ces deux capteurs appartiennent au même *cluster* est assez élevée due à

leur proximité, et le chef de *cluster* se chargera de détecter les redondances et de ne pas les diffuser dans le reste du réseau. Slama et al. (2007) proposent un système permettant d'optimiser : (1) la durée de vie de chaque *cluster* en agissant sur le planning des communications, et généralisant ce problème en un problème de *flow graph* pour la résolution et (2) optimiser les communications entre les chefs de *cluster* ainsi que le traitement des données, permettant ainsi de maximiser la durée de vie du réseau. Le modèle énergétique prend en compte l'énergie dépensée dans les communications ainsi que dans le traitement des données par les processeurs. Liu et al. (2012) ont traité un problème combinant la minimisation de l'intersection de la couverture des *clusters* ainsi que la minimisation du nombre de chefs de *cluster*. Il s'agit ici de couverture de capteurs, c'est-à-dire de connectivité. Le but de ce travail est de minimiser les interférences, améliorant ainsi la stabilité du réseau ainsi que le fonctionnement de certains algorithmes tels que les algorithmes de routage et d'agrégation de données. L'utilisation de cet algorithme de résolution permet également une augmentation de la durée de vie du réseau.

2.2.4 Couverture et localisation

La localisation est un des objectifs les plus évidents de la couverture d'une zone par un réseau de capteurs. En effet, détecter une cible ou un intrus est certes utile, mais pouvoir le localiser permet de guider sa recherche. Chakrabarty et al. (2001) et Chakrabarty et al. (2002) définissent un nouveau problème d'identification de cible dans une grille. En se basant sur un problème de couverture classique en deux et trois dimensions, ils cherchent à minimiser le coût de déploiement de deux types de capteurs, chaque type ayant un coût et une portée différente. Les auteurs présentent un modèle linéaire de couverture semblable à celui vu précédemment, et proposent une linéarisation des distances entre les positions. Les auteurs définissent également le problème de séquences, chaque position devant être couverte par une séquence unique maximale de capteurs. Cette partie du problème est associée à un problème de couverture de nœuds dans un graphe, prouvé comme étant NP-complet (Rao, 1993). Cette approche de localisation est reprise dans (Dhillon and Chakrabarty, 2003), où les auteurs adaptent l'idée des séquences dans un problème de déploiement pour les situations d'urgence. Un algorithme est proposé afin de tenir compte de la robustesse du réseau en cas de catastrophes, anticipant alors les effets d'effondrements de murs ou de pertes de nœuds dans le réseau.

2.3 *Set Covering Problem*

Le *Set Covering Problem* (SCP) est un problème NP-complet classique, un des 21 problèmes de référence décrits dans (Karp, 1972), et démontré comme étant NP-complet dans (Garey and Johnson, 1979). Il s'agit d'un simple problème d'activation de capteurs sous contrainte de couverture de cibles. Soit un ensemble de cibles à surveiller, et un ensemble de capteurs activables, l'objectif

est de trouver le coût minimal d'activation de capteurs en assurant la couverture de chaque cible. Il est formulé de la manière suivante (Roth, 1969) :

$$Minimiser\ z = \sum_{i \in S} c_i X_i \tag{2.1}$$

$$S.c. :$$

$$\forall j \in T, \sum_{i \in S} A_{i,j} X_i \geq 1 \tag{2.2}$$

$$\forall i \in S, X_i \in \{0, 1\} \tag{2.3}$$

Où T et S représentent respectivement l'ensemble des cibles à couvrir et l'ensemble des capteurs disponibles. La variable X_i représente l'état d'activation du capteur i et c_i son coût d'activation. La matrice A est la matrice de relation de couverture entre chaque couple de capteurs et cibles. En d'autres termes, $A_{i,j}$ est égal à 1 si et seulement si le capteur i peut couvrir la cible j.

Ce problème est utilisé dans un nombre très important de travaux, que ce soit dans les réseaux de capteurs et problèmes de couverture, où encore dans les problèmes de tournées de véhicules. Il existe de nombreuses variantes, nous citerons les suivantes :

— Le SCP dit *unicost* ou encore *Location Set Covering Problem*, où chaque valeur c_i est égale à 1. Ce problème est en pratique plus difficile à optimiser que le SCP classique, le manque d'information sur l'impact du choix des variables pénalise la plupart des heuristiques et métaheuristiques dédiées à ce problème.

— Le SCP est parfois écrit avec des contraintes de la forme $\sum_{i \in S} A_{i,j} X_i \geq B_j$, où B_j est le nombre d'observateurs nécessaires pour surveiller la cible j. Cette version est souvent utilisée pour les problèmes de déploiement de réseaux de capteurs, notamment pour le *K-coverage* et *Q-coverage*. Si tous les B_j sont de même valeur, le SCP est parfois appelé *Set k-Covering Problem*.

Dans la partie qui suit, nous tâcherons de faire un état de l'art le plus exhaustif possible concernant l'optimisation de *Set Covering Problem*. Nous ne nous intéresserons qu'aux articles dédiés à ce problème, bien qu'il soit probable qu'un grand nombre de travaux exploitent ce problème dans le cadre d'applications spécifiques.

Les méthodes d'optimisation proposées pour ce problème sont nombreuses, étant donné qu'il est traité depuis une quarantaine d'années. Nous choisirons de diviser cet état de l'art de la manière suivante : (1) les méthodes exactes et bornes inférieures (*Branch and Bound*, *Branch and Cut*, ...), (2) les différentes heuristiques proposées pour ce problème (algorithmes gloutons, GRASP, ...) et enfin (3) les adaptations de métaheuristiques (algorithmes évolutionnaires et recherches locales).

2.3.1 Propriétés

Un grand nombre de travaux sur le *Set Covering Problem* utilisent des propriétés de réduction du problème (Beslay, 1987) (Caprara et al., 2000) (Fisher and Kedia, 1990), afin de réduire les nombres de lignes et de colonnes, et ainsi faciliter l'optimisation.

1. Si pour une ligne $j \in T$, la somme $\sum_{i \in S} A_{i,j}$ est nulle, alors le problème n'est pas réalisable. Dans le cas contraire, il existe au moins une solution au problème.
2. Si pour une ligne $j \in T$, il n'existe qu'une seule variable X_i telle que $A_{i,j} = 1$, alors il est nécessaire de fixer cette colonne à 1.
3. Soit un couple de lignes $(j, k) \in T^2, j \neq k$. Si pour chaque $A_{i,j}$ non nul, le coefficient $A_{i,k}$ est également non nul, alors la contrainte k est absorbée par la contrainte j, la contrainte k étant automatiquement satisfaite lors de la satisfaction de la contrainte j.
4. Soit un couple de colonnes $(i, l) \in S^2, i \neq l$. Si pour chaque $A_{i,j}$ non nul, le coefficient $A_{l,j}$ est également non nul, alors la colonne l absorbe la colonne i, cette dernière pouvant être fixée à 0.

2.3.2 Méthodes exactes

En plus du modèle linéaire utilisable avec les solveurs actuels tels que Gurobi, Ilog Cplex ou GLPK, de nombreuses approches exactes ont été étudiées pour ce problème. Un algorithme de *Branch and Bound* a été développé dans (Leigh et al., 1988). Les auteurs effectuent une transformation préliminaire du problème en un *Set Partitioning Problem*, afin que les contraintes soient exprimées sous la forme d'égalités strictes. Chaque ligne doit être couverte par au moins une colonne. La borne supérieure d'un nœud de l'arbre de recherche est calculée par une heuristique gloutonne. Beslay (1987) a également proposé un algorithme de *Branch and Bound*, basé sur la relaxation linéaire et la relaxation lagrangienne. Dans (Balas and Carrera, 1996), les auteurs reprennent le principe de la relaxation lagrangienne et de l'algorithme de sous-gradient afin de guider un algorithme de *Branch and Bound*. Les études réalisées dans (Caprara et al., 2000) ont montré que les solveurs linéaires tels que Ilog Cplex étaient les plus performants pour la résolution exacte du *Set Covering Problem*.

2.3.3 Relaxations et bornes inférieures

Umetani and Yagiura (2007) présentent les différentes méthodes de relaxation du *Set Covering Problem*. Celles-ci permettent d'obtenir (i) une borne inférieure au problème et (ii) des informations supplémentaires sur l'affectation des variables. La première est la relaxation linéaire classique, consistant à éliminer les contraintes sur les domaines des variables. Elle est généralement optimisée par l'algorithme du simplexe, incorporé dans les solveurs linéaires actuels. La deuxième borne inférieure présentée est celle de la relaxation lagrangienne. Le principe est le suivant : on choisit les

contraintes du problème à relaxer, et on les pénalise dans l'objectif, à chaque contrainte $j \in T$ est attribué un coefficient de pénalisation u_j. La relaxation lagrangienne du *Set Covering Problem* est la suivante :

$$Minimiser\ z_{LR}(u) = \sum_{i \in S} c_i X_i + \sum_{j \in T} u_j (1 - \sum_{i \in S} A_{i,j} X_i) \tag{2.4}$$

$$S.c.\ :$$

$$\forall i \in S, X_i \in \{0,1\} \tag{2.3}$$

Où pour un vecteur u donné, l'objectif est de minimiser la valeur de l'objectif en attribuant les valeurs aux variables X_i du problème. L'objectif peut également être écrit sous une forme plus simple :

$$Minimiser\ z_{LR}(u) = \sum_{j \in T} u_j + \sum_{i \in S} [(c_i - \sum_{j \in T} A_{i,j} u_j) X_i] \tag{2.4}$$

$$S.c.\ :$$

$$\forall i \in S, X_i \in \{0,1\} \tag{2.3}$$

Cette forme permet d'obtenir la valeur optimale de $z_{LR}(u)$ en $O(|S|)$. En effet, l'objectif est défini par deux composantes : $\sum_{j \in T} u_j$ est une constante en fonction de u, l'affectation des variables n'a pas d'impact sur cette partie de l'objectif. On remarque que la deuxième partie est la somme des variables du problème, chaque variable étant pondérée par un coût $c_i - \sum_{j \in T} A_{i,j} u_j$. Afin de minimiser $z_{LR}(u)$, il suffit de fixer les variables de la manière suivante :

$$X_i = \begin{cases} 1 & \text{si } c_i - \sum_{j \in T} a_{i,j} u_j \leq 0 \\ 0 & \text{sinon} \end{cases}$$

Cette procédure peut être appelée de manière itérative au sein d'un algorithme de sous-gradient, qui cherchera à maximiser la borne inférieure en résolvant le problème précédent avec des valeurs u_j différentes. Une troisième borne inférieure est également présentée, appelée *surrogate relaxation* (SR). Le principe de cette borne consiste en la réunion de toutes les contraintes de couverture du modèle en une seule, sous la forme d'une somme pondérée par un vecteur w.

$$Minimiser\ z_{SR}(w) = \sum_{i \in S} c_i X_i \tag{2.4}$$

$$S.c.\ :$$

$$\sum_{j \in T} w_j (\sum_{i \in S} A_{i,j} X_i) \geq \sum_{j \in T} w_j \tag{2.5}$$

$$\forall i \in S, X_i \in \{0,1\} \tag{2.3}$$

Ces bornes ont été utilisées dans la littérature afin de concevoir des heuristiques spécialisées au *Set Covering Problem*, ces travaux seront présentés dans la partie suivante.

2.3.4 Heuristiques

La présentation des heuristiques se fera dans l'ordre suivant : tout d'abord les heuristiques de construction basées sur l'algorithme glouton, puis les heuristiques basées sur les différentes relaxations abordées dans la partie précédente et enfin les heuristiques plus modernes telles que GRASP.

2.3.4.1 Algorithme glouton et variantes

La première heuristique que nous présenterons ici est l'algorithme glouton (GH), présenté la première fois dans (Chvatal, 1979) (voir l'Algorithme 2.1). Cet algorithme permet de construire rapidement une solution au problème SCP *unicost* (tous les coûts de déploiement sont égaux).

Algorithme 2.1 Algorithme glouton

Paramètres S, T, A

X *as* $\forall i \in S, X_i = 0$ Tous les capteurs sont désactivés.

$NbSat \leftarrow 0$ Aucune contrainte n'est satisfaite.

$\forall i \in S, score_i \leftarrow \sum_{j \in T} A_{i,j}$ Le score de chaque capteur correspond au nombre de contraintes satisfaites par son activation.

Tant que $NbSat < |T|$ **Faire**

$Choice \leftarrow i \in S$ *as* $\nexists j \in S, j \neq i, score_i < score_j$

$X_{Choice} \leftarrow 1$ On active le capteur ayant le score le plus élevé.

$NbSat \leftarrow NbSat + score_{Choice}$ On met à jour le nombre de contraintes satisfaites.

Pour $j = 1$ to $|T|$ **Faire**

Si $A_{Choice,j} = 1$ **Alors**

$\forall i \in P|, score_i \leftarrow score_i - A_{i,j}$ On met à jour les scores des capteurs.

Fin Si

Fin Pour

Fin Tant que

Retourne X

Le processus est simple : tant qu'il reste des cibles à couvrir, on active le capteur couvrant le plus de cibles non couvertes. En cas de score identique entre plusieurs variables, l'algorithme choisit celle qui a l'indice le plus faible. Les performances de cette heuristique sont relativement modestes, mais elle permet d'obtenir une solution de qualité acceptable en un temps relativement faible. De nombreuses variantes de cet algorithme ont également été développées.

Dans (Grossman and Wool, 1997), les auteurs comparent neuf algorithmes dédiés au SCP, dont quatre différentes versions de l'algorithme glouton. Mise à part la version originale, on retrouve l'algorithme R-Gr (*Randomized greedy algorithm*) : une version randomisée de l'algorithme glouton, dont la seule différence est de choisir de manière aléatoire le capteur à activer en cas d'égalité. L'algorithme T-Gr (*Tresh greedy algorithm*) reprend le schéma principal de l'algorithme glouton,

cependant il se réfère au résultat de la relaxation linéaire pour construire la liste des variables à considérer, de la même façon que l'algorithme de Hochbaum (ou Tresh) (Hochbaum, 1982). Enfin, l'algorithme Alt-Gr (*Alternative greedy algorithm*) fonctionne en deux phases : à chaque itération, il commence par ajouter la variable ayant le score le plus élevé Δ, puis retire successivement des variables de la solution tant que l'impact est inférieur à Δ. En plus des variantes de l'algorithme glouton, les auteurs prennent en compte l'algorithme NoLP (Gavril, 1974), qui inspecte les contraintes une à une, et quand l'une d'elles n'est pas satisfaite, il ajoute toutes les variables satisfaisant cette contrainte à la solution. Les algorithmes RR (*Randomized rounding*) et SortLP sont basés sur la relaxation linéaire et seront décrits dans la partie suivante. Enfin, un algorithme à base de réseau de neurones est également présenté. Les tests ont été effectués sur un grand nombre d'instances aléatoires et combinatoires (ces dernières ont été générées en fonction d'un problème classique d'optimisation). Les résultats montrent que les algorithmes basés sur l'algorithme glouton sont meilleurs que les méthodes NoLP, Tresh et RR. Les réseaux de neurones offrent également de bonnes performances. Cependant, même si les scores sont très proches, l'algorithme R-Gr est clairement meilleur sur la plupart des instances.

Deux algorithmes gloutons sont proposés dans (Almiñana and Pastor, 1994) : FMC (voir l'Algorithme 2.2) et CMA. La principale différence par rapport au premier type d'algorithme glouton est qu'ici les méthodes ne se basent pas directement sur les colonnes mais sur les lignes. A chaque itération l'algorithme va rechercher la ou les lignes les plus difficiles à satisfaire (i.e. celles ayant la somme des $A_{i,j}$ la plus faible). Un fois cet ensemble déterminé, l'algorithme va établir la liste des variables éligibles, c'est-à-dire celles satisfaisant au moins une des contraintes de l'ensemble. La variable inclue dans la solution sera celle possédant le meilleur score de satisfabilité en considérant toutes les contraintes non satisfaites du problème. Les tests ont été réalisés sur des instances de petite taille, et les algorithmes FMC et CMA ont procuré de meilleures performances que les algorithmes gloutons classiques.

2.3.4.2 Heuristiques basées sur la relaxation

Dans la partie précédente, nous avons présenté l'algorithme T-Gr basé sur la relaxation linéaire. Nous avons également cité l'algorithme SortLP, l'algorithme de Hochbaum, et enfin l'algorithme RR. Le premier consiste simplement à affecter les variables ayant la valeur de relaxation linéaire la plus élevée, jusqu'à ce que la solution soit réalisable. L'algorithme de Hochbaum ajoute toutes les variables ayant une valeur de relaxation linéaire supérieure à un certain seuil, calculée en fonction de la contrainte ayant le plus grand nombre de coefficients $A_{i,j}$ de valeur 1. Enfin, l'algorithme RR génère de façon aléatoire des solutions pendant un certain nombre d'itérations. Les variables sont choisies en fonction du résultat de la relaxation linéaire multiplié par un coefficient d'échelle, le tout définissant la probabilité d'insertion de la variable dans la solution. Seules les solutions réalisables sont mises en mémoire. Les performances de ces algorithmes sont inférieures à celles des variantes

Algorithme 2.2 Algorithme FMC

Paramètres S, T, A

X *as* $\forall i \in S, X_i \leftarrow 0$ Tous les capteurs sont désactivés.

$NbSat \leftarrow 0$ Aucune contrainte n'est satisfaite.

$\forall i \in S, score_i^{col} \leftarrow \sum_{j \in T} A_{i,j}$ Le score de chaque capteur correspond au nombre de contraintes satisfaites par son activation.

$\forall j \in T, score_i^{ligne} \leftarrow \sum_{i \in S} A_{i,j}$ Le score de chaque contrainte correspond au nombre de capteurs satisfaisant la contrainte par leur activation.

Pour tout $j \in T$ **Faire**

Si $score_j^{ligne} = 1$ **Alors**

$X_i \leftarrow 1$ on active le seul capteur i satisfaisant j

$NbSat \leftarrow NbSat + 1$

Fin Si

Fin Pour

Si $NbSat = |T|$ **Alors**

Retourne X La solution est optimale

Sinon

Tant que $NbSat < |T|$ **Faire**

$E \leftarrow \{j \in T \text{ as } \nexists k \in T, score_j^{ligne} > score_k^{ligne}\}$

$E' \leftarrow \{i \in S \text{ as } \exists j \in E, A_{i,j} = 1\}$

$Choice \leftarrow i \in E' \text{ as } \nexists j \in E', j \neq i, score_i < score_j$

$X_{Choice} \leftarrow 1$ On active le capteur ayant le score le plus élevé.

$NbSat \leftarrow NbSat + score_{Choice}$ On met à jour le nombre de contraintes satisfaites.

Pour $j = 1$ to $|E|$ **Faire**

Si $A_{Choice,j} = 1$ **Alors**

$\forall i \in P, score_i \leftarrow score_i - A_{i,j}$ On met à jour les scores des capteurs.

Fin Si

Fin Pour

Fin Tant que

Retourne X

Fin Si

de l'algorithme glouton.

Dans (Beslay, 1990), les auteurs proposent une heuristique basée sur la relaxation lagrangienne, qui se montre compétitive sur un grand nombre d'instances. Cette approche est réutilisée dans (Haddadi, 1997), où l'auteur développe l'heuristique LHSCP basée sur la relaxation lagrangienne, l'algorithme de sous gradient et l'algorithme glouton. Le processus est le suivant : à chaque itération de l'algorithme de sous-gradient, la solution optimale du sous problème est calculée puis corrigée afin de satisfaire le problème SCP original. Les performances sont comparables voire meilleures que celles de l'algorithme de (Beslay, 1990). L'auteur précise cependant que son heuristique est destinée à des problèmes de faible densité (où le nombre de variables est bien plus important que le nombre de contraintes). On peut retrouver d'autres heuristiques basées sur la relaxation lagrangienne dans la littérature, notamment (Caprara et al., 1999) et (Ceria et al., 1998), qui sont également des combinaisons de l'algorithme de sous-gradients et de l'algorithme glouton. Les principales différences sont l'apparition de procédures pour fixer les variables lors des itérations (i.e. similaires à la méthode Ballas) dans (Ceria et al., 1998), et également une technique de *pricing* permettant de réduire le temps d'exécution de la méthode (Caprara et al., 1999). En se basant sur le fait que la relaxation *surrogate* est de qualité supérieure ou égale à la relaxation lagrangienne pour le SCP (Parker and Rardin, 1988), Almiñana and Pastor (1997) ont développé une heuristique RS reprenant le principe de celles citées précédemment. Cependant, les auteurs exploitent la relaxation surrogate et des propriétés mathématiques permettant aux itérations un enchaînement rapide. Les heuristiques utilisées pour l'initialisation de la borne supérieure et la correction de la solution pour le SCP sont les algorithmes FMC et CMA. Les auteurs se comparent à une heuristique exploitant elle aussi la relaxation *surrogate* SH (Lopes and Lorena, 1994), et montrent que la méthode RS est plus rapide et plus efficace sur la majorité des instances testées. Les auteurs se comparent également aux heuristiques FMC et CMA qui, bien que plus rapides, s'avèrent moins performantes que l'algorithme RS.

2.3.4.3 Heuristiques constructives et itératives

Nous aborderons ici des heuristiques plus modernes, généralement plus proches des métaheuristiques que des heuristiques spécialisées. Bautista and Pereira (2007) sont les premiers à adapter l'algorithme GRASP au problème SCP. Les auteurs ont tout d'abord exploité la transformation du SCP en problème MAXSAT, pour ensuite optimiser ce dernier par l'algorithme GRASP. Cette méthode se décompose en deux phases : la première phase consiste en une construction casualisée d'une configuration réalisable pour le problème MAXSAT, puis une recherche locale WALKSAT est utilisée pour optimiser cette solution. Les performances de l'algorithme sont ensuite comparées à celles de l'algorithme R-Gr. L'algorithme GRASP est repris dans (Pessoa et al., 2010), où les auteurs adaptent cette méthode au problème de *Set k-Covering Problem*. La première étape est la construction de la solution, effectuée par un algorithme glouton casualisé. Puis la solution est améliorée par

une recherche locale prenant en compte deux voisinages : la suppression de variables inutiles (1-0), et le déplacement (1-1) se basant sur la réduction du coût de la fonction objectif. La troisième procédure est le *Path-relinking* : en considérant les couples (x^s, x^t) extraits d'une archive de solutions de bonne qualité, le *Path-relinking* explore les solutions intermédiaires de la transformation de x^s en x^t. Enfin, l'hybridation avec une heuristique lagrangienne est réalisée, afin d'obtenir une solution initiale de bonne qualité, l'algorithme résultant est appelé LAGRASP. Plusieurs algorithmes dont des heuristiques lagrangiennes sont utilisés dans les expérimentations, et la méthode LAGRASP offre les meilleures performances sur les instances testées.

La méthode Meta-RaPS (DePuy et al., 2002) est basée sur un principe similaire à celui de l'algorithme GRASP : à chaque itération, la méthode construit une solution au problème, puis l'optimise. Dans (Lan and DePuy, 2006) et (Lan et al., 2007), les auteurs proposent une adaptation de cette méthode au SCP. La construction de la solution est réalisée par une procédure gloutonne casualisée. La procédure d'amélioration cherche tout d'abord à réduire aléatoirement le nombre de colonnes du problème, puis applique l'heuristique de construction au sous-problème résultant. Les auteurs mettent également l'accent sur deux mesures destinées à améliorer la qualité des solutions : la casualisation dans le choix des variables à ajouter ainsi que la pénalisation des variables les moins utiles. Les auteurs utilisent également des procédures de réduction de problèmes proposées par (Beslay, 1987), afin de réduire les temps de calcul de l'algorithme. La méthode Meta-RaPS a été comparée à un grand nombre d'heuristiques et de métaheuristiques dédiées au SCP sur des instances classiques et *unicost*. Les résultats montrent de bonnes performances sur les instances non *unicosts*, mais également que la méthode domine tous les autres algorithmes implémentés sur les instances *unicosts*.

2.3.5 Métaheuristiques

On peut trouver dans la littérature un grand nombre de travaux dédiés à l'adaptation de métaheuristiques au problème SCP. L'une des plus populaires est l'algorithme génétique proposé initialement par (Holland, 1975). La procédure générale d'un algorithme génétique est décrite dans l'Algorithme 2.3, où X_{best} représente la meilleure solution trouvée par l'algorithme, P_i la population à l'itération i, $Taille_{pop}$ et $Taille_{sel}$ sont respectivement la taille de la population et le nombre d'individus sélectionnés pour la reproduction. Les probabilités p_c et p_m sont utilisées dans les opérateurs de croisement et de mutation. La première étape est l'initialisation : l'algorithme va générer (de manière aléatoire ou guidée) un ensemble de solutions diversifié. Puis à chaque itération, l'algorithme va sélectionner un certain nombre d'individus dans la population en cours pour procéder à la reproduction, qui aboutira à la génération de nouvelles solutions. Enfin, la sélection pour la survie permet de maintenir une population de taille constante, en se basant sur les valeurs de fonction objectif des individus. Dans (Beslay and Chu, 1996), les auteurs proposent un algorithme génétique adapté pour les problèmes SCP non *unicosts*. La représentation des solutions choisie est la représen-

Algorithme 2.3 Algorithme génétique

Paramètres $Taille_{pop}, Taille_{sel}, p_c, p_m, Iter_{max}$

$P_0 = initialisation(Taille_{pop})$
$X_{best} = argmin(f(X), X \in P_0)$
Pour $i = 0$ to $Iter_{max}$ **Faire**
$selectionReproduction(P_i)$
$Q_i = croisement + mutation(P_i)$
$X'_{best} = argmin(f(X), X \in P_0)$
Si $f(X_{best}) > f(X'_{best})$ **Alors**
$X_{best} = X'_{best}$
Fin Si
$P_{i+1} = selectionSurvie(P_i + Q_i)$
Fin Pour
Retourne X_best

tation binaire. L'initialisation se fait de manière aléatoire. La sélection, l'opérateur de croisement et l'opérateur de mutation choisis sont respectivement le tournoi binaire, l'opérateur de fusion (un croisement uniforme probabiliste) et l'opérateur d'inversion de bits. Une heuristique gloutonne est utilisée pour corriger les nouvelles solutions. Parmi les paramètres décrits dans l'article, les auteurs soulignent le fait que la probabilité de mutation est variable. Les tests sont réalisés sur des instances de la littérature, et l'algorithme génétique présenté améliore la plupart des meilleurs résultats de l'époque. Dans (Solar et al., 2002), les auteurs proposent un algorithme génétique parallélisé (PGA) qu'ils comparent à l'algorithme génétique simple, à la recherche tabou ainsi qu'au recuit simulé. Le PGA offre de meilleures performances que tous les autres algorithmes testés. On retrouve également des algorithmes à colonies de fourmis (Ren et al., 2010), ainsi que des métaheuristiques plus exotiques telles que l'algorithme électromagnétique (Naji-Azimi et al., 2010). Ce dernier a été implémenté pour la résolution de problèmes *unicost* et se compare à l'algorithme génétique de (Beslay and Chu, 1996) et la MetaRaps de (Lan et al., 2007). Malgré le fait que la méthode électromagnétique soit plus efficace que l'algorithme génétique, la méthode MetaRaps reste la meilleure sur les instances traitées, notamment les instances géométriques. De nombreuses adaptations de recherches locales ont également été faites sur ce problème, notamment un algorithme à liste tabou (Caserta, 2007), un algorithme de recuit simulé (Brusco et al., 1999), ainsi qu'une recherche locale basée sur la pénalisation de la violation des contraintes (Yagiura et al., 2006).

2.4 Optimisation multiobjectif

Alors que les méthodes présentées dans les parties précédentes se concentrent sur l'optimisation d'un objectif unique, un grand nombre de problèmes réels nécessitent l'optimisation simultanée

de plusieurs objectifs souvent contradictoires. On peut prendre l'exemple de l'achat d'un produit, qui nécessitera la minimisation du prix d'achat tout en maximisant la qualité. Le fait d'optimiser un de ces objectifs aura tendance à détériorer le second. Si la comparaison de deux solutions est naturelle dans le cadre de l'optimisation mono-objectif, elle est toutefois plus complexe lorsque l'on doit prendre en compte plusieurs critères. Pour cela, des règles dites de dominance ont été mises en place. La plus utilisée est certainement la dominance de Pareto. Soit S l'ensemble des solutions au problème traité, u et v deux solutions appartenant à S, et $f : S \to R^m$ l'ensemble des m fonctions objectifs à minimiser. La dominance de Pareto peut être expliquée par les trois définitions suivantes :

Définition 1. La solution u est dominée par v au sens strict ($u \prec v$) si et seulement si : $\forall i \leq m, f_i(v) < f_i(u)$.

Définition 2. La solution u est dominée par v au sens faible ($u \preceq v$) si et seulement si : $\forall i \leq m, f_i(v) \leq f_i(u)$ et $\exists j \leq m, f_j(v) < f_j(u)$.

Définition 3. Les solutions u et v sont dites incomparables ou non-dominées ($u \sim v$) si et seulement si : (1) $\exists i \leq m, f_i(v) < f_i(u)$ et $\exists j \leq m, f_j(v) > f_j(u)$ ou (2) $\forall i \leq m, f_i(v) = f_i(u)$.

Ces règles permettent donc de définir trois zones de l'espace des objectifs relatives à une solution u : (1) la zone de dominance contenant l'ensemble des solutions dominant u, (2) la zone de préférence contenant l'ensemble des solutions dominées par u et enfin (3) l'ensemble des solutions incomparables à u.

Définition 4. La solution u est dite optimale si et seulement si $\nexists v \in S, v \preceq u$. On dit alors que la solution u est non-dominée.

Définition 5. L'ensemble $S^* \subseteq S$ est l'ensemble des solutions non-dominées, $\forall u \in S^*, \nexists v \in S, v \preceq u$. Les solutions appartenant à S^* forment le front de Pareto optimal.

L'optimisation multiobjectif se basant sur la dominance de Pareto se fait à l'aide d'algorithmes d'optimisation semblables à ceux utilisés dans l'optimisation mono-objectif. On retrouve des algorithmes évolutionnaires tels que les algorithmes génétiques ou encore des recherches locales. Cependant, le but d'un algorithme multiobjectif n'est pas de fournir une solution de bonne qualité en considérant le problème, mais d'approcher le plus possible le front de Pareto optimal. Il existe de nombreux algorithmes dédiés à l'optimisation multiobjectif, les plus courants étant des algorithmes évolutionnaires. Leurs performances varient selon le nombre d'objectifs à traiter ainsi que la nature de l'espace des solutions (optimisation continue ou combinatoire, variables binaires, ...). Dans cette thèse, nous nous intéresserons à quatre algorithmes évolutionnaires : les deux algorithmes génétiques NSGA-II et SPEA2, ainsi que les algorithmes MOPSO et MOEA/D. Nous décrirons ces algorithmes dans les sections suivantes.

2.4.1 *Strength Pareto Evolutionnary Algorithm 2*

L'algorithme SPEA2 a été introduit dans (Zitzler et al., 2001). Il s'agit d'une variante de l'algorithme SPEA développé par le même auteur dans (Zitzler and Thiele, 1998). Cet algorithme reprend les bases d'un algorithme génétique classique, il s'agit de faire évoluer une population de solutions en prenant en compte leur potentiel pour la reproduction et la survie. Ce potentiel se base sur un calcul de valeur d'adaptation, lui-même basé sur la dominance de Pareto. Soit une itération t, l'algorithme entretient simultanément une population P_t de taille N et une archive $\overline{P}_t$ qui contient les $\overline{N}$ meilleurs individus rencontrés. A chaque individu i est attribuée une valeur de fonction *fitness* $F(i)$. Celle-ci est calculée par l'intermédiaire de trois valeurs. La valeur $S(i)$ correspond au nombre de solutions dominées par i.

$$S(i) = |\{j | j \in P_t + \overline{P}_t \wedge i \preceq j\}| \tag{2.4}$$

Cette valeur permet de calculer $R(i)$, correspondant à la somme des valeurs $S(j)$, pour tout individu j dominant i. Une valeur $R(i)$ nulle correspondra à un individu non-dominé.

$$R(i) = \sum_{j \in P_t + \overline{P}_t \wedge j \preceq i} S(j) \tag{2.5}$$

La dernière valeur $D(i)$ correspond à une information de diversité, elle permet de départager des individus ayant la même valeur de dominance $R(i)$. Elle permet alors de privilégier les individus des zones peu denses de l'espace des objectifs. L'attribution de $D(i)$ se fait via le calcul des distances entre les solutions, afin d'en déduire le k^{eme} plus proche individu de i, dont la distance relative est exprimée par σ_i^k.

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{2.6}$$

L'auteur préconise de choisir $k = \sqrt{N + \overline{N}}$. La valeur de fonction *fitness* $F(i)$ est donc calculée comme suit :

$$F(i) = R(i) + D(i) \tag{2.7}$$

Cette méthode permet donc d'évaluer un individu en se basant non seulement sur la dominance de Pareto, mais également sur la répartition des solutions dans l'espace des objectifs. L'algorithme en lui-même est détaillé dans l'Algorithme 2.4. La gestion de l'archive intervient dans deux situations : soit le nombre d'individus non-dominés est trop faible, soit il est plus grand que $\overline{N}$. Dans le premier cas, l'archive est complété en insérant des individus dominés de la population et l'archive courante par ordre croissant de fonction *fitness*. Dans le deuxième, il faut choisir quels individus sortir de l'archive. Cependant, les individus sont alors tous non-dominés, et le choix doit alors se baser sur la répartition et donc les distances entre les individus. La réduction de l'archive est effectuée par le test booléen suivant :

$$\begin{aligned} i \leq_d j : \quad & \forall 0 < k < |\overline{P}_{t+1}| : \sigma_i^k = \sigma_j^k \vee \\ & \exists 0 < k < |\overline{P}_{t+1}| : [\forall 0 < l < k, \sigma_i^l = \sigma_j^l] \wedge \sigma_i^k < \sigma_j^k \end{aligned} \tag{2.8}$$

Algorithme 2.4 Algorithme SPEA2

Paramètres $Iter, N, \overline{N}$

$P_0 \leftarrow$ initialisation de la population
$\overline{P}_0 \leftarrow \emptyset$
Pour $t = 0$ to $Iter$ **Faire**
  Calcul des valeurs fitness des individus de P_t et $\overline{P}_t$
  Copie de tous les individus $i \in P_t + \overline{P}_t, R(i) = 0$ dans $\overline{P}_{t+1}$
  Si $|\overline{P}_{t+1}| < \overline{N}$ **Alors**
    Complétion de l'archive par des individus dominés de $P_t + \overline{P}_t$
  Sinon
    Si $|\overline{P}_{t+1}| > \overline{N}$ **Alors**
      Appel de l'opérateur de réduction de l'archive
    Fin Si
  Fin Si
  Sélection des individus pour la reproduction parmi P_{t+1}
  $P_{t+1} \leftarrow$ croisements + mutations +réparations
Fin Pour
Retourne Individus non-dominés

Cela permet de classer les individus en fonction des distances dans l'espace des objectifs. Tant que la taille de l'archive est trop élevée, l'individu possédant la densité de voisinage la plus élevée est supprimé de l'archive.

2.4.2 *Non-dominated Sorting Genetic Algorithm II*

L'algorithme génétique NSGA-II a été introduit dans (Deb et al., 2002). Il s'agit d'un des algorithmes multiobjectif les plus utilisés, notamment à cause de ses bonnes performances, et ainsi que sa facilité d'adaptation à la plupart des problèmes continus et combinatoires. Il reprend le schéma d'un algorithme génétique classique, à savoir un cycle de reproduction et un cycle de sélection (voir l'Algorithme 2.5).

