Étude et résolution de problèmes d'ordonnancement d'opérations d'évacuation

Kaouthar Deghdak

UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

École Doctorale MIPTIS
Laboratoire d'Informatique (EA6300)
Équipe Ordonnancement et Conduite (ERL CNRS 6305)

THÈSE présentée par :

Kaouthar DEGHDAK

soutenue le : 01 décembre 2015

pour obtenir le grade de : Docteur de l'université François - Rabelais de Tours

Discipline/ Spécialité : INFORMATIQUE

Étude et résolution de problèmes d'ordonnancement d'opérations d'évacuation

THÈSE dirigée par :

Bouquard Jean-Louis	Maître de Conférences HDR, Université François Rabelais Tours
T'kindt Vincent	Professeur des Universités, Université François Rabelais Tours

RAPPORTEURS :

Artigues Christian	Directeur de recherche HDR, LAAS CNRS, Toulouse
Leus Roel	Professeur, Université catholique de Louvain, Belgique

JURY :

Artigues Christian	Directeur de recherche HDR, LAAS CNRS, Toulouse
Bouquard Jean-Louis	Maître de Conférences HDR, Université François Rabelais Tours
Hamacher Horst	Professeur, Université Kaiserslautern, Allemagne
Leus Roel	Professeur, Université catholique de Louvain, Belgique
Prodhon Caroline	Maître de Conférences HDR, Université de Technologie de Troyes
T'kindt Vincent	Professeur des Universités, Université François Rabelais Tours

Remerciements

Durant mes trois années de thèse, de nombreuses personnes m'ont soutenue tant sur le plan personnel que professionnel.

Tout d'abord, je tiens à remercier mon directeur de thèse Vincent T'kindt pour sa bonne humeur, sa patience et le temps qu'il a investi pour ma thèse. Je voudrais par ces lignes lui exprimer toute ma gratitude pour son aide, ses conseils, son soutien indéfectible tant sur le plan professionnel que personnel. Grâce à lui, j'ai pu rencontrer et travailler avec des chercheurs de grande qualité, ce qui a été un atout et une chance exceptionnelle. Pour tout ça, je lui en serai toujours reconnaissante. Merci encore Vincent.

Je remercie également Jean-Louis Bouquard pour son encadrement et ses critiques constructives qui m'ont permise d'avancer sur mes recherches tout au long de la première année de thèse. Je remercie aussi le directeur du projet DSS_Evac_Logistic Horst Hamacher et l'équipe dont il a la direction à Kaiserslautern, pour leur gentillesse et leur hospitalité durant mon séjour en Allemagne. Merci à Jean-charles Billaut qui m'a initié à la recherche durant mon stage de master.

Merci également à l'ensemble des membres du jury et plus particulièrement aux deux rapporteurs Roel Leus et Christian Artigues pour avoir pris le temps de lire ce mémoire dans les moindres détails et jugé ce travail.

C'est d'une manière profondément sincère que je dis un grand merci à mes parents et ma famille : Yanis, Inès, Imène, Amine et Mohcine pour leur soutien, la fierté et la confiance qu'ils me portent. C'est grâce à leur encouragement que j' en suis arrivée ou je suis et c'est à eux que je dois cette réussite.

Petite dédicace aux thésards qui ont partagés mon quotidien et de simples moments : mes amis Mohamed le philosophe, Ismaila le diplomate, Azzedine l'algérien typique, Tanmoy, The-anth et bien évidemment Zeina avec laquelle j'ai partagée des moments inoubliables et tous les autres. Grâce à eux, des fous-rires ont égayé mes trois années, même dans les moments les plus difficiles. Petite pensée aussi aux anciens doctorants du laboratoire LI pour leur conseil au début de ma thèse.

Enfin, merci à mes amis en dehors du laboratoire pour les moments de détentes partagés et à tous ceux que j'aurai pu oublier.

Kaouthar DEGHDAK

Résumé

Les travaux présentés dans cette thèse, qui s'inscrivent dans le cadre du projet franco-allemand DSS_Evac_Logistic, visent à proposer des méthodes permettant de calculer des plans d'évacuation macroscopiques d'une ville lors d'une catastrophe majeure. Deux problèmes d'évacuations sont considérés dans cette thèse : le problème d'évacuation par bus et le problème d'évacuation par bus et voitures.

Le problème d'évacuation par bus a pour objectif de définir un plan d'évacuation afin de mettre à l'abri les évacués. Dans cette thèse, nous nous sommes intéressés à l'étude de trois versions du problème d'évacuation par bus. La première version est monocritère où nous cherchons à minimiser la date de fin d'évacuation. Puis, dans le second problème et afin d'assurer la sécurité des évacués, nous avons considéré une version bicritère qui généralise le cas monocritère, en incluant le risque encouru lors de l'évacuation des personnes. Les deux critères à minimiser sont la date de fin d'évacuation et le risque. La troisième version est une version robuste bicritère qui permet d'appréhender l'incertitude sur les données. Le but est de minimiser à la fois la date de fin d'évacuation et les modifications apportées sur une solution, de sorte qu'elle soit réalisable pour n'importe quel scénario de données. Pour résoudre ces problèmes d'évacuation par bus, nous avons proposé des méthodes exactes et des méthodes heuristiques.

Le second problème d'évacuation considéré par bus et voitures est multicritère. Il a pour but de définir pour chaque groupe de personnes, sa date de début d'évacuation, le centre de secours de rattachement, et le chemin menant à ce centre de secours. Nous cherchons à déterminer les centres de secours à ouvrir afin de minimiser la date de fin d'évacuation et l'exposition aux risques. Un modèle mathématique est proposé pour la résolution exacte des instances de petite taille de ce problème. Pour résoudre des instances de grande taille, des méthodes évolutionnaires et des méthodes basées sur le calcul de chemins multiobjectifs sont développées.

Dans cette thèse, toutes les méthodes proposées sont testées et validées sur des instances aléatoires et des instances réelles de la ville de Kaiserslautern du coté allemand et de la ville de Nice du coté français. Ce choix des villes a été imposé par le projet sur lequel s'inscrit cette thèse.

Mots clés : Ordonnancement d'opérations d'évacuation, plan macroscopique, optimisation robuste, optimisation multicritère, problème de plus court chemin, ordonnancement et localisation.

Abstract

The work presented in this thesis, which is a part of the Franco-German project DSS_Evac_Logistic, aims at proposing methods to calculate macroscopic evacuation plans for mid-size towns after a tremendous disaster. Two evacuation problems have been tackled in this thesis : the bus evacuation problem and bus-and-vehicle evacuation problem.

The bus evacuation problem aims at calculating an evacuation plan to relocate evacuees outside the endangered area. In this thesis, we consider three versions of the bus evacuation problem. The first one is a monocriterion problem, where the objective is to minimize the maximum evacuation time. In order to guarantee the safety of evacuees, we have considered a bicriteria problem, which is a generalization of the monocriterion version, in which we take into consideration the risk exposure of the evacuees. Consequently, the bicriteria problem is solved by minimizing the total evacuation time and the risk. The third version is a bicriteria robust version because most of the planning data is subject to uncertainty. The goal is to minimize both the evacuation time and the vulnerability of the schedule that is subject to different evacuation circumstances. To solve all the versions of the bus evacuation problem, we have developed exact solutions based on mathematical formulation to address small instances and heuristic solutions to deal with larger instances.

The second evacuation problem studied in this thesis is a multi-criteria problem, which aims at defining the departure time, the route and the shelters for each group of people. Specifically, we would like to determine a set of appropriate shelters for hosting people while minimizing the total evacuation time and the risk exposure of the evacuees. For this purpose, a mathematical formulation has been proposed to deal with small instances. For larger instances, evolutionary methods and methods based on multi-objectives shortest path searches have been developed.

For validation and experiments, all the proposed methods have been tested using random instances and real instances of Kaiserslautern city in Germany and Nice city in France. The choice of these towns is imposed by the project in which this thesis takes place.

Keywords : Scheduling evacuation operations, macroscopic plan, robust optimisation, multicriteria optimization, shortest path problems, scheduling and location.

Table des matières

Liste des tableaux

Table des figures

Introduction

Depuis toujours, les catastrophes qu'elles soient naturelles (séismes, inondations, ouragans, ...), artificielles (catastrophes nucléaires,...) ou sanitaires, touchent des populations entières dans le monde de manière hasardeuse et imprévisible. Ces catastrophes nécessitent une évacuation rapide et efficace des personnes, des zones dangereuses vers des endroits plus sûrs. Pendant la période 1970-1978, les problèmes d'évacuation lors des catastrophes naturelles causées par des ouragans ont été étudiés par plusieurs chercheurs [Urbanik, 1978, (COE) and (SWFRPC), 1979]. Après l'accident nucléaire "Three Mile Island" en 1979, les recherches sur l'évacuation ont été focalisées sur les problèmes d'évacuation d'origine nucléaire [HMM Associates, 1980, Sheffi et al., 1982a, Sheffi et al., 1982b, KLD Associates, 1984]. À la suite des ouragans Andrew 1992, Gordon 1994 et Floyd 1999, un intérêt a été porté encore une fois sur les problèmes d'évacuation d'origine naturelle. Ces problèmes concernent l'évacuation des zones urbaines ou les bâtiments (voir [Chamlet et al., 1982]).

Cette thèse se situe dans la mouvance de ces travaux et s'inscrit dans le cadre du projet franco-allemand : DSS_Evac_Logistic, qui porte sur l'évacuation à grande échelle d'une ville après une catastrophe naturelle (séisme ou tsunami) ou artificielle (crash d'avion, bombe), avec comme cas d'études les ville de Nice (France) et de Kaiserslautern (Allemagne). Nous supposons que les évacués peuvent changer éventuellement leur lieu de vie d'une période allant de quelques jours à des mois. Nous cherchons à développer des méthodes d'aide à la décision pour la logistique des problèmes d'évacuation. Ce développement est subordonné à la mise au point d'algorithmes d'optimisation multicritère, d'algorithmes de simulation et leur validation sur des scénarios réels.

Le projet DSS_Evac_Logistic est constitué de huit grandes tâches dont la quatrième tâche est l'objet de cette thèse. Cette tâche suppose la réalisation des trois premières de ce projet [Neron et al., 2011].

Nous supposons connues les données suivantes, obtenues à partir des trois premières tâches du projet : les différents scénarios de catastrophe et leurs impacts sur les infrastructures, la localisation des centres de secours et la configuration du réseau de transport (la suppression de certaines routes, la réorientation des voies à double sens en voies à sens unique et la capacité de chaque route). Pour réaliser notre tâche, nous devons ordonnancer dans le temps les différentes opérations d'évacuation. Sachant que l'évacuation simultanée de toute une population est irréalisable, nous devons décider pour chaque personne un centre de secours de rattachement ainsi que son heure d'évacuation selon des critères : urgence, sécurité des personnes, confort, etc.

Cette thèse est composée de quatre chapitres. Dans le **chapitre 1**, nous présentons un état de l'art sur les différents problèmes d'évacuation abordés dans la littérature, les modèles et les approches utilisés pour la résolution de ces problèmes. Ce chapitre servira de base bibliographique pour les travaux réalisés dans le cadre de cette thèse. Aussi, nous établissons des liens entre les problèmes existant dans la littérature et les problèmes d'évacuation abordés dans cette thèse.

Dans le **chapitre 2**, nous présentons les trois versions du problème d'évacuation par bus pour un graphe routier approximé. Dans la première version qui est monocritère, nous nous intéressons au calcul d'un planning pour les bus afin de minimiser le temps d'évacuation total. Puisque les données des problèmes d'évacuation sont incertaines, nous considérons la version robuste qui consiste à définir un plan d'évacuation réalisable pour n'importe quel scénario et performante en termes de critère à optimiser pour tous les scénarios. La troisième version est la version bicritère d'évacuation par bus où les critères à minimiser sont la date de fin d'évacuation et l'exposition des évacués au risque. Pour résoudre ces problèmes d'évacuation $\mathcal{NP}-$difficile, nous proposons des méthodes exactes qui visent à calculer des plans d'évacuation optimaux. Nous développons également des méthodes heuristiques pour calculer rapidement des solution approchées.

Dans le **chapitre 3**, nous nous intéressons à un problème général d'évacuation. Nous voulons évacuer des flots de bus et de voitures. Contrairement aux problèmes présentés dans le chapitre 2, nous considérons un graphe routier réel. Le but est de définir des plans d'évacuation pour les voitures et pour les conducteurs de bus afin de choisir les centres de secours à ouvrir tout en minimisant la date de fin d'évacuation et le risque encouru durant l'évacuation. Pour une résolution optimale des instances de petite taille de ce problème, nous avons proposé un modèle mathématique en nombre entier. Par contre pour la résolution des instances de grande taille, nous développons trois types d'heuristiques : un algorithme génétique et une heuristique par décomposition.

Dans le **chapitre 4**, nous nous intéressons à l'étude d'un problème d'ordonnancement et de localisation (*ScheLoc Problem*). Ce problème implique deux niveaux de décision : la localisation des machines et l'ordonnancement des travaux sur ces machines. L'objectif est double : nous choisissons un sous-ensemble de l'ensemble des localisations pour placer ces machines, puis nous ordonnançons les travaux sur ces machines afin d'optimiser un ou plusieurs critères d'ordonnancement. Pour résoudre ce problème, nous avons proposé un modèle mathématique et des heuristiques basées sur le clustering. Pour évaluer l'efficacité de ces heuristiques proposées dans le cas des instances de grande taille, nous avons développé des bornes inférieures. L'intérêt du problème *ScheLoc*, qui est un travail prospectif par rapport aux chapitres 2 et 3, est d'aborder conjointement localisation et ordonnancement comme ce qui pourrait être fait efficacement à termes pour le projet.

Dans le **chapitre 5**, nous présentons le logiciel d'aide à la décision nommé *Visual Flow* incorporant la majorité des algorithmes développés dans cette thèse.

Chapitre 1

Les problèmes d'évacuation sous l'angle de la RO

Le calcul des plans d'évacuation a été développé pour des structures (bâtiments, bateaux et avions), des parties de zones urbaines ou des villes. Dans ce chapitre, nous nous intéressons essentiellement aux travaux portant sur le calcul des plans d'évacuation des villes. Ce chapitre est constitué de deux sections principales. Dans la première, nous présentons un état de l'art sur les problèmes d'évacuation qui servira de près ou de loin à l'élaboration de cette thèse. Cet état de l'art débute par des rappels sur les modèles d'évacuation existant dans la littérature. Plus précisément, les modèles macroscopiques et les modèles microscopiques. Ensuite, nous présentons des approches basées sur les problèmes d'optimisation et les approches basées sur la simulation qui visent respectivement à déterminer des plans d'évacuation et à évaluer un plan d'évacuation existant. Dans la deuxième section de ce chapitre, nous présentons brièvement les problèmes d'évacuation abordés dans cette thèse en les positionnant par rapport à l'état de l'art.

1 État de l'art

1.1 Différents modèles d'évacuation

Avant de présenter les modèles d'évacuation abordés dans la littérature, nous commençons par la définition de la notion de la durée d'évacuation. D'après [Lovas, 1998] et [Graat et al., 1999], le temps d'évacuation est défini comme la durée nécessaire pour évacuer les personnes présents dans la zone sinistrée. Cette durée est composée de trois parties : la première est le temps nécessaire pour détecter la nécessité de l'évacuation des personnes. La deuxième est le temps nécessaire pour déclencher le processus d'évacuation. Enfin, la troisième partie est le temps nécessaire pour évacuer les personnes vers des zones sécurisées (figure 1.1). Dans la plupart des modèles d'évacuation proposés, le temps d'évacuation des personnes est le temps de l'évacuation de la dernière phase. C'est-à-dire une borne inférieure de la solution du problème d'évacuation réel.

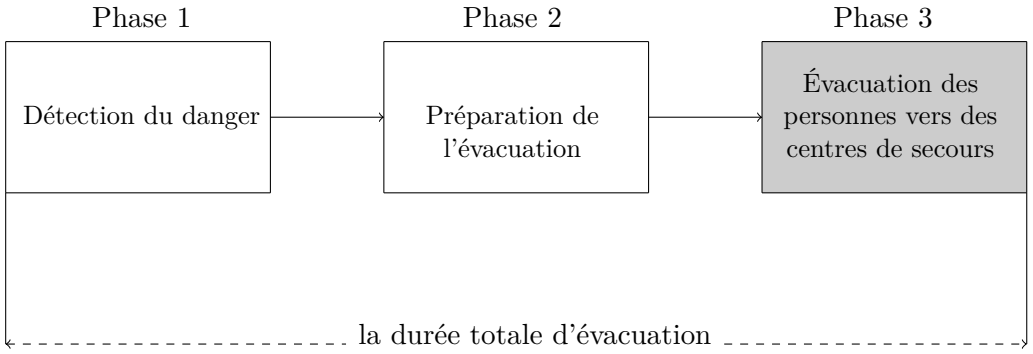


FIGURE 1.1 – Les phases d'évacuation

Dans la littérature, la modélisation des problèmes d'évacuation est basée sur l'une des approches suivantes. La première utilise les problèmes de flots (*network flow problem*). La seconde est basée sur les problèmes d'affectation du trafic (*traffic assignment problem*). Enfin, la troisième repose sur des modèles de simulation du trafic. Ces modèles sont détaillés dans les sections 1.2.1 et 1.2.2. En général, l'objectif de ces modèles est de minimiser la date de fin d'évacuation ou de maximiser le nombre de personnes évacuées. Principalement, ils sont classés en deux grandes catégories. La première concerne les modèles macroscopiques pour lesquels les personnes sont homogènes, i.e., ayant un comportement commun connu. La deuxième est celle des modèles microscopiques qui permettent d'appréhender des comportements spécifiques à toutes les personnes, résultant du contexte de l'évacuation (stress, panique, vitesse, etc.).

Les modèles macroscopiques sont utilisés pour avoir une bonne première estimation sur la durée de l'évacuation. Pour cette raison, la plupart de ces modèles sont basés sur les problèmes de flots statiques ou dynamiques (le problème du plus court chemin, le problème de flots à coût minimum, etc.).

Pour les modèles microscopiques, leur but est de décrire le comportement de chaque individu. Principalement, ils sont classés en quatre catégories : les modèles de voiture-

suiveuse ou modèles de poursuite, les modèles à automates cellulaires, les modèles de file d'attente et les modèles multi-agents. L'inconvénient de ces modèles est leur utilisation d'un nombre assez grand de données. Par exemple, pour l'évacuation par véhicules, le modèle doit traiter toutes les manœuvres qu'un conducteur peut être amené à réaliser : changement de voie, dépassement, gestion des distances entre les véhicules, etc.

1.2 Problèmes d'évacuations : modèles et approches de résolution

Nous distinguons deux types de modélisation des problèmes d'ordonnancement d'opérations d'évacuation : les modèles basés sur des problèmes d'optimisation et ceux basés sur des modèles de simulation.

1.2.1 Modèles basés sur les problèmes d'optimisation

Le choix des différents chemins d'évacuation pour les personnes est un aspect crucial lors d'une opération d'évacuation car il détermine sa réussite ou son échec. L'optimisation des opérations d'évacuation peut être vue selon deux approches, que nous détaillons dans les sections suivantes : soit cette optimisation est réalisée sur des modèles de flots, soit sur des modèles d'affectation du trafic. L'approche par flots est généralement utilisée pour les modèles macroscopiques. Pour les modèles microscopiques, l'approche utilisée est l'affectation du trafic. Parfois, cette dernière est utilisée pour les modèles macroscopiques.

Modèles de flots

Les premières modélisations sont basées sur des calculs de plus courts chemins. [Fahy, 1994] a proposé un algorithme de plus court chemin pour l'évacuation d'une tour. Ce genre d'algorithme a été utilisé par [Yamada, 1996] pour définir un plan d'évacuation d'une ville. Ce plan fait abstraction de la capacité d'accueil des centres de secours. [Yamada, 1996] a proposé également une autre modélisation plus efficace en utilisant les problèmes de flots statiques à coût minimum qui permettent de prendre en compte la capacité des centres de secours. La fonction objectif est la minimisation de la distance totale parcourue par tous les évacués pour rallier les centres de secours.

Une variante du problème du plus court chemin, appelée problème du chemin le plus rapide (*quickest path problem*) a été également utilisée (voir [Chen and Chin, 1990], [Hung and Chen, 1991], [Kagaris et al., 1999]). Son objectif est d'évacuer rapidement les personnes (des flots) à travers un seul chemin à partir d'un nœud source vers un nœud destination. Les flots de personnes sont envoyés à plusieurs reprises. Le nombre de flots envoyés est calculé à partir de la durée nécessaire pour traverser ce chemin et de sa capacité. Cette modélisation est couramment utilisée dans le cas de l'existence d'un seul chemin d'évacuation. Par exemple, l'évacuation par une seule sortie des spectateurs d'un concert ou un événement sportif. Malgré leur efficacité, les algorithmes de flots statiques ne sont pas applicables pour les problèmes d'évacuation tenant compte de l'aspect temporel. Par conséquent, il est pertinent de les modéliser par des problèmes de flots dynamiques (flots statiques dupliqués dans le temps).

Les problèmes de flots dynamiques sont utilisés de manière extensive pour modéliser

les problèmes d'évacuation d'une structure (des immeubles par exemple). Nous citons les travaux de [Chamlet et al., 1982], [Choi et al., 1988], [Kisko and Francis, 1985], [Dunn, 1992] et [Opasanon and Miller-Hooks, 2009]. Dans ce contexte, [Choi et al., 1988] ont proposé trois modélisations où ils ont montré que la capacité résiduelle d'arc, à chaque instant de temps, dépend des flots passés par cet arc. Nous trouvons également un état de l'art détaillé sur la modélisation mathématique des problèmes d'évacuation de structures dans [Hamacher and Tjandra, 2001]. Dans cet état de l'art, ils ont extensivement étudié les modèles macroscopiques et d'une façon générale les modèles microscopiques. Tous ces modèles sont capables de refléter les flots d'évacuation des personnes au fil du temps. L'approche macroscopique peut optimiser le système indépendamment des comportements des évacués. L'approche microscopique, quant à elle, est capable de capturer et d'utiliser les comportements de chaque évacué. Pour l'approche macroscopique, le modèle de flots dynamiques à coût minimum permet d'estimer la durée moyenne d'évacuation par évacué. Les modèles de flots maximum (dynamiques) sont utilisés pour estimer le nombre maximal de personnes évacuées durant une période de temps. Les modèles du chemin le plus rapide permettent d'estimer la durée minimale nécessaire pour évacuer un nombre donné de personnes. L'approche microscopique est utilisée souvent pour simuler les plans d'évacuation. Les modèles les plus répandus sont les automates cellulaires. [Hamacher and Tjandra, 2001] stipulent que ces modélisations peuvent être appliquées dans le cas d'une évacuation à grande échelle.

[Opasanon and Miller-Hooks, 2009] ont proposé un nouveau problème qui est une variante du problème du chemin de probabilité maximale (*Maximum probability path*). Cette variante est appelée problème d'évacuation la plus sûre (*the safest Escape problem, SEP*). Ce problème permet de fournir un plan d'évacuation des personnes lors d'une évacuation d'un immeuble ou d'une zone urbaine. Son objectif est de déterminer un ensemble d'arcs et un nombre d'évacués qui peuvent être envoyés à travers chaque chemin, de sorte que la probabilité de succès de la personne la plus en danger soit maximale. Le réseau routier considéré est dynamique et varie au cours du temps. Les capacités des arcs sont recalculées au cours du temps, et les durées de traversées des arcs et les probabilités de succès sur chaque arc changent également au cours du temps. Ce réseau routier est modélisé par un graphe avec une seule source et une seule destination. [Opasanon and Miller-Hooks, 2009] ont proposé un algorithme optimal de complexité pseudo-polynomial. Cet algorithme a été testé sur des instances dont le réseau routier est inférieur à 500 nœuds. L'exemple suivant montre le principe de ce problème.

Exemple 1 *Nous considérons deux graphes du problème SEP. Nous supposons que nous voulons évacuer trois personnes du nœud 1 au nœud 4. Le chemin qui a une probabilité minimale de succès est coloré en bleu. La solution optimale du graphe de la figure (b) est optimale pour le problème SEP (deux personnes traversent le chemin 1-2-4 et une personne traverse le chemin 1-3-4), car la probabilité de succès associée au chemin 1-2-4 dans la figure (b) est supérieure à la probabilité minimale du chemin 1-3-4 du graphe (a) qui est égale à 0,08.*

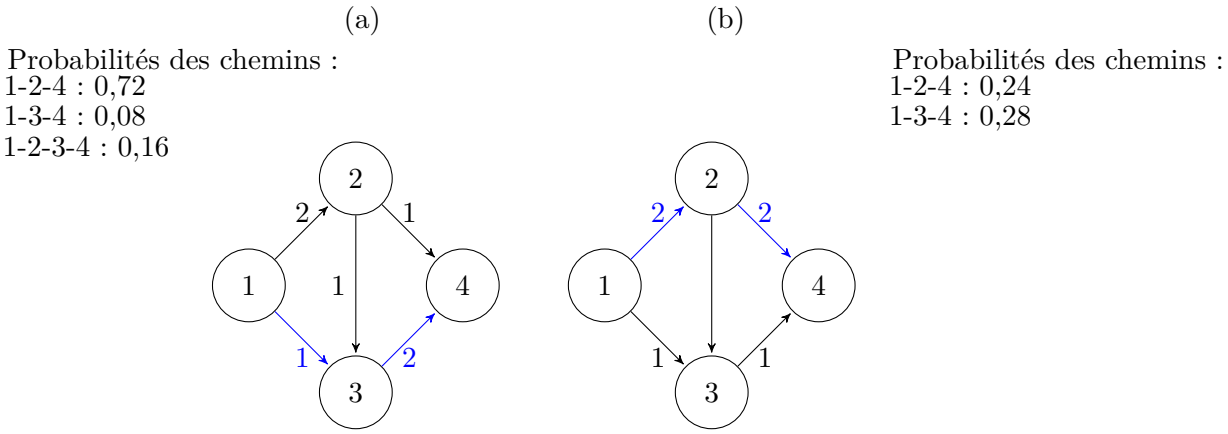


FIGURE 1.2 – Problème d'évacuation le plus sûr ([Opasanon and Miller-Hooks, 2009])

[Bretschneider and Kimms, 2012] s'intéressent au problème de choix du nombre de voies sur chaque intersection, à l'organisation des flots de voitures aux intersections et aux distributions des flots sur le réseau routier afin de minimiser la date de fin d'évacuation. Ils introduisent une heuristique basée sur un modèle mathématique qui résout ce problème en deux étapes. Des tests expérimentaux sont menés sur des instances aléatoires de petites tailles.

Une étude très récente qui traite des problèmes d'évacuation de personnes par bus est due à [Bish, 2011]. Cette étude est la première qui a géré la tournée de bus pour évacuer les personnes. Cela s'explique par le fait que l'évacuation par véhicules s'est avérée dangereuse, coûteuse et la gestion de la logistique associée est complexe. À titre d'exemple, nous citons le relevé des conclusions suite à l'ouragan Katrina, [United States Coast Guard , 2006]. Le réseau routier est modélisé par un graphe orienté qui ne tient pas compte des capacités des arcs (la capacité est le nombre maximal de bus pouvant traverser un arc par unité de temps). La figure 1.3 représente un exemple de réseau routier pour le problème d'évacuation par bus ([Bish, 2011]). Dans cet exemple, nous avons un dépôt qui est le point de départ des bus, deux points de rassemblement (PR) et trois centres de secours (CS). Sur chaque arc, une valeur numérique représente la durée de sa traversée.

Le problème d'évacuation par bus, noté BEP, est un problème de flots discrétisés qui présente des analogies avec les problèmes de tournées de véhicules (VRP). Les différences entre les deux concernent la fonction objectif et la structure du réseaux routier. Dans le BEP, la fonction objectif est de type $\min - \max$. Par contre, dans le VRP la fonction objectif est de type $\min -cost$ (le coût de transport de l'évacuation n'est pas un élément prioritaire). Concernant la structure du réseau routier, pour le VRP, nous avons un seul dépôt. Par contre dans le BEP, celui-ci est divisé en plusieurs points : le point de départ (l'emplacement initial des bus avant l'évacuation), et les centres de secours (qui ont une restriction sur la capacité d'accueil des personnes évacuées). Ces différences rendent le problème BEP significativement plus difficile à résoudre. [Bish, 2011] a considéré pour le même tour qu'un bus qui n'est pas plein peut évacuer plusieurs points de rassemblement. Il a montré que la solution optimale du problème BEP dépend du nombre de bus disponibles.

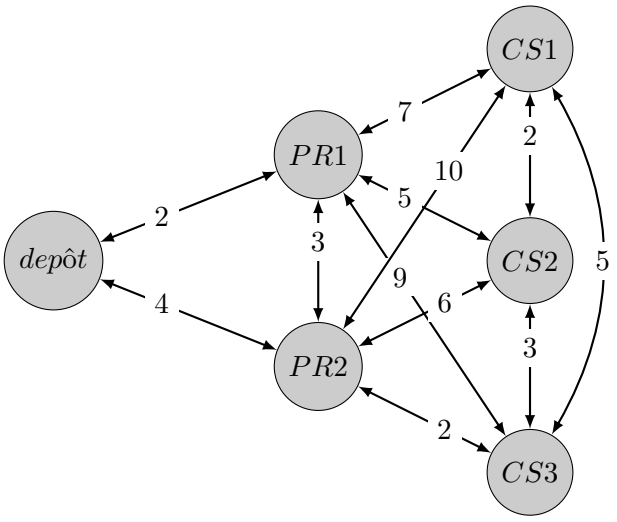


FIGURE 1.3 – Type de réseau routier pour le problème BEP ([Bish, 2011])

Pour la résolution de ce problème, [Bish, 2011] a proposé deux modèles mathématiques de programmation linéaire en nombres entiers. Ces modèles ont été testés sur des réseaux de petite taille. Pour des tailles plus grandes, il a proposé deux heuristiques, notées respectivement H_1 et H_2. L'heuristique H_1 est basée sur une recherche locale et l'heuristique H_2 utilise conjointement H_1 et les deux modèles mathématiques. L'heuristique H_2 a pour but d'améliorer la solution obtenue via l'heuristique H_1. L'heuristique H_2 est composée de trois étapes. La première étape concerne le calcul d'une solution par H_1. La deuxième étape est la résolution de la relaxation continue du premier modèle. Enfin, la dernière étape est la résolution du deuxième modèle en imposant les conditions suivantes : une limite du temps, toutes les variables qui déterminent les routes (obtenues à partir de la première et la deuxième étape en éliminant les duplicatas) sont fixées. Les résultats expérimentaux ont montré que H_1 permet de calculer des solutions faisables mais non optimales. Par contre, dans le cas de petites instances, H_2 donne généralement des solutions optimales. Dans le cas de grandes instances, H_2 donne des solutions qui améliorent les solutions de H_1 avec un temps minimal de 2000 secondes.

Par la suite, [Goerigk et al., 2013] ont étudié une variante du problème introduit par [Bish, 2011] afin de minimiser la date de fin d'évacuation. Pour cela, ils ont proposé un modèle mathématique et une procédure par séparation et évaluation. Cette dernière permet de résoudre des instances réelles de la ville de Kaiserslautern (Allemagne). Par la suite, une version robuste de ce problème a été abordée par [Goerigk and Grun, 2014], où l'incertitude liée au nombre de personnes à évacuer. La résolution exacte de ce problème est obtenue grâce à un modèle mathématique en nombres entiers. Ce dernier est capable de résoudre à l'optimalité des instances aléatoires de petites tailles. Également, ils ont proposé une méthode tabou pour résoudre des instances de grandes tailles et des instances de la ville de Kaiserslautern. Une étude similaire à celle de [Goerigk and Grun, 2014] a été proposée par [Kulshrestha et al., 2014] qui considèrent le problème robuste d'évacuation par bus. Comme dans [Goerigk and Grun, 2014], l'incertitude concerne le nombre d'évacués et le

critère à minimiser est la date de fin d'évacuation. Contrairement à [Goerigk and Grun, 2014], [Kulshrestha et al., 2014] considèrent le problème de localisation des points de rassemblement et supposent qu'un bus est affecté au début de l'évacuation à un seul point de rassemblement. En revanche, [Goerigk and Grun, 2014] abordent le problème de routage des bus depuis les points de rassemblement vers les centres de secours et inversement.

[Goerigk et al., 2014b] ont aussi étudié le problème d'évacuation par bus avec l'intégration des décisions de localisation. Ce modèle permet la planification des tournées de bus et aide le planificateur à choisir la localisation des points de rassemblement et la localisation des centres de secours à ouvrir. Une méthode de résolution de type Branch and Price a été proposée pour la résolution simultanée du problème d'ordonnancement des tournées de bus et du problème de localisation, afin de minimiser la date de fin d'évacuation. Cette méthode a été testée sur des instances réelles et aléatoires.

[Bretschneider, 2012] a abordé un autre problème d'évacuation avec commodités multiples (voitures et bus). Le but de l'étude de ce problème est de définir un planning et des routes pour les bus et les voitures. Le critère à minimiser est la combinaison linéaire des flots des commodités arrivées à chaque destination et le nombre de lignes d'urgence utilisées pour l'évacuation par bus. Pour cela, [Bretschneider, 2012] a proposé une heuristique basée sur la programmation mathématique capable de résoudre des instances de petites tailles en un temps raisonnable.

Modèles d'affectation de trafic

L'affectation du trafic est l'une des approches qui a été excessivement utilisée pour la modélisation des problèmes d'évacuation. Dans cette section, nous commençons par des rappels sur les problèmes d'affectation du trafic : les types d'affectation et leurs principes, les avantages et les inconvénients de chaque type. Ensuite, nous citons les travaux utilisant les problèmes d'affectation du trafic.

Le problème d'affectation du trafic consiste à répartir les demandes de déplacement pour chaque couple (origine-destination, O-D) sur le réseau en tenant compte du choix des usagers et l'offre du système de transport. Ce problème implique, la modélisation de choix des usagers, et la modélisation de l'écoulement du trafic. Ce dernier consiste à représenter le choix individuel des usagers en termes de déplacement. Nous distinguons deux types d'affectation du trafic : l'affectation statique et l'affectation dynamique.

Un problème d'affectation statique est une représentation simplifiée des déplacements sur une surface d'étude et une période de temps données. Cette représentation considère que la durée de traversée des routes ne change pas au cours du temps est indépendante des dates de départ des usagers. En plus, la congestion est supposée constante. Ces modèles sont largement utilisés pour évaluer les modifications importantes du système de transport (ouverture d'une nouvelle route, amélioration de la fréquence de desserte d'une ligne de transport en commun, ...) ou pour des politiques publiques (aménagements urbains, tarification des transports, ...) sur les déplacements de la zone étudiée.

De manière succincte, les modèles dynamiques s'appuient sur une demande de transport qui évolue dans la période d'étude (la matrice O-D est dépendante du temps). Ils sont décrits comme suit : les usagers font face à un double choix, celui de l'heure de départ et celui de l'itinéraire. Le coût généralisé associé à un trajet est la somme pondérée du temps

de déplacement et d'une fonction des pénalités dues aux arrivées précoces ou tardives. Chaque individu choisit l'heure de départ et l'itinéraire qui minimisent cette fonction de coût généralisé. Un équilibre dynamique est atteint lorsqu'aucun usager ne peut modifier ni son itinéraire ni son heure de départ pour optimiser son coût généralisé.

L'affectation du trafic se fait selon l'un des deux principes suivants. Le premier principe ([Wardrop, 1952]) correspond à un *équilibre d'usagers* (UE) : "chaque individu utilise l'itinéraire qui lui procure le coût minimal généralisé pour un couple origine-destination O-D donné". Ce principe suppose que les usagers sont identiques, autonomes et ont une information complète sur le réseau routier. Ce principe d'affectation a été utilisé dans le cas de la modélisation des problèmes d'évacuation par [Rathi and Solanki, 1993] et [Hobeika and Kim, 1998]. Il est fiable pour la gestion du trafic de la vie courante, et à un degré moindre, dans le cas des problèmes d'évacuation. En effet, les évacués n'ont pas des informations fiables et complètes sur l'état du réseau routier (routes supprimées, congestion de certaines routes,...). En plus, dans certains cas les évacués ne sont pas autonomes : ils peuvent être guidés par des informations dans le but de gérer la saturation du réseau de transport et de minimiser la date de fin d'évacuation. Dans ce cas, le principe d'affectation du trafic utilisé est le deuxième principe de Wardrop ([Wardrop, 1952]), appelé l'*optimum social* (SO) : "le partage des usagers entre les itinéraires concurrents se fait de telle sorte que le coût social généralisé (pour l'ensemble du système) soit minimum". En d'autres termes, l'affectation du trafic pour certains usagers n'est optimale que pour le bénéfice de tout le système. En général, ces deux équilibres ne sont pas équivalents ([Henn, 2001]). Pour les travaux sur l'évacuation utilisant ce dernier principe, nous citons [Barrett et al., 2000], [Chiu, 2004], [Tuydes and Ziliaskopoulos, 2004], [Tuydes and Ziliaskopoulos, 2006]), [Liu et al., 2005], [Liu et al., 2006], [Sbayti and Mahmassani, 2006], [Yuan et al., 2006] et [Chiu et al., 2007].

Exemple 2 *Dans cet exemple, nous essayons d'éclaircir le principe d'équilibre d'usager ([Wardrop, 1952]). Nous avons un réseau avec une seule destination et trois zones. Nous voulons faire passer des personnes à partir de la première et la deuxième région vers la destination (D). Plusieurs alternatives sont possibles pour arriver à cette destination. Par exemple, les personnes se trouvant dans la zone 1 ont deux possibilités : rester sur la zone 1 (1D), ou bien passer par la zone 3 (1-3-D).*

Les figures 1.5a et 1.5b représentent respectivement la vitesse des personnes dans chaque zone et le ratio de la population qui a choisi de prendre le chemin 1→D (θ_{1D}) ou le chemin 2 → D (θ_{2D}). À $t = 0$, nous supposons que la région 3 est congestionnée et la vitesse des personnes est très faible (figure 1.5a). Par conséquent, environ 90% ($\theta_{1D} = \theta_{2D} \simeq 0.9$) des personnes de la première zone et les personnes de la deuxième zone choisissent respectivement les chemins (1→D) et (2 → D) au lieu de passer par la troisième zone (figure 1.5b). Après une période de temps, les zones 1 et 2 seront les plus utilisées et aussi risquent d'être congestionnées. Donc une partie des personnes ont tendance à choisir les chemins qui passent par la zone 3 : 1→ 3 →D et 2→ 3 →D. Par exemple, après une heure de temps, juste 40% ($\theta_{1D} \simeq 0.4$) des personnes choisissent le chemin 1→ D, par contre 60% utilisent le chemin 1→ 3 →D (cf. figure (1.5b)). À la fin de cette simulation (cf. figure 1.5a), les vitesses sur les trois zones indiquent que le trafic sera fluide (pas de congestion). Ceci entraine que le pourcentage des choix de toutes les

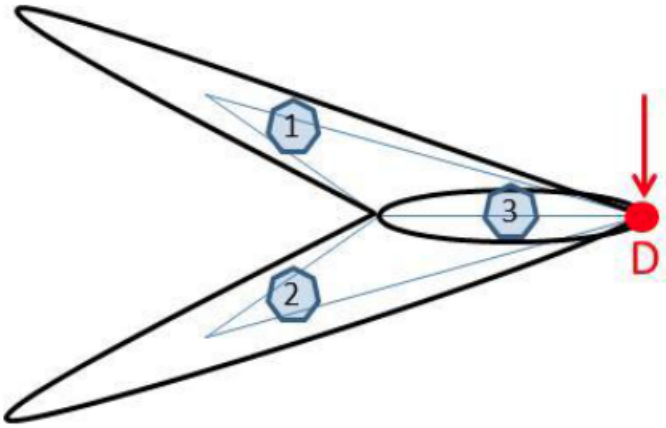
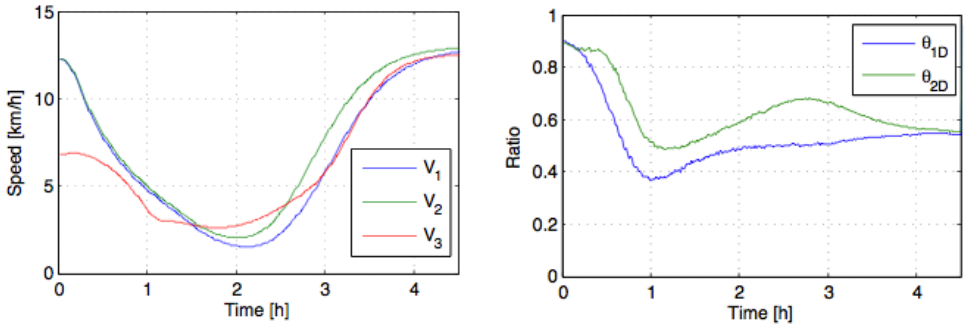


FIGURE 1.4 – Réseau à une destination et multiples régions [Yildirimoglu and Geroliminis, 2013]

routes est aux environ de 50% (cf. figure (1.5b)).



(a) Évolution de la vitesse de chaque zone (b) Ratio d'évolution de choix des routes

FIGURE 1.5 – Évolution de l'affectation du trafic[Yildirimoglu and Geroliminis, 2013]

Cet exemple montre que les usagers choisissent les chemins qui leurs procurent une arrivée au plutôt à la destination (D), en prenant en compte l'état du réseau routier ainsi que leurs localisations à l'intérieur de leurs régions.

Concernant l'évacuation lors d'une catastrophe nucléaire, [Hobeika and Kim, 1998] ont proposé un algorithme basé sur les problèmes d'affectation statique du trafic et ont implémenté l'algorithme de [Dial, 1969] pour établir un plan d'évacuation d'une ville. Ils ont conclu que la méthode d'équilibre d'usagers (UE) donne une minimisation de la date de fin d'évacuation et un temps de calcul meilleurs que l 'algorithme de Dial.

Le principe d'équilibre d'usagers a été largement appliqué pour modéliser le comportement attendu des utilisateurs du réseau routier dans une situation normale. Ce principe est évidemment inutilisable dans le cas où les utilisateurs n'ont pas d'informations com-

plètes sur la situation du réseau routier. Dans ce cas, ils doivent suivre des informations de routage obtenues par le principe de l'optimum social ([Tuydes, 2005]).

Les premières études qui montrent la nécessité de considérer une affectation dynamique du trafic (DTA) ont été proposées par [Sheffi et al., 1982a] et [Sheffi et al., 1982b]. Ils ont mis l'accent sur l'importance de fournir des informations aux évacués concernant l'état du réseau routier. Ce dernier est une fonction de leur comportement. Cette stratégie est utile dans le cas des problèmes d'évacuation où les informations fournies améliorent la performance (l'objectif) globale du système. Pour que celle-ci soit réalisable, il faut modéliser l'évolution du trafic au sein du réseau de transport. D'où la nécessité d'utiliser une approche d'affectation dynamique du trafic.

[Barrett et al., 2000] sont les premiers à avoir utilisé le principe de l'optimum social (SO) dans le contexte d'une évacuation à grande échelle. Le modèle proposé est basé sur les problèmes d'affectation dynamique du trafic et a pour but de minimiser la date de fin d'évacuation. Malheureusement, ce modèle n'a été testé que sur un petit réseau routier. D'autres études similaires utilisant le principe de l'optimum social (SO) se trouvent dans [Tuydes, 2005], ([Liu et al., 2006]) et [Chiu et al., 2007].

[Ziliaskopoulos, 2000] a modélisé un problème d'affectation dynamique du trafic par un modèle de transmission cellulaire ([Daganzo, 1994], [Daganzo, 1995]). Le principe des modèles de transmission cellulaire est de diviser le réseau routier en plusieurs et petites cellules homogènes inter-connectées. Le débit (i.e. du nombre de véhicules par unité de temps) dépend linéairement de la densité (i.e. nombre de véhicule par unité de distance). Malgré sa simplicité, le modèle de transmission cellulaire peut décrire avec précision des phénomènes de propagation de la circulation (les perturbations et les congestions dans les réseaux routiers). Par la suite, en s'inspirant de [Ziliaskopoulos, 2000], [Tuydes, 2005] et [Liu et al., 2006] ont proposé des formulations de programmation linéaire pour trouver un plan d'évacuation optimal vers une seule destination. Malheureusement, ces formulations n'ont été testées que sur des instances aléatoires de petites taille.

Une version stochastique du modèle de transmission cellulaire ([Ziliaskopoulos, 2000]) a été proposée par [Yazici and Ozbay, 2007] dans le cas d'inondations. Les capacités des routes peuvent être décroissantes. Pour tenir compte de ces changements, les contraintes de capacité sont probabilistes. Le but de ce modèle est de minimiser la somme des dates de fin et de définir les capacités des centres de secours durant l'évacuation. Une généralisation du modèle de [Yazici and Ozbay, 2007] se trouve dans [Ng and Waller, 2010] où le nombre de personnes à évacuer et la capacité des routes sont incertaines. La fiabilité de ce modèle est assurée en incorporant l'inflation du nombre de personnes à évacuer et la déflation des capacités routières. Ce modèle a donné de bons résultats pour une instance de petite taille. L'approche stochastique nécessite la connaissance de la distribution de probabilité des données incertaines. En pratique, cette distribution de probabilité est difficile à déterminer dans le cas d'une catastrophe. Pour cette raison, le modèle robuste du modèle de transmission cellulaire a été étudié par [Yao et al., 2009] dans le cas où le nombre de personnes à évacuer est incertain. Ce nombre n'est pas supposé aléatoire, mais varie dans un ensemble borné. La fonction objectif pondérée par un poids est la minimisation de la somme des dates de fin d'évacuation. Ce poids est utilisé pour pénaliser une solution dans le cas où des personnes ne sont pas évacuées durant l'horizon de temps.

Une étude très intéressante sur l'évacuation d'une partie du réseau routier a été proposée par [Sbayti and Mahmassani, 2006]. Le but de cette étude est d'évacuer les voitures de la zone impactée vers des endroits plus sûrs à l'aide des informations de guidage reçues par les conducteurs au cours de leurs trajets. Plus précisément, ces informations concernent leurs dates de départ de la zone impactée, leurs destinations d'évacuation et leurs chemins pour atteindre ces destinations. Il est important de noter que le réseau routier est supposé vide avant l'opération d'évacuation. Pour modéliser ce problème [Sbayti and Mahmassani, 2006] ont proposé un modèle mathématique basé sur l'affectation dynamique du trafic de l'optimum social.

Une autre étude a été proposée par [Sbayti et al., 2007] qui ont utilisé la méthode des moyennes successives pour construire un plan d'évacuation d'une zone après une catastrophe. Le but est de définir pour chaque groupe de personnes, la date de début d'évacuation, le centre de secours de rattachement et le chemin à suivre pour minimiser la date de fin de l'évacuation. La procédure a été appliquée sur un sous-ensemble de véhicules qui appartiennent à la zone impactée. La simulation est faite avec un simulateur mesoscopique nommé DYNASMART ([Mahmassani, 2001]).

En marge des problèmes de calcul d'un plan d'évacuation par des piétons et des voitures, [Sherali et al., 1991] ont été les premiers à mettre l'accent sur l'importance de la localisation pour le calcul d'un plan d'évacuation. Notamment, lorsqu'il s'agit de la minimisation de la date de fin d'évacuation. [Sherali et al., 1991] supposent que les autorités ont le pouvoir de choisir la localisation des centres de secours et les routes pour les évacués. Des études similaires à celle de [Sherali et al., 1991] ont été proposées par [Kongsomsaksakul et al., 2005] et [Coutinho-Rodrigues et al., 2012]. [Coutinho-Rodrigues et al., 2012] ont abordé le problème de localisation et du choix des routes pour les piétons. Ce problème est multiobjectif et le modèle mathématique associé est en nombres entiers. Ce modèle a été appliqué pour la ville de Coimbra (Portugal). Un modèle plus réaliste que celui de [Sherali et al., 1991] a été proposé par [Kongsomsaksakul et al., 2005]. Les autorités ont la possibilité de choisir le nombre de centres et leurs localisations. Par contre, les évacués ont la prérogative de choisir leurs routes selon le premier principe de [Wardrop, 1952] (équilibre d'usager). La résolution de ce problème est obtenue par un modèle mathématique et un algorithme génétique.