Les principales innovations de NSGA-II sont :

— Une décomposition de la population en fronts de Pareto successifs. Chaque front est composé d'individus de même rang, étant donc incomparable du point de vue de la dominance de Pareto. Le premier front contient tous les individus non-dominés de la population courante, le second contient les individus uniquement dominés par des individus du premier front, etc. D'un point de vue général, les individus du front i seront dominés par des individus contenus dans les fronts précédents. Cette décomposition interviendra dans la sélection pour survie, qui permettra la transition à la prochaine itération de l'algorithme.

Algorithme 2.5 Algorithme NSGA-II

Paramètres $Iter, Size_{Pop}, Size_{Sel}$

$P_0 \leftarrow$ Population initiale

Pour $i = 1$ to $Iter$ **Faire**

$M_i \leftarrow$ sélectionReproduction(P_{i-1})

$O_i \leftarrow$ croisement + mutation +réparation (O_i)

$P_i \leftarrow P_{i-1} + O_i$

Calcul des fronts et des distances

$P_i \leftarrow$ sélectionSurvie (P_i)

Fin Pour

Retourne configurations non-dominées

— La sélection pour la survie consiste en l'insertion successive des fronts de la population courante dans la prochaine population jusqu'à atteindre exactement la taille requise. Il est courant que le dernier front à insérer contienne plus d'individus qu'il n'en faut, et par conséquent il faut alors procéder à une sélection à l'intérieur du front. Deb et al. (2002) proposent d'utiliser la distance de *crowding* pour procéder au choix des individus à insérer. Les priorités sont alors mises sur les deux points extrêmes du front, puis sur les individus classés selon leur espacement dans l'espace des objectifs. L'opérateur de *crowding* cherche alors une distribution uniformisée du front.

Comparé au SPEA2 présenté dans la partie précédente, NSGA-II détient l'avantage de ne pas avoir à entretenir une archive annexe à la population courante. Pour cette raison, NSGA-II est souvent plus rapide pour effectuer un même nombre d'itérations, et offre donc de meilleures performances. L'évaluation d'un individu ne se fait que par le rang du front auquel il appartient. NSGA-II se différencie également sur le calcul des distances, qui n'intervient qu'à l'intérieur du front à fractionner.

2.4.3 *Multi Objective Evolutionnary Algorithm based on Decomposition*

L'algorithme évolutionnaire MOEA/D, proposé dans (Zhang and Li, 2007), est basé sur la décomposition afin de traiter des problèmes multiobjectif. Son fonctionnement se base sur la création d'un ensemble de N problèmes mono-objectif en tenant compte du problème multiobjectif initial. La décomposition peut se faire de plusieurs manières, la plus efficace pour les problèmes que nous

traiterons étant la somme pondérée :

$$Minimiser\ g^j = \sum_{i=1}^{m} \lambda_i^j f_i \tag{2.9}$$

$Où$:

$\sum_{i=1}^{m} \lambda_i^j = 1$

$1 \leq j \leq N$

L'algorithme MOEA/D est défini dans l'Algorithme 2.6.

Algorithme 2.6 MOEA/D

Paramètres $Iter, N, T$

$EP \leftarrow \emptyset$ L'archive des solutions non dominées

$\lambda \leftarrow random$ Les N vecteurs de poids utilisés pour l'optimisation des sous-problèmes

$\forall i \leq Size_{Pop}, B(i) = \{i_1, ..., i_T\}$ Les T plus proches vecteurs λ par rapport à λ_i

$X = \{X^1, ..., X^N\}$ La population initiale

Pour $j = 1$ to $Iter$ **Faire**

Pour $i = 1$ to N **Faire**

Choisir aléatoirement deux solutions X^A et X^B dans $B(i)$, puis utiliser les opérateurs de reproduction pour générer X'.

$\forall k \in B(i)$, si $g_i(X') \leq g_i(X^k)$ alors $X^k = X'$

Mettre à jour EP

Fin Pour

Fin Pour

Retourne EP

Le premier pas est l'initialisation des N vecteurs de poids utilisés dans l'optimisation des sous-problèmes. Le paramètre N est fixé en fonction du nombre de niches disponibles dans le front de Pareto. Pour tout $i \leq N$, une configuration X^i est créée et insérée dans X. A chaque itération i et pour tout indice $i \leq N$, les opérateurs de reproduction sont utilisés pour créer de nouvelles configurations. La survie de celles-ci est décidée en fonction de leur score sur le sous-problème considéré. L'archive des configurations non-dominées est mise à jour à chaque création d'individu.

2.4.4 *Multi Objective Particle Swarm Optimization*

L'algorithme à essaim particulaire est une métaheuristique présentée dans (Kennedy and Eberhart, 1995). S'inspirant de la recherche de nourriture par les oiseaux, il s'agit d'un algorithme évolutionnaire combinant l'inertie de la trajectoire, la mémoire de la particule ainsi que l'influence sociale afin d'explorer l'espace. Chaque particule p est composée de trois vecteurs $x(p)$, $v(p)$ et $m(p)$ étant respectivement la position, la vitesse et la mémoire de la meilleure position traversée par la

particule. La particule possède également un voisinage $V(p)$ géographique ou social qui lui permettra d'influer sur sa trajectoire en fonction des particules proches. Les paramètres de l'algorithme sont les trois coefficients w, c_1 et c_2 qui sont respectivement l'inertie, l'impact de la mémoire et l'impact du voisinage sur la trajectoire courante. Soit une itération t, la transition à l'itération suivante se fait grâce aux équations ci-dessous :

$$v_{t+1}(p) = wv_t(p) + c_1(m_t(p) - x_t(p)) + c_2(x_t(q) - x_t(p)) \tag{2.10}$$

$$x_{t+1}(p) = v_{t+1}(p) + x_t(p) \tag{2.11}$$

$$q = argmin(f(x_t(q)), q \in V(p))$$

L'inertie $wv(t)$ représente la confiance de la particule en sa trajectoire courante. Elle permet une exploration plus diversifiée de l'espace et éventuellement de sortir des optima locaux. La mémoire s'actualise à chaque itération, afin de conserver l'emplacement de la meilleure position du point de vue de l'objectif. Enfin, la meilleure particule dans le voisinage est évaluée selon l'objectif et la position courante. Le processus de PSO est défini dans l'Algorithme 2.7.

Algorithme 2.7 Algorithme PSO

Paramètres $Iter, Size_{Swarm}, w, c_1, c_2$

$Swarm \leftarrow$ initialisation des particules de l'essaim

$X* \leftarrow$ la meilleure solution extraite de P_0

Pour $i = 1$ to $Iter$ **Faire**

actualisation des vitesses et des positions des particules

actualisation de $X*$

Fin Pour

Retourne $X*$

L'algorithme MOPSO est l'adaptation du PSO classique pour les problèmes multiobjectif (Coello and Lechuga, 2002). Les principales différences sont l'entretien d'une archive contenant l'ensemble des solutions non-dominées rencontrées, ainsi qu'un mécanisme multiobjectif pour traiter l'influence des positions en mémoire et du voisinage. Nous utiliserons également un procédé tiré de (Pampara and Engelbrecht, 2011), afin de traiter des problèmes binaires avec un algorithme à essaim particulaire. Plus précisément, nous avons choisi la première approche présentée, considérant la vitesse comme une probabilité de fixer la variable à 1.

2.5 Optimisation multiobjectif dans les réseaux de capteurs

L'optimisation multiobjectif est appliquée lorsque le but est d'optimiser plusieurs objectifs (souvent contradictoires) de manière explicite. On pourra prendre l'exemple de la maximisation de la couverture ainsi que de la durée de vie. L'application de l'optimisation multiobjectif dans le cadre

des réseaux de capteurs a fait l'objet de nombreuses études. J. Jia et al. (2009) et Jia et al. (2009) ont proposé deux travaux traitant de l'activation de capteurs pré-déployés. Tout d'abord, le problème multiobjectif suivant fut considéré : (i) la maximisation du taux de couverture, en utilisant le modèle probabiliste de couverture et (ii) la minimisation du nombre de capteurs activés. Les capteurs sont initialement déployés dans la zone de manière aléatoire. Un nouvel algorithme ECCA (*Energy-efficient Coverage Control Algorithm*) basé sur NSGA-II est également proposé pour déterminer l'ensemble minimum de capteurs à activer. Cet algorithme a été comparé à plusieurs méthodes telles que PEAS (Ye et al., 2003) et OGDC (Zhang and Hou, 2005). Les résultats ont montré que l'algorithme ECCA a fourni de meilleures performances que les autres algorithmes sur ce problème. Le deuxième travail ajoute un objectif relatif à la consommation d'énergie au modèle, en considérant le rayon de couverture des capteurs comme une variable de décision. Les auteurs utilisent alors un algorithme NSGA-II amélioré et se comparent à l'algorithme OGDC sur plusieurs instances. Dans (Masazade et al., 2010), les auteurs ont développé une nouvelle méthode NBI (*Normal Boundary Intersection*) pour résoudre le problème biobjectif suivant :(i) la minimisation de la probabilité de l'erreur de détection et (ii) la minimisation de la consommation d'énergie. Les variables de décision de ce problème sont les seuils énergétiques de détection qui déterminent le minimum d'intensité du signal reçu par un capteur provoquant l'alerte de détection. Un seuil trop faible provoquera des faux positifs (i.e. une intrusion est détectée alors qu'aucun intrus ne se trouve dans la zone) alors qu'un seuil trop élevé risque de provoquer la non-détection des intrus. La méthode mise en place a été comparée à NSGA-II, et les expérimentations ont montré un net avantage pour la nouvelle méthode.

Le déploiement est parfois considéré comme un élément de décision. Cela s'applique dans le cas où l'utilisateur a la capacité de placer les capteurs à des positions bien précises. On retrouve cette approche dans (Konstantinidis et al., 2010), où les auteurs présentent le problème DPAP (*Deployment and Power Assignment Problem*). Soit un ensemble de capteurs de taille définie, les objectifs sont de maximiser la couverture de la zone pixellisée (modèle binaire) ainsi que de maximiser la durée de vie du réseau. La durée de vie correspond à l'instant où l'un des capteurs vient à manquer d'énergie. Les variables de décision sont les positions des capteurs ainsi que la puissance attribuée aux dispositifs de communication. Pour résoudre ce problème, l'auteur propose un algorithme de décomposition MOEA/D qu'il compare à NSGA-II. Les auteurs redéfinissent ce problème dans (Konstantinidis and Yang, 2011), où ils intègrent un modèle énergétique plus réaliste et s'intéressent aux déploiements denses (un grand nombre de capteurs dans une zone de taille réduite). MOEA/D précédemment utilisé est modifié et hybridé avec des heuristiques spécifiques au problème, il s'est avéré plus performant que les précédentes méthodes. Une modélisation plus réaliste de la zone a été traitée dans (Lee et al., 2012), où les auteurs ont considéré les obstacles, la variation du rayon de détection et les positions inaccessibles pour le déploiement. Les auteurs ont également développé un algorithme MOASA (*Multi-objective Optimization Approach for Sensor Arrangement*), inspiré de l'algorithme SPEA2 pour résoudre le problème multiobjectif suivant : (i) la maximisation de la

couverture binaire, (ii) la minimisation de la redondance de la couverture et (iii) la minimisation du nombre de capteurs à déployer. Wei et al. (2009) ont développé un algorithme génétique multiobjectif FD-MOGA (*Forced-Driven Multi-Objective Genetic Algorithm*) pour résoudre un problème de déploiement en trois dimensions où les objectifs sont la maximisation de la couverture probabiliste et la maximisation des niveaux de détection, et la minimisation de la consommation d'énergie. L'algorithme FD-MOGA s'est révélé plus performant que l'algorithme MOGA. Dans l'article (Oh et al., 2007), les auteurs ont utilisé un algorithme NSGA-II pour optimiser les quatre objectifs suivants : (i) la maximisation de la couverture binaire (en prenant en compte plusieurs schémas géométriques de couverture), (ii) la minimisation du nombre de capteurs à déployer, (iii) la maximisation de la préférence de l'utilisateur, en prenant en compte un classement des types de capteurs à utiliser et (iv) la minimisation de la distance entre la cible et les capteurs.

2.6 Conclusion

Dans cet état de l'art, nous avons présenté des travaux de quatre thématiques différentes : (i) les problématiques liées à l'optimisation dans les réseaux de capteurs, (ii) le *Set Covering Problem*, (iii) l'optimisation multiobjectif et (iv) l'application de l'optimisation multiobjectif dans les problématiques liées aux réseaux de capteurs. Aucun des travaux présentés ne satisfait les différentes contraintes liées à la problématique de cette thèse, notamment en ce qui concerne le traitement simultané de la localisation et du déploiement. Nous chercherons donc par la suite à proposer des modèles mono-objectif et multiobjectif permettant d'effectuer des déploiements favorisant la localisation à l'aide de signaux instables.

Chapitre 3

Déploiement des capteurs et localisation de cibles

3.1 Introduction

Les applications actuelles nécessitent de plus en plus l'intégration du suivi dans les décisions de déploiement des réseaux de capteurs. On peut prendre l'exemple des GPS permettant de se repérer sur les réseaux routiers, mais également des applications mobiles affinant leurs résultats ou leurs services en exploitant la position de l'utilisateur. Nous nous intéressons ici à la localisation à l'intérieur de bâtiments tels que les hôpitaux et les musées afin de guider les visiteurs et de leur permettre de se déplacer plus facilement. Si les techniques de localisation utilisant les signaux GSM sont connues depuis longtemps, leur utilisation à l'intérieur des bâtiments, notamment les hôpitaux, est proscrite pour plusieurs raisons. Tout d'abord, l'effet sanitaire de l'exposition aux signaux GSM est encore peu connu, ce qui a pour conséquence de provoquer une certaine méfiance de la part des utilisateurs. De plus, la couverture à l'intérieur des bâtiments est parfois inégale, en fonction des fréquences utilisées par les opérateurs téléphoniques. Pour ces raisons, nous privilégions les technologies déjà omniprésentes dans les bâtiments, et plus précisément les signaux WiFi. La triangulation est l'une des méthodes les plus utilisées pour procéder à l'estimation de la position de cible. Cependant, cette technique s'avère inefficace lorsque l'on utilise un signal instable.

Dans le chapitre précédent, nous avons donné un aperçu des différentes thématiques abordées dans l'optimisation des déploiements de réseaux de capteurs, et notamment l'application de l'optimisation multiobjectif dans ce domaine. Si le fait d'assurer la couverture représente la première étape permettant l'estimation de position, une modification importante des modèles présentés précédemment est nécessaire afin de prendre en compte la précision de la localisation.

3.2 Description du problème

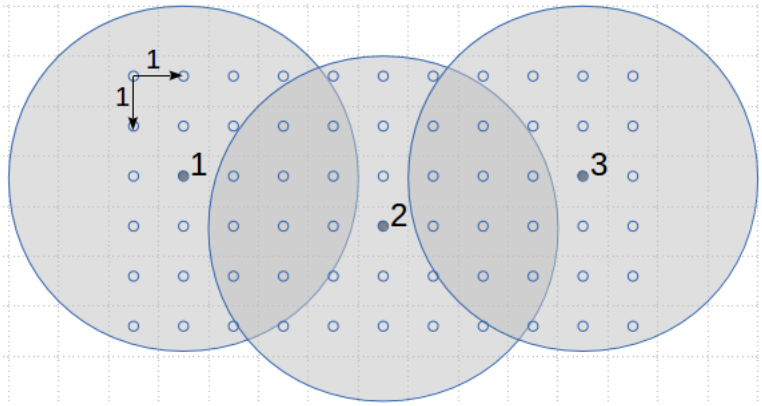

FIGURE 3.1 – Exemple de déploiement et couverture

Nous cherchons à proposer un déploiement de capteurs dans une zone optimisant la précision de la localisation. Soit P l'ensemble des positions résultant de la discrétisation de la zone à couvrir, et $|P|$ le nombre de positions dans la zone. Chaque position peut accueillir un capteur, et chaque position doit être couverte par au moins un capteur.

La Figure 3.1 montre un exemple de déploiement sur une grille de 6 lignes par 11 colonnes. Les positions de couleur sombre représentent les capteurs déployés, qui couvrent toutes les positions. Pour chaque capteur, la couverture correspondante est représentée par un disque binaire. Tout point à l'intérieur du disque est couvert par le capteur. Dans ce chapitre, nous nous intéressons à la modélisation de l'erreur de la localisation d'un événement dans la grille. Si un événement est détecté par un capteur, on peut dire avec certitude qu'il se trouve dans l'ensemble de détection correspondant au capteur. En d'autres termes, l'événement se trouve dans le disque correspondant. Il est possible d'affiner la détection en reliant les informations fournies par l'ensemble des capteurs. En effet, on peut dissocier les ensembles de détection fournis par les différentes combinaisons de capteurs. Dans l'exemple précédent, trois capteurs sont déployés, ce qui correspond aux $2^{|P|} - 1$ combinaisons suivantes : $\{\{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$. La Figure 3.2 représente l'apparition d'une cible dans la zone aux coordonnées $(4,4)$. La position correspondante est représentée par un carré. On peut voir que la cible est détectée par le premier et le deuxième capteur, la position de la cible appartenant aux deux disques de couverture correspondants.

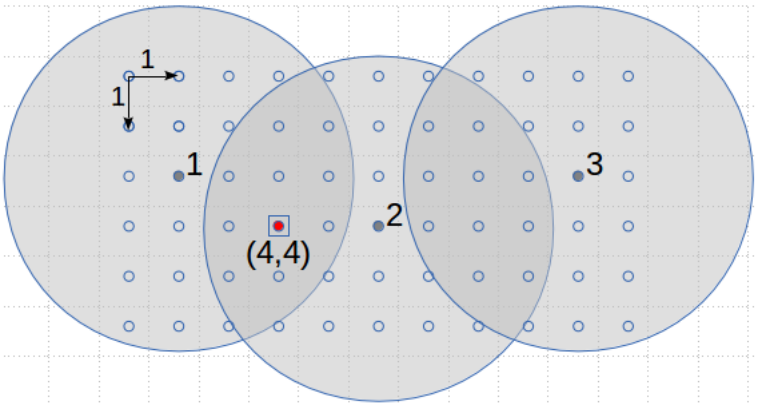

FIGURE 3.2 – Détection d'une cible

La Figure 3.3 représente l'estimation de la position de la cible par le réseau de capteurs. Dans le cas d'un modèle binaire de couverture, la seule information disponible est la détection binaire, c'est-à-dire que l'on peut seulement déterminer quels capteurs détectent la cible. A partir de cette information, ainsi que des ensembles couverts par chaque disque de couverture, le réseau de capteurs localise la cible dans un ensemble de 10 positions représentées par les losanges. Cette taille correspond à la taille de la zone de recherche suite à la détection d'un événement par la combinaison de capteurs concernés. Il est donc important de minimiser la taille de cet ensemble afin d'affiner la localisation et ainsi faciliter la recherche d'un événement suite à une détection.

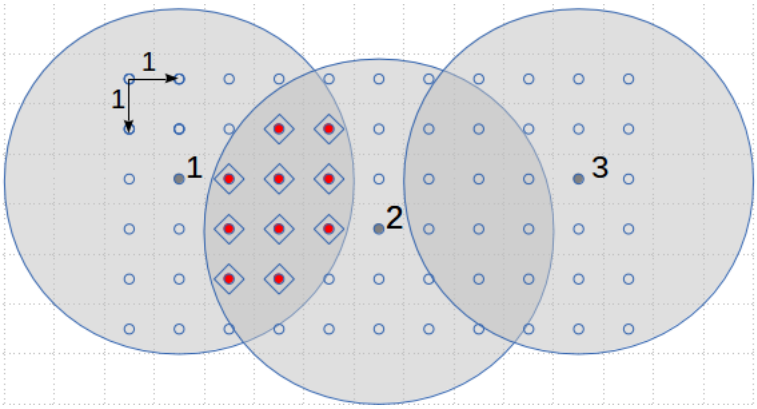

FIGURE 3.3 – Estimation de la position de la cible

Nous abordons ici un problème biobjectif : le premier critère est la minimisation du nombre de capteurs à déployer, qui est un objectif courant dans les problématiques de déploiement. En effet, cet objectif présente un intérêt majeur selon plusieurs points de vue. Premièrement, un nombre moins important de capteurs induit un coût de déploiement plus léger, au niveau de l'achat des unités à déployer et de la main d'œuvre à employer pour le déploiement (temps ou nombre d'employés). La réduction de ce nombre est également importante d'un point de vue invasif. En effet, la réduction

de la visibilité du réseau de capteurs peut être souhaitable au niveau de l'esthétique du bâtiment à couvrir. Le deuxième critère est la maximisation de la précision de localisation. L'intérêt de ce nouvel objectif se concentre sur les applications de localisation, et on le représentera ici comme une qualité de service.

3.3 Modélisation mathématique

Soit P l'ensemble des positions résultant de la discrétisation de la zone à couvrir. A chaque position i est attribuée une variable de décision binaire X_i, correspondant à la présence d'un capteur déployé sur la position i. La formulation du premier objectif est la suivante :

$$Minimiser\ z_1 = \sum_{i \in P} X_i \tag{3.1}$$

$$S.c. :$$

$$\sum_{i \in P} a_{i,j} \cdot X_i \geq 1, \forall j \in P \tag{3.2}$$

$$X_i \in \{0,1\}, \forall i \in P \tag{3.3}$$

Les contraintes (3,2) expriment la nécessité de la couverture de chaque position par au moins un capteur. Le coefficient binaire $a_{i,j}$ exprime la couverture de la position j par un hypothétique capteur déployé en i. Ce modèle est fréquemment utilisé dans les problèmes de déploiement utilisant une modélisation binaire de la couverture du capteur. Il s'agit également du *Set Covering Problem* présenté dans l'état de l'art.

3.3.1 Calcul de la précision de localisation des cibles

Nous définissons ici la précision de localisation des cibles par l'erreur de détection. L'objectif revient à minimiser la taille de la zone de recherche suite à la détection d'un événement. En partant du fait que la zone contient $|P|$ positions, il y a au maximum $|P|$ ensembles de détection, l'ensemble de détection S_j attribué à une position j correspondant à l'ensemble des positions en conflit avec j. Tout d'abord, nous définissons $Y_{j,k}$ une variable binaire représentant l'appartenance de j et k au même ensemble de détection.

$$Y_{j,k} = \begin{cases} 1 & \text{si } \forall i \in P \text{ tel que } |a_{i,j} - a_{i,k}| X_i = 0 \\ 0 & \text{sinon} \end{cases}$$

La variable $Y_{j,k}$ est égale à 1 si et seulement s'il n'existe pas de capteur déployé permettant de distinguer j de k. Autrement dit, $Y_{j,k}$ est égale à 1 si tout capteur activé voyant j voit également k. La contrainte liée à $Y_{j,k}$ est la suivante :

$$\sum_{i \in P} (|a_{i,j} - a_{i,k}| \cdot X_i) + Y_{j,k} \geq 1, \forall (j,k) \in P^2, j \neq k \tag{3.4}$$

En se basant sur le calcul des variables binaires $Y_{j,k}$, la taille de l'ensemble de détection S_j peut être calculée de la manière suivante :

$$|S_j| = \sum_{k \in P, k \neq j} Y_{j,k}, \forall j \in P \tag{3.5}$$

Où la valeur de $|S_j|$ est comprise dans l'ensemble $\{0, .., N_{max}\}$. Dans le cas idéal, l'ensemble S_j ne contiendra aucune position, permettant une localisation idéale d'une cible apparaissant sur cette position. Le pire cas est calculé en fonction du rayon de couverture, c'est-à-dire que N_{max} est le nombre maximum de positions couvertes par un seul capteur. Cela permet de définir le deuxième objectif de minimisation de l'erreur de détection comme suit :

$$Minimiser\ z_2 = \frac{1}{|P|} \sum_{j \in P} [\frac{1}{N_{max}} \sum_{k \in P, k \neq j} Y_{j,k}] \tag{3.6}$$

Le modèle complet est le suivant :

$$Minimiser\ z_1 = \frac{1}{|P|} \sum_{i \in P} X_i \tag{3.7}$$

$$Minimiser\ z_2 = \frac{1}{|P|} \sum_{j \in P} [\frac{1}{N_{max}} \sum_{k \in P, k \neq j} Y_{j,k}] \tag{3.6}$$

$S.c.$:

$$\sum_{i \in P} a_{i,j} \cdot X_i \geq 1, \forall j \in P \tag{3.2}$$

$$\sum_{i \in P} (|a_{i,j} - a_{i,k}| \cdot X_i) + Y_{j,k} \geq 1, \forall (j,k) \in P^2, j \neq k \tag{3.4}$$

$$X_i \in \{0,1\}, \forall i \in P \tag{3.3}$$

$$Y_{j,k} \in \{0,1\}, \forall (j,k) \in P^2, j \neq k \tag{3.8}$$

Il est à noter que le premier objectif (3,7) a été légèrement modifié afin d'avoir deux objectifs homogénéisés entre 0 et 1. On obtient donc un modèle linéaire, dont le nombre de variables et de contraintes dépend de la taille de la grille et de sa discrétisation.

3.3.2 Optimisation linéaire et front de Pareto optimal

La modélisation mathématique du problème sous forme linéaire n'a d'intérêt que si l'on procède à l'optimisation linéaire afin d'obtenir le front de Pareto optimal. Ceci n'est possible que sur des instances de taille restreinte, et sert généralement à tester la qualité de convergence de méthodes approchées avant de procéder à des tests sur des instances de grande taille. Il existe différentes méthodes pour construire le front de Pareto optimal, notamment la méthode ϵ-contraintes, la pondération des objectifs et la méthode à 2 phases (TPM).

Pour la construction du front de Pareto, nous commençons par calculer les deux points extrêmes du front de Pareto $P_{z1} = (z_1^{min}, z_2^{max})$ et $P_{z2} = (z_1^{max}, z_2^{min})$ représentant respectivement la préférence pour le premier et le second objectif (voir Figure 3.4).

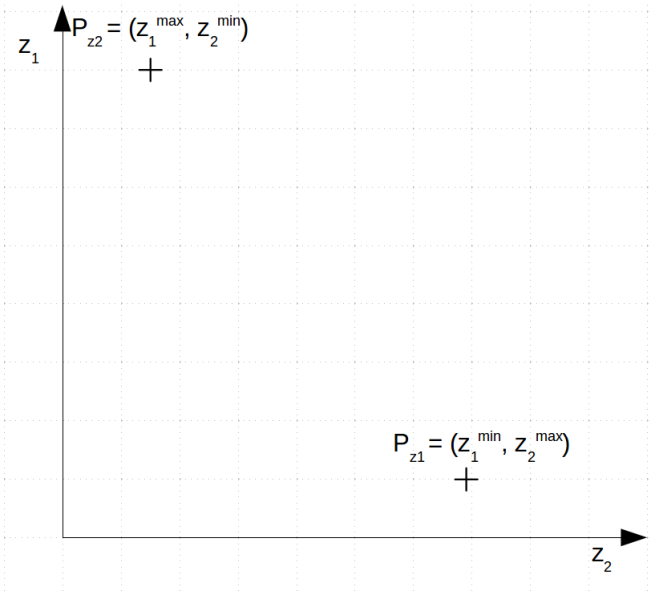

FIGURE 3.4 – Calcul des points extrêmes du front de Pareto

La valeur de z_1^{min} se calcule en minimisant z_1 tout en ne prenant en compte que les contraintes de couverture. Ensuite, on minimise z_2 en prenant en compte toutes les contraintes tout en ajoutant la contrainte suivante :

$$\frac{1}{|P|}\sum_{i\in P} X_i \leq z_1^{min} \tag{3.9}$$

Ce qui nous permet d'obtenir la valeur de z_2^{max}. Le second point extrême s'obtient de manière similaire : on commence par optimiser z_2 sans se préoccuper de la valeur de z_1, ce qui nous permet d'obtenir z_2^{min}. Puis on minimise z_1 en ajoutant la contrainte suivante au modèle :

$$\frac{1}{|P|}\sum_{j\in P}[\frac{1}{N_{max}}\sum_{k\in P, k\neq j} Y_{j,k}] \leq z_2^{min} \tag{3.10}$$

Ce qui nous permet d'obtenir la valeur de z_1^{max}. Une fois les points extrêmes obtenus, la complétion du front de Pareto se fait d'une manière itérative en utilisant deux modèles $M1$ et $M2$, permettant d'obtenir respectivement les prochaines valeurs de z_2 et z_1.

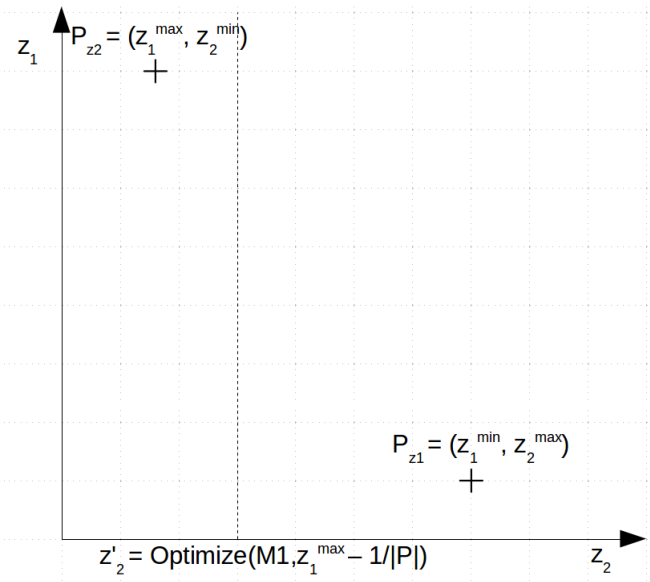

FIGURE 3.5 – Détermination de la prochaine valeur z_2

Le modèle $M1$ consiste en la minimisation de z_2, en prenant en considération toutes les contraintes du modèle général, ainsi que la valeur maximale de z_1 passée en paramètre. La première valeur de z_2 est calculée en fonction de la valeur limite $z_1^{max} - \frac{1}{P}$. La valeur $\frac{1}{P}$ provient de l'homogénéisation de la fonction objectif. Si l'on regarde l'objectif initial (3.1), on remarque que les valeurs retournées par cette fonction sont entières, étant donné qu'il s'agit d'une somme de variables binaires. Dans ce cas, si l'on prend en compte la division par la constante $|P|$ de l'objectif z_1, écrire l'inégalité $\frac{1}{|P|}\sum_{i\in P} X_i < z_1^{max}$ revient à écrire $\frac{1}{|P|}\sum_{i\in P} X_i \leq z_1^{max} - \frac{1}{|P|}$. Nous obtenons donc z_2' la valeur du second objectif du point suivant sur le front de Pareto (voir Figure 3.5).

Une fois la valeur z_2' acquise, le modèle $M2$ permet l'obtention de la valeur z_1' du point de la même manière que pour le point extrême P_{z2} (voir Figure 3.6). On minimise donc le premier objectif, en ajoutant la contrainte suivante :

$$\frac{1}{|P|}\sum_{j\in P}[\frac{1}{N_{max}}\sum_{k\in P, k\neq j} Y_{j,k}] \leq z_2' \tag{3.11}$$

Les points suivants s'obtiennent itérativement de la même manière, en remplaçant la valeur z_1^{max} par celle du dernier point calculé afin de procéder au calcul de la prochaine valeur du second objectif.

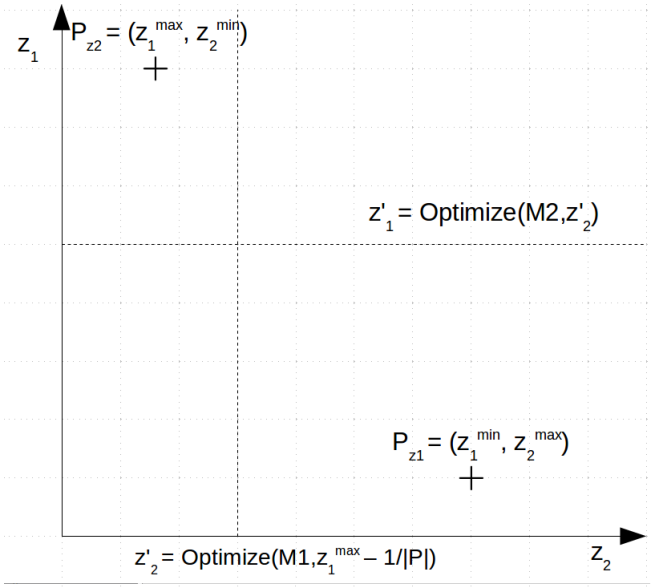

FIGURE 3.6 – Détermination de la prochaine valeur z_1

3.4 Adaptation des algorithmes multiobjectif

Pour traiter ce problème, nous procéderons à l'adaptation des méthodes NSGA-II, SPEA2, MOEA/D et MOPSO dont les algorithmes généraux ont été décrits dans l'état de l'art. L'avantage des trois premiers algorithmes est la facilité d'implémentation pour les problèmes binaires. En effet, le principe évolutionnaire se base sur la génétique et s'adapte à la fois aux problèmes d'optimisation combinatoires et continus. La principale difficulté est la création d'opérateurs appropriés permettant l'efficacité de ces méthodes pour le problème. La première étape est la représentation de la solution par un chromosome. Ici les variables de décision sont les positions des capteurs. Deux solutions s'offrent à nous : la première est de stocker les positions (lignes et colonnes) des capteurs dans la grille dans un vecteur de taille variable. La seconde hérite directement de la modélisation mathématique décrite ci-dessus : le chromosome est alors représenté par un vecteur de taille fixe, correspondant aux nombres de positions dans la grille, dont chaque élément est un booléen représentant la présence d'un capteur déployé dans la position correspondante. Si la première est plus compacte d'un point de vue informatique, et provoque une accélération d'un point de vue algorithmique, notamment pour les procédures de copie d'individus, la seconde est plus naturelle et facilite la création des opérateurs de croisement et de mutation. Nous privilégierons donc cette dernière :

Soit X une solution proposée au problème, $N = |P|$:

$$X = \{X_1, ..., X_N\},$$
$$\forall i \in \{1, .., N\}, X_i \in \{0, 1\}$$

Cette représentation est la même que celle utilisée lors de la modélisation mathématique. Le

calcul des fonctions objectifs et de la faisabilité d'une solution se fait également de la même manière. Les algorithmes génétiques multiobjectif agissent selon des schémas généralement semblables (voir

Algorithme 3.1 Schéma général de l'algorithme génétique multiobjectif

$P_0 = initialisation()$
Tant que Le critère d'arrêt n'est pas atteint **Faire**
$Q_i = \{croisement + mutation\}(P_i)$
$Q_i = \{reparation + optimisation\}(Q_i)$
$P_{i+1} = selectionSurvie(P_i + Q_i)$
Fin Tant que
Retourne Solutions non dominées

l'Algorithme 3.1). La première étape est l'initialisation de la population, permettant de créer un ensemble de départ à l'optimisation. Puis des opérateurs de reproduction sont appliqués afin de procéder à une simulation de l'évolution naturelle. Un opérateur de réparation est parfois nécessaire afin de garantir la faisabilité des solutions proposées pour le problème. Il est également possible d'intégrer une méthode d'optimisation à ce cycle afin d'accroître les performances des solutions engendrées à chaque itération. Cette dernière étape, présente dans les algorithmes dits "hybrides", utilisés dans la partie 3.6, permet une accélération de la convergence, mais consomme cependant plus de temps.

3.4.1 Initialisation de la population

L'initialisation de la population est la première étape de la plupart des algorithmes évolutionnaires. Le but de cette procédure est de créer un grand nombre de solutions réalisables pour le problème, tout en gardant une certaine diversité. Les deux opérateurs d'initialisation que nous utiliserons sont les suivants : (i) aléatoire et (ii) SCP-guidée.

3.4.1.1 Initialisation aléatoire

L'opérateur d'initialisation aléatoire permet d'obtenir une population à très forte diversité au détriment des valeurs objectifs des individus. La création d'un individu est la suivante : tant que toutes les positions ne sont pas couvertes, on ajoute aléatoirement un capteur couvrant au moins une position. Cette dernière condition est souhaitable, surtout lors de l'exécution sur de grandes instances, afin d'éviter de consommer trop de temps lors de la création de la population.

3.4.1.2 Initialisation SCP-guidée

Nous présentons ici un opérateur d'initialisation permettant de créer des individus de manière heuristique en considérant une pondération des objectifs variable. Cet opérateur tient compte de

trois cas : le premier est la création du point extrême considérant la minimisation du nombre de capteurs en ne tenant compte que des contraintes de couverture. Le deuxième point extrême est obtenu dans le cas où la contrainte $z_2 = 1$ ne rend pas l'instance du problème irréalisable. Dans les deux cas, on utilise une heuristique gloutonne dédiée au SCP résultant. Les points intermédiaires sont obtenus en procédant à une pondération des objectifs, via les coefficients α et β qui permettent d'obtenir une solution représentative d'une zone particulière du front de Pareto (i.e. $z = \alpha z_1 + \beta z_2$). Chaque variable est associée à un poids calculé en fonction des coefficients. Cette procédure permet la création d'une solution réalisable pour le problème dont la construction de la solution se termine lorsque chaque contrainte de couverture est satisfaite.

3.4.2 Opérateur de croisement

L'opérateur de croisement permet d'intensifier la population au sein d'un algorithme génétique. Cette procédure permet le mélange des informations de deux configurations données, résultant d'une ou plusieurs solutions selon l'opérateur choisi. Pour notre problème, nous nous intéressons à trois opérateurs très utilisés dans la littérature : le croisement en un point, le croisement multi-points et le croisement uniforme.

3.4.2.1 Croisement en un point

Le croisement en un point permet de mélanger les données de deux individus tout en gardant une certaine cohérence. La première étape consiste à choisir le point k, celui-ci étant aléatoirement fixé entre 2 et $N-1$. Soit deux configurations choisies X et X', la création des nouvelles configurations Y et Y' se fait de la manière suivante :

$$Y_i = \begin{cases} X_i & \forall i \leq k \\ X'_i & \forall k < i \leq N \end{cases}$$

$$Y'_i = \begin{cases} X'_i & \forall i \leq k \\ X_i & \forall k < i \leq N \end{cases}$$

3.4.2.2 Croisement en multi-points

Le croisement en multi-points est une variante du précédent opérateur, le but ici est d'accroître le mélange des données en générant aléatoirement K points $\{k_1, .., k_K\}$ ordonnés, avec $k_j < k_{j+1}$, les valeurs de ces points étant comprises entre 2 et $N-1$. Soit deux configurations choisies X et X', la création des nouvelles configurations Y et Y' se fait de la manière suivante :

$$\forall i \leq k_1, Y_i = X_i \text{ et } Y'_i = X'_i$$
$$\forall 1 < j \leq K, k_{j-1} < i \leq k_j$$
$$(Y_i, Y'_i) = \begin{cases} (X_i, X'_i) & \text{si } j \text{ est pair} \\ (X'_i, X_i) & \text{si } j \text{ est impair} \end{cases}$$

3.4.2.3 Croisement uniforme

Le croisement uniforme permet de générer un nouvel individu en sélectionnant les meilleurs gènes parmi les parents. L'évaluation d'un gène est dépendante du problème, on peut toutefois se baser sur le principe des heuristiques de construction telles que l'algorithme glouton dédié au SCP. Soit une variable Y_i, sa probabilité d'insertion est calculée selon l'équation suivante :

$$\forall i \leq N$$
$$P(Y_i = 1) = \begin{cases} 0 & \text{si } X_i = X'_i = 0 \\ 1 & \text{si } X_i = X'_i = 1 \\ \frac{\sum_{j \in P} a_{i,j}}{\max_{k \in P}(\sum_{j \in P} a_{k,j})} & \text{sinon} \end{cases}$$

3.4.3 Opérateur de mutation

L'opérateur de mutation a pour vocation de diversifier la population. Il s'agit généralement de mutation aléatoire, permettant de sortir d'optimums locaux et poursuivre la convergence. Cet opérateur se sert donc de la randomisation pour effectuer des changements "aveugles". Soit p_m la probabilité de mutation, l'opérateur de mutation suit la procédure suivante :

$$\forall i \leq N$$
$$tirage = U(0, 1)$$
$$X_i = \begin{cases} X_i & \text{si } tirage > p_m \\ 1 - X_i & \text{si } tirage \leq p_m \end{cases}$$

Où $U(0, 1)$ est une loi uniforme entre 0 et 1.

3.4.4 Opérateur de réparation

L'opérateur de réparation intervient lors de la création d'une nouvelle solution via les opérateurs de reproduction, qui, à l'inverse des solutions proposées par les opérateurs d'initialisation, ne garantit pas sa faisabilité par rapport au problème. Le but de cet opérateur est de rendre la nouvelle solution réalisable. L'opérateur que nous proposons se base une fois encore sur l'heuristique gloutonne du SCP. La première étape est d'identifier les contraintes non respectées, puis de créer le SCP résultant. L'opérateur va ensuite choisir les capteurs à insérer afin de satisfaire l'ensemble des contraintes.

3.5 Expérimentations

Tous les algorithmes présentés précédemment ont été implémentés en C++, et leur exécution a été faite sur un Core-I5 sous Linux. Nous proposons deux ensembles d'instances rectangulaires SI et LI. Le premier ensemble d'instances a également été traité par le solveur linéaire Gurobi 5.6.

3.5.1 Paramétrages et opérateurs

Des tests préliminaires nous ont permis de sélectionner la combinaison suivante d'opérateurs pour les algorithmes génétiques : Initialisation guidée et Croisement en un point. MOPSO disposera également de l'initialisation guidée. Les probabilités de croisement et de mutation ont respectivement été fixées à 0,9 et $\frac{1}{N}$. NSGA-II dispose d'une population de 120 individus, dont 60 sélectionnés pour le croisement. L'archive de SPEA2 sera réglée à 250 individus, et la taille de la population courante à 100. MOEA/D dispose d'une taille de population variable, correspondant à l'écart en nombre de capteurs séparant les deux points extrêmes du Pareto. Le paramètre de proximité T est fixé à 30% de la population. Enfin, les trois paramètres w, c_1 et c_2 de MOPSO sont respectivement fixés à 0,9, 2,5 et 2,5, et la taille de l'essaim est réglée à 200 particules.

3.5.2 Instances

Les instances sont définies selon les informations suivantes : la taille de la grille (H et W étant respectivement les nombres de lignes et de colonnes) et le rayon d'acquisition R. La valeur N_{max} est calculée en fonction du rayon et de la géométrie de la grille.

TABLEAU 3.1 – Ensemble d'instances SI

	H	W	R		H	W	R
Instance $SI1$	5	5	1	Instance $SI10$	6	6	1
Instance $SI2$	5	5	2	Instance $SI11$	6	6	2
Instance $SI3$	5	5	3	Instance $SI12$	6	6	3
Instance $SI4$	6	5	1	Instance $SI13$	7	6	1
Instance $SI5$	6	5	2	Instance $SI14$	7	6	2
Instance $SI6$	6	5	3	Instance $SI15$	7	6	3
Instance $SI7$	7	5	1	Instance $SI16$	7	7	1
Instance $SI8$	7	5	2	Instance $SI17$	7	7	2
Instance $SI9$	7	5	3	Instance $SI18$	7	7	3

Les instances du premier ensemble SI (voir Tableau 3.1) sont de taille réduite, et ont pour but de vérifier la bonne convergence des algorithmes en comparant leur résultat au front de Pareto optimal, celui-ci étant fourni par l'algorithme d'optimisation linéaire présenté dans la section 3.3.2. Ces expérimentations permettront de sélectionner la meilleure combinaison d'opérateurs pour chaque algorithme avant de débuter la phase d'expérimentation sur les instances de grande taille. Les

instances appartenant à l'ensemble LI sont de grande taille (voir Tableau 3.2). L'obtention du front de Pareto optimal n'est plus possible compte tenu du temps d'exécution des solveurs linéaires sur ce problème.

TABLEAU 3.2 – Ensemble d'instances LI

	H	W	R		H	W	R
Instance $LI1$	10	10	1	Instance $LI13$	20	20	1
Instance $LI2$	10	10	2	Instance $LI14$	20	20	2
Instance $LI3$	10	10	3	Instance $LI15$	20	20	3
Instance $LI4$	10	10	4	Instance $LI16$	20	20	4
Instance $LI5$	10	20	1	Instance $LI17$	20	30	1
Instance $LI6$	10	20	2	Instance $LI18$	20	30	2
Instance $LI7$	10	20	3	Instance $LI19$	20	30	3
Instance $LI8$	10	20	4	Instance $LI20$	20	30	4
Instance $LI9$	10	30	1	Instance $LI21$	20	40	1
Instance $LI10$	10	30	2	Instance $LI22$	20	40	2
Instance $LI11$	10	30	3	Instance $LI23$	20	40	3
Instance $LI12$	10	30	4	Instance $LI24$	20	40	4

3.5.3 Métriques de comparaison

Afin de comparer les méthodes d'optimisation multiobjectif, nous utiliserons différentes métriques (Zitzler and Thiele, 1999). En ce qui concerne les instances de petite taille, nous étudierons le nombre de solutions optimales trouvées par chaque algorithme. Soit FP le front de Pareto obtenu par l'optimisation du problème via un algorithme et FPO le front de Pareto optimal du problème, nous définissons cette métrique de la manière suivante :

$$Opt(FP) = \frac{|\{p \in FP \cap FPO\}|}{|FPO|} \tag{3.12}$$

Où la valeur obtenue se situe entre 0 et 1, 0 signifiant qu'aucune solution optimale n'a été trouvée, et 1 signifiant que le front de Pareto optimal a été obtenu dans son intégralité.

Pour ce qui est des instances de grande taille, nous utiliserons les métriques de comparaison classiques dont la première est la métrique C. Soient deux fronts Pareto FP et FP', cette métrique est calculée de la manière suivante :

$$C(FP, FP') = \frac{|\{q \in FP', \exists p \in FP, p \preceq q\}|}{|FP'|} \tag{3.13}$$

Cette métrique permet d'obtenir la dominance de FP sur le second front FP', dont le résultat est compris entre 0 et 1. Une valeur égale à 0 signifie que le front FP ne domine aucune solution de FP', alors qu'une valeur égale à 1 impliquera que le front FP domine complètement le front FP'. Cette métrique n'est pas symétrique, c'est-à-dire que l'égalité $C(FP, FP') = 1 - C(FP', FP)$ n'est pas toujours respectée. C'est pourquoi il est généralement nécessaire de calculer ces deux valeurs pour évaluer les algorithmes. Nous utiliserons également la métrique d'hypervolume S, correspondant

dans le cas d'un problème biobjectif de minimisation à l'aire se trouvant sous le front de Pareto. Son résultat est compris entre 0 et 1, une valeur plus faible impliquant une meilleure convergence.

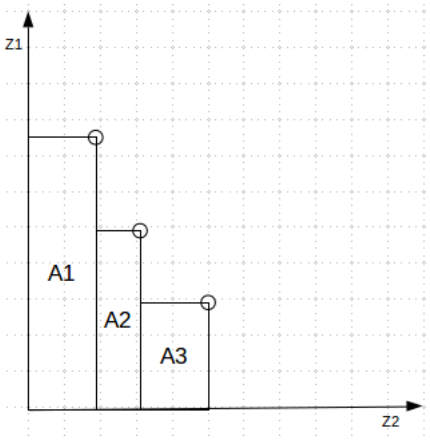

FIGURE 3.7 – Calcul de S

La Figure 3.7 montre un exemple de calcul de la métrique S pour un front donné. Elle consiste simplement en l'addition des aires des rectangles formés par les solutions.

3.5.3.1 Résultats sur les instances SI

Les expérimentations sur l'ensemble d'instances SI ont été menées en comparant les fronts donnés par les différents algorithmes aux fronts optimaux obtenus par optimisation linéaire, via la procédure détaillée dans la section 3.3.2. Cela permet d'une part de vérifier la bonne convergence des algorithmes, mais également de sélectionner les meilleures combinaisons d'opérateurs pour chaque méthode. Les temps d'exécution des trois algorithmes ont été fixés à 60 secondes, et chaque instance a été traitée 10 fois par chaque algorithme. Les résultats de la métrique *Opt* sont décrits dans le Tableau 3.3, où l'on peut trouver les valeurs minimum, moyenne et maximum de cette métrique pour chaque algorithme et chaque instance.

Les résultats montrent que les trois algorithmes trouvent des solutions optimales pour chaque instance. Les algorithmes NSGA-II et SPEA2 semblent plus à l'aise que l'algorithme MOEA/D sur les instances de petite taille. Cela peut s'expliquer par le fait que les deux premiers algorithmes disposent de populations de taille conséquente, dû au paramétrage décrit dans la section précédente. Dans le cas de MOEA/D, il n'existe qu'une seule solution par niche du front de Pareto. Le nombre de niches est relativement faible dans les instances de petite taille, la différence entre les deux points extrêmes du front de Pareto en termes de nombre de capteurs déployés est généralement faible, de l'ordre de quelques dizaines. Cependant, MOEA/D semble mieux adapté pour les dernières instances, où il fournit des performances similaires aux deux autres algorithmes génétiques. En revanche, si MOPSO donne de bons résultats sur les premières instances, les valeurs moyennes de

la métrique tendent à baisser à mesure que la taille de l'instance augmente, tout en gardant des valeurs maximales hautes.

TABLEAU 3.3 – Résultats de la métrique *Opt*

	$SI1$			$SI2$			$SI3$		
	min	avg	max	min	avg	max	min	avg	max
Opt(NSGAII)	0,75	0,92	1,00	0,75	0,75	0,75	1,00	1,00	1,00
Opt(MOEAD)	0,75	0,75	0,75	0,75	0,75	0,75	1,00	1,00	1,00
Opt(SPEA2)	0,75	0,83	1,00	0,75	0,75	0,75	1,00	1,00	1,00
Opt(MOPSO)	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,96	1,00
	$SI4$			$SI5$			$SI6$		
	min	avg	max	min	avg	max	min	avg	max
Opt(NSGAII)	0,80	0,91	1,00	0,40	0,60	0,80	0,88	0,90	1,00
Opt(MOEAD)	0,60	0,69	0,80	0,40	0,64	0,80	0,88	0,92	1,00
Opt(SPEA2)	0,60	0,80	1,00	0,40	0,44	0,60	1,00	1,00	1,00
Opt(MOPSO)	1,00	1,00	1,00	0,60	0,78	1,00	0,62	0,87	1,00
	$SI7$			$SI8$			$SI9$		
	min	avg	max	min	avg	max	min	avg	max
Opt(NSGAII)	0,67	0,69	0,83	0,50	0,63	0,83	0,75	0,86	1,00
Opt(MOEAD)	0,50	0,56	0,67	0,17	0,35	0,67	0,12	0,53	0,75
Opt(SPEA2)	0,67	0,67	0,67	0,33	0,52	0,67	0,25	0,65	0,88
Opt(MOPSO)	0,67	0,98	1,00	0,60	0,81	1,00	0,50	0,65	1,00
	$SI10$			$SI11$			$SI12$		
	min	avg	max	min	avg	max	min	avg	max
Opt(NSGAII)	1,00	1,00	1,00	0,86	0,97	1,00	0,75	0,85	0,88
Opt(MOEAD)	0,50	0,93	1,00	0,86	0,89	1,00	0,38	0,71	0,75
Opt(SPEA2)	1,00	1,00	1,00	0,86	0,98	1,00	0,88	0,88	0,88
Opt(MOPSO)	0,67	1,00	1,00	0,60	0,96	1,00	0,25	0,50	1,00
	$SI13$			$SI14$			$SI15$		
	min	avg	max	min	avg	max	min	avg	max
Opt(NSGAII)	0,43	0,54	0,71	0,33	0,65	1,00	0,50	0,60	0,75
Opt(MOEAD)	0,14	0,29	0,57	0,17	0,44	0,67	0,00	0,29	0,62
Opt(SPEA2)	0,29	0,48	0,71	0,17	0,41	1,00	0,38	0,57	0,75
Opt(MOPSO)	0,57	0,69	1,00	0,50	0,61	1,00	0,25	0,43	1,00
	$SI16$			$SI17$			$SI18$		
	min	avg	max	min	avg	max	min	avg	max
Opt(NSGAII)	0,44	0,54	0,78	0,00	0,33	0,43	0,14	0,56	0,71
Opt(MOEAD)	0,22	0,49	0,67	0,29	0,38	0,43	0,57	0,57	0,57
Opt(SPEA2)	0,44	0,58	0,89	0,43	0,43	0,43	0,57	0,59	0,71
Opt(MOPSO)	0,44	0,53	1,00	0,29	0,37	1,00	0,14	0,27	1,00

3.5.3.2 Résultats sur les instances LI

Les instances de l'ensemble LI ont également été traitées 10 fois par chaque algorithme, dont les temps d'exécution ont été limités à 180 secondes. Nous comparons ici les performances des algorithmes via les métriques décrites précédemment. Au vu du nombre d'instances et d'algorithmes testés, nous commencerons par étudier successivement les résultats des métriques C représentant les dominances des algorithmes NSGA-II, SPEA2, MOEA/D et enfin MOPSO.

Le Tableau 3.4 représente la dominance de NSGA-II sur les algorithmes MOEA/D, SPEA2 et MOPSO lors des expérimentations sur les instances $LI1$ à $LI12$. NSGA-II semble avoir un certain

avantage sur MOEA/D sur les trois premières instances, où les valeurs moyennes de la métrique sont comprises entre 0,46 et 0,69, avec des pics à 0,89. Cependant cet avantage tend à disparaître lorsque la taille de l'instance augmente, tendance que l'on peut notamment observer dans les trois dernières instances, où les valeurs moyennes fournies par la métrique sont respectivement égales à 0,04, 0,04 et 0,1. Ceci peut s'expliquer par le fait que MOEA/D semble tirer avantage de l'augmentation du nombre de variables, impliquant un front de Pareto plus étendu, ce qui implique également un plus grand nombre de niches à occuper et donc un plus grand nombre de solutions.

TABLEAU 3.4 – $C(NSGAII, \_)$ sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,36	0,69	0,87	0,33	0,46	0,67	0,15	0,53	0,89
C(NSGAII,SPEA2)	0,00	0,24	0,57	0,23	0,34	0,62	0,06	0,17	0,41
C(NSGAII,MOPSO)	0,62	0,73	0,76	0,69	0,75	0,77	0,41	0,61	0,76
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,00	0,13	0,40	0,23	0,50	0,86	0,04	0,37	0,75
C(NSGAII,SPEA2)	0,00	0,06	0,33	0,00	0,05	0,18	0,00	0,05	0,4
C(NSGAII,MOPSO)	0,62	0,69	0,85	0,35	0,45	0,53	0,44	0,48	0,62
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,03	0,23	0,53	0,07	0,22	0,37	0,06	0,12	0,34
C(NSGAII,SPEA2)	0,00	0,00	0,03	0,00	0,00	0,00	0,00	0,00	0,00
C(NSGAII,MOPSO)	0,22	0,39	0,61	0,47	0,54	0,67	0,36	0,43	0,50
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,00	0,04	0,12	0,00	0,04	0,13	0,00	0,10	0,21
C(NSGAII,SPEA2)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C(NSGAII,MOPSO)	0,04	0,07	0,17	0,05	0,09	0,12	0,06	0,07	0,12

On peut remarquer une tendance similaire, et encore plus accentuée, en ce qui concerne la dominance de NSGA-II sur SPEA2 ainsi que sur MOPSO. Si NSGA-II domine faiblement SPEA2 sur les trois premières instances, avec des valeurs moyennes de 0,24, 0,34 et 0,17, la dominance est nulle en ce qui concerne les 6 dernières instances. L'ensemble des opérateurs étant similaire pour les deux algorithmes, il semblerait que l'organisation en front et la sélection via la distance de *crowding* de NSGA-II ne surpasse pas la gestion d'archive de SPEA2 sur notre problème. Le Tableau 3.5 présente les résultats de la dominance de NSGA-II sur les 12 dernières instances $LI13$ à $LI24$. Les valeurs fournies par la métrique C sont nulles ou très proches de 0 pour l'ensemble des instances testées, ce qui montre une non-dominance de l'algorithme NSGA-II par rapport aux autres algorithmes. Ces résultats peuvent impliquer deux possibilités : la première est que les algorithmes fournissent des fronts équivalents pour les grandes instances, et donc que les processus de sélection sont équivalents pour notre problème, la deuxième est que le processus de NSGA-II est inférieur à ceux de SPEA2, MOPSO et MOEA/D pour notre problème. Les résultats suivants permettront de confirmer l'une ou l'autre de ces hypothèses.

TABLEAU 3.5 – $C(NSGAII, \_)$ sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,00	0,02	0,09	0,00	0,01	0,02	0,00	0,00	0,02
C(NSGAII,SPEA2)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,02
C(NSGAII,MOPSO)	0,06	0,23	0,33	0,00	0,00	0,04	0,00	0,00	0,00
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,00	0,00	0,02	0,00	0,00	0,01	0,00	0,01	0,04
C(NSGAII,SPEA2)	0,00	0,00	0,00	0,00	0,01	0,02	0,00	0,00	0,00
C(NSGAII,MOPSO)	0,00	0,00	0,00	0,00	0,03	0,06	0,00	0,00	0,00
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,00	0,00	0,00	0,00	0,01	0,05	0,00	0,08	0,24
C(NSGAII,SPEA2)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C(NSGAII,MOPSO)	0,00	0,00	0,00	0,00	0,02	0,11	0,00	0,00	0,00
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,MOEAD)	0,00	0,01	0,03	0,00	0,04	0,15	0,00	0,00	0,00
C(NSGAII,SPEA2)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
C(NSGAII,MOPSO)	0,00	0,00	0,00	0,00	0,04	0,13	0,04	0,06	0,10

TABLEAU 3.6 – $C(SPEA2, \_)$ sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	0,00	0,31	0,58	0,42	0,54	0,75	0,28	0,59	0,83
C(SPEA2,MOEAD)	0,46	0,77	1,00	0,50	0,66	0,83	0,53	0,74	0,84
C(SPEA2,MOPSO)	0,73	0,80	0,83	0,77	0,80	0,85	0,59	0,66	0,76
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	0,29	0,51	0,64	0,62	0,85	1,00	0,43	0,91	1,00
C(SPEA2,MOEAD)	0,13	0,29	0,47	0,91	0,98	1,00	0,61	0,83	0,96
C(SPEA2,MOPSO)	0,77	0,88	0,93	0,47	0,51	0,58	0,62	0,65	0,69
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	0,87	0,95	1,00	0,88	0,95	1,00	1,00	1,00	1,00
C(SPEA2,MOEAD)	0,66	0,90	1,00	0,88	0,90	0,96	0,91	0,98	1,00
C(SPEA2,MOPSO)	0,78	0,83	0,89	0,76	0,84	0,94	0,50	0,54	0,58
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	1,00	1,00	1,00	0,95	1,00	1,00	0,83	0,93	0,97
C(SPEA2,MOEAD)	0,82	0,93	1,00	0,79	0,89	1,00	0,62	0,77	0,86
C(SPEA2,MOPSO)	0,57	0,62	0,68	0,65	0,74	0,83	0,38	0,48	0,53

Le Tableau 3.6 présente les résultats de la dominance de SPEA2 sur les autres algorithmes. Si elle est partielle en ce qui concerne la comparaison entre SPEA2 et NSGA-II sur les quatre premières instances, où les valeurs moyennes de la métrique sont comprises entre 0,31 et 0,51, elle devient forte voire totale sur les instances suivantes. Il semblerait que si les performances des deux algorithmes sont comparables sur les instances 10×10, la gestion de l'archive de SPEA2 prenne l'avantage dès que la taille de la grille augmente. On peut également remarquer que SPEA2 offre de meilleures

performances que MOEA/D, les scores de métriques étant hauts pour la plupart des instances, mais SPEA2 ne semble dominer totalement MOEA/D qu'occasionnellement. Ceci pourrait impliquer que les fronts de Pareto de SPEA2 et MOEA/D sont relativement proches, malgré une dominance prononcée de SPEA2. On remarque une tendance inverse concernant la dominance de SPEA2 sur MOPSO. Si SPEA2 domine fortement MOPSO sur les premières instances, la valeur de la métrique semble s'atténuer à mesure que la taille des instances augmente.

TABLEAU 3.7 – $C(SPEA2, \_)$ sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	0,98	0,99	1,00	1,00	1,00	1,00	0,96	0,99	1,00
C(SPEA2,MOEAD)	0,82	0,93	1,00	0,91	0,96	1,00	0,70	0,80	0,95
C(SPEA2,MOPSO)	0,75	0,79	0,82	0,65	0,71	0,75	0,70	0,75	0,79
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	0,98	0,99	1,00	0,96	0 0,98	1,00	0,98	0,99	0,99
C(SPEA2,MOEAD)	0,82	0,91	0,95	0,90	0,93	0,97	0,84	0,91	0,98
C(SPEA2,MOPSO)	0,57	0,64	0,75	0,70	0,76	0,82	0,55	0,64	0,70
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	1,00	1,00	1,00	0,97	0,98	0,98	0,98	0,99	1,00
C(SPEA2,MOEAD)	0,67	0,76	0,82	0,79	0,88	0,95	0,79	0,89	0,94
C(SPEA2,MOPSO)	0,50	0,56	0,71	0,59	0,70	0,79	0,76	0,79	0,84
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,NSGAII)	0,96	0,98	0,99	0,97	0,98	0,99	0,98	0,98	0,98
C(SPEA2,MOEAD)	0,73	0,84	0,94	0,73	0,82	0,90	0,50	0,65	0,75
C(SPEA2,MOPSO)	0,43	0,57	0,62	0,14	0,31	0,43	0,38	0,47	0,60

Les résultats disponibles dans le Tableau 3.7 confirment ces tendances sur les instances $LI13$ à $LI24$. La dominance de SPEA2 est totale sur NSGA-II, les scores de métrique étant de 1 ou très proches. Les scores de métrique concernant la dominance de SPEA2 sur MOEA/D sont également très hauts, mais cependant plus faibles que ceux concernant la comparaison entre SPEA2 et NSGA-II. MOPSO semble plus compétitif sur les dernières instances, la dominance de SPEA2 étant de plus en plus faible comme le montrent les valeurs moyennes de la métrique C. Il est donc clair que les fronts de Pareto de SPEA2 sont généralement devant ceux fournis par les algorithmes NSGA-II, MOEA/D et MOPSO.

Les Tableaux 3.8 et 3.9 fournissent les résultats concernant la dominance de MOEA/D sur les autres algorithmes. Si les scores concernant la dominance de MOEA/D sur NSGA-II sont faibles pour les premières instances, ils augmentent en même temps que la taille des instances, jusqu'à obtenir une dominance presque totale. En ce qui concerne les scores de dominance par rapport au SPEA2, ils concordent avec les résultats précédemment obtenus sur la métrique inverse : MOEA/D domine de moins en moins SPEA2 sans que toutefois le score de la métrique atteigne zéro. Cela implique une proximité des fronts de Pareto obtenus par ces deux algorithmes. La dominance de

MOEA/D sur MOPSO suit la même tendance que précédemment : si MOPSO est fortement dominé par MOEA/D dans les premières instances, les scores de métrique deviennent plus faibles sur les dernières instances de l'ensemble L.

TABLEAU 3.8 – $C(MOEAD, \_)$ sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,00	0,07	0,36	0,08	0,26	0,50	0,05	0,33	0,69
C(MOEAD,SPEA2)	0,00	0,09	0,36	0,00	0,18	0,38	0,00	0,11	0,24
C(MOEAD,MOPSO)	0,78	0,81	0,82	0,77	0,78	0,85	0,35	0,46	0,65
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,29	0,49	0,64	0,00	0,27	0,57	0,26	0,54	0,96
C(MOEAD,SPEA2)	0,00	0,13	0,33	0,00	0,00	0,00	0,00	0,07	0,19
C(MOEAD,MOPSO)	0,67	0,85	0,93	0,44	0,49	0,53	0,62	0,65	0,75
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,33	0,61	0,84	0,52	0,66	0,88	0,51	0,79	0,94
C(MOEAD,SPEA2)	0,00	0,04	0,13	0,00	0,01	0,08	0,00	0,00	0,03
C(MOEAD,MOPSO)	0,53	0,60	0,79	0,61	0,72	0,78	0,46	0,51	0,54
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,74	0,94	1,00	0,79	0,90	0,98	0,70	0,79	0,97
C(MOEAD,SPEA2)	0,00	0,03	0,10	0,00	0,03	0,10	0,00	0,02	0,14
C(MOEAD,MOPSO)	0,48	0,53	0,59	0,13	0,27	0,42	0,06	0,19	0,33

TABLEAU 3.9 – $C(MOEAD, \_)$ sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,90	0,97	1,00	0,88	0,93	1,00	0,90	0,94	0,98
C(MOEAD,SPEA2)	0,00	0,03	0,10	0,00	0,03	0,08	0,04	0,15	0,25
C(MOEAD,MOPSO)	0,59	0,69	0,77	0,42	0,54	0,61	0,29	0,44	0,52
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,82	0,88	0,93	0,96	0,97	0,99	0,90	0,97	1,00
C(MOEAD,SPEA2)	0,00	0,02	0,06	0,02	0,05	0,09	0,01	0,05	0,10
C(MOEAD,MOPSO)	0,05	0,09	0,19	0,49	0,56	0,70	0,30	0,36	0,38
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,82	0,86	0,90	0,89	0,94	1,00	0,75	0,89	1,00
C(MOEAD,SPEA2)	0,03	0,06	0,12	0,00	0,02	0,04	0,04	0,07	0,11
C(MOEAD,MOPSO)	0,08	0,14	0,25	0,04	0,16	0,26	0,30	0,47	0,76
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(MOEAD,NSGAII)	0,96	0,97	1,00	0,71	0,84	0,93	0,84	0,89	0,92
C(MOEAD,SPEA2)	0,01	0,07	0,14	0,02	0,05	0,08	0,03	0,05	0,06
C(MOEAD,MOPSO)	0,23	0,28	0,34	0,14	0,24	0,35	0,10	0,22	0,37

On peut donc affirmer que les fronts de Pareto, malgré une proximité et des intersections fréquentes, sont généralement classés dans l'ordre suivant : tout d'abord SPEA2, puis MOEA/D et

enfin NSGA-II. Les performances de MOPSO restent floues, mais elles semblent s'améliorer à mesure que la taille de la grille augmente. Les Tableaux 3.10 et 3.11 fournissent les résultats concernant la dominance de MOPSO sur les autres algorithmes. MOPSO ne domine que rarement et partiellement les autres algorithmes sur les douze premières instances. Cependant, sa dominance sur NSGA-II augmente en même temps que la taille des instances. Il en est de même pour les instances suivantes.

TABLEAU 3.10 – $C(MOPSO, \_)$ sur les instances $LI1 - LI12$

	$L1$			$L2$			$L3$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,00	0,01	0,08	0,00	0,09	0,17	0,05	0,25	0,39
C(MOPSO,SPEA2)	0,00	0,00	0,00	0,00	0,03	0,08	0,11	0,19	0,29
C(MOPSO,MOEAD)	0,00	0,01	0,07	0,00	0,07	0,08	0,11	0,22	0,33
	$L4$			$L5$			$L6$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,08	0,16	0,23	0,00	0,02	0,05	0,04	0,12	0,19
C(MOPSO,SPEA2)	0,00	0,02	0,07	0,00	0,00	0,00	0,00	0,00	0,04
C(MOPSO,MOEAD)	0,00	0,03	0,20	0,00	0,00	0,00	0,00	0,03	0,04
	$L7$			$L8$			$L9$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,07	0,20	0,31	0,15	0,20	0,24	0,03	0,05	0,11
C(MOPSO,SPEA2)	0,00	0,01	0,03	0,00	0,01	0,07	0,00	0,00	0,00
C(MOPSO,MOEAD)	0,00	0,02	0,03	0,00	0,01	0,04	0,00	0,00	0,00
	$L10$			$L11$			$L12$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,35	0,46	0,56	0,50	0,55	0,62	0,45	0,50	0,56
C(MOPSO,SPEA2)	0,00	0,01	0,06	0,00	0,00	0,02	0,00	0,00	0,03
C(MOPSO,MOEAD)	0,06	0,07	0,09	0,00	0,01	0,03	0,03	0,09	0,14

TABLEAU 3.11 – $C(MOPSO, \_)$ sur les instances $LI13 - LI24$

	$L13$			$L14$			$L15$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,19	0,36	0,54	0,61	0,67	0,72	0,83	0,86	0,90
C(MOPSO,SPEA2)	0,00	0,00	0,00	0,04	0,06	0,10	0,00	0,02	0,04
C(MOPSO,MOEAD)	0,00	0,02	0,05	0,07	0,11	0,15	0,02	0,07	0,09
	$L16$			$L17$			$L18$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,65	0,72	0,79	0,71	0,75	0,85	0,75	0,82	0,90
C(MOPSO,SPEA2)	0,04	0,07	0,08	0,01	0,02	0,02	0,04	0,05	0,08
C(MOPSO,MOEAD)	0,15	0,18	0,21	0,00	0,03	0,06	0,09	0,11	0,13
	$L19$			$L20$			$L21$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,73	0,78	0,84	0,88	0,91	0,93	0,62	0,64	0,66
C(MOPSO,SPEA2)	0,00	0,00	0,00	0,00	0,01	0,01	0,00	0,00	0,00
C(MOPSO,MOEAD)	0,00	0,00	0,00	0,02	0,08	0,15	0,00	0,00	0,03
	$L22$			$L23$			$L24$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,NSGAII)	0,85	0,87	0,90	0,38	0,48	0,54	0,78	0,85	0,91
C(MOPSO,SPEA2)	0,05	0,06	0,07	0,00	0,01	0,01	0,01	0,02	0,03
C(MOPSO,MOEAD)	0,06	0,08	0,10	0,00	0,01	0,03	0,04	0,12	0,16

TABLEAU 3.12 – Métrique S sur les instances $LI1 - LI24$

	$L1$			$L2$			$L3$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,23	0,24	0,27	0,04	0,06	0,06	0,03	0,04	0,05
S(SPEA2)	0,24	0,25	0,26	0,05	0,05	0,05	0,04	0,04	0,05
S(MOEAD)	0,26	0,27	0,29	0,05	0,06	0,07	0,04	0,05	0,05
S(MOPSO)	0,37	0,37	0,38	0,09	0,09	0,10	0,05	0,05	0,05
	$L4$			$L5$			$L6$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,01	0,01	0,01	0,22	0,24	0,25	0,05	0,06	0,06
S(SPEA2)	0,01	0,01	0,02	0,22	0,23	0,24	0,04	0,05	0,06
S(MOEAD)	0,01	0,02	0,02	0,23	0,24	0,24	0,05	0,06	0,06
S(MOPSO)	0,02	0,02	0,02	0,34	0,37	0,39	0,08	0,09	0,10
	$L7$			$L8$			$L9$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,02	0,03	0,03	0,01	0,01	0,02	0,25	0,25	0,25
S(SPEA2)	0,03	0,03	0,03	0,01	0,01	0,02	0,23	0,23	0,24
S(MOEAD)	0,03	0,03	0,03	0,01	0,01	0,02	0,23	0,24	0,24
S(MOPSO)	0,04	0,04	0,05	0,02	0,02	0,02	0,33	0,34	0,36
	$L10$			$L11$			$L12$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,06	0,06	0,06	0,03	0,03	0,03	0,01	0,01	0,01
S(SPEA2)	0,05	0,05	0,06	0,02	0,03	0,03	0,01	0,01	0,01
S(MOEAD)	0,06	0,06	0,06	0,03	0,03	0,03	0,01	0,01	0,01
S(MOPSO)	0,08	0,09	0,09	0,04	0,04	0,04	0,02	0,02	0,02
	$L13$			$L14$			$L15$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,32	0,32	0,33	0,08	0,09	0,09	0,03	0,03	0,03
S(SPEA2)	0,30	0,30	0,30	0,07	0,08	0,08	0,02	0,03	0,03
S(MOEAD)	0,30	0,30	0,32	0,08	0,08	0,08	0,02	0,03	0,03
S(MOPSO)	0,35	0,37	0,38	0,09	0,09	0,09	0,03	0,04	0,04
	$L16$			$L17$			$L18$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,02	0,02	0,02	0,32	0,32	0,32	0,09	0,09	0,09
S(SPEA2)	0,02	0,02	0,02	0,31	0,31	0,31	0,07	0,08	0,08
S(MOEAD)	0,02	0,02	0,02	0,31	0,31	0,31	0,08	0,08	0,08
S(MOPSO)	0,02	0,02	0,02	0,33	0,34	0,36	0,08	0,09	0,09
	$L19$			$L20$			$L21$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,03	0,03	0,03	0,02	0,02	0,02	0,32	0,32	0,32
S(SPEA2)	0,02	0,02	0,02	0,01	0,01	0,02	0,29	0,29	0,29
S(MOEAD)	0,02	0,02	0,02	0,02	0,02	0,02	0,29	0,29	0,30
S(MOPSO)	0,03	0,03	0,03	0,02	0,02	0,02	0,32	0,34	0,35
	$L22$			$L23$			$L24$		
	min	avg	max	min	avg	max	min	avg	max
S(NSGAII)	0,08	0,09	0,11	0,03	0,03	0,03	0,02	0,02	0,02
S(SPEA2)	0,08	0,08	0,08	0,03	0,03	0,03	0,02	0,02	0,02
S(MOEAD)	0,08	0,08	0,08	0,02	0,02	0,02	0,02	0,02	0,02
S(MOPSO)	0,08	0,09	0,09	0,03	0,03	0,04	0,02	0,02	0,02

Le Tableau 3.12 fournit les résultats de la métrique S. Si les résultats sont en adéquation avec ceux obtenus par la métrique C, on peut cependant déduire la proximité des fronts de Pareto par la faible différence des scores de la métrique. L'évolution des valeurs suit celle observée sur la métrique précédente. On remarque également que les valeurs pour une instance et un algorithme donnés

varient faiblement, la différence entre les valeurs minimum et maximum étant la plupart du temps faible, impliquant la stabilité des algorithmes.