1.2.2 Modèles de simulation

Les techniques de simulation sont utilisées pour visualiser la circulation dans le cadre des modèles d'affectation dynamique du trafic. Ces techniques permettent d'appréhender les comportements de chaque conducteur et de prendre facilement les mesures de contrôle adéquates. Ces techniques s'adaptent parfaitement aux modèles microscopiques. Par contre, elles sont difficiles à valider à cause du comportement aléatoire humain qui est difficile à modéliser.

Logiciels de simulation microscopiques

Les logiciels de simulation microscopiques sont utilisés pour étudier localement les problèmes de congestion du trafic. Les logiciels les plus connus sont NETSIM, VISSIM,

PARAMICS, AIMSUN/GETRAM et CORSIM.

NETSIM ([KLD Associates, 1984]) est l'un des premiers logiciels de simulation microscopique stochastique développé utilisant l'affectation du trafic. Les performances du trafic sont simulées en fonction de plusieurs stratégies de gestion du trafic et d'un grand nombre de demandes (évacués). Ce logiciel n'est utilisé que dans le cas d'évacuations après une catastrophe nucléaire. Il permet d'estimer la date de fin d'évacuation.

Le logiciel de simulation du trafic appelé PARAMICS ([University of Edinburgh, 1990]). PARAMICS (figure 1.6) a été utilisé par [Church and Sexton, 2002] pour obtenir la date de fin d'évacuation pour plusieurs scénarios d'évacuation. Ils supposent plusieurs scénarios d'évacuation et une combinaison de plusieurs niveaux de demandes d'évacuation.

Le logiciel VISSIM (*Verkher In Stadten-Simulation*) ([Visual Solutions Incorporated, 1989], mis au point par la firme allemande PTV AG Karlsuhe, modélise de manière microscopique le comportement des véhicules pas à pas. Il est composé d'un modèle de poursuite psychophysique qui associe aux actions du conducteur, en plus des manœuvres d'accélération et de freinage, la notion de niveaux de perception. Ces niveaux donnent des seuils de réaction sous l'influence des manœuvres du véhicule qui précède (voir [Wiedermann, 1974]. Le modèle attribue à chaque véhicule, de manière stochastique, une vitesse, une accélération, une décélération et une distance de sécurité de référence.

FIGURE 1.6 – PARAMICS

AIMSUN/GETRAM (*Advanced Interactive Microscopic Simulation for Urban and Nonurban Networks*) [TSS-Transport Simulation Systems, 1980] est un logiciel de simulation microscopique développé à l'Université Polytechnique de Catalogne à Barcelone. Ce logiciel contient une série d'outils (logiciels) : AIMSUN est un simulateur microscopique du trafic, qui simule les mouvements des véhicules sur le réseau routier et les piétons. AIMSUN 3D est un logiciel d'animation, TEDI est un éditeur graphique du réseau et l'expansion, tandis que GETRAM est une interface de programmation.

Logiciels de simulation macroscopiques

Contrairement aux logiciels microscopiques, les logiciels macroscopiques permettent de décrire la congestion de manière générale. [Sheffi et al., 1982a] ont développé NETVAC1 qui a été utilisé pour l'évacuation lors d'une catastrophe nucléaire. Ce logiciel macroscopique utilise des modèles de flots de trafic existants pour simuler le processus d'évacuation

à travers le réseau. [Sheffi et al., 1982a] supposent que les conducteurs (les évacués) font le choix de leur itinéraire à partir de leur connaissance préalable du réseau routier. Cet outil peut gérer de grands réseaux et est capable d'évaluer la majorité des stratégies d'évacuation. Il permet de donner des sorties sur les flux, les files d'attente, des vitesses et des temps de déplacement lors de l'évacuation. L'inconvénient de cet outil est qu'il est insensible au comportement des personnes à évacuer.

MASSVAC a été développé par [Hobeika and Jamei, 1985] pour la gestion d'évacuations dues à un ouragan. Ce modèle comporte deux niveaux d'analyse : un niveau macroscopique et un niveau microscopique. Le processus d'évacuation au niveau macroscopique prend en considération les principaux axes routiers du réseau de transport. Ce niveau offre une estimation du temps maximum d'évacuation selon différents niveaux de gravité de la catastrophe et des conditions de circulation. Le niveau microscopique traite des petites parties du réseau routier pour visualiser la congestion du trafic, la congestion des intersections et les blocages causés par les accidents. MASSVAC utilise des algorithmes d'affectation dynamique du trafic basés sur le principe d'optimum individuel.

Un autre outil qui a été utilisé dans l'évacuation après un ouragan est OREMS. OREMS est un excellent exemple de modèle de simulation conçu pour analyser les scénarios d'évacuation possibles des zones urbaines. Il identifie les meilleures itinéraires d'évacuation dans le but d'estimer le temps de la fin d'évacuation.

METACOR ([Elloumi et al., 1994], voir figure 1.7) est destiné à l'étude des voiries urbaines et autoroutières, avec une affectation dynamique. Le réseau étudié est constitué d'un graphe orienté, les accès et les sorties autoroutières sont représentés par des nœuds. Tandis que les liaisons entre ces nœuds sont représentées par des tronçons. Les deux directions de circulation sur un tronçon sont modélisées par deux tronçons disjoints. En outre, sur chacun des tronçons, les caractéristiques géométriques telles que le nombre de voies et la courbure sont supposées homogènes.

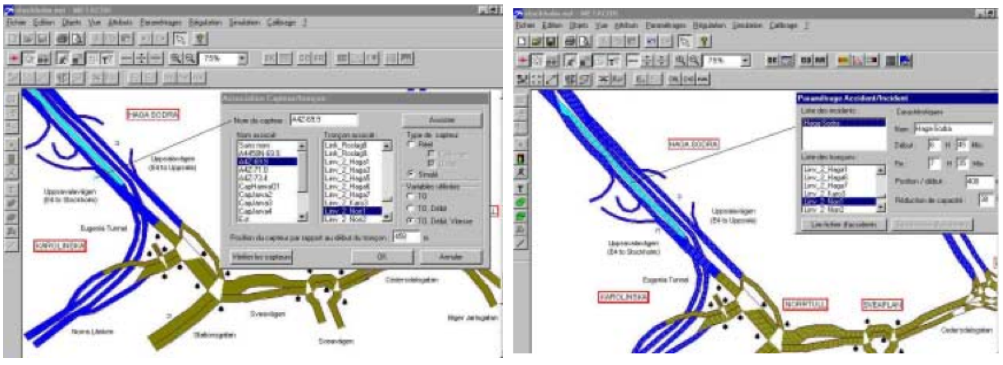


FIGURE 1.7 – METACOR

Logiciels de simulation mesoscopique

Les logiciels mesoscopiques sont une troisième catégorie de logiciels de simulation qui sont un compromis entre les modèles microscopiques et les modèles macroscopiques cités.

Ils sont caractérisés par une description d'entités de trafic à un niveau global (macroscopique) et une description des phénomènes d'interaction entre ces entités à l'échelle microscopique. Une première forme consiste à représenter un ensemble de véhicules sous forme de paquets évoluant sur des liens. La vitesse du paquet sur chaque segment est donnée par la relation vitesse/densité du lien sur lequel il se trouve.

L'un des premiers logiciels de simulation mesoscopique est CONTRAM ([Taylor, 2003]). Ce logiciel a été utilisé pour prédire les voies de circulation, les flots sur chaque route, les files d'attente et les retards au niveau des intersections. Un autre logiciel de ce type est INTEGRATION (voir [Aerde and Yagar, 1988]). Sa particularité est de combiner une modélisation individuelle des véhicules et une loi débit-vitesse pour chaque tronçon.

DYNASMART (*Dynamic Network Assignment-Simulation Model for Advanced Road Telematics*) (figure 1.8) est le logiciel de simulation le plus connu dans cette catégorie et le plus utilisé pour l'évacuation ([Sbayti et al., 2007] et [Yuan et al., 2006]). Il est basé sur la méthode d'affectation dynamique du trafic et est décomposé en deux parties. La première est la simulation mesoscopique qui repose sur le modèle proposé par [Jayakrishnan et al., 1994]. La deuxième partie est basée sur l'algorithme généralisé de moindre coût du plus court chemin (*least-cost shortest path algorithm*) développé par [Ziliaskopoulos and Mahmassani, 1993] et [Ziliaskopoulos and Mahmassani, 1996]. L'affectation des routes aux utilisateurs est faite de manière itérative à l'aide d'un algorithme développé par [Mahmassani and Peeta, 1995], [Peeta, 1994], [Peeta and Mahmassani, 1995a] et [Peeta and Mahmassani, 1995b]. Il existe deux versions de DYNASMART : P et X. La première version DYNASMART-P est une version off-line, tandis que DYNASMART-X a été conçu pour estimer et prévoir l'état courant et futur du réseau routier en temps-réel (on-line). Une description complète de ce simulateur se trouve dans [Mahmassani, 2001] .

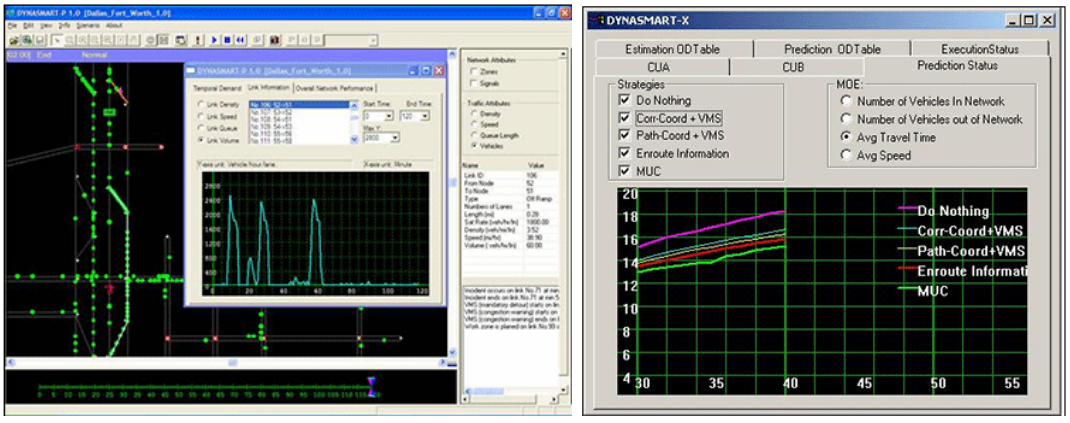


FIGURE 1.8 – DYNASMART

1.2.3 Stratégies d'évacuation

Une option pour gérer le trafic lors des évacuations est de planifier les demandes d'évacuation sur des périodes plus longues pour réduire la saturation du réseau et retarder ou éviter la dégradation du réseau. Un tel processus est appelé dans la littérature "évacuation

par niveaux". Dans certain cas, l'évacuation est plus rapide si certains évacués restent en attente. Dans la littérature, nous distinguons deux types d'approches d'évacuation par niveaux : l'évacuation par zones et l'évacuation des flots (voitures ou piétons). Dans la première, les zones sont évacuées séquentiellement en commençant par la zone la plus urgente. Dans cette approche, soit chaque zone a sa propre date de début d'évacuation, ou bien, l'évacuation peut être strictement séquentielle (i.e. pas de chevauchement entre les zones à évacuer). L'évacuation par flots est faite par véhicules où chaque véhicule a sa propre date d'évacuation.

[Chen et al., 2006] ont été les premiers à proposer des algorithmes pour l'évacuation par niveaux et l'évacuation simultanée. Ces algorithmes ont été simulés par le simulateur microscopique, nommé PARAMICS (voir section 1.2.2). Ces simulations n'ont pas été testées sur un réseau routier de taille réelle. Dans le cas où il n'y a pas de congestion sur le réseau de transport, ils ont montré que l'évacuation simultanée est meilleure (donne une date de fin d'évacuation minimale) que l'évacuation par niveaux. En revanche, l'évacuation par niveaux est meilleure que l'évacuation simultanée dans le cas de saturation du réseau de transport et d'une population de grande densité.

Une autre étude utilisant l'évacuation séquentielle par zones a été proposée par [Mitchell and Radwan, 2006]. Des plans d'évacuation sont déterminés par des heuristiques afin de réduire la date de fin de l'évacuation. Ces heuristiques utilisent un réseau de transport simplifié sous forme d'une grille contenant 100 nœuds. La densité de la population est supposée la même sur tous les points de départ. L'ensemble des points de départ est subdivisé en quatre parties. [Mitchell and Radwan, 2006] ont supposé une première date de fin d'évacuation. Elle est obtenue en évacuant simultanément à $t = 0$ toutes les personnes de tous les points de départ. Cette date est notée $TCE_{do\ nothing}$. Les heuristiques ont une même stratégie basée sur la distance entre les points d'origine et les points de destination. Elles visent à réduire la date de fin de l'évacuation en établissant un rang d'évacuation entre les quatre parties. Des tests sur les six heuristiques proposées pour un nombre de voitures compris entre 200 et 350 sur chaque point de départ et avec quatre destinations ont montré que seulement deux heuristiques améliorent $TCE_{do\ nothing}$ de 2.5 % en moyenne. Cette étude exclue la structure réelle du réseau de transport, le comportement des personnes durant l'évacuation et la densité d'une population à grande taille.

[Afshar and Haghani, 2008] ont proposé des heuristiques pour définir un plan d'évacuation avant une catastrophe naturelle (un ouragan). Pour déterminer la date de départ, le centre de secours de chaque groupe de véhicules et les chemins qu'ils doivent suivre, Afshar et Haghani ont proposé des heuristiques basées sur l'affectation dynamique des flots en fonction de la demande. Les heuristiques proposées sont classées en deux catégories. Les heuristiques de type “Spreed" visent à propager les demandes sur le réseau de transport sur des grands intervalles de temps. Ceci a pour but d'éviter la congestion du réseau routier. Par contre, ces heuristiques s'opposent à l'objectif de la minimisation de la date de fin de l'évacuation. Les heuristiques de types “squeeze" ont pour but de diminuer l'attente des personnes, c'est-à-dire, évacuer les personnes à chaque fois qu'il n'y a pas congestion du réseau de transport. Ce qui permet de minimiser la date de début d'évacuation du dernier groupe de véhicules. Par conséquent, la minimisation de la date de fin de l'évacuation. Il est clair que la meilleure solution d'évacuation est une combinaison entre ces deux types d'heuristiques. En ce sens, quatre combinaisons de ces heuristiques ont été testées sur un

réseau de transport de taille moyenne avec un nombre de véhicules égal à 10000. Le temps d'exécution de ces heuristiques est de quelques secondes.

[Tuydes, 2005] a étudié le problème d'évacuation à grande échelle par zones. Chaque zone a une fenêtre de temps où l'évacuation sera sécurisée. Le modèle mathématique proposé est basé sur les modèles de cellules de transmission, et vise à minimiser la date de fin d'évacuation. Les variables de décision sont l'ordre d'évacuation des zones. [Tuydes, 2005] a testé ce modèle sur un réseau routier de petite taille et a proposé une méthode tabou pour les instances de grande taille.

Une étude récente du problème d'évacuation de la ville de Virginie (USA) après un ouragan a été faite par [Bish et al.,]. Ils ont proposé des modèles mathématiques basés sur les modèles de transmission cellulaire. L'idée est de définir des stratégies pour ordonner l'évacuation par véhicules des personnes, ensuite, définir le chemin à suivre pour les évacués afin de minimiser la date de fin d'évacuation. Contrairement aux travaux utilisant les modèles de transmission cellulaire, le modèle de [Bish et al.,] prend en considération la capacité des centres de secours et les exigences des évacués (choix du lieu d'évacuation). Pour une population inférieure à la ville de Virginie, ces modèles donnent des plans d'évacuation optimaux. [Bish et al.,] ont également proposé une heuristique qui a été testée sur des instances dont la grandeur est la suivante : 212626 voitures distribuées sur 78 points de départ, 757 routes et 3 centres de secours de capacité assez grande.

1.2.4 Systèmes d'aide à la décision

Le premier système d'aide à la décision est TEVACS. Il a été développé par [Han and Yuan, 2005]. Ce logiciel a été utilisé pour évaluer plusieurs scénarios d'évacuation. Pour un plan d'évacuation donné, le logiciel donne la date de fin d'évacuation. L'utilisateur peut voir dynamiquement sur un graphique le taux de congestion du réseau routier sur des intervalles de temps.

[Tufekci and Kisko, 1991] ont développé un outil d'aide à la décision régional en-ligne (REMS). Cet outil est capable de tester plusieurs scénarios d'évacuation après un ouragan, un déversement chimique ou un accident radiologique. Le logiciel a la capacité de visualiser le processus d'évacuation dans le temps et d'afficher les flux de circulation sur les voies du réseau de transport.

[Hobeika et al., 1994] et[Hobeika, 2002] ont développé un outil d'aide à la décision du réseau de transport (TEDSS). Il est utilisé dans les cas d'évacuation due à une catastrophe nucléaire (i.e. évacuation des personnes autour des centrales nucléaires). Ce support d'aide à la décision tient compte des conditions météorologiques. Le logiciel de simulation utilisé par TEDSS est MASSVAC qui est un logiciel de simulation en temps réel. Les sorties de ce système sont : la date de fin de l'évacuation, l'affectation des routes aux évacués et la congestion éventuelle du réseau de transport. Ce système offre la possibilité aux décideurs de choisir les règles d'affectation intégrées pour obtenir la meilleure stratégie d'évacuation. TEDSS a été mis à jour en 2002 après la mise à jour de MASSVAC en 1998.

[Pidd et al., 1996] ont développé un logiciel qui est une combinaison entre un simulateur et un système d'information géographique, nommé (CEMPS). CEMPS permet aux utilisateurs de spécifier les incidents, les choix de route et les conditions météorologiques.

Le logiciel de simulation utilisé pour CEMPS ne permet que l'affectation statique du trafic.

[Franzese and Han, 2001] ont développé un système d'aide à la décision dans le cas d'évacuation des personnes après un ouragan. L'évacuation est faite en deux phases. La première consiste à fixer le périmètre de la zone impactée puis à estimer la population de cette zone et décider du nombre de personnes qui seront effectivement évacuées. Pour cela, une analyse est faite sur les comportements des personnes. La deuxième phase est l'évaluation de l'efficacité d'évacuation par le simulateur de ce système qui utilise les sorties de la première phase.

Un autre système en temps réel, STEMS ([Hamza-Lup et al., 2005]), permet après une détection d'un incident, une génération de plans d'évacuation et de contrôle du trafic. En premier lieu, le décideur définit la zone impactée et le système considère tous les points qui sont en dehors de cette zone comme des points de sortie. Deux approches pour gérer l'évacuation des personnes ont été développées. La première consiste à utiliser toutes les routes de réseau routier pour satisfaire toutes les demandes d'évacuation. La deuxième utilise un algorithme de calcul de plus courts chemins pour évacuer les personnes.

Le schéma de la figure 1.9 récapitule les différents modèles d'évacuation ainsi que leurs méthodes de résolution.

FIGURE 1.9 – Méthodes de résolution des problèmes d'évacuation

2 Présentation du problème et contexte

2.1 Présentation du contexte

Cette thèse s'inscrit dans le cadre du projet franco-allemand nommé : DSS_Evac_Logistique, qui porte sur l'évacuation d'une ville de taille moyenne après une catastrophe majeure. Nous nous plaçons dans le cas de gestion de crise, c'est-à-dire : évacuer les personnes depuis la zone sinistrée vers des endroits plus sûrs. Pour la partie française, nous prenons comme cadre applicatif la ville de Nice, dont les scénarios de catastrophes sont : un séisme pouvant entrainer un tsunami ou des glissement de terrain dans les montagnes de l'arrière-pays Niçois. Du côte allemand, le site retenu est la ville de Kaiserslautern confrontée aux risques de crash d'avions militaires et incendies, ou de découverte d'une bombe.

Le projet DSS_Evac logistique a pour objectif final de fournir un logiciel d'aide à la décision permettant de choisir un plan d'évacuation dans une base de données de type Optima de Pareto. Un tel plan correspond à un bon compromis entre les différents critères d'évaluation d'un plan d'évacuation. Il est important de mettre l'accent sur son aspect Aide à la Décision. D'une part, il fournit des paramètres d'évaluation (e.g. coût, temps, effets, ...) des décisions permettant d'aider les décideurs à choisir un plan d'évacuation. D'autre part, cet outil va intégrer des algorithmes d'optimisation combinatoire et des moteurs de simulation dédiés. Ce projet est bien plus que le développement d'un logiciel, il s'agit d'un travail de recherche novateur, scientifiquement difficile, et nécessitant des compétences multiples, dont l'objectif sera de permettre, en cas de crise, l'évacuation des villes touchées.

Dans ce projet nous analysons des situations de crise afin d'apporter des réponses aux tâches suivantes :

- Modéliser les scénarios multirisques et l'endommagement de l'environnement. Par exemple, l'impact de la catastrophe sur le réseaux de transport ;
- Proposer des modèles et algorithmes d'optimisation pour la détermination des plans d'évacuation ;
- Établir des plans d'évacuation qui permettent d'appréhender le comportement de chaque individu à évacuer. Notamment, dans les situations de fort stress.

Une telle analyse des situations d'urgence et de scénarios d'évacuation associés requiert l'intervention de plusieurs domaines d'expertise : aménagement du territoire et sociologie, modélisation des catastrophes et de leurs impacts sur les infrastructures, sécurité, localisation, transport, calcul d'itinéraires sous contraintes, simulation, etc. En conséquence, un des aspects de ce projet est de construire un logiciel d'aide à la décision qui exploite les compétences multidisciplinaires des chercheurs et industriels qui y participent et qui sont représentatifs des différents domaines d'expertises précédents. Dans ce projet, plusieurs partenaires ont collaboré pour apporter des solutions partielles ou totales sur les problèmes posés ci-dessus. Dans les deux sections suivantes nous présentons les différents partenaires de ce projet ainsi que les tâches affectées à chacun d'eux.

2.1.1 Les partenaires du projet

Dans ce paragraphe, nous décrivons brièvement les partenaires des deux consortiums français et allemand.

Consortium français

Les différents partenaires du consortium français sont :

1. L'équipe "Ordonnancement et Conduite" (OC) du Laboratoire d'Informatique (LI EA 6300, ERL OC CNRS 6305) de l'université de Tours.
2. Le laboratoire d'aménagement de territoire CITERES (UMR CNRS 6173) de l'Université François Rabelais de Tours.
3. Le BRGM, service Risques Naturels et Sécurité du Stockage de CO2 (RNSC).
4. La société CERVVAL basée à Brest.

Consortium allemand

Les différents partenaires de ce consortium sont :

1. L'équipe "Optimization Research Group " du Département de mathématique de l'université de Kaiserslautern.
2. Les autorités responsables de la sécurité de Kaiserslautern.
3. La société INFORMS basée à Aachen.

2.1.2 Tâches du projet

Le projet DSS_Evac logistique est constitué de huit grandes tâches dont la quatrième tâche est l'objet de cette thèse. Nous présentons ci-dessous ces tâches en précisant qui en assure la coordination. Ces tâches sont représentées dans la figure (1.10).

Tâche 1 : Definition of scenarios and outcomes

Pour la partie française, cette tâche est confiée au service RNSC du BRGM et au laboratoire CITERES. Pour la partie allemande, cette tâche est pilotée par les autorités responsables de la sécurité de Kaiserslautern. Elle porte sur une étude des scénarios de catastrophe plausibles et une évaluation du sur-endommagement de l'environnement causé par ces catastrophes. Cela conduira à deux types de résultats : le premier est une modélisation de la catastrophe, le second est une simulation des risques et des impacts sur la zone endommagée. Cette étude sera le point de départ des travaux de recherches menant à la conception de l'outil d'aide à la décision.

Tâche 2 : Evac-Location

Cette tâche est confiée à l'équipe "Optimization Research Group ". Son but est de déterminer les centres de secours adéquats pour l'accueil des personnes (analyse de la localisation). Cette étude consiste dans un premier lieu à déterminer les endroits potentiels pour les centres de secours tout en tenant compte du coût, confort, espace, etc. Ensuite, les centres réellement opérationnels seront choisis parmi les centres de la première étape.

Tâche 3 : Evac-EvalTraffic

Cette tâche a pour objectif de déterminer les paramètres du réseau routier en fonction de la catastrophe : les capacités maximales de chaque route, le temps de parcours de chaque route, la suppression de certaines routes, la réorientation des voies à double sens en voies à sens unique, etc. Cette tâche a été confiée à l'équipe “Optimization Research Group ".

Tâche 4 : Evac-Scheduling

Cette tâche a été placée sous la responsabilité de l'équipe OC. Sa réalisation tient compte des résultats des trois tâches précédentes : les différents scénarios de la catastrophe et leurs impacts sur les infrastructures, la localisation des centres de secours et la reconfiguration du réseau de transport. Pour réaliser cette tâche nous devons ordonnancer dans le temps les différentes opérations d'évacuation. Sachant que l'évacuation simultanée de toute une population est irréalisable, nous devons décider pour chaque personne un centre de secours de rattachement ainsi que son heure d'évacuation selon les critères : urgence, sécurité des personnes, confort, etc.

Tâche 5 : Evac-SolveTraffic

Cette tâche est réalisée par l'équipe OC et vise à proposer des algorithmes d'évacuation multicritères pour l'évacuation des piétons et des individus disposant de moyens de transport personnels, depuis leurs habitations vers les points de rassemblement. Les deux critères pris en compte sont la durée d'évacuation qui doit être minimiser et la dangerosité du plan d'évacuation. Les sorties de ces algorithmes, qui sont les répartitions des personnes sur les points de rassemblement, sont utilisées comme des entrées de la tâche 4.

Tâche 6 : Evac-MCDM

Comme les problèmes abordés dans ce projet sont multicritères, le but de cette tâche est double : proposer des algorithmes permettant de d' échantillonner les ensembles des optima de Pareto pour ces problèmes, ensuite, développer un outil de représentation et de comparaison entre les solutions de front de Pareto. Cette tâche est pilotée par l'équipe “Optimization Research Group ".

Tâche 7 : Evac-Simulate

Cette tâche est effectuée par CERVVAL, qui est le partenaire industriel du projet, et spécialiste dans le domaine des logiciels de simulation. Le but de cette tâche est la conception d'un logiciel de simulation pour simuler et évaluer les plans d'évacuations.

Tâche 8 : Evac-Tool

Cette tâche est effectuée en collaboration entre les deux sociétés CERVVAL et INFORMS. La finalité de cette tâche est la conception d'un logiciel final d'aide à la décision permettant la gestion de la crise.

Les travaux réalisés dans les tâches précédentes se feront dans le but de fournir non seulement des résultats pour les villes de test de Nice et Kaiserslautern mais également

des algorithmes d'optimisation combinatoire et d'aide à la décision multicritère utilisables quelle que soit la ville de taille moyenne considérée.

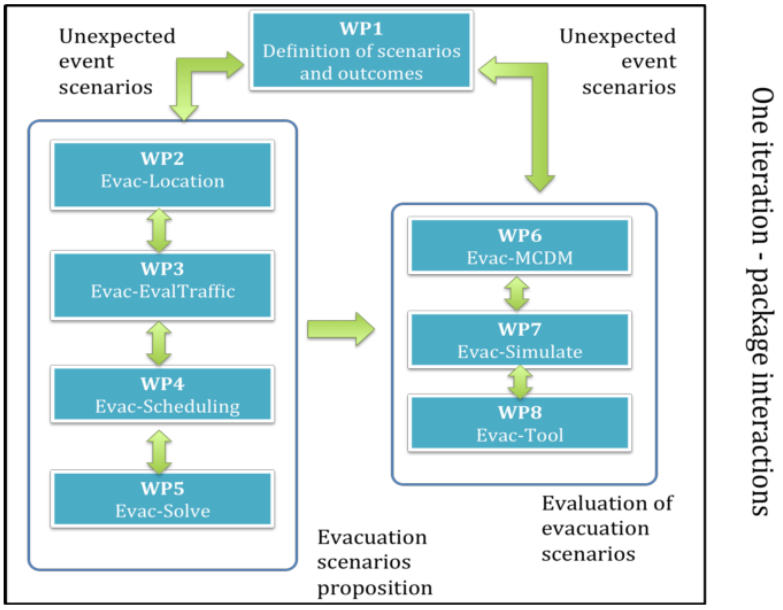


FIGURE 1.10 – Les différentes tâches du projet ([Neron et al., 2011])

2.2 Présentation du problème

Le problème d'évacuation global abordé dans cette thèse consiste à définir un ou plusieurs plans d'évacuation macroscopiques de personnes après une catastrophe. Plus précisément, nous considérons un ensemble fini de points de rassemblement dans la zone sinistrée où des personnes sont regroupées pour être évacuées, et un ensemble de centres de secours localisés en dehors de la zone sinistrée pour accueillir les évacués. Nous disposons d'un certain nombre de bus pour évacuer une partie de la population exposée au danger. Un plan d'évacuation macroscopique est caractérisé par : (1) la détermination, en fonction du réseau routier disponible, des centres de secours de rattachement pour chaque évacué ou groupe d'évacués, (2) la planification temporelle des évacués vers ces centres de secours. Ceci dans le but de minimiser un ou plusieurs critères qui seront détaillés par la suite.

Nous avons subdivisé la résolution de notre problème global en deux sous problèmes principaux. Le premier sous problème est un problème d'évacuation par bus avec approximation du réseau routier où nous nous intéressons uniquement à l'évacuation des personnes par bus. Ce premier problème est donc une simplification du problème général, et dont la résolution doit nous permettre d'obtenir une première estimation de l'évacuation. Le deuxième sous problème généralise le premier en considérant que l'évacuation s'effectue par voitures et par bus, tout en prenant en compte le routage des véhicules. Il s'agit donc là du problème originale à résoudre.

Dans les deux sections suivantes nous présentons en détail chaque sous problème et

nous positionnons notre travail par rapport aux problèmes d'évacuation présentés dans l'état de l'art (cf. section 1).

2.2.1 Problème d'évacuation par bus avec approximation du réseau routier

Dans la première partie de cette thèse, nous considérons, comme dans [Bish, 2011], le problème d'évacuation par bus. Notre modélisation diffère des travaux de l'état de l'art (cf. section 1) qui sont basés sur l'un des problèmes ou modèles suivants : les problèmes de flots ([Fahy, 1994], [Yamada, 1996],[Hamacher and Tjandra, 2001],...), les problèmes d'affectation dynamique du trafic ([Hobeika and Kim, 1998], [Sbayti and Mahmassani, 2006], [Chiu et al., 2007],...) ou les modèles de simulation ([Sheffi et al., 1982a],[Peeta and Mahmassani, 1995a], [Peeta and Mahmassani, 1995b],....). Elle est basée sur l'optimisation combinatoire où les méthodes de résolution proposées reposent fortement sur la programmation mathématique. Contrairement à [Bish, 2011], la durée d'un trajet varie au cours du temps et dépend de la date de début d'évacuation et de la situation du réseau routier (saturation, réplique de séisme,....). Les fonctions objectifs sont : la date de fin d'évacuation et la minimisation du risque des personnes. De plus, nous étudions le cas monocritère et le cas multicritère. L'étude de la littérature montre que le modèle définissant les durées du trajet de notre problème d'évacuation n'a pas été étudié. Considérer que la durée d'évacuation dépend de la date de début et de l'état du réseau routier conduit donc à des problèmes d'ordonnancement d'opérations d'évacuation intéressants de ce point de vue là. Nous verrons également, dans le chapitre suivant, que l'hypothèse de durées opératoires dépendantes du temps peut être abandonnée au profit de la robustesse.

Le but de cette partie de thèse est de déterminer des plans d'évacuation macroscopiques c'est-à-dire : définir pour chaque groupe de personnes (bus), la date de début d'évacuation et le centre de secours de rattachement. Comme dans [Bish, 2011], nous ferons la restriction suivante : il existe un arc entre chaque point de rassemblement et chaque centre de secours. La durée de passage par arc est estimée par des calculs de chemins prenant en compte les phénomènes impactant le réseau routier (réplicat de séisme, réparation, congestion).

2.2.2 Problèmes d'évacuation par bus et par voiture avec réseau routier réel

Le problème présenté dans la section 2.2.1 est restrictif : le réseau routier est approximatif et l'évacuation se fait uniquement par bus. Nous y supposons également que tous les évacués sont disponibles aux points de rassemblement dès le début de l'évacuation. Dans la deuxième partie de cette thèse, nous considérons un problème d'évacuation général. Le calcul des plans d'évacuation se fait sur un réseau routier réel, ce qui rend le problème significativement plus difficile à traiter. Sur chaque arc de ce réseau routier, nous avons les données suivantes : la durée de passage par cet arc, la capacité maximale de cet arc, et une valeur de risque associée au risque d'effondrement des bâtiments sur cet arc.

Nous considérons deux types de flots : les flots de bus pour les personnes arrivant à pied aux points de rassemblement et les flots de voitures pour les personnes utilisant leurs propres voitures pour quitter la zone endommagée. Il est supposé que les conducteurs de voitures partent aux points de rassemblements afin de récupérer un plan d'évacuation. De plus, les décideurs ont la prérogative de choisir le nombre de centres à ouvrir. L'objectif

de ce problème est de choisir les centres de secours à ouvrir afin de minimiser la date de fin d'évacuation et la somme des risques encourus durant l'opération d'évacuation.

Dans la littérature, nous distinguons deux types de problèmes d'évacuation traitant simultanément la localisation et le routage. Dans le premier type, nous citons les travaux de [Sherali et al., 1991], [Coutinho-Rodrigues et al., 2012], [Bish, 2011], [Goerigk et al., 2013], qui supposent que les autorités ont le pouvoir total de choisir les centres de secours ainsi que les chemins à emprunter par les évacués pour atteindre un centre de secours. Cette hypothèse sera supposée tout au long de cette thèse. Dans le deuxième type de travaux ([Kongsomsaksakul et al., 2005]), les autorités ont uniquement la capacité de choisir les localisations des centres. Les centres de secours et les chemins vers ces centres sont choisis par les évacués. Certes cette deuxième approche peut paraitre réaliste mais n'est pas applicable dans le cas d'un désastre où les évacués n'ont pas forcément une connaissance totale de l'état du réseau routier. Dans ces deux types de problème, les flots d'évacués sont soit les piétons, les bus, ou les voitures.

Une étude qui gère l'évacuation à commodités multiples (les bus et les voitures) a été proposée par [Bretschneider, 2012]. [Bretschneider, 2012] aborde uniquement le problème de routage des voitures et des bus. De plus, la capacité de chaque centre est considérée suffisamment grande pour accueillir toute la population à évacuer. Le but de son étude est de minimiser une combinaison linéaire de deux critères. Le premier critère est le nombre de lignes d'urgence utilisées par les bus durant l'évacuation. Tandis que le deuxième critère est le nombre de flots arrivés à chaque centre de secours.

Le problème abordé dans cette thèse est inédit puisqu'il tient compte de la localisation des centres de secours et le routage de deux types de flots (voitures et bus). En plus, nous minimisons deux critères en énumérant les solutions du front de Pareto. Ce problème sera présenté en détail dans le chapitre 3.

3 Conclusion du chapitre

Les problèmes d'ordonnancement d'opérations d'évacuation ont été modélisés en deux catégories. La première catégorie est basée sur des problèmes d'optimisation et la deuxième sur des modèles de simulation.

La première catégorie est appliquée pour déterminer un plan d'évacuation optimal en utilisant la programmation mathématique. En général, les fonctions objectifs utilisées sont la minimisation de la durée totale d'évacuation et la maximisation du nombre de personnes à évacuer. D'autres part, les modèles d'optimisation sont statiques ou dynamiques.

Les modèles statiques d'affectation du trafic sont basés sur le principe d'*équilibre d'usager* ou sur le principe d'*optimum social* ([Wardrop, 1952]). Dans ce cas, le réseau routier est constant durant le temps, en termes de durée de traversée des arcs et de demandes des utilisateurs de réseaux (les demandes sont connues en avance et ne changent pas au fil du temps). Ces caractéristiques permettent aux modèles statiques de manipuler des réseaux routiers de grandes tailles. Par contre, l'utilisation de ces modèles dans un contexte d'évacuation n'est pas claire : dépendance du temps et de l'incertitude des données. Par exemple, la durée de traversée des arcs peut croître à cause de la congestion des routes.

Contrairement aux modèles statiques, les modèles dynamiques sont les plus utilisés dans le contexte d'évacuation. Ils sont capables de représenter les flots de trafic, la congestion de réseau routier, l'aspect temporel et la gestion des données dépendantes du temps. Cependant, les modèles dynamiques présentent quelques inconvénients : le temps de calcul pour définir des plans d'évacuation est assez grand, car ils manipulent énormément de données : l'optimalité des solutions de problèmes de grandes tailles n'est donc pas assurée. Dans les modèles dynamiques, nous distinguons les modèles basés sur des problèmes de flots dynamiques et ceux basés sur des problèmes d'affectation dynamique du trafic. Les problèmes de flots sont utilisés pour définir des plans d'évacuation macroscopiques. Tandis que les problèmes d'affectation dynamique sont utilisés pour déterminer des plans d'évacuation microscopiques et dans certains cas des plans d'évacuation macroscopiques.

Les modèles de simulation sont développés pour analyser et évaluer les plans d'évacuation. Il existe trois types de modèles de simulation : macroscopique, meososcopique et microscopique. Beaucoup d'entre eux sont utilisés dans le cas des catastrophes spécifiques (des ouragans ou des catastrophes nucléaires) : NETVACI, OREMS, DYNEV, NETSIM, etc.

Dans tous ces modèles, il est très important de noter que l'aspect crucial des comportements des évacués n'est pas pris en compte. De plus, la plupart de ces modèles traitent de l'évacuation des piétons ou l'évacuation par voiture. [Bish, 2011] est le premier qui a étudié le problème d'évacuation par bus. Enfin, dans ces modèles, l'aspect incertain n'a pas reçu beaucoup d'attention bien qu'il existe des données incertaines dans les problèmes d'évacuation. Par exemple : le nombre d'évacués, la durée de passage de chaque arc, la capacité des routes, etc.

Nous apportons dans cette thèse des éléments de réponse à ces nouvelles problématique.

Chapitre 2

Évacuation par bus avec durées de transport

Dans le premier chapitre de cette thèse, nous avons présenté un état de l'art sur les problèmes d'évacuation et le contexte général de cette thèse. De plus, nous avons présenté brièvement les deux principaux problèmes abordés dans cette thèse. Dans ce chapitre, nous nous intéressons à l'étude du premier problème : le problème d'évacuation par bus avec durées de transport. La section 2 de ce chapitre, introduit la modélisation formelle de ce problème. La section 3 présente succinctement un état de l'art sur les problèmes d'ordonnancement à durées dépendantes du temps. Dans la section 4, nous donnons deux modélisations du problème monocritère d'évacuation considéré où les durées de transport sont dépendantes du temps. Le but est de minimiser la date de fin d'évacuation. Ensuite, nous présentons les différentes méthodes exactes et heuristiques proposées pour la résolution de ce problème. Puisque les données de ce problème d'évacuation sont incertaines, la section 5 est consacrée à la présentation d'une version robuste à certain aléas. Enfin, dans la section 6, nous présentons le problème bicritère d'évacuation par bus dont le but est de minimiser la date de fin d'évacuation et ainsi que le risque encouru durant l'évacuation.

1 Contribution du chapitre

Dans ce chapitre, nous avons étudié trois variantes du problème d'évacuation par bus introduit par [Bish, 2011]. La première partie de ce chapitre est consacrée à l'étude du problème d'évacuation par bus monocritère. Nous avons proposé des méthodes exactes basées sur la programmation mathématique et des méthodes heuristiques pour la résolution de ce problème. Ces travaux ont fait l'objet à plusieurs communications dans des conférences nationales [Deghdak et al., 2013b, Deghdak et al., 2014b] et internationales [Deghdak et al., 2013a, Deghdak et al., 2014a, Deghdak et al., 2014c] ainsi à une publication dans une revue internationale [Deghdak et al., 2015b].

Ensuite dans la seconde partie, nous avons traité une version robuste du problème d'évacuation par bus monocritère. Pour résoudre ce problème, nous avons proposé des méthodes exactes et des méthodes heuristiques. Ce travail a fait l'objet d'une publication dans une revue internationale [Goerigk et al., 2015].

Enfin dans la troisième partie, nous avons étudié une version bicritère du problème d'évacuation par bus. Une méthode par séparation et évaluation et un modèle mathématique ont été proposé pour la résolution exacte de ce problème. Pour la résolution approchée, nous avons proposé des heuristiques de liste et une méthode de recherche par voisinage. Ce travail a donné l'objet à une communication scientifique dans une conférence internationale [Deghdak et al., 2015a].

2 Présentation du problème et positionnement scientifiques des problèmes abordés

Soit le graphe biparti complet $G = (\mathcal{V}, \mathcal{A})$, où $\mathcal{V}$ et $\mathcal{A}$ représentent respectivement l'ensemble des nœuds et l'ensemble des arcs. $\mathcal{V}$ est composé de deux sous-ensembles de nœuds notés $\mathcal{P} = \{1, ..., P\}$ et $\mathcal{S} = \{1, ..., S\}$. $\mathcal{P}$ est l'ensemble des points de rassemblement où les évacués sont regroupés, et $\mathcal{S}$ est l'ensemble des centres de secours. Un arc $(i, j) \in \mathcal{A}$ existe si est seulement si les évacués peuvent être transportés depuis un point de rassemblement i vers un centre j. Un point de rassemblement i a une demande e_i, $i \in \mathcal{P}$, et un centre de secours j a une capacité cap_j, $j \in \mathcal{S}$, exprimées en nombre de bus qui peuvent accueillir les évacués. Chaque arc (i, j) a une longueur p_{ijt} qui représente la durée de trajet entre chaque couple (P_i, S_j) à la date t. Une valeur de risque r_{ijt} est associée à un arc (i, j) qui représente le risque d'effondrement des bâtiments sur cet arc à l'instant t. La durée et le risque d'un trajet sont dépendants du temps ceux-ci est une conséquence de l'évolution du réseau routier. En effet, le réseau routier évolue dans le temps en présence des répliques de séisme, la congestion des routes, déblayage des routes, etc. Dans la suite, nous supposons que k événements peuvent se produire à des dates connues d_l modifiant ainsi les valeurs des durées de transport et les valeurs de risque entre un point de rassemblement i vers un centre de secours j. Ces dates sont supposées établies par des estimations et simulations en amont. Les fonctions définissant les durées et les valeurs du risque sont

respectivement définies par :

$$p_{ijt} = p_0 + \begin{cases} a_{ij}^0 & \text{if } t \in]0, d_1] \\ a_{ij}^1 & \text{if } t \in]d_1, d_2] \\ \vdots & \\ a_{ij}^{k-1} & \text{if } t \in]d_{k-1}, d_k] \end{cases} \tag{2.1}$$

$$r_{ijt} = \begin{cases} b_{ij}^0 & \text{if } t \in]0, d_1] \\ b_{ij}^1 & \text{if } t \in]d_1, d_2] \\ \vdots & \\ b_{ij}^{k-1} & \text{if } t \in]d_{k-1}, d_k] \end{cases} \tag{2.2}$$

où a_{ij}^x et b_{ij}^x sont, respectivement, la durée de trajet et la valeur de risque, dans le cas où les évacués sont transportés à partir d'un point de rassemblement i vers un centre de secours j en une période de temps x. La date de début de chaque évacué est $t \in]d_{x-1},\ d_x]$. Les intervalles $[d_{l-1},\ d_l]$ sont déterminés par une étude (prévision) préliminaire de l'évolution du réseau de transport. La constante p_0 est le temps moyen de retour des bus à vide vers le centre de la zone sinistrée. Elle permet d'estimer le temps de retour à vide des bus. Le rôle de la fonction p_{ijt} est d'approximer le routage des bus vers les points de rassemblement.

Nous disposons d'un ensemble de bus $\mathcal{B} = \{1, ..., B\}$ de même capacité pour évacuer les personnes. Les deux critères considérés sont la minimisation de la date de fin d'évacuation, notée T_{evac} et la minimisation de la somme des risques encourus durant l'évacuation, notée R_{evac}.

Dans la suite, nous notons respectivement par MBEP, RBEP et BBEP le problème monocritère d'évacuation par bus, le problème robuste d'évacuation par bus et le problème bicritère d'évacuation par bus.

La figure 2.1 représente un exemple de la structure d'un réseau de BBEP à un instant de temps t avec trois points de rassemblement et deux centres de secours. Le nombre de bus nécessaires pour l'évacuation des personnes depuis les points de rassemblement vers les centres de secours est de 2, 4, et 5. Les capacités des deux centres de secours sont 6 et 9 bus d'évacués. Les deux valeurs associées à chaque arc sont respectivement la durée du trajet et le risque.

3 État de l'art sur les problèmes d'ordonnancement dépendant du temps

L'état de l'art présenté au chapitre 1 ne met pas en évidence des liens entre problèmes d'évacuation et problèmes d'ordonnancement. Pourtant, sous l'hypothèse d'approximation du réseau de transport par des durées de transport, de tels liens existent. Ils permettent alors d'aller puiser dans la littérature en ordonnancement des résultats et algorithmes potentiellement intéressants. Notamment, suite à une catastrophe, le réseau routier utilisé pour l'évacuation évolue : il peut être amélioré par répartition ou nettoyage des routes, mais aussi dégradé, notamment lorsqu'il y a des répliques après un séisme. Cela implique

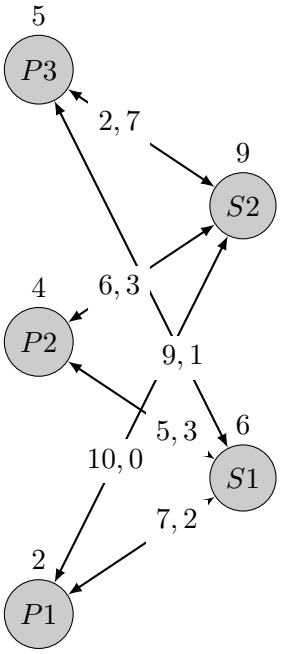


FIGURE 2.1 – Exemple de réseau BBEP à un instant de temps t.

que les durées des opérations d'évacuation évoluent au cours de temps. Il est donc pertinent d'étudier l'état d'avancement des travaux dans la littérature traitant des problèmes d'ordonnancement avec durées opératoires dépendantes du temps. Nous nous sommes limités aux problèmes à une machine et aux problèmes à machines parallèles qui sont les plus proches des problèmes d'évacuation.

Les problèmes d'ordonnancement dont les durées opératoires des tâches dépendent de leurs dates de début ont été largement étudiés ces dernières années. Cette motivation est justifiée par le fait que ces problèmes reflètent des situations réelles de la vie courante. Par exemple, ces problèmes trouvent leurs applications dans différents domaines : économique (allocation de ressource), financier (finance de management), etc. Pour plus de détails sur les applications de ces problèmes, le lecteur peut se référer aux travaux de [Mosheiov, 1994], [Mosheiov, 1996] et [Kunnathur and Gupta, 1990]. La plupart des travaux abordés dans la littérature portent sur les problèmes d'ordonnancement à une machine. Les critères couramment considérés dans le cas d'une machine ou des machines parallèles sont :

- la date de fin de l'ordonnancement, C_{max}
- la somme des dates de fin, $\sum C_j$
- la somme pondérée des dates de fin, $\sum w_j C_j$
- le maximum des retards absolus, T_{max}
- le nombre de travaux en retard, $\sum U_j$

Les durées opératoires des tâches sont représentées soit par des fonctions linéaires, soit par des fonctions non linéaires incluant *la détérioration* (la durée opératoire croît au cours du temps) ou *la réduction* (la durée opératoire décroît au cours du temps).