TABLEAU 3.13 – Métrique NDS sur les instances $LI1 - LI24$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	12,00	12,60	14,00	11,00	11,90	12,00	16,00	17,60	19,00
NDS(MOEAD)	13,00	14,30	16,00	12,00	12,50	14,00	17,00	18,50	20,00
NDS(SPEA2)	13,00	13,40	14,00	13,00	13,00	13,00	17,00	17,60	19,00
NDS(MOPSO)	15,00	16,70	18,00	13,00	13,00	13,00	17,00	17,00	17,00
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	13,00	13,80	14,00	21,00	22,00	23,00	21,00	22,80	24,00
NDS(MOEAD)	14,00	14,90	15,00	21,00	22,30	23,00	23,00	23,50	25,00
NDS(SPEA2)	14,00	14,80	15,00	20,00	21,60	23,00	21,00	21,80	23,00
NDS(MOPSO)	12,00	13,10	14,00	17,00	18,70	20,00	16,00	16,10	17,00
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	30,00	31,20	32,00	25,00	26,00	28,00	33,00	34,60	37,00
NDS(MOEAD)	31,00	32,10	34,00	26,00	27,00	28,00	30,00	32,70	36,00
NDS(SPEA2)	31,00	31,80	34,00	25,00	26,30	27,00	30,00	31,50	33,00
NDS(MOPSO)	18,00	18,50	19,00	16,00	16,80	18,00	25,00	26,00	28,00
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	32,00	34,20	35,00	41,00	41,60	42,00	30,00	32,10	34,00
NDS(MOEAD)	33,00	33,60	34,00	37,00	38,60	40,00	28,00	31,10	34,00
NDS(SPEA2)	30,00	31,80	33,00	42,00	42,40	44,00	35,00	35,30	36,00
NDS(MOPSO)	21,00	22,70	23,00	22,00	23,60	25,00	15,00	16,40	17,00
	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	55,00	58,80	62,00	46,00	50,10	54,00	48,00	51,00	54,00
NDS(MOEAD)	51,00	54,60	57,00	40,00	46,20	52,00	36,00	42,70	45,00
NDS(SPEA2)	57,00	57,90	59,00	48,00	49,60	51,00	52,00	52,40	54,00
NDS(MOPSO)	29,00	31,50	33,00	23,00	23,90	25,00	26,00	27,20	28,00
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	41,00	47,40	52,00	76,00	79,70	83,00	61,00	67,20	75,00
NDS(MOEAD)	34,00	38,90	44,00	63,00	66,70	71,00	43,00	53,60	62,00
NDS(SPEA2)	49,00	50,40	51,00	81,00	86,80	90,00	73,00	76,90	79,00
NDS(MOPSO)	20,00	20,40	21,00	31,00	33,80	37,00	29,00	29,70	32,00
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	54,00	61,20	67,00	53,00	58,10	61,00	94,00	98,10	104,00
NDS(MOEAD)	42,00	45,60	49,00	30,00	38,50	43,00	61,00	66,60	71,00
NDS(SPEA2)	73,00	75,50	76,00	70,00	71,20	72,00	107,00	111,80	116,00
NDS(MOPSO)	23,00	24,30	26,00	26,00	26,90	28,00	31,00	35,10	37,00
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
NDS(NSGAII)	70,00	75,20	78,00	59,00	68,40	74,00	54,00	59,80	66,00
NDS(MOEAD)	58,00	64,60	72,00	54,00	64,60	77,00	36,00	44,60	52,00
NDS(SPEA2)	98,00	100,00	101,00	94,00	97,00	98,00	95,00	95,40	96,00
NDS(MOPSO)	32,00	34,40	35,00	21,00	22,30	23,00	27,00	28,40	30,00

Le nombre de solutions non-dominées fournies par les algorithmes est présenté dans le Tableau 3.13. Outre le fait que ce nombre soit croissant en fonction de la taille de l'instance, les tailles

des fronts sont similaires pour chaque algorithme génétique implémenté en ce qui concerne les 12 premières instances. De plus, la faible variation de cette taille semble confirmer la stabilité des algorithmes. Pour ce qui est des douze dernières instances, les résultats montrent que SPEA2 tend à donner plus de solutions que NSGA-II et MOEA/D à mesure que la taille de l'instance augmente, ces derniers offrant un nombre comparable de solutions. MOPSO semble quant à lui avoir des difficultés à fournir un nombre important de solutions, notamment sur les instances de plus grande taille.

3.6 Améliorations des algorithmes

Dans cette section, nous proposons des modifications pour les algorithmes précédemment présentés afin d'accroître leurs performances sur ce problème. Le principe de pondération des objectifs permet en effet de conserver une diversité dans l'espace des objectifs. En s'inspirant des mécanismes de MOEA/D, on associe à chaque solution X de la population deux valeurs λ_1^X et λ_2^X, correspondant à la pondération des objectifs. Ce vecteur doit intervenir dans les différentes étapes des algorithmes. Soit X une solution proposée au problème :

$$\begin{aligned} &X = \{X_1, ..., X_N\}, \\ &\forall i \in \{1, .., N\}, X_i \in \{0, 1\} \\ &\lambda_1^X, \lambda_2^X \in [0, 1] \end{aligned}$$

Le croisement en un point est modifié pour prendre en compte les coefficients λ_1^X et λ_2^X. Les configurations au sein de l'ensemble sélectionné pour la reproduction sont mises en couple en fonction de leur proximité, calculée par la distance de leurs coefficients λ_1^X et λ_2^X. Soit deux configurations choisies X et X', la création des valeurs λ_1 et λ_2 des nouvelles configurations Y et Y' se fait de la manière suivante :

$$\begin{aligned} \lambda_1^Y &= \alpha\lambda_1^X + (1-\alpha)\lambda_1^{X'} \\ \lambda_2^Y &= \alpha\lambda_2^X + (1-\alpha)\lambda_2^{X'} \\ \lambda_1^{Y'} &= (1-\alpha)\lambda_1^X + \alpha\lambda_1^{X'} \\ \lambda_2^{Y'} &= (1-\alpha)\lambda_2^X + \alpha\lambda_2^{X'} \end{aligned}$$

Où α est un réel entre 0 et 1, généré en fonction du point de croisement. Nous proposons également d'hybrider chaque algorithme à une méthode d'optimisation dédiée au SCP. Cette phase interviendra à la place de l'opérateur de réparation. En considérant un individu p, la réparation et l'optimisation se feront en considérant l'équation suivante :

$$\forall i \in P, score_i \leftarrow \lambda_1^X \sum_{j \in P} a_{i,j} + \lambda_2^X \sum_{(j,k) \in P^2, j \neq k} |a_{i,j} - a_{i,k}| \tag{3.14}$$

La réparation s'opère via la méthode décrite dans l'Algorithme 3.2.

Algorithme 3.2 Opérateur d'optimisation

Paramètres $\lambda_1^X, \lambda_2^X, a, P, X$

$NbSat \leftarrow |\{j \in P, \sum_{i \in P} a_{i,j} \cdot X_i \geq 1\}|$

$\forall j \in P, sat_j^{cov} = min(1, \sum_{i \in P} a_{i,j} \cdot X_i)$

$\forall (j,k) \in P^2, j \neq k, sat_{j,k}^{conflict} = min(1, \sum_{i \in P} |a_{i,j} - a_{i,k}| \cdot X_i)$

$\forall i \in P, score_i \leftarrow \lambda_1^X \sum_{j \in P} a_{i,j} \cdot (1 - sat_j^{cov}) + \lambda_2^X \sum_{(j,k) \in P^2, j \neq k} |a_{i,j} - a_{i,k}| \cdot (1 - sat_{j,k}^{conflict})$

Tant que $NbSat < |P|$ **Faire**

$max \leftarrow i \in P \ as \ \nexists j \in P, j \neq i, score_i < score_j$

$X_{max} \leftarrow 1$

Pour tout $j \in P$ **Faire**

Si $a_{max,j} = 1$ et $sat_j = 0$ **Alors**

$\forall i \in P, score_i \leftarrow score_i - \lambda_1^X a_{i,j}$

$sat_j^{cov} = 1$

$NbSat \leftarrow NbSat + 1$

Fin Si

Fin Pour

Pour tout $(j,k) \in P^2, j \neq k$ **Faire**

Si $|a_{max,j} - a_{max,k}| = 1$ et $sat_{j,k}^{conflict} = 0$ **Alors**

$\forall i \in P, score_i \leftarrow score_i - \lambda_2^X |a_{i,j} - a_{i,k}|$

$sat_{j,k}^{conflict} = 1$

Fin Si

Fin Pour

Fin Tant que

Retourne X

Soit $NbSat$ le nombre de contraintes de couverture initialement satisfaites, sat_j^{cov} et $sat_{j,k}^{conflict}$ respectivement les valeurs binaires exprimant la satisfaction de la contrainte de couverture sur j et la satisfaction du conflit sur (j,k). A chaque variable i, on attribue un score $score_i$ dépendant de l'individu considéré. Le premier coefficient λ_1^X favorise l'activation de variables résolvant un grand nombre de contraintes de couverture, alors que le deuxième coefficient λ_2^X favorise les variables intervenant dans les résolutions de conflits de localisation. La première étape est de lister les contraintes de couverture non satisfaites, ainsi que les conflits entre positions non résolus, puis de calculer les scores des variables en considérant l'équation décrite ci-dessus. Puis la procédure suivante est appliquée tant que des contraintes de couverture ne sont pas satisfaites (i.e $NbSat < |P|$) : la variable possédant le score le plus haut est sélectionnée et fixée à 1, puis les contraintes, les conflits et les scores des variables sont actualisés en fonction des contraintes de couverture et des conflits de localisation non satisfaits.

Dans le cas de MOPSO, les coefficients interviennent également dans la sélection du meilleur

voisin et celle de la meilleure position en mémoire. Les particules non-dominées sont triées selon la pondération des objectifs, et celle possédant le score le plus faible est sélectionnée comme représentante. MOPSO dispose également de l'intégration du nouvel opérateur d'optimisation décrit ci-dessus.

Les versions hybridées des algorithmes seront par la suite nommées H-NSGA-II, H-SPEA2 et H-MOPSO.

3.7 Expérimentations II

3.7.1 Résultats sur les instances SI

TABLEAU 3.14 – Résultats de la métrique *Opt*

	$SI1$			$SI2$			$SI3$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	1,00	1,00	1,00	0,75	0,75	0,75	1,00	1,00	1,00
Opt(H-SPEA2)	0,75	0,81	1,00	0,75	0,75	0,75	1,00	1,00	1,00
Opt(H-MOPSO)	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,97	1,00
	$SI4$			$SI5$			$SI6$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,80	0,80	0,80	0,80	0,84	1,00	1,00	1,00	1,00
Opt(H-SPEA2)	0,60	0,73	0,80	0,00	0,36	0,80	0,88	0,96	1,00
Opt(H-MOPSO)	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,90	1,00
	$SI7$			$SI8$			$SI9$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,67	0,67	0,67	0,50	0,63	0,83	0,88	0,97	1,00
Opt(H-SPEA2)	0,67	0,67	0,67	0,50	0,57	0,83	0,12	0,51	0,75
Opt(H-MOPSO)	0,83	0,98	1,00	0,83	0,96	1,00	0,88	0,97	1,00
	$SI10$			$SI11$			$SI12$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	1,00	1,00	1,00	1,00	1,00	1,00	0,88	0,88	0,88
Opt(H-SPEA2)	0,83	0,96	1,00	1,00	1,00	1,00	0,75	0,85	0,88
Opt(H-MOPSO)	1,00	1,00	1,00	1,00	1,00	1,00	0,75	0,85	1,00
	$SI13$			$SI14$			$SI15$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,43	0,56	0,71	0,83	0,98	1,00	0,62	0,90	1,00
Opt(H-SPEA2)	0,43	0,46	0,57	0,33	0,67	1,00	0,38	0,54	0,75
Opt(H-MOPSO)	0,71	0,86	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	$SI16$			$SI17$			$SI18$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,44	0,73	1,00	0,43	0,48	0,71	0,57	0,70	0,86
Opt(H-SPEA2)	0,56	0,65	0,78	0,43	0,44	0,57	0,57	0,63	0,71
Opt(H-MOPSO)	0,89	0,90	1,00	0,71	0,83	1,00	0,86	0,89	1,00

Les expérimentations sur l'ensemble d'instances SI ont été réalisées en comparant les fronts donnés par les différents algorithmes aux fronts optimaux obtenus par optimisation linéaire, via la procédure détaillée dans la section 3.3.2.

Les résultats disponibles dans le Tableau 3.14 montrent une nette amélioration pour les algo-

rithmes H-NSGA-II et H-MOPSO par rapport aux versions originales. Les proportions de solutions optimales trouvées ont augmenté pour toutes les instances données, ainsi que le nombre d'instances résolues de manière optimale (i.e. le front de Pareto optimal est trouvé dans son intégralité). H-SPEA2 ne présente pas d'amélioration particulière quant à l'hybridation. Les résultats semblent les mêmes pour la plupart des instances, à l'exception des dernières où l'on peut observer une faible amélioration. L'algorithme H-MOPSO est la méthode qui semble la plus puissante concernant la résolution des instances de l'ensemble SI.

3.7.2 Résultats sur les instances LI

Dans cette partie, nous commencerons par comparer les algorithmes NSGA-II, SPEA2 et MOPSO à leurs versions hybridées via la métrique C, puis nous comparerons les algorithmes H-NSGA-II, H-SPEA2 et H-MOPSO à l'algorithme SPEA2, et enfin nous procéderons à la comparaison des algorithmes hybridés les uns par rapport aux autres.

TABLEAU 3.15 – Métrique C sur NSGA-II et H-NSGA-II sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,03	0,25	0,00	0,00	0,00	0,06	0,12	0,28
C(H-NSGAII,NSGAII)	0,33	0,72	1,00	0,83	0,89	0,92	0,42	0,69	0,89
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,11	0,40	0,00	0,00	0,00	0,00	0,00	0,00
C(H-NSGAII,NSGAII)	0,29	0,53	0,64	0,90	0,98	1,00	0,95	0,97	1,00
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,00	0,03	0,00	0,00	0,04	0,00	0,00	0,00
C(H-NSGAII,NSGAII)	0,93	0,99	1,00	0,81	0,95	1,00	1,00	1,00	1,00
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,03
C(H-NSGAII,NSGAII)	0,97	1,00	1,00	0,90	0,95	1,00	0,83	0,91	0,97

Le Tableau 3.15 recense les résultats de la métrique C appliquée aux algorithmes NSGA-II et H-NSGA-II sur les 12 premières instances. Une nette amélioration est visible sur toutes les instances, et la dominance de H-NSGA-II sur NSGA-II est presque totale à partir de l'instance $LI5$. Il semble donc que l'hybridation offre une meilleure convergence à NSGA-II, permettant une meilleure exploration des différentes zones du front de Pareto. Ces résultats sont confirmés par ceux donnés dans le Tableau 3.16, où la dominance de H-NSGA-II est totale sur toutes les 12 plus grandes instances. Les résultats exposés dans le Tableau 3.17 indiquent les valeurs de la métrique C appliquée à SPEA2 et H-SPEA2. Ces résultats nous indiquent que l'opérateur n'a pas eu l'effet escompté, l'avantage allant à l'un ou l'autre des algorithmes selon l'instance traitée. Le fait que l'opérateur n'aide pas à la convergence de SPEA2 peut s'expliquer par le nombre d'individus générés à chaque itération de l'algorithme.

En effet, il est nécessaire de créer une nouvelle population complète à chaque génération, ce qui correspond à 100 individus selon les paramètres choisis. L'opérateur d'optimisation et réparation nécessite plus de temps de calcul que l'opérateur de réparation classique, et est appliqué à chaque nouvel individu afin de garantir sa faisabilité. Or le temps d'exécution total est partagé entre le temps de calcul consommé dans l'optimisation des individus et le temps attribué au processus évolutionnaire de l'algorithme. On peut donc conclure que l'opérateur d'optimisation ne permet pas de rattraper la convergence perdue par la diminution du nombre d'itérations total effectué par l'algorithme dans le temps imparti.

TABLEAU 3.16 – Métrique C sur NSGA-II et H-NSGA-II sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,00	0,00	0,00	0,00	0,02	0,00	0,00	0,00
C(H-NSGAII,NSGAII)	0,98	0,99	1,00	0,98	1,00	1,00	0,96	0,97	0,98
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,00	0,02	0,00	0,00	0,02	0,00	0,00	0,01
C(H-NSGAII,NSGAII)	0,96	0,98	1,00	0,99	1,00	1,00	0,98	1,00	1,00
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,00	0,00	0,00	0,00	0,01	0,00	0,00	0,00
C(H-NSGAII,NSGAII)	1,00	1,00	1,00	0,95	0,97	1,00	1,00	1,00	1,00
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(NSGAII,H-NSGAII)	0,00	0,00	0,01	0,00	0,00	0,01	0,00	0,00	0,00
C(H-NSGAII,NSGAII)	0,99	0,99	1,00	0,96	0,98	0,99	0,98	0,99	1,00

TABLEAU 3.17 – Métrique C sur SPEA2 et H-SPEA2 sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,00	0,16	0,54	0,00	0,17	0,46	0,12	0,21	0,35
C(H-SPEA2,SPEA2)	0,15	0,41	0,71	0,23	0,67	0,92	0,06	0,28	0,67
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,00	0,18	0,40	0,00	0,17	0,68	0,00	0,28	0,81
C(H-SPEA2,SPEA2)	0,07	0,21	0,53	0,25	0,66	1,00	0,00	0,52	0,91
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,16	0,30	0,52	0,12	0,50	0,96	0,00	0,26	0,50
C(H-SPEA2,SPEA2)	0,29	0,52	0,71	0,00	0,28	0,69	0,31	0,58	0,94
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,16	0,41	0,65	0,43	0,57	0,86	0,17	0,37	0,57
C(H-SPEA2,SPEA2)	0,29	0,45	0,73	0,07	0,23	0,40	0,03	0,07	0,14

La même irrégularité peut être observée dans les résultats de métrique sur les instances $LI13$ à $LI24$ (voir Tableau 3.18). On remarque cependant que l'algorithme H-SPEA2 semble plus à l'aise

sur les instances $LI13$, $LI17$ et $LI21$. Le point commun de ces instances est la taille de rayon $R = 1$, ce qui implique une plus faible densité de la matrice (a) ainsi qu'un nombre plus faible de conflits possibles. Ce dernier point implique un temps de calcul plus faible dédié à l'opérateur d'optimisation, notamment dans la boucle de mise à jour des conflits. Cette hypothèse est renforcée par le fait que plus le rayon augmente, plus la dominance de SPEA2 sur H-SPEA2 tend à augmenter.

TABLEAU 3.18 – Métrique C sur SPEA2 et H-SPEA2 sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,00	0,06	0,20	0,18	0,55	0,96	0,67	0,86	0,96
C(H-SPEA2,SPEA2)	0,66	0,86	0,97	0,04	0,36	0,73	0,00	0,07	0,15
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,62	0,77	0,90	0,01	0,16	0,46	0,29	0,47	0,65
C(H-SPEA2,SPEA2)	0,02	0,12	0,33	0,50	0,81	0,98	0,34	0,48	0,69
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,40	0,66	0,86	0,49	0,72	0,83	0,00	0,13	0,64
C(H-SPEA2,SPEA2)	0,01	0,15	0,36	0,06	0,16	0,38	0,36	0,85	1,00
	$LI22$			$LI23$			$LI24$		
19	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-SPEA2)	0,02	0,23	0,39	0,10	0,23	0,48	0,47	0,56	0,71
C(H-SPEA2,SPEA2)	0,46	0,68	0,94	0,40	0,67	0,76	0,08	0,18	0,27

TABLEAU 3.19 – Métrique C sur MOPSO et H-MOPSO sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,H-MOPSO)	0,00	0,10	0,13	0,00	0,08	0,17	0,00	0,06	0,12
C(H-MOPSO,MOPSO)	0,73	0,80	0,88	0,77	0,85	0,922	0,71	0,75	0,82
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,H-MOPSO)	0,00	0,05	0,14	0,04	0,07	0,11	0,04	0,07	0,12
C(H-MOPSO,MOPSO)	0,79	0,88	0,92	0,71	0,79	0,84	0,71	0,83	0,94
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,H-MOPSO)	0,00	0,05	0,07	0,04	0,06	0,08	0,00	0,04	0,07
C(H-MOPSO,MOPSO)	0,58	0,70	0,79	0,59	0,68	0,75	0,81	0,86	0,92
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,H-MOPSO)	0,00	0,03	0,09	0,00	0,04	0,07	0,03	0,05	0,08
C(H-MOPSO,MOPSO)	0,78	0,85	0,91	0,68	0,75	0,83	0,29	0,46	0,59

Le Tableau 3.19 recense les résultats de la métrique C appliquée aux algorithmes MOPSO et H-MOPSO sur les 12 premières instances. Une nette amélioration est visible sur toutes les instances, avec des valeurs moyennes de $C(H - MOPSO, MOPSO)$ rarement inférieures à 0,7.

TABLEAU 3.20 – Métrique C sur MOPSO et H-MOPSO sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,H-MOPSO)	0,00	0,03	0,07	0,00	0,03	0,05	0,00	0,03	0,06
C(H-MOPSO,MOPSO)	0,69	0,73	0,77	0,75	0,82	0,88	0,77	0,84	0,89
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
é C(MOPSO,H-MOPSO)	0,04	0,07	0,09	0,01	0,03	0,06	0,00	0,04	0,10
C(H-MOPSO,MOPSO)	0,65	0,70	0,76	0,62	0,68	0,72	0,62	0,81	0,90
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,H-MOPSO)	0,00	0,03	0,06	0,00	0,05	0,08	0,00	0,01	0,02
24 C(H-MOPSO,MOPSO)	0,46	0,58	0,68	0,43	0,53	0,67	0,64	0,74	0,81
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(MOPSO,H-MOPSO)	0,00	0,04	0,10	0,00	0,02	0,05	0,01	0,04	0,07
C(H-MOPSO,MOPSO)	0,65	0,78	0,91	0,22	0,41	0,57	0,36	0,37	0,38

Il semble donc que l'opérateur modifié cause une meilleure convergence au MOPSO, permettant une meilleure exploration des différentes zones du front de Pareto. Ces résultats sont confirmés par ceux donnés dans le Tableau 3.20, même si les scores de métrique tendent à être plus faibles que pour les instances précédentes.

TABLEAU 3.21 – Métrique C sur SPEA2 et H-NSGA-II sur les instances $LI1 - LI12$

	$L1$			$L2$			$L3$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,00	0,05	0,08	0,00	0,03	0,17	0,00	0,20	0,50
C(H-NSGAII,SPEA2)	0,15	0,65	0,93	0,54	0,88	1,00	0,17	0,39	0,72
	$L4$			$L5$			$L6$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,00	0,18	0,33	0,00	0,01	0,14	0,00	0,08	0,48
C(H-NSGAII,SPEA2)	0,07	0,22	0,40	0,67	0,91	1,00	0,36	0,79	1,00
	$L7$			$L8$			$L9$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,00	0,12	0,27	0,04	0,35	0,76	0,00	0,01	0,07
C(H-NSGAII,SPEA2)	0,58	0,76	0,88	0,04	0,43	0,88	0,77	0,94	1,00
	$L10$			$L11$			$L12$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,00	0,05	0,24	0,19	0,30	0,45	0,11	0,22	0,37
C(H-NSGAII,SPEA2)	0,71	0,90	1,00	0,40	0,58	0,71	0,08	0,18	0,28

Nous nous intéressons désormais à la comparaison entre les algorithmes hybridés et SPEA2, qui nous a semblé être le plus concurrentiel lors de la première phase d'expérimentations. Le Tableau 3.21 présente la comparaison des algorithmes SPEA2 et H-NSGA-II sur les 12 premières instances. Les résultats de la métrique $C(H-NSGAII, SPEA2)$ sont en moyenne équivalents ou meilleurs que ceux de la métrique inverse. On remarque cependant la même tendance que lors de la comparaison entre SPEA2 et sa version hybridée, SPEA2 semblant plus à l'aise avec les instances possédant un

rayon de grande taille.

TABLEAU 3.22 – Métrique C sur SPEA2 et H-NSGA-II sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,00	0,10	0,43	0,00	0,28	0,63	0,39	0,53	0,67
C(H-NSGAII,SPEA2)	0,57	0,83	1,00	0,38	0,69	0,96	0,31	0,44	0,60
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,37	0,51	0,72	0,05	0,29	0,44	0,11	0,27	0,50
C(H-NSGAII,SPEA2)	0,27	0,46	0,57	0,51	0,66	0,91	0,44	0,69	0,89
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,51	0,61	0,67	0,41	0,54	0,66	0,06	0,18	0,35
C(H-NSGAII,SPEA2)	0,29	0,32	0,36	0,27	0,39	0,54	0,65	0,79	0,92
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-NSGAII)	0,13	0,31	0,45	0,25	0,48	0,60	0,43	0,57	0,62
C(H-NSGAII,SPEA2)	0,51	0,62	0,74	0,33	0,45	0,70	0,20	0,27	0,39

La tendance se poursuit pour les douze dernières instances, dont les résultats sont disponibles dans le Tableau 3.22. L'opérateur d'optimisation proposé permet donc à H-NSGA-II de rattraper son retard sur SPEA2, sans toutefois parvenir à le dominer. NSGA-II est plus à l'aise sur les instances où le rayon R est fixé à 1 ou à 2. L'hypothèse appliquée à H-SPEA2 est valable ici : le temps de calcul requis pour les instances avec un rayon de détection plus grand implique plus de temps de calcul pour l'opérateur d'optimisation.

TABLEAU 3.23 – Métrique C sur SPEA2 et H-MOPSO sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,07	0,19	0,33	0,00	0,02	0,08	0,06	0,17	0,29
C(H-MOPSO,SPEA2)	0,15	0,58	0,79	0,92	0,96	1,00	0,39	0,59	0,67
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,14	0,29	0,43	0,07	0,20	0,45	0,00	0,00	0,04
C(H-MOPSO,SPEA2)	0,36	0,53	0,73	0,14	0,50	0,81	0,81	0,96	1,00
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,32	0,46	0,57	0,12	0,23	0,38	0,03	0,20	0,46
C(H-MOPSO,SPEA2)	0,45	0,53	0,61	0,60	0,75	0,85	0,27	0,51	0,81
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,00	0,00	0,00	0,14	0,37	0,55	0,30	0,39	0,46
C(H-MOPSO,SPEA2)	1,00	1,00	1,00	0,33	0,50	0,69	0,37	0,45	0,49

Les résultats de la comparaison entre les algorithmes H-MOPSO et SPEA2 sont disponibles dans les Tableaux 3.23 et 3.24. Ici l'algorithme H-MOPSO prend visiblement l'avantage, notamment sur les instances où le rayon de couverture $R = 2$. Les valeurs moyennes de la métrique $C(H -$

$MOPSO, SPEA2$) sont supérieures à celles de la métrique $C(SPEA2, H-MOPSO)$ sur toutes les instances traitées, et l'écart tend à croître à mesure que la taille des instances augmente. Compte tenu des comparaisons précédentes par rapport à SPEA2, il semble ici que H-MOPSO soit le plus adapté pour notre problème, cependant des tests supplémentaires sont nécessaires afin de confirmer cette hypothèse.

TABLEAU 3.24 – Métrique C sur SPEA2 et H-MOPSO sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,15	0,31	0,47	0,00	0,02	0,07	0,12	0,22	0,33
C(H-MOPSO,SPEA2)	0,46	0,63	0,85	0,94	0,98	1,00	0,63	0,75	0,83
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,07	0,18	0,31	0,05	0,09	0,18	0,00	0,01	0,04
C(H-MOPSO,SPEA2)	0,67	0,80	0,90	0,81	0,89	0,93	0,96	0,99	1,00
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,14	0,20	0,27	0,23	0,33	0,44	0,02	0,04	0,07
C(H-MOPSO,SPEA2)	0,71	0,76	0,88	0,56	0,65	0,72	0,89	0,92	0,95
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(SPEA2,H-MOPSO)	0,00	0,01	0,03	0,13	0,18	0,23	0,22	0,26	0,31
C(H-MOPSO,SPEA2)	0,97	0,99	1,00	0,74	0,78	0,83	0,61	0,63	0,67

TABLEAU 3.25 – Métrique C sur H-SPEA2 et H-NSGA-II sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,00	0,08	0,25	0,00	0,10	0,33	0,00	0,17	0,44
C(H-NSGAII,H-SPEA2)	0,00	0,49	0,92	0,25	0,60	1,00	0,18	0,37	0,59
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,00	0,10	0,40	0,00	0,06	0,35	0,00	0,12	0,43
C(H-NSGAII,H-SPEA2)	0,00	0,12	0,33	0,50	0,83	1,00	0,38	0,76	1,00
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,00	0,17	0,47	0,00	0,25	0,54	0,00	0,05	0,31
C(H-NSGAII,H-SPEA2)	0,42	0,69	0,91	0,24	0,59	0,84	0,50	0,86	1,00
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,00	0,05	0,18	0,07	0,22	0,47	0,00	0,13	0,26
C(H-NSGAII,H-SPEA2)	0,69	0,86	1,00	0,45	0,66	0,77	0,09	0,34	0,57

Nous nous intéressons maintenant à la comparaison entre les différents algorithmes hybrides, afin de déterminer le ou les meilleurs pour l'optimisation de notre problème. Les Tableaux 3.25 et 3.26 comparent les algorithmes H-NSGA-II et H-SPEA2 au travers des 24 instances de l'ensemble LI. Les écarts entre les scores de métriques C ne sont pas suffisamment élevés pour départager les deux algorithmes. Cependant les valeurs moyennes de la métrique $C(H-NSGAII, H-SPEA2)$

sont généralement plus élevées que celles de la métrique inverse, même si le cas contraire est présent dans les instances $LI19$ et $LI23$. On peut donc affirmer que H-NSGA-II présente un léger avantage par rapport à la version hybridée de SPEA2.

TABLEAU 3.26 – Métrique C sur H-SPEA2 et H-NSGA-II sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,13	0,35	0,46	0,02	0,21	0,38	0,04	0,30	0,38
C(H-NSGAII,H-SPEA2)	0,56	0,64	0,79	0,61	0,76	0,92	0,49	0,64	0,89
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,16	0,34	0,42	0,30	0,49	0,62	0,11	0,33	0,55
C(H-NSGAII,H-SPEA2)	0,53	0,62	0,73	0,35	0,49	0,65	0,42	0,64	0,87
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,35	0,52	0,61	0,10	0,35	0,42	0,20	0,44	0,71
C(H-NSGAII,H-SPEA2)	0,31	0,39	0,51	0,45	0,55	0,78	0,30	0,53	0,76
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-NSGAII)	0,14	0,40	0,69	0,37	0,65	0,80	0,22	0,39	0,53
C(H-NSGAII,H-SPEA2)	0,30	0,58	0,85	0,16	0,31	0,66	0,28	0,39	0,51

TABLEAU 3.27 – Métrique C sur H-SPEA2 et H-MOPSO sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,13	0,23	0,47	0,00	0,06	0,15	0,06	0,19	0,29
C(H-MOPSO,H-SPEA2)	0,08	0,45	0,69	0,58	0,87	1,00	0,47	0,62	0,76
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,07	0,30	0,71	0,17	0,42	0,67	0,00	0,03	0,13
C(H-MOPSO,H-SPEA2)	0,07	0,48	0,73	0,00	0,24	0,59	0,81	0,92	1,00
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,32	0,45	0,62	0,12	0,19	0,28	0,15	0,39	0,69
C(H-MOPSO,H-SPEA2)	0,32	0,50	0,61	0,68	0,79	0,84	0,07	0,34	0,66
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,00	0,00	0,03	0,11	0,24	0,33	0,26	0,32	0,39
C(H-MOPSO,H-SPEA2)	0,97	0,99	1,00	0,40	0,62	0,74	0,43	0,52	0,60

On retrouve une situation similaire dans le Tableau 3.27, où la comparaison des algorithmes H-SPEA2 et H-MOPSO ne montre pas d'avantage particulier sur les premières instances. Cependant H-MOPSO présente des scores de métrique moyens plus élevés sur la plupart des instances. L'écart se creuse sur les 12 dernières instances, mise à part l'instance $LI13$ (voir Tableau 3.28). On peut également remarquer que H-MOPSO réussit mieux en règle générale les instances où le rayon de détection R est fixé à 2. La dernière comparaison est présentée dans les Tableaux 3.29 et 3.30. Il s'agit des scores de métrique entre les algorithmes H-NSGA-II et H-MOPSO. Dans le premier

Tableau, nous constatons qu'il n'est pas possible de déterminer lequel des algorithmes est le plus performant : d'une instance à l'autre, l'algorithme avantagé varie, sans que toutefois l'un domine totalement le second. Le second Tableau présente des scores de métrique plus en faveur de H-MOPSO, qui augmentent en même temps que la taille des instances.