De façon plus formelle ces problèmes d'ordonnancement sont définis comme suit : nous

avons un ensemble de n travaux $TS = (\{T_j\}, \{a_j\}, \{b_j\}, \{r_j\}, \{d_j\})$. À chaque travail T_j, on associe une durée opératoire basique a_j, un taux de détérioration b_j, une date de début souhaitée r_j et une date de fin impérative d_j. La durée opératoire de T_j dépend de sa date de début s_j et est donnée par $p_j = a_j \pm b_j s_j$. Cette formule des durées opératoires est la plus étudiée ; nous citons par exemple les travaux de [Tanaev et al., 1994], [Mosheiov, 1994], [Cheng and Sun, 2005], [Gawiejnowicz, 2007] et [Ji et al., 2006].

Contrairement au cas d'une machine, les travaux abordant le cas de machines parallèles sont peu nombreux. La plupart de ces problèmes sont $\mathcal{NP}-$difficiles. En général, la fonction des durées opératoires est linéaire ([Jeng and Lin, 2006], [Gawiejnowicz et al., 2006][Cheng et al., 2007], etc.). Pour plus de détails, le lecteur peut se référer à l'état de l'art de [Deghdak et al., 2014d].

Comme nous l'avons vu dans la section 2, nous avons défini la fonction des durées comme une fonction non-linéaire par palier. Une telle classe de fonctions n'a pas été traitée dans la littérature.

4 Le problème de la minimisation de la durée d'évacuation

Maintenant, nous allons considérer le problème monocritère d'évacuation par bus (MBEP). Le but est de minimiser la date de fin d'évacuation (T_{evac}) sachant que les durées des trajets dépendent de leurs dates de début.

La figure 2.2 illustre un exemple d'ordonnancement d'une instance de MBEP. Nous avons deux centres de secours, sept opérations d'évacuation et deux bus.

4.1 Formulations mathématiques

Dans cette section, nous présentons deux formulations mathématiques indexées sur le temps du problème d'évacuation par bus (MBEP). Ceci est motivé d'une part, par le fait que cette formulation donne les meilleurs résultats pour le problème d'affectation des quais de chargement et de déchargement des camions et qui est proche de notre problème ([Berghman and Spieksma, 2015]). L'inconvénient de cette formulation est le nombre de variables qui est pseudo-polynomiale.

4.1.1 Première formulation

Dans la première modélisation, un ensemble d'opérations d'évacuation est associé à un point de rassemblement. Le nombre d'opération est égal au nombre de personnes regroupées dans ce point de rassemblement. L'unité de mesure est la capacité d'un bus. Pour formuler ce problème, nous utilisons les notations suivantes.

Données

- $\mathcal{S}$: l'ensemble des centres de secours
- $\mathcal{P}$: l'ensemble des points de rassemblement
- cap_j : la capacité d'un centre de secours

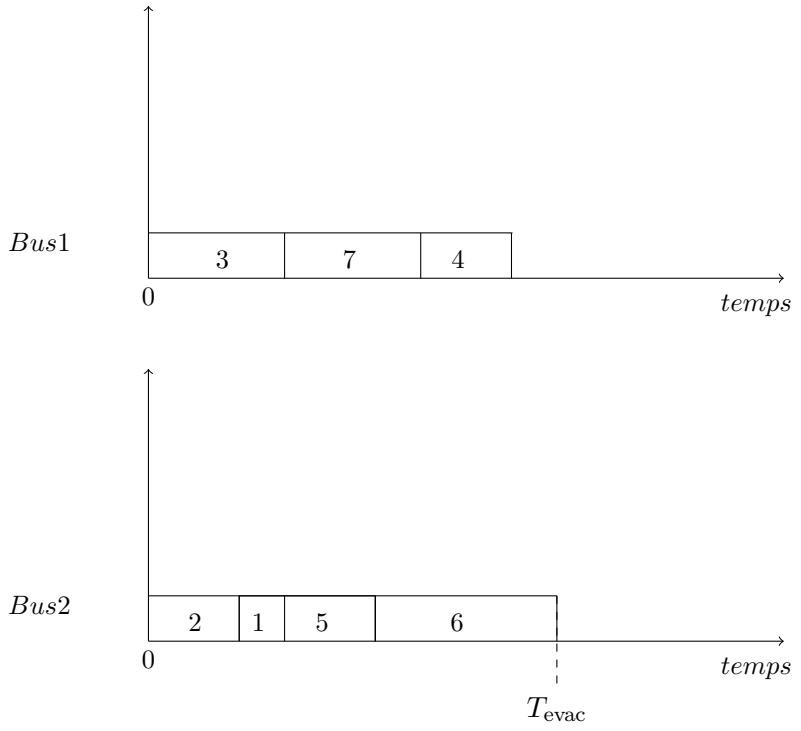


FIGURE 2.2 – Exemple d'ordonnancement d'une instance de MBEP.

- $\mathcal{N}$: l'ensemble des opérations d'évacuation
- T : l'horizon temporel
- p_{ijt} : la durée de transport d'une opération d'évacuation, associée a un point de rassemblement i, vers un centre de secours j pour un bus partant à l'instant t
- $\mathcal{B}$: l'ensemble des bus disponibles pour l'évacuation

Variables

- $\forall i \in \mathcal{N}, \forall j \in \mathcal{S}, \forall t \in \mathcal{T},$

$$x_{ijt} = \begin{cases} 1 & \text{si un bus débute l'opération d'évacuation } i \\ & \text{vers } j \text{ à } [t, t+1[, \\ 0 & \text{sinon.} \end{cases}$$

- T_{evac} : la date de fin d'évacuation.

Objectif

$$\min T_{\text{evac}} \tag{2.3}$$

Contraintes

$$T_{\text{evac}} \geq (t + p_{ijt})x_{ijt} \qquad \forall i \in \mathcal{N}, \forall j \in \mathcal{S}, \forall t \in \mathcal{T} \tag{2.4}$$

$$\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{N}} x_{ijt} \leq cap_j \qquad \forall j \in \mathcal{S} \tag{2.5}$$

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{S}} \sum_{t' \in [0,t] \ p_{i,j,t'} + t' > t} x_{i,j,t'} \leq B \qquad \forall t \in \mathcal{T} \tag{2.6}$$

$$\sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{S}} x_{ijt} = 1 \qquad \forall i \in \mathcal{N} \tag{2.7}$$

$$x_{ijt} \in \{0, 1\} \tag{2.8}$$

La fonction objectif (2.3) minimise la date de fin d'évacuation. Les contraintes (2.4) définissent la valeur du critère T_{evac}. Les contraintes (2.5) sont les contraintes de capacité des centres de secours. Les contraintes (2.6) stipulent qu'à chaque instant t, B bus au plus seront utilisés. Les contraintes (2.7) assurent que chaque opération d'évacuation est transporté une seule fois à un centre de secours. Les contraintes (2.8) sont les contraintes d'intégrité sur les variables x_{ijt}.

4.1.2 Deuxième formulation

Cette modélisation vise à réduire le nombre de variables. Contrairement à la première, nous raisonnons par point de rassemblement et les opérations d'un même point de rassemblement peuvent être transportées sur des centres de secours différents.

Données

- $\mathcal{S}$: l'ensemble des centres de secours
- cap_j : la capacité d'un centre de secours j exprimée par nombre de bus
- $\mathcal{P}$: l'ensemble des points de rassemblement
- T : l'horizon temporel
- e_i : le nombre d'opérations d'un point de rassemblement i exprimé en nombre de bus
- p_{ijt} : la durée d'évacuation depuis un point de rassemblement i vers un centre de secours j à l'instant t
- $\mathcal{B}$: l'ensemble des bus disponibles pour l'évacuation

Variables

- $\forall i \in \mathcal{P}, \forall j \in \mathcal{S}, \forall t \in \mathcal{T}, \forall b \in \mathcal{B}$,

$$x_{ijtb} = \begin{cases} 1 & \text{si le bus } b \text{ commence l'évacuation de } i \text{ vers } j \text{ à } [t, t+1[. \\ 0 & \text{sinon.} \end{cases}$$

- T_{evac} : la date de fin d'évacuation.

Objectif

$$\min T_{\text{evac}} \tag{2.9}$$

Contraintes

$$T_{\text{evac}} \geq (t + p_{ijt})x_{ijtm} \qquad \forall i \in \mathcal{P}, \forall j \in \mathcal{S}, \forall t \in \mathcal{T}, \forall m \in \mathcal{B} \tag{2.10}$$

$$\sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{P}} \sum_{b \in \mathcal{B}} x_{ijtb} \leq cap_j \qquad \forall j \in \mathcal{S} \tag{2.11}$$

$$\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} \sum_{t' \in [0,t] \ p_{ijt'} + t' > t} x_{ijt'b} \leq 1 \qquad \forall t \in \mathcal{T}, \forall b \in \mathcal{B} \tag{2.12}$$

$$\sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{S}} \sum_{b \in \mathcal{B}} x_{ijtb} = e_i \qquad \forall i \in \mathcal{P} \tag{2.13}$$

$$x_{ijtb} \in \{0, 1\} \tag{2.14}$$

Les contraintes (2.10) définissent le critère à minimiser. Les contraintes (2.11) sont des contraintes de capacité des centres de secours. Les contraintes (2.12) signifient qu'à chaque instant de temps, un seul bus est utilisé par une opération d'évacuation. Les contraintes (2.13) assurent l'exécution de toutes les opérations d'évacuation d'un point de rassemblement. Les contraintes (2.14) sont des contraintes d'intégrité sur les variables de décision.

4.2 Prétraitement

La technique de prétraitement repose sur la relaxation continue d'un modèle mathématique MIP (Mixed integer programming) constitué de variables booléennes. Cette technique fonctionne de la façon suivante : nous relâchons les contraintes d'intégrité sur le domaine de ces variables booléennes, c'est-à-dire : ces variables peuvent avoir des valeurs non entières. Le modèle résultant est un modèle de programmation linéaire (LP). Ce modèle diffère de son équivalent (MIP) et permet de faire du prétraitement. L'objectif principal de cette technique est de réduire optimalement et en temps polynomiale la taille des instances à résoudre avec n'importe quel solveur mathématique. Pour notre cas, nous avons utilisé le solveur CPLEX.

Dans notre travail, nous avons appliqué le prétraitement sur le premier modèle présenté dans la section 4.1.1 : les variables booléennes x_{ijt} déterminent si un bus débute une opération d'évacuation i vers un centre de secours j à l'instant t. Le modèle LP construit est résolu par l'algorithme du simplexe. Soit z^*_{LP} sa solution optimale, B^* la base de variables associée et z^*_{MIP} la solution optimale du MIP. La technique du prétraitement va

permettre de déduire la valeur des variables booléennes en utilisant la relation suivante entre un MIP et son LP associé ([Savelsbergh, 1994]) :

$$z^*_{MIP} = z^*_{LP} + \sum_{x_{ijt} \notin B^*} c_{ijt} x_{ijt},$$

où c_{ijt} est le coût réduit de la variable x_{ijt}. Les coûts réduits sont des composantes (résultats) de l'algorithme du simplexe. Chaque coût réduit c_{ijt} correspond grossièrement à l'augmentation de la valeur de la solution optimale du LP lorsque la variable x_{ijt} est égale 1. Soit UB la borne supérieure calculée par une heuristique au problème à variables entières. De la relation précédente, nous déduisons la relation suivante :

$$UB \geq z^*_{LP} + \sum_{x_{ijt} \notin B^*} c_{ijt} x_{ijt} \Longleftrightarrow \sum_{x_{ijt} \notin B^*} c_{ijt} x_{ijt} \leq UB - z^*_{LP}$$

Cette dernière inégalité donne les règles de fixation suivantes :

- $\forall x_{ijt} \notin B^*$, si $c_{ijt} > UB - z^*_{LP}$ alors dans toute solution optimale du MIP, $x_{ijt} = 0$.
- en utilisant un raisonnement similaire sur les variables d'écart s_{ijt}, on peut déduire quand la variable $x_{ijt} = 0$.

Ces deux règles de fixation permettent de fixer les variables hors base. Pour la fixation des variables de base, nous utilisons les pseudo-coûts l_i et u_i, qui sont calculés dans notre cas avec les pénalités de Driebeek ([Driebeek, 1966]). Ces pénalités correspondent à une évaluation par défaut de l'augmentation de z^*_{LP} si x_{ijt} est mise à 0 ou 1 : elles dépendent des valeurs des coûts réduits des variables hors base. On a alors :

- $\forall x_{ijt} \in B^*$, si $(\ell_{ijt} x_{ijt}) \geq (UB - z^*_{LP})$ alors dans une solution optimale du MIP on a $x_{ijt} = 1$;
- $\forall x_{ijt} \in B^*$, si $(u_{ijt}(1 - x_{ijt})) \geq (UB - z^*_{LP})$ alors dans une solution optimale du MIP on a $x_{ijt} = 0$;

Cette technique permet donc de connaître en amont de la résolution du modèle, les valeurs de certaines variables dans une solution optimale du MIP.

Il est bien connu que la résolution de la relaxation continue d'un modèle est très rapide. Par conséquent, l'utilisation en amont du prétraitement est meilleure que la résolution directe du modèle. Dans le cas de notre modèle mathématique, cette technique peut être très intéressante. En effet, si toutes les valeurs des variables sont connues à l'avance, l'ordonnancement de tous les travaux est connu et donc la solution du problème. De plus, connaître les valeurs des variables permet aussi d'aider à la résolution du MIP pendant sa phase de "branch-and-cut". À priori, la taille de l'arbre d'exploration est réduite puisqu'il y a moins de variables à valeurs inconnues.

L'algorithme de prétraitement fonctionne comme indiqué dans l'Algorithme 1 :

Algorithme 1 Algorithme de prétraitement

1: Calculer une borne supérieure pour MBEP
2: Résoudre la relaxation continue (LP)
3: Fixer les variables x_{ijt}
4: Résoudre (MIP) avec les variables fixées

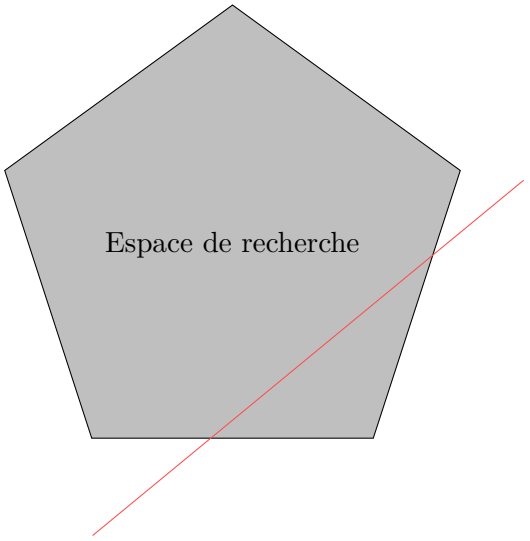


FIGURE 2.3 – schéma d'une coupe

4.3 Inégalités valides

En pratique l'efficacité du prétraitement peut dépendre de l'ajout d'informations liées au problème comme par exemple : une borne supérieure ou des inégalités valides. Avec l'ajout des inégalités valides sur le modèle relâché (LP), il est aussi possible de réduire l'espace des solutions tout en augmentant la valeur de z^*_{LP} utilisée pour fixer les variables.

Une inégalité valide permet de définir une relation entre les variables et donc d'interdire certaines valeurs de ces variables. Cela conduit à une réduction de l'espace de recherche (voir figure 2.3). Une inégalité valide est généralement spécifique au modèle dans lequel elle est ajoutée. Cependant l'ajout d'une inégalité valide peut ralentir la résolution du modèle puisque cela revient à l'alourdir par l'ajout de contraintes. Cela implique un choix efficace des inégalités valides à rajouter.

4.3.1 Élimination de la symétrie

Cette inégalité valide a pour but d'éliminer la symétrie entre les variables du modèle. En effet, soit P le nombre de points de rassemblements. Chaque point de rassemblement k possède N_k opérations d'évacuation, notées o^k_i, $\forall i = \{1, ..., N_k\}$. Les opérations d'évacuation d'un même point de rassemblement sont identiques (évacuation d'un groupe de personnes par bus). On peut imposer sans perte d'optimalité un ordre (contraintes de précédence) sur l'exécution des opérations d'un même point de rassemblement. C'est-à-dire : $\forall k = \{1, ..., L\}$, $o^1_k \rightarrow o^2_k \rightarrow \cdots \rightarrow o^{N_k}_k$; l'opération 2 du point de rassemblement k est transportée en même temps ou après la première opération du point de rassemblement k. Ainsi, cela permet d'éliminer $(N_k! - 1)$ séquences redondantes.

Dans notre modèle mathématique, cela peut être traduit par la contrainte suivante :

$$\forall k \in \mathcal{P}, \forall i \in \mathcal{N} :_k / N_k, \forall t \in \mathcal{T}, \quad \sum_{t'=0}^{t} \sum_{j=1}^{S} x_{o_k^i j t'} \geq \sum_{t'=0}^{t} \sum_{j=1}^{S} x_{o_k^{i+1} j t'} \tag{2.15}$$

4.3.2 Espacements minimaux

Considérons les contraintes d'élimination de la symétrie présentée dans la section 4.3.1. Ces contraintes imposent que l'ordre d'exécution des opérations d'un même point de rassemblement i est imposé : $o_i^1 \rightarrow o_i^2 \rightarrow ... \rightarrow o_i^l \rightarrow ... \rightarrow o_i^{N_i}$.

Soit B le nombre de bus. On sait donc qu'une opération o_i^l ne débutera que si celles qui la précèdent sont déjà débutées. De plus, à un instant de temps t au plus B opérations o_i^l d'un même point de rassemblement i peuvent être traitées. Soit l'ensemble $I_i = \{B + 1, 2B + 1, 3B + 1, ..., u_i B + 1\}$ avec $u_i \in \mathbb{N}$ tel que $u_i = \lfloor \frac{B-1}{N_i} \rfloor$.

Cela peut s'écrire de la façon suivante :

$$t_{o_i^l} \geq \min_{l-B \leq v \leq l-1} (t_{o_i^v} + \min_{1 \leq i \leq S} (p_{ijo_i^v})) \qquad \forall l \in I_i,$$

où $t_{o_i^l}$ est la date de début de l'opération o_i^l. Cette inégalité n'est pas directement implémentable dans le modèle mathématique puisqu'elle dépend directement de la date de début. Par contre, il est possible d'écrire une version relâchée, plus faible :

$$\sum_{j \in \mathcal{S}} x_{o_i^l j t'} \leq \sum_{t'=1}^{t-\delta_{it}^l} \sum_{j \in \mathcal{S}} x_{o_i^{l-B} j t'} \qquad \forall t \in \mathcal{T}, \forall i \in \mathcal{N}, \forall o_i^l \in I_i.$$

où

$$\delta_{it}^l = \{ \quad \min p_{o_i^v j t'}, \qquad \forall l - B \leq v \leq l - 1, \forall 1 \leq j \leq S, \forall 1 \leq t' \leq t\}.$$

L'algorithme ci-dessous présente l'Algorithme de prétraitement avec intégration des inégalités valides.

Algorithme 2 Algorithme de prétraitement en intégrant les inégalités valides

1: Calculer une borne supérieure pour MBEP
2: Résoudre la relaxation continue (LP) en ajoutant les inégalités valides
3: Fixer les variables x_{ijt} en utilisant le prétraitement
4: Résoudre (MIP) avec les variables fixées

4.4 Heuristiques

Dans cette section, nous nous intéressons à la résolution approchée de notre problème. Nous avons proposé plusieurs versions de l'algorithme de liste. La meilleure solution calculée par ces versions sera améliorée par une matheuristique qui est une méthode de recherche par voisinage.

4.4.1 Algorithmes de liste

Dans cette section, nous décrivons plusieurs versions d'algorithmes de liste pour résoudre le problème d'évacuation par bus. Ces heuristiques ont pour but de calculer des solutions très rapidement mais sans aucune garantie d'optimalité. Nous soulignons la nécessité de ces heuristiques dans le cas d'instances de tailles réelles (donc de grande taille).

L'Algorithme 3 représente l'algorithme général des heuristiques de liste. Il prend pour entrée une règle $\mathcal{R}$, qui sera définie par la suite. Soit *ListOpr* l'ensemble des opérations qui seront affectées aux bus, *ListTriée* est l'ensemble des opérations triées selon la règle $\mathcal{R}$ à l'instant de temps t et T^b_{evac} la date de fin d'évacuation d'un bus b. La solution obtenue par l'algorithme de liste est sauvegardée dans *ListOrdo*. Chaque opération ajoutée dans la liste *ListOrdo* sera supprimée de *ListOpr* et de *ListTriée*.

La fonction *event* présentée dans le pseudo-code de l'Algorithme de liste 3, vérifie s'il y a un évènement nécessitant un recalcul de la liste des priorités *ListTriée*. Chaque règle $\mathcal{R}$ a sa propre fonction *event*.

Algorithme 3 Algorithme générique ($\mathcal{R}$)

Entrées: /* une règle de trie $\mathcal{R}$*/
1: $ListOrdo \leftarrow \emptyset$, $ListTriée \leftarrow \emptyset$
2: $ListOpr \leftarrow \{opr_1, opr_2, ..., opr_N\}$
3: $t \leftarrow 0$, $k \leftarrow B$
4: Trier les opérations dans *ListTriée* en utilisant la règle $\mathcal{R}$
5: **Répéter**
6: Ajouter les k premières opérations dans *ListOrdo*
7: Supprimer les k opérations ajoutées dans *ListOrdo* de *ListOpr* et de *ListTriée*
8: **si** t=0 **alors** $k \leftarrow 0$
9: Soit $b \in \mathcal{B}$ un bus qui a le T_{evac} minimum
10: $t \leftarrow T^b_{\text{evac}}$
11: $k \leftarrow k + 1$
12: **si** event() **alors**
13: $ListTriée \leftarrow \emptyset$
14: Trier les opérations dans *ListTriée* en utilisant la règle $\mathcal{R}$
15: **fin si**
16: **jusqu'à** $|ListOpr| = 0$ ou $t > T$
Sorties: une solution réalisable *ListOrdo*

Maintenant, nous présentons les différentes règles proposées :

1. Version 1 : cette version utilise la règle $\mathcal{R}_1$. Pour chaque opération opr_i d'un point de rassemblement j, on associe le nombre total tno_i d'opérations du point de rassemblement j. Les opérations opr_i sont triées selon l'ordre croissant de leurs valeurs tno_i. La fonction *event* retourne toujours "faux" parce que tno_i est constante tout au long de l'algorithme.
2. Version 2 : cette version utilise la règle $\mathcal{R}_2$. Dans cette règle, les opérations opr_i sont triées selon l'ordre décroissant de leurs valeurs tno_i. La fonction *event* a les mêmes

propriétés que celles de la version 1.

3. Version 3 : cette version utilise la règle $\mathcal{R}_3$. Cette version consiste à placer les opérations d'évacuation dans l'ordre croissant des durées opératoires. Cette version est une adaptation de la règle connue dans la théorie d'ordonnancement *Shortest Processing Time (SPT)* ([Smith, 1956]). L'adaptation a pour but de traiter les durées opératoires dépendantes de leurs dates de début p_{ijt}. Dans ce cas, la fonction *event* retourne "vrai" s'il y une modification dans les valeurs des durées opératoires des opérations d'évacuation.
4. Version 4 : cette version utilise la règle $\mathcal{R}_4$. Les opérations sont triées selon l'ordre croissant de leurs ratio $\frac{nos_j}{tno_j}$ avec nos_j le nombre d'opérations du point de rassemblement j qui sont déjà traitées, et tno_j est le nombre total des opérations du point de rassemblement j. Dans le cas où il y a plusieurs opérations qui ont la même priorité, ces opérations seront triées par la règle *SPT*. La fonction *event* retourne "vrai" s'il y a au moins une opération opr_k qui a un plus petit ratio que la première opération de la liste *ListTriée*.
5. Version 5 : cette version utilise la règle $\mathcal{R}_5$. Dans cette règle, les opérations sont triées selon l'ordre croissant du ratio $\frac{tno_j}{nto} - \frac{nos_j}{tno_j}$ avec tno_j le nombre total d'opérations du point de rassemblement j. La fonction *event* retourne "vrai" s'il y a au moins une opération opr_k qui a un plus petit ratio que la première opération de la liste *ListTriée*.
6. Version 6 : cette version utilise la règle $\mathcal{R}_6$. À chaque opération opr_i, soit nuo_i le nombre d'opérations du même point de rassemblement j qui ne sont pas encore traitées. Puis, les opérations sont triées selon l'ordre décroissant de leurs valeurs nuo_i. La fonction *event* retourne "vrai" si une opération opr_k a un nombre d'opérations non ordonnancées plus grand que la première opération de la liste *ListTriée*.
7. Version 7 : cette version utilise la règle $\mathcal{R}_7$. À chaque opération opr_i, soit nuo_i le nombre d'opérations qui ne sont pas encore ordonnancées du même point de rassemblement j. Ensuite, les opérations sont triées selon l'ordre croissant de leurs valeurs nuo_i. La fonction *event* retourne "vrai" si une opération opr_k a un nombre d'opérations non ordonnancées plus petit que la première opération dans la liste *ListTriée*.
8. Version 8 : cette version utilise la règle $\mathcal{R}_8$. Dans cette règle, nous affectons des priorités aléatoires à chaque point de rassemblement. La fonction *event* retourne "faux" sauf dans le cas d'un changement de la durée opératoire d'une opération.

4.4.2 Matheuristique

Dans cette section, nous proposons une méthode de recherche locale appelée : matheuristique. L'idée générale de la matheuristique est d'exploiter la performance des algorithmes exacts et des algorithmes traditionnels de recherche par voisinage. La combinaison de ces deux types d'algorithmes, donne une méthode dite hybride ([Della Croce et al., 2011]). La matheuristique proposée permet donc d'améliorer la qualité des solutions obtenues par les algorithmes de liste (cf. 4.4.1).

La construction de la matheuristique pour notre problème MBEP repose sur la première formulation indexée sur le temps introduite dans la section 4.1.1. Soit une solution

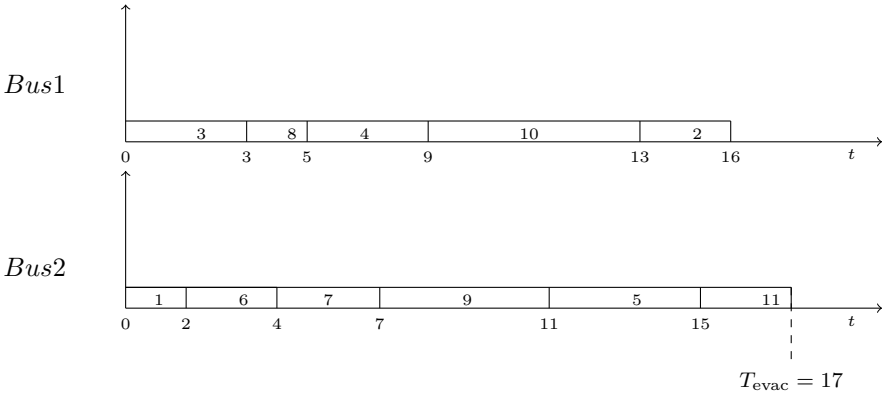
initiale qui est la meilleure des solutions obtenues par les algorithmes de liste. La matheuristique essaye d'améliorer cette solution en explorant son voisinage comme suit. Soit un ordonnancement réalisable $\bar{x} =< \bar{x}_{ijt}, i \in \mathcal{N}, j \in \mathcal{S}, t \in \mathcal{T} >$, avec $\bar{x}_{ijt} = 1$, si un bus débute l'évacuation des personnes depuis i vers j à l'instant de temps t. Nous définissons le voisinage $\mathcal{N}\ (\bar{x}, r, h)$ de cette solution $\bar{x}$ en choisissant une date r dans l'ordonnancement et un paramètre de taille h. Soit $\tilde{S}(r, h)$ l'ensemble des indices des opérations qui commencent dans l'intervalle de temps $[r, r + h[$. Ce sous-ensemble d'opérations est appelé "fenêtre-opérations". La meilleure solution dans le voisinage $\mathcal{N}\ (\bar{x}, r, h)$ est calculée en minimisant la date de fin d'ordonnancement de cette fenêtre T^w_{evac}, sous les contraintes (2.4)-(2.8) et les contraintes supplémentaires suivantes :

$$x_{ijt} = \bar{x}_{ijt} \qquad \forall i \notin \tilde{S}(r, h), \forall j \in \mathcal{M}, t \in [0, r[\tag{2.16}$$

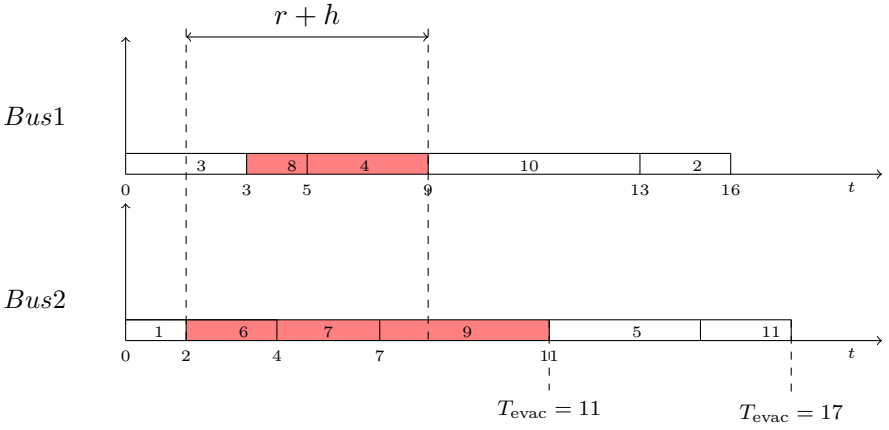
Le problème de la minimisation de T^w_{evac} est appelé problème de ré-optimisation de la fenêtre d'opérations. Ce problème est résolu à l'aide d'un solveur mathématique comme par exemple CPLEX. Les contraintes additionnelles (2.16) forcent les changements à avoir lieu dans la fenêtre d'opérations. Dans le cas d'une amélioration du critère T^w_{evac} dans la nouvelle solution $\tilde{x}$, les dates de début de toutes les opérations qui ont été placées après la date $r + h$ dans la solution initiale $\bar{x}$, seront décalées à gauche tout en gardant leurs anciennes positions et en respectant les contraintes de bus (2.6). Dans le cas contraire, une nouvelle fenêtre d'opérations est ré-optimisée en choisissant une nouvelle valeur de r. On continue le procédé jusqu'à ce que toutes les fenêtres soient sélectionnées. La recherche s'arrête s'il n'y a aucune solution optimale du problème de ré-optimisation qui améliore la solution courante, ou bien si le temps prédéfini pour la maheuristique est expiré. Le fonctionnement de la matheuristique est présenté dans la figure 2.4.

Exemple 3 *On considère l'exemple de la figure 2.4. Nous disposons de deux centres de secours et 11 opérations. Nous avons deux bus disponibles (Bus 1 et Bus 2). La solution courante est représentée dans la figure 2.4a : les opérations 3, 8, 4, 10, et 2 sont placées sur le Bus 1 et les opérations 1, 6, 7, 9, 5, et 11 sont ordonnancées sur le Bus 2. La date de début de la fenêtre d'opérations est* $r = 2$*, et la taille de cette fenêtre est* $h = 7$ *(figure 2.4b). La date de fin de cette solution est* $T_{evac} = 17$*, et la date de fin de la fenêtre à ré-optimiser est* $T^w_{evac} = 11$*. La solution obtenue* $\tilde{x}$ *par le calcul de voisinage, où le problème de ré-optimisation de la fenêtre d'opérations a été résolu, est donnée par la figure 2.4c. Cette figure montre que la date de fin de la fenêtre d'opérations et la date de fin de la solution complète ont été réduites. Les dates de début des opérations 5, 10, 2 et 11 ont été décalées à gauche, tout en gardant leurs positions antérieures. La date de fin de la nouvelle solution est* $T_{evac} = 15$*.*

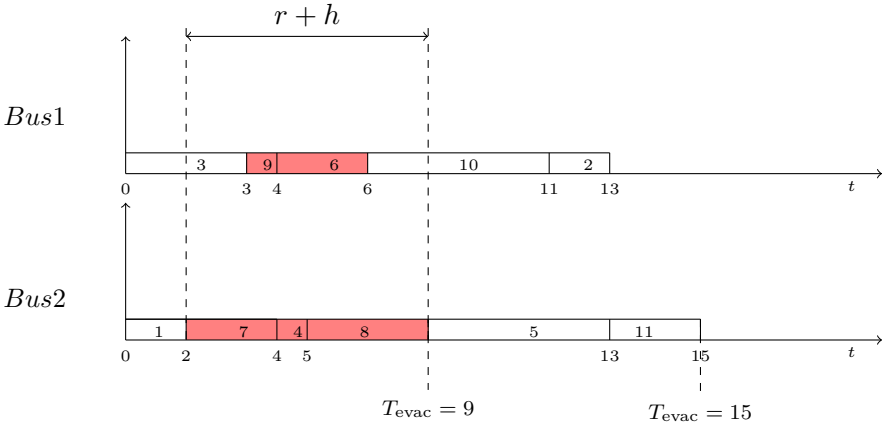
Les Algorithmes 4 et 8 représentent respectivement les étapes principales de la matheuristique et la fonction qui calcule le voisinage d'une solution.



(a) La solution initiale $\bar{x}$



(b) La fenêtre de ré-optimisation



(c) La solution voisine $\tilde{x}$

FIGURE 2.4 – Exemple d'une fenêtre de ré-optimisation optimisée.

Algorithme 4 Algorithme de la matheuristique

Entrées: $\bar{x}$ =<une solution approchée calculée par les algorithmes de liste ou d'autres heuristiques>

vect =<vecteur des dates de début de toutes les opérations de $\bar{x}$ >
Répéter
  amélioré $\leftarrow faux$
  $i \leftarrow 0$
  Répéter
    $r \leftarrow vect[i]$
    $\tilde{x} \leftarrow$ Voisinage$(\bar{x}, r, h)$
    si $(T_{\text{evac}}(\bar{x}) > T_{\text{evac}}(\tilde{x}))$ **alors**
      $\bar{x} \leftarrow \tilde{x}$
      improved $\leftarrow vrai$
      Pour $r \leftarrow r + h$ **à** $T_{\text{evac}} - r + h$ **faire**
        $\bar{y} \leftarrow$ Voisinage$(\bar{x}, r, h)$
        si $(T_{\text{evac}}(\bar{x}) > T_{\text{evac}}(\bar{y}))$ **alors**
          $\bar{x} \leftarrow \bar{y}$
        fin si
      fin Pour
      recalculer *vect* à partir de $\bar{x}$
    sinon $i \leftarrow i + 1$
    fin si
  jusqu'à $i \geq |vect|$ **ou** amélioré = vrai **ou** limite de temps expiré
jusqu'à amélioré= faux **ou** limite de temps expiré

Sorties: une solution réalisable $\bar{x}$

Algorithme 5 La procédure Voisinage

Calculer $\tilde{S}(r, h)$
Réordonnancer les opérations dans la fenêtre de temps $[r, r + h[$ avec la minimisation de $T^w_{\text{evac}}(\bar{x})$ sous les contraintes (2.5)-(2.8), et (2.16)
  si (T^w_{evac} est amélioré) **alors**
    Décaler à gauche les dates début des opérations commençant après $r + h$
    Soit $\tilde{y}$ la solution obtenue
    Retourner la solution $\tilde{y}$
  sinon
    Retourner la solution $\bar{x}$
  fin si

4.5 Expérimentations

Cette section présente les résultats expérimentaux réalisés pour évaluer l'efficacité des deux modélisations proposées, du prétraitement, des inégalités valides introduites dans la section (4.3) et des heuristiques. Tout d'abord, nous commençons par une présentation de l'environnement de développement.

Environnement : Toutes les expérimentations sont faites sur un ordinateur avec 8 cœurs Intelprocessor/2.60 GHz avec 20MB de mémoire cache, 8 GB de RAM et sous windows 7. Le code a été implémenté avec C++. Nous avons utilisé le solveur CPLEX v.12.2 (ILOG (2012)). Un seul cœur a été utilisé par CPLEX pour la résolution des formulations mathématiques.

Les données : les instances sont générées aléatoirement de sorte que nous ayons toujours des instances proches de celles de la ville de Nice. Le nombre de centres de secours S prend les valeurs suivantes : 2, 4, 6, 8 et 10. La capacité de chaque centre de secours j, cap_j est générée aléatoirement dans $\{20, 21, 22, ..., 40\}$. Le nombre de point de rassemblement $P \in \{10, 20, 30, 40\}$. Pour éviter de se retrouver avec un nombre d'opérations supérieur à la capacité totale des centres de secours, nous choisissons le nombre d'opérations d'évacuation associées à chaque point de rassemblement i dans l'intervalle :

$$e_i \in \left[\frac{1}{4}\frac{0.9\sum_j cap_j}{L}; \frac{7}{4}\frac{0.9\sum_j cap_j}{L}\right]$$

Nous supposons que la période de temps nécessaire pour l'évacuation est $[J; J+1[$, où J est le jour de la catastrophe. Ce qui implique que l'horizon temporel T est égal à 48 heures. Dans ce qui suit, nous faisons une discrétisation de temps égale à un quart d'heure. Par conséquent, l'horizon temporel T sera égal à 192 quarts d'heure.
La durée opératoire des opérations et les valeurs du risque sont dépendantes de leurs dates de début : nous considérons six événements qui se produisent à des dates connues et qui peuvent éventuellement modifier les durées opératoires des opérations. Il y a deux dates de dégradation qui sont communes à toutes les opérations et quatre dates d'amélioration spécifiques à chaque opération. Les dates de dégradation sont générées dans l'intervalle suivant : T_1^D,$T_2^D \in [0, 192[$. Quant aux dates d'amélioration, elles sont générées dans les intervalles : $T_1^A \in [0, T_1^D]$,$T_2^A \in]T_1^D, T_2^D]$, $T_3^A \in]T_2^D, 192[$. Les durées opératoires des opérations $p_{ijt} \in [2, 12]$. Les valeurs de risque $r_{ijt} \in [0, 10]$. Une petite valeur de risque indique que le chemin est sûr, dans le cas contraire le chemin est risqué.

Pour calculer le nombre de bus disponibles pour l'opération d'évacuation, nous utilisons la formule suivante :

$$B = \frac{\sum e_i * \bar{P}}{180}$$

où $\sum e_i * \bar{P}$ est la durée moyenne des durées opératoires. Nous supposons qu'un bus effectue l'opération d'évacuation pendant 45 heures (180 quarts d'heure).

Pour chaque instance, nous avons effectué les expérimentations suivantes :

– algorithmes exacts : tout d'abord, nous avons testé avec le solveur CPLEX les deux formulations présentées précédemment. Nous notons respectivement par IP1 et IP2 les solutions obtenues par la première formulation et les solutions obtenues par la deuxième formulation. Dans un second temps, en utilisant CPLEX, nous évaluons l'impact du prétraitement sur la résolution exacte de la première formulation (cf. section 4.1.1). Soient Pre-IP ces solutions. Enfin, nous évaluons l'impact des inégalités valides sur les résultats du prétraitement. Ces solutions seront notées par PRE-IP-VI. Notons que pour initialiser la valeur de l'horizon de temps T pour ces modèles, nous utilisons la meilleur valeur de T_{evac} obtenue par les heuristiques. Nous imposons à CPLEX une limite de temps de 1800 secondes et une limite mémoire de 1 GB.

– algorithmes heuristiques :
  – pour chaque instance, nous avons testé les huit versions de l'heuristique de liste, et nous gardons la meilleure solution calculée.
  – nous avons testé la matheuristique avec une limite de temps de 600 secondes. Selon des expérimentations préliminaires antérieures, les meilleures solutions sont obtenues avec une taille de la fenêtre $h = 25$. La solution initiale est la meilleure solution obtenue par les algorithmes de liste.
  – nous imposons à CPLEX une limite de temps égale à celle imposée à la matheuristique, pour résoudre la première formulation (cf. section 4.1.1). Cette heuristique sera notée par IPheuristique.

Notons que les instances sont générées aléatoirement à partir du nombre de points de rassemblement et du nombre de centres de secours. Cependant, la difficulté de la résolution (exacte et/ou heuristique) de ces instances est due principalement au nombre total d'opérations d'évacuation, $\sum_l e_l$. Par conséquent, ces instances sont regroupées par classe "de taille équivalente", en termes de nombre d'opérations, nombre de centres de secours et nombre de bus.

Les Tableaux 2.1, 2.2, et 2.3 présentent les différentes comparaisons entre les deux formulations IP1 et IP2.

Dans le Tableau 2.1, nous comparons entre le temps d'exécution de IP1 et de IP2. Les statistiques présentées dans ce tableau concernent les instances pour lesquelles CPLEX résout à l'optimalité IP1 et IP2. Ce tableau présente le nombre d'opérations total (#oprs), le nombre de centres de secours (#centres), le nombre de bus (#bus), le nombre d'instances total pour chaque taille du problème (#inst_tot), et le nombre d'instances qui sont résolues à l'optimalité par CPLEX pour IP1 et IP2 (#inst_opt). Nous observons que pour les petites instances, le temps d'exécution de IP1 est meilleur que celui de IP2. Pour les mêmes instances de grande taille, CPLEX échoue à résoudre à l'optimalité les deux formulations IP1 et IP2.

#oprs	#centres	#bus	#inst_tot	#inst_opt	IP1 (temps)			IP2 (temps)		
					min	avg	max	min	moy	max
[20,35[	2	[1,2]	32	31	27	244,9	588	60	479,8	1258
[35,40[	2	2	17	10	387	834,2	1587	421	1221	1787
[40,60[	2	[2,3[	31	11	175	827,7	1405	324	954	1737
[40,60[	4	[2,3[	6	0	/	/	/	/	/	/
[60,70[	4	3	13	0	/	/	/	/	/	/
[70,80[	4	[3,4[	17	0	/	/	/	/	/	/
[80,90[	4	4	26	0	/	/	/	/	/	/
[90,110[	4	[4,5[	18	0	/	/	/	/	/	/

TABLE 2.1 – Comparaisons des temps d'exécution

Les résultats du Tableau 2.2 présentent le nombre de nœuds générés par CPLEX pour la résolution optimale de IP1 et IP2. Nous constatons que la résolution des instances à l'optimalité par IP1 génère moins de nœuds que pour IP2.

Afin de mesurer la qualité des solutions obtenues par les deux modélisations, nous considérons la déviation entre les solutions des instances, où au moins une des deux modélisations n'arrive pas à trouver la solution optimale, et la meilleure solution calculée. La

#oprs	#centres	#bus	#inst_tot	#inst_opt	IP1 (nœuds)			IP2 (nœuds)		
					min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	31	0	1806	2150	0	1068	3324
[35,40[	2	2	17	10	729	1983	3873	617	2146	4844
[40,60[	2	[2,3[	31	11	512	1476	3324	541	2197	6245
[40,60[	4	[2,3[	6	0	/	/	/	/	/	/
[60,70[	4	3	13	0	/	/	/	/	/	/
[70,80[	4	[3,4[	17	0	/	/	/	/	/	/
[80,90[	4	4	26	0	/	/	/	/	/	/
[90,110[	4	[4,5[	18	0	/	/	/	/	/	/

TABLE 2.2 – Nombre de nœuds explorés

formule donnant la déviation est la suivante :

$$deviation = \frac{T_{\text{evac}}(IP1/IP2) - \min\left(T_{\text{evac}}(IP1), T_{\text{evac}}(IP2)\right)}{\min\left(T_{\text{evac}}(IP1), T_{\text{evac}}(IP2)\right)} * 100 \tag{2.17}$$

Le Tableau 2.3 présente les différentes valeurs de la déviation pour chaque formulation. D'après les résultats de ce tableau, nous observons que IP1 (#inst_opt) arrive à résoudre à l'optimalité plus d'instances que IP2. De plus, en moyenne, pour la majorité des classes d'instances, la qualité des solutions réalisables par IP1 est meilleure que celle des solutions obtenues par IP2.

#oprs	#centres	#bus	#inst_opt		#inst_prbm		#inst_prbt		IP1 (déviation%)			IP2 (déviation%)		
			IP1	IP2	IP1	IP2	IP1	IP2	min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	31	0	0	0	1	0	0	0	34,38	34,38	34,38
[35,40[	2	2	13	13	0	0	4	4	0	16,22	52,7	4,05	22,2	36,49
[40,60[	2	[2,3[	21	19	0	0	10	12	0	25,62	123,5	0	33,27	114,8
[40,60[	4	[2,3[	4	0	0	0	2	6	0	57,49	158	0	94,69	179,7
[60,70[	4	3	0	0	0	0	13	13	0	66,49	188,1	35,82	154	192,5
[70,80[	4	[3,4[	0	0	0	0	17	17	0	62,61	211,1	93,65	172,5	211,1
[80,90[	4	4	0	0	0	0	26	26	0	97,67	142,7	89,02	122,7	142,7
[90,110[	4	[4,5[	0	0	0	0	18	18	1.65	4,86	8,84	0	3,31	8,84

TABLE 2.3 – La déviation (instances non résolues)

D'après les Tableaux présentés ci-dessus, IP1 est meilleur que IP2 en terme de temps de calcul. Aussi les solutions obtenues par IP1 sont meilleures que celles obtenus par IP2. Pour ces raisons, nous utiliserons par la suite la formulation IP1 avec le prétraitement et aussi dans la matheuristique.

Les Tableaux 2.4, 2.5 et 2.6 montrent les différentes comparaisons entre IP1 et Pre-IP. Rappelons que dans la section 4.2, le nombre de variables de décision fixées dépend de la valeur de la borne supérieure (UB). Si la borne est proche de la solution optimale, alors ce nombre devrait être élevé. Cependant, il est intéressant de voir l'influence du prétraitement en utilisant la meilleure borne supérieure possible, c'est-à-dire : la solution optimale elle même. Ces résultats nous permettent d'évaluer l'impact du prétraitement dans le meilleur cas où la borne supérieure est la solution optimale. Par conséquent, les pourcentages des variables fixées dans les Tableaux 2.4, 2.5, et 2.6 sont des bornes supérieures aux pourcentages de variables fixées dans le cas où une borne supérieure est donnée par une

heuristique. En effet, pour chaque instance la borne supérieure utilisée dans 2.4, 2.5, et 2.6 est la solution calculée par IP1 qui est optimale pour la majorité des instances de petite taille (dû à la limite de temps imposée au solveur CPLEX). Par contre, dans les Tableaux 2.7, 2.8, et 2.9, la borne supérieure du prétraitement est la meilleure des solutions trouvées par les heuristiques.