TABLEAU 3.28 – Métrique C sur H-SPEA2 et H-MOPSO sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,35	0,60	0,88	0,00	0,02	0,05	0,00	0,06	0,16
C(H-MOPSO,H-SPEA2)	0,06	0,30	0,58	0,96	0,98	1,00	0,85	0,94	1,00
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,00	0,05	0,09	0,11	0,26	0,54	0,00	0,01	0,04
C(H-MOPSO,H-SPEA2)	0,88	0,91	0,98	0,41	0,70	0,91	0,96	0,99	1,00
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,02	0,12	0,18	0,17	0,24	0,35	0,02	0,13	0,27
C(H-MOPSO,H-SPEA2)	0,76	0,83	0,92	0,61	0,69	0,78	0,67	0,81	0,93
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(H-SPEA2,H-MOPSO)	0,00	0,01	0,03	0,11	0,21	0,31	0,16	0,20	0,22
C(H-MOPSO,H-SPEA2)	0,97	0,99	1,00	0,63	0,74	0,83	0,62	0,66	0,69

TABLEAU 3.29 – Métrique C sur H-NSGA-II et H-MOPSO sur les instances $LI1 - LI12$

	$LI1$			$LI2$			$LI3$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,13	0,35	0,53	0,00	0,13	0,33	0,12	0,28	0,50
C(H-MOPSO,H-NSGAII)	0,00	0,13	0,25	0,50	0,73	0,92	0,35	0,50	0,78
	$LI4$			$LI5$			$LI6$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,14	0,31	0,43	0,59	0,67	0,81	0,12	0,34	0,57
C(H-MOPSO,H-NSGAII)	0,33	0,45	0,53	0,00	0,03	0,19	0,19	0,47	0,81
	$LI7$			$LI8$			$LI9$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,48	0,58	0,71	0,12	0,28	0,52	0,59	0,69	0,78
C(H-MOPSO,H-NSGAII)	0,23	0,38	0,50	0,40	0,68	0,88	0,00	0,03	0,10
	$LI10$			$LI11$			$LI12$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,18	0,29	0,36	0,39	0,54	0,67	0,32	0,39	0,43
C(H-MOPSO,H-NSGAII)	0,48	0,62	0,79	0,26	0,39	0,56	0,37	0,46	0,54

Cette dernière phase nous permet de dire que si l'hybridation est réussie, il est désormais difficile de confirmer la supériorité de l'un ou l'autre des algorithmes. Cependant, H-MOPSO montre plus d'aisances sur les instances de grande taille que les autres algorithmes, même si H-NSGA-II prend l'avantage dans certaines instances de l'ensemble LI.

TABLEAU 3.30 – Métrique C sur H-NSGA-II et H-MOPSO sur les instances $LI13 - LI24$

	$LI13$			$LI14$			$LI15$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,58	0,65	0,69	0,17	0,32	0,56	0,24	0,42	0,59
C(H-MOPSO,H-NSGAII)	0,29	0,35	0,42	0,49	0,69	0,82	0,46	0,61	0,75
	$LI16$			$LI17$			$LI18$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,16	0,32	0,41	0,43	0,50	0,59	0,00	0,03	0,12
C(H-MOPSO,H-NSGAII)	0,60	0,70	0,79	0,47	0,54	0,60	0,88	0,96	1,00
	$LI19$			$LI20$			$LI21$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,17	0,25	0,31	0,34	0,39	0,44	0,18	0,37	0,58
C(H-MOPSO,H-NSGAII)	0,69	0,75	0,84	0,56	0,61	0,67	0,43	0,61	0,78
	$LI22$			$LI23$			$LI24$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,00	0,02	0,05	0,16	0,23	0,38	0,16	0,24	0,32
C(H-MOPSO,H-NSGAII)	0,96	0,98	1,00	0,64	0,77	0,83	0,57	0,60	0,63

3.8 Conclusion et perspectives

Dans ce chapitre, nous avons proposé une nouvelle problématique de déploiement et localisation, exprimée sous la forme d'un problème d'optimisation biobjectif. La formulation mathématique linéaire nous a permis la construction des fronts de Pareto optimaux pour les instances de petite taille, et quatre algorithmes multiobjectif ont été implémentés et comparés sur les instances de grande taille. Lors de la première séance d'expérimentation, il apparut que l'algorithme SPEA2 est le meilleur en ce qui concerne l'optimisation de notre problème. Dans un deuxième temps, nous avons proposé une hybridation des algorithmes via une heuristique constructive pondérée afin d'accroître les performances des algorithmes. Une amélioration a été notée pour la plupart des algorithmes à l'exception de SPEA2, et la deuxième séance d'expérimentation nous a permis de constater que les algorithmes hybridés H-MOPSO et H-NSGA-II sont les meilleurs quant à l'optimisation de ce problème.

De nombreuses pistes s'ouvrent à la suite de ce travail, notamment du point de vue de la modélisation de la zone à surveiller. L'intégration des obstacles, ainsi que la prise en compte des préférences sur les différents points de la grille, sont deux perspectives permettant un intérêt concernant les applications de ce travail. Le premier point s'explique par le fait que le signal est généralement sensible aux obstacles, ce qui influe sur les portées des capteurs utilisés. La deuxième perspective présente un intérêt quant à la localisation de cible, une position se trouvant dans une partie fréquentée de la zone ayant généralement plus d'importance qu'une autre.

Les perspectives concernant la modification des méthodes utilisées sont nombreuses, mais la plus intéressante de notre point de vue est l'hybridation des algorithmes. Ici nous utilisons une heuristique constructive dédiée au SCP, qui permet l'optimisation de la solution en un temps relati-

vement rapide. Cependant, le nombre de méthodes dédiées au SCP étant grand, l'expérimentation de nouvelles adaptations de ces méthodes pour l'hybridation est intéressante.

Chapitre 4

Localisation par zonage et déploiement de capteurs

4.1 Introduction

Dans le chapitre précédent, nous avons présenté une nouvelle modélisation permettant la prise en compte d'exigences des applications de localisation et de *tracking* de cible via un objectif de minimisation des ensembles de détection. Nous nous intéressons entre autres à une application potentielle concernant l'application *indoor*, c'est-à-dire à l'intérieur de bâtiments accueillant des visiteurs tels que les hôpitaux, les musées, les centres commerciaux ou encore les établissements publics. Dans ce chapitre, l'objectif est de proposer un déploiement de capteurs permettant la localisation d'un utilisateur tout en minimisant le nombre de capteurs déployés, ainsi qu'en prenant en compte une nouvelle définition de la précision.

Contrairement à la modélisation précédente, nous nous intéresserons au rapport entre la précision et le zonage. Nous définirons le zonage comme étant le pré-découpage de la zone à couvrir en un ensemble de sous-zones, telles que des salles, des couloirs, ... L'approche utilisée ici propose une localisation par zone, l'utilisateur devra donc être localisé avec un certain degré d'incertitude comme étant actuellement dans une zone. Dans ce chapitre, nous présentons un nouveau modèle biobjectif ainsi que l'adaptation de deux méthodes précédemment utilisées dans le chapitre 3 afin de proposer un compromis entre le coût de déploiement et la qualité de la localisation. Nous proposons également une heuristique spécifique au problème étudié, dont nous comparons les performances à celles des autres algorithmes implémentés.

4.2 Description du problème

Tout comme pour le chapitre 3, nous cherchons à proposer un déploiement de capteurs dans une zone optimisant la précision de la localisation. La Figure 4.1 représente un exemple de bâtiment à couvrir, divisé en 4 zones $\{Salle1, Salle2, Salle3, Couloir1\}$.

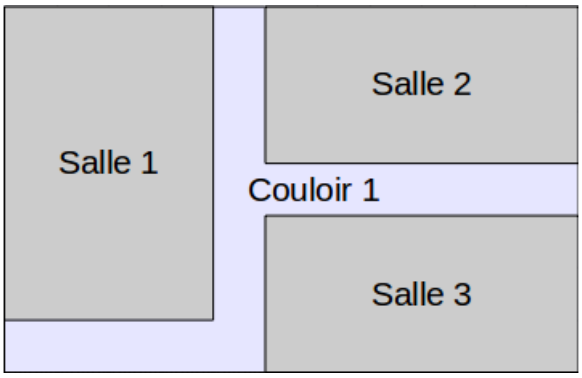

FIGURE 4.1 – Exemple d'un bâtiment à couvrir

La première étape est la discrétisation de la zone en un ensemble de positions P. Le résultat de la discrétisation du bâtiment en une grille de 11×7 est présenté dans la Figure 4.2.

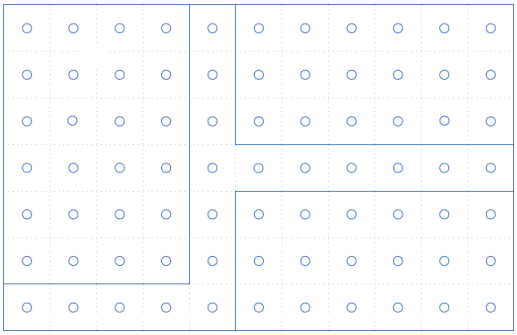

FIGURE 4.2 – Discrétisation du bâtiment

Chaque position de P peut accueillir un capteur, et chaque position doit être couverte par au moins un capteur. La Figure 4.3 montre un exemple de déploiement sur la grille précédemment présentée.

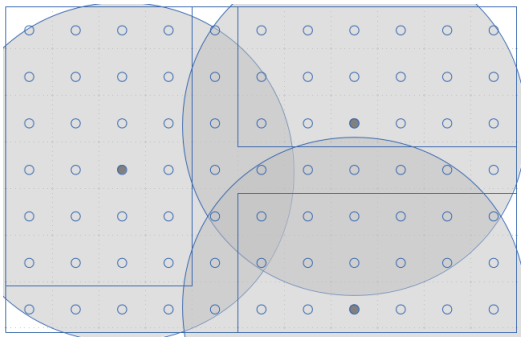

FIGURE 4.3 – Exemple de déploiement sur le bâtiment

Les positions de couleur sombre représentent les capteurs déployés, et sont associées à un disque représentant la couverture binaire de chaque capteur. Si dans le chapitre 3, la précision de la localisation d'un événement dans la grille était modélisée comme étant la taille de l'ensemble de positions correspondant à l'estimation de la position de la cible, la définition du zonage induit ici une nouvelle définition de la localisation. En effet, nous considérons qu'il n'est plus nécessaire de localiser la cible sur une position unique, mais de définir la zone à laquelle elle appartient.

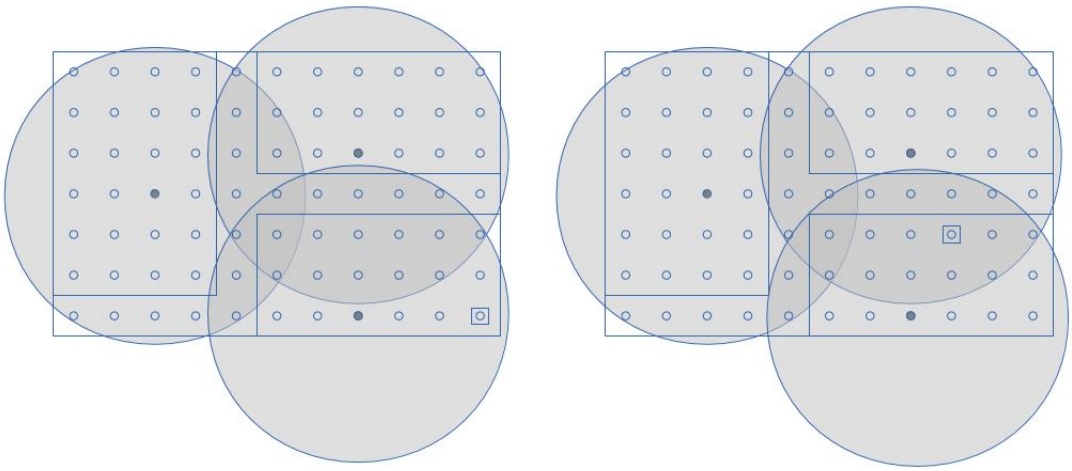

Figure 4.4.A : Détection de la cible A Figure 4.4.B : Détection de la cible B

FIGURE 4.4 – Détection de la cible A et B

Les Figures 4.4.A et 4.4.B montrent deux exemples d'apparition de cibles dans le bâtiment. Chaque cible est identifiée par une position signalée par un carré. Ces deux positions se trouvent toutes les deux dans la zone $Salle3$, et doivent être localisées dans celle-ci dans le cas d'un déploiement idéal.

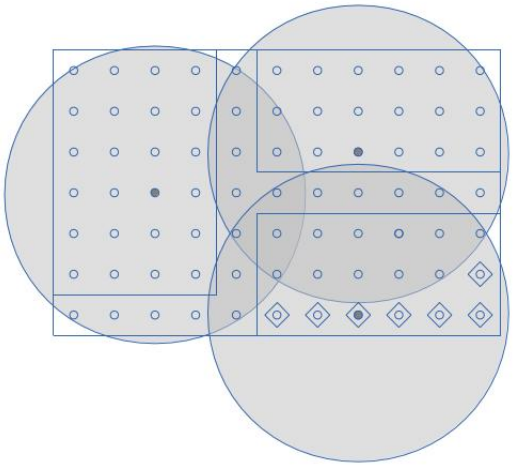

Figure 4.5.A : Conflits de positions A

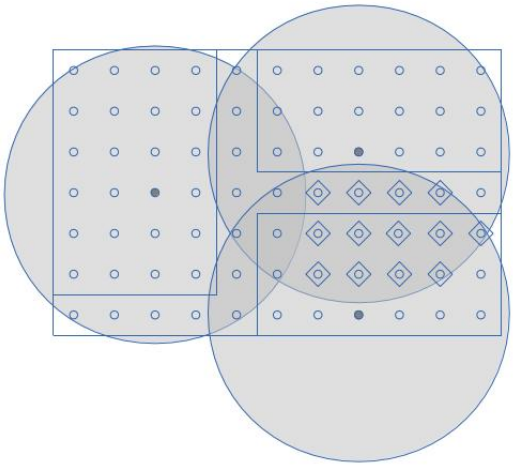

Figure 4.5.B : Conflits de positions B

FIGURE 4.5 – Conflits de positions A et B

Les Figures 4.5.A et 4.5.B montrent les ensembles de positions en conflit avec celles contenant les cibles A et B. Si dans le premier cas toutes les positions se trouvent dans la zone *Salle*3, certaines positions du deuxième exemple sont contenues dans la zone *Couloir*1.

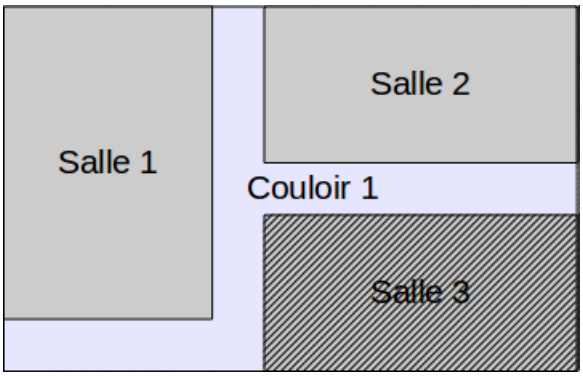

Figure 4.6.A : Localisation A

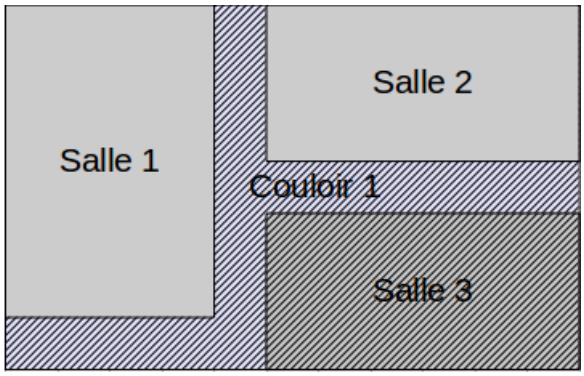

Figure 4.6.B : Localisation B

FIGURE 4.6 – Localisation A et B

Les Figures 4.6.A et 4.6.B montrent le résultat de la localisation des cibles A et B par le réseau de capteurs. Dans le premier cas, la localisation est correcte, la cible étant placée dans la zone *Salle*3. Le deuxième cas présente un conflit, et le réseau de capteur n'est pas capable de déterminer si la cible se trouve dans la zone *Salle*3 ou la zone *Couloir*1. Nous définirons ici la précision comme le booléen exprimant le succès ou l'échec de la localisation d'une cible sur une position donnée. Dans le premier exemple, la précision relative à la position est correcte, la cible étant placée dans la bonne salle, alors que dans le deuxième cas, la localisation n'est pas assurée.

Nous abordons ici un problème biobjectif : le premier critère est le même que dans le chapitre 3, c'est-à-dire la minimisation du nombre de capteurs à déployer. Les raisons invoquées dans le chapitre précédent restent les mêmes, cependant le motif concernant la réduction de la visibilité du réseau de capteurs gagne en importance au vu des applications cibles de ce modèle, pour la localisation *indoor* et la localisation *outdoor*. Le deuxième critère est dédié aux applications de localisation, et est défini comme étant la minimisation des conflits de zones.

4.3 Modélisation mathématique

La première partie du modèle mathématique est identique à celle du chapitre précédent. Soit P correspondant à l'ensemble des positions résultant de la discrétisation de la zone à couvrir. A chaque position i est attribuée une variable de décision binaire X_i, correspondant à la présence d'un capteur déployé sur la position i. La formulation du premier objectif est la suivante :

$$Minimiser\ z_1 = \sum_{i \in P} X_i \tag{4.1}$$

$$S.c. :$$

$$\sum_{i \in P} a_{i,j} X_i \geq 1, \forall j \in P \tag{4.2}$$

$$X_i \in \{0,1\}, \forall i \in P \tag{4.3}$$

Où les contraintes 4.2 expriment la nécessité de la couverture de chaque position par au moins un capteur. Le coefficient binaire $a_{i,j}$ exprime la couverture de la position j par un hypothétique capteur déployé en i.

4.3.1 Calcul de la précision

Soit E l'ensemble des zones constituant le bâtiment à couvrir et P l'ensemble des positions de la grille, nous définissons $e_j \in E$ la zone à laquelle la position $j \in P$ appartient. Nous définissons également Y_j une variable binaire représentant un conflit provoqué par la position $j \in P$. La variable Y_j sera réglée à 1 dans le cas où il existe une position $k \in P, e(k) \neq e(j)$ entrant en conflit avec la position j.

$$Y_j = \begin{cases} 1 & \text{si } \exists k \in P, e_k \neq e_j, \sum_{i \in P} |a_{i,j} - a_{i,k}| X_i = 0 \\ 0 & \text{sinon} \end{cases}$$

La variable Y_j est égale à 1 si et seulement s'il n'existe pas de capteur déployé permettant de distinguer j d'une autre position k, k appartenant à une zone $e_k \neq e_j$. La contrainte liée à Y_j est la suivante :

$$\sum_{i \in P} |a_{i,j} - a_{i,k}| X_i + \frac{1}{2}(Y_j + Y_k) \geq 1, \forall (j,k) \in P^2, j < k, e_j \neq e_k \tag{4.4}$$

Dans le cas où il n'existe aucun capteur distinguant j de k, les deux variables Y_j et Y_k seront fixées à 1, exprimant ainsi un conflit sur les deux positions. Cela permet de définir le deuxième objectif de minimisation de l'erreur de détection comme suit :

$$Minimiser\ z_2 = \frac{1}{|P|}\sum_{j\in P} Y_j \tag{4.5}$$

Le modèle complet est le suivant :

$$Minimiser\ z_1 = \frac{1}{|P|}\sum_{i\in P} X_i \tag{4.1}$$

$$Minimiser\ z_2 = \frac{1}{|P|}\sum_{j\in P} Y_j \tag{4.5}$$

$S.c.$:

$$\sum_{i\in P} a_{i,j}X_i \geq 1, \forall j \in P \tag{4.2}$$

$$\sum_{i\in P} |a_{i,j} - a_{i,k}|X_i + \frac{1}{2}(Y_j + Y_k) \geq 1, \forall (j,k) \in (e_j, e_k), e_j \neq e_k \tag{4.4}$$

$$X_i \in \{0,1\}, \forall i \in P \tag{4.3}$$

$$Y_j \in \{0,1\}, \forall j \in P \tag{4.7}$$

On obtient donc un modèle linéaire, dont le nombre de variables et de contraintes est polynomial par rapport à la taille de la grille.

4.4 Optimisation et adaptation des méthodes

Dans un premier temps, nous adapterons certaines méthodes du chapitre 3, et nous réutiliserons la même procédure afin d'obtenir les fronts de Pareto optimaux pour les petites instances. Nous choisissons ici d'utiliser les deux algorithmes H-NSGA-II et H-MOPSO, qui ont fourni les meilleurs résultats sur les instances du précédent problème de localisation.

Pour la construction du front de Pareto, nous commençons par calculer les deux points extrêmes du front de Pareto $P_{z1} = (z_1^{min}, z_2^{max})$ et $P_{z2} = (z_1^{max}, z_2^{min})$ représentant respectivement la préférence pour le premier et le second objectif.

La valeur de z_1^{min} se calcule en minimisant z_1 tout en ne prenant en compte que les contraintes de couverture. Ensuite, on minimise z_2 en prenant en compte toutes les contraintes tout en ajoutant la contrainte suivante :

$$\frac{1}{|P|}\sum_{i\in P} X_i \leq z_1^{min} \tag{4.9}$$

Ce qui nous permet d'obtenir la valeur de z_2^{max}. Le second point extrême s'obtient de manière similaire : on commence par optimiser z_2 sans se préoccuper de la valeur de z_1, ce qui nous permet

d'obtenir z_2^{min}. Puis on minimise z_1 en ajoutant la contrainte suivante au modèle :

$$\frac{1}{|P|}\sum_{j\in P} Y_j \tag{4.10}$$

Ce qui nous permet d'obtenir la valeur de z_1^{max}. Une fois les points extrêmes obtenus, la complétion du front de Pareto se fait d'une manière itérative en utilisant deux modèles $M1$ et $M2$, permettant d'obtenir respectivement les prochaines valeurs de z_2 et z_1.

Le modèle $M1$ consiste en la minimisation de z_2, en prenant en considération toutes les contraintes du modèle général, ainsi que la valeur maximale de z_1 passée en paramètre. La première valeur de z_2 est calculée en fonction de la valeur limite $z_1^{max} - \frac{1}{P}$. La valeur $\frac{1}{P}$ provient de l'homogénéisation de la fonction objectif. Si l'on regarde l'objectif initial (4.1), on remarque que les valeurs retournées par cette fonction sont entières, étant donné qu'il s'agit d'une somme de variables binaires. Dans ce cas, si l'on prend en compte la division par la constante $|P|$ de l'objectif z_1, écrire l'inégalité $\frac{1}{|P|}\sum_{i\in P} X_i < z_1^{max}$ revient à écrire $\frac{1}{|P|}\sum_{i\in P} X_i \leq z_1^{max} - \frac{1}{|P|}$. Nous obtenons donc z_2' la valeur du second objectif du point suivant sur le front de Pareto.

Une fois la valeur z_2' acquise, le modèle $M2$ permet l'obtention de la valeur z_1' du point de la même manière que pour le point extrême P_{z2}. On minimise donc le premier objectif, en ajoutant la contrainte suivante :

$$\frac{1}{|P|}\sum_{j\in P} Y_j \leq z_2' \tag{4.11}$$

Les points suivants s'obtiennent itérativement de la même manière, en remplaçant la valeur z_1^{max} par celle du dernier point calculé afin de procéder au calcul de la prochaine valeur du second objectif.

Les Figures 4.7 et 4.8 représentent deux déploiements de capteurs dans une grille de 9 lignes par 9 colonnes. Le rayon des capteurs est fixé à 4 unités, et 4 salles et un couloir sont représentés par des rectangles. La première figure représente le déploiement obtenu par le solveur suite à la minimisation du nombre de capteurs pour assurer la couverture. Chaque position de capteur est indiquée par un carré. Les positions en rouge provoquent un conflit de localisation, c'est-à-dire que pour chaque position j en rouge, Y_j est égale à 1.

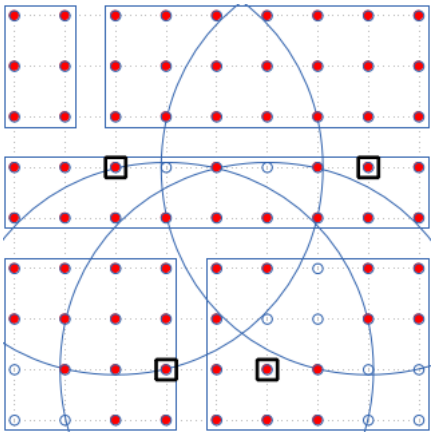

FIGURE 4.7 – Déploiement pour la couverture

Sur la Figure 4.7, quatre capteurs sont déployés, couvrant ainsi toute la zone. Le nombre de conflits de localisation provoqués par ce déploiement est égal à 69 sur 81 positions au total. La Figure 4.8 représente le résultat de la minimisation du nombre de conflits, sous contrainte de couverture, en exigeant de déployer au plus quatre capteurs.

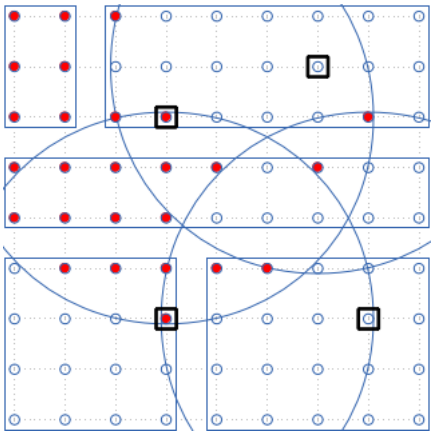

FIGURE 4.8 – Minimisation du nombre de conflits

Les positions des capteurs sont alors actualisées par le solveur, et on remarque une nette diminution du nombre de conflits de localisation : ici ce nombre est de 26 sur 81.

4.5 Expérimentations

Les conditions d'expérimentation ainsi que le paramétrage utilisés sont les mêmes que précédemment. Les paramètres des instances sont les suivants : H le nombre de lignes, W le nombre de colonnes et N le nombre de salles. En fonction de la taille de la grille et du nombre de salles requis,

10 instances sont générées aléatoirement, en faisant varier la disposition et la taille des salles. La procédure est la suivante : un couloir est généré dans la grille, sur deux lignes et l'ensemble des colonnes, séparant ici le bâtiment en une zone haute et une zone basse.

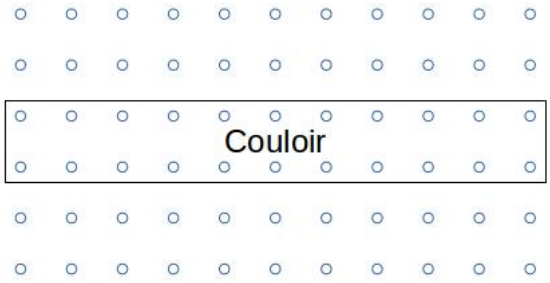

FIGURE 4.9 – Création d'une instance : phase1

La Figure 4.9 présente l'exemple d'une grille de 7 lignes par 11 colonnes. Le couloir est placé sur les lignes 3 et 4, et prend toute la largeur de la grille. Puis deux nombres N_1 et N_2 sont choisis aléatoirement, représentant respectivement le nombre de salles de la partie haute et de la partie basse de la grille.

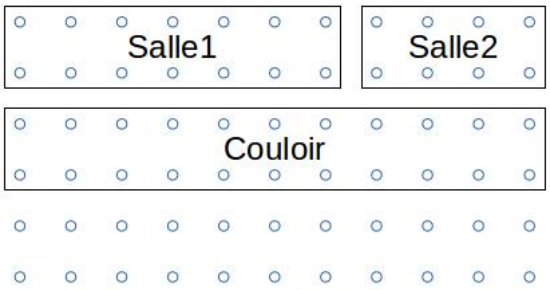

FIGURE 4.10 – Création d'une instance : phase2

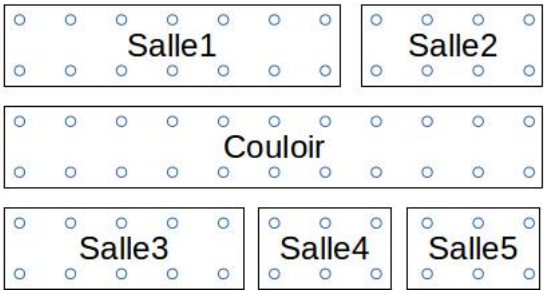

FIGURE 4.11 – Création d'une instance : phase3

Les Figures 4.10 et 4.11 montrent l'exemple d'une disposition aléatoire de 5 salles dans la zone, les tailles des salles étant variables.

TABLEAU 4.1 – Ensemble d'instances $S1$

	W	H	N		W	H	N
Instances $S1-1$	6	6	4	Instances $S1-11$	8	7	4
Instances $S1-2$	6	6	6	Instances $S1-12$	8	7	6
Instances $S1-3$	7	6	4	Instances $S1-13$	9	7	4
Instances $S1-4$	7	6	6	Instances $S1-14$	9	7	6
Instances $S1-5$	8	6	4	Instances $S1-15$	8	8	4
Instances $S1-6$	8	6	6	Instances $S1-16$	8	8	6
Instances $S1-7$	9	6	4	Instances $S1-17$	9	8	4
Instances $S1-8$	9	6	6	Instances $S1-18$	9	8	6
Instances $S1-9$	7	7	4	Instances $S1-19$	9	9	4
Instances $S1-10$	7	7	6	Instances $S1-20$	9	9	6

TABLEAU 4.2 – Ensemble d'instances $S2$

	W	H	N		W	H	N
Instances $S2-1$	10	10	4	Instances $S2-7$	12	10	4
Instances $S2-2$	10	10	6	Instances $S2-8$	12	10	6
Instances $S2-3$	10	10	8	Instances $S2-9$	12	10	8
Instances $S2-4$	11	10	4				
Instances $S2-5$	11	10	6				
Instances $S2-6$	11	10	8				

Nous commencerons les expérimentations sur des instances de taille réduite. Le premier ensemble $S1$ (voir Tableau 4.1) propose les plus petites instances, permettant de vérifier le bon fonctionnement des algorithmes sur notre problème. Le deuxième ensemble $S2$ (voir Tableau 4.2) propose des instances de plus grande taille, afin de marquer la différence de convergence des algorithmes implémentés. Toutes les instances ont été résolues optimalement via la procédure précédemment utilisée.

Les instances sont classées en fonction de leurs paramètres. Par exemple, toutes les instances de la classe $S1-1$ sont une grille de 6 lignes par 6 colonnes, avec 4 salles. Chaque classe possède 10 instances générées aléatoirement en fonction des paramètres donnés.Pour chaque instance, le rayon de couverture est fixé à 4. Le temps accordé à chaque algorithme est limité à 180 secondes, chaque instance étant traitée une seule fois. Les métriques utilisées seront les métriques *Opt* et S. La première métrique reste la même que dans le chapitre 3. Soit F l'ensemble des solutions fournies par un algorithme, et OPT le front de Pareto optimal de l'instance, $S(F, OPT)$ est définie comme le ratio de l'hypervolume $S(F)$ sur l'hypervolume $S(OPT)$.

4.5.1 Résultats sur l'ensemble $S1$

Le Tableau 4.3 présente la proportion de solutions optimales trouvées par les algorithmes H-NSGA-II et H-MOPSO sur les 200 instances $S1$. Les résultats sont regroupés par classes d'instances,

et pour chaque classe les valeurs minimum, maximum et moyenne de la métrique *Opt* sont reportées dans le tableau. Les deux algorithmes H-NSGA-II et H-MOPSO trouvent une grande partie des solutions optimales. Cependant, le score des algorithmes tend à décroître à mesure que la taille des instances augmente. On remarque une nette baisse pour les instances de tailles 8×9 et 9×9.

TABLEAU 4.3 – Métrique *Opt* sur $S1$

	$S1-1$			$S1-2$			$S1-3$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,80	0,98	1,00	0,50	0,50	0,50	0,67	0,89	1,00
Opt(H-MOPSO)	0,80	0,94	1,00	0,67	0,67	0,67	0,60	0,79	1,00
	$S1-4$			$S1-5$			$S1-6$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,33	0,73	1,00	0,50	0,70	0,83	0,43	0,68	0,86
Opt(H-MOPSO)	0,50	0,57	0,83	0,50	0,68	0,83	0,43	0,61	0,86
	$S1-7$			$S1-8$			$S1-9$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,17	0,55	0,83	0,43	0,58	0,83	0,60	0,76	1,00
Opt(H-MOPSO)	0,67	0,87	1,00	0,29	0,61	0,86	0,50	0,77	1,00
	$S1-10$			$S1-11$			$S1-12$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,50	0,83	1,00	0,50	0,70	1,00	0,14	0,70	1,00
Opt(H-MOPSO)	0,50	0,53	0,67	0,50	0,75	1,00	0,29	0,57	0,86
	$S1-13$			$S1-14$			$S1-15$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,00	0,50	0,83	0,25	0,53	0,86	0,20	0,64	1,00
Opt(H-MOPSO)	0,50	0,79	1,00	0,43	0,63	0,86	0,20	0,65	1,00
	$S1-16$			$S1-17$			$S1-18$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,57	0,71	0,83	0,00	0,19	0,33	0,00	0,24	0,67
Opt(H-MOPSO)	0,17	0,39	0,50	0,20	0,53	0,83	0,17	0,40	0,71
	$S1-19$			$S1-20$					
	min	avg	max	min	avg	max			
Opt(H-NSGAII)	0,00	0,30	0,75	0,00	0,20	0,80			
Opt(H-MOPSO)	0,25	0,49	0,75	0,00	0,24	0,60			

H-NSGA-II trouve en moyenne un plus grand nombre de solutions non-dominées sur les classes d'instances $S1-1$ à $S1-10$. Les résultats sont ensuite mitigés jusqu'à l'instance $S1-10$, où l'avantage s'oriente vers l'un ou l'autre des deux algorithmes, selon la classe d'instances étudiée. H-MOPSO quant à lui réussit mieux les quatre dernières instances, sans toutefois trouver plus de la moitié des solutions optimales. Les résultats de la métrique S disponibles dans le Tableau 4.4 montrent une forte proximité des fronts des algorithmes par rapport au front de Pareto optimal. Les résultats de la métrique peuvent être interprétés comme suit : une valeur inférieure à 1 induit des solutions optimales non trouvées, et une valeur supérieure à 1 induit des solutions dominées par le front de Pareto optimal. On remarque que H-NSGA-II s'écarte fortement de l'optimal sur les deux dernières instances, alors que H-MOPSO fournit des solutions proches du front de Pareto optimal.

TABLEAU 4.4 – Métrique S sur $S1$

	$S1-1$			$S1-2$			$S1-3$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	1,00	1,00	1,02	1,06	1,06	1,06	1,00	1,01	1,03
S(H-MOPSO,OPT)	1,00	1,01	1,02	1,03	1,03	1,03	1,00	1,03	1,06
	$S1-4$			$S1-5$			$S1-6$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	1,00	1,02	1,05	0,75	0,92	1,03	0,87	1,00	1,03
S(H-MOPSO,OPT)	1,00	1,08	1,13	1,02	1,02	1,04	1,00	1,04	1,07
	$S1-7$			$S1-8$			$S1-9$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	0,82	0,99	1,06	1,01	1,03	1,06	1,00	1,04	1,13
S(H-MOPSO,OPT)	1,00	1,01	1,02	1,01	1,03	1,05	1,00	1,02	1,06
	$S1-10$			$S1-11$			$S1-12$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	0,82	0,97	1,02	0,74	0,93	1,06	1,00	1,03	1,10
S(H-MOPSO,OPT)	1,05	1,09	1,11	1,00	1,02	1,05	1,00	1,04	1,09
	$S1-13$			$S1-14$			$S1-15$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	0,81	0,98	1,07	1,01	1,08	1,29	1,00	1,06	1,15
S(H-MOPSO,OPT)	1,00	1,01	1,03	1,00	1,07	1,27	1,00	1,03	1,06
	$S1-16$			$S1-17$			$S1-18$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	0,58	0,97	1,14	0,48	0,63	0,76	0,52	0,74	1,08
S(H-MOPSO,OPT)	1,04	1,06	1,09	1,00	1,03	1,05	1,01	1,04	1,08
	$S1-19$			$S1-20$					
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	1,10	1,28	1,65	1,02	1,25	1,50			
S(H-MOPSO,OPT)	1,01	1,06	1,13	1,02	1,09	1,23			

4.5.2 Résultats sur l'ensemble $S2$

TABLEAU 4.5 – Métrique *Opt* sur $S2$

	$S2-1$			$S2-2$			$S2-3$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,00	0,23	0,50	0,00	0,22	0,83	0,00	0,14	0,57
Opt(H-MOPSO)	0,20	0,45	0,80	0,00	0,37	0,83	0,00	0,16	0,43
	$S2-4$			$S2-5$			$S2-6$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,00	0,29	0,43	0,00	0,12	0,43	0,00	0,01	0,12
Opt(H-MOPSO)	0,00	0,34	0,60	0,14	0,24	0,50	0,00	0,13	0,29
	$S2-7$			$S2-8$			$S2-9$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H-NSGAII)	0,00	0,10	0,50	0,00	0,04	0,14	0,00	0,03	0,12
Opt(H-MOPSO)	0,17	0,29	0,67	0,00	0,19	0,29	0,00	0,11	0,25

Les résultats disponibles dans le Tableau 4.5 montrent que les deux algorithmes trouvent encore des solutions optimales pour chaque classe d'instances, cependant dans de plus faibles proportions que dans l'ensemble d'instances précédent. Si les deux algorithmes peinent à fournir des solutions optimales, H-MOPSO est néanmoins clairement plus performant sur ces instances que H-NSGA-II.