Le Tableau 2.4 présente les temps d'exécution de IP1 et de Pre-IP, et le pourcentage de variables fixées. Nous considérons uniquement les instances pour lesquelles CPLEX résout IP1 et Pre-IP à l'optimalité. Ce tableau présente aussi le nombre total d'opérations (#oprs), le nombre de centres de secours (#centres), le nombre de bus (#bus), le nombre total d'instances pour chaque taille du problème (#inst_tot), le nombre d'instances qui ont été résolues à l'optimalité par IP1 et par Pre-IP (#inst_opt). La colonne IP1 (temps(s)) correspond aux temps minimaux, moyens, et maximaux d'exécution en secondes de CPLEX pour la résolution de IP1. De même, la colonne Pre-IP(temps(s)) présente les temps d'exécution pour la résolution de Pre-IP. Nous remarquons que pour le calcul des solutions optimales, Pre-IP est toujours plus rapide que IP1. Par ailleurs, pour certaines instances, Pre-IP est très efficace ; CPLEX est capable de fixer plus de 85% des variables. Malheureusement, dans le cas où le nombre d'opérations varie entre 90 et 110, IP1 et Pre-IP ne peuvent résoudre les instances à l'optimalité.

#oprs	#centres	#bus	#inst_tot	#inst_opt	IP1 (temps(s))			Pre-IP (temps(s))			Pre-IP (%varfix)		
					min	moy	max	min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	15	23,72	87	1	4,78	10	4,52	30,46	85,12
[35,40[	2	2	17	16	14	63,06	200	10	18,81	32	4,15	15,83	53,95
[40,60[	2	[2,3]	31	30	17	106,8	384	9	26,77	65	2,33	21,85	64,55
[40,60[	4	[2,3]	6	5	85	323	859	62	173,6	356	3,84	35,61	92,05
[60,70[	4	3	13	6	175	921,2	1710	175	261	424	7,9	45,88	94,15
[70,80[	4	[3,4]	17	1	1220	1220	1220	164	164	164	88,81	88,81	88,81
[80,90[	4	4	26	1	610	610	610	366	366	366	15,2	15,2	15,2
[90,110[	4	[4,5]	18	0	/	/	/	/	/	/	/	/	/

TABLE 2.4 – Comparaison des temps d'exécution (UB=Solution opt)

Dans le Tableau 2.5, nous présentons le nombre de nœuds nécessaires qui sont explorés pour le calcul d'une solution optimale par IP1 et Pre-IP. Évidemment, CPLEX avec le prétraitement explore moins de nœuds que IP1. Nous remarquons que pour certaines instances (nombre d'opérations $\in$ [40, 60[), en utilisant le Pre-IP, CPLEX est capable de calculer la solution optimale au nœud racine en n'explorant aucun nœud. En revanche, sans le prétraitement, CPLEX explore en moyenne 207 nœuds.

Dans le Tableau 2.6, nous comparons la qualité des solutions obtenues par IP1 et Pre-IP. Nous considérons les instances pour lesquelles IP et/ou Pre-IP ne donnent pas la solution optimale. Soit dev(A) la déviation associée à l'algorithme A :

$$deviation(A) = \frac{T_{\text{evac}}(A) - \min(T_{\text{evac}}(IP1), T_{\text{evac}}(Pre-IP))}{\min(T_{\text{evac}}(IP1), T_{\text{evac}}(Pre-IP))} * 100$$

Le Tableau 2.6 présente le nombre d'instances résolues à l'optimalité (#inst_opt), le nombre d'instances non résolues quand CPLEX atteint la limite de temps (#inst_prbt), et

#oprs	#centres	#bus	#inst_tot	#inst_opt	IP1 (nœuds)			Pre-IP (nœuds)		
					min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	0	217,7	901	0	10,6	262
[35,40[	2	2	17	16	0	243,5	784	0	29,31	274
[40,60[	2	[2,3]	31	30	0	367,7	4251	0	37,8	568
[40,60[	4	[2,3]	6	5	0	207,8	602	0	0	0
[60,70[	4	3	13	6	0	359,5	872	0	0,33	1
[70,80[	4	[3,4]	17	1	1015	1015	1015	10	10	10
[80,90[	4	4	26	1	147	147	147	1	1	1
[90,110[	4	[4,5]	18	0	/	/	/	/	/	/

TABLE 2.5 – Nombre de nœuds explorés (UB= Solution opt)

le nombre d'instances non résolues quand l'arbre de recherche de CPLEX dépasse la limite de mémoire imposée (#inst_prbm). Les résultats du Tableau 2.6 montrent que la déviation de Pre-IP est toujours inférieure à celle de IP1. Ce qui signifie que pour les instances non résolues à l'optimalité, la valeur de la date de fin d'ordonnancement T_{evac} calculée par Pre-IP est toujours plus petite que celle calculée par IP1. D'autre part, pour quelques instances où le nombre de centres de secours est égale à 4 et le nombre d'opérations est supérieur à 40 opérations, Pre-IP est capable de résoudre 26% des instances à l'optimalité. Tandis que, pour ces mêmes instances, IP1 fournit des solutions réalisables. Notons que dans le cas où le nombre d'opérations appartient à $[90, 110[$, IP1 échoue à calculer des solutions réalisables pour 17 parmi 18 instances.

#oprs	#centres	#bus	#inst_opt		#inst_prbm		#inst_prbt		IP1 (déviation%)			Pre-IP (déviation%)		
			IP	Pre-IP	IP	Pre-IP	IP	Pre-IP	min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	0	0	0	0	/	/	/	/	/	/
[35,40[	2	2	16	16	0	0	1	1		0	0	0	0	0
[40,60[	2	[2,3]	30	30	0	0	1	1	0	0	0	0	0	0
[40,60[	4	[2,3]	5	6	0	0	1	0	0	0	0	0	0	0
[60,70[	4	3	6	9	0	0	7	4	0	4,27	18,09	0	0	0
[70,80[	4	[3,4]	1	7	0	0	16	10	0	8,09	17,39	0	0	0
[80,90[	4	4	1	5	0	0	25	21	0	5,61	16,44	0	0	0
[90,110[	4	[4,5]	0	9	0	0	18	9	19,23	19,23	19,23	0	0	0

TABLE 2.6 – La déviation (instances non résolues, UB= Solution opt)

Par la suite, la borne supérieure utilisée pour Pre-IP est la meilleure solution calculée par les heuristiques (matheuristique et IPheuristique). Les Tableaux 2.7, 2.8 et 2.9 présentent les résultats de cette expérimentation. Ces tableaux sont disposés de la même manière que les Tableaux 2.4, 2.5, et 2.6.
À partir du Tableau 2.7, nous observons que pour les instances où le nombre d'opérations est inférieur à 60 le choix de la borne supérieure n'influe pas sur le nombre d'instances résolues à l'optimalité par Pre-IP, ainsi que sur le nombre de variables fixées (cf. Tableau 2.4). En revanche, la résolution à optimalité de Pre-IP nécessite beaucoup plus de temps que la résolution de Pre-IP en utilisant la solution optimale comme une borne supérieure. Pour un nombre d'opérations plus grand, le nombre d'instances résolues à l'optimalité est petit. De plus, dans le cas où le nombre d'opérations appartient à $[80, 90[$, le temps CPU n'est pas significatif vu qu'une seule instance est résolue en moins de temps par IP que par Pre-IP. Notons que le temps CPU n'inclue pas le temps de calcul des heuristiques, parce

que ces dernières ont été utilisées pour initialiser à la fois l'horizon de temps de IP1 et la borne supérieure de Pre-IP.

#oprs	#centres	#bus	#inst_tot	#inst_opt	IP1 (temps(s))			Pre-IP (temps(s))			Pre-IP(%varfix)		
					min	moy	max	min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	15	23,72	87	1	4,78	10	4, 52	30,46	85,12
[35,40[	2	2	17	16	14	63,06	200	10	18,69	32	4,15	15,83	53,95
[40,60[	2	[2,3]	31	30	17	106,8	384	9	26,87	65	2,33	21,85	64,55
[40,60[	4	[2,3]	6	5	85	323	859	74	212	483	3,34	17,17	63,04
[60,70[	4	3	13	6	175	921	1710	245	938	1747	4,34	5,25	6,11
[70,80[	4	[3,4]	17	1	1220	1220	1220	829	829	829	5,09	5,09	5,09
[80,90[	4	4	26	1	610	610	610	675	675	675	5,94	5,94	5,94
[90,110[	4	[4,5]	18	0	/	/	/	/	/	/	/	/	/

TABLE 2.7 – Comparaisons des temps d'exécution

Le Tableau 2.8 présente le nombre de nœuds explorés par IP1 et Pre-IP. Bien que la borne supérieure soit calculée par les heuristiques, le nombre de nœuds explorés par Pre-IP est plus petit que le nombre de nœuds explorés par IP1, à l'exception des opérations dont le nombre appartient à l'intervalle $[60, 70[$.

#oprs	#centres	#bus	#inst_tot	#inst_opt	IP1 (nœuds)			Pre-IP (nœuds)		
					min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	0	217,7	901	0	10,6	262
[35,40[	2	2	17	16	0	243,5	784	0	29,31	274
[40,60[	2	[2,3]	31	30	0	367,7	4251	0	37,8	568
[40,60[	4	[2,3]	6	5	0	207,8	602	0	31,4	156
[60,70[	4	3	13	6	0	359,5	872	0	444,2	1120
[70,80[	4	[3,4]	17	1	1015	1015	1015	180	180	180
[80,90[	4	4	26	1	147	147	147	38	38	38
[90,110[	4	[4,5]	18	0	/	/	/	/	/	/

TABLE 2.8 – Nombre de nœuds explorés

Les résultats du Tableau 2.9 concernent la déviation. Nous constatons que les solutions calculées par Pre-IP sont souvent meilleures que celles calculées par IP1. Par exemple, dans le cas où le nombre d'opérations est plus grand que 40, Pre-IP résout à l'optimalité 13% des instances, alors que IP1 calcule des solutions réalisables pour les mêmes instances. Comme prévu, le pourcentage de variables fixées est plus petit que le pourcentage obtenu par Pre-IP qui utilise la solution optimale comme une borne supérieure.

Nous précisons que les résultats présentés dans les Tableaux (2.10, 2.11 et 2.12) ont été obtenus par la version CPLEX 12.6. Le Tableau 2.10 présente les temps CPU nécessaires pour le calcul des solutions optimales en utilisant Pre-IP et Pre-IP avec les inégalités valides, noté par Pre-IP-IV. Ce tableau est construit de la même manière et avec les mêmes données que dans le Tableau 2.7.

Les résultats des Tableaux 2.10 et 2.11 montrent que l'utilisation des inégalités valides diminue légèrement le temps d'exécution du Pre-IP. De plus, le nombre de variables fixées est supérieur à celui de Pre-IP. Notamment, dans le cas où le nombre d'opérations d'évacuation $\#oprs \in [20, 35[$.

Les statistiques du Tableau 2.12 sont faites sur les instances où CPLEX n'arrive pas

#oprs	#centres	#bus	#inst_opt		#inst_prbm		#inst_prbt		IP1 (déviation%)			Pre-IP (déviation%)		
			IP	Pre-IP	IP	Pre-IP	IP	Pre-IP	min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	0	0	0	0	0	0	0	0	0	0
[35,40[	2	2	16	16	0	0	1	1		0	0	0	0	0
[40,60[	2	[2,3]	30	30	1	1	0	0	/	/	/	/	/	/
[40,60[	4	[2,3]	5	6	0	0	1	0	0	0	0	0	0	0
[60,70[	4	3	6	8	0	0	7	5	0	9,58	15,07	0	0	0
[70,80[	4	[3,4]	1	4	0	0	25	13	0	10,57	27,03	0	1,08	10,81
[80,90[	4	4	1	6	0	0	25	20	0	7,74	18,92	0	3,82	34,43
[90,110[	4	[4,5]	0	0	0	0	18	18	/	/	/	/	/	/

TABLE 2.9 – La déviation (instances non résolues)

#oprs	#centres	#bus	#inst_tot	#inst_opt	Pre-IP (temps(s))			Pre-IP-IV (temps(s))		
					min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	1	2	9	1	2	7
[35,40[	2	2	17	17	7	21,94	128	7	25,12	118
[40,60[	2	[2,3]	31	30	8	35,17	123	8	26,1	189
[40,60[	4	[2,3]	6	6	76	179,5	322	79	190	328
[60,70[	4	3	13	13	214	621,5	1516	217	547	1400
[70,80[	4	[3,4]	17	14	369	1115	1739	310	1109	1739
[80,90[	4	4	26	9	374	873	1317	392	811	1341
[90,110[	4	[4,5]	18	1	1076	1076	1076	1133	1133	1133

TABLE 2.10 – Comparaisons des temps d 'exécution

#oprs	#centres	#bus	#inst_tot	#inst_opt	Pre-IP(%varfix)			Pre-IP-IV(%varfix)		
					min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	6, 25	24,4	85,89	6,25	35,04	90
[35,40[	2	2	17	17	5,06	16,94	72,58	5,23	19,4	72,98
[40,60[	2	[2,3]	31	30	5,15	23,55	66,4	5,15	24,01	67,76
[40,60[	4	[2,3]	6	6	6,7	17,26	64,21	6,7	18	66
[60,70[	4	3	13	13	5,10	6,35	7,7	5,10	7	9
[70,80[	4	[3,4]	17	14	4,5	6,5	8,35	4,5	7	8,9
[80,90[	4	4	26	9	5,35	7,77	9,76	5,36	9,08	11,04
[90,110[	4	[4,5]	18	1	6,99	6,99	6,99	6,99	6,99	6,99

TABLE 2.11 – Le pourcentage de variables fixées

à résoudre à l'optimalité Pre-IP ou Pre-IP-VI. Le Tableau 2.12 présente le nombre d'instances résolues à l'optimalité par Pre-IP et Pre-IP-VI (#inst_opt), le nombre d'instances pour lesquelles CPLEX atteint la limite de temps (#inst_prbt), les déviations respectives du Pre-IP (Pre-IP (déviation%)) et Pre-IP-VI (Pre-IP-IV (déviation%)) par rapport à la meilleure solution obtenue par Pre-IP ou Pre-IP-VI. Nous observons que l'utilisation des inégalités valides aide à résoudre plus d'instances à l'optimalité. Nous remarquons que les solutions obtenues par Pre-IP-VI sont meilleures que celles obtenues par Pre-IP. Nous concluons que l'utilisation du prétraitement et les inégalités valides améliorent la phase de résolution exacte du modèle IP1 en terme de temps d'exécution, le nombre de nœuds explorés, et le nombre d'instances résolues à l'optimalité.

Dans ce qui suit, nous évaluons l'efficacité des heuristiques proposées.

Le Tableau 2.13 présente les résultats de IPheuristique qui sont le temps CPU et la

#oprs	#centres	#bus	#inst_opt		#inst_prbt		Pre-IP (déviation%)			Pre-IP-IV (déviation%)		
			Pre-IP	Pre-IP -IV	Pre-IP	Pre-IP -IV	min	moy	max	min	moy	max
[20,35[	2	[1,2]	32	32	0	0	0	0	0	0	0	0
[35,40[	2	2	16	**17**	0	0	0	0	0	0	0	0
[40,60[	2	[2,3]	30	30	0	0	/	/	/	/	/	/
[40,60[	4	[2,3]	6	6	0	0	0	0	0	0	0	0
[60,70[	4	3	13	13	0	0	0	0	0	0	0	0
[70,80[	4	[3,4]	14	**16**	3	1	0	4,92	19,7	0	0	0
[80,90[	4	4	9	**11**	17	15	0	4,67	14,1	0	1,35	8,6
[90,110[	4	[4,5]	1	1	18	18	0	6,5	24,39	0	0,56	6,37

TABLE 2.12 – La déviation (instances non résolues)

déviation de la solution obtenue par IPheuristique par rapport à la meilleure solution obtenue par les heuristiques. En effet, la meilleure solution est obtenue soit par IPheuristique elle même ou bien par la matheuristique (les résultats de la matheuristique sont au moins aussi bons que les solutions des algorithmes de liste). La déviation d'une heuristique "H" est calculée comme suit :

$$deviation(H) = \frac{T_{\text{evac}}(H) - \min(T_{\text{evac}}, T_{\text{evac}}(meilleure - solution))}{\min(T_{\text{evac}}, T_{\text{evac}}(meilleure - solution))} * 100$$

IPheuristique obtient de bons résultats quand le nombre d'opérations est plus petit que 40 et nécessite moins de 600 secondes pour fournir la meilleure solution (meilleure-solution). Par contre dans le cas où le nombre d'opérations appartient à $[40, 70[$, nous observons une détérioration dans la qualité de cette heuristique. Par exemple, dans certains cas, les solutions calculées par la IPheuristique sont loin de la meilleure solution de 95%. En outre, nous remarquons que pour les plus grandes instances, la IPheuristique devient moins compétitive, et échoue à calculer des solutions de bonne qualité dans la limite de temps de 600 secondes.

#oprs	#centres	#bus	deviation%			CPU temps (s)		
			min	moy	max	min	moy	max
[20,35[	2	[1,2]	0	0	0	17	49	101
[35,40[	2	2	0	0	0	98	217,8	600
[40,60[	2	[2,3]	0	0	0	53	277,8	600
[40,60[	4	[2,3]	0	1.35	42,06	575	597	600
[60,70[	4	3	0	1	3,84	600	600	600
[70,80[	4	[3,4]	38,64	71,24	115,25	600	600	600
[80,90[	4	4	26,42	64,37	105,1	600	600	600
[90,110[	4	[4,5]	23,77	58,73	95.35	600	600	600

TABLE 2.13 – Évaluation de la IPheuristique

Dans le Tableau 2.14, nous comparons les solutions obtenues par les algorithmes de liste et les meilleures solutions trouvées (donnée soit par IPheuristique, soit par la matheuristique). Toutes les règles de priorité ont été testées sur des instances de petite et de grande taille. Pour une instance donnée, chaque règle proposée fournie au moins une meilleure

#oprs	#centres	#bus	déviation%		
			min	moy	max
[20,35[	2	[1,2]	5,26	12,89	27,42
[35,40[	2	2	7,5	14,58	25,29
[40,60[	2	[2,3]	9	15,85	30,84
[40,60[	4	[2,3]	4,34	8,91	13,04
[60,70[	4	3	0	2,45	12,35
[70,80[	4	[3,4]	0	0,72	3,15
[80,90[	4	4	0	0,92	6,94
[90,110[	4	[4,5]	0	1,57	9,57

TABLE 2.14 – Évaluation des algorithmes de liste (I)

solution. Nous précisons que le temps CPU de ces heuristiques de liste est négligeable (résultats instantanés).

Le Tableau 2.15 présente la déviation de la matheuristique par rapport à la meilleure solution obtenue par les heuristiques. Par exemple, pour certaines instances où le nombre d'opérations est inférieur à 70 et la valeur de la déviation minimale est égale à 0, la matheuristique donne la meilleure solution. Pour plus de 60 opérations, la matheuristique est devenue en moyenne plus performante que IPheuristique. Dans le cas où le nombre d'opérations appartient à $[60, 70[$ et pour 12 instances parmi 13, la matheuristique fournit les meilleures solutions. De plus, dans le cas où le nombre d'opérations appartient à $[70, 80[$, les meilleures solutions sont toujours calculées par la matheuristique dans la limite de temps de 600 secondes. On peut noter néanmoins que la taille des données augmentent grandement, les heuristiques de liste tendent à donner des résultats largement acceptables en termes de déviation moyenne.

#oprs	#centres	#bus	deviation %			CPU time (s)		
			min	avg	max	min	avg	max
[20,35[	2	[1,2]	0	5,2	10,29	32	157,8	370
[35,40[	2	2	0	8.6	24,14	97	326,2	603
[40,60[	2	[2,3]	0	7,97	20	166	399,7	600
[40,60[	4	[2,3]	0	3.57	11,11	600	600	600
[60,70[	4	3	0	0,38	5,05	600	600	600
[70,80[	4	[3,4]	0	0	0	600	600	600
[80,90[	4	4	0	0	0	600	600	600
[90,110[	4	[4,5]	0	0	0	600	600	600

TABLE 2.15 – Évaluation de la matheuristique (I)

Le Tableau 2.16 présente l'évaluation des performances des algorithmes de liste uniquement par rapport à la matheuristique. Les trois dernières colonnes présentent la déviation des algorithmes de liste par rapport à la matheuristique. Les résultats du Tableau 2.16 indiquent que la matheuristique proposée améliore significativement la meilleure solution des heuristique de liste en un temps de 600 secondes. La matheuristique est moins performante pour un nombre assez grand d'opérations à cause de la limite de temps imposée.

Le Tableau 2.17 compare la qualité des solutions obtenues par la matheuristique avec les solutions optimales obtenues par IP1. La quatrième colonne de ce tableau est le nombre total d'instances, la cinquième colonne représente le nombre d'instances résolues à l'optimalité par IP1. Les trois dernières colonnes présentent les déviations minimale, moyenne et

#oprs	#centres	#bus	deviation%		
			min	moy	max
[20,35[	2	[1,2]	0	9.8	80.85
[35,40[	2	2	0	6.26	16.22
[40,60[	2	[2,3]	0.95	8.11	30.84
[40,60[	4	[2,3]	0	4.58	11.11
[60,70[	4	3	0	2.06	12.36
[70,80[	4	[3,4]	0	0,72	3,15
[80,90[	4	4	0	0,92	6,94
[90,110[	4	[4,5]	0	1,57	9,57

TABLE 2.16 – Évaluation des algorithmes de liste (II)

maximale des solutions de la matheuristique par rapport aux valeurs correspondant aux solutions optimales. Dans le cas de certaines petites instances, la matheuristique est capable de calculer des solutions optimales. Les autres solutions obtenues par la matheuristique sont proches des solutions optimales.

#oprs	#centres	#bus	#inst	#opt	déviation%		
					min	moy	max
[20,35[	2	[1,2]	32	31	0	4.98	10.29
[35,40[	2	2	17	16	0	7.79	24.14
[40,60[	2	[2,3]	31	26	0	7.55	20
[40,60[	4	[2,3]	6	5	4	7.94	16.67
[60,70[	4	3	13	6	4.47	9.13	12.68
[70,80[	4	[3,4]	18	1	0	0	0
[80,90[	4	4	27	1	11.43	11.43	11.43

TABLE 2.17 – Évaluation de la matheuristique (II)

Nous avons aussi testé toutes les heuristiques sur de grandes instances, c'est-à-dire des instances avec 6, 8, 10, 20, et 30 centres de secours et aussi pour plus de 110 opérations d'évacuation. Pour ces instances, nous avons trouvé que les heuristiques de liste calculent des solutions réalisables en quelques millisecondes. De plus, à cause de la limite de temps imposée qui est trop petite, la matheuristique ne peut pas améliorer la meilleure solution obtenue par les heuristiques. Nous remarquons aussi que pour la majorité des instances, IPheuristique échoue à calculer des solutions réalisables.

À travers les tableaux de cette section, nous concluons que pour des instances de grande taille et pour une limite de temps suffisante, la matheuristique est la meilleure heuristique. Ces résultats semblent être un bon compromis entre la qualité et le temps CPU. Si le temps imparti pour résoudre le problème est très limité alors les heuristiques de liste constituent des alternatives intéressantes.

5 Le problème robuste

D'une manière générale les données des problèmes d'évacuation sont connues avec incertitude. Jusqu'à présent nous avons fait l'hypothèse que les données du problème étaient connues avec certitude (le nombre d'évacués, nombre de bus, etc.) où variables dans le temps (durée de transport). Dans ce dernier cas, nous avons pris en compte une première

estimation de la variabilité de certaines données. Dans cette section nous allons plus loin, en remettant en cause l'aspect certain de plusieurs données et en ne considérant donc plus des durées dépendantes du temps.

Dans le chapitre 1, nous avons signalé que peu de travaux ont abordé les problèmes d'évacuation robuste. L'optimisation robuste (voir, par exemple, [Ben-Tal et al., 2009, Aissi et al., 2009, Goerigk, 2012]) est l'une des approches utilisées pour la résolution de problèmes où certaines données sont connues avec imprécision. D'un point de vu pratique, cette approche est très bien adaptée aux situations d'urgence. L'idée de base de cette approche est de calculer une solution robuste, réalisable pour n'importe quel scénario et performante en termes du critère à optimiser pour tous les scénarios. Dans la littérature récente, de nombreux concepts de robustesse et plusieurs mesures de performance ont été proposés. Pour un état de l'art sur l'optimisation robuste, le lecteur peut se référer à [Goerigk, 2012]. Notons que l'approche robuste est déterministe, contrairement à l'approche stochastique, car aucune loi de distribution des données n'est supposée.

Cette partie que nous allons détailler a été réalisée en collaboration avec Marc Goerigk (Optimization Research Group, Université de Kaiserslautern, Allemagne). Comme mentionné précédemment, nous souhaitons calculer un plan d'évacuation macroscopique avant que la catastrophe ait lieu. L'évacuation des personnes se fait par un ensemble de bus depuis la zone sinistrée vers les centres de secours. Nous utilisons, pour le calcul d'un ordonnancement des bus, le modèle simplifié proposé dans [Bish, 2011]. Aussi, pour ordonnancer les tournées des bus permettant la mise à jour des ordonnancements, de façon "simpliste", nous présentons un modèle bicritère en tenant compte des changements des données. Ces changements constituent un ensemble de scénarios possibles, appelé l'ensemble incertain. Pour deux scénarios différents, nous définissons une fonction qui permet de mesurer la différence entre les deux ordonnancements associés à ces deux scénarios.

5.1 Le problème d'évacuation par bus simplifié

Dans cette section, nous présentons le modèle basique (non-robuste) du problème d'évacuation par bus sur lequel nous appliquons la robustesse. Puisque les durées opératoires ne sont plus dépendantes du temps, le problème simplifié peut être vu comme un cas particulier du problème abordé par [Bish, 2011]. Il est important de noter que dés lors que les durées opératoires sont indépendantes du temps, les modèles mathématiques indexés sur le temps deviennent moins pertinent que le modèle mathématique indexé sur les "tours" ([Bish, 2011]) : des résultats expérimentaux préliminaires montrent même que ce dernier est le plus efficacement résolu par un solveur comme CPLEX. Nous ferons donc le choix d'une telle approche par la suite. Nous supposons que l'incertitude affecte les données suivantes : le nombre de personnes à évacuer, les durées des trajets et le nombre de bus disponibles. Nous notons par p_{ij} la durée du trajet entre un point de rassemblement i vers un centre de secours j, incluant le temps de retour au centre de la zone endommagée.

Après avoir présenté les hypothèses de notre modèle, nous présentons maintenant le modèle mathématique. Les variables de ce modèle sont indexées par des tours : un "tour " correspond à un trajet entre un point de rassemblement et un centre de secours, plus le retour vers le centre de la zone endommagée. Une borne supérieure triviale R du nombre de tours est $R = \sum_{i \in \mathcal{P}} e_i$. La modélisation basique du problème non-robuste est donc :

Données

- $\mathcal{P}$: l'ensemble des points de rassemblement.
- $\mathcal{S}$: l'ensemble des centres de secours.
- $\mathcal{B}$: l'ensemble des bus.
- R : une borne supérieure du nombre de tours.
- e_i : le nombre de personnes en attente sur un point de rassemblement i.
- cap_j : la capacité du centre de secours j.
- p_{ij} : la durée du trajet entre un point de rassemblement i est un centre de secours j.

Variables

- $\forall i \in \mathcal{P}, \forall j \in \mathcal{S}, \forall r \in \mathcal{R}, \forall b \in \mathcal{B}$;

$$x_{ijrb} = \begin{cases} 1 & \text{si un bus } b \text{ part depuis un point de rassemblement } i \text{ vers un centre } j \\ & \text{dans le tour } r, \\ 0 & \text{sinon.} \end{cases}$$

- T_{evac} : la date de fin d'évacuation.

$$(P) \quad \min T_{\text{evac}} \tag{2.18}$$

$$\sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{S}} p_{ij}x_{ijrb} \leq T_{\text{evac}} \qquad \forall b \in \mathcal{B} \tag{2.19}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{S}} x_{ijrb} \leq 1 \qquad \forall b \in \mathcal{B}, r \in \mathcal{R} \tag{2.20}$$

$$\sum_{j\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{b\in\mathcal{B}} x_{ijrb} \geq e_i \qquad \forall i \in \mathcal{P} \tag{2.21}$$

$$\sum_{i\in\mathcal{P}}\sum_{r\in\mathcal{R}}\sum_{b\in\mathcal{B}} x_{ijrb} \leq cap_j \qquad \forall j \in \mathcal{S} \tag{2.22}$$

$$x \in \{0,1\}^{P\times S\times R\times B} \tag{2.23}$$

$$T_{\text{evac}} \in \mathbb{N} \tag{2.24}$$

La fonction objectif (2.18) minimise la date de fin d'évacuation. Les contraintes (2.19) permettent de calculer les durées des trajets. Les contraintes (2.20) signifient qu'un bus ne peut faire qu'un seul trajet par tour. Les contraintes (2.21) assurent que toutes les personnes sont évacuées. Enfin, les contraintes (2.22) sont des contraintes de capacité des centres de secours.

La formulation P ci-dessus inclue le temps de retour du dernier bus vers le centre de la ville. Vu que le temps de retour vers la ville est petit par rapport au temps de sortie de la zone endommagée, l'erreur résultante dans la date de fin d'évacuation est relativement faible en raison des temps de trajets très asymétriques.

5.2 Définition de l'ensemble incertain

Une particularité du problème que nous abordons ici est le manque d'informations sur l'état du réseau de transport après la catastrophe. Dans ce qui suit, nous montrons

comment l'incertitude peut affecter les données du modèle basique P. Nous supposons que le planning est calculé avant la catastrophe en utilisant un scénario initial estimé par des experts. Ce scénario est appelé probable ou nominal. Évidemment, cette estimation ne sera pas exacte durant l'évacuation et nécessite une approche plus réaliste qui consiste à définir un ensemble de scénarios plausibles $\mathcal{U}$ qui seront considérés pour la planification de l'évacuation. Comme nous l'avons mentionné auparavant, nous supposons que les données incertaines sont :

- les durées des trajets p_{ij} : certains trajets peuvent prendre plus de temps que prévu à cause des congestions par exemple. Nous prenons $p_{ij} \in \{\underline{p}_{ij}, \ldots, \overline{p}_{ij}\}$ où les bornes $\underline{p}_{ij}$ et $\overline{p}_{ij}$ sont connues ;
- le nombre d'évacués e_i : la répartition des évacués peut être différente, variant en fonction des scénarios. Nous considérons que $e_i \in \{\underline{e}_i, \ldots, \overline{e}_i\}$ où $\underline{e}_i$ et $\overline{e}_i$ sont connues ;
- le nombre de bus B : certains bus peuvent être détruits en raison de la catastrophe, ou pour absence de certains chauffeurs de bus. B est supposé appartenir à l'ensemble $\{\underline{B}, \ldots, \overline{B}\}$ où $\underline{B}$ et $\overline{B}$ sont connus.

La génération de l'ensemble des données incertaines $\mathcal{U}$ nécessite l'élimination des cas triviaux et des cas extrêmes, comme par exemple le cas extrême : $B = \underline{B}$, $e_i = \overline{e}_i$ et $p_{ij} = \overline{p}_{ij}$ pour chaque $(i,j) \in \mathcal{P} \times \mathcal{S}$. Ceci nécessite l'ajout des contraintes suivantes :

$$\frac{T^{\text{exp}}_{\text{evac}}}{\frac{1}{P \cdot S}\sum_{i,j} p_{ij}} B \geq \sum_i e_i$$

où $\frac{1}{P \cdot S}\sum_{i,j} p_{ij}$ est le temps moyen d'un tour effectué par un bus, et $T^{\text{exp}}_{\text{evac}} \in \mathbb{R}$ une estimation de la borne supérieure de la date de fin d'évacuation fixée par le décideur. En d'autres termes, nous considérons uniquement les instances (les scénarios) où la date de fin d'évacuation

$$\frac{\sum_{i,j} p_{ij}}{P \cdot S} \cdot \frac{\sum_i e_i}{B}$$

est inférieure ou égale $T^{\text{exp}}_{\text{evac}}$. Ceci permet d'éliminer les cas extrêmes où une évacuation peut prendre beaucoup de temps. Notons aussi que, pour une valeur donnée $T^{\text{exp}}_{\text{evac}}$ nous pouvons déduire le nombre de bus nécessaires pour évacuer les personnes en $T^{\text{exp}}_{\text{evac}}$ unités de temps.

Par la suite, nous pouvons écrire l'ensemble d'incertitude $\mathcal{U}$ de la façon suivante :

$$\begin{aligned}\mathcal{U} = \Big\{(e,p,B) \ : e_i \in \{\underline{e}_i, \ldots, \overline{e}_i\}, p_{ij} \in \{\underline{p}_{ij}, \ldots, \overline{p}_{ij}\}, B \in \{\underline{B}, \ldots, \overline{B}\}, \\ \frac{T^{\text{exp}}_{\text{evac}}}{\frac{1}{P \cdot S}\sum_{i,j} p_{ij}} B \geq \sum_i e_i \Big\}. \end{aligned} \tag{2.25}$$

Nous notons par $P(\mathcal{U})$ le problème d'optimisation robuste où l'ensemble d'incertitude est $\mathcal{U}$. Par ailleurs, nous supposons l'existence d'un scénario *nominal* $(\hat{e}, \hat{p}, \hat{B}) \in \mathcal{U}$ qui représente le scénario le plus probable. En d'autres termes, ce scénario est celui considéré dans le cas du problème sans incertitude (voir [Jenkins, 2000]). Un scénario est noté par $\xi = (e^\xi, p^\xi, B^\xi)$ et $\hat{\xi}$ dénotera le scénario nominal.

5.3 Modèle robuste bicritère basé sur la réparation

Plusieurs concepts ont été proposés dans la littérature pour définir la robustesse d'une solution. Par exemple, la *robustesse stricte*, qui est le point de départ de l'optimisation robuste ([Ben-Tal and Nemirovski, 2000]). Elle a pour but de calculer une solution vérifiant les propriétés suivantes : a) réalisable pour n'importe quel scénario, b) optimise la valeur du critère considéré dans le pire des cas. Dans notre problème d'évacuation on cherche, d'une part, à trouver une solution qui soit réalisable pour tous les scénarios possibles. D'autres part, ces derniers incluent le scénario extrême suivant : plusieurs trajets sont nécessaires pour l'évacuation des personnes, le nombre de bus est significativement petit pour faire l'évacuation et chaque trajet est trop lent. Pour une valeur de $T_{\text{evac}}^{\text{exp}}$ suffisamment grande, le problème résultant est équivalent au problème P en utilisant les paramètres $\overline{e}$, $\overline{p}$, et $\underline{B}$. Une telle approche est inapproprié à notre problème. Plusieurs concepts récents ont été proposés où la restriction sur la faisabilité de la solution est relaxée. Un de ces concepts est l'approche robuste basée sur la réparation (voir [Liebchen et al., 2009, Stiller, 2008, Erera et al., 2009]), qui est une approche à deux étapes : nous considérons un ensemble d'algorithmes de réparation qui prennent en entrée une solution en la modifiant de sorte quelle soit réalisable pour un scénario donné. Donc, nous cherchons une solution qui : a) nécessite des changements modérés pour quelle soit réalisable, et b) sa performance est bonne dans le pire des cas (la valeur du critère dans le pire des cas est acceptable).

Dans ce qui suit, nous expliquons en détails l'application de cette approche. À cette dernière, nous introduisons pour chaque scénario $\xi \in \mathcal{U}$ un ensemble de variables de deuxième niveau x^ξ et une mesure de similitude entre la solution nominale et les solutions des scénarios. Diverses définitions ont été introduites dans la littérature pour mesurer la différence entre deux solutions. Dans la suite, nous nous concentrons sur *la distance de réparation* :

$$\begin{aligned}\Delta : \{0,1\}^{PSRB} \times \{0,1\}^{PSRB} &\to \mathbb{N} \\ (x, x^\xi) &\mapsto \sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{S}} \sum_{r\in\mathcal{R}} \sum_{b\in\mathcal{B}} \left| x_{ijrb} - x^\xi_{ijrb} \right|.\end{aligned}$$

Avec cette distance, nous comptons combien de fois un bus fait un voyage différent dans un tour. Maintenant, expliquons une autre raison pour laquelle la modélisation indexée sur le temps n'a pas été utilisée dans cette partie. Supposons que nous ayons une solution nominale et une solution d'un autre scénario, si les plannings des bus dans ces deux solutions sont exactement les mêmes, à l'exception que tous les bus de la solution de ce scénario commencent leurs trajets à des dates différentes de la solution nominale, alors la solution de ce scénario est considérée comme différente de la solution nominale.

Le but est de trouver une solution de premier niveau x et des solutions de deuxième niveau x^ξ qui minimisent la valeur $\Delta(x) := \max_{\xi\in\mathcal{U}} \Delta(x, x^\xi)$. En réalité, ceci revient à trouver un planning facilement modifiable pour les bus : pour n'importe quel scénario le chauffeur de bus doit avoir un planning similaire pour n'importe quel scénario. La solution du premier niveau x est obtenue en résolvant le problème P avec le scénario nominal $(\hat{l}, \hat{d}, \hat{B})$ où T_{evac}^{nom} est la date de fin de cette solution. Ce qui revient, d'une part, à minimiser la distance de réparation Δ entre la solution du premier niveau et les solutions

de deuxième niveau. D'autre part, minimiser à la fois la date de fin de la solution nominale $T^{\text{nom}}_{\text{evac}}(x)$ et la date de fin d'évacuation dans le pire des cas sur tous les scénarios, notée $T^{wc}_{\text{evac}}(x)$. Cette dernière doit être assez petite pour permettre une évacuation dans un horizon temporel acceptable pour le décideur. Donc le problème est multicritère où nous cherchons à minimiser à la fois Δ, $T^{\text{nom}}_{\text{evac}}(x)$ et $T^{wc}_{\text{evac}}(x)$.

D'un point de vue pratique, il est raisonnable de coupler les deux critères $T^{\text{nom}}_{\text{evac}}$ et $T^{\text{wc}}_{\text{evac}}$, en exprimant un critère en fonction de l'autre. Par exemple, le décideur accepte toute solution dont la valeur de sa fonction objectif est inférieure ou égale au double de la date de fin d'évacuation nominale. Ceci nous permet de considérer un seul critère de date de fin d'évacuation ce qui simplifie par la suite l'énumération du front de Pareto. Comme les deux critères $\Delta(x)$ et $T^{wc}_{\text{evac}}(x)$ sont conflictuels, nous nous sommes intéressés à l'énumération du front de Pareto. Une solution x est une solution du front de Pareto s'il n'existe pas une autre solution x' tel que $\Delta(x') \leq \Delta(x)$ et $T^{wc}_{\text{evac}}(x') \leq T^{wc}_{\text{evac}}(x)$ avec au moins une inégalité stricte. Plusieurs méthodes existent dans la littérature pour calculer ces solutions (voir [T'kindt and Billaut, 2006]). Ici, Nous utilisons la méthode ε-contrainte et considérons le problème monocritère avec une contrainte additionnelle qui assure un seuil désiré pour l'autre fonction objectif. D'un point de vue pratique, il est préférable d'assurer une certaine qualité de la date de fin d'évacuation et de trouver sous cette dernière condition les plannings des bus qui minimisent la distance de réparation. Cela se justifie par deux raisons : la première, est que la date de fin d'évacuation doit être la première priorité, et une date limite ne doit pas être dépassée. Deuxièmement, la valeur de la distance de réparation est abstraite et en pratique la fixation de sa qualité désirée est difficile.

Dans ce qui suit, nous notons ce problème par $RP(\mathcal{U})$, et nous introduisons sa formulation mathématique basée sur la formulation du problème P (cf. section 5.1).

Données

- T^{nom}_{evac} : la date de fin d'évacuation du scénario nominal.
- T^{wc}_{evac} : la borne imposée sur la date de fin dans le pire des cas sur tous les scénarios.

Variables

- $\forall i \in \mathcal{P}, \forall j \in \mathcal{S}, \forall r \in \mathcal{R}, \forall b \in \mathcal{B}$;

$$x_{ijrb} = \begin{cases} 1 & \text{si un bus } b \text{ part depuis un point de rassemblement } i \text{ vers un centre } j \\ & \text{dans le tour } r, \text{ pour la solution du premier niveau,} \\ 0 & \text{sinon.} \end{cases}$$

- $\forall i \in \mathcal{P}, \forall j \in \mathcal{S}, \forall r \in \mathcal{R}, \forall b \in \mathcal{B}^{\xi}$;

$$x^{\xi}_{ijrb} = \begin{cases} 1 & \text{si un bus } b \text{ part depuis un point de rassemblement } i \text{ vers un centre } j \\ & \text{dans le tour } r, \text{ pour un scénario } \xi \in \mathcal{U}, \\ 0 & \text{sinon.} \end{cases}$$

- δ_b et δ^{ξ}_b : déterminent respectivement, si un bus $b \in \overline{B}$ est utilisé dans la solution du premier niveau et dans la solution du deuxième niveau. Ces variables sont nécessaires vu que le nombre de bus est différent pour chaque scénario.

- Δ : la distance de réparation.
- y^{ξ}_{ijrb} : déterminent si les variables x_{ijrb} et x^{ξ}_{ijrb} sont différentes, cela permet de déterminer la valeur de variable Δ
- z_{ijrb} : déterminent que le bus utilisé dans les trajets x_{ijrb} transportent des évacués, ou bien il est vide.
- z^{ξ}_{ijrb} : déterminent que le bus utilisé dans les trajets x^{ξ}_{ijrb} transporte des évacués, ou bien est vide. Cela peut se produire à cause de l'usage des trajets redondants afin de réduire le nombre de modifications nécessaires pour que la solution du premier niveau soit réalisable. Cependant, ces trajets redondants ne doivent pas affecter les capacités des centres de secours.

Le modèle mathématique est le suivant :

$$\min \Delta = \max_{\xi \in \mathcal{U}} \Big(\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{b \in \overline{\mathcal{B}}} y^{\xi}_{ijrb} \Big) \tag{2.26}$$

$$\sum_{r \in \mathcal{R}} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} \hat{p}_{ij} x_{ijrb} \leq T^{\text{nom}}_{\text{evac}} \qquad \forall b \in \overline{\mathcal{B}} \tag{2.27}$$

$$\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} x_{ijrb} \leq 1 \qquad \forall b \in \overline{\mathcal{B}}, r \in \mathcal{R} \tag{2.28}$$

$$\sum_{j \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{b \in \overline{\mathcal{B}}} z_{ijrb} \geq \hat{e}_i \qquad \forall i \in \mathcal{P} \tag{2.29}$$

$$\sum_{i \in \mathcal{P}} \sum_{r \in \mathcal{R}} \sum_{b \in \overline{\mathcal{B}}} z_{ijrb} \leq cap_j \qquad \forall j \in \mathcal{S} \tag{2.30}$$

$$\sum_{r \in \mathcal{R}} \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} p^{\xi}_{ij} x^{\xi}_{ijrb} \leq T^{\text{wc}}_{\text{evac}} \qquad \forall b \in \overline{\mathcal{B}}, \xi \in \mathcal{U} \tag{2.31}$$

$$\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} x^{\xi}_{ijrb} \leq 1 \qquad \forall b \in \overline{\mathcal{B}}, r \in \mathcal{R}, \xi \in \mathcal{U} \tag{2.32}$$

$$\sum_{j \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{b \in \overline{\mathcal{B}}} z^{\xi}_{ijrb} \geq e^{\xi}_i \qquad \forall i \in \mathcal{P}, \xi \in \mathcal{U} \tag{2.33}$$

$$\sum_{i \in \mathcal{P}} \sum_{r \in \mathcal{R}} \sum_{b \in \overline{\mathcal{B}}} z^{\xi}_{ijrb} \leq cap_j \qquad \forall j \in \mathcal{S}, \xi \in \mathcal{U} \tag{2.34}$$

$$x_{ijrb} \leq \delta_b \qquad \forall i \in \mathcal{P}, j \in \mathcal{S}, r \in \mathcal{R}, b \in \overline{\mathcal{B}} \tag{2.35}$$

$$x^{\xi}_{ijrb} \leq \delta^{\xi}_b \qquad \forall i \in \mathcal{P}, j \in \mathcal{S}, r \in \mathcal{R}, b \in \overline{\mathcal{B}}, \xi \in \mathcal{U} \tag{2.36}$$

$$\sum_{b \in \overline{\mathcal{B}}} \delta_b \leq \hat{B} \tag{2.37}$$

$$\sum_{b \in \overline{\mathcal{B}}} \delta^{\xi}_b \leq B^{\xi} \qquad \forall \xi \in \mathcal{U} \tag{2.38}$$

$$-y^{\xi}_{ijrb} \leq x_{ijrb} - x^{\xi}_{ijrb} \leq y^{\xi}_{ijrb} \qquad \forall i \in \mathcal{P}, j \in \mathcal{S}, r \in \mathcal{R}, b \in \overline{\mathcal{B}}, \xi \in \mathcal{U} \tag{2.39}$$

$$z_{ijrb} \leq x_{ijrb} \qquad \forall i \in \mathcal{P}, j \in \mathcal{S}, r \in \mathcal{R}, b \in \overline{\mathcal{B}} \tag{2.40}$$

$$z^{\xi}_{ijrb} \leq x^{\xi}_{ijrb} \qquad \forall i\in\mathcal{P}, j\in\mathcal{S}, r\in\mathcal{R}, b\in\overline{\mathcal{B}}, \xi\in\mathcal{U} \qquad (2.41)$$

$$x, z, x^{\xi}, z^{\xi}, y^{\xi} \in \mathbb{B}^{ST\overline{B}R} \qquad \forall \xi\in\mathcal{U} \qquad (2.42)$$

$$\delta, \delta^{\xi} \in \mathbb{B}^{\overline{B}} \qquad \forall \xi\in\mathcal{U} \qquad (2.43)$$

$$\Delta \in \mathbb{N} \qquad (2.44)$$

Les constraints (2.27)-(2.30) sont les contraintes du problème P imposant la faisabilité de la solution du premier niveau. Les contraintes (2.31)-(2.34) ont le même effet sur les variables du deuxième niveau. Nous assurons avec les contraintes (2.35)-(2.38) que le nombre de bus disponibles pour chaque scénario est respecté. Notons que les contraintes (2.39) couplent les variables du premier et du deuxième niveau. Ces dernières contraintes imposent aux variables y^{ξ}_{ijrb} d'être égales à 1, si la solution du premier niveau et la solution du deuxième niveau sont différentes. Enfin, si le planning d'un bus à été totalement établi alors les contraintes (2.40) et (2.41) assurent que les trajets avec des bus vides sont possibles.

Notons que, comme $\mathcal{U}$ est un ensemble fini, l'ensemble des solutions de la formulation ci-dessus est fini. Cependant, $|\mathcal{U}|$ est très grand et un solveur mathématique ne peut pas résoudre $RP(\mathcal{U})$ pour des instances de taille réelle. Cela, nous conduit à proposer dans la section suivante une approche alternative.

5.4 Une approche itérative

Tout au long de cette section, nous simplifions la notation du problème RP et nous notons respectivement par x et x^{ξ} les solutions du premier niveau et les solutions du second niveau. Pour la simplification de la présentation, nous négligeons aussi les variables z et z^{ξ}.

5.4.1 Un algorithme basé sur l'oracle

Dans cette section, nous supposons que tous les scénarios $\xi \in \mathcal{U}$ sont réalisables. Donc, nous définissons un ensemble de solutions réalisables pour un scénario ξ comme suit :

$$\mathcal{F}(\xi) = \Big\{ x^{\xi} \in \mathbb{B}^{PS\overline{B}R} : \sum_{r\in\mathcal{R}}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{S}} d^{\xi}_{ij} x^{\xi}_{ijrb} \leq T^{\text{wc}}_{\text{evac}} \qquad \forall b\in\overline{\mathcal{B}} \qquad (2.45)$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{S}} x^{\xi}_{ijrb} \leq 1 \qquad \forall b\in\overline{\mathcal{B}}, r\in\mathcal{R} \qquad (2.46)$$

$$\sum_{j\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{b\in\overline{\mathcal{B}}} x^{\xi}_{ijrb} \geq l^{\xi}_{i} \qquad \forall i\in\mathcal{P} \qquad (2.47)$$

$$\sum_{i\in\mathcal{P}}\sum_{r\in\mathcal{R}}\sum_{b\in\overline{\mathcal{B}}} x^{\xi}_{ijrb} \leq cap_j \qquad \forall j\in\mathcal{S} \qquad (2.48)$$

$$\exists \delta^{\xi} : \; x^{\xi}_{ijrb} \leq \delta^{\xi}_{b} \qquad \forall b\in\overline{\mathcal{B}} \qquad (2.49)$$

$$\sum_{b\in\overline{\mathcal{B}}} \delta^{\xi}_{b} \leq B^{\xi} \qquad \Big\} \qquad (2.50)$$

et nous supposons que l'ensemble de ces solutions n'est pas vide. Autrement dit, il existe des solutions qui respectent la borne imposée sur la date de fin d'évacuation et les capacités des centres. Cet ensemble n'est pas vide si la valeur donnée de $T_{\text{evac}}^{\text{nom}}$ est suffisamment grande pour permettre la faisabilité de la solution du premier niveau.

Comme l'ensemble d'incertitude est très grand, la résolution directe du problème $RP(\mathcal{U})$ est impossible puisque tous les scénarios sont considérés simultanément. Pour cette raison, nous présentons une procédure qui, pour un sous-ensemble de scénarios, en génère séquentiellement de nouveaux et les ajoute au problème. Nous considérons un sous-ensemble de scénarios $\mathcal{U}' \subseteq \mathcal{U}$, et soit $(x, x^\xi)_{\xi \in \mathcal{U}'}$ la solution de $RP(\mathcal{U}')$.

Il est évident que le fait d'ajouter un scénario à chaque itération peut conduire à choisir des scénarios où la solution du premier niveau ne sera pas changée (des scénarios faciles pour la solution courante). Pour cette raison, L'algorithme se compose de plusieurs parties. Nous définissons d'abord des scénarios dans le pire des cas et nous montrons dans ce cas l'existence d'une solution optimale. Ensuite, nous considérons deux sous-problèmes :

1. le problème de la détermination de la meilleure distance de réparation pour la solution courante,
2. le problème de calcul des scénarios dans le pire des cas.

Initialement, l'ensemble $\mathcal{U}' = \emptyset$, ce qui revient à résoudre le problème original d'évacuation défini dans la section 5.1 où la date de fin d'évacuation est constante. La résolution de ce problème donne une première solution $(x^{(0)}, x^{(0),\xi})_{\xi \in \mathcal{U}'}$. Ensuite, nous ajoutons itérativement des scénarios à l'aide d'un *oracle* défini comme suit :

Définition 1 *Un* oracle Ω *est une fonction qui prend en entrée un ensemble d'incertitude* $\mathcal{U}' \subseteq \mathcal{U}$ *et une solution* $(x, x^\xi)_{\xi \in \mathcal{U}'}$ *et retourne un scénario* $\xi = \Omega((x, x^\xi)_{\xi \in \mathcal{U}'}) \in \mathcal{U} \setminus \mathcal{U}'$ *ou "Non" si la génération d'un scénario n'est pas possible.*

Nous supposons maintenant qu'il existe un oracle disponible qui génère un scénario dans le pire des cas $\xi \in \mathcal{U}$ pour une solution courante $(x^{(k)}, x^{(k),\xi})_{\xi \in \mathcal{U}^{(k)}}$. Nous ajoutons ce scénario à $\mathcal{U}^{(k)}$, et nous obtenons un nouveau ensemble d'incertitudes $\mathcal{U}^{(k+1)} = \mathcal{U}^{(k)} \cup \{\xi\}$. En résolvant $RP(\mathcal{U}^{(k+1)})$, nous obtenons une nouvelle solution $(x^{(k+1)}, x^{(k+1),\xi})_{\xi \in \mathcal{U}^{(k+1)}}$. On réitère ce procédé jusqu'à ce qu'aucun scénario ne soit ajouté. L'Algorithme 6 récapitule cette approche.

Notons que la terminaison de l'Algorithme 6 est assurée car l'ensemble des scénarios $\mathcal{U}$ est fini. Pour montrer la convergence vers une solution optimale, nous devons raffiner le concept d'oracle pour produire des scénarios qui sont "mauvais" pour la solution courante. En d'autres termes, ces scénarios nécessitent des modifications de la solution courante et conduisent à une augmentation de la valeur de la distance de réparation Δ.

Définition 2 *Soit la qualité désirée de la distance de réparation* Δ*, supposée donnée. Nous appelons un oracle* Ω *un oracle de* pire cas*, si pour tout* $\mathcal{U}' \subseteq \mathcal{U}$ *et tout* $(x, x^\xi)_{\xi \in \mathcal{U}'}$*, les conditions suivantes sont vérifiées pour la génération d'un scénario* ξ *: Il n'existe pas de solution* $x^\xi \in \mathcal{F}(\xi)$ *pour le scénario* ξ *avec* $\Delta(x, x^\xi) \leq \Delta$*. L'oracle retourne "Non" si un tel scénario n'existe pas.*

Algorithme 6

Entrées: Une instance du problème $RP(\mathcal{U})$, et un oracle Ω
Sorties: Une solution $(x, x^{\xi})_{\xi \in \mathcal{U}'}$ pour un ensemble de $\mathcal{U}' \subseteq \mathcal{U}$

1: Initialiser $k = 0$, et $\mathcal{U}^{(0)} = \emptyset$
2: Résoudre $RP(\mathcal{U}^{(k)})$. Soit $(x^{(k)}, x^{(k),\xi})_{\xi \in \mathcal{U}^{(k)}}$ la solution calculée, et $\Delta^{(k)}$ la distance de réparation résultante
3: $k = k + 1$.
4: **si** $\Omega((x^{(k-1)}, x^{(k-1),\xi})_{\xi \in \mathcal{U}^{(k-1)}}) \neq$ "Non" **alors**
5: $\mathcal{U}^{(k)} = \mathcal{U}^{(k-1)} \cup \Omega((x^{(k-1)}, x^{(k-1),\xi})_{\xi \in \mathcal{U}^{(k-1)}})$.
6: Résoudre $RP(\mathcal{U}^{(k)})$. Soit $(x^{(k)}, x^{(k),\xi})_{\xi \in \mathcal{U}^{(k)}}$ la solution calculée, et $\Delta^{(k)}$la distance de réparation résultante
7: Allez à l'étape 3
8: **sinon**
9: **Retourner** $(x^{(k-1)}, x^{(k-1),\xi})_{\xi \in \mathcal{U}^{(k-1)}}$
10: **finsi**

Dans ce qui suit, nous montrons que si l'oracle est un oracle de pire cas, la solution obtenue par l'Algorithme 6 peut être étendue à une solution optimale complète du $RP(\mathcal{U})$ sans l'augmentation de la distance de réparation.

Théorème 1 *Si Ω est oracle de pire cas, avec la garantie de respect de la valeur $\Delta^{(k-1)}$ des étapes 4 et 5, l'Algorithme 6 produit une solution qui peut être étendue à une solution optimale du $RP(\mathcal{U})$ sans l'augmentation de la valeur de la distance de réparation.*

Preuve 1 *Soit Ω est un oracle de pire cas qui respecte $\Delta^{(k-1)}$ à chaque itération, et soit $(x^{(k^*)}, x^{(k^*),\xi})_{\xi \in \mathcal{U}^{(k^*)}}$ la solution calculée par l'Algorithme 6. Nous montrons qu'une extension de la solution $(x^{(k^*)}, x^{(k^*),\xi})_{\xi \in \mathcal{U}}$ à l'ensemble des scénarios $\mathcal{U}$, cela ne conduit pas à une augmentation de valeur de sa fonction objectif,*

$$\max_{\xi \in \mathcal{U}^{(k^*)}} \Delta(x^{(k^*)}, x^{(k^*),\xi}) = \max_{\xi \in \mathcal{U}} \Delta(x^{(k^*)}, x^{(k^*),\xi})$$

Nous supposons qu'il existe $\xi \in \mathcal{U} \setminus \mathcal{U}^{(k^*)}$ avec $\min_{x^{\xi} \in \mathcal{F}(\xi)} \Delta(x, x^{\xi}) \geq \Delta^{(k^*)}$, où le minimum est calculé sur toutes les solutions réalisables d'un scénario ξ. Ensuite, par définition de Ω qui est oracle de pire cas, l'Algorithme 6 n'aurait pas terminé après l'étape 4.

Puisque $RP(\mathcal{U}')$ est une relaxation du problème $RP(\mathcal{U})$ alors la valeur optimale de la fonction objectif du premier problème est inférieure ou égale à la valeur de la fonction objectif du dernier problème.

Le point crucial de l'Algorithme 6 est de définir de manière efficace l'oracle du pire cas. Dans ce qui suit, nous allons présenté comment cela peut se faire.

5.4.2 Calcul d'un oracle du pire des cas

Pour construire un scénario de pire cas, nous devons trouver un scénario ξ pour une solution du premier niveau donnée x pour laquelle toutes les solutions ont une distance

de réparation supérieure à la meilleure distance courante. Une condition suffisante pour déterminer un tel scénario est donnée par la solution optimale du problème suivant que nous appelons WC :

$$\max_{\xi \in \mathcal{U}} \min_{x^\xi \in \mathcal{F}(\xi)} \Delta(x, x^\xi)$$

Clairement ce problème est un problème non-linéaire (où la non-linéarité est présente dans l'ensemble d'incertitude $\mathcal{U}$). Dans ce qui suit, nous proposons une méthode itérative pour résoudre ce sous-problème de l'Algorithme 6. Notons que nous essayons d'énumérer tous les scénarios ξ, mais en terme de temps d'exécution, ce choix n'est pas possible pour un ensemble d'incertitude $\mathcal{U}$ assez grand. Pour cette raison, nous considérons l'approche suivante : pour un scénario ξ donné, nous calculons une solution x^ξ qui minimise la distance de réparation ; et pour un ensemble de solutions candidates pré-calculées $\{x^\xi, \xi \in \mathcal{U}'\}$, nous vérifions s'il existe un scénario ξ' qui soit non réalisable pour toutes les solutions candidates. On répète toutes ces étapes jusqu'à ce qu'on trouve un scénario de pire cas, ou qu'il soit montré que cela est impossible. Comme nous allons voir, une telle approche calcule une solution optimale pour WC.

Maintenant, nous décrivons en détails cette approche itérative. D'abord, nous supposons qu'il existe un $\xi \in \mathcal{U}$ et une solution du premier niveau x. Ensuite, nous déterminons la distance minimale de réparation en résolvant le problème suivant, que nous dénotons par WC_1 :

$$\min \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} \sum_{r \in \mathcal{R}} \sum_{b \in \overline{\mathcal{B}}} y^\xi_{ijrb} \tag{2.51}$$

$$x^\xi \in \mathcal{F}(\xi) \tag{2.52}$$

$$-y^\xi_{ijrb} \leq x_{ijrb} - x^\xi_{ijrb} \leq y^\xi_{ijrb} \qquad \forall i \in \mathcal{P}, j \in \mathcal{S}, r \in \mathcal{R}, b \in \overline{\mathcal{B}} \tag{2.53}$$

$$x^\xi, y^\xi \in \mathbb{B}^{PS\overline{B}R} \tag{2.54}$$

Comme précédemment, nous utilisons les variables x^ξ_{ijrb} pour modéliser la solution réparée, les variables y^ξ_{ijrb} pour calculer la distance de réparation, et les variables δ^ξ_b pour compter le nombre de bus. Notons que la variable additionnelle Δ n'est pas nécessaire ici, parce qu'un seul scénario est considéré, et la distance de réparation est exprimée directement dans la fonction objectif (2.51). Les contraintes (2.52) assurent la faisabilité de la solution réparée $x^\xi \in \mathcal{F}(\xi)$. Enfin, les contraintes (2.53) sont utilisées pour déterminer la différence entre la solution réparée (modifiée) et la solution du premier niveau. Notons que WC_1 est $\mathcal{NP}$-difficile puisqu'il contient le sous-problème P.

Maintenant, nous supposons que l'ensemble des solutions candidates $\{x^\xi, \xi \in \mathcal{U}'\}$ est fixé, avec une distance de réparation inférieure ou égale à la valeur donnée de Δ. Dans le cas où il n'existe pas un scénario $\xi \in \mathcal{U}$ pour lequel toutes ces solutions seront irréalisables, nous pouvons conclure qu'il n'existe pas une solution pour l'oracle du pire cas, donc nous retournons "Non". Dans le cas contraire, si ce scénario ξ existe, nous l'utilisons pour ajouter une nouvelle solution à l'ensemble des solutions candidates, nous résolvons WC_1 et nous vérifions sa distance de réparation. Si une telle solution n'existe pas, nous pouvons retourner ξ comme un scénario dans le pire des cas.

Soit un ensemble donné $x^{\xi_k}, k = \{1, \ldots, N\}$, de solutions candidates. Nous cherchons à trouver un scénario $\xi \in \mathcal{U}$ tel que toutes les solutions x^{ξ_k} sont non réalisables pour ce scénario, $x^{\xi_k} \notin \mathcal{F}(\xi)$. Nous dénotons ce problème par WC_2. Nous montrons dans le théorème suivant qu'il est $\mathcal{NP}$-difficile.

Théorème 2 *WC_2 est $\mathcal{NP}$-difficile pour l'ensemble incertain de la forme (2.25), même si $\underline{B} = \overline{B}$ et $\underline{e}_i = \overline{e}_i$ pour tout $i \in \mathcal{P}$.*

Preuve 2 *Nous montrons ici que WC_2 est $\mathcal{NP}$-difficile en utilisant la réduction polynomiale depuis le problème* de couverture par ensembles *vers WC_2. La définition du* problème de couverture par ensembles *est la suivante :*

> Problème de couverture par ensembles
> *Entrée : un ensemble $\mathcal{S} = [s]$ et des sous-ensembles finis $\mathcal{M}_i \subseteq \mathcal{S}$, $\mathcal{M}_i \neq \emptyset$, $\forall i \in [m]$. Un entier K.*
> *Question : existe-il un ensemble d'indices $\mathcal{I} \subseteq [m]$ tel que $\cup_{i \in \mathcal{I}} \mathcal{M}_i = \mathcal{S}$ et $|\mathcal{I}| \leq K$?*

Tout d'abord, nous construisons une instance de WC_2 à partir d'une instance du problème de couverture par ensembles. *Nous supposons l'instance suivante du problème WC_2, $P = 1$ and $S = m + 1$. De plus, nous définissons l'ensemble d'incertitudes suivant :*

$$\mathcal{U} = \Big\{(e, p, B) \ : e_1 = m(m+1),\ B = m,\ p_{ij} \in \{0,1\}\ \forall j \in [m],\ p_{i(m+1)} = 0,\ \frac{T^{exp}_{evac} \cdot P \cdot S \cdot B}{e_1} \geq \sum_{i,j} p_{ij}\Big\},$$

où $T^{exp}_{evac} = K$. Nous substituons les valeurs de l'ensemble $\mathcal{U}$ dans l'inégalité de cet ensemble, et nous avons $\sum_{ij} p_{ij} \leq K$. Nous posons $T^{wc}_{evac} = 1$, et supposons que les capacités des centres de secours sont suffisamment grandes. Nous les initialisons de la façon suivante : $cap_j = m(m+1)$ pour tout $j \in \mathcal{S}$.

Pour chaque $j \in \mathcal{P}$, nous générons pour le problème WC_2 un planning de bus x^{ξ_j} comme suit : pour chaque $j \in \mathcal{M}_i$, nous créons deux trajets de bus identiques avec un bus $i \in \mathcal{B}$ à partir d'un point de rassemblement j vers un centre de secours $i \in [S]$. De plus, un bus fait le trajet $m + 1$ depuis le point de rassemblement vers le centre de secours $m + 1$.

Tous ces plannings de bus sont faisables en respectant le nombre de bus utilisés (qui est au maximum m), et le nombre de personnes à évacuer qui est au minimum $m(m+1)$.

La transformation du WC_2 en une instance du problème de couverture par ensembles *est évidemment polynomiale. Montrons maintenant que c'est une réduction, c'est-à-dire que :*

1. *s'il existe un scénario $\xi \in \mathcal{U}$ tel que toutes les solutions x^{ξ_j} sont non réalisables, alors la réponse au* problème de couverture par ensembles *est oui,*
2. *s'il n'existe pas une solution pour WC_2 alors le* problème de couverture par ensembles *n'admet pas une solution.*

Commençons par la première partie. Il existe un scénario $\xi \in \mathcal{U}$ tel que toutes les x^{ξ_j} sont infaisables, cela est possible si est seulement si, les contraintes (2.45) *sur la date de fin d'évacuation maximale ne sont pas respectées avec $T^{wc}_{evac} = 1$. Nous supposons qu'un tel scénario existe, alors pour chaque solution x^{ξ_j}, il existe des trajets de bus i avec $\sum_i p_{1,i} = 2 > 1$. À partir des solutions x^{ξ_j}, nous pouvons construire une solution du* PROBLÈME DE COUVERTURE PAR ENSEMBLES *comme suit : nous mettons un élément $i \in \mathcal{I}$, si est seulement si il existe une solution x^{ξ_j} pour laquelle $\sum_i p_{1i} = 2$. Alors, nous avons $\cup_{i\in\mathcal{I}}\mathcal{M}_i = \mathcal{S}$ et $|\mathcal{I}| \leq K$.*
La deuxième partie de cette démonstration est une conséquence directe de la première. S'il n'existe pas un scénario qui rend toutes les solutions x^{ξ_j} non réalisables, alors le PROBLÈME DE COUVERTURE PAR ENSEMBLES *est infaisable. Puisque toutes les étapes de la démonstration nécessite un temps polynomial, alors WC_2 est $\mathcal{NP}$-difficile.*

Le problème WC_2 peut être résolu en utilisant le modèle mathématique suivant, où les variables e et p sont utilisées pour la linéarisation de l'ensemble $\mathcal{U}$.

$$\max\ z \tag{2.55}$$

$$\underline{e}_i \leq e_i \leq \overline{e}_i \qquad \forall i\in\mathcal{P} \tag{2.56}$$

$$\underline{p}_{ij} \leq p_{ij} \leq \overline{p}_{ij} \qquad \forall i\in\mathcal{P}, j\in\mathcal{S} \tag{2.57}$$

$$\underline{B} \leq B \leq \overline{B} \tag{2.58}$$

$$z \leq s^{\xi_k} \qquad \forall k\in\mathcal{N} \tag{2.59}$$

$$s^{\xi_k} \leq e_i - \sum_{j\in\mathcal{S}}\sum_{r\in\mathcal{R}}\sum_{b\in\overline{\mathcal{B}}} x^{\xi_k}_{ijrb} + M(1 - c_i^{\xi_k,1}) \qquad \forall i\in\mathcal{P}, k\in\mathcal{N} \tag{2.60}$$

$$s^{\xi_k} \leq \sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{S}}\sum_{r\in\mathcal{R}} p_{ij}x^{\xi_k}_{ijrb} - T^{\text{wc}}_{\text{evac}} + M(1 - c_b^{\xi_k,2}) \qquad \forall b\in\overline{B}, k\in\mathcal{N} \tag{2.61}$$

$$s^{\xi_k} \leq B(k) - B + M(1 - c^{\xi_k,3}) \qquad \forall k\in\mathcal{N} \tag{2.62}$$

$$\sum_{i\in\mathcal{P}} c_i^{\xi_k,1} + \sum_{b\in\overline{\mathcal{B}}} c_b^{\xi_k,2} + c^{\xi_k,3} = 1 \qquad \forall k\in\mathcal{N} \tag{2.63}$$

$$\gamma_{ii'jqq'} \geq \alpha_{iq} + \beta_{i'jq'} - 1 \qquad \forall i,i'\in\mathcal{P}, j\in\mathcal{S}, q\in[Q^i], q'\in[Q^{i'j}] \tag{2.64}$$

$$\sum_{q\in[Q^i]} 2^q\alpha_{iq} = e_i \qquad \forall i\in\mathcal{P} \tag{2.65}$$

$$\sum_{q\in[Q^{ij}]} 2^q\beta_{ijq} = p_{ij} \qquad \forall i\in\mathcal{P}, j\in\mathcal{S} \tag{2.66}$$

$$STT^{\text{exp}}_{\text{evac}} \cdot B \geq \sum_{i\in\mathcal{P}}\sum_{i'\in\mathcal{P}}\sum_{j\in\mathcal{S}}\sum_{q\in[Q^i]}\sum_{q'\in[Q^{i'j}]} 2^{q+q'}\gamma_{ii'jqq'} \tag{2.67}$$

$$c^{\xi,1}, c^{\xi,2}, c^{\xi,3} \in \{0,1\} \tag{2.68}$$

$$\gamma, \alpha, \beta \in \{0,1\} \tag{2.69}$$

$$s^{\xi}, e, p, B \in \mathbb{N} \tag{2.70}$$

où

$$B(k) := \left|\left\{b \in \overline{\mathcal{B}} : \exists i \in \mathcal{P}, j \in \mathcal{S}, r \in \mathcal{R} : x^{\xi}_{ijrb} > 0\right\}\right|$$

dénote le nombre de bus utilisés dans la solution x^{ξ}_{k}, et

$$Q^{i} = \lceil \log_2(\overline{e}_i), \rceil \qquad\qquad Q^{ij} = \lceil \log_2(\overline{p}_{ij}) \rceil$$

sont utilisées pour la reformulation des contraintes non-linéaires.

Maintenant, nous expliquons cette formulation avec plus de détails. Les variables e, p, et B modélisent le scénario que nous essayons de déterminer. Les variables s^{ξ_k} modélisent l'infaillibilité d'une solution x^{ξ_k}. Ces variables auront une valeur supérieure à zéro, si au moins une contrainte qui contient la variable x^{ξ_k} est violée, $x^{\xi_k} \notin \mathcal{F}(e, p, B)$. Les variables booléennes $c^{\xi_k,1}$, $c^{\xi_k,2}$, et $c^{\xi_k,3}$ sont utilisées pour déterminer les contraintes violées. Enfin, les variables α, β, et γ sont utilisées pour linéariser les contraintes non-linéaires de l'ensemble $\mathcal{U}$. Les variables α, β, etγ sont la représentation binaire des entiers e, p, et leur produit. Étant donné que les valeurs de e et p sont bornées, il en est de même pour les variables α, β, et γ. Nous pouvons calculer facilement ces valeurs en utilisant le logarithme en base 2.

En d'autres termes, pour chaque scénario, nous déterminons une contrainte qui sera violée et nous voulons maximiser la violation minimale.

5.4.3 L'algorithme complet

Dans cette section, nous décrivons comment le problème $RP(\mathcal{U})$ est résolu en combinant ce que nous avons présenté auparavant. Initialement, l'ensemble d'incertitude est vide $\mathcal{U}' = \emptyset$ et nous calculons la solution de $RP(\emptyset)$. Ensuite, nous procédons à la génération d'un scénario pour lesquelles les solutions du premier niveau et du second niveau ne sont pas réalisables. Cela sera fait en résolvant le problème WC_2. Si un tel scénario existe ce scénario sera ajoutée à l'ensemble $\mathcal{U}'$. Puis, en utilisant WC_1 nous vérifions si la solution courante du premier niveau peut être réparée en gardant la même valeur courante de la fonction objectif Δ. Si c'est le cas, nous procédons alors à la génération d'un nouveau scénario en gardant la même solution courante. Dans le cas contraire, nous devons résoudre le problème $RP(\mathcal{U}')$ une autre fois, ce qui donne une nouvelle solution de premier niveau et des solutions de second niveau. Si un scénario de pire cas n'existe pas alors la solution courante est optimale et l'algorithme s'arrête. Notons que la résolution de WC_1, WC_2 et $RP(\mathcal{U}')$ est faite par le Solveur CPLEX.

Cette procédure est résumé dans l'Algorithme 7. Dans l'étape 2, nous résolvons le problème RP comme été défini dans la section 5.3 pour un sous-ensemble de scénarios. Au début de l'algorithme, ce sous-ensemble est vide. Dans l'étape 3, nous fixons la solution courante (les solutions du premier et du second niveau), et nous déterminons un nouveau scénario pour lequel la solution courante sera infaisable. Si un tel scénario existe, dans l'étape 6 nous fixons les variables de la solution du premier niveau et nous considérons la distance de réparation en respectant ce scénario. Si la réparation de la solution est possible sans augmentation de la valeur courante de la distance de réparation, nous retournons à l'étape 3 et nous générons un nouveau scénario. Si une telle réparation n'est pas possible, alors le scénario du pire cas est retrouvé et nous déterminons dans l'étape 2 une nouvelle

solution du premier niveau et des solutions du second niveau. Ces processus seront répétés jusqu'à ce qu'aucun nouveau scénario ne puisse être généré.

Algorithme 7 Méthode Iterative $RP(\mathcal{U})$

1: Mettre $\mathcal{U}' = \emptyset$
2: Résoudre $RP(\mathcal{U}')$
3: Trouver un scénario de pire des cas qui rend les solutions de l'étape 2 infaisables
4: **si** ce scénario existe **alors**
5: Ajouter le scénario généré ξ à $\mathcal{U}'$
6: Minimiser la différence entre la solution nominale et la solution obtenue par le scénario ξ
7: **si** (la valeur de la nouvelle Δ est supérieure à la valeur courante de Δ) **alors**
8: Allez à l'étape 2
9: **sinon**
10: Allez à l'étape 3
11: **finsi**
12: **sinon**
13: **Retourner** la solution optimale trouvée
14: **finsi**

La partie la plus gourmande en temps de calcul est la résolution du problème $RP(\mathcal{U}')$. Les deux problèmes WC_1 et WC_2 seront résolus en un temps considérablement inférieur au temps de résolution de $RP(\mathcal{U}')$. Ainsi, pour améliorer les performances de l'Algorithme 7, un scénario est ajouté à l'ensemble $\mathcal{U}'$ comme décrit dans l'étape 5, si la solution courante du premier niveau n'est pas réalisable pour une valeur de Δ (si la condition de l'étape 7 est respectée). De cette manière, le nombre total d'itérations peut être très grand, mais la taille de l'ensemble d'incertitude est maintenue aussi petit que possible (Étape 2).

5.5 Expérimentations

Dans cette section, nous présentons les résultats expérimentaux réalisés pour tester l'efficacité du modèle mathématique et l'efficacité de l'algorithme basé sur l'oracle. Nous commençons par une description des instances utilisées et des algorithmes que nous avons testés. Enfin, nous discutons les résultats obtenus.

Environnement Toutes les expérimentations ont été faites sur un Intel Xeon CPU E5-2670 avec 2.60GHz. Seulement 16 cœurs ont été utilisés pour chaque exécution de l'algorithme. Une mémoire de 96GB est disponible, mais toutes les exécutions nécessitent moins de 1GB de mémoire. Les programmes ont été compilés sous Ubuntu 4.6.3, et les modèles mathématiques ont été résolus par le solveur CPLEX 12.4.

Données et scénarios Nous considérons les instances de la ville de Kaiserslautern (Allemagne) lors d'une catastrophe dans le centre de la ville (crash d'avion ou une bombe) qui conduit à une zone d'évacuation de 500 mètres. Notons que cette approche n'a pas été

testée sur les instances de la ville de Nice (France) car ces instances sont de trop grande taille.

Le nombre total de personnes à évacuer par bus est de 1750 personnes ; chaque bus a une capacité égale à 80, ce qui correspond à un nombre total de trajets égal à 22. Les personnes sont réparties sur quatre points de rassemblement dans la zone endommagée. De plus, Il y a quatre centres de secours disponibles, qui sont des gymnases de différentes tailles. Trois bus sont disponibles.

Pour prendre en compte l'incertitude, nous supposons que chaque trajet peut prendre une unité de plus ou de moins que prévu : $p_{ij} \in \{\hat{p}_{ij} - 1, \hat{p}_{ij}, \hat{p}_{ij} + 1\}$. Pour chaque point de rassemblement, il y a un trajet de bus en plus ou en moins : $e_i \in \{\hat{e}_i - 1, \hat{e}_i, \hat{e}_i + 1\}$. En outre, un seul bus ne sera pas disponible ($B \in \{\hat{B} - 1, \hat{B}\}$).

Puisque la cardinalité de l'ensemble incertain dépend de la valeur de $T^{\text{exp}}_{\text{evac}}$ imposée dans la contrainte (2.25), nous notons cet ensemble par $\mathcal{U}(T^{\text{exp}}_{\text{evac}})$. Sa cardinalité est présenté dans la figure 2.5.

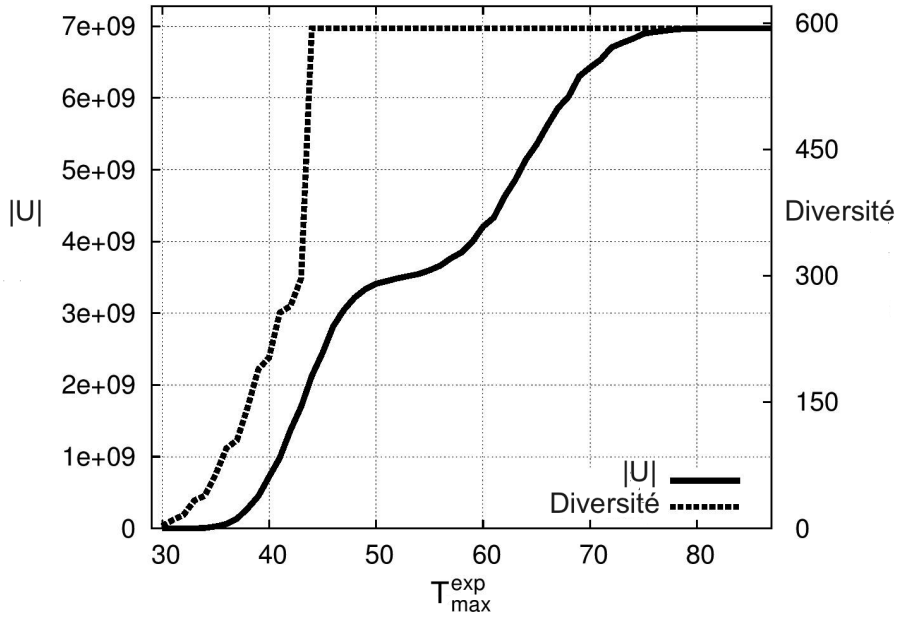


FIGURE 2.5 – Cardinalité et diversité de l'ensemble $\mathcal{U}$ dépendantes de $T^{\text{exp}}_{\text{evac}}$.

Notons que pour une valeur assez grande de $T^{\text{exp}}_{\text{evac}}$ ($T^{\text{exp}}_{\text{evac}} \geq 87$), dans notre cas, la contrainte (2.25) devient redondante et le nombre de scénarios est égal au nombre de points entiers dans l'hypercube (ici, $|\mathcal{U}(87)| = 3^{20} \cdot 2 \approx 7 \cdot 10^9$). Puisque l'ensemble d'incertitude est très grand, la résolution directe du problème $RP(\mathcal{U})$ avec un solveur mathématique est impossible.

Pour mieux comprendre la structure de l'ensemble $\mathcal{U}$, nous présentons dans la figure 2.5 la valeur appelée "diversité" qui compte les valeurs minimales et maximales dans l'ensemble $\mathcal{U}$. La diversité peut être considérée comme une mesure de points extrêmes de $\mathcal{U}$. Cette valeur est calculé par :

$$diversité(\mathcal{U}) = (B_{\max} - B_{\min} + 1) \cdot (e_{\max} - e_{\min} + 1) \cdot (p_{\max} - p_{\min} + 1),$$

où

$$B_{\max} = \max \{B : \exists(e,p) \text{ s.c. } (e,p,B) \in \mathcal{U}\}$$
$$e_{\max} = \max \left\{ \sum_{i \in \mathcal{P}} e_i : \exists(p,B) \text{ s.c. } (e,p,B) \in \mathcal{U} \right\}$$
$$p_{\max} = \max \left\{ \sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{S}} p_{ij} : \exists(e,B) \text{ s.c. } (e,p,B) \in \mathcal{U} \right\}$$

Les valeurs minimales sont calculées de manière analogue que les valeurs maximales. Puisque l'augmentation de la valeur de $T^{\text{exp}}_{\text{evac}}$ cause l'augmentation du temps de calcul, nous présentons uniquement les résultats pour le cas $29 \leq T^{\text{exp}}_{\text{evac}} \leq 52$.

Avant de présenter les résultats expérimentaux, nous rappelons les différentes notations utilisées pour modéliser le problème robuste.

P	le problème non-robuste d'évacuation par bus.
RP	le problème robuste d'évacuation par bus.
WC_1, WC_2	les sous-problèmes pour déterminer la solution de premier niveau de pire des cas.
T_{evac}	la fonction objectif de P.
T^*_{evac}	la valeur optimale de la fonction objectif de P.
$T^{\text{exp}}_{\text{evac}}$	une borne supérieure de la valeur de date de fin d'évacuation souhaitée (donnée par le décideur).
$T^{\text{nom}}_{\text{evac}}$	une borne supérieure de la valeur nominale de date de fin d'évacuation (donnée par l'approche ϵ-contrainte)
$T^{\text{wc}}_{\text{evac}}$	une borne supérieure de la valeur de date de fin d'évacuation dans le scénario du pire cas (donnée par l'approche ϵ-contrainte)
S	le nombre de points de rassemblement.
T	le nombre de centres de secours.
$\underline{B}$, $\overline{B}$	borne inférieure et borne supérieure du nombre de bus disponibles.
$\underline{p}_{ij}$, $\overline{p}_{ij}$	borne inférieure et borne supérieure de la durée de trajet depuis $i \in \mathcal{P}$ vers $j \in \mathcal{S}$ incluant le retour des bus vers la zone endommagée.
$\underline{e}_i$, $\overline{e}_i$	borne inférieure et borne supérieure du nombre d'évacués regroupés dans $i \in \mathcal{P}$.
$\hat{\xi} = (\hat{l}, \hat{d}, \hat{B})$	le scénario nominal.
$\Delta(x, x^{\xi})$	la distance de réparation entre la solution de premier niveau et les solutions du second niveau x, x^{ξ}.

Afin de résoudre le problème d'incertitude présenté ci-dessus, nous utilisons l'Algorithme itératif 7 pour les valeurs $T^{\text{exp}}_{\text{evac}} \in [29, 52]$. Pour des expérimentations préliminaires,

nous avons constaté que si on augmente l'ensemble des scénarios la résolution du problème $RP(\mathcal{U}')$ avec le solveur mathématique est la plus consommatrice en terme de temps. Par contre, les problèmes WC_1 et WC_2 sont relativement faciles à résoudre.

Pour cette raison, nous considérons une approche alternative pour résoudre $RP(\mathcal{U}')$ sans l'utilisation d'un solveur mathématique pour le modèle tel qu'il est. Comme dans [Fischetti and Monaci, 2013], nous réécrivons le modèle comme un problème de faisabilité, en considérant la fonction objectif comme une contrainte du modèle. Nous commençons par la valeur de la fonction objectif de la dernière solution obtenue par la procédure itérative, et nous vérifions l'existence d'une solution avec la même valeur de la fonction objectif. Si une telle solution n'existe pas, la valeur de la fonction objectif sera augmentée d'une unité, et cette procédure sera répétée.

Pour accélérer la résolution du problème de faisabilité, et au lieu d'utiliser la fonction objectif égale à 0, nous considérons une fonction objectif qui minimise la différence entre Δ et sa dernière valeur déjà calculée. Ce qui revient à calculer une solution qui soit proche de cette dernière.

Tout au long de cette section, nous appelons "IP", l'approche utilisant le modèle de $RP(\mathcal{U}')$ tel qu'il est, et "LOC" l'approche utilisant la faisabilité ("recherche locale").

En outre, nous considérons deux approches différentes pour résoudre les sous-problèmes de type WC_2. Dans la première approche, nous résolvons chaque sous-problèmes à l'optimalité (nous déterminons le scénario qui a la plus grande valeur de violation), et dans la deuxième approche, nous arrêtons la résolution quand la première solution réalisable est trouvée. La première approche est nommée "OPT", tandis que la deuxième est appelée "FEAS".

Chaque instance est résolue en combinant les méthodes IP/OPT, IP/FEAS, LOC/OPT et LOC/FEAS avec les valeurs suivantes de $T^{\text{nom}}_{\text{evac}}$ et $T^{\text{wc}}_{\text{evac}}$:

$$\begin{aligned}
T^{\text{nom}}_{\text{evac}} &= 1.00 \cdot T^{*}_{\text{evac}} \text{ and } T^{\text{wc}}_{\text{evac}} = 1.50 \cdot T^{*}_{\text{evac}};\\
T^{\text{nom}}_{\text{evac}} &= 1.25 \cdot T^{*}_{\text{evac}} \text{ and } T^{\text{wc}}_{\text{evac}} = 1.75 \cdot T^{*}_{\text{evac}};\\
T^{\text{nom}}_{\text{evac}} &= 1.50 \cdot T^{*}_{\text{evac}} \text{ and } T^{\text{wc}}_{\text{evac}} = 2.00 \cdot T^{*}_{\text{evac}},
\end{aligned}$$

où T^{*}_{evac} dénote la valeur optimale du problème non-robuste (voir la section 5.1).

Pour chaque solution, nous mesurons la valeur de la fonction objectif Δ, et le temps de calcul. De plus, nous considérons la valeur Δ de la solution nominale robuste. Pour le calcul de cette valeur, nous générons les scénarios à l'aide de l'Algorithme 7.

Nous présentons dans la figure 2.6 la valeur de la fonction objectif Δ associée à la solution nominale qui dépend de $T^{\text{wc}}_{\text{evac}}$.

Nous observons aussi que dans la figure 2.6 une augmentation de la valeur de la fonction objectif entre $T^{\text{exp}}_{\text{evac}} = 43$ et $T^{\text{exp}}_{\text{evac}} = 44$, qui correspond à un saut de diversité des scénarios (cf. figure 2.5). Cela implique que la diversité et non la cardinalité qui a un impact sur la valeur de la fonction objectif Δ. En outre, la solution robuste avec $T^{\text{nom}}_{\text{evac}} = 1.00$ a la même valeur de la fonction objectif que la fonction objectif de la solution nominale.

Les temps d'exécution sont représentés dans la figure 2.7 pour l'échelle logarithmique sur l'axe des y.

La limite de temps imposée est de $36000s$ ce qui est suffisant pour la résolution des

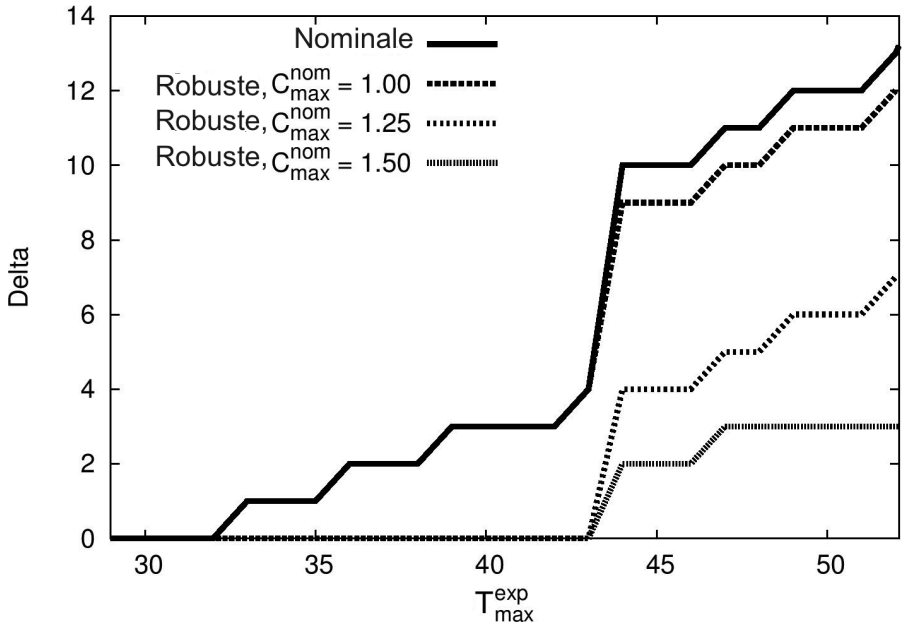


FIGURE 2.6 – Valeurs de la fonction objectif Δ dependament de $T_{\text{evac}}^{\text{exp}}$.

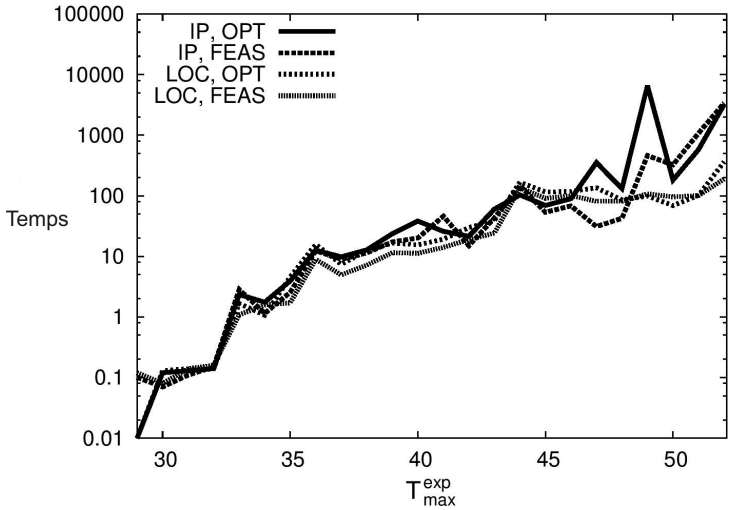
deux formulations mathématiques IP. En général, les résultats présentés indiquent que :

- les temps d'exécution de LOC sont inférieurs aux temps nécessaires pour la résolution de IP,
- il n'existe pas d'indicateur efficace pour déterminer si le temps d'exécution de FEAS est meilleur que celui de OPT. Pour les mêmes instances et pour des valeurs de $T_{\text{evac}}^{\text{exp}}$ entre 43 et 44, nous observons un écart important dans le temps de résolution. Les Tableaux 2.19, 2.20 et 2.21 présentent en détails les données qui ont générées ces figures.

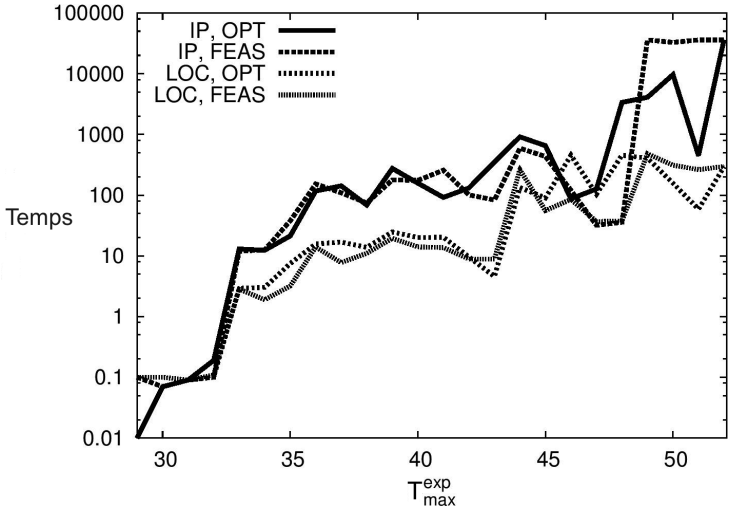
Pour résumer les résultats présentés, nous soulignons que a) le modèle robuste proposé est capable de produire des solutions avec un compromis raisonnable entre les valeurs nominale et robuste de la fonction objectif. Cela permettra une plus grande flexibilité au planificateur dans son choix des plans d'évacuation possibles ; et b) les temps de résolution sont encore raisonnables pour une planification préventive à condition que la taille et la diversité de l'ensemble des scénarios ne soient pas trop élevées.

Exemple 4 *Le scénario nominal utilisé dans cette expérimentation est le suivant :*

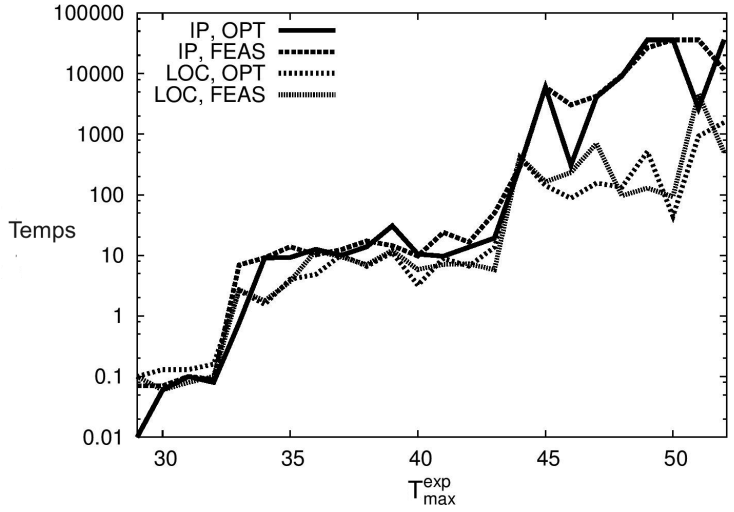
$$\hat{e} = (6, 6, 7, 8)^T$$
$$cap = (12, 5, 7, 7)^T$$
$$B = 3$$
$$\hat{p} = \begin{pmatrix} 8 & 6 & 4 & 7 \\ 7 & 6 & 5 & 6 \\ 5 & 5 & 4 & 8 \\ 5 & 3 & 6 & 9 \end{pmatrix}$$



(a) Temps d'éxécution en *secondes* pour $T_{\text{evac}}^{\text{nom}} = 1.00$



(b) Temps d'éxécution en *secondes* pour$T_{\text{evac}}^{\text{nom}} = 1.25$



(c) Temps d'éxécution en *secondes* pour $T_{\text{evac}}^{\text{nom}} = 1.50$

FIGURE 2.7 – Temps d'éxécution en *secondes* dépendant de $T_{\text{evac}}^{\text{exp}}$.

La carte de la ville de Kaiserslautern est donné par la figure 2.8.

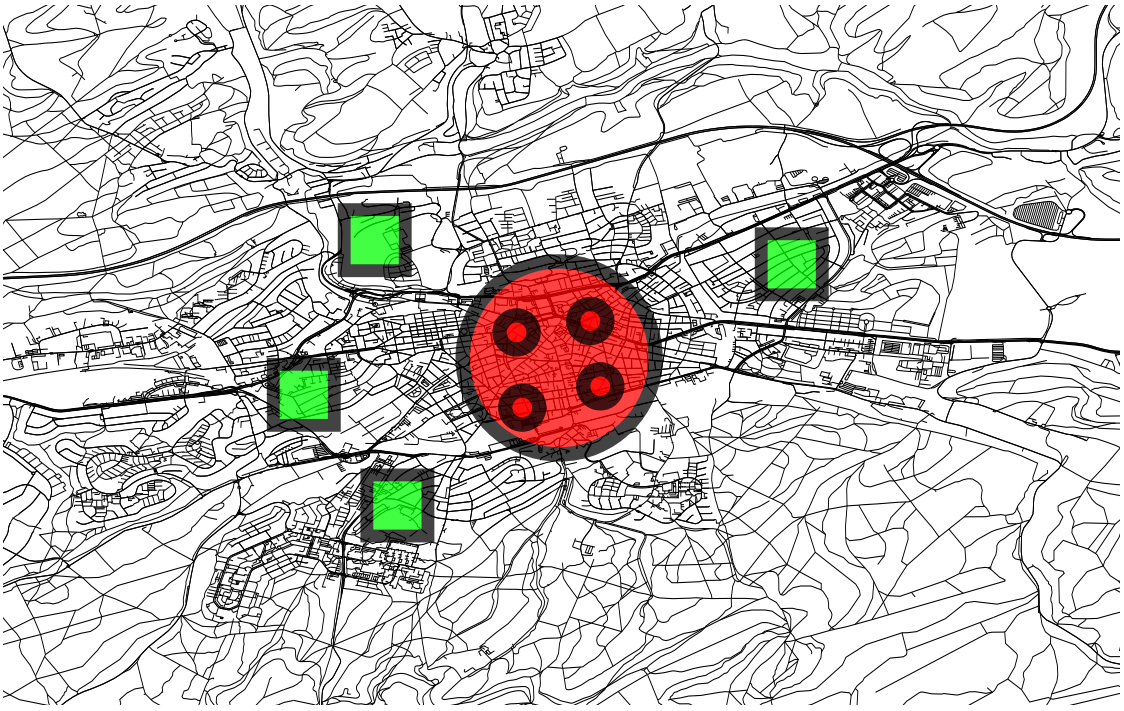


FIGURE 2.8 – Carte de la ville de Kaiserslautern incluant les points de rassemblement et les centres de secours.