TABLEAU 4.6 – Métrique S sur $S2$

	$S2-1$			$S2-2$			$S2-3$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	1,04	1,13	1,37	1,03	1,13	1,23	1,04	1,15	1,31
S(H-MOPSO,OPT)	1,01	1,04	1,10	1,00	1,05	1,08	1,03	1,08	1,28
	$S2-4$			$S2-5$			$S2-6$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	1,02	1,08	1,22	1,03	1,14	1,34	1,05	1,10	1,22
S(H-MOPSO,OPT)	1,01	1,05	1,08	1,02	1,04	1,06	1,06	1,08	1,13
	$S2-7$			$S2-8$			$S2-9$		
	min	avg	max	min	avg	max	min	avg	max
S(H-NSGAII,OPT)	0,62	0,96	1,18	0,85	1,01	1,16	0,73	1,06	1,21
S(H-MOPSO,OPT)	1,02	1,08	1,13	1,04	1,08	1,15	1,05	1,08	1,14

Malgré le fait que les fronts fournis par les algorithmes soient en grande partie dominés par le front de Pareto optimal, les résultats de la métrique S (voir Tableau 4.6) laissent à penser que la distance entre les fronts est relativement faible, les scores des métriques étant généralement à peine supérieurs à 1.

Les résultats des expérimentations montrent que les algorithmes ne sont qu'en partie efficaces pour ce problème. Ceci peut être expliqué par le fait que l'opérateur d'optimisation ne prend en compte que le nombre d'occurrences des variables dans les contraintes du modèle, sans faire le lien avec les variables Y_j. Si cette approche s'est révélée efficace dans le chapitre 3, puisque la résolution de chaque contrainte avait un impact positif sur une variable $Y_{j,k}$, l'impact des variables sur le deuxième objectif est ici mal estimé par cet opérateur.

4.6 Heuristique

Si l'on observe de plus près le modèle mathématique, on peut remarquer que chaque affirmation $Y_j = 0$ implique la satisfaction des contraintes suivantes :

$$\sum_{i \in P} |a_{i,j} - a_{i,k}| X_i \geq 1, \forall k \in P, e_j \neq e_k \tag{4.12}$$

Soit $Y = \{Y_1, .., Y_{|P|}\}$ une affectation de l'ensemble des variables Y_j correspondant à la résolution des conflits sur les positions $j \in P$,

$$Minimiser\ z_1(Y) = \frac{1}{|P|}\sum_{i\in P} X_i \tag{4.13}$$

$S.c.$:

$$\sum_{i\in P} a_{i,j}X_i \geq 1, \forall j \in P \tag{4.2}$$

$$\sum_{i\in P} |a_{i,j} - a_{i,k}|X_i \geq 1, \forall k \in P, e_j \neq e_k, \forall j \in P, Y_j = 0 \tag{4.14}$$

$$X_i \in \{0,1\}, \forall i \in P \tag{4.3}$$

Ce qui montre que pour une affectation des variables Y_j donnée, le sous-problème correspondant est un SCP classique. Nous allons donc chercher à effectuer une optimisation en nous inspirant de l'approche suivante : premièrement générer une affectation de variables Y_j, puis procéder à l'optimisation du *Set Covering Problem* résultant. L'avantage de la deuxième partie est le grand nombre de travaux propres à l'optimisation de ce problème.

Nous proposons ici une heuristique itérative semblable aux algorithmes GRASP et ILS. Les itérations de l'heuristique sont décomposées en trois phases : (i) le regroupement des variables Y_j en *clusters*, (ii) la construction du front de Pareto en fonction des *clusters* et (iii) un opérateur de déconstruction. L'algorithme général de l'heuristique proposée est décrit dans l'Algorithme 4.1.

Algorithme 4.1 Heuristique *H3P*

Paramètres $critereArret, Max_C$

Création de $SCP_j, \forall Y_j$
$Archive \leftarrow \emptyset$
Tant que La condition *critereArret* n'est pas atteinte **Faire**
$k \leftarrow Rand(2, Max_C)$
$C \leftarrow PhaseRegroupement(k)$
$X \leftarrow PhaseConstruction(C)$
$PhaseDeconstruction(X)$
Fin Tant que
Retourne *Archive*

Où C représente le *clustering* réalisé par la première phase, et X la solution obtenue à la fin de la deuxième phase. Le paramètre k représente le nombre de *clusters* souhaités, choisi aléatoirement entre 2 et Max_C, ce dernier étant un paramètre de l'algorithme.

4.6.1 Phase de regroupement

La construction des *clusters* se fait par l'algorithme des nuées dynamiques (voir l'Algorithme 4.2). Chaque *cluster* est initialisé avec une variable Y_j choisie au hasard. Puis l'algorithme affecte à chaque variable le *cluster* le plus proche, en fonction d'une mesure de ressemblance.

Algorithme 4.2 Nuées dynamiques

Paramètres $Y, k, |P|$

$C = \emptyset$
Pour $i = 1$ to $|P|$ **Faire**
$affectation_i \leftarrow -1$
Fin Pour
Pour $i = 1$ to k **Faire**
$choix \leftarrow Rand(1, |P|)$
$C \leftarrow C + \{Y_{choix}\}$
$affectation_{choix} \leftarrow i$
Fin Pour
$stop = false$
Tant que $stop = false$ **Faire**
$stop = true$
Pour $i = 1$ to $|P|$ **Faire**
$choix \leftarrow argmin(distance(C_j, Y_i))$
Si $affectation_i \neq choix$ **Alors**
$C_{affectation_i} \leftarrow C_{affectation_i} - \{Y_i\}$
$C_{choix} \leftarrow C_{choix} + \{Y_i\}$
$affectation_i \leftarrow choix$
$stop = false$
Fin Si
Fin Pour
Fin Tant que
Retourne C

L'algorithme des nuées dynamiques effectue une nouvelle itération tant qu'un changement s'opère quant à l'attribution d'un *cluster* à une variable. Il est utile de préciser qu'une contrainte de taille maximum a été ajoutée à l'algorithme des nuées dynamiques, afin d'éviter de fournir des *clusters* de tailles trop inégales.

La première phase consiste au regroupement des variables Y_j selon des règles d'affinité. Le but est de diviser l'ensemble des variables Y_j en des sous-ensembles distincts, c'est-à-dire en *clusters*. Pour cela nous utiliserons l'algorithme de *clustering* des nuées dynamiques. Afin d'effectuer le calcul des distances nécessaire au fonctionnement de l'algorithme, il est nécessaire d'attribuer à chaque variable Y_j un vecteur représentatif V_j :

$$V_j = \{V_j^1, ..., V_j^{|P|}\} \tag{4.15}$$

$$V_j^i = \sum_{k \in P, e_k \neq e_j} |a_{i,j} - a_{i,k}|, \forall i \in P \tag{4.16}$$

Nous avons choisi de représenter le problème SCP_j par le vecteur d'occurrences des variables dans ce problème, c'est-à-dire par le nombre de contraintes résolues par chaque variable X_i. Soit $C = \{C_1, .., C_k\}$ les k *clusters* construits par l'algorithme, le calcul de la distance $distance(C_h, Y_j)$ entre une variable Y_j et un *cluster* C_h se fait de la manière suivante :

$$distance(C_h, Y_j) = \sum_{i \in P} (V_j^i - \frac{1}{|C_h|} \sum_{l \in C_h} V_l^i)^2 \tag{4.17}$$

4.6.2 Phase de construction

Algorithme 4.3 Phase de construction

Paramètres $C, Archive$

Pour tout $C_k \in C$ **Faire**
$SCP_{C_k} \leftarrow \bigcup_{Y_j \in C_k} SCP_j$
$insere_{C_k} \leftarrow false$
Fin Pour
$SCP \leftarrow SCP_{cov}$
$X \leftarrow optimise(SCP)$
Tant que $z_2(X) > 0$ **Faire**
$C_{choix} \leftarrow argmin_{C_k \in C}(GH(SCP_{C_k}, X), insere_{C_k} = false)$
$SCP \leftarrow SCP \cup SCP_{C_{choix}}$
$insere_{C_{choix}} \leftarrow true$
$X \leftarrow optimise(SCP)$
$actualise(Archive, X)$
Fin Tant que
Retourne X

Une fois les *clusters* définis, la phase de construction du front de Pareto commence (voir l'Algorithme 4.3). La première étape est de déterminer l'ordre de traitement des *clusters*. Plusieurs approches ont été testées : (i) aléatoire, (ii) le classement par bornes inférieures et (iii) le classement par scores gloutons. La première consiste simplement à choisir l'ordre des *clusters* au hasard, sans profiter des informations liées au problème. La deuxième approche se sert du calcul des bornes inférieures des *Set Covering Problem* liés aux *clusters*. La troisième approche classe les *clusters* en fonction du score obtenu par l'algorithme glouton sur les *Set Covering Problem* liés aux *clusters*. Les tests ont montré une importante instabilité pour la première approche, que nous avons donc écartée. La deuxième approche s'est révélée moins performante que la troisième, notamment à cause du temps de calcul nécessaire pour l'obtention d'une borne inférieure acceptable, et ceci pour chaque *cluster* et chaque itération. La troisième approche donne une information relative aux *clusters* en un temps négligeable, ce qui permet de ne pas pénaliser le nombre d'itérations effectuées dans le temps donné à l'algorithme. Soit X une configuration, et SCP un *Set Covering Problem*, le score

glouton $GH(SCP, X)$ est donné par l'exécution de l'algorithme glouton, qui détermine les variables à ajouter à X afin de rendre la configuration faisable pour le problème SCP. En d'autres termes, chaque contrainte de SCP doit être satisfaite par une ou plusieurs variables de X.

La première étape est la création des problèmes représentant les *clusters*. Pour chaque *cluster* C_k et pour toute variable $Y_j \in C_k$, le problème SCP_{C_k} intègre toutes les contraintes du problème SCP_j. L'algorithme va résoudre itérativement un problème SCP, en ajoutant progressivement les contraintes attribuées aux *clusters*. Tout d'abord, le problème SCP est initialisé comme étant le problème de couverture de la zone. La configuration X est obtenue via l'heuristique MetaRaps présentée dans l'état de l'art, et qui compte parmi les méthodes ayant les meilleurs scores sur les instances *unicost* proposées par (Beslay, 1987). A chaque création d'une nouvelle solution, l'archive est actualisée afin d'enregistrer les solutions non-dominées rencontrées tout au long de l'algorithme. Cette phase se termine lorsque la solution X n'engendre plus de conflits (i.e. $z_2(X) = 0$).

4.6.3 Phase de déconstruction

Algorithme 4.4 Phase de déconstruction

Paramètres $X, SCP, Archive$

$stop \leftarrow false$
Tant que $stop = false$ **Faire**
Pour tout $X_i \in X, X_i = 1$ **Faire**
$interdit_i \leftarrow (\exists j \in SCP_{cov}, a_{i,j} = 1, \sum_{k \in P} a_{i,k} X_k = 1)$
$utilite_i \leftarrow |\{j \in SCP, a_{i,j} = 1, \sum_{k \in P} a_{i,k} X_k = 1)\}|$
Fin Pour
Si $\nexists X_i = 1, interdit_i = false$ **Alors**
$stop = true$
Sinon
$choix = argmin_{i \in P}(utilite_i, X_i = 1, interdit_i = false)$
$X_{choix} = false$
$actualise(Archive, X)$
Fin Si
Fin Tant que

Une fois la configuration $X(z_2(X) = 0)$ obtenue via la phase précédente, on procède à une phase de retour en arrière (voir l'Algorithme 4.4), consistant à retirer progressivement les variables de la solution tout en minimisant l'impact sur le deuxième objectif, et en garantissant la satisfaction des contraintes de couverture. Soit SCP le problème intégrant toutes les contraintes de conflits, pour

chaque variable $X_i = 1$, on définit :

$$interdit_i = (\exists j \in SCP_{cov}, a_{i,j} = 1, \sum_{k \in P} a_{i,k} X_k = 1) \tag{4.18}$$

$$utilite_i = |\{j \in SCP, a_{i,j} = 1, \sum_{k \in P} a_{i,k} X_k = 1\}| \tag{4.19}$$

où $interdit_i$ est un booléen quantifiant la nécessité de la variable X_i pour la satisfaction du problème de couverture, et $utilite_i$ est le nombre de contraintes dans le problème complet nécessitant X_i pour leur satisfaction. A chaque itération de la phase de déconstruction, les booléens $interdit_i$ et les scores $utilite_i$ sont mis à jour. Puis la variable ayant le score d'utilité le plus faible est fixée à zéro, et l'archive Pareto est mise à jour en prenant en compte X. Cette phase se termine lorsqu'il ne reste plus de variables superflues pour la satisfaction du problème de couverture.

4.7 Expérimentations II

Les paramètres de l'heuristique sont les suivants : le temps d'exécution et le nombre de *clusters* maximum Max_C de l'algorithme général, le nombre d'itérations et la proportion de colonnes à enlever pour la MetaRaps. Le temps d'exécution est le même que celui de H-MOPSO et de H-NSGA-II, c'est-à-dire 180 secondes. Le paramètre Max_C a été fixé à $\frac{|P|}{3}$, afin d'éviter de générer un trop grand nombre de *clusters*, et de consommer trop de temps à la création des problèmes SCP correspondants. La MetaRaps est exécutée sur 100 itérations, et la proportion de colonnes supprimées est de 70%.

4.7.1 Résultats sur l'ensemble $S1$

Les scores de la métrique *Opt* relatifs aux résultats de l'heuristique sur les instances de l'ensemble $S1$ sont visibles dans le Tableau 4.7. On note une nette amélioration des résultats par rapport à ceux obtenus par les algorithmes H-NSGA-II et H-MOPSO. La plupart des instances sont résolues de manière optimale, avec des scores *Opt* moyens de 100%. En ce qui concerne les instances restantes, l'heuristique fournit une importante partie des solutions optimales, le score minimum de la métrique étant 83%.

TABLEAU 4.7 – Métriques *Opt* et *S* sur $S1$

	$S1-1$			$S1-2$			$S1-3$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	1,00	1,00	1,00	1,00	1,00	1,00	0,80	0,98	1,00
S(H3P,OPT)	1,00	1,00	1,00	1,00	1,00	1,00	0,73	0,97	1,00
	$S1-4$			$S1-5$			$S1-6$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	1,00	1,00	1,00	0,83	0,88	1,00	0,86	0,91	1,00
S(H3P,OPT)	1,00	1,00	1,00	0,74	0,82	1,00	0,83	0,90	1,00
	$S1-7$			$S1-8$			$S1-9$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	0,83	0,85	1,00	0,83	0,91	1,00	1,00	1,00	1,00
S(H3P,OPT)	0,77	0,80	1,00	0,82	0,90	1,00	1,00	1,00	1,00
	$S1-10$			$S1-11$			$S1-12$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	0,83	0,98	1,00	0,83	0,83	0,83	0,86	0,87	1,00
S(H3P,OPT)	0,80	0,98	1,00	0,70	0,75	0,80	0,81	0,86	1,00
	$S1-13$			$S1-14$			$S1-15$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	0,83	0,85	1,00	0,71	0,84	0,88	1,00	1,00	1,00
S(H3P,OPT)	0,74	0,79	1,00	0,81	0,90	1,20	1,00	1,00	1,00
	$S1-16$			$S1-17$			$S1-18$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	1,00	1,00	1,00	1,00	1,00	1,00	0,86	0,99	1,00
S(H3P,OPT)	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
	$S1-19$			$S1-20$					
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	1,00	1,00	1,00	1,00	1,00	1,00			
S(H3P,OPT)	1,00	1,00	1,00	1,00	1,00	1,00			

Les résultats de la métrique S sont également disponibles dans le Tableau 4.7. Les résultats sont concordants avec ceux de la métrique précédente, la plupart des instances étant résolues optimalement. Les scores sur les instances restantes sont toujours inférieurs ou égaux à 1, ce qui permet de dire que l'heuristique ne fournit pas de solutions dominées par le front de Pareto optimal, mais ne trouve pas toujours l'ensemble des solutions non-dominées.

4.7.2 Résultats sur l'ensemble $S2$

L'heuristique a également été testée sur les instances de l'ensemble $S2$. Les résultats du Tableau 4.8 montrent une faible diminution de la métrique *Opt*, dont les valeurs sont comprises entre 0,5 et 1. Les scores moyens sont toutefois élevés, avec des valeurs entre 0,87 et 0,99. L'heuristique fournit donc un grand nombre de solutions optimales, et continue de dépasser les algorithmes H-MOPSO et H-NSGA-II dans des proportions qui tendent à augmenter en même temps que la taille des instances.

TABLEAU 4.8 – Métrique *Opt* sur $S2$

	$S1-1$			$S1-2$			$S1-3$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	0,67	0,93	1,00	0,83	0,95	1,00	0,71	0,91	1,00
	$S1-4$			$S1-5$			$S1-6$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	0,67	0,96	1,00	0,67	0,96	1,00	0,50	0,90	1,00
	$S1-7$			$S1-8$			$S1-9$		
	min	avg	max	min	avg	max	min	avg	max
Opt(H3P)	0,67	0,99	1,00	0,67	0,95	1,00	0,50	0,87	1,00

Les résultats de la métrique S (voir Tableau 4.9) montrent que l'heuristique a tendance à fournir des solutions dominées pour ces instances, tout en restant proches du front de Pareto optimal.

TABLEAU 4.9 – Métrique *Opt* sur $S2$

	$S2-1$			$S2-2$			$S2-3$		
	min	avg	max	min	avg	max	min	avg	max
S(H3P,OPT)	1,00	1,00	1,01	1,00	1,00	1,01	1,00	1,00	1,01
	$S2-4$			$S2-5$			$S2-6$		
	min	avg	max	min	avg	max	min	avg	max
S(H3P,OPT)	1,00	1,00	1,01	1,00	1,00	1,01	1,00	1,01	1,01
	$S2-7$			$S2-8$			$S2-9$		
	min	avg	max	min	avg	max	min	avg	max
S(H3P,OPT)	1,00	1,00	1,01	1,00	1,00	1,02	1,00	1,01	1,02

Au vu des résultats sur les ensembles d'instances $S1$ et $S2$, l'heuristique proposée dans ce chapitre se montre bien plus efficace que les algorithmes H-NSGA-II et H-MOPSO pour ce problème. Il semble que l'approche en trois phases de l'heuristique permette de pallier au problème d'estimation de l'impact des variables X sur le deuxième objectif.

4.7.3 Expérimentations sur de grandes instances

Afin de confirmer les constatations observées dans la section précédente, des tests sur des instances de grande taille doivent être réalisés. Pour cela, nous proposons un ensemble d'instances L (voir Tableau 4.10). Les deux tailles de grilles traitées sont 10×20 et 10×30, et le nombre de salles prend ses valeurs dans l'ensemble $\{10, 12, 14, 16, 18, 20\}$. La génération des instances se fait de la même manière que pour les ensembles précédents, cependant l'irrégularité des formes des salles a été autorisée : un quart des salles parmi celles générées dans l'instance auront une forme non rectangulaire, et empiéteront sur une salle voisine.

TABLEAU 4.10 – Ensemble d'instances L

	H	W	N		H	W	N
Instances $L-1$	10	20	10	Instances $L-7$	10	30	10
Instances $L-2$	10	20	12	Instances $L-8$	10	30	12
Instances $L-3$	10	20	14	Instances $L-9$	10	30	14
Instances $L-4$	10	20	16	Instances $L-10$	10	30	16
Instances $L-5$	10	20	18	Instances $L-11$	10	30	18
Instances $L-6$	10	20	20	Instances $L-12$	10	30	20

Nous utiliserons les mêmes paramétrages pour les trois algorithmes, y compris le temps d'exécution fixé à 180 secondes. La métrique C sera utilisée afin d'évaluer la dominance des algorithmes les uns par rapport aux autres.

4.7.3.1 Résultats sur l'ensemble L

TABLEAU 4.11 – $C(H-NSGAII, \_)$ sur l'ensemble d'instances L

	$L-1$			$L-2$			$L-3$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,00	0,18	0,55	0,00	0,14	0,42	0,00	0,28	0,67
C(H-NSGAII,H3P)	0,00	0,00	0,00	0,00	0,03	0,10	0,00	0,07	0,38
	$L-4$			$L-5$			$L-6$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,19	0,42	0,86	0,00	0,40	0,69	0,12	0,38	0,69
C(H-NSGAII,H3P)	0,00	0,08	0,25	0,00	0,09	0,27	0,00	0,09	0,33
	$L-7$			$L-8$			$L-9$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,00	0,18	0,50	0,00	0,07	0,27	0,00	0,17	0,47
C(H-NSGAII,H3P)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	$L-10$			$L-11$			$L-12$		
	min	avg	max	min	avg	max	min	avg	max
C(H-NSGAII,H-MOPSO)	0,00	0,23	0,68	0,00	0,10	0,45	0,00	0,18	0,58
C(H-NSGAII,H3P)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,03	0,26

Les Tableaux 4.11 , 4.12 et 4.13 recensent les résultats de la métrique C sur les fronts de Pareto fournis par les trois algorithmes pour les 120 instances de l'ensemble L. Dans le premier tableau, on peut observer la dominance de H-NSGA-II sur les deux autres algorithmes. Il semble que l'augmentation des tailles des instances ne favorise absolument pas cet algorithme par rapport à l'heuristique $H3P$, les scores de $C(H-NSGAII, H3P)$ étant à zéro pour les instances $L-1$, $L-7$, $L-8$, $L-9$, $L-10$ et $L-11$. Pour les instances de taille de grille de 10×20, il arrive que H-NSGA-II trouve des solutions dominant l'heuristique, avec des scores de dominance moyens ne dépassant pas les 0,09, ainsi qu'un pic à 0,38. En ce qui concerne les instances de taille 10×30, la dominance est nulle mis à part pour la dernière classe d'instances, où l'on trouve un score moyen de dominance de 0,03. La dominance de H-NSGA-II sur H-MOPSO n'est que partielle pour les

instances traitées, avec un score moyen ne dépassant pas 0,42, et tend à diminuer lorsque la taille de la grille augmente.

TABLEAU 4.12 – $C(H-MOPSO, \_)$ sur l'ensemble d'instances L

	$L-1$			$L-2$			$L-3$		
	min	avg	max	min	avg	max	min	avg	max
C(H-MOPSO,H-NSGAII)	0,20	0,77	1,00	0,67	0,86	1,00	0,07	0,66	1,00
C(H-MOPSO,H3P)	0,00	0,04	0,18	0,00	0,01	0,10	0,00	0,01	0,08
	$L-4$			$L-5$			$L-6$		
	min	avg	max	min	avg	max	min	avg	max
C(H-MOPSO,H-NSGAII)	0,08	0,50	0,76	0,27	0,55	1,00	0,29	0,60	0,89
C(H-MOPSO,H3P)	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
	$L-7$			$L-8$			$L-9$		
	min	avg	max	min	avg	max	min	avg	max
C(H-MOPSO,H-NSGAII)	0,31	0,79	1,00	0,58	0,91	1,00	0,55	0,81	1,00
C(H-MOPSO,H3P)	0,00	0,01	0,08	0,00	0,01	0,06	0,00	0,01	0,06
	$L-10$			$L-11$			$L-12$		
	min	avg	max	min	avg	max	min	avg	max
C(H-MOPSO,H-NSGAII)	0,06	0,72	1,00	0,59	0,90	1,00	0,48	0,81	1,00
C(H-MOPSO,H3P)	0,00	0,02	0,12	0,00	0,01	0,08	0,00	0,02	0,07

Le Tableau 4.12 montre la dominance de H-MOPSO sur les autres algorithmes. Les valeurs moyennes de $C(H-MOPSO, H3P)$ sont faibles, ne dépassant pas les 0,04, et la dominance est nulle sur les instances $L-4$ à $L-6$. Cependant la dominance de H-MOPSO sur H-NSGA-II est plus prononcée, avec des scores moyens assez élevés pour la plupart des instances, et des valeurs maximum en règle générale égales à 1.

TABLEAU 4.13 – $C(H3P, \_)$ sur l'ensemble d'instances L

	$L-1$			$L-2$			$L-3$		
	min	avg	max	min	avg	max	min	avg	max
C(H3P,H-NSGAII)	1,00	1,00	1,00	0,92	0,98	1,00	0,50	0,91	1,00
C(H3P,H-MOPSO)	0,73	0,93	1,00	0,80	0,95	1,00	0,87	0,97	1,00
	$L-4$			$L-5$			$L-6$		
	min	avg	max	min	avg	max	min	avg	max
C(H3P,H-NSGAII)	0,75	0,91	1,00	0,60	0,90	1,00	0,71	0,89	1,00
C(H3P,H-MOPSO)	1,00	1,00	1,00	0,93	0,99	1,00	0,94	0,99	1,00
	$L-7$			$L-8$			$L-9$		
	min	avg	max	min	avg	max	min	avg	max
C(H3P,H-NSGAII)	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
C(H3P,H-MOPSO)	0,92	0,99	1,00	0,94	0,99	1,00	0,94	0,99	1,00
	$L-10$			$L-11$			$L-12$		
	min	avg	max	min	avg	max	min	avg	max
C(H3P,H-NSGAII)	0,95	0,99	1,00	1,00	1,00	1,00	0,76	0,98	1,00
C(H3P,H-MOPSO)	0,88	0,97	1,00	0,92	0,98	1,00	0,81	0,97	1,00

Le dernier Tableau 4.13 présente la dominance de l'heuristique H3P sur H-NSGA-II et H-MOPSO. Les valeurs moyennes des deux métriques $C(H3P, H-NSGAII)$ et $C(H3P, H-MOPSO)$ sont très élevées, généralement proches de 1. Les quatre figures suivantes montrent les

résultats donnés par les trois algorithmes : les deux premières figures concernent deux instances de taille 10×20, avec 10 et 20 salles respectivement. Sur la première instance (voir Figure 4.12), on remarque une proximité flagrante des fronts fournis par H-MOPSO et H-NSGA-II, mais également une nette distinction du front fourni par l'heuristique. Cette différence s'accentue à mesure que l'on se rapproche du point extrême favorisant le deuxième objectif.

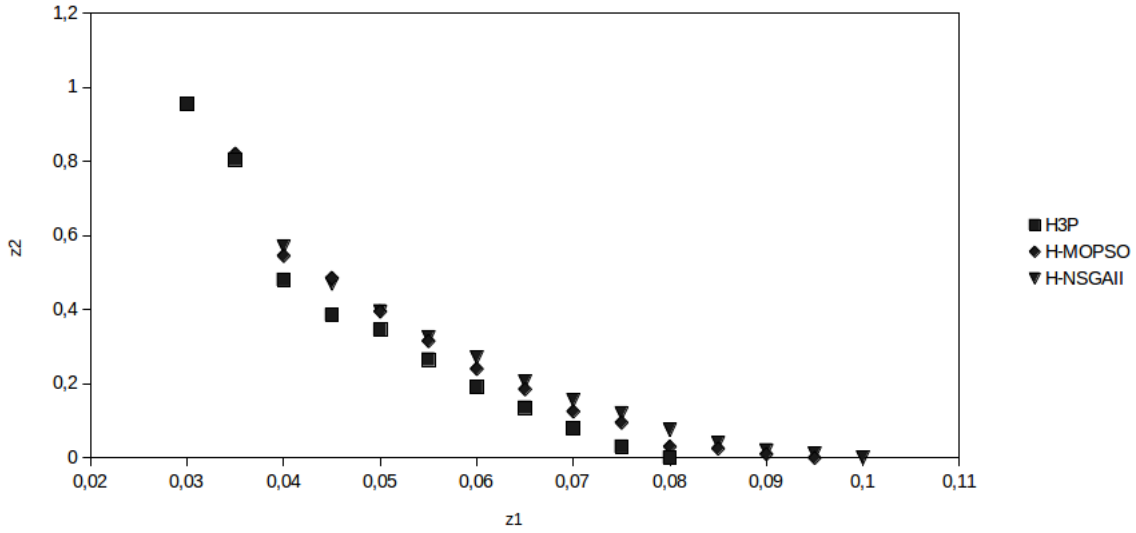

FIGURE 4.12 – Fronts de Pareto sur $L - 1$

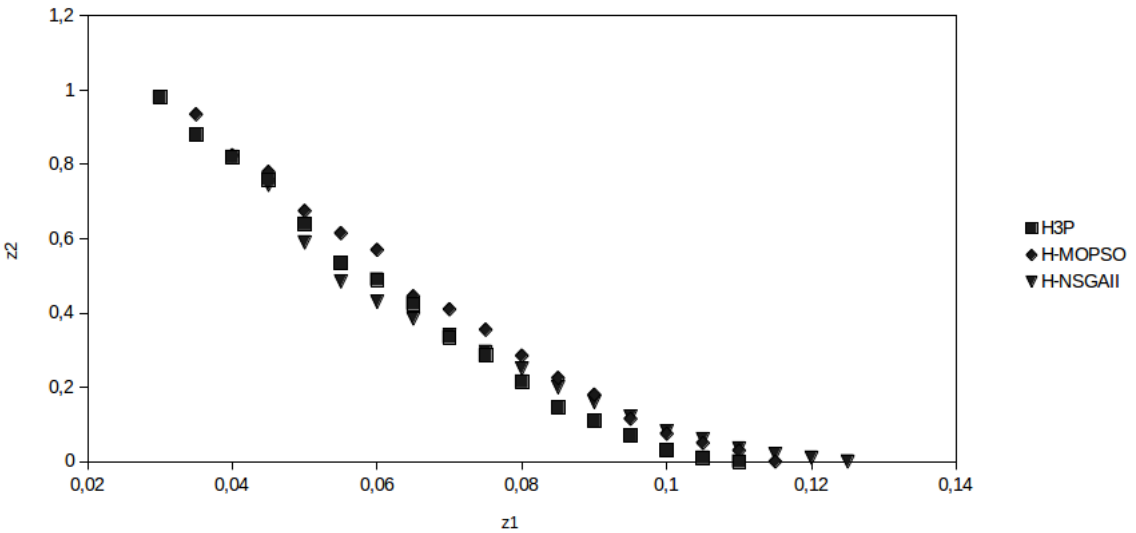

FIGURE 4.13 – Fronts de Pareto sur $L - 6$

La figure 4.13 dévoile les résultats sur une instance ayant la même taille de grille, mais avec le double de nombre de salles. Ici, l'heuristique ne se démarque pas autant que précédemment, les

fronts se croisant à plusieurs endroits. H-NSGA-II domine même partiellement l'heuristique sur la zone centrale du front de Pareto.

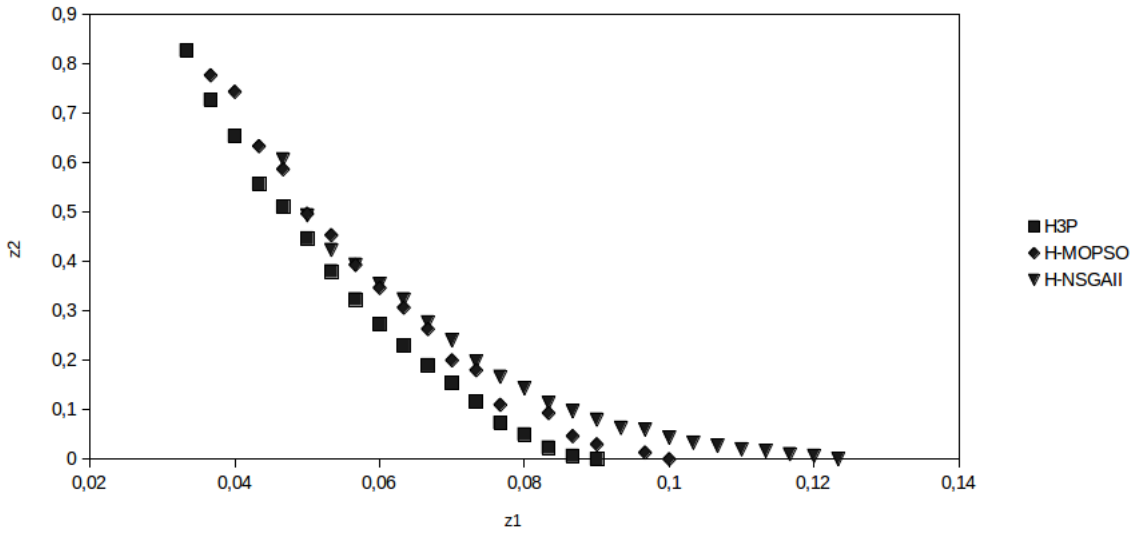

FIGURE 4.14 – Fronts de Pareto sur $L-7$

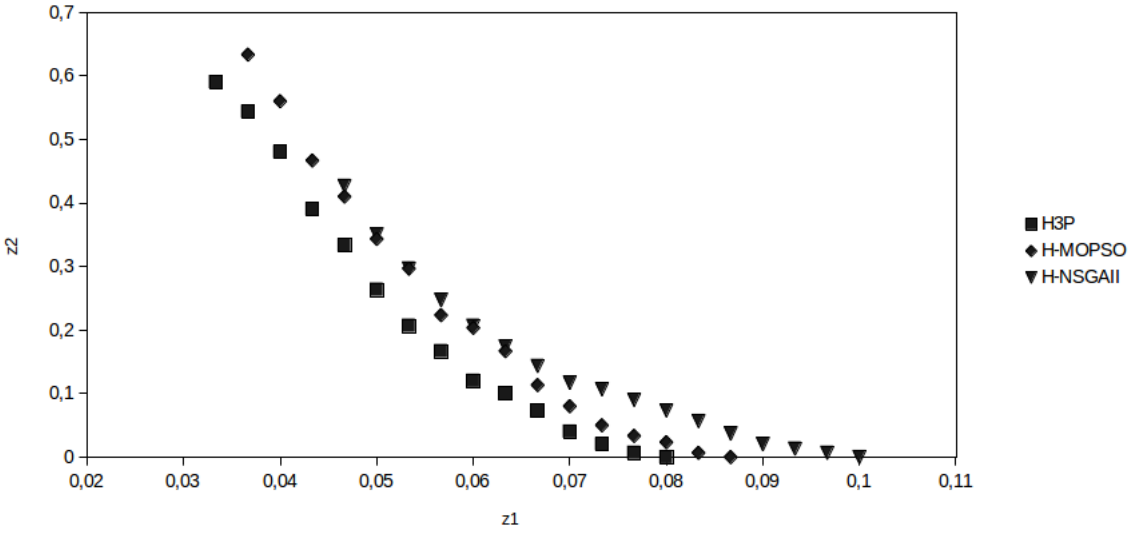

FIGURE 4.15 – Fronts de Pareto sur $L-12$

Les deux dernières figures 4.14 et 4.15 montrent les résultats sur des instances des classes $L-7$ et $L-12$. Les deux instances ont donc une taille de 10 lignes par 30 colonnes, contenant 10 et 20 salles respectivement. Les fronts ont tendance à s'écarter par rapport aux deux visuels précédents. La dominance de l'heuristique devient plus marquée, ainsi que la distance séparant les solutions données par les heuristiques et les autres fronts. Si H-MOPSO et H-NSGA-II étaient proches sur

les instances de tailles 10×20, ici H-MOPSO prend l'avantage sur H-NSGA-II, notamment pour les points favorisant le deuxième objectif. Au vu des résultats des métriques sur les différents ensembles d'instances, on peut affirmer que l'heuristique H3P domine les deux autres algorithmes pour l'optimisation de ce problème, tendance qui se confirme à mesure que les tailles d'instances augmentent. Ceci est également confirmé par les différents visuels observés, montrant une distinction croissante des fronts de Pareto.