Nous montrons dans le Tableau 2.18 la différence entre la solution non-robuste (nominale) et la solution robuste en prenant $T_{evac}^{exp} = 45$ *dans cette exemple.*

Nom	b_1	(1,2,4)	(2,4,2)	(3,4,2)	(4,1,3)	(5,1,3)	(6,2,3)	(7,2,3)	(9,4,2)			
	b_2	(1,1,3)	(2,2,4)	(3,3,1)	(4,3,1)	(5,3,1)	(6,3,1)	(7,4,2)				
	b_3	(1,1,3)	(2,3,1)	(4,2,4)	(5,2,4)	(6,3,1)	(13,4,2)	(22,1,3)				
$R^{1.00}$	b_1	(1,3,1)	(2,4,2)	(3,4,2)	(4,1,3)	(5,1,3)	(8,3,3)	(9,4,2)	(12,4,2)	(14,3,3)		
	b_2	(1,1,3)	(2,2,4)	(4,2,4)	(5,3,1)	(12,2,4)	(21,2,4)					
	b_3	(1,1,3)	(4,2,4)	(5,2,4)	(6,3,1)	(8,4,2)	(13,3,1)	(22,1,3)				
$R^{1.50}$	b_1	(1,2,1)	(16,2,4)									
	b_2	(2,2,4)	(3,3,3)	(4,3,1)	(5,3,1)	(6,3,1)	(7,4,2)	(10,4,2)	(15,1,3)	(17,3,1)	(20,2,3)	(22,1,3)
	b_3	(1,1,3)	(2,3,1)	(4,2,4)	(5,2,4)	(6,3,1)	(9,4,2)	(13,4,2)	(16,2,4)	(17,1,3)	(21,4,2)	(22,1,3)

TABLE 2.18 – Trois solutions pour $T_{\text{evac}}^{\text{exp}} = 45$: la solution nominale (Nom), une solution robuste avec $T_{\text{evac}}^{\text{nom}} = 1.00T_{\text{max}}^{*}$ ($R^{1.00}$), et une solution robuste avec $T_{\text{evac}}^{\text{nom}} = 1.50T_{\text{max}}^{*}$ ($R^{1.50}$).

Chaque case de ce tableau est composée de trois champs (r, i, j)*, où* r *est le tour dans lequel un trajet a lieu,* i *est le point de rassemblement (origine) et* j *le centre de secours (destination). Les valeurs de la fonction objectif* Δ *sont présentées dans la figure 2.6 : ces valeurs sont 10 pour la solution nominale, 9 et 2 sont respectivement les valeurs associées à la première solution robuste et à la deuxième solution robuste.*

Si nous comparons la solution nominale avec la solution robuste $R^{1.00}$*, nous constatons*

que la solution nominale distribue les trajets de façon uniforme dans le but de minimiser la date de fin d'évacuation. Tandis que que pour la solution robuste, le bus b_2 fait 6 trajets. Cette solution prend en considération le fait qu'un bus peut être non disponible. De plus la valeur de δ est inférieure à celle de la solution nominale par une unité. Notons aussi que de nouveaux trajets ont été utilisés pour cette configuration, par exemple le trajet à partir du point de rassemblement 3 vers le centre de secours 3.

Comme la solution robuste $R^{1.50}$ a une grande valeur de la date de fin, la distribution des trajets sera non équitable entre les bus. Ce qui donne une solution avec deux trajets pour le bus b_1. L'absence de ce bus sera le scénario du pire cas qui nécessite une distance de réparation égale à 2.

Les Tableaux 2.19–2.21 montrent les valeurs numériques utilisées dans les figures 2.7a–2.7c.

$T_{\text{evac}}^{\text{exp}}$	IP/OPT	IP/FEAS	LOC/OPT	LOC/FEAS
29	0.01	0.10	0.01	0.12
30	0.12	0.07	0.13	0.08
31	0.13	0.11	0.14	0.14
32	0.14	0.15	0.14	0.16
33	2.34	2.84	1.67	1.09
34	1.75	1.15	1.07	1.58
35	3.92	2.55	4.56	1.71
36	12.65	12.38	15.74	8.90
37	9.82	8.93	7.40	4.96
38	12.77	11.40	12.43	7.17
39	23.65	17.37	16.58	11.49
40	38.19	20.08	15.50	11.24
41	25.89	45.88	19.18	14.11
42	21.45	15.00	29.85	18.91
43	60.27	42.95	39.00	24.27
44	103.40	147.32	167.13	134.93
45	69.17	53.31	115.55	90.67
46	89.49	68.05	117.34	100.26
47	352.42	31.04	135.46	81.18
48	131.11	42.42	85.08	81.65
49	6557.75	459.08	99.15	107.37
50	181.84	324.24	68.95	95.97
51	577.02	1075.41	101.48	100.60
52	3184.44	3363.61	361.67	188.83

TABLE 2.19 – Temps d'éxécution en *secondes* pour $T_{\text{evac}}^{\text{nom}} = 1.00 \cdot T_{\text{evac}}^{*}$ et $T_{\text{evac}}^{\text{wc}} = 1.50 \cdot T_{\text{evac}}^{*}$

$T_{\text{evac}}^{\text{exp}}$	IP/OPT	IP/FEAS	LOC/OPT	LOC/FEAS
29	0.01	0.10	0.10	0.01
30	0.07	0.07	0.07	0.10
31	0.09	0.09	0.09	0.09
32	0.19	0.10	0.10	0.11
33	12.97	11.94	2.91	2.83
34	12.39	12.56	3.02	1.89
35	21.24	37.59	7.46	3.19
36	117.06	152.26	15.74	13.91
37	142.55	109.33	16.83	7.76
38	68.75	74.15	13.82	11.02
39	275.08	178.09	24.70	19.41
40	157.70	175.47	20.05	14.05
41	91.84	256.81	20.27	13.70
42	131.15	101.30	9.65	8.91
43	351.91	83.11	4.56	8.78
44	909.38	590.83	132.91	262.65
45	653.43	431.70	90.37	55.77
46	86.81	120.18	454.68	85.09
47	126.23	31.98	103.51	36.73
48	3369.52	35.84	454.59	37.65
49	4085.34	$>$ 36000.00	405.19	480.02
50	9651.69	32818.30	157.28	310.47
51	449.00	$>$ 36000.00	58.93	262.88
52	$>$ 36000.00	$>$ 36000.00	292.13	296.4

TABLE 2.20 – Temps d'éxécution en *secondes* pour $T_{\text{evac}}^{\text{nom}} = 1.25 \cdot T_{\text{evac}}^{*}$ et $T_{\text{evac}}^{\text{wc}} = 1.75 \cdot T_{\text{evac}}^{*}$

$T_{\text{evac}}^{\text{exp}}$	IP/OPT	IP/FEAS	LOC/OPT	LOC/FEAS
29	0.01	0.07	0.10	0.10
30	0.06	0.07	0.13	0.06
31	0.10	0.10	0.13	0.08
32	0.08	0.09	0.16	0.10
33	0.77	6.95	2.63	2.69
34	9.09	9.13	1.55	1.77
35	9.21	13.89	4.04	3.73
36	12.59	10.22	4.83	11.94
37	10.01	12.41	10.14	9.44
38	13.87	17.28	6.61	7.01
39	30.69	14.45	11.09	12.02
40	10.49	9.77	3.22	5.82
41	9.69	23.93	8.92	7.09
42	13.66	16.49	6.64	7.25
43	19.31	49.59	13.59	5.78
44	296.70	288.36	405.22	416.53
45	5944.44	5971.53	141.61	163.53
46	307.65	3044.33	87.68	233.42
47	4088.59	4238.50	155.55	699.85
48	9109.35	9446.29	133.81	96.84
49	> 36000.00	26658.50	515.10	129.00
50	> 36000.00	> 36000.00	44.22	93.00
51	2642.12	> 36000.00	926.62	4555.57
52	> 36000.00	11624.60	1558.56	505.16

TABLE 2.21 – Temps d'exécution en *secondes* pour $T_{\text{evac}}^{\text{nom}} = 1.50 \cdot T_{\text{evac}}^{*}$ et $T_{\text{evac}}^{\text{wc}} = 2.00 \cdot T_{\text{evac}}^{*}$

6 Le problème de la minimisation de la durée et du risque de l'évacuation

Un des objectifs d'une évacuation est de garantir la sécurité des personnes. Pour cette raison, nous nous intéressons dans cette section au problème bicritère d'évacuation par bus défini dans la section 2. Nous cherchons à minimiser à la fois la date de fin d'évacuation et la somme des risques pour le problème dans lequel les durées de transport (p_{ijt}) sont dépendante du temps. Ce problème est $\mathcal{NP}$-difficile car sa version monocritère est $\mathcal{NP}$-difficile (voir [Goerigk and Grun, 2014]).

Dans le domaine de l'optimisation multicritère, plusieurs méthodes de calcul des solutions de front de Pareto sont connues ([T'kindt and Billaut, 2006]). Dans ce travail, nous utilisons l'approche ε-contrainte décrite de la façon suivante : minimiser la somme des risques sous la contrainte que la date de fin d'évacuation ne dépasse pas une certaine valeur ε. La résolution de ce problème pour différentes valeurs de ε permet d'énumérer les optima de Pareto. Pour cela, on prend ε égale à une borne supérieure du critère T_{evac} : la solution obtenue (T_{evac}, R_{evac}) est ajoutée à l'ensemble des optima Pareto. Puis, nous fixons $\varepsilon = T_{\text{evac}} - 1$ et nous réitérons jusqu'à ce qu'il n'existe aucune solution réalisable.

D'un point de vu pratique, la résolution de ce problème avec la méthode ε-contrainte, telle qu'elle est soulignée ci-dessus, est logique : la valeur de ε représente un seuil qui garantit que l'évacuation ne dépasse pas une certaine date ε. En outre, le temps d'évacuation est une valeur très descriptive, par contre la somme des risques est une valeur abstraite, et la fixation de sa valeur souhaitée est difficile en pratique. Dans cette section, nous nous intéressons à la résolution itérative de ce problème avec la méthode ε-contrainte pour énumérer l'ensemble des optimas de Pareto pour les critères T_{evac} et R_{evac}.

6.1 Formulation Mathématique

Pour la modélisation du BBEP, nous avons proposé un modèle mathématique indexé sur le temps. La motivation de ce choix a été indiquée dans la section 4.1. La modélisation du problème bicritère d'évacuation par bus où T_{evac}^{exp} est la qualité désirée du critère T_{evac} et T est l'horizon temporel est donnée par :

Variables

- $\forall i \in \mathcal{P}, \forall j \in \mathcal{S}, \forall t \in \mathcal{T}$;

$$x_{ijt} = \begin{cases} 1 & \text{si un bus débute une opération d'évacuation} \\ & i \text{ vers } j \text{ à } [t, t+1[, \\ 0 & \text{sinon.} \end{cases}$$

- R_{evac} : La somme des risques encourus.

La formulation (IP) proposée pour une valeur ε donnée, est la suivante :

Objectif

$$\min R_{evac} \tag{2.71}$$

Contraintes

$$R_{evac} \geq \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{S}} x_{ijt} * r_{ijt} \tag{2.72}$$

$$T^{exp}_{\text{evac}} \geq (t + p_{ijt})x_{ijt} \qquad \forall i \in \mathcal{P}, \forall j \in \mathcal{S}, \forall t \in \mathcal{T} \tag{2.73}$$

$$\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{P}} x_{ijt} \leq cap_j \qquad \forall j \in \mathcal{S} \tag{2.74}$$

$$\sum_{i\in\mathcal{P}}\sum_{j\in\mathcal{S}} \sum_{\substack{t'\in[0,t] \\ p_{ijt'}+t'>t}} x_{i,j,t'} \leq B \qquad \forall t \in \mathcal{T} \tag{2.75}$$

$$\sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{S}} x_{ijt} = 1 \qquad \forall i \in \mathcal{P} \tag{2.76}$$

$$x_{ijt} \in \{0, 1\} \tag{2.77}$$

La fonction objectif (2.71) est de minimiser la somme d'exposition aux risques. La valeur du critère R_{evac} est calculée par la contrainte (2.72). Les contraintes (2.73) définissent la valeur du critère T_{evac} et assurent que l'évacuation ne dépasse pas la date T^{exp}_{evac} de la fonction l'objectif T_{evac}. Les contraintes (2.74) sont des contraintes de capacité des centres de secours. Les contraintes (2.75) sont des contraintes de capacité de bus ; on ne peut pas excéder le nombre de bus disponibles. Les contraintes (2.76) assurent qu'une opération d'évacuation ne se fait qu'une seule fois. Les contraintes (2.77) sont des contraintes d'intégrité des variables x_{ijt}.

6.2 Une méthode par séparation et évaluation

Dans cette section, nous présentons les différents éléments d'une procédure par séparation et évaluation (Branch-and-Bound) permettant de résoudre à l'optimalité le problème ε-contrainte d'évacuation par bus.

6.2.1 Le schéma de branchement

Le schéma de branchement définit la manière d'explorer l'espace de recherche lorsque l'on passe d'un nœud père à ses nœuds fils, c'est-à-dire quelles décisions faut-il prendre entre le nœud père et les nœuds fils ? La réponse à cette question est que, souvent, le schéma de branchement dépend de la représentation d'une solution.

Dans notre cas, une solution consiste à définir un planning pour un ensemble de bus afin transporter des personnes depuis les points de rassemblement vers les centres de secours. Pour la construction d'une solution partielle Seq d'un nœud intermédiaire n, nous affectons temporellement une opération d'évacuation, associée à un point de rassemblement P_i, à un bus disponible B_k. Ce dernier assure le transfert de cette opération vers un centre de secours S_j, qui a une capacité résiduelle positive. La figure 2.9 présente un arbre de recherche utilisant ce schéma de branchement. Le choix de ce schéma permet d'éviter le branchement aussi sur les instants de temps pour placer les opérations évacuation, puisque les durées de transport et les valeurs de risque sont dépendantes du temps. Cet aspect temporel est pris

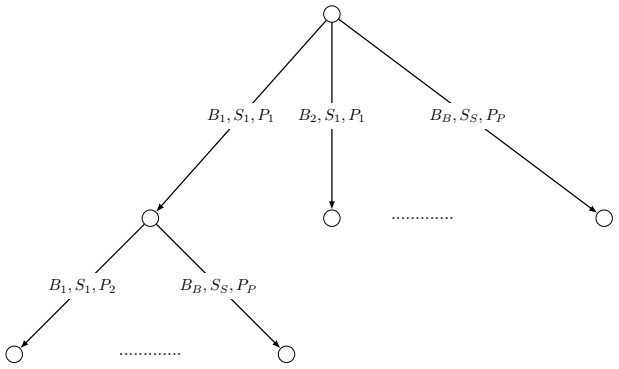


FIGURE 2.9 – Schéma de branchement.

en considération lors du calcul de la solution optimale d'une séquence partielle Seq d'un nœud de l'arbre de recherche. Pour ce faire, nous avons utilisé un programme dynamique. Ce dernier utilise la fonction de récurrence $F(Seq, t)$ qui représente la valeur minimale de la somme des risques pour une séquence partielle d'opérations Seq, où la première opération de cette séquence commence à la date t. Cette valeur peut être calculée à l'aide de la fonction récursive suivante :

$$F(Seq, t) = 0 \ \text{, si } Seq = \emptyset, \forall t \tag{2.78}$$

$$F(Seq, t) = \min_{\substack{t' \in T_i \text{ et } t' \geq t \\ i = Seq[1]}} \left(F(Seq/\{i\}, t' + p_{i,j,t'}) + r_{i,j,t'}\right) \tag{2.79}$$

avec $Seq[j]$ est l'opération qui est en position j dans la séquence Seq. T_i est l'ensemble des instants de temps qui provoquent un changement sur les temps des trajets et sur les valeurs de risque d'une opération i. Notons que les contraintes (2.73) sont vérifiées pour la sous-séquence Seq. Il suffit de poser pour $t' + p_{i,j,t'} > T_{\text{evac}}$, $F(Seq/\{i\}, t' + p_{i,j,t'}) + r_{i,j,t'} = +\infty$. La solution optimale est donnée par $F(Seq, 0)$. Cette solution peut être calculée en $O(k^{|Seq|})$ avec $k = \max_{i \in seq(|T_i|)}$.

6.2.2 Borne supérieure

Pour construire une solution réalisable du problème BBEP, nous proposons une méthode de recherche par voisinage appelée matheuristique (cf. section 4.4.2). Cette approche est utilisée pour construire une solution réalisable. Comme pour n'importe quelle méthode de recherche par voisinage, cette matheuristique prend en entrée une solution réalisable et essaye de l'améliorer en explorant son voisinage. Pour obtenir une solution réalisable initiale, nous avons proposé deux heuristiques de liste. Celles-ci ont pour but d'énumérer le front de Pareto. L'Algorithme 8 présenté ci-dessous est un algorithme des heuristiques de liste qui prend en entrée une règle $\mathcal{R}$ et la valeur souhaitée de la date de fin d'évacuation T_{evac}^{exp}. Soit $ListOpr$ l'ensemble des opérations d'évacuation qui seront affectées aux bus.

ListTriée est l'ensemble des opérations triées selon la règle $\mathcal{R}$ à instant t, *FaisListTriée* est un sous-ensemble de l'ensemble *ListTriée* qui contient les opérations qui vérifient la contrainte $LB_{T_{\text{evac}}} < T^{exp}_{\text{evac}}$, T^{b}_{evac} est la date de fin d'évacuation d'un bus b, et R^{b}_{evac} est la somme des risques cumulée lorsqu'un bus b est utilisé. La solution d'une heuristique de liste est sauvegardée dans *ListOrdo*. Chaque opération ajoutée à *ListOrdo* sera supprimée des ensembles *ListOpr*, *ListTriée*, et *FaisListTriée*.

La fonction *event* présentée dans l'Algorithme 8 ci-dessous a pour but de vérifier si un événement nécessite un recalcule de la liste des priorités *ListTriée*. Chaque règle $\mathcal{R}$ a sa fonction spécifique *event*. Notons que nous possédons une archive externe, nommée *nondominée*, qui est maintenue pour les différents appels aux heuristiques de liste. Cette archive contient l'ensemble des solutions du front de Pareto déjà calculées. De plus, les heuristiques de liste sont exécutées pour différentes valeurs de T^{exp}_{evac}. Pour chaque valeur donnée de T^{exp}_{evac}, les algorithmes de liste calculent une solution *ListOrdo*. La fonction MJArchive() est appelée pour mettre à jour l'ensemble des solutions de front de Pareto. Si la solution *ListOrdo* n'est pas dominée par une solution de l'ensemble *nondominée*, alors cette solution sera ajoutée à l'archive *nondominée*. Si une solution de *nondominée* est dominée par *ListOrdo* alors cette solution est supprimée de l'archive.

Dans ce qui suit nous présentons deux versions de l'algorithme de liste.

1. Version 1 : cette version utilise la règle $\mathcal{R}_1$. Les opérations d'évacuation sont triées selon l'ordre croissant des valeurs du risque r_{ijt}.
2. Version 2 : cette version utilise la règle $\mathcal{R}_2$. dans cette règle, des opérations d'évacuation opr_i sont ordonnancées selon l'ordre croissant de leurs valeurs :

$$\frac{r_{ijt}}{\sqrt{\frac{1}{N}\left(\sum_{j\in\mathcal{S}}\sum_{t\in\mathcal{T}}(r_{ijt}-\bar{r}_{ijt})^2\right)}}$$

   où le dénominateur est l'écart type du risque r_{ijt} associé à l'opération i. En utilisant cette règle, nous assurons qu'une opération opr_i sera placée à l'instant t, si à cet instant elle a une petite valeur de risque et une plus grande valeur du risque pour les dates suivantes.

   Dans ces deux versions, la fonction *event* retourne vrai s'il y a une ou des modifications dans les valeurs des risques associées aux opérations.

Maintenant, nous expliquons en détails les étapes de la matheuristique pour résoudre BBEP, qui sont analogues à celles développées dans la section 4.4.2. Soit une solution initiale du front de Pareto calculée par les algorithmes de liste. Nous utilisons la formulation indexée sur le temps présentée dans la section 6.1 afin de construire une matheuristique pour le problème BBEP. La matheuristique essaye d'améliorer la solution initiale en explorant son voisinage défini comme suit. Soit un ordonnancement réalisable (solution initiale) $\bar{x} =< \bar{x}_{ijt}, i \in \mathcal{P}, j \in \mathcal{S}, t \in \mathcal{T} >$, où $\bar{x}_{ijt} = 1$, si l'opération i est transporté vers le centre j à la date t. Nous définissons un voisinage de cette solution $\mathcal{N}$ $(\bar{x}, r, h)$ en choisissant la date r dans l'ordonnancement et un paramètre de taille h. Soit $\tilde{S}(r, h)$ l'ensemble des indices des opérations qui commencent dans l'intervalle $[r, r + h[$. Nous appelons un tel sous-ensemble "fenêtre-opérations" et nous le dénotons par w.

Algorithme 8 Algorithme générique de l'algorithme de liste ($\mathcal{R}$)

Entrées: /* Une règle de trie $\mathcal{R}$, T_{evac}^{exp} */
1: *ListOrdo* $\leftarrow \emptyset$, *ListTriée* $\leftarrow \emptyset$
2: *ListOpr* $\leftarrow \{opr_1, opr_2, ..., opr_M\}$
3: $t \leftarrow 0$, $k \leftarrow B$, $faisable \leftarrow false$
4: Trier les opérations de *ListOpr* dans la liste *ListTriée* en utilisant la règle $\mathcal{R}$
5: **Répéter**
6: Trier les opérations dans *FaisListTriée* à partir de *ListTriée*
7: **si** $|$*FaisListTriée*$| < k$ et $|$*ListTriée*$| > 0$ **alors**
8: $faisable \leftarrow$faux
9: **sinon**
10: $faisable \leftarrow$ vrai
11: Ajouter les k premières opérations dans *ListOrdo*(i,j,t) à partir de *FaisListTriée*
12: Supprimer les k opérations ajouter dans *ListOrdo* à partir de *ListOpr*, *ListTriée* et *FaisListTriée*
13: **si** t=0 **alors** $k \leftarrow 0$
14: Soit $b \in \mathcal{B}$ est le bus qui a le minimum T_{evac}^{b}
15: $t \leftarrow T_{\text{evac}}^{b}$
16: $k \leftarrow k + 1$
17: **si** event() **alors**
18: *ListTriée* $\leftarrow \emptyset$
19: Trier les opérations à partir *ListOpr* dans *ListTriée* en utilisant la règle $\mathcal{R}$
20: **finsi**
21: **finsi**
22: **finsi**
23: **Jusqu'à** $|$*ListOpr*$| = 0$ et $t > T$ et $faisable$ =vrai
24: MJArchive(*ListOrdo*)

La meilleure solution dans le voisinage $\mathcal{N}(\bar{x}, r, h)$ est calculée en minimisant la somme des risques de cette fenêtre R^w, sous les contraintes (2.72)-(2.77) et la contrainte supplémentaire suivante :

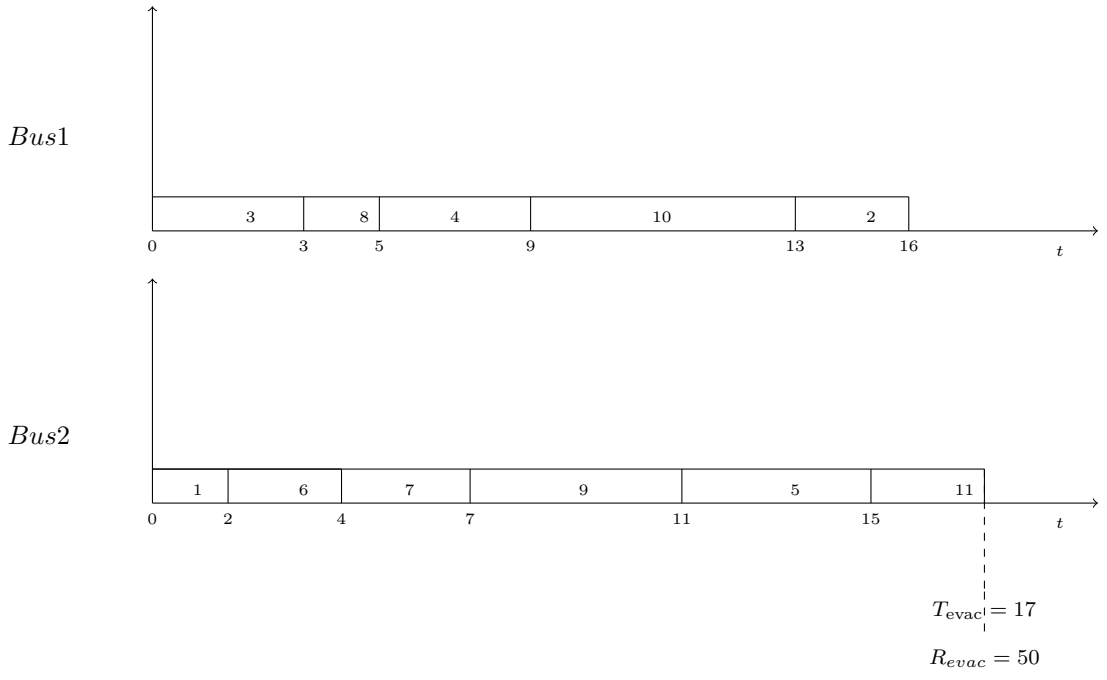
$$x_{ijt} = \bar{x}_{ijt} \qquad \forall i \notin \tilde{S}(r,h), \forall j \in \mathcal{S}, t \in [0, r[\tag{2.80}$$

Ce problème a été appelé problème de ré-optimisation de la fenêtre d'opérations (cf. section 4.4.2), et il est résolu à l'optimalité à l'aide d'un solveur mathématique comme CPLEX. La contrainte additionnelle (2.80) force l'apparition des changements dans la fenêtre d'opérations. S'il y a une amélioration de la valeur du critère R^w, alors, dans la nouvelle solution $\tilde{x}$, toutes les opérations qui ont commencé après la date $r + h$ seront décalées à gauche (décaler leurs dates de début) tout en gardant leurs anciennes positions et en respectant les contraintes de bus (2.75). La figure 2.10 présente une illustration de l'exploration du voisinage.

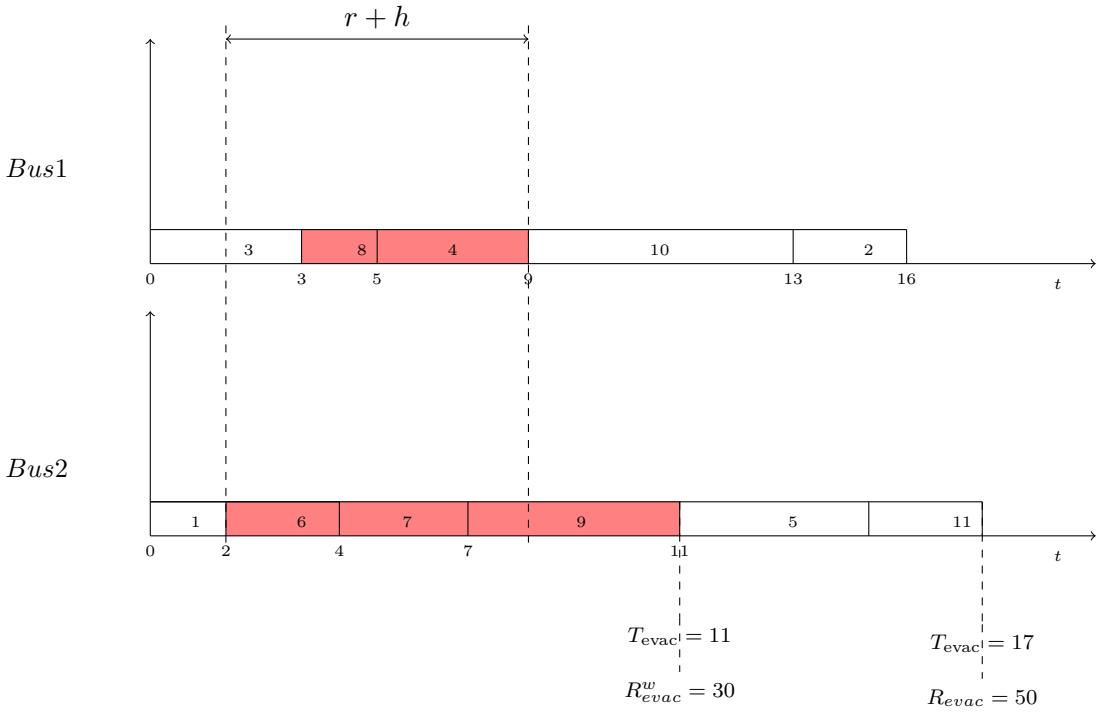
S'il n'y a pas d'amélioration de la solution, tout d'abord, nous testons en utilisant la fonction MJArchive(). Si cette solution est non dominée par une autre dans l'ensemble

nondominée, alors cette solution sera ajoutée à l'ensemble *nondominée*. Ensuite, une nouvelle fenêtre d'opérations (une nouvelle valeur de r) est sélectionnée pour qu'elle soit optimisée. On réitère le procédé pour toutes les valeurs possible de r. La recherche dans le voisinage s'arrête si le problème de ré-optimisation de la fenêtre d'opérations ne conduit pas à une meilleure solution que la solution courante ou si la limite de temps prédéfinie est expiré.

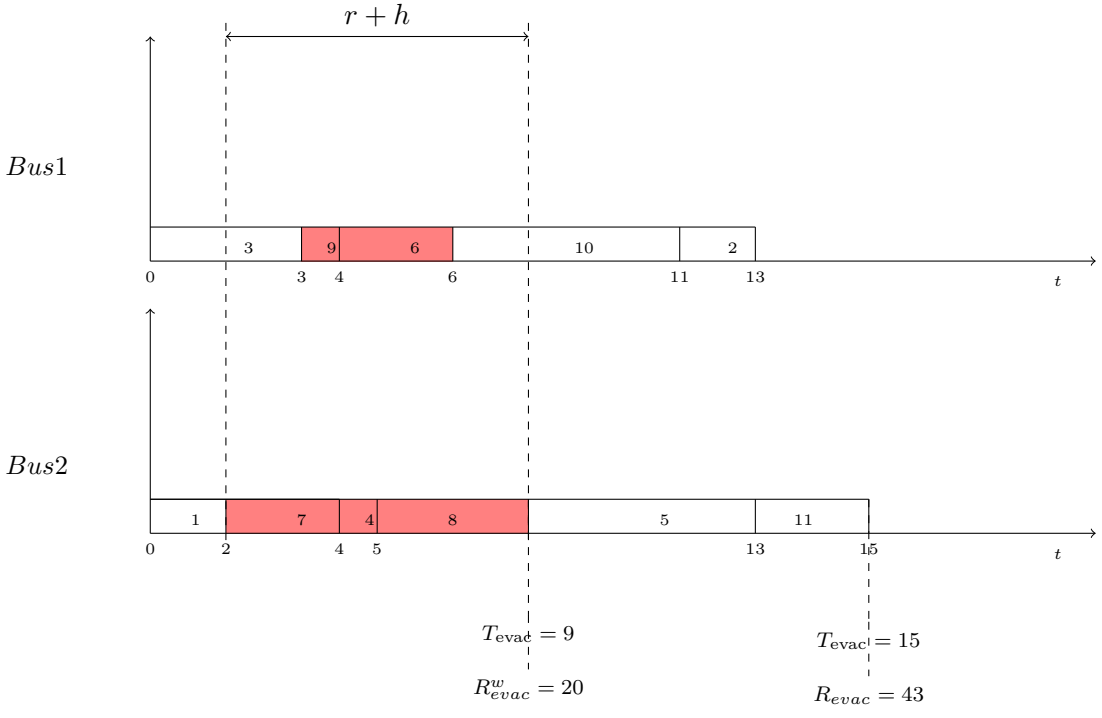
Exemple 5 *Nous considérons l'exemple de la figure 2.10. Nous disposons de deux centres de secours et de deux bus (bus 1 et bus 2). Nous voulons effectuer onze opérations d'évacuation. La solution courante est présentée dans la figure 2.10a : les opérations 3, 8, 4, 10, et 2 sont effectuées par le Bus 1 et les opérations 1, 6,7, 9, 5, et 11 sont effectuées par le Bus 2. La date de début de la fenêtre d'opérations est* $r = 2$ *et la taille de cette fenêtre est* $h = 7$ *(figure 2.10b). La date de fin d'évacuation de cette solution est* $T_{evac} = 17$*, la somme totale des risques est* $R_{evac} = 50$*, la date de fin de la fenêtre d'opérations est* $T^w_{evac} = 11$ *et la somme des risques de cette fenêtre est* $R^w_{evac} = 30$*. La solution obtenue* $\tilde{x}$ *à partir de ce voisinage, après la ré-optimisation de cette fenêtre, est donnée dans la figure 2.10c. Cette figure montre que la somme des risques, la date de fin d'évacuation de cette fenêtre, la date de fin et la somme des risques de la solution complète ont été réduites. Les opérations 5, 10, 2, et 11 ont été réordonnancées en décalant leurs dates de début à gauche et en gardant leurs positions antérieures. La somme des risques et la date de fin d'évacuation de la nouvelle solution sont respectivement* $T_{evac} = 15$ *et* $R_{evac} = 43$*.*



(a) La solution initiale $\bar{x}$



(b) La fenêtre de ré-optimisation



(c) La solution voisine $\tilde{x}$

FIGURE 2.10 – Exemple d'une fenêtre de ré-optimisation optimisée.

Les différentes étapes de la matheuristique sont illustrées dans l'Algorithme 9. La procédure Voisinage() est détaillée dans l'Algorithme 10

Algorithme 9 Algorithme de la matheuristique

Entrées:

$\bar{x}$ =< solution réalisable obtenue par l'algorithme de liste>
vect =< vecteur des dates de début de toutes les opérations $\bar{x}$ >
nondominée =<Ensemble des solutions de front de Pareto >
Répéter
  amélioré $\leftarrow false$
  $i \leftarrow 0$
  Répéter
    $r \leftarrow vect[i]$
    $\tilde{x} \leftarrow$ Voisinage$(\bar{x}, r, h)$
    si ($R_{evac}(\bar{x}) > R_{evac}(\tilde{x})$ et $T_{\text{evac}}(\bar{x}) \geq T_{\text{evac}}(\tilde{x})$) **alors**
      $\bar{x} \leftarrow \tilde{x}$
      amélioré $\leftarrow vrai$
      Pour $r \leftarrow r + h$ **à** $T_{\text{evac}} - r + h$ **faire**
        $\bar{y} \leftarrow$ Voisinage$(\bar{x}, r, h)$
        si ($R_{evac}(\bar{x}) > R_{evac}(\bar{y})$ et $T_{\text{evac}}(\bar{x}) \geq T_{\text{evac}}(\bar{y})$) **alors**
          $\bar{x} \leftarrow \bar{y}$
        sinon MJArchive$(\bar{y})$
        fin si
      fin Pour
      recalculer les valeurs de *vect* à partir de $\bar{x}$
    sinon $i \leftarrow i + 1$
      MJArchive$(\tilde{x})$
    fin si
  Jusqu'à $i \geq |vect|$ **ou** amélioré **ou** limite de temps expiré
Jusqu'à amélioré=faux **ou** limite de temps expiré

Sorties: Retourner l'ensemble des solution non-dominées *nondominée*

Algorithme 10 Procédure Voisinage

1: Calculer $\tilde{S}(r, h)$
2: Réordonnancer les opérations qui commencent dans la fenêtre de temps $[r, r + h[$ en minimisant $R^w(\bar{x})$ sous les contraintes (2.72)-(2.77), et (2.80)
3: Soit $\tilde{y}$ la solution obtenue
4: **Retourner** la solution $\tilde{y}$

6.2.3 Bornes inférieures

Dans cette section, nous présentons trois bornes inférieures au critère R_{evac} pour une instance de BBEP. La première est basée sur la relaxation continue du modèle de BBEP,

la deuxième est basée sur le problème de sac à dos et la dernière est une heuristique de liste. Notons que ces trois heuristiques peuvent être calculées en temps polynomial.

LB1 La borne inférieure la plus intuitive consiste à résoudre la relaxation continue du modèle mathématique de la section 6.1. À un nœud de l'arbre de recherche, le modèle relâché est résolu avec des variables de la séquence partielle de ce nœud qui sont déjà fixées.

LB2 Malgré le fait que le problème de sac à dos étudié dans la littérature ([Pisinger, 1995]) soit un problème de maximisation, nous pouvons utiliser une de ses bornes supérieures proposées dans la littérature afin de calculer une borne inférieure au critère la somme des risques. Nous pouvons modéliser le problème d'évacuation par bus dont l'objectif est la minimisation de la somme des risques comme un problème de sac à dos multiple avec la relaxation des contraintes de bus (2.75), comme suit :

$$\max \sum_{l \in \mathcal{P}*\mathcal{T}} \sum_{j \in \mathcal{S}} x'_{l,j} * r'_{l,j} \tag{2.81}$$

$$\sum_{l \in \mathcal{P}*\mathcal{T}} x'_{l,j} \leq cap_j \qquad \forall j \in \mathcal{S} \tag{2.82}$$

$$x'_{l,t} \in \{0,1\} \tag{2.83}$$

où toutes les données sont entières et positives. Chaque variable x'_{lj},$\forall l \in \mathcal{P}*\mathcal{T}$, $\forall j \in \mathcal{S}$, est associée à une variable x_{ijt} du modèle (IP) (6.1). De même, chaque valeur r'_{lj} est associée à une valeur du risque r_{ijt} et est définie par $r'_{lj} = C - r_{ijt}$, avec C une constante fixée vérifiant $C \gg r_{u,v,w}$, $\forall u \in \mathcal{P}$, $v \in \mathcal{S}$, $w \in \mathcal{T}$. Nous utilisons la borne supérieure proposée par [Pisinger, 1995]. Les étapes suivantes sont utilisées pour calculer cette borne inférieure pour le critère de la somme des risques :

- pour chaque nœud et pour chaque bus, nous calculons T^{opt}_{evac} pour la sous-séquence d'opérations qui est déjà ordonnancée. Ceci se fait en temps polynomial en déterminant itérativement les dates de début des opérations en respectant pour un bus donné l'ordre d'apparition dans la séquence. La valeur de T^{opt}_{evac} est calculée pour obtenir la date de début minimale des opérations qui ne sont pas encore placées.
- mettre à jour la capacité résiduelle cap_j du centre de secours j en tenant compte des opérations déjà ordonnancées.
- les variables x'_{lj} liées aux opérations non ordonnancées sont triées selon l'ordre décroissant des r'_{lj}. La première variable est choisie et fixée à 1, et toutes les variables x'_{uv} liées à la même opération sont retirées de la liste de tri et sont fixées à 0. De plus, la capacité résiduelle est mise à jour. Ce processus est réitéré jusqu'à ce que toutes les variables soient fixées.

À partir des valeurs obtenues des x'_{lj}, on peut donc déduire les valeurs des variables x_{ijt}. Ensuite LB2 est calculée en additionnant la somme des risques de ces variables et la somme des risques des variables déjà ordonnancées à ce nœud. Cette dernière somme est calculée par la fonction récursive (2.78) et (2.79).

LB3 Cette borne est calculée de la même manière que LB2 mais d'une façons plus simpliste. Ici, pour un nœud donné et pour les opérations qui restent à placer, nous calculons

$r'_i = \min_{j \in \mathcal{S}, t \in \mathcal{T}} r_{ijt}$. La borne inférieure du risque généré par les opérations non ordonnancées est donnée par $\sum r'_i$.

6.3 Expérimentations

Cette section présente les résultats expérimentaux réalisés pour évaluer l'efficacité de la formulation mathématique, les heuristiques (les algorithmes de liste et la matheuristique) et la procédure par séparation et évaluation. Toutes ces méthodes ont été testées sur les instances présentées dans la partie expérimentation de la section 4 de ce chapitre. Pour chaque instance, les méthodes suivantes ont été évaluées :

- méthodes exactes : nous avons résolu le modèle mathématique de la section 6.1 par le solveur CPLEX (12.6). Cette méthode sera nommé IP. Nous appelons B&B les solutions obtenues par la procédure par séparation et évaluation en utilisant simultanément les trois bornes inférieures LB1, LB2 et LB3, comme suit : dans les premiers niveaux de l'arbre de recherche, il est clair que LB1 est plus performante que LB2 et LB3. Pour cette raison, pour le premier niveau de l'arbre de recherche, nous calculons à chaque nœud LB1. Pour le reste des niveaux de l'arbre, pour chaque nœud nous calculons d'abord LB3. Si le nœud n'est pas supprimé alors nous calculons LB2. Cela peut conduire à couper ce nœud.

  Notons que la limite de temps et la limite mémoire imposées sont respectivement 2000 secondes et 1 GB pour chaque instance et pour chaque méthode exacte.
- heuristiques : Pour chaque solution du front de Pareto obtenue par les algorithmes de liste, nous avons testé la matheuristique avec une limite de temps égale à 600 secondes. Des résultats préliminaires antérieures ont montré que les meilleurs résultats sont obtenus avec une taille de fenêtre $h = 30$.

Dans ce qui suit, nous appelons `RS` l'ensemble de référence qui est l'ensemble des optima de Pareto obtenu par CPLEX, les ensembles approchés `MAS` et `HAS` représentent respectivement les optima de Pareto obtenues par la matheuristique et les algorithmes de liste. Afin de comparer les solutions obtenues par la matheuristique avec celles obtenues par CPLEX, nous utilisons les métriques suivantes proposées dans [Jaszkiewicz, 2004].

La première métrique a pour but d'évaluer la qualité des solutions de l'ensemble `AS` par rapport aux solutions de l'ensemble de référence `RS`. En d'autres termes, cette métrique calcule le pourcentage de solutions non dominées présentes dans l'ensemble de référence.

$$Q_1(\texttt{MAS}) = \frac{|\texttt{MAS} \cap \texttt{RS}|}{|\texttt{RS}|}$$

La deuxième métrique est le pourcentage de solutions non dominées dans l'ensemble des solutions approchées.

$$Q_2(\texttt{MAS}) = \frac{|\texttt{MAS} \cap \texttt{RS}|}{|\texttt{MAS}|}$$

La troisième métrique vise à mesurer la distance entre les solutions non dominées de l'ensemble de référence et des solutions non dominées de l'ensemble approché. En d'autres

termes, cette métrique mesure la distance moyenne entre un point de l'ensemble de référence et à un point le plus proche de l'ensemble AS :

$$Q_3(\texttt{MAS}) = \frac{1}{|\texttt{RS}|}\sum_{r\in\texttt{RS}}\min_{z\in\texttt{MAS}}\{d(z,r)\}$$

$$Q_3(\texttt{HAS}) = \frac{1}{|\texttt{RS}|}\sum_{r\in\texttt{RS}}\min_{z\in\texttt{HAS}}\{d(z,r)\}$$

avec $d(.,.)$ est la distance euclidienne dans l'espace de recherche.

La dernière métrique a pour but de calculer la dispersion des points de l' ensemble des solutions non dominées.

$$Q_4(\texttt{MAS}) = \frac{\sum_{z_i,z_{i+1}\in\texttt{MAS}}\{d(z_i,z_{i+1})\}}{|\texttt{MAS}|-1} - \frac{\sum_{r_i,r_{i+1}\in\texttt{RS}}\{d(r_i,r_{i+1})\}}{|\texttt{RS}|-1}$$

$$Q_4(\texttt{HAS}) = \frac{\sum_{z_i,z_{i+1}\in\texttt{HAS}}\{d(z_i,z_{i+1})\}}{|\texttt{HAS}|-1} - \frac{\sum_{r_i,r_{i+1}\in\texttt{RS}}\{d(r_i,r_{i+1})\}}{|\texttt{RS}|-1}$$

Le Tableau 2.22 présente les valeurs moyennes des métriques Q_3 et Q_4. Les solutions de l'ensemble de référence sont obtenues par le solveur CPLEX, les solutions approchés sont obtenues par la matheuristique ou par les algorithmes de liste. La colonne #oprs représente le nombre d'opérations d'évacuation, la colonne #centres représente le nombre de centres de secours, la colonne #bus représente le nombre de bus, et la colonne #inst_tot représente le nombre d'instances pour chaque taille du problème.

Dans le Tableau 2.22, les valeurs moyennes de $Q_3(\texttt{MAS})$ sont très petites par rapport aux valeurs moyennes de $Q_3(\texttt{HAS})$ particulièrement, dans le cas où le nombre de centres de secours est égale à 2. Ce qui signifie que la matheuristique améliore la qualité des solutions obtenues par les algorithmes de liste. D'autre part, selon la métrique Q_4, nous observons que pour toutes les instances, la matheuristique aide à répartir les points sur l'espace de recherche. En outre, la matheuristique permet d'approcher les points de l'ensemble des optima de Pareto, obtenus par les algorithmes de liste, aux points de l'ensemble de référence obtenu par CPLEX. Notons que les valeurs moyennes des métriques Q_1 et Q_2 sont toujours égales à 0.

#oprs	#centres	#bus	#inst	Q3(moy)		Q4(moy)	
				HAS	MAS	HAS	MAS
[20,35[	2	[1,2]	32	71,33	37.91	-32,82	-6,29
[35,40[	2	2	17	73,02	38.04	-26,18	-3,17
[40,60[	2	[2,3]	31	83,73	43.39	-36,98	-5,80
[40,60[	4	[2,3]	6	90,63	20,43	-92,92	-90,85
[60,70[	4	3	13	93,43	50,25	-100,56	-93,96
[70,80[	4	[3,4]	18	114,9	62,35	-88,43	-57,67
[80,90[	4	4	27	132,3	100,57	-120,6	-100,6
[90,110[	4	[4,5]	18	171,5	155,32	-147,9	-120,50

TABLE 2.22 – Évaluation de la matheuristique

Le Tableau 2.23 présente le temps CPU et le nombre de nœuds explorés par IP et B&B. La colonne IP (temps(s)) présente les temps minimaux, moyens et maximaux d'exécution

de CPLEX pendant l'énumération exacte de l'ensemble des solutions non dominées. De même, la colonne B&B représente le temps CPU de B&B pendant l'énumération des solutions. La colonne #solved_opt présente le nombre d'instances qui sont résolues à l'optimalité à la fois par IP et par B&B. En outre, le nombre de nœuds explorés est rapporté. Les résultats du Tableau 2.23 montrent que pour un nombre de centres de secours égal à 2, IP et B&B dépassent la limite de temps pour énumérer l'ensemble des solutions non dominées. Par ailleurs, pour quelques instances, IP est très efficace : il est capable d'énumérer l'ensemble des solutions non dominées pour 12 instances. Tandis que B&B est capable juste d'énumérer le front de Pareto pour 2 instances. De plus, IP explore moins de nœuds que la méthode B&B. Malheureusement, dans certains cas, quand le nombre d'opérations est supérieur à 40, à cause de la limite mémoire imposée, IP a échoué de calculer une solution réalisable. D'où, le nombre de nœuds explorés toujours égal à 0 et le temps CPU qui est inférieure à 2000 secondes. Nous observons aussi une diminution significative dans le nombre de nœuds explorés par le B&B, cela peut être interprété par le fait que la relaxation continue utilisée du IP pour le calcul de la LB1 est très lente.

À partir de Tableau 2.23, nous pouvons conclure que IP est meilleur que la B&B pour certaines instances dont le nombre de centres de secours est égal à 2. Mais pour les instances de grande taille, IP a échoué de trouver une solution réalisable à cause de la limite mémoire. Nous pouvons conclure que B&B sera peut être compétitive si d'autres bornes inférieures efficaces sont utilisées.

#oprs	#centres	#bus	#inst_tot	# solved_opt		IP (temps(s))		B&B(temps(s))		#nœuds (IP)		#nœuds(B&B)	
				IP	B&B	*moy*	*max*	*moy*	*max*	*moy*	*max*	*moy*	*max*
[20,35[	2	[1,2]	32	7	2	2000	2000	2000	2000	9781	84290	27370	139800
[35,40[	2	2	17	3	0	2000	2000	2000	2000	7930	12390	17600	84760
[40,60[	2	[2,3]	31	2	0	2000	2000	2000	2000	12260	209600	2070	115500
[40,60[	4	[2,3]	6	0	0	545	560	2000	2000	0	0	45,17	70
[60,70[	4	3	13	0	0	522,3	2000	2000	2000	0	0	27,77	32
[70,80[	4	[3,4]	17	0	0	590,5	2000	2000	2000	0	0	19,65	22
[80,90[	4	4	26	0	0	599,5	2000	2000	2000	0	0	16,58	18
[90,110[	4	[4,5]	18	0	0	636	2000	2000	2000	0	0	12,86	14

TABLE 2.23 – Comparaisons des temps d'exécution et le nombre de nœuds explorés

À travers les Tableaux de cette section, nous concluons que CPLEX est meilleur que la procédure par séparation et évaluation pour la résolution des instances de petite taille. Cette procédure nécessite une amélioration supplémentaire, par exemple, la proposition de conditions de dominance et d'autres bornes inférieures efficaces et rapides. La matheuristique proposée permet d'améliorer les solutions obtenuées par les heuristiques de liste. Donc pour des instances de grande taille et pour une limite de temps suffisante, la matheuristique est un meilleur compromis entre la qualité des solutions et le temps de calcul.

7 Conclusion du chapitre

Dans ce chapitre, nous avons présenté des méthodes exactes et heuristiques pour la résolution du problème d'évacuation par bus avec approximation du réseau routier. Nous avons abordé trois versions de ce problème : le problème d'évacuation par bus dont la fonction objectif est la minimisation de la date de fin d'évacuation, le problème robuste d'évacuation par bus et enfin le problème bicritère d'évacuation par bus où les deux critères à minimiser sont la date de fin d'évacuation et la somme des risques.

Nous avons proposé deux modélisations mathématiques pour la résolution exacte de MBEP. Nous avons comparé ces deux modélisations. Puis, nous avons renforcé la meilleure modélisation en utilisant le prétraitement et les inégalés valides. Pour la résolution approchée, des heuristiques de liste et une matheuristique ont été proposées.

Ensuite, nous avons étudié le problème robuste d'évacuation par bus. Nous avons proposé des méthodes exactes et des méthodes heuristiques pour la résolution de ce problème.

Enfin, nous avons considéré la version bicritère du problème d'évacuation par bus. Nous avons proposé une formulation mathématique ainsi qu'une procédure par séparation et évaluation. La procédure par séparation et évaluation proposée nécessite une amélioration supplémentaire, par exemple, la proposition de conditions de dominance et d'autres bornes inférieures efficaces et rapides. Cette méthode est à l'heure actuelle dans une version préliminaire.

Chapitre 3

Évacuation par bus et voiture avec réseau de transport

Dans ce chapitre, nous nous intéressons à la détermination de l'ensemble des solutions non dominées d'un problème d'évacuation multicritère où les fonctions objectifs sont la date de fin d'évacuation, la somme des risques encourus durant l'évacuation et le nombre de centres à ouvrir. Comme le calcul des plans d'évacuation se fait en amont (avant la catastrophe), une approche possible consiste à calculer la totalité des solutions de compromis (plans d'évacuation). Ici, nous utilisons une approche *à posteriori* où tous les plans d'évacuation sont calculés et ensuite, un de ces plans est choisi par le décideur.

Dans la section 2 nous présentons le problème d'évacuation de manière formelle. Puis, nous proposons dans la section 3, la modélisation mathématique en nombre entiers de ce problème. Les sections 3 et 4 présentent les différentes heuristiques proposées pour la résolution de ce problème en un temps de calcul raisonnable. La section 6 présente une méthode d'agrégation, permettant de réduire la taille du graphe. Enfin, la section 7 est consacrée aux résultats expérimentaux.

1 Contribution du chapitre

Dans ce chapitre, nous avons abordé un problème multicritère d'évacuation par bus et voitures. L'objectif est de choisir les centres de secours à ouvrir afin de minimiser la date de fin d'évacuation et la somme des risques encourus durant l'opération d'évacuation. Pour la résolution exacte des instances de petite taille, nous avons proposé une modélisation mathématique en nombres entiers. Pour résoudre les instances réelles de Nice et de Kaiserslautern, nous avons développé un algorithme génétique et une heuristique par décomposition. Ces travaux ont donné lieu à une communication dans une conférence nationale ([Deghdak and T'kindt, 2015b]), à une communication dans une conférence internationale ([Deghdak and T'kindt, 2015a]) et à une publication dans une revue internationale ([Goerigk et al., 2014a]).

2 Présentation du problème et positionnement scientifique des contributions

Nous reprenons la majorité des notations utilisées dans le chapitre 2 : $\mathcal{P} = \{1, .., P\}$ est l'ensemble des points de rassemblement des personnes localisées dans la zone sinistrée, et $\mathcal{S} = \{1, ..., S\}$ est l'ensemble des centres de secours. Chaque centre $j \in \mathcal{S}$ est défini par ses coordonnées géographiques, et des capacités maximales notées c_j^S et c_j^P où $c_j^{\mathcal{S}}$ représente le nombre de personnes que le centre j peut accueillir et $c_j^{\mathcal{P}}$ est l'espace de stationnement disponible pour les voitures se rendant au centre. Nous disposons d'un ensemble de bus $\mathcal{B} = \{1, .., B\}$ de même capacité pour évacuer les personnes à partir des points de rassemblement vers les centres de secours. Initialement, tous les bus sont localisés à un même dépôt.

Soit $\mathcal{G}' = (\mathcal{N}, \mathcal{A}')$ un graphe orienté qui modélise le réseau de transport réel. $\mathcal{N}$ est l'ensemble des nœuds et $\mathcal{A}'$ est l'ensemble des arcs de ce graphe. Un nœud $n_i \in \mathcal{N}$ peut représenter un carrefour, un centre de secours $i \in \mathcal{S}$ ou un point de rassemblement $j \in \mathcal{P}$. Un arc $a_{i,j} = (n_i, n_j)$ signifie qu'un flot de bus et/ou de voitures peut aller d'un nœud n_i vers un nœud n_j. De plus, à chaque arc $a_{i,j}$ est associée une capacité $c_{i,j}^v$ (le nombre maximum de voitures pouvant utiliser cet arc à un instant donné t) et deux valeurs r_{ij} et p_{ij} qui représentent respectivement le risque de passage par cet arc et son temps de traversée. Puisqu'un bus prend plus de places sur un arc qu'une voiture, alors nous supposons que nous avons un ratio α entre la capacité d'un arc occupé par une voiture et la capacité d'arc occupé par un bus (par exemple un bus est égale à deux voitures). L'horizon temporel nécessaire pour l'évacuation est noté par $\mathcal{T}$.

Nous supposons que les personnes arrivent aux points de rassemblement graduellement : certaines personnes préfèrent être évacuées en famille, ou ne réalisent pas immédiatement qu'elles doivent être évacuées, ou bien doivent d'abord atteindre des points de rassemblement éloignés pour qu'elles puissent être évacuées. Pour chaque point de rassemblement et à chaque instant de temps t, nous supposons connu le nombre de personnes arrivant à pied ou par voitures à ce point de rassemblement. Le nombre de personnes qui arrivent à pied à un point de rassemblement $i \in \mathcal{P}$ à instant t est noté par e_{it}^b, ces personnes seront évacuées par bus. Le nombre de voitures qui arrivent à un point de ras-

semblent $i \in \mathcal{P}$ à l'instant t est noté par e^v_{it}. Les conducteurs de voitures se rendent à un point de rassemblement afin de récupérer les consignes d'évacuation.

Dans ce qui suit, nous expliquons en détails les hypothèses de ce problème.

Initialement, les bus sont localisés au dépôt où ils seront préparés pour l'évacuation et les plans sont communiqués à leurs conducteurs. Dans le cas d'une évacuation à grande échelle le nombre de bus est significativement petit par rapport au nombre de personnes à évacuer. Pour cette raison, un bus est amené à faire plusieurs trajets depuis les points de rassemblement vers les centres de secours. En outre, chaque bus peut passer par plusieurs points de rassemblement pour récupérer les personnes. Si le bus est plein, alors il part directement au centre de secours.

Nous supposons que le planificateur a le contrôle sur les conducteurs de bus et de voitures. En d'autres termes, les conducteurs respectent les instructions sur les plans d'évacuation qui leurs été fournis. Dans la littérature des problèmes d'évacuation par voiture, le choix des routes se fait selon l'un des deux principes de Wardrop ([Wardrop, 1952])(cf. chapitre 1). Le premier principe est l'*équilibre d'usager* où les conducteurs de voitures choisissent les chemins qui optimisent leurs fonctions objectifs, par exemple choisir le chemin le plus court pour quitter la zone endommagée. Le deuxième principe est l'*optimum social* où tous les conducteurs de voitures partagent le réseau routier afin d'optimiser une ou plusieurs fonctions objectifs globales. Dans notre cas, nous avons opté pour le principe d'*optimum social*, puisque : (a) les voies de transport public sont déterminées par le planificateur d'évacuation ; (b) le choix des routes pour les voitures peut être influencé par les forces de l'ordre et les informations de guidage du trafic ; et (c) les résultats d'optimisation obtenus peuvent être complétés et validés par des modèles de simulation (la septième tâche du projet DSS_Evac).

Dans la plupart des problèmes d'évacuation le critère à minimiser est la date de fin d'évacuation, notée T_{evac}. Cependant, le risque sur les routes est important, notamment dans le cas d'un séisme où des effondrements éventuels des bâtiments sur les routes sont possibles. Cela peut réduire les capacités de ces routes et aussi causer un risque sur la vie des évacués. À partir de ce raisonnement, nous considérons la somme des risques comme un second objectif de notre problème, notée R_{evac} : R_{evac} est la somme des valeurs de risque associées à tous les chemins utilisés durant l'évacuation. Par définition, le risque d'un chemin est la somme des risques des arcs de ce chemin multiplié par le nombre de personnes traversant ce chemin. Le troisième critère est le nombre de centres nécessaires à ouvrir pour évacuer les personnes.

Une fois que la localisation des centres de secours à ouvrir est choisie, les bus et les voitures sont routés vers ces centres. Les voitures sont modélisées par des flots dynamiques et les bus par des flots à commodité multiples où chaque bus représente une commodité.

Ce problème d'évacuation sera noté par GEP (*General Evacuation Problem*).

3 Un programme mathématique en nombres entiers

Données

- B : le nombre de bus,
- T : l'horizon du temps,
- $\mathcal{A}'$: l'ensemble des arcs,
- $\mathcal{N}$: l'ensemble des nœuds,
- $\mathcal{P}$: l'ensemble des points de rassemblement,
- $\mathcal{N}^V \subseteq \mathcal{P}$: l'ensemble des points de rassemblement où les voitures arrivent,
- $\mathcal{N}^B \subseteq \mathcal{P}$: l'ensemble des points de rassemblement où les piétons arrivent,
- $\mathcal{S}$: l'ensemble des centres de secours,
- C : la capacité de bus,
- e^b_{it} : le nombre d'évacués (bus) arrivant au point de rassemblement i à l'instant t,
- e^v_{it} : le nombre d'évacués (voitures) arrivant au point de rassemblement i à l'instant t,
- c^S_j : la capacité du centre de secours j,
- c^P_j : la capacité de parking du centre de secours j,
- α : le ratio entre la capacité d'arc occupée par une voiture et la capacité d'arc occupée par un bus,
- p_{ij} : le temps de traversée de l'arc (i, j),
- r_{ij} : le risque associé à l'arc (i, j),
- c^a_{ij} : la capacité de l' arc (i, j).

Variables

- T_{evac} : la date de fin d'évacuation,
- R_{evac} : la somme des risques pour tous les évacués,
- $S_{\max}$: le nombre de centres secours qui peuvent être ouverts au maximum,
- f_{ijt} : le nombre d'évacués (voitures) sur l' arc (i, j) arrivant au nœud j à l'instant t ,
- g_{ijbt} : le nombre d'évacués par le bus b sur l'arc (i, j) arrivant au nœud j à l'instant t,
- x_{ijbt} : =1, si le bus b arrive au nœud j à l'instant t après avoir traversé l' arc (i, j), et = 0 sinon,
- y_i : =1, si le centre de secours i est ouvert, et =0 sinon,
- z_{ijt} : =1 s'il y a un trafic sur l'arc (i,j) à l'instant t, et 0 sinon,
- b_{ibt} : nombre de personnes évacuées par bus b à partir du nœud i à l'instant t,
- v_{it} : le nombre de personnes démarrant par voiture du nœud i à l'instant t,
- w^b_{ibt} : le nombre de personnes déposées par le bus b au nœud i à l'instant t,
- w^v_{it} : le nombre d'évacués arrivant au centre de secours i par voiture à l'instant t.

Objectif

$$\min\ (T_{\text{evac}}, R_{evac}, S_{max}) \tag{3.1}$$

Contraintes

$$T_{\text{evac}} \geq t z_{ijt} \qquad \forall (i,j) \in \mathcal{A}', t \in \mathcal{T} \tag{3.2}$$

$$T_{\text{evac}} \geq t x_{ijbt} \qquad \forall (i,j) \in \mathcal{A}', t \in \mathcal{T}, b \in \mathcal{B} \tag{3.3}$$

$$S_{max} = \sum_{i \in \mathcal{S}} y_i \tag{3.4}$$

$$R_{evac} \geq \sum_{t \in \mathcal{T}} \sum_{(i,j) \in \mathcal{A}'} r_{ij} \left(f_{ijt} + \sum_{b \in \mathcal{B}} g_{ijbt} \right) \tag{3.5}$$

$$\sum_{i:(i,j) \in \mathcal{A}'} \sum_{s=0}^{t} x_{ijbs} \geq \sum_{l:(j,l) \in \mathcal{A}'} \sum_{s=0}^{t+p_{ij}} x_{jlbs} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T}, j \in \mathcal{N} \tag{3.6}$$

$$\sum_{i:(0,i) \in \mathcal{A}'} \sum_{t \in \mathcal{T}} x_{0ibt} \leq 1 \qquad \forall b \in \mathcal{B} \tag{3.7}$$

$$v_{jt} + \sum_{i:(i,j) \in \mathcal{A}'} f_{ijt} = \sum_{l:(j,l) \in \mathcal{A}'} f_{jl(t+p_{jl})} \qquad \forall t \in \mathcal{T}, j \in \mathcal{N} \diagup \mathcal{S} \tag{3.8}$$

$$\sum_{i:(i,j) \in \mathcal{A}'} f_{ijt} - w^v_{jt} = \sum_{l:(j,l) \in \mathcal{A}'} f_{jl(t+p_{jl})} \qquad \forall t \in \mathcal{T}, j \in \mathcal{S} \tag{3.9}$$

$$\alpha f_{ijt} \leq c^a_{ij} z_{ijt} \qquad \forall t \in \mathcal{T}, (i,j) \in \mathcal{A}' \tag{3.10}$$

$$\sum_{s=1}^{t} v_{js} \leq \sum_{s=1}^{t} e^v_{js} \qquad \forall t \in \mathcal{T}, j \in \mathcal{N}^V \tag{3.11}$$

$$\sum_{t \in \mathcal{T}} v_{jt} = \sum_{t \in \mathcal{T}} e^v_{jt} \qquad \forall j \in \mathcal{N}^V \tag{3.12}$$

$$b_{jkt} + \sum_{i:(i,j) \in \mathcal{A}'} g_{ijbt} = \sum_{l:(j,l) \in \mathcal{A}'} g_{jlb(t+p_{jl})} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T}, j \in \mathcal{N} \diagup \mathcal{S} \tag{3.13}$$

$$\sum_{i:(i,j) \in \mathcal{A}'} g_{ijbt} - w^b_{jbt} = \sum_{l:(j,l) \in \mathcal{A}'} g_{jlb(t+p_{jl})} \qquad \forall b \in \mathcal{B}, t \in \mathcal{T}, j \in \mathcal{S} \tag{3.14}$$

$$g_{ijkt} \leq C x_{ijbt} \qquad \forall k \in \mathcal{B}, t \in \mathcal{T}, (i,j) \in \mathcal{A}' \tag{3.15}$$

$$\sum_{s=1}^{t} \sum_{k \in \mathcal{B}} b_{jks} \leq \sum_{s=1}^{t} e^b_{js} \qquad \forall t \in \mathcal{T}, j \in \mathcal{N}^B \tag{3.16}$$

$$\sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{B}} b_{jkt} = \sum_{t \in \mathcal{T}} e^b_{jt} \qquad \forall j \in \mathcal{N}^B \tag{3.17}$$

$$\sum_{t \in \mathcal{T}} w^v_{jt} \leq c^P_j y_j \qquad \forall j \in \mathcal{S} \tag{3.18}$$

$$\sum_{t \in \mathcal{T}} (w^v_{jt} + \sum_{k \in \mathcal{B}} w^b_{jkt}) \leq c^S_j y_j \qquad \forall j \in \mathcal{S} \tag{3.19}$$

$$\alpha f_{ijt} + \sum_{b \in \mathcal{B}} x_{ijbt} \leq c^a_{ij} \qquad \forall t \in \mathcal{T}, (i,j) \in \mathcal{A}' \tag{3.20}$$

$$f_{ijt} \geq 0 \qquad \forall t \in \mathcal{T}, (i,j) \in \mathcal{A}' \tag{3.21}$$

$$g_{ijkt} \geq 0 \qquad \forall k \in \mathcal{B}, t \in \mathcal{T}, (i,j) \in \mathcal{A}' \tag{3.22}$$

$$x_{ijkt} \in \mathbb{B} \qquad \forall k \in \mathcal{B}, t \in \mathcal{T}, (i,j) \in \mathcal{A}' \tag{3.23}$$

$$z_{ijt} \in \mathbb{B} \qquad t \in \mathcal{T}, (i,j) \in \mathcal{A}' \tag{3.24}$$

$$b_{ikt} = 0 \qquad i \in \mathcal{N} \diagup \mathcal{N}^B, \forall k \in \mathcal{B}, t \in \mathcal{T} \tag{3.25}$$

$$v_{it} = 0 \qquad i \in \mathcal{N} \diagup \mathcal{N}^V, \forall t \in \mathcal{T} \tag{3.26}$$

$$w^b_{ikt} = 0 \qquad \forall k \in \mathcal{B}, t \in \mathcal{T}, i \in \mathcal{N} \diagup \mathcal{S} \tag{3.27}$$

$$w^v_{it} = 0 \qquad \forall t \in \mathcal{T}, i \in \mathcal{N} \diagup \mathcal{S} \tag{3.28}$$

$$y_i \in \mathbb{B} \qquad \forall i \in \mathcal{S} \tag{3.29}$$

$$T_{\text{evac}} \geq 0 \tag{3.30}$$

$$R \geq 0 \tag{3.31}$$

La fonction objectif (3.1) indique qu'il faut minimiser la date de fin d'évacuation T_{evac}, la somme des risques R_{evac} et le nombre de centres de secours à ouvrir S_{max}. Ces trois fonctions objectifs sont calculées en utilisant les contraintes (3.2), (3.3), (3.4) et (3.5). Les contraintes (3.2) et (3.3) assurent que T_{evac} est la date d'évacuation maximale. La valeur du risque R_{evac} dépend du nombre de personnes qui passent par un arc. Cette relation est exprimée par les contraintes (3.5). Les contraintes (3.6) assurent que dans le cas des tournées de bus, un bus peut quitter un nœud déjà visité. Les contraintes (3.7) assurent que tous les bus débutent l'évacuation à partir d'un dépôt d. Le nombre de centres de secours qui seront utilisés est déterminé par les contraintes (3.4).

Le trafic des voitures est modélisé par les contraintes (3.8) – (3.12). Les contraintes (3.8) et (3.9) sont des contraintes de conservation de flots des voitures. Les contraintes (3.10) assurent que les variables $z_{ijt} = 1$ à un instant de temps t, s'il y a un trafic de voitures sur l'arc (i,j). Le début d'évacuation des voitures est modélisé par les contraintes (3.11) et (3.12). Les contrainte (3.11) assurent que le nombre de personnes évacuées jusqu'à la date t est inférieur à celui des personnes qui apparaissaient jusqu'à la date t. Les contraintes (3.12) assurent que toutes les personnes sont évacuées.

De façon similaire à l'évacuation par voiture, nous avons aussi des flots de passagers utilisant les bus. Ces flots sont modélisés par les contraintes (3.13) – (3.17). Les contraintes (3.13) et (3.14) sont des contraintes de conservation des flots de bus. Les contraintes (3.15) sont des contraintes de capacité d'un bus. Elles assurent aussi que les passagers affectés à un bus seront sur ce bus effectivement. Les contraintes sur les dates de début d'évacuation par bus (3.16) et (3.17) sont analogues à celles des contraintes (3.11) et (3.12).

Les flots de voitures et de bus sont reliés entre eux par des contraintes de capacité sur les arcs (3.20). Les contraintes (3.18) et (3.19) sont des contraintes de capacité des centres de secours. Enfin, les évacués sont récupérés à un point de rassemblement et seront déposés ensuite à un centre de secours. Ceci est modélisé par les contraintes (3.25) – (3.28).

Notons que le problème GEP est $\mathcal{NP}$-difficile (il contient le sous-problème d'évacuation par bus [Goerigk et al., 2013, Goerigk et al., 2014b]).

Puisque la modélisation en nombres entiers proposée ne permet pas de résoudre des instances réelles de grande taille, nous proposons deux types d'heuristiques : un algorithme génétique et une heuristique par décomposition. Les étapes de ces heuristiques seront détaillées dans les sections suivantes.

4 Algorithme génétique

La classe des métaheuristiques incontestablement la plus employée en optimisation multicritères est celle des algorithmes évolutionnaires en raison de leur possibilité de calculer un ensemble de solutions en une seule itération. Plusieurs variantes de l'algorithme génétique ont été proposées pour la résolution des problèmes d'optimisation multicritères (voir [Konak et al., 2006]). Ici, nous présentons la méthode NSGA-II ([Deb et al., 2002]) pour résoudre le problème GEP, reconnue comme étant d'une grande efficacité. Elle est aujourd'hui l'une des méthodes de références pour l'optimisation multicritères évolutionnaire.

4.1 Représentation d'une solution

Dans cette section, nous nous intéressons au codage d'une solution dite chromosome. L'idée principale est de pré-calculer l'ensemble des chemins possibles, et d'affecter ces chemins aux unités de flots discrets de bus et de voitures.

Pour chaque (i, j) dans $(\mathcal{N}^V \cup \mathcal{N}^{\mathcal{B}}) \times \mathcal{S}$, et pour chaque (i', j') dans $(\mathcal{S} \cup \{d\}) \times \mathcal{N}^B$, nous déterminons l'ensemble des chemins candidats *Paths*. L'ensemble des chemins $(\mathcal{N}^V \cup \mathcal{N}^B) \times \mathcal{S}$ représente le trafic entre les points de rassemblement et les centres de secours. L'ensemble $(\mathcal{S} \cup \{d\}) \times \mathcal{N}^B$ modélise les routes utilisées par les bus depuis les centres de secours vers les points de rassemblement, et contient aussi le premier trajet des bus depuis le dépôt vers les points de rassemblement.

La détermination de l'ensemble des optima de Pareto est obtenue en minimisant la date de fin d'évacuation, la somme des risques et en résolvant sur $(\mathcal{N}, \mathcal{A}')$ plusieurs problèmes de plus court chemin sous les contraintes de ressources en ne tenant compte que de la distance. En utilisant les chemins *Paths*, une solution pour GEP est représentée par un "paquet de flots".

Pour le trafic des voitures, un paquet de flots F^V est donné par (i, t, s), où $i \in \mathcal{N}^V$ est son point de départ, t est sa date de départ au plus tôt, et s dénote sa taille (le nombre de voitures de ce paquet). Nous dénotons par $\mathcal{F}^V$ l'ensemble des paquets de flots de voitures. Un paquet de flots de voitures est généré s'il y a des voitures arrivant en même temps à un point de rassemblement. En d'autre termes, un paquet est un groupe de véhicules partant au même instant du même point de rassemblement. Par exemple, si 3 voitures apparaissent à un nœud i à la date 0, et 2 voitures apparaissent au nœud i à la date 1, nous les modélisons en utilisant deux paquets de flots $(i, 0, 3)$ et $(i, 1, 2)$. Une solution pour le trafic de voitures est donnée par une affectation de chaque paquet $(i, t, s) \in \mathcal{F}^V$ à un centre de secours $j \in \mathcal{S}$ en passant par un chemin menant à ce centre.

Puisqu'un bus peut faire plusieurs allers et retours depuis et vers les centres de secours, la représentation de la solution pour le trafic de bus est légèrement plus élaborée. Comme dans le cas du trafic de voitures, un paquet de flots de bus est noté $F^B = (i, t, s)$ et l'ensemble de ces paquets est noté $\mathcal{F}^B$. Chaque paquet de flots de bus n'est généré que si le nombre de personnes est égale à la capacité d'un bus. Une solution du problème d'évacuation par bus est donnée par une séquence de paquets de bus. Chaque bus (paquet de flot) utilise un chemin pour partir d'un point de rassemblement vers un centre de

secours.

Notons que la représentation de la solution de GEP est réalisable, si est seulement si, la capacité totale des centres est suffisante. Aussi, nous avons besoin d'ordonnancer chaque paquet en déterminant sa date de départ et sa date d'utilisation d'un arc. Dans ce cas, les contraintes de capacité peuvent éventuellement interdire un paquet d'utiliser un chemin pré-affecté. Pour cela, nous mettons en œuvre la stratégie suivante pour générer le chromosome à partir de cette solution. Pour chaque paquet, nous calculons le temps d'arrivée prévu en supposant que le calcul d'itinéraire n'est pas entravé par les conflits de capacité. Les paquets de bus $\mathcal{F}^B$ et les paquets de voiture $\mathcal{F}^V$ sont triés selon l'ordre décroissant de leurs dates d'arrivées. Ensuite, les paquets de bus sont ordonnancés en premier, suivis par les paquets de voitures en utilisant une méthode gloutonne. Cette dernière, consiste à router les paquets selon l'ordre décroissant de leur date d'arrivé prévue. Quand un arc à instant de temps n'a pas une capacité suffisante, ce paquet doit attendre sur le nœud courant jusqu'à ce que la capacité nécessaire soit disponible à nouveau.

La figure 3.1 illustre comment une solution d'une instance du GEP est codée. Nous avons un ensemble de paquet de flot de bus $\mathcal{F}^B = \{F_1^B, F_2^B\}$ et un ensemble de paquets de flot de voitures $\mathcal{F}^V = \{F_1^V, F_2^V\}$. Nous disposons d'un seul bus pour évacuer les personnes. Les paquets F_1^B et F_2^B contiennent 20 personnes et l'évacuation de ces personnes s'effectue par bus à partir des points de rassemblement P_1 et P_2 . Les dates d'évacuation au plutôt des paquets F_1^B et F_1^B sont respectivement 0 et 2. Les paquets F_1^V et F_2^V contiennent respectivement 1 et 2 voitures et leur évacuation s'effectue à partir de P_1 et P_2. Une solution d'une instance du GEP consiste à déterminer pour chaque paquet de flot x, sa date de début d'évacuation $t(x)$, le centre de secours de rattachement $S(x)$ et le chemin $C(x)$ menant à ce centre.

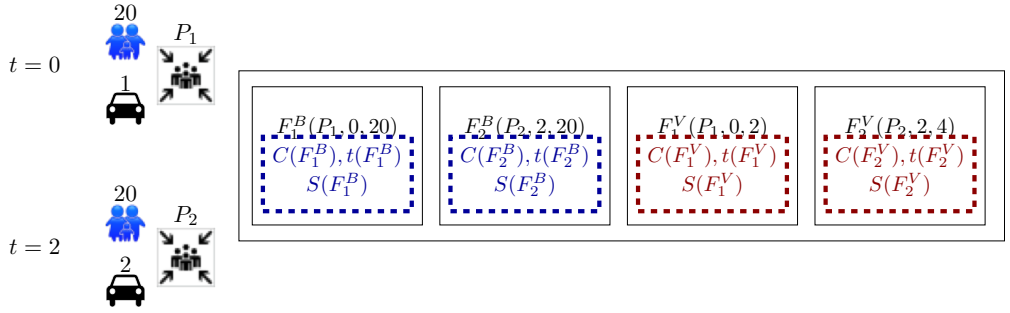


FIGURE 3.1 – Représentation d'une solution du GEP

4.2 Mécanismes de l'algorithme génétique

Dans cette section, nous décrivons les mécanismes de base de l'algorithme proposé qui suivent les étapes directrices de la méthode NSGA-II ([Deb et al., 2002]).

Population initiale

La population de base qui contient des solutions réalisables est générée de la manière suivante. Tout d'abord, nous déterminons un nombre minimal de centres de secours n_S nécessaires pour l'hébergement des personnes, en utilisant un modèle mathématique en nombres entiers. Nous utilisons la variable binaire x_j qui est égale à 1 si un centre de secours est ouvert, et 0 dans le cas contraire. Le modèle est le suivant :

$$
\begin{aligned}
\min \ & \sum_{j \in \mathcal{S}} x_j & & \\
\text{s.c.} \ & \sum_{j \in \mathcal{S}} y^I_{pj} = 1 & & \forall p \in \mathcal{F}^V \\
& \sum_{j \in \mathcal{S}} y^B_{pj} = 1 & & \forall p \in \mathcal{F}^B \\
& \sum_{p=(i,t,s) \in \mathcal{F}^V} sy^I_{pj} \leq c^P_j x_j & & \forall j \in \mathcal{S} \\
& \sum_{p=(i,t,s) \in \mathcal{F}^V} sy^I_{pj} + \sum_{p=(i,t,s) \in \mathcal{F}^B} sy^B_{pj} \leq c^S_j x_j & & \forall j \in \mathcal{S} \\
& x, y^B, y^I \in \mathbb{B} & &
\end{aligned}
$$

La valeur optimale de la fonction objectif représente le nombre minimal de centres qui seront ouverts pour accueillir tous les évacués.

Ensuite, nous générons quatre affectations différentes vers les centres de secours. Quand n_S centres sont utilisés, la première affectation minimise la somme des risques, la deuxième minimise la date de fin d'évacuation. La troisième et la quatrième minimisent respectivement la somme des risques et la date de fin d'évacuation lorsque n_S+1 centres sont utilisés. Pour chaque affectation et pour chaque paquet, les chemins sont choisis aléatoirement de l'ensemble des chemins pré-calculés. Le modèle suivant permet de calculer ces affectations :

$$
\begin{aligned}
\min \ & \sum_{j \in \mathcal{S}} \left(\sum_{p \in \mathcal{F}^V} val_{pj} y^I_{pj} + \sum_{p \in \mathcal{F}^B} val_{pj} y^B_{pj} \right) & & \\
\text{s.c.} \ & \sum_{j \in \mathcal{S}} y^I_{pj} = 1 & & \forall p \in \mathcal{F}^V \\
& \sum_{j \in \mathcal{S}} y^B_{pj} = 1 & & \forall p \in \mathcal{F}^B \\
& \sum_{p=(i,t,s) \in \mathcal{F}^V} sy^I_{pj} \leq c^P_j x_j & & \forall j \in \mathcal{S} \\
& \sum_{p=(i,t,s) \in \mathcal{F}^V} sy^I_{pj} + \sum_{p=(i,t,s) \in \mathcal{F}^B} sy^B_{pj} \leq c^S_j x_j & & \forall j \in \mathcal{S} \\
& \sum_{j \in \mathcal{S}} x_j \leq n_S & & \\
& x, y^B, y^I \in \mathbb{B} & &
\end{aligned}
$$

où, n_S est une constante du modèle déterminant le nombre maximal de centres ouverts. La valeur val_{pj} peut présenter la distance du plus court chemin utilisé par le paquet p pour arriver au centre j, ou bien la valeur du chemin le plus sûr conduisant au centre j.

La raison pour laquelle nous avons commencé avec une population initiale dont ses solutions ont un nombre de centres minimal est qu'il est plus "facile" pour l'algorithme génétique d'ajouter un centre additionnel, que de fermer un centre de secours : l'opération d'ouverture nécessite seulement le routage d'un paquet vers ce nouveau centre, tandis que la fermeture d'un centre nécessite la redirection de tous ses paquets vers d'autres centres de secours.

À une itération τ, nous dénotons la population courante par P_τ dont le cardinale est L. En utilisant la mutation et le croisement, nous déterminons une nouvelle population P'_τ constituée de $2L$ solutions candidates. À partir de ces $2L$ solutions, nous formons la nouvelle génération $P_{\tau+1}$ en utilisant les mécanismes suivants.

Évaluation

Soit une population d'individu P'_τ, nous avons besoin d'une fonction d'évaluation qui détermine pour chaque individu sa probabilité d'existence dans la prochaine génération. Cette fonction utilise un classement de Pareto selon les critères : la somme des risques, la date de fin d'évacuation et le nombre de centres ouverts. Elle se déroule en deux phases : une phase d'initialisation suivie d'une phase d'affectation des rangs. Durant la phase d'initialisation, à chaque individu i de la population P'_τ sont associés un compteur de domination α_i donnant le nombre d'individus qui dominent i et l'ensemble des individus S_i dominés par i. Les individus pour lesquels α_i est nul constituent l'ensemble des individus non dominés de rang 1, noté F_1.

Après la phase d'affectation des rangs de non domination pour tous les individus de la population, nous supposons que l'ensemble F_k des individus non dominés de rang k a été construit. Il est possible de déterminer l'ensemble des solutions nondominées F_{k+1} dont le rang est $k+1$ comme suit : pour tous individu i appartenant à F_k, les compteurs α_j des individus j dominés par i sont décrémentés. Les individus j pour lesquels $\alpha_j = 0$ constituent l'ensemble F_{k+1}.

Sélection

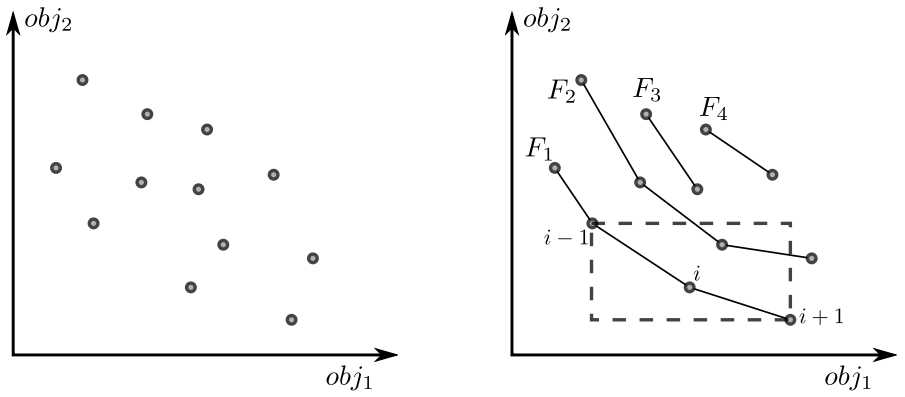
À la phase d'évaluation succède la phase de sélection. Après avoir trié les individus de la population en rangs selon les critères à optimiser, nous déterminons pour chaque individu i de la population P'_τ une distance de surpeuplement d_i (crowding distance). Cette distance est relative sur chaque dimension (critère). Elle est calculée entre chaque individu i nondominé du rang r_i et les individus nondominés du même rang.

Pour créer $P_{\tau+1}$, les solutions sont choisies en utilisant le "tournoi de surpeuplement" (([Deb et al., 2002])) qui est fondé sur un opérateur de comparaison dit de "superpeuplement ", noté $\prec_n$. À chaque individu i de rang r_i ayant une distance de superpeuplement d_i, l'opérateur de comparaison de surpeuplement est défini de la façon suivante :

$$i \prec_n j \Longleftrightarrow r_i < r_j \; ou \; (r_i = r_j \; et \; d_i > d_j)$$

Le tournoi de surpeuplement entre deux individus i et j sélectionne i si $i \prec_n j$. Ce tournoi est conçu pour favoriser la sélection d'individus de même rang de domination dans les zones peu denses de l'espace des objectifs.

Le tri est illustré dans la figure 3.2 pour deux critères. Les valeurs des fonctions objectifs des solutions courantes sont indiquées dans la figure 3.2a. Le rang correspondant aux fronts $F_1, \ldots, F_4$ est donné par la figure 3.2b. La zone en pointillée indique la méthode de calcul de la distance de surpeuplement de la solution i en utilisant les deux voisins $i-1$ et $i+1$ du même front F_1.



(a) population non triée. (b) population triée.

FIGURE 3.2 – Méthode de classement utilisée dans l'algorithme génétique.

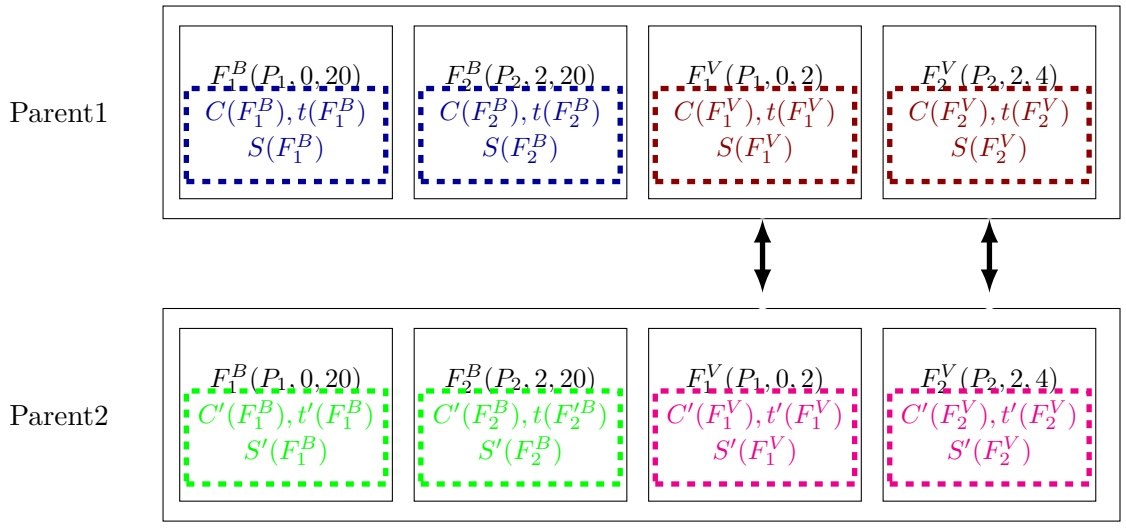
Pour créer les chromosomes de la population fille, nous utilisons les opérateurs de croisement et de mutation

Croisement et mutation

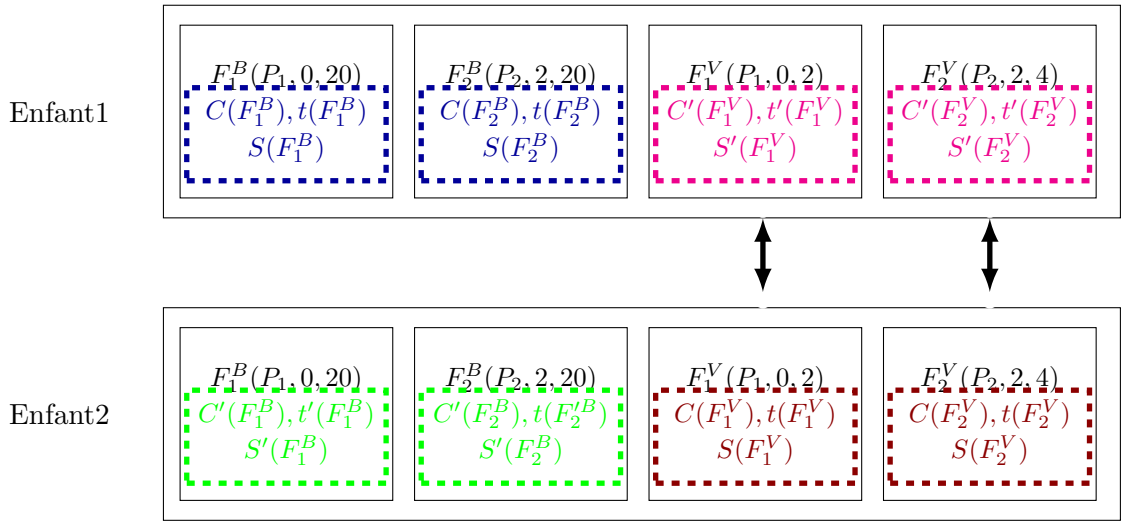
Pour deux solutions parents de la population, nous générons deux nouvelles solutions en interchangeant respectivement le trafic des voitures des deux solutions parents. La figure 3.3 illustre l'opération de croisement. Ensuite, nous mutons aléatoirement les gènes de ces solutions en utilisant le voisinage suivant : (a) pour chaque paquet de voitures, un centre différent et un chemin différent parmi les chemins de l'ensemble *Paths* peuvent être affectés, (b) Pour chaque paquet de bus, un centre différent et un chemin différent, appartenant à l'ensemble *Paths*, peuvent être affectés. Si un centre différent est choisi et que le bus fait un autre trajet après le trajet qui vient d'être changé, alors aussi le chemin qui mène vers le prochain paquet doit être changé. (c) Un paquet de bus peut être affecté à n'importe quel bus et à n'importe quelle position dans l'ordonnancement d'un bus. Comme pour le cas (b), des ajustements sur les trajectoires peuvent devenir nécessaires.

Recherche locale

Enfin, les solutions sont améliorées en utilisant une méthode de recherche locale analogue à la mutation aléatoire. Pour réduire le temps de calcul, la recherche locale est



(a) Deux individues parents



(b) Deux individues decendants

FIGURE 3.3 – Croisement

effectuée pour un nombre d'itérations pré-défini ou bien lorsqu'un optimum local est atteint.

L'algorithme 11 récapitule les aspects principaux de l'algorithme génétique proposé.

5 Heuristique par décomposition

Pour définir des plans d'évacuation avec des temps de calcul acceptables pour des instances de grande taille, nous avons proposé une heuristique par décomposition. Nous supposons dans cette version que l'évacuation des voitures sera optimisée en premier lieu

Algorithme 11 Algorithm génitique pour GEP

Entrées: une instance de GEP
1: Prétraitement :
2: Déterminer les paquets de flots $\mathcal{F}^V$, $\mathcal{F}^B$
3: Déterminer l'ensemble des chemins $Paths$
4: Générer la population de base (initiale)
5: **Tant que** (la limite de temps n'est pas expirée) **faire**
6: Classement de Pareto sur la population
7: Calculer les distances de surpeuplement
8: Construire la prochaine population
9: Croisement et mutation
10: Recherche locale
11: **finTq**
Sorties: Ensemble de solutions

et sera suivie par l'optimisation de l'évacuation par bus. Ce choix est réaliste car lors du déclenchement de l'évacuation, les conducteurs en possession de voitures vont partir à un point de rassemblement afin de récupérer les consignes d'évacuation. Tandis que les conducteur de bus doivent partir au dépôt afin de récupérer les directives concernant l'évacuation, pour récupérer ensuite les personnes à partir des points de rassemblement. Aussi, considérer que chronologiquement les voitures vont utiliser le réseau routier avant les bus a du sens.

L'heuristique calcule un plan d'évacuation en décomposant le problème en sous-problèmes. Ces sous-problèmes sont résolu avec des méthodes exactes ou des heuristiques efficaces. Le premier sous-problème consiste à énumérer les ensembles des centres de secours à ouvrir. Pour chacun de ces ensembles, nous résolvons les deux problèmes de routage des voitures et des bus. Nous utilisons l'approche ϵ-contrainte et nous minimisons R_{evac} en garantissant que la date de fin d'évacuation ne dépasse pas T^{exp}_{evac}. Les étapes de cette heuristique sont détaillées ci-dessous.

Dans cette heuristique, nous transformons notre graphe en ajoutant un super-nœud source et un super-nœud destination. Le super-nœud source est relié par des arcs à tous les points de rassemblement. Les informations communiquées sur ces arcs sont le nombre de voitures, le nombre de piétons, arrivant à ces points de rassemblement, la durée de trajet et le risque. Ces deux dernières valeurs sont égales à 0. De même, tous les centres de secours sont reliés par des arcs au super-nœud destination. Les informations associées à chacun de ces arcs sont respectivement la capacité de parking du centre relié au super-nœud destination, sa capacité en termes de nombre de personnes, la durée de traversée et le risque. Ces deux dernières valeurs valent également 0. Afin d'assurer une évacuation d'une durée maximale T^{exp}_{evac}, fixée par le décideur, nous calculons sur chaque nœud une borne inférieure à la date d'arrivée au super-nœud destination.

L'algorithme 12 décrit les différentes étapes de cette heuristique.

Algorithme 12 Heuristique de décomposition

Entrées: Instance du problème

1: Transformation du graphe routier en un graphe avec un super-nœud source et un super-nœud destination
2: calcul des bornes inférieures
3: Soit W l'ensemble des ensembles des centres de secours choisis
4: **Pour** (chaque $w \in W$) **faire**
5: $\varepsilon \leftarrow$ UB pour le critère T_{evac}
6: **Tant que**(il existe une solution réalisable pour ε)**faire**
7: Calcul des chemins pour évacuer les voitures
8: Calcul des chemins pour les bus depuis et vers les centres de secours
9: Soit s la solution calculée
10: $\varepsilon \leftarrow T_{\text{evac}}(s) - 1$
11: Mettre à jour l'ensemble des solutions de front de Pareto avec $(T_{evac}(s), R_{evec}(s))$
12: **finTq**
13: **finPour**

Sorties: Un ensemble de solutions du front de Pareto

5.1 Localisation des centres de secours

Contrairement à l'algorithme génétique, nous supposons que le nombre maximal de centres de secours à ouvrir est fixé par le planificateur. Dans cette étape, nous voulons décider quels sont les centres à ouvrir parmi les centres disponibles. Ce problème de localisation peut se ramener à un problème de sac à dos à trois contraintes. Pour cela, nous avons utilisé un programme dynamique du sac à dos multidimensionnel pour énumérer les ensembles des centres de secours pouvant accueillir tous les évacués (arrivant par bus et par voitures). Le programme linéaire en nombre entiers (3.32-3.35) ci-dessus modélise ce problème de localisation. En pratique, le nombre de centres de secours est faible et l'algorithme pour le sac-à-dos sera très rapide. Pour chaque $j \in \mathcal{S}$, notons par x_j une variable binaire qui est égale à 1 si le centre de secours j sera ouvert et 0 sinon et $\bar{p}_j$ les durées moyennes pour arriver à un centre de secours j. Notons que S_{max} est une constante fixée par le décideur.

$$\max \left[\sum_{j \in \mathcal{S}} (\max_j(\bar{p}_j) - \bar{p}_j + 1) * x_j) \right] \tag{3.32}$$

s.c :

$$\sum_{j \in \mathcal{S}} c_j^S * x_j \geq \sum_{i,t} \left[e_{it}^b + e_{it}^v \right] \tag{3.33}$$

$$\sum_{j \in \mathcal{S}} c_j^P * x_j \geq \sum_{i,t} e_{it}^v \tag{3.34}$$

$$\sum_{j \in \mathcal{S}} x_j \leq S_{max} \tag{3.35}$$

La fonction objectif 3.32 implique de maximiser l'écart entre la distance moyenne maximale et la distance moyenne. Ce qui revient à privilégier les centres de secours les plus proches de la zone sinistrée. Les contraintes 3.33 assurent que les centres ouverts peuvent accueillir toutes les personnes. Les contraintes 3.34 signifient que ces centres ont une capacité de parking suffisante pour le stationnement des voitures. Les dernières contraintes 3.35 sont des contraintes sur le nombre de centres qui seront ouverts.