4.8 Conclusions et perspectives

Dans ce chapitre nous avons proposé une nouvelle formulation du problème de déploiement en tenant compte de la précision d'une localisation par zonage. Un problème d'optimisation multiobjectif est ainsi posé. La formulation mathématique linéaire nous a permis la construction des fronts de Pareto optimaux pour les instances de petite et moyenne taille, et les algorithmes hybridés H-MOPSO et H-NSGA-II ont été également utilisés. Au vu des scores mitigés offerts par ces algorithmes, nous avons développé une heuristique en trois phases basée sur le *clustering* et la méthodologie du *Set Covering Problem*. Lors d'une deuxième série d'expérimentations, l'heuristique a permis de trouver une importante proportion des solutions optimales pour toutes les instances des premiers ensembles, et s'est également démarquée lors des expérimentations sur les instances de grande taille. On peut donc confirmer la supériorité de l'heuristique proposée par rapport aux algorithmes multiobjectif testés dans le cadre du problème étudié.

Les perspectives sont nombreuses, tant au niveau de la modélisation qu'au point de vue des méthodes d'optimisation. La préférence des positions d'intérêt est la suite logique du modèle, certaines positions étant plus susceptibles d'être traversées par des visiteurs que d'autres. La méthode d'optimisation du sous-problème au sein de l'heuristique pourrait alors perdre en efficacité, au vu des nombreux travaux sur le *Set Covering Problem* non *unicost*. De plus, si les performances de la méthode MetaRaps sur les instances de la littérature sont avérées, d'autres méthodes peuvent se montrer plus performantes pour traiter les sous-problèmes créés par l'heuristique.

Chapitre 5

Localisation *outdoor* et connectivité

5.1 Introduction

Dans les chapitres précédents, nous avons proposé de nouvelles problématiques de déploiement et localisation, exprimées sous la forme de problèmes d'optimisation biobjectif. Les applications visées par ce type de problème étaient essentiellement des applications de localisation intérieure, ne nécessitant ni la gestion de l'énergie, ni l'expression de la connectivité. Dans ce chapitre, nous nous intéressons au cas de la localisation *outdoor*. Si les contraintes liées à l'utilisation de signaux instables ne sont plus applicables, la connectivité devient ici essentielle. Il est en effet important de garantir la couverture de la zone, mais également le transfert et l'acquisition des données pour l'utilisateur. Dans ce chapitre, nous nous intéressons à la formulation linéaire de la connectivité afin d'assurer l'acquisition des informations recueillies par les capteurs, tout en assurant la couverture de la zone afin de localiser les cibles. Deux modèles sont proposés, ainsi qu'un *Branch and Bound* avec une définition géométrique de borne inférieure.

5.2 Description du problème et modélisation

La zone à couvrir est représentée par une grille, les positions étant représentées par l'ensemble P. Une station connectée à un réseau d'information est déployée dans la zone, à une position connue. On cherche à minimiser le nombre de capteurs à déployer, tout en garantissant la couverture de la zone, ainsi que la connexion entre chaque capteur et la station. Les notations utilisées pour la définition mathématique du problème sont données dans le Tableau 5.1 :

TABLEAU 5.1 – Notations

Notations	Significations
R_{cov}	Le rayon du dispositif d'acquisition
R_{com}	Le rayon du dispositif de communication
$Sink_i$	Valeur binaire égale à 1 si la distance entre la position i et la station est inférieure ou égale à R_{com}
$a_{i,j}$	Valeur binaire égale à 1 si la distance entre les positions i et j est inférieure ou égale à R_{cov}
$b_{i,j}$	Valeur binaire égale à 1 si la distance entre les positions i et j est inférieure ou égale à R_{com}
L	Le nombre de lignes de la grille
C	Le nombre de colonnes de la grille
(l_i, c_i)	La ligne et la colonne de la position i

La Figure 5.1 illustre les zones de communication et de couverture d'un capteur déployé dans la grille.

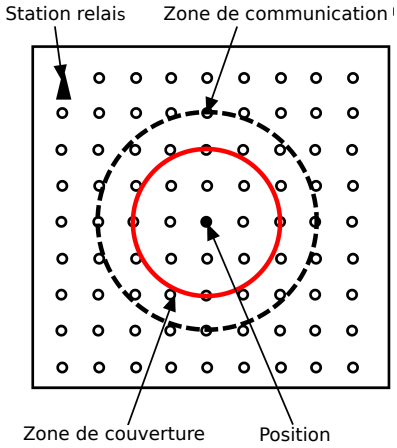

FIGURE 5.1 – Représentation d'un capteur

5.2.1 Modélisation linéaire

Soit P l'ensemble des positions de la grille, à chaque position $i \in P$ on attribue X_i la variable binaire exprimant la présence d'un capteur déployé sur la position i. Le problème de minimisation du nombre de capteurs sous contrainte de couverture totale peut s'écrire de la manière suivante :

$$Minimiser\ z = \sum_{i \in P} X_i \tag{5.1}$$

$$S.c.\ :$$

$$\sum_{i \in P} a_{i,j} X_i \geq 1, \forall j \in P \tag{5.2}$$

$$X_i \in \{0, 1\}, \forall i \in P \tag{5.3}$$

Afin de gérer la connectivité au sein du modèle, nous choisissons d'ajouter une dimension aux variables, caractérisant le degré de connectivité du capteur déployé sur la position i par rapport à la station. La variable binaire X_i^t exprime la connectivité de la position i à la station principale par t autres capteurs. Autrement dit, X_i^t est égal à 1 si et seulement si : (1) il existe un capteur en i et (2) on choisit de se connecter à un capteur en j tel que $X_j^{t-1} = 1$. En d'autres termes, il existe un chemin entre la position i et la station principale composé de p capteurs, chaque couple successif étant à une distance inférieure ou égale à R^{com}. La première contrainte à intégrer est la connexion directe entre la station et les positions à portée. La donnée $Sink_i$ représente un booléen exprimant la connexion directe de i par rapport à la station.

$$X_i^0 \leq Sink_i, \forall i \in P \tag{5.4}$$

Les capteurs déployés sur les positions i telles que $Sink_i = 1$ seront donc à portée directe de la station, et ne nécessiteront pas de passer par des capteurs intermédiaires pour transmettre les données à l'utilisateur. En ce qui concerne les positions plus éloignées, il est nécessaire de passer par 1 ou plusieurs capteurs.

$$X_i^t \leq \sum_{j \in P} b_{i,j} X_j^{t-1}, \forall i \in P, 1 \leq t \leq \lceil \frac{|P|}{R_{com}} \rceil \tag{5.5}$$

Le nombre maximum de capteurs intermédiaires est borné par $\lceil \frac{|P|}{R_{com}} \rceil$. En se basant sur les variables de décision X_i^t, le modèle linéaire global s'écrit :

$$Minimiser\ z = \sum_{i \in P} \sum_{t=0}^{\lceil \frac{|P|}{R_{com}} \rceil} X_i^t \tag{5.7}$$

S.c. :

$$\sum_{i \in P} a_{i,j} \sum_{t=0}^{\lceil \frac{|P|}{R_{com}} \rceil} X_i^t \geq 1, \forall j \in P \tag{5.8}$$

$$X_i^0 \leq Sink_i, \forall i \in P \tag{5.5}$$

$$X_i^t \leq \sum_{j \in P} b_{i,j} X_j^{t-1}, \forall i \in P, 1 \leq t \leq \lceil \frac{|P|}{R_{com}} \rceil \tag{5.6}$$

$$X_i^t \in \{0,1\}, \forall i \in P, 1 \leq t \leq \lceil \frac{|P|}{R_{com}} \rceil \tag{5.9}$$

Le modèle proposé implique la minimisation du nombre de capteurs déployés (équation 5.7), sous la contrainte que chaque position de la grille soit couverte par au moins un capteur (équation 5.8). Chaque capteur déployé doit être soit à portée directe de la station (équation 5.5), soit à portée d'un capteur déjà connecté (équation 5.6).

5.2.2 Amélioration du modèle

Le modèle précédemment proposé présente un inconvénient majeur : le nombre de variables est égal à $|P|\lceil\frac{|P|}{R_{com}}\rceil$. Afin de réduire le nombre de colonnes du problème, nous proposons de définir une nouvelle définition de la connectivité. Soit S_i la section de grille formée par la station et la position i, S_i contient toutes les positions intermédiaires pouvant intervenir dans une connexion entre la position i et la station. Un capteur déployé en i sera connecté si l'une des deux conditions suivantes est respectée :

- connexion directe : le capteur déployé en i est connexion directe si (1) la position i est à une distance inférieure ou égale à R_{com} de la station ou (2) il existe un capteur déployé en $j \in S_i$, lui-même en connexion directe, tel que $b_{i,j} = 1$.
- connexion indirecte : si le capteur n'est pas en connexion directe et qu'il existe un chemin de t capteurs, permettant au capteur déployé en i de se connecter à un capteur j en connexion directe.

Nous définissons la variable X_i correspondant à la connexion directe entre la station et le capteur, via la sous-grille S_i, et d'une manière récursive :

$$X_i = \begin{cases} 1 & \text{si un capteur est déployé en } i \text{ et que } \exists j \in S_j, b_{i,j}X_j = 1 \\ 0 & \text{sinon} \end{cases}$$

Un capteur en i sera connecté directement si et seulement s'il existe un capteur déployé sur une position de S_i, lui-même en connexion directe à la station. La contrainte liée à la variable X_i est donc la suivante :

$$\sum_{j \in S_i} b_{i,j}X_j + Sink_i - X_i \geq 0, \forall i \in P \tag{5.10}$$

En ce qui concerne la connexion indirecte, la première étape est de fixer Y_i^0 la valeur de la variable de degré 0 :

$$\sum_{j \in P} b_{i,j}X_j - Y_i^0 \geq 0, \forall i \in P \tag{5.11}$$

La variable $Y_i^0 = 1$ signifie que le capteur déployé en i n'est pas connecté directement, mais est à portée d'un capteur en $j \notin S_i$ tel que $X_j = 1$, en d'autres termes le capteur déployé en j est en connexion directe avec la station. On définit la variable Y_i^t de degré t de la manière suivante :

$$\sum_{j \in P} b_{i,j}Y_j^{t-1} - Y_i^t \geq 0, \forall i \in P, 1 \leq t \leq H \tag{5.12}$$

Les différents degrés de connexion s'expriment via la contrainte 5.12. La variable Y_i^t est réglée en fonction des variables Y_j^{t-1}, avec $b_{i,j} = 1$. Elle signifie que le capteur déployé en i n'est pas connecté via la sous-grille S_i, mais par t autres capteurs, eux-mêmes en connexions indirectes de degrés inférieurs à t. On définit $H = \lceil\frac{L+C}{R_{com}}\rceil$ le degré de connexion maximum permettant d'atteindre une grille connectée. Une nouvelle formulation du problème est donc la suivante :

$$Minimiser\ z = \sum_{i \in P}(X_i + \sum_{t=0}^{H} Y_i^t) \tag{5.13}$$

$S.c.$:

$$\sum_{i \in P} a_{i,j}(X_i + \sum_{p=0}^{H} Y_i^t) \geq 1, \forall j \in P \tag{5.14}$$

$$\sum_{j \in S_i} b_{i,j} X_j + Sink_i - X_i \geq 0, \forall i \in P \tag{5.10}$$

$$\sum_{j \in P} b_{i,j} X_j - Y_i^0 \geq 0, \forall i \in P \tag{5.11}$$

$$\sum_{j \in P} b_{i,j} Y_j^{t-1} - Y_i^t \geq 0, \forall i \in P, 1 \leq t \leq H \tag{5.12}$$

$$X_i, Y_i^t \in \{0,1\}, \forall i \in P, 0 \leq t \leq H \tag{5.15}$$

On obtient donc un modèle linéaire exprimant la connectivité et la couverture, dont le nombre de variables est $|P| * (1 + \lceil \frac{L+C}{R_{com}} \rceil)$

5.3 Définition de la borne inférieure

Bien que la méthode que nous présentons soit adaptable dans le cas où le rayon de couverture est différent du rayon de communication, nous supposerons ici que $R_{com} = R_{cov} = R$. Le but est de proposer un algorithme de *Branch and Bound* utilisant une borne inférieure alternative, celui-ci sera comparé aux résultats fournis par la programmation linéaire utilisant le modèle proposé dans les sections précédentes.

La borne inférieure que nous proposons se base sur la géométrie de la grille, et plus précisément sur la décomposition de la grille principale en sous-grilles. Chaque sous-grille est caractérisée par un nombre de colonnes $c \leq C$ et un nombre de lignes $l \leq L$ (L et C étant les nombres de lignes et colonnes de la grille principale), à condition que les valeurs de l et c soit supérieures à $2 * R$. En résolvant le problème de couverture dans chaque sous-grille, tout en exigeant la connectivité au sein des sous-grilles, et que la distance entre les capteurs de la grille soit supérieure ou égale au rayon R, nous procédons à la définition d'une borne inférieure au problème.

Définitions :

1-Soit MP le problème traité de couverture totale et de connectivité.

2-Soit B_i le sous-problème de couverture totale et de connectivité au sein d'une sous-grille i, exigeant que la distance entre capteurs (au sein de la sous-grille, ainsi que ceux déjà déployés) soit supérieure ou égale à R.

3-Soit D le nombre de sous-grilles résultant de la décomposition de la grille principale.

4-Soit MP' le problème de couverture totale de l'union des sous-grilles.

Proposition :

Si γ_i est la valeur de la fonction objectif d'une solution optimale au problème B_i, alors $\delta = \sum_{i=1}^{D} \gamma_i$ est une borne inférieure au problème MP'. Par conséquent, δ est une borne inférieure pour le problème MP.

Démonstration :

Soit une grille de taille $L \times C$, celle-ci peut être décomposée de multiples manières en un sous-ensemble de sous-grilles contiguës, chacune des sous-grilles étant une partie de la grille principale. Dans ce cas, une borne inférieure au problème MP' est également une borne inférieure au problème MP.

Concernant MP', en résolvant itérativement chaque sous-problème B_i, deux scénarios peuvent être rencontrés :

— Dans le premier scénario, la connectivité entre les sous-grilles est satisfaite. Dans ce cas, δ est une solution optimale au problème MP, les contraintes de connectivité et de couverture étant satisfaites dans la grille principale. De plus, la suppression d'un capteur aura pour effet de (i) violer la contrainte de couverture ou (ii) violer la contrainte de connectivité à l'intérieur de la sous-grille correspondante, violations qui se propageront au problème principal MP. Si la couverture totale n'est pas assurée pour la grille principale, δ reste une borne inférieure au problème principal MP.

— Dans le deuxième scénario, la connectivité entre les sous-grilles n'est pas assurée. Dans ce cas, δ est une borne inférieure pour le problème, du fait que la contrainte de connectivité est automatiquement relaxée alors que la distance entre les capteurs est supérieure ou égale à R.

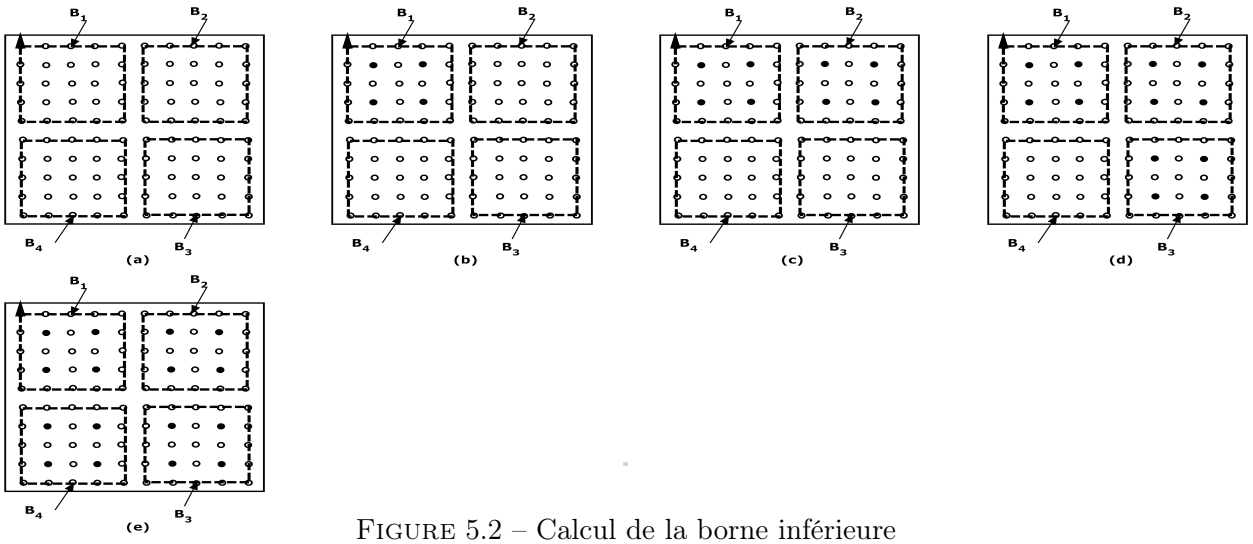

FIGURE 5.2 – Calcul de la borne inférieure

La Figure 5.2 détaille un exemple de calcul de la borne inférieure pour une grille de 10 lignes par 10 colonnes, avec un rayon R fixé à 2. La Figure (a) montre une décomposition de la grille en quatre sous-parties. Chaque sous-grille est composée de 5 lignes et 5 colonnes. Les Figures $(b), (c), (d)$ et (e) présentent le processus à suivre afin d'obtenir la valeur de la borne inférieure. Dans la Figure (b), seul le sous-problème B_1 est résolu optimalement. Dans la Figure (c), le sous-problème B_2 est résolu en considérant les capteurs déployés lors de l'optimisation du sous-problème B_1. D'une manière générale, un sous-problème B_i est résolu en prenant en compte les variables fixées dans les sous-problèmes $B_j, j < i$. La Figure (e) montre que chaque γ_i est égal à 4, ce qui implique que $\delta = \sum_{i=1}^{4} \gamma_i = 16$. Nous pouvons observer que si la contrainte de couverture est satisfaite dans toute la grille, la contrainte de connectivité n'est pas respectée. De plus, le déplacement d'un capteur engendrera soit une perte de couverture, soit la violation de la connectivité à l'intérieur d'une sous-grille. Il est également important de noter que la couverture n'est pas obligatoirement satisfaite dans certains cas. Par exemple, si l'on considère une grille de 13 colonnes et de 12 lignes, la décomposition en sous-grilles n'utilisera pas toute la grille. Malgré cela, la borne inférieure reste correcte, même si elle perd en qualité.
Note : Lors des expérimentations, chaque sous-grille sera composée de$\lceil \frac{5 \cdot R}{2} \rceil$ lignes et $\lceil \frac{5 \cdot R}{2} \rceil$ colonnes. La valeur des solutions optimales de chaque sous-problème B_i déterminé par la décomposition est égale à 4 (i.e, $\gamma_i = 4$) pour $R \geq 2$. Cependant, la connexion entre deux sous-grilles contiguës n'est jamais assurée.

5.4 Algorithme de *Branch and Bound* $(B\&B)$

L'algorithme *B&B* proposé a pour but de proposer un graphe connecté en choisissant un nombre limité d'arcs entre des capteurs connectés. La complexité de l'algorithme est $O(2^{|P|})$. Dans cette section, nous détaillons les principaux éléments de l'algorithme.

5.4.1 Initialisation de la borne supérieure

La première solution utilisée dans l'algorithme *B&B* est déterminée de la manière suivante : la première position est choisie parmi celles couvrant la position $(1, 1)$, on choisit alors la plus lointaine de la position $(1, 1)$, à condition que la station ne s'y trouve pas. Ensuite, l'ensemble F est défini comme étant l'ensemble des positions non occupées à portée de communication du réseau. La position présentant le plus grand ensemble de couverture est alors choisie parmi F. Les deux dernières étapes sont répétées tant que la couverture totale n'est pas assurée. Cette procédure permet alors d'obtenir une première borne supérieure de bonne qualité. L'Algorithme 5.1 résume la procédure d'initialisation utilisée.

Un nœud de niveau k de l'arbre de recherche est défini par les variables fixées dans la solution partielle, ainsi que les positions restantes. Celles-ci correspondent à la limite de la

Algorithme 5.1 Calcul de la borne supérieure initiale

Paramètres $a, P, X, Sink$

$F \leftarrow \emptyset$
Pour tout $i \in P, Sink_i = 1$ **Faire**
$F \leftarrow F \cup \{i\}$
Fin Pour
$\forall i \in P, score_i \leftarrow \sum_{j \in P} a_{i,j}$
$choice \leftarrow argmin_{i \in F}(distance(i, 1), a_{1,i} = 1)$
Pour tout $j \in P, a_{choice,j} = 1$ **Faire**
$covered_j \leftarrow 1$
$NbCovered \leftarrow NbCovered + 1$
$F \leftarrow F \cup \{j\}$
Pour tout $i \in P, a_{i,j} = 1$ **Faire**
$score_i \leftarrow score_i - 1$
Fin Pour
Fin Pour
Tant que $NbCovered < |P|$ **Faire**
$max \leftarrow i \in F \ as \ \nexists j \in F, j \neq i, score_i < score_j$
$X_{max} \leftarrow 1$
Pour tout $j \in P$ **Faire**
Si $a_{max,j} = 1$ et $covered_j = 0$ **Alors**
$\forall i \in P, score_i \leftarrow score_i - a_{i,j}$
$covered_j \leftarrow 1$
$F \leftarrow F \cup \{j\}$
$NbCovered \leftarrow NbCovered + 1$
Fin Si
Fin Pour
Fin Tant que
Retourne X

solution partielle, c'est-à-dire aux positions couvertes connectées et non occupées, présentant un intérêt du point de vue de la couverture. Ces positions représentent les nœuds possibles du niveau $k+1$ de l'arbre de recherche. L'ensemble des solutions limites S est donc défini comme étant l'ensemble des positions couvertes (l_i, c_i) $\forall l_i = 1 \dots L; \forall c_i = 1 \dots C$ par au moins un capteur, et chacune ayant dans son voisinage une ou plusieurs positions non couvertes appartenant à l'ensemble suivant $B = \{(Max\{l_i - 1, 1\}, Max\{c_i - 1, 1\}); (Max\{l_i - 1, 1\}, j); (Max\{l_i - 1, 1\}, Min\{c_i + 1, C\}); (l_i, Max\{c_i - 1, 1\}); (l_i, Min\{c_i + 1, C\}); (Min\{l_i + 1, L\}, Max\{c_i - 1, 1\}); (Min\{l_i + 1, L\}, c_i); (Min\{l_i + 1, L\}, Min\{c_i + 1, C\}); \}$.

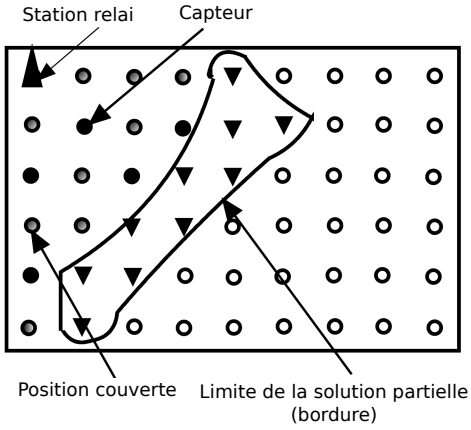

FIGURE 5.3 – Limite de la solution partielle (R=2)

La Figure 5.3 représente la limite de la solution partielle définie par le déploiement $(2,2); (2,4); (3,1); (3,3); (5,1)$. La limite contient donc l'ensemble des positions couvertes non-occupées, en contact direct (horizontalement, verticalement ou en diagonale) avec une position non couverte.

Le branchement est défini de la manière suivante : au premier niveau, la limite de la solution partielle est déterminée en calculant l'ensemble des positions $i \in P$ à portée de communication de la station. La séparation est ensuite effectuée en ajoutant un capteur sur la dernière position contenue dans S, et en ajoutant l'ensemble couvert par cette position à l'ensemble S. On répète cette procédure jusqu'à ce que la solution fournisse une couverture totale de la zone. La Figure 5.4 montre un exemple de branchement sur les trois premiers niveaux de l'arbre de recherche. La station est déployée sur la position $(1,1)$ de la grille, ce qui permet de déterminer l'ensemble S du premier niveau : $S = \{(1,2); (1,3); (2,1); (3,1); (2,2)\}$. La séparation se fait sur le dernier élément de S, un capteur est donc déployé sur la position $(2,2)$. L'ensemble S du deuxième niveau est recalculé en fonction de l'impact du dernier déploiement sur la couverture : $S = \{(3,1); (3,2); (4,2); (3,3); (2,3); (1,3); (2,4)\}$. La séparation du deuxième niveau a lieu sur le dernier élément $(2,4)$, et un capteur y est déployé. Cette procédure se répète jusqu'à l'acquisition de la couverture totale.

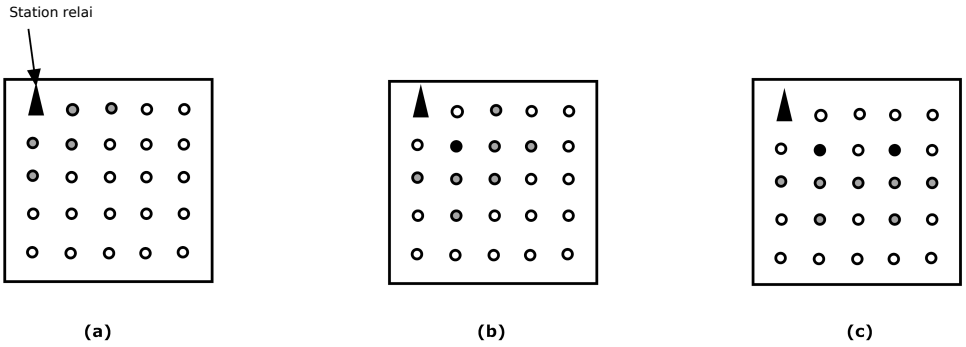

FIGURE 5.4 – Le schéma de branchement sur la grille

La Figure 5.5 représente le schéma de branchement sur l'exemple précédent en utilisant l'arbre de recherche.

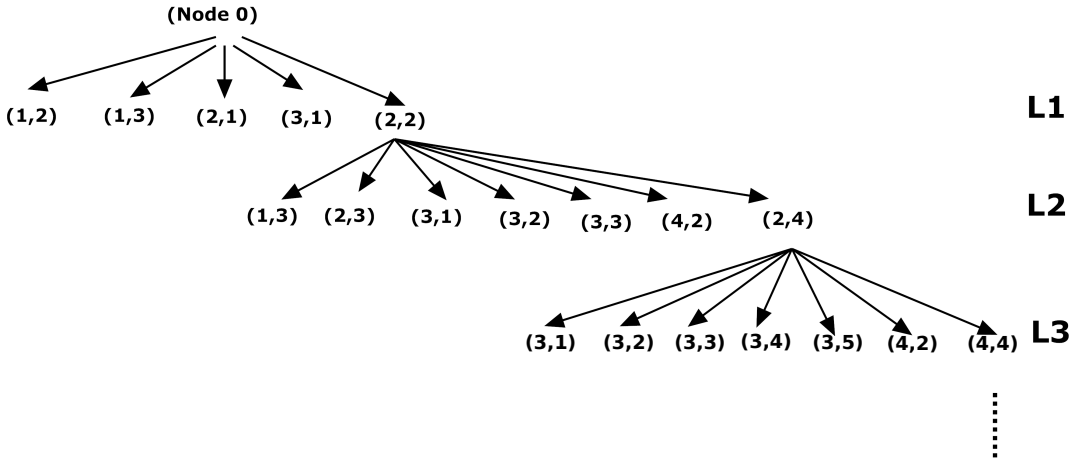

FIGURE 5.5 – Le schéma de branchement dans l'arbre de recherche

Afin d'améliorer les performances de l'algorithme ($B\&B$), deux propriétés sont développées pour permettre d'éliminer un plus grand nombre de nœuds dans l'arbre de recherche.

Définitions :

1- Soit S_k^i l'ensemble des positions appartenant à la limite de la solution partielle du nœud i au niveau k (i.e. S_k^i correspond aux positions disponibles pour le déploiement au nœud i).

2- Soit P_k^i la solution partielle du nœud i au niveau k de l'arbre de recherche.

2- Soit (l_k^i, c_k^i) la dernière position ajoutée à la solution P_k^i.

Propriété 1 :

Au niveau k de l'arbre de recherche, si les deux conditions $(l_k^i, c_k^i) \in S_k^j$ et $(l_k^j, c_k^j) \in S_k^i$ sont respectées, alors les solutions $P_k^i + (l_j, c_j)$ et $P_k^j + (l_i, c_i)$ sont équivalentes. Par conséquent, il est possible de supprimer l'un des deux éléments du nœud correspondant.

Démonstration :

Au niveau k de l'arbre de recherche, $P_k^i \backslash \{(l_k^i, c_k^i)\} = P_k^j \backslash \{(l_k^j, c_k^j)\}$. Si $(l_k^i, c_k^i) \in S_k^j$ et $(l_k^j, c_k^j) \in S_k^i$, on trouvera obligatoirement les deux solutions partielles $P_{k+1}^i = P_k^i \bigcup \{(l_k^j, c_k^j)\}$ et $P_{k+1}^j = P_k^j \bigcup \{(l_k^i, c_k^i)\}$ au niveau suivant. Il est donc possible de couper l'une des deux branches correspondantes, qui aboutiront toutes deux aux mêmes solutions. Si l'on reprend la Figure 5.5 et que l'on examine le cas du nœud $(3, 1)$ dans le deuxième niveau, la solution partielle correspondante est $(2, 2); (3, 1)$. La séparation au premier niveau sur le nœud $(3, 1)$ engendrera l'ensemble de nœuds $S = (2, 1); (2, 2); (3, 3); (4, 1)$, dont l'une des solutions partielles de deuxième niveau sera $(3, 1); (2, 2)$. Il est donc préférable de directement éliminer la position $(2, 2)$ de la liste des successeurs du nœud $(3, 1)$.

Propriété 2 :

Soit deux noeuds i et j au niveau k. Si $S_k^i = S_k^j$ et la couverture assurée par les deux solutions partielles est la même, alors on peut couper l'une des branches.

Démonstration :

Si i et j sont tous les deux au niveau k, les deux solutions partielles P_k^i et P_k^j sont de tailles égales. Etant donné que les successeurs des deux noeuds sont les mêmes et que les couvertures sont identiques, les deux solutions partielles représentent un même état, et donc les deux noeuds sont équivalents.

5.4.2 Procédure de séparation

La procédure de séparation consiste à calculer la valeur de la borne inférieure Lb_k. Celle-ci est calculée en ne tenant compte que des positions non couvertes. La borne inférieure décrite précédemment est utilisée dans l'algorithme $(B\&B)$. En premier lieu, le nombre de sous-grilles est déterminé, et les sous-grilles sont disposées dans la grille principale. Puis chaque sous-grille est complétée par quatre capteurs assurant la connectivité et la couverture à l'intérieur de la grille. Les positions de la solution partielle sont ensuite ajoutées dans la grille. La valeur de borne inférieure est égale à $Lb_k = \delta - \theta + \alpha$ où :

- La valeur δ correspond au nombre total de capteurs ajoutés par la résolution de la couverture et de la connectivité au sein des sous-grilles.
- La valeur θ correspond au nombre de capteurs déployés lors de la procédure de calcul de la borne inférieure communicant avec des capteurs appartenant à la solution partielle.

— La valeur α correspond au nombre de capteurs déployés dans la solution partielle.

Si la valeur retournée est supérieure ou égale à la valeur actuelle de la borne supérieure, le branche est coupée.

5.5 Expérimentations

Nous définissons une instance au problème de couverture et connectivité comme suit : L et C définissent respectivement le nombre de lignes et le nombre de colonnes, qui prennent leur valeur dans $\{7, 10, 13, 15\}$. Les rayons de couverture et de communication sont tous deux fixés à la même valeur R, incluse dans l'ensemble $\{2, 3, 4, 5\}$. La dernière information caractérisant une instance est la position de la station, celle-ci étant définie par l'une des cinq positions : $(1, 1); (1, N); (M, 1); (M, N)$ et $(\frac{M}{2}, \frac{N}{2})$. Les quatre premières sont les coins de la grille, et la dernière est positionnée dans le centre de la zone. Deux positions possibles sont également ajoutées, choisies aléatoirement dans la grille. L'algorithme de $(B\&B)$ a été développé en C, et les expérimentations ont été faites sur un intel core i 7-3720QM. Le temps de résolution a été fixé à 3600 secondes. Le Tableau 5.2 indique les nombres moyens de nœuds explorés pour chaque taille d'instances et chaque valeur de R. Si ce nombre est faible lorsque la taille de l'instance est faible et que le rayon a une valeur élevée, la tendance inverse est observée lors du traitement de grandes instances avec une valeur de R faible. On remarque également que pour une même taille de grille, le nombre moyen de nœuds explorés a tendance à diminuer à mesure que la valeur du rayon augmente.

TABLEAU 5.2 – Nombre moyen de nœuds explorés

(L,C)	$R = 2$	$R = 3$	$R = 4$	$R = 5$
(7,7)	6129	142	20	0
(7,10)	553777	391	68	29
(7,13)	14033250	3679	113	63
(7,15)	17219208	9466	313	80
(10,7)	323031	502	87	24
(10,10)	17638408	3885	569	234
(10,13)	7101873	91887	3130	455
(10,15)	4454523	656984	11217	1009
(13,7)	15122894	7426	286	35
(13,10)	5503218	219679	2843	392
(13,13)	2604277	8773123	27282	970
(13,15)	1414861	7105747	1164924	1013
(15,7)	11191603	14985	545	100
(15,10)	4057838	1013352	11921	879
(15,13)	1515591	7466677	215911	1626
(15,15)	920680	4503609	1758296	2624

L'impact des propriétés proposées peut être observé dans le Tableau 5.3. La première colonne représente la taille de l'instance observée, et le nombre d'éliminations est donné dans les colonnes suivantes, pour chaque valeur de rayon. Les tendances sont les mêmes que pour le nombre de nœuds

explorés : le nombre d'éliminations augmente en même temps que la taille de la grille, mais diminue à mesure que le rayon augmente. Le Tableau 5.4 présente le pourcentage d'instances résolues par taille de grille et valeur de rayon. Le nombre de variables dépendant de la taille de la grille a évidemment un impact sur la difficulté de la résolution de l'instance, ce qui était prévisible au vu de la complexité de l'algorithme. Cependant la valeur du rayon R semble avoir un impact important sur le temps de résolution de l'instance, toutes les instances traitées avec un rayon égal à 4 ou 5 ayant été résolues optimalement. Les instances avec un rayon égal à 2 semblent plus difficiles à résoudre, une majorité d'instances n'ayant pas été résolues optimalement.

TABLEAU 5.3 – Nombre moyen de nœuds éliminés

$R=2$			$R=3$		$R=4$		$R=5$	
(L,C)	Pr_1	Pr_2	Pr_1	Pr_2	Pr_1	Pr_2	Pr_1	Pr_2
(7,7)	3975	2087	23	36	3	0	0	0
(7,10)	641858	222329	146	102	8	5	2	0
(7,13)	22736250	6299444	1960	1406	20	28	5.	1
(7,15)	24516300	7942545	6474	3263	71	126	7	5
(10,7)	351072	126614	175	161	9	9	1	0
(10,10)	21931507	6464089	2474	1361	144	162	28	14
(10,13)	9608302	2946136	81159	34868	1234	981	45	103
(10,15)	8025261	2536092	882544	317526	7586	4622	299	324
(13,7)	24619290	8009267	5475	3441	89	99	19	0
(13,10)	11547112	2856468	234931	105072	1429	1166	138	148
(13,13)	6479998	1530140	12832925	4660130	19822	11564	354	340
(13,15)	4308342	916808	11073074	3650293	1715195	588276	470	307
(15,7)	17365939	5561607	12961	7687	183	234	15	15
(15,10)	8132338	2128375	1284577	503507	7628	5426	162	373
(15,13)	3942029	907265	12143771	4026823	197663	96653	782	617
(15,15)	3300098	692202	6851489	2114096	1888873	840005	1220	925

TABLEAU 5.4 – (%) d'instances résolues optimalement

(L,C)	$R=2$	$R=3$	$R=4$	$R=5$
(7,7)	100	100	100	100
(7,10)	100	100	100	100
(7,13)	57	100	100	100
(7,15)	0	100	100	100
(10,7)	0	100	100	100
(10,10)	0	100	100	100
(10,13)	0	100	100	100
(10,15)	28	100	100	100
(13,7)	0	100	100	100
(13,10)	0	100	100	100
(13,13)	0	0	100	100
(13,15)	0	0	100	100
(15,7)	0	100	100	100
(15,10)	0	100	100	100
(15,13)	0	0	100	100
(15,15)	0	0	100	100

La Figure 5.6 présente les temps d'exécution moyens de l'algorithme ($B\&B$) pour chaque taille d'instance. Chaque courbe représente l'utilisation d'une valeur de rayon différente. Les résultats concordent avec ceux présentés dans le Tableau 5.4. La valeur maximum du temps d'exécution est de 3600 secondes, auquel cas l'algorithme ne peut garantir l'optimalité de la solution proposée. On peut remarquer que les instances avec R égal à 5 sont toutes résolues en un temps négligeable. La même observation peut être faite quant aux instances traitées avec un rayon de 3 ou 4 unités, mises à part celles présentant le plus grand nombre de variables. En ce qui concerne les instances traitées avec $R = 2$, seules les plus petites instances ont été résolues de manière optimale avant la limite de temps. Ces résultats permettent de confirmer l'impact important de la valeur de R sur la difficulté de l'instance. La cause peut en être l'impact sur la borne inférieure, ou sur le calcul des successeurs des nœuds. En effet, la définition des sous-grilles utilisées dans la borne inférieure est dépendante du rayon : pour une même taille de grille, plus le rayon est grand, plus le nombre de sous-grilles utilisées est faible. L'impact au niveau du calcul des successeurs provient du fait que l'ajout d'un nouveau capteur aura un impact plus ou moins important dans l'actualisation de la liste des successeurs S, en fonction de la couverture.

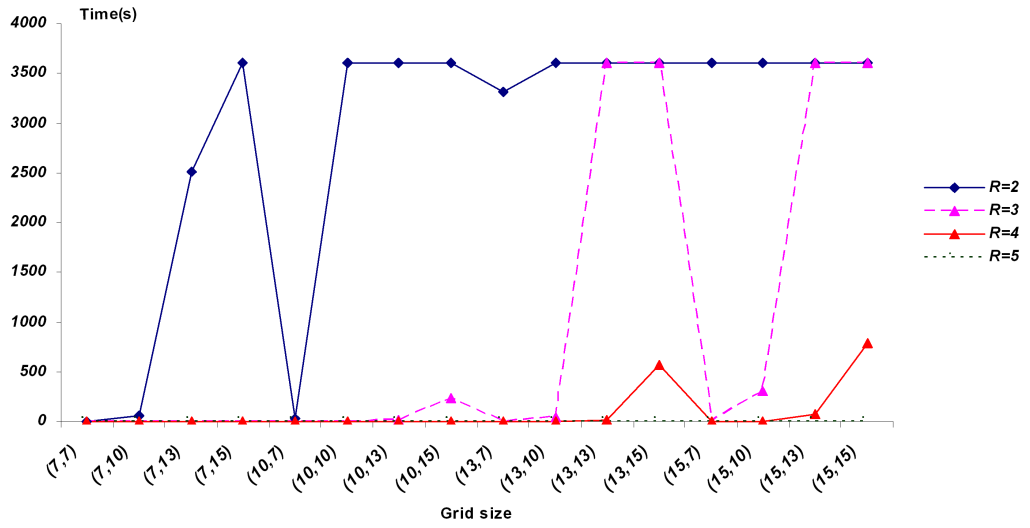

FIGURE 5.6 – Temps de résolution moyen du ($B\&B$)

La Figure 5.7 présente les temps d'exécution moyens de la résolution des instances par le solveur linéaire Ilog Cplex. Chaque courbe représente l'utilisation d'une valeur de rayon différente. Le modèle linéaire proposé dans les sections précédentes est utilisé. Les courbes représentant les performances de l'algorithme ($B\&B$) et Ilog Cplex ont les mêmes allures, notamment en ce qui concerne la variation du rayon R. Le solveur linéaire Ilog Cplex utilisant la relaxation linéaire pour le calcul des bornes inférieures, il semble donc que la borne inférieure que nous avons proposée ne soit pas mise en cause dans la perte de performance relative à la diminution de la valeur de R. On peut

également remarquer que les temps d'exécution moyens sont plus faibles pour l'algorithme ($B\&B$), celui-ci résolvant optimalement un plus grand nombre d'instances, et de manière plus rapide.

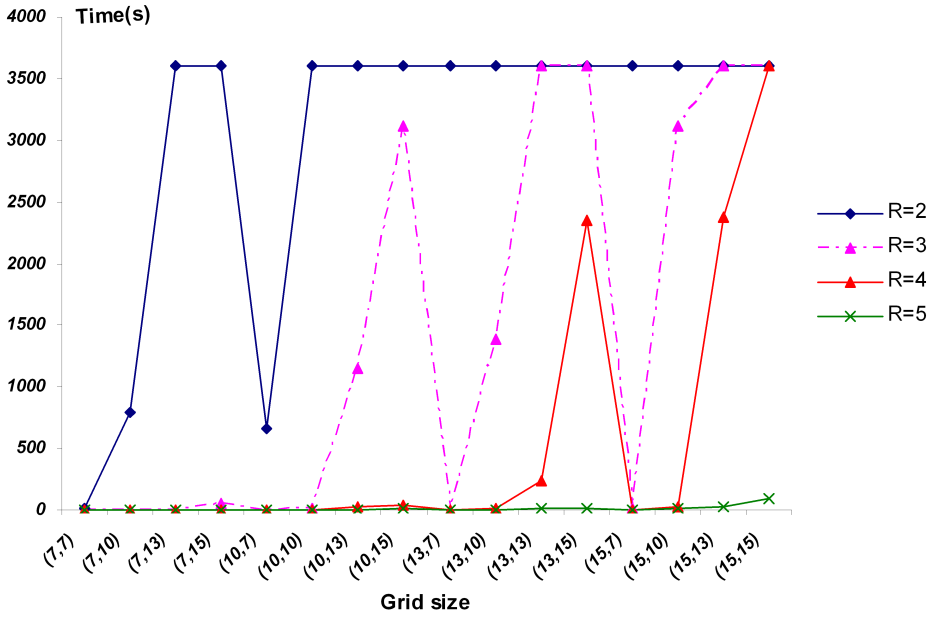

FIGURE 5.7 – Temps de résolution moyen de Ilog Cplex

Au vu des résultats présentés, nous pouvons conclure par le fait que l'algorithme ($B\&B$) offre de meilleures performances que la programmation linéaire pour ce problème. En effet, 74% des instances ont été résolues optimalement par l'algorithme, alors que Ilog Cplex n'en a résolu que 69%. Nous pouvons donc en conclure que l'algorithme ($B\&B$) est plus approprié quant à l'optimisation de ce problème.

5.6 Conclusion et perspectives

Dans ce chapitre, nous avons abordé une problématique de déploiement de capteurs dans un milieu extérieur, en nous concentrant sur les contraintes de couverture et de connectivité. Un premier modèle a été présenté, puis a été amélioré afin de réduire le grand nombre de variables. Une borne inférieure basée sur la décomposition de la grille a également été développée afin d'être utilisée dans un algorithme de *Branch and Bound* ($B\&B$). A cet algorithme nous avons ajouté deux propriétés afin d'accélérer le temps d'exécution de l'algorithme afin d'éviter les symétries dans l'arbre de recherche. L'algorithme a été testé sur plusieurs instances, faisant varier la taille de la grille, le rayon de communication et l'emplacement de la station relais. Ses performances ont été comparées à celles du solveur Ilog Cplex, et les résultats ont montré la dominance de l'algorithme ($B\&B$) pour l'optimisation de ce problème.

Les perspectives faisant suite à ce travail sont les suivantes : premièrement une amélioration de la borne inférieure. En effet, la taille des sous-grilles est fixée, mais plusieurs combinaisons pourraient être utilisées afin d'accroître sa qualité. La deuxième perspective est plus informatique : une parallélisation de l'algorithme permettrait de diminuer sensiblement les temps d'exécution moyens. Enfin, nous chercherons à étendre cette borne inférieure aux différents problèmes de localisation abordés dans les chapitres précédents.

Chapitre 6

Conclusions et perspectives

Au cours de cette thèse, nous avons étudié plusieurs problématiques liées au déploiement de réseaux de capteurs destinés aux applications de localisation de cibles. Dans le troisième chapitre, nous avons proposé un modèle linéaire pour l'optimisation d'un problème biobjectif de déploiement et de localisation. Le premier objectif pris en compte consiste en la minimisation du nombre de capteurs déployés, permettant ainsi la minimisation du coût de déploiement, autant du point de vue de l'achat des unités que de la main d'œuvre nécessaire à l'installation du réseau. Le deuxième objectif est formulé comme étant la minimisation de l'erreur d'estimation de la position des cibles en cas de détection dans la zone, cet objectif étant orienté principalement pour les applications de *tracking*. Afin de procéder à l'optimisation de ce problème, nous avons tout d'abord mis en place une procédure faisant appel à un solveur linéaire, dans le but d'obtenir les fronts de Pareto optimaux pour les instances de taille réduite. Puis, nous avons procédé à l'adaptation de quatre méthodes d'optimisation multiobjectif de la littérature : les algorithmes NSGA-II, SPEA2, MOEA/D et MOPSO. Après une série d'expérimentations préliminaires, nous avons procédé à l'hybridation de trois de ces méthodes via une heuristique constructive basée sur la pondération, ce qui a donné les trois algorithmes modifiés suivants : H-NSGA-II, H-SPEA2 et H-MOPSO. Une nouvelle série d'expérimentations nous a permis de déterminer que les algorithmes H-NSAGA-II et H-MOPSO étaient les plus performants quant à l'optimisation de notre problème. Le quatrième chapitre présenta une évolution du chapitre précédent. Le modèle linéaire a été modifié afin de prendre en compte une problématique de localisation à l'intérieur des bâtiments en se basant sur une notion de "zonage". Si dans le troisième chapitre nous avons basé la localisation sur les positions résultant de la discrétisation de la grille, ce chapitre prend en compte la division du bâtiment en sous-zones : les couloirs, les salles, ... La procédure d'obtention du front de Pareto optimal a été réutilisée, et les deux algorithmes H-NSGA-II et H-MOPSO ont été soumis à une première série d'expérimentations. Les résultats mitigés nous ont permis de conclure sur le fait que ces algorithmes n'étaient que peu adaptés à l'optimisation de notre problème. Nous avons donc entrepris le développement d'une nouvelle heuristique itérative, basée sur le *clustering* et l'optimisation du *Set Covering Problem*.

Cette heuristique, décomposée en une phase de regroupement des variables, une phase d'optimisation du sous-problème et de construction du front de Pareto, et une phase de retour en arrière, a été soumise aux mêmes ensembles d'expérimentations que les algorithmes précédents. L'heuristique a montré qu'elle était en mesure de trouver la majorité des solutions optimales pour la plupart des instances traitées. Les expérimentations sur les grandes instances ont également montré que l'heuristique dominait les deux métaheuristiques utilisées sur l'optimisation de ce problème. Le dernier chapitre se focalise sur les applications de déploiement extérieur, en intégrant la connectivité à un modèle de déploiement et couverture classique. Deux modèles linéaires ont été proposés, ainsi qu'une nouvelle définition de borne inférieure basée sur les propriétés géométriques du problème. Cette borne a été intégrée dans un algorithme de *Branch and Bound*, et celui-ci a été soumis à une série d'expérimentations faisant varier les tailles de grilles et les rayons relatifs à la communication et la couverture des capteurs. L'algorithme a montré plus d'aisance que le solveur linéaire dans la résolution des instances proposées, malgré un comportement similaire quant à la variation des instances, notamment la diminution de la valeur du rayon utilisé.

Quant aux perspectives de ce travail de thèse, on en considère deux classes : les perspectives sur les méthodologies et les perspectives sur les problèmes et applications liées.

Les perspectives concernant les méthodologies font référence à la proximité des problèmes étudiées et le *Set Covering Problem*, que nous avons par ailleurs exploitée lors du chapitre 4. L'optimisation des sous-problèmes est une phase importante de l'heuristique que nous avons mise en place, et nous avons choisi d'utiliser la méthode MetaRaps au vu des performances présentées dans la littérature. Cependant, le caractère géométrique de notre problème pourrait influer sur la structure des sous-problèmes engendrés par l'heuristique. Il est donc nécessaire de (i) expérimenter le plus grand nombre possible de méthodes du *Set Covering Problem* afin de valider ou d'invalider la supériorité de la méthode MetaRaps pour l'optimisation des sous-problèmes résultants et (ii) étudier les propriétés mathématiques et géométriques du problème pour la création de méthodes spécialisées.

Les perspectives liées aux problèmes et aux applications tendent à rapprocher le plus possible les modèles du cas réel. Nous avons cité les exemples de l'intégration des obstacles et la mise en valeur de certaines positions par rapport aux autres qui constituent des évolutions intéressantes aux problèmes étudiés, mais nécessitent une réadaptation complète des méthodes utilisées. Enfin, durant des expérimentations sur les signaux WiFi, nous avons commencé à définir un nouveau modèle de couverture sous la forme de sous-ensembles flous, afin de faire évoluer le modèle binaire classique. Le but de ce modèle, naturellement compatible avec les modèles linéaires des chapitres 3 et 4, est de faciliter la résolution des contraintes de localisation, sans affecter la satisfaction des contraintes de couverture. Si l'on reprend les deux types de contraintes du modèle étudié dans le chapitre 3 :

$$\sum_{i \in P} a_{i,j} \cdot X_i \geq 1, \forall j \in P \tag{6.1}$$

$$\sum_{i \in P} (|a_{i,j} - a_{i,k}| \cdot X_i) + Y_{j,k} \geq 1, \forall (j,k) \in P^2, j \neq k \tag{6.2}$$

les contraintes (6,1) représentent la couverture et les contraintes (6,2) représentent la localisation. Dans un modèle de couverture binaire, $a_{i,j}$ voit sa valeur fixée soit à 0, soit à 1, en fonction de la distance séparant les deux positions i et j. Si le signal WiFi est trop instable pour déterminer exactement la distance entre la source du signal et le récepteur, le modèle que nous mettons en place permet néanmoins d'en faire une estimation floue, par exemple : la cible est proche, lointaine, ou non détectée. Dans ce cas, la donnée $a_{i,j}$ prendra ses valeurs dans l'ensemble $\{0, 1, 2\}$. On pourra alors réécrire les contraintes du modèle sous la forme suivante :

$$\sum_{i \in P} (a_{i,j} \neq 0) \cdot X_i \geq 1, \forall j \in P \tag{6.3}$$

$$\sum_{i \in P} ((a_{i,j} \neq a_{i,k}) \cdot X_i) + Y_{j,k} \geq 1, \forall (j,k) \in P^2, j \neq k \tag{6.4}$$

L'impact sur les contraintes de couverture sera négligeable : peu importe de quelle manière on détecte la cible, l'essentiel est de la couvrir. Le plus intéressant est l'impact sur les contraintes de localisation : si précédemment il fallait que les deux données binaires $a_{i,j}$ et $a_{i,k}$ soient différentes afin de différencier les deux positions à l'aide d'un capteur déployé en i, il suffit maintenant que le capteur détecte les deux positions de manière différente, en d'autres termes le capteur peut être à portée des deux positions, mais il est capable de les différencier si les coefficients $a_{i,j}$ et $a_{i,k}$ sont différents. Ce modèle sera au cœur des prochains travaux que nous réaliserons, afin d'offrir l'optimisation de déploiement pour la localisation tout en tenant compte des caractéristiques physiques du signal WiFi.

Cette thèse a donné lieu à plusieurs publications dont deux articles de revue internationale (Rebai et al., 2014) (**M. Le Berre** et al.), deux communications de conférence internationale (**M. Le Berre** et al., 2011) (**M. Le Berre** et al., 2013) et une communication de conférence nationale (**M. Le Berre** et al., 2012). Deux autres articles de revue internationale ont également été soumis, ainsi qu'un brevet d'invention.

Références bibliographiques

S. M. Nazrul Alam and Z. J. Haas. Coverage and connectivity in three-dimensional networks. *In Proceedings of the 12th annual International Conference on Mobile computing and networking*, pages 346–357, 2006.

M. Almiñana and J. T. Pastor. Two new heuristics for the location set covering problem. *Top*, 2 : 315–328, 1994.

M. Almiñana and J. T. Pastor. An adaptation of sh heuristic to the location set covering problem. *European Journal of Operational Research*, 100 :586–593, 1997.

I. K. Altinel, N. Aras, E. Güney, and C. Ersoy. Binary integer programming formulation and heuristics for differentiated coverage in heterogeneous sensor networks. *Computer Networks*, 52 : 2419–2431, 2008.

H. M. Ammari and S. K. Das. Coverage, connectivity, and fault tolerance measures of wireless sensor networks. *In Proceedings of the 8th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 35–49, 2006.

H. M. Ammari and S. K. Das. Centralized and clustered k-coverage protocols for wireless sensor networks. *IEEE Transactions On Computers*, 61 :118–133, 2012.

T. Andersen and S. Tirthapura. Wireless sensor deployment for 3d coverage with constraints. *In the 6th International Conference on Networked Sensing Systems*, pages 78–81, 2009.

X. Bai, S. Kumar, D. Xuan, Z. Yun, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. *In Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 131–142, 2006.

E. Balas and M.C. Carrera. A dynamic subgradient-based branch-and-bound procedure for set covering. *Operations Research*, 44 :875–890, 1996.

J. Bautista and J. Pereira. A grasp algorithm to solve the unicost set covering problem. *Computers and Operations Research*, 34 :3162–3173, 2007.

J.E. Beslay. An algorithm for set covering problem. *European Journal of Operational Research*, 31 :85–93, 1987.

J.E. Beslay. A lagrangian heuristic for the set covering problem. *Naval Research Logistics*, 37 : 151–164, 1990.

J.E. Beslay and P.C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94 :392–404, 1996.

M.J. Brusco, L.W. Jacobs, and G.M. Thompson. A morphing procedure to supplement a simulated annealing heuristic for cost- and coverage-correlated set-covering problems. *Annals of Operations Research*, 86 :611 –627, 1999.

J. Burrell, T. Brooke, and R. Beckwith. Vineyard computing : Sensor networks in agricultural production. *IEEE Pervasive Computing*, 3 :38–45, 2004.

A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research*, 47 :730–743, 1999.

A. Caprara, P. Toth, and M. Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98 :353–371, 2000.

M. Caserta. Tabu search-based metaheuristic algorithm for large-scale set covering problems. *K.F. Doerner et al. (Eds.), Metaheuristics : Progress in complex systems optimization. New York : Springer*, pages 43–63, 2007.

S. Ceria, P. Nobili, and A. Sassano. A lagrangian-based heuristic for large-scale set covering problems. *Mathematical programming*, 81 :215–228, 1998.

R. Cerulli, R. D. Donato, and A. Raiconi. Exact and heuristic methods to maximize network lifetime in wireless sensor networks with adjustable sensing ranges. *European Journal of Operational Research*, 220 :58–66, 2012.

K. Chakrabarty, S.S. Iyengar, H. Qi, and E. Cho. Coding theory for surveillance and target location in distributed sensor networks. *In the International Symposium on Information Technology : Coding and Computing*, pages 130–134, 2001.

K. Chakrabarty, S.S. Iyengar, H. Qi, and E. Cho. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Transactions on Computers*, 51 :1448–1453, 2002.

V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4 :233–235, 1979.

C. A. Coello Coello and M. S. Lechuga. Mopso : A proposal for multiple objective particle swarm. *In Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1051–1056, 2002.

K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm : Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6 :182–197, 2002.

G.W. DePuy, G.E. Whitehouse, and R.J. Moraga. Using the meta-raps approach to solve combinatorial problems. *In Proceedings of the 19th Industrial Engineering Research Conference*, page 21, 2002.

S.S. Dhillon and K. Chakrabarty. Sensor placement for effective coverage and surveillance in distributed sensor networks. *In the 3th IEEE International Conference on Wireless Communications and Networking*, pages 1609–1614, 2003.

M.L. Fisher and P. Kedia. Optimal solution of set covering/partitioning problems using dual heuristics. *Management Science*, 36 :674–688, 1990.

G. Fusco and H. Gupta. ϵ-net approach to sensor k-coverage. *In Proceedings of the 4th International Conference on Wireless Algorithms, Systems, and Applications*, pages 104–114, 2009.

M.R. Garey and D.S. Johnson. Computers and intractability : A guide to the theory of np-completeness. *San Francisco, CA : Freeman*, 1979.

F. Gavril. Apparait dans Garey and Johnson (1979). 1974.

A. Ghosh and S. K. Das. Coverage and connectivity issues in wireless sensor networks : A survey. *Pervasive and Mobile Computing*, 4 :303–334, 2008.

T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, 101 :81–92, 1997.

H. Gupta, S. R. Das, and Q. Y. Gu. Connected sensor cover : Selforganization of sensor networks for efficient query execution. *IEEE/ACM Transactions Networking*, 14 :55–67, 2006.

S. Haddadi. Simple lagrangian heuristic for the set covering problem. *European Journal of Operational Research*, 97 :200–204, 1997.

D. S. Hochbaum. Approximation algorithms for the set covering and vertex cover problems. *Society for Industrial and Applied Mathematics Journal on Computing*, 11 :555–556, 1982.

J.H. Holland. Adaptation in natural and artificial systems. *MIT Press Cambridge, MA, USA*, 1975.

J. Chen J. Jia, G. Chang, Y. Wen, and J. Song. Multi-objective optimization for coverage control in wireless sensor network with adjustable sensing radius. *Computer and Mathematics with Applications*, 57 :1767–1775, 2009.

J. Jia, J. Chen, G. Chang, and Z. Tan. Energy efficient coverage control in wireless sensor networks based on multi-objective genetic algorithm. *Computer and Mathematics with Applications*, 57 : 1756–1766, 2009.

R. M. Karp. Reducibility amnong combinatorial problems. *In Complexity of Computer Computations*, pages 85–103, 1972.

W. C. Ke, B. H. Liu, and M. J. Tsai. The critical-square-grid coverage problem in wireless sensor networks is np-complete. *Computer Networks*, 55 :2209–2220, 2011.

J. Kennedy and R. Eberhart. Particle swarm optimization. *In the 4th IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

A. Konstantinidis and K. Yang. A multi-objective energy efficient dense deployment in wireless sensor networks using a hybrid problem specific moea/d. *Applied Soft Computing*, 11 :4117–4134, 2011.

A. Konstantinidis, K. Yang, Q. Zhang, and D. Zeinalipour-Yazti. A multi-objective evolutionay algorithm for the deployment and power assignment problem in wireless sensor networks. *Computer Networks*, 54 :960–976, 2010.

G. Lan and G.W. DePuy. On the effectivenes of incorporating randomness and memory inti a multi-start metaheuristic with application to the set covering problem. *Computers & Industrial Engineering*, 51 :362–374, 2006.

G. Lan, G.W. DePuy, and G.E. Whitehouse. An effective and simple heuristic for the set covering problem. *European Journal of Operational Research*, 176 :1387–1403, 2007.

J.Y. Lee, J-H Seok, and J-J Lee. Multiobjective optimization approach for sensor arrangement in a complex indoor environment. *IEEE transactions on systems, man, and cybernetics-part B : applications and reviews*, 42 :174–186, 2012.

W. Leigh, D. Ali, C. Ezell, and N. Paz. A branch-and-bound algorithm for implementing set covering model expert systems. *Computers & Operations Research*, 11 :464–467, 1988.

W. H. Liao, Y. Kao, and Y. S. Li. A sensor deployment approach using glowworm swarm optimization algorithm in wireless sensor networks. *Expert Systems with Applications*, 38 :12180–12188, 2011.

C. Lin, G. Wu, F. Xia, M. Li, L. Yao, and Z. Pei. Energy efficient ant colony algorithms for data aggregation in wireless sensor networks. *Journal of Computer and System Sciences*, 78 :1686–1702, 2012.

C. Liu, K. Wu, Y. Xiao, and B. Sun. Random coverage with guaranteed connectivity : Joint scheduling for wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17 :562–575, 2006.

Z. Liu, Q. Zheng, L. Xue, and X. Guan. A distributed energy-efficient clustering algorithm with improved coverage in wireless sensor networks. *Future Generation Computer Systems*, 28 :780–790, 2012.

F.B. Lopes and L.A. Lorena. Surrogate heirstic for set covering problems. *European Journal of Operational Research*, 79 :138–150, 1994.

E. Masazade, P. K. Varshney, and G. K. Sendur. A multiobjective optimization approach to obtain decision thresholds for distributed detection in wireless sensor networks. *IEEE transactions on systems, man, and cybernetics-part B : cybernetics*, 40 :444–457, 2010.

S. Mini, S. K. Udgata, and S. L. Sabat. Artificial bee colony based sensor deployment algorithm for target coverage problem in 3-d terrain. *In Proceedings of the 7th International Conference on Distributed Computing and Internet Technology*, pages 313–324, 2011.

M. Le Berre, M. Rebai, F. Hnaien, and H. Snoussi. A bi-objective model for wireless sensor deployment considering coverage and tracking applications. *Accepted in the International Journal of Sensor Networks.*

M. Le Berre, F. Hnaien, and H. Snoussi. Multi-objective optimization in wireless sensors networks. *In International Conference on Microelectronics*, pages 1–4, 2011.

M. Le Berre, F. Hnaien, and H. Snoussi. Optimisation multiobjectif dans les réseaux de capteurs sans fils distribués. *Dans le 13eme congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision*, pages 1–2, 2012.

M. Le Berre, F. Hnaien, and H. Snoussi. A multi-objective modeling of k-coverage problem under accuracy constraint. *In the 5th International Conference on Modeling, Simulation and Applied Optimization*, pages 1–6, 2013.

W. Mo, D. Qiao, and Z. Wang. Lifetime maximization of sensor networks under connectivity and k-coverage constraints. *In Proceedings of the 2nd IEEE International Conference on Distributed Computing in Sensor Systems*, pages 422–442, 2006.

Z. Naji-Azimi, P. Toth, and L. Galli. An electromagnetism metaheuristic for the unicost set covering problem. *European Journal of Operational Research*, 205 :290–300, 2010.

S.C. Oh, C.H. Tan, F.W. Kong, Y.S. Tan, K.H. Ng, G.W. Ng, and K. Tai. Multiobjective optimization of sensor network deployment by a genetic algorithm. *In the IEEE Congress on Evolutionary Computation*, pages 3917–3921, 2007.

G. Pampara and AP. Engelbrecht. Binary artificial bee colony optimization. *In the IEEE Symposium on Swarm Intelligence*, pages 1–8, 2011.

R.G. Parker and R.L. Rardin. Discrete optimization. *Academic Press. New York*, 1988.

L.S. Pessoa, M.G.C. Resende, and C.C. Ribeiro. A hybrid lagrangean heuristic with grasp and path-relinking for set k-covering. *Technical report, AT&T Labs Research*, 2010.

M. N. Rahman and M. A. Matin. Efficient algorithm for prolonging network lifetime of wireless sensor networks. *Tsinghua science and technology*, 16 :561–568, 2011.

N.S.V. Rao. Computational complexity issues in operative diagnosis of graph-based systems. *IEEE Transactions on Computers*, 42 :447–457, 1993.

M. Rebai, I. Khoukhi, H. Snoussi, and F. Hnaien. Linear models for the total coverage problem in wireless sensor networks. *In the 5th International Conference on Modeling, Simulation and Applied Optimization*, pages 1–4, 2013.

M. Rebai, **M. Le Berre**, F. Hnaien, H. Snoussi, and L. Khoukhi. A branch and bound algorithm for the critical grid coverage problem in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 1 :1–9, 2014.

Z-G. Ren, Z-R. Feng, L-J. Ke, and Z-J. Zhang. New ideas for applying ant colony optimization to the set covering problem. *Computers & Industrial Engineering*, 58 :774–784, 2010.

A. Rossi, A. Singh, and M. Sevaux. An exact approach for maximizing the lifetime of sensor networks with adjustable sensing ranges. *Computers & Operations Research*, 39 :3166–3176, 2012.

A. Rossi, A. Singh, and M. Sevaux. Lifetime maximization in wireless directional sensor network. *European Journal of Operational Research*, 231 :229–241, 2013.

R. Roth. Computer solutions to minimum cover problems. *Operations Research*, 17 :455–465, 1969.

J. O. Rourke. Art gallery theorems and algorithms. *Oxford University Press, Inc, Oxford*, 1092, 1987.

S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids : Coverage, connectivity and diameter. *Ad Hoc Networks*, 3 :702–716, 2005.

I. Slama, M. Chedly Ghedira, B. Jouaber, and H. Afifi. Cluster based wireless sensor networks' optimization under energy constraints. *In the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, pages 745–750, 2007.

M. Solar, V. Parada, and R. Urrutia. A parallel genetic algorithm to solve the set covering problem. *Computers & Operations Research*, 29 :1221–1235, 2002.

C. Song, M. Liu, J. Cao, Y. Zheng, H. Gong, and G. Chen. Maximizing network lifetime based on transmission range adjustment in wireless sensor networks. *Computer Communications*, 32 : 1316–1325, 2009.

D. Tian and N. Georganas. Connectivity maintenance and coverage preservation in wireless sensor networks. *Ad Hoc Networks*, 3 :744–761, 2005.

C. K. Ting and C. C. Liao. A memetic algorithm for extending wireless sensor network lifetime. *Information Sciences*, 180 :4818–4833, 2010.

Y.C. Tseng, Y.C. Wang, and K.Y. Cheng. An integrated mobile surveillance and wireless sensor (imouse) system and its detection delay analysis. *In Proceedings of the 8th ACM International Symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 178–181, 2005.

Y.C. Tseng, M.S. Pan, and Y.Y. Tsai. Wireless sensor networks for emergency navigation. *Computer*, 39 :55–62, 2006.

Y. B. Türkogullari and C. Ersoy N. Aras, I. K. Altinel. A column generation based heuristic for sensor placement, activity scheduling and data routing in wireless sensor networks. *European Journal of Operational Research*, 207 :1014–1026, 2010.

S. Umetani and M. Yagiura. Relaxation heuristics for the set covering problem. *Journal of Operational Research Society of Japan*, 50 :350–375, 2007.

X. Wang and S. Wang. Hierarchical deployment optimization for wireless sensor networks. *IEEE Transactions on mobile computing*, 10 :1028–1041, 2011.

L-C Wei, C-W Kang, and J-H Chen. A force-driven evolutionary approach for multi-objective 3d differentiated sensor network deployment. *In the IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pages 983–988, 2009.

Y. Wu, M. Li, Z. Cai, and E. Zhu. A distributed algorithm to approximate node-weighted minimum α-connected (θ,k)-coverage in dense sensor networks. *In Proceedings of the 2nd annual international workshop on Frontiers in Algorithmics*, pages 221–232, 2008.

M. Yagiura, M. Kishida, and T. Ibaraki. A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, 172 :472–499, 2006.

F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. Peas : A robust energy conserving protocol for long-lived sensor networks. *In Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 28–37, 2003.

Z. Yun, X. Bai, D. Xuan, T. H. Lai, and W. Jia. Optimal deployment patterns for full coverage and k -connectivity ($k \leq 6$) wireless sensor networks. *IEEE/ACM Transactions Networking*, 18 : 934–947, 2010.

H. Zhang and J.C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. *Ad-hoc and Sensor Wireless Networks*, 1 :89–124, 2005.

Q. Zhang and H. Li. Moea/d : A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11 :712–722, 2007.

Q. Zhao and M. Gurusamy. Lifetime maximization for connected target coverage in wireless sensor networks. *IEEE/ACM Transactions Networking*, 16 :1378–1391, 2008.

Z. Zhou, S. Das, and H. Gupta. Connected k-coverage problem in sensor networks. *In Proceedings of the 13th International Conference on Computer Communications and Networks*, pages 373–378, 2004.

C. Zhu, C. Zheng, L. Shu, and G. Han. A survey on coverage and connectivity issues in wireless sensor networks. *Journal of Network and Computer Applications*, 35 :619–632, 2011.

E. Zitzler and L. Thiele. An evolutionary algorithm for multiobjective optimization : The strength pareto approach. *Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich*, 1998.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms : a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3 :257–271, 1999.

E. Zitzler, M. Laumanns, and L. Thiele. Spea2 : Improving the strength pareto evolutionary algorithm. *Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich*, 2001.

Matthieu LE BERRE

Doctorat : Optimisation et Sécurité des Systèmes

Année 2014

Optimisation de déploiement et localisation de cible dans les réseaux de capteurs

Au cours de cette thèse, nous avons abordé des problématiques liées à l'optimisation de déploiement et la localisation de cible dans les réseaux de capteurs. Nous avons tout d'abord proposé un premier modèle pour l'optimisation de deux objectifs contradictoires : le nombre de capteurs déployés ainsi que la précision de la localisation. Quatre algorithmes multi-objectifs classiques ont été implémentés, et des versions hybrides ont également été proposées.
Une variante du précédent problème est également étudiée, dédiée aux applications de localisation indoor. Les algorithmes proposés pour le premier problème n'ont montré qu'une efficacité relative au cours des premières expérimentations. Une nouvelle heuristique est alors développée, et les résultats ont montré de très bonnes performances sur les instances de taille réduite, ainsi que de bien meilleures performances que les autres algorithmes implémentés sur des instances de grande taille. Enfin, la notion de connectivité et de couverture est également traitée et intégrée dans un modèle linéaire de déploiement. Un algorithme Branch and Bound a été développé afin de traiter ce problème, puis des tests ont été effectués afin de le comparer aux solveurs linéaires actuels.

Mots clés : optimisation combinatoire - réseaux de capteurs (technologie) - algorithmes d'approximation – algorithmes optimaux.

Deployment Optimization and Target Tracking in Sensor Networks

In this thesis, a joint approach for deployment optimization and target tracking in sensor networks is developed. First, we have proposed a linear model to minimize the number of deployed sensors and maximize the accuracy of the localization. We have also implemented several multi-objective methods and proposed hybridization for some of them.
We have also proposed a modification of the previous model, taking into account the indoor localization constraints. Two methods of the previous problem have been used, and a specific heuristic has been developed.
Finally, two linear models taking into account coverage and connectivity have been proposed. A Branch and Bound algorithm has also been developed, considering a geometric lower bound and two properties to reduce the number of fathomed nodes.

Keywords: combinatorial optimization - sensor networks - approximation algorithms – exact methods.

Thèse réalisée en partenariat entre :

l'Europe s'engage en Champagne-Ardenne avec le FEDER

utt université de technologie Troyes

Ecole Doctorale "Sciences et Technologies"