Pour chaque ensemble de centres de secours, nous effectuons les étapes de routage des voitures, de routage des bus et une recherche locale décrites ci-dessous.

5.2 Routage des voitures

Le routage des voitures se fait en utilisant une adaptation de l'algorithme de *min cost flow* proposée par ([Ndiaye et al., 2014a]) et qui permet de résoudre un problème d'évacuation bicritère des piétons sur un graphe indexé sur le temps. La complexité de cet algorithme est $\mathcal{O}(nbEvac \times (|\mathcal{A}'| \log T + (|\mathcal{N}| \log |\mathcal{N}|))$, où $nbEvac$ est le nombre de personnes à évacuer. Cet algorithme a été adapté afin que la contrainte T^{exp}_{evac} imposée sur la date de fin d'évacuation soit respectée lors du calcul des chemins pour l'évacuation des voitures. En d'autres termes, le critère à minimiser durant l'évacuation des voitures est la somme des risques en excluant tous les chemins dont la durée dépasse la valeur T^{exp}_{evac}. Pour ce faire, nous utilisons les bornes inférieures calculées sur chaque nœud du graphe.

5.3 Routage des bus

Après avoir effectué l'évacuation par voitures, nous voulons évacuer les personnes par bus. Les bus se déplacent initialement depuis le dépôt vers les points de rassemblement où les personnes sont en attente. Leurs déplacements s'effectuent soit selon le plus court chemin entre le dépôt et les points de rassemblement. En utilisant le graphe résultant de l'étape précédente, les choix des chemins des bus depuis les points de rassemblement vers les centres de secours sont déterminés par l'adaptation de *min-cost flow* ([Ndiaye et al., 2014a]) : minimisation de la somme des risques tout en respectant la contrainte imposée sur le critère T_{evac}. Afin d'assurer cette contrainte, nous utilisons les bornes inférieures calculées à chaque nœud du graphe. Donc, pour chaque bus arrivant sur un nœud j, nous ajoutons à sa date d'arrivée au nœud j, la borne inférieure calculée sur j, si cette valeur calculée est inférieure à T^{exp}_{evac} alors le nœud j ferait partie des nœuds constituant le chemin de bus. Dans le cas contraire, j ne sera pas choisi. Une fois le bus arrivé à un centre de secours deux cas se présentent, soit il n'y a plus de personnes à évacuer ou bien soit le bus doit faire un retour vers un point de rassemblement afin de récupérer des personnes. Dans le premier cas, un plan d'évacuation est calculé. Dans le deuxième cas, nous appliquons la méthode du plus court chemin en prenant compte que de la distance. Puisque le bus sera vide, l'impact sur le critère somme des risques est minime. Le couplage flot/plus court chemin est itéré jusqu'à ce que tous les évacués se trouvent à un centre de secours.

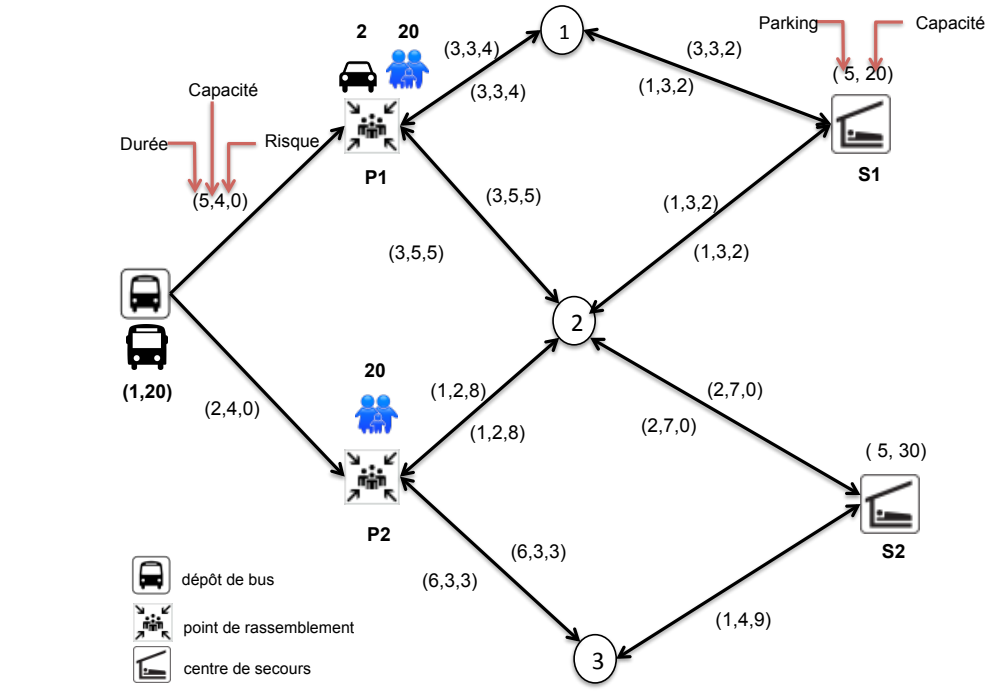
Afin d'éviter la congestion sur les routes et pour ne pas ralentir l'évacuation, nous supposons que l'attente sur les nœuds du réseau routier est impossible. Il est plus judicieux, à priori, que les personnes restent en attente sur les points de rassemblement.

L'exemple simplifié suivant illustré les étapes de cette heuristique

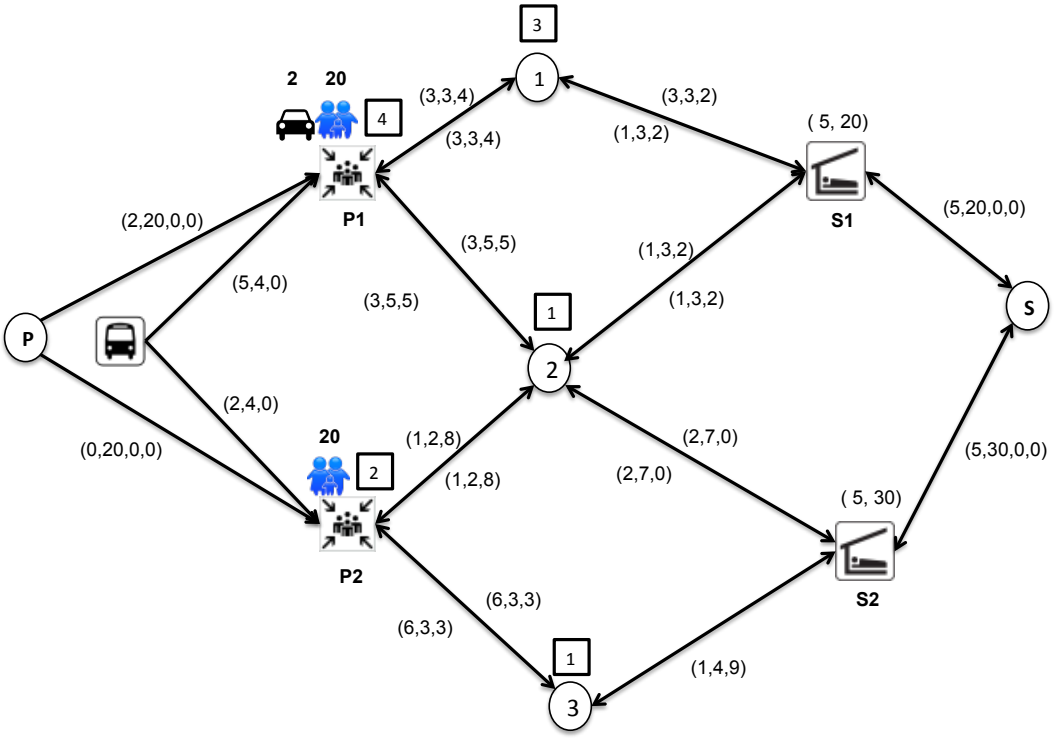
Exemple 6 *Soit le graphe initial représenté dans la figure(3.4a), nous disposons de deux centres de secours et de deux points de rassemblement. Un seul bus est disponible pour évacuer les piétons. À l'instant* $t = 0$*, deux voitures et 20 piétons sont apparus sur le premier point de rassemblement et 20 personnes sont arrivées au deuxième point de rassemblement. Les valeurs qui sont sur les arcs sont respectivement la durée de traversée de cet arc, la capacité de l'arc en termes de nombre de voitures et le risque de passage par cet arc. Les valeurs qui sont sur les centres de secours sont respectivement la capacité du parking du centre et la capacité du centre. Le décideur souhaite évacuer les personnes avant la date* $t = 15$*. La transformation du graphe en un graphe avec une seule source et une seule destination ainsi que les bornes inférieures de la durée de parcours depuis chaque nœud vers le nœud destination sont représentées dans la figure (3.4b).*

L'évacuation se fait en premier lieu par voitures. Pour cela, nous calculons le flot max à coût min, ce chemin ne doit pas dépasser la date 15. Le chemin qui minimise la somme des risques et qui vérifie la condition sur la date de fin est le chemin "$P, P_1, 2, S_2, S$*". Ensuite, nous mettons à jour les capacités résiduelles des arcs empruntés par les deux voitures (3.5b). La date de fin d'évacuation est de* $T_{evac} = 5$ *et la somme des risques est égale à* $R_{evac} = 10$*.*

Une fois que l'évacuation par voitures est achevée, pour évacuer les personnes par bus, nous procédons d'abord à la conversion des capacités des arcs en nombre de bus ; à l'exception des capacités des arcs sortants du super-nœud source, des arcs sortants et entrants depuis et vers le super-nœud destination. Dans cet exemple, nous supposons qu'un bus occupe deux fois la place d'une voiture. Le bus est routé vers le point de rassemblement le plus proche qui est P_2*. Le chemin le plus sûr depuis* P_2 *et qui respecte la date de fin fixée est le chemin "*$P, P_2, 2, S_2, S$*". Donc l'évacuation par bus des 20 personnes par ce chemin débute à la date* $t = 2$*. Nous mettons à jour les capacités résiduelles des arcs de ce chemin. La date de fin d'évacuation devient* $T_{evac} = 5$ *et le risque* $R_{evac} = 10 + (20 * 8) = 170$*. Puisqu'il reste des personnes en attente sur* P_1*, le bus doit faire un retour vers ce point. À* $t = 5$*, nous choisissons le plus court chemin depuis le centre de secours* S_2 *vers* P_1*. Le chemin en vert de la figure (3.5c) est le plus court chemin. Pareillement, nous mettons à jour les capacités résiduelles des arcs utilisées. Nous obtenons* $T_{evac} = 10$ *et* $R_{evac} = 175$*. Une fois que le bus est arrivé à* P_1*, l'évacuation des personnes commence à la date* $t = 10$*. Puisque la capacité d'accueil du* S_2 *est insuffisante, les chemins possibles pour atteindre* S_1 *sont "*$P, P_1, 1, S_1, S$*" et "*$P, P_1, 2, S_1, S$*". Le chemin "*$P, P_1, 1, S_1, S$*" est moins risqué que "*$P, P_1, 2, S_1, S$*". Par contre, l'évacuation par "*$P, P_1, 1, S_1, S$*" dépasse la date de fin fixée. Donc, le chemin choisi est "*$P, P_1, 2, S_1, S$*". Enfin, nous obtenons* $T_{evac} = 14$ *et* $R_{evac} = 175 + 140 = 315$*.*

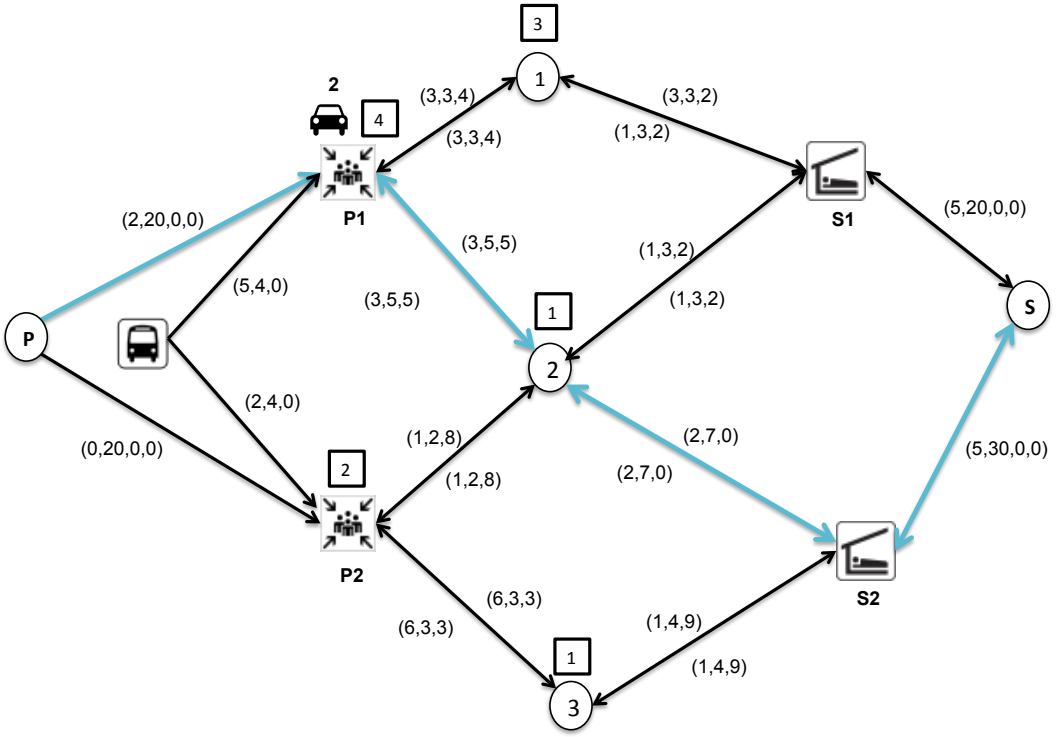


(a) Graphe initial

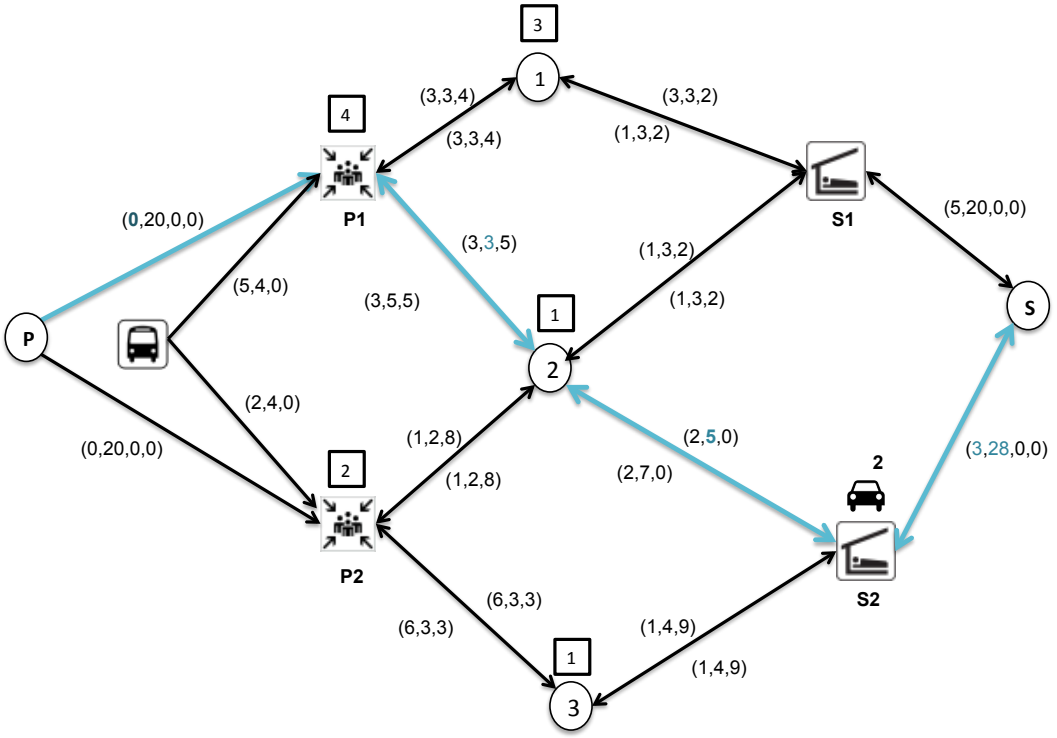


(b) Transformation du graphe initial et calcul des bornes inférieures

FIGURE 3.4 – Exemple d'instance du GEP

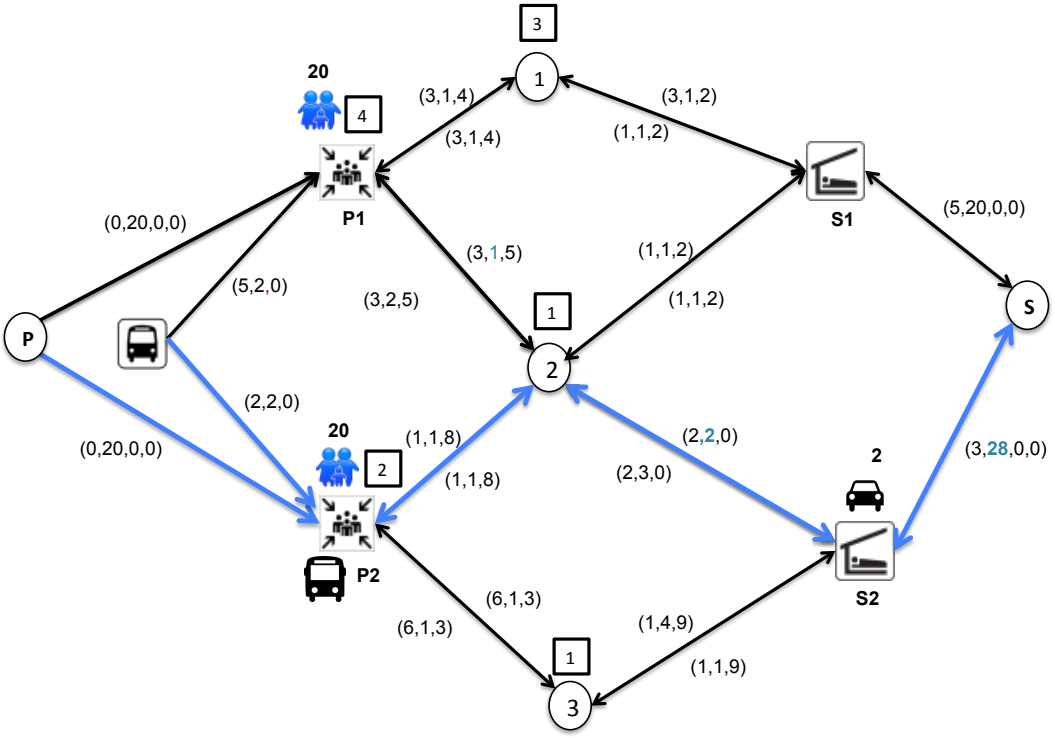


(a) Choix du chemin le plus sûr

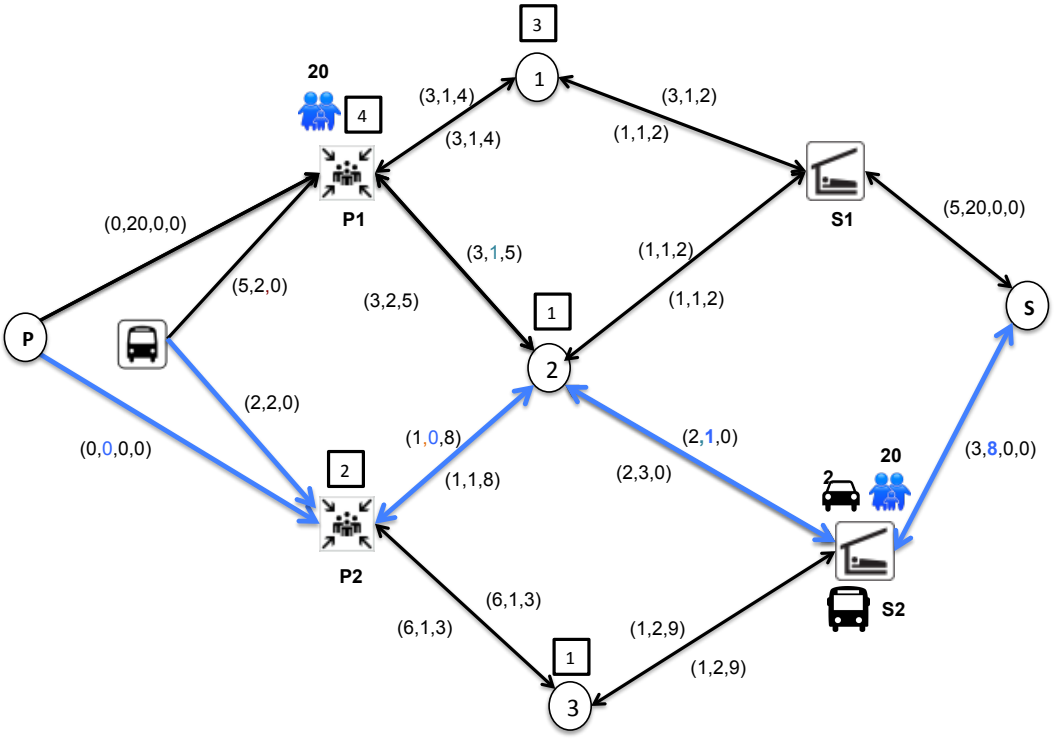


(b) Èvacuation des voitures

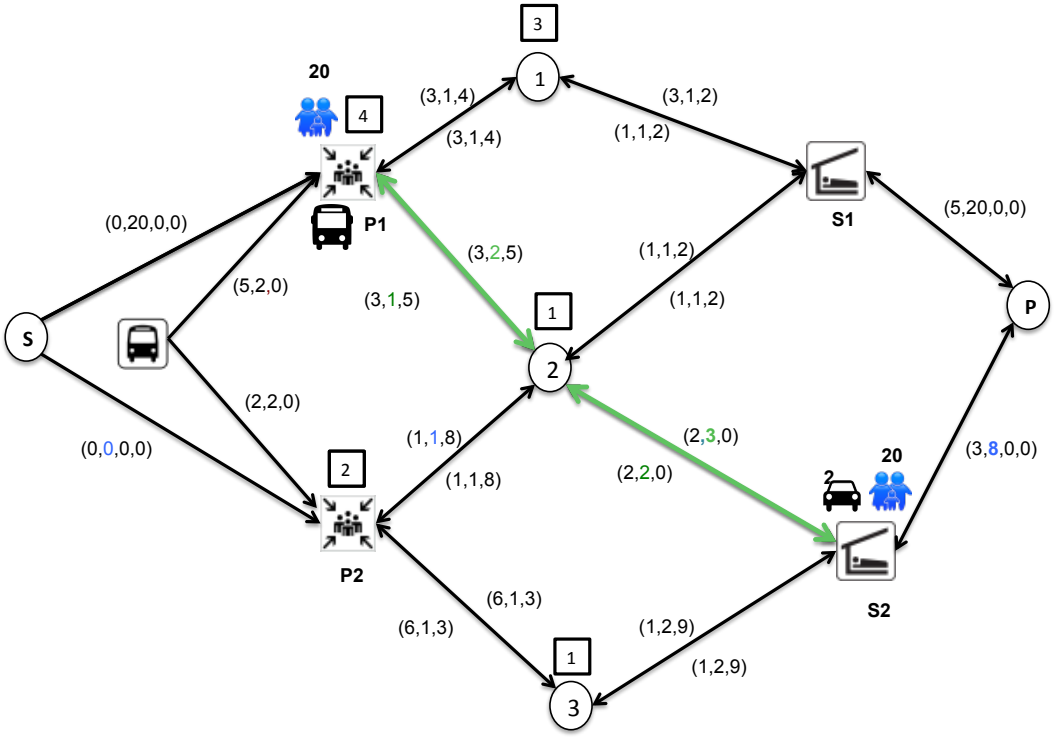
FIGURE 3.5 – Les étapes de l'évacuation par voiture



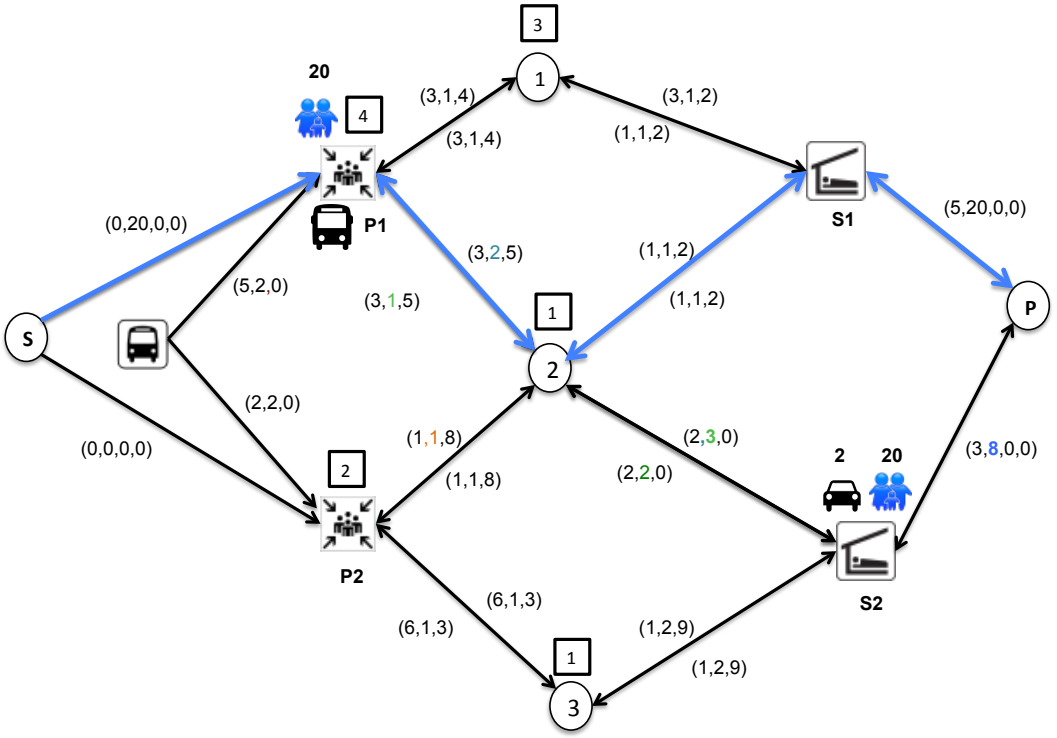
(a) Déplacement de bus depuis le dépôt



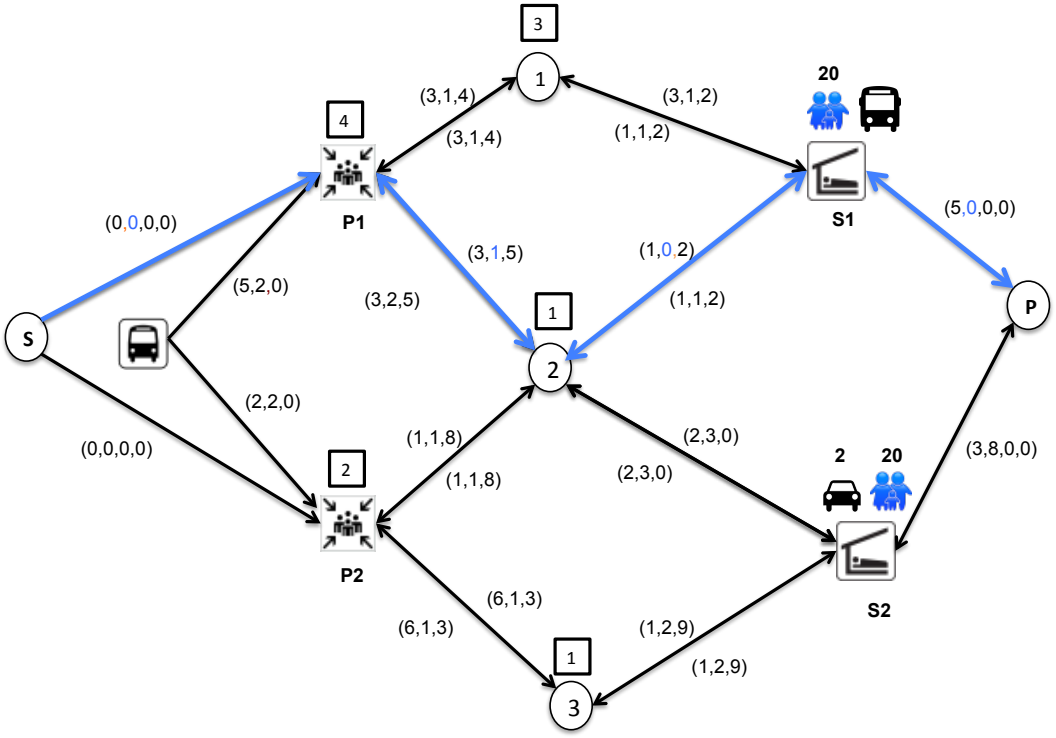
(b) Èvacuation des personnes depuis P_2 vers S_2



(c) Retour de bus vers P_1



(d) Choix du chemin le plus sûr



(e) Èvacuation des personnes depuis P_1 vers S_1

FIGURE 3.4 – Les étapes de l'évacuation par bus

6 Réduction du graphe

Il peut être pertinent de réduire la taille du graphe avant de résoudre une instance du problème GEP en utilisant l'algorithme génétique ou l'heuristique par décomposition. Dans cette section, nous présentons une méthode qui d'agrégation du graphe développé par Bob Grun, doctorant à l'université de Kaiserslautern et membre du projet DSS_Evac_Logistic.

6.1 Agrégation des données

Afin de résoudre des instances de taille réelle en un temps raisonnable nous procédons à la réduction du graphe routier. La solution calculée par l'algorithme génétique sur le graphe réduit est étendu pour avoir une solution sur le graphe initial.

Tout d'abord, nous décrivons comment une instance du GEP peut être agrégée afin de réduire la taille du réseau routier.

Soit un graphe. Sur chaque arc de ce graphe, nous avons la durée de traversée et la valeur du risque de passage. Nous créons des super-nœuds constituant un ensemble de nœuds et un ensemble d'arcs. La figure 3.5 montre un exemple de cette réduction. Les nœuds i_2, i_3 et i_4 sont remplacés par un seul super-nœud, ce dernier contenant aussi les arcs e_2, e_3, et e_4.

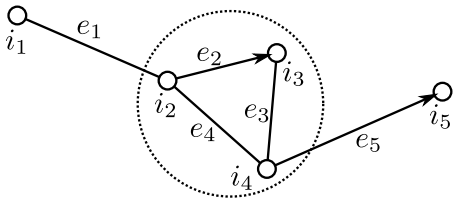


FIGURE 3.5 – Exemple d'une réduction de graphe.

Notons que dans cet exemple, le super-nœud constitue un sous graphe connexe. Cette condition est nécessaire pour pouvoir calculer par la suite les chemins dans ces nœuds afin d'obtenir une solution globale.

L'agrégation du graphe est obtenue en affectant une valeur "d'importance" v_e à chaque arc e, qui représente une estimation de l'impact qu'un arc peut avoir lors du calcul de la solution. Nous calculons ces valeurs en mettant :

$$v_e = \frac{r_e - r_e^{\min}}{r_e^{\max} - r_e^{\min}} + \frac{p_e - p_e^{\min}}{p_e^{\max} - p_e^{\min}},$$

où $r_e^{\max} = \max_{e \in \mathcal{A}'} r_e$, $r_e^{\min} = \min_{e \in \mathcal{A}'} r_e$, et respectivement de manière analogue pour $p^{\max}$ et $p^{\min}$. Donc plus la valeur de v_e est grande, plus il est prévu que l'arc ait une influence sur la qualité d'une solution. nous trions l'ensemble des arcs dans l'ordre croissant des valeurs v_e. Selon cet ordre, nous éliminons de cette ensemble un sous-ensemble jusqu'à ce que le nombre désiré de nœuds (ou d'arcs) soit atteint. Par définition, un arc est éliminé en intégrant son nœud destination à un super-nœud. L'élimination d'un arc est possible exceptée pour les deux cas suivants : (1) si le nœud destination de l'arc et l'un des nœuds du super-nœud sont dans $\mathcal{N}^V \cup \mathcal{N}^{\mathcal{B}} \cup \mathcal{S}$ ou (2) si le super-nœud résultant n'est pas fortement connecté, c'est-à-dire : un arc représente un seul chemin de la route. Dans ce dernier cas, cet arc peut être ajouté à un super-nœud lors d'une itération postérieure.

Dans le cas où un point de rassemblement fait partie d'un super-nœud source, les personnes à évacuer sont transférées à ce super-nœud. De même, un super-nœud destination sera utilisé comme un centre de secours s'il contient un centre de secours.

Extension de la solution

Partant d'une solution construite sur un réseau agrégé, nous avons besoin d'étendre cette solution à une solution de l'instance originale. Par conséquent, nous devons étendre les chemins depuis le graphe agrégé vers le graphe original. Dans la figure 3.6, nous expliquons comment étendre une solution agrégée. Nous considérons le chemin "e_1, e_2,e_3" menant depuis le super-nœud sn_1 vers le super-nœud sn_4. Ce chemin doit être étendu à un chemin de s vers t.

Un chemin doit être déterminé depuis s vers le nœud source de l'arc e_1 qui est intérieur à sn_1, puis, à partir de la destination de e_1 vers le nœud source de l'arc e_2 qui intérieur à sn_2. À chaque super-nœud, le chemin le plus court sera utilisé. Enfin, dans le super-nœud final, un autre chemin élémentaire doit être déterminé à partir de la destination de l'arc

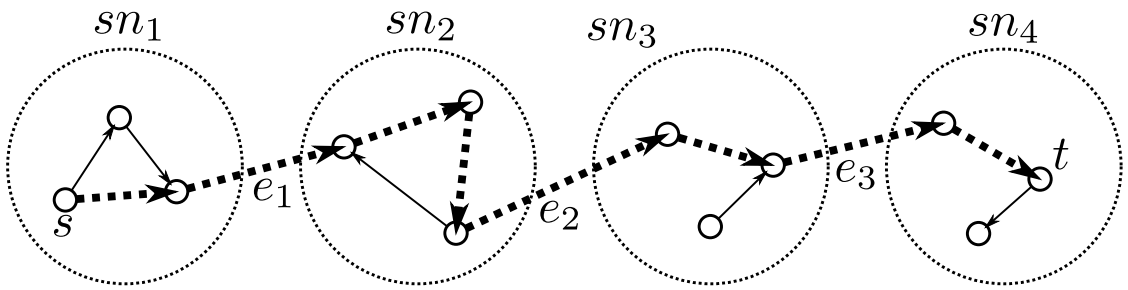


FIGURE 3.6 – Extension d'une solution agrégée.

e_3 vers t. De cette façon, chaque chemin utilisé dans l'instance agrégée est étendu à un trajet dans l'instance originale.

Exemple 7 *Soit le graphe* G *présenté dans la figure (3.7). Ce graphe est constitué de 10 nœuds dont un point de rassemblement P et un centre de secours S et 17 arcs notés* $e_1, e_2, ..., e_{17}$*. Les deux valeurs associées à chaque arc sont respectivement la durée du trajet et le risque. Nous souhaitons réduire la taille du graphe pour obtenir d'un nombre de nœuds égal à 5.*

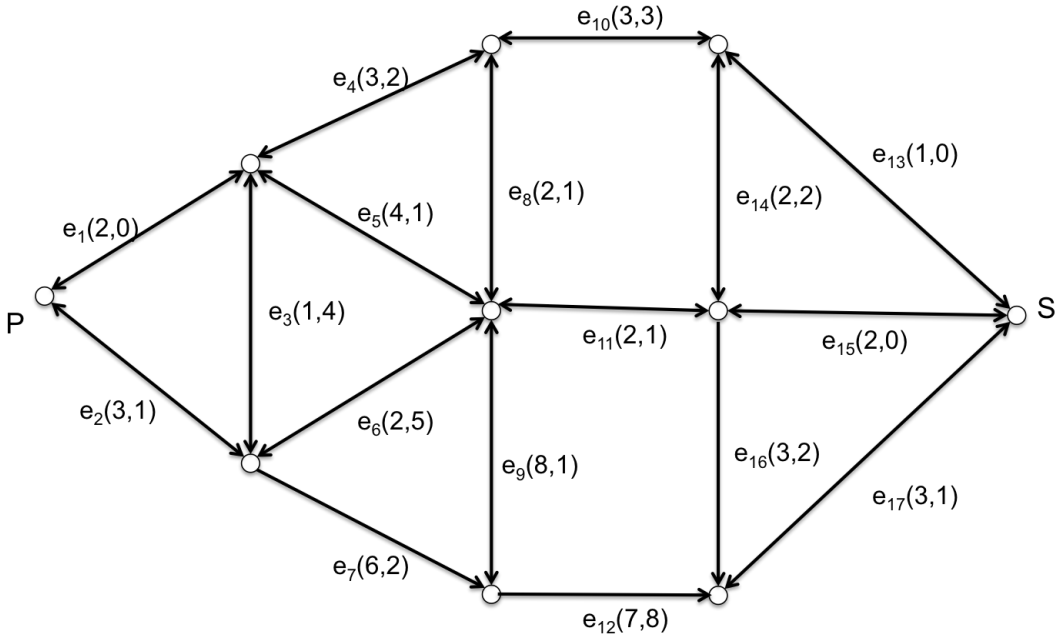


FIGURE 3.7 – Le graphe G

Pour ce faire, nous calculons, d'abord, la valeur v_i de chaque arc e_i, tel que $i \in [1, 17]$, en utilisant les valeurs de durée et du risque. Ensuite, les valeurs v_i pour tout $i \in [1, 17]$ sont triées selon l'ordre croissant. Ces valeurs son représentées dans le tableau 3.1.

v_{15}	v_1	v_{16}	v_8	v_{13}	v_{14}	v_2	v_3	v_{17}	v_4	v_5	v_{11}	v_{10}	v_6	v_7	v_9	v_{12}
0.1	0.3	0.3	0.4	0.4	0.5	0.6	0.6	0.6	0.7	0.7	0.7	0.8	0.9	1.1	1.3	2.0

TABLE 3.1 – Les valeurs v_i calculées pour chaque arc e_i

Selon l'ordre croissant des v_i, nous construisons des super-nœuds en ajoutant progressivement les arcs à ces super-nœuds. Les deux premiers super-nœuds construits sont notés par Sn_1 et Sn_2 et contiennent respectivement l'arc e_{15} et l'arc e_1 (figure 3.7). Dans la prochaine itération, l'arc e_{16} ne peut pas être ajouté au super-nœud Sn_1 car sinon Sn_1 sera non connexe. Cette connexité est indispensable pour assurer l'existence d'une chemin de P vers S lors de la phase d'extension de la solution agrégée.

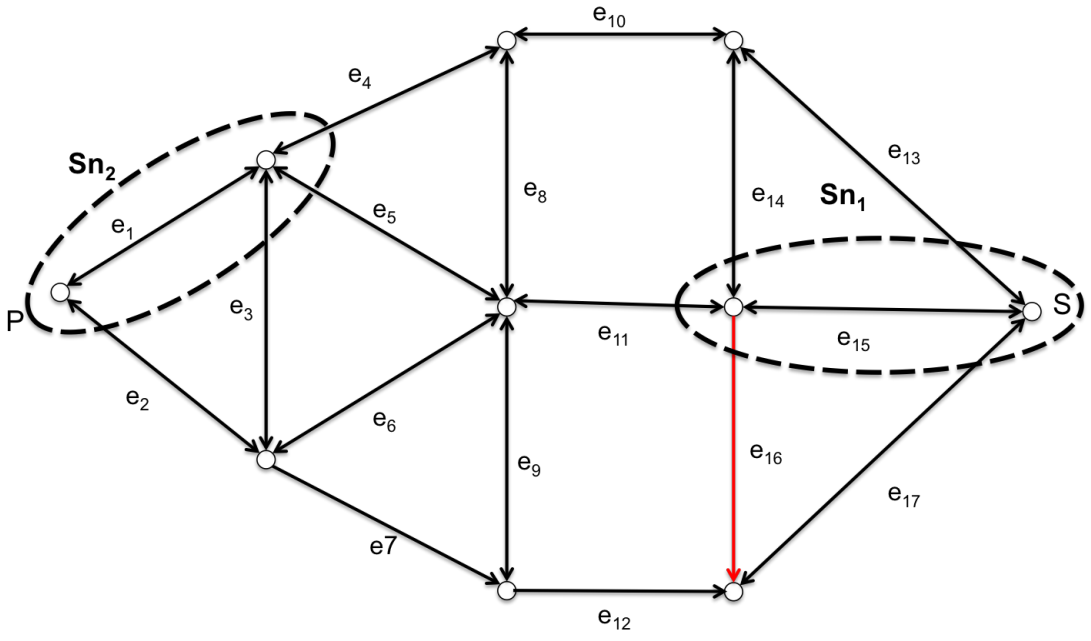


FIGURE 3.8 – Construction des super-nœuds

De la même manière, nous ajoutons les arcs aux super-nœuds jusqu'à l'obtention d'un graphe dont le nombre de nœuds est égale à 5. La solution du graphe agrégé est présentée dans la figure 3.9.

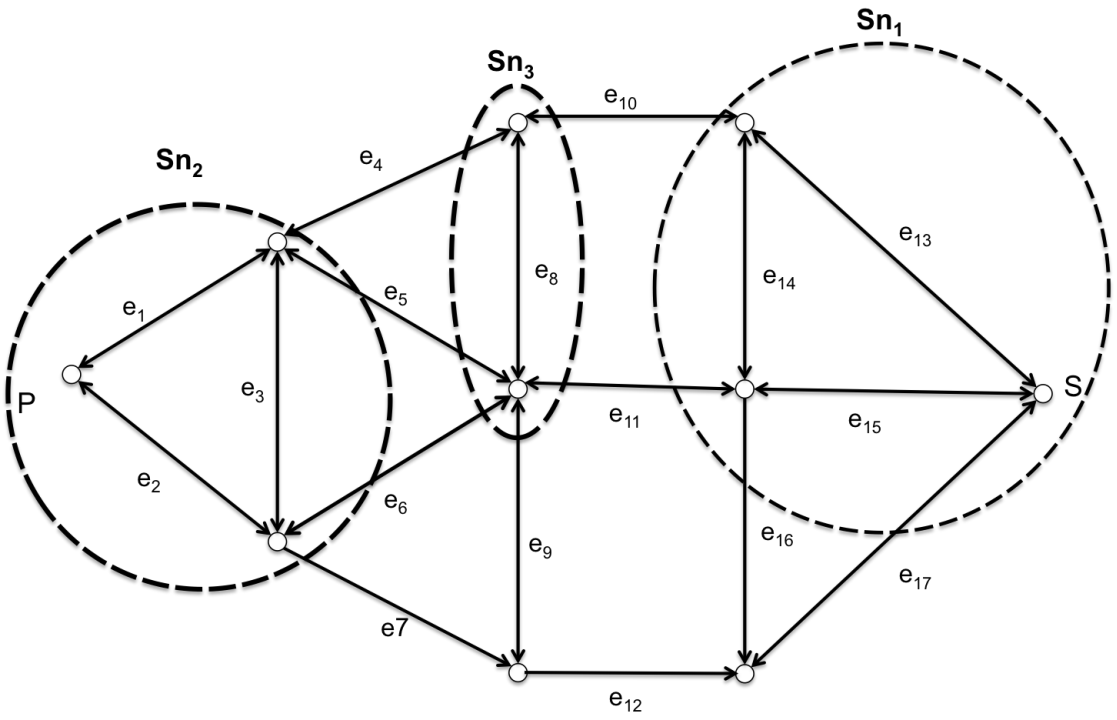


FIGURE 3.9 – Le graphe G agrégé

7 Expérimentations

Dans cette section, nous nous focalisons sur les tests et la comparaison des méthodes développées dans ce chapitre. Toutes les méthodes sont codées en C++ et exécutées sur un PC, 16-cœurs Intel Xeon E5-2670 avec une fréquence 2.60 GHz, 8 GB RAM, et sous windows 7. Le solveur mathématique utilisée est ILOG CPLEX 12.6 avec un nombre de cœurs égal à 4.

La formulation mathématique a été testée sur des instances générées aléatoirement. Cette formulation a été résolue par le solveur CPLEX auquel une limite de temps de 3600 minutes est imposée afin de calculer une seule solution de front de Pareto. Les résultats de ces tests sont présentés dans la section 7.1 suivante.

L'algorithme génétique et l'heuristique par décomposition sont testés sur des instances réelles de la ville de Nice (France) et la ville de Kaiserslautern (Allemagne). Les modèles mathématiques intégrés dans l'algorithme génétique sont résolus en utilisant le solveur SCIP 3.1.0 avec SoPlex 1.7.2 [Achterberg, 2009]. La taille de la population d'individus est de 125. Pour résoudre les instances de la ville de Nice, nous avons imposé à l'algorithme génétique et à l'heuristique par décomposition une limite de temps égale à 3600 minutes. Quant à l'instance de Kaiserslautern, cette limite est réduite à une minute. Les résultats de ces tests sont présentés dans la section 7.2.

7.1 Instances Aléatoires

Les instances sont générées aléatoirement de sorte que nous ayant toujours des instances réalisables. Le nombre de nœuds du graphe est généré aléatoirement dans $[10, 60]$. Les données d'un arc (i, j) sont générées de la façon suivante : $p_{ij} \in [1, 10]$, (i, j), $r_{ij} \in [1, 10]$, $c^a_{ij} \in [10, 15]$ et le ratio $\alpha = 0.5$. Le nombre de centres de secours S prend les valeurs suivantes : 2,3, 4,5 et 6. La capacité c^S_j d'un centre de secours j est générée aléatoirement dans $[80, 200]$ et sa capacité de parking $c^P_j \in [10, 40]$. Le nombre de points de rassemblement $P \in [1, 6]$. Le nombre total de personnes à évacuer par bus prend les valeurs suivantes $\{40, 60, 80, 100, 120\}$. Le nombre total de personnes à évacuer est généré dans $[10, 70]$. Le nombre de bus $B \in [1, 3]$ et la capacité de chaque bus est égale à 20.

Notons que nous avons généré aléatoirement 20 instances à partir du nombre de nœuds, du nombre de personnes à évacuer par bus, du nombre de personnes à évacuées par voitures et du nombre de bus disponibles.

Le Tableau 3.2 présente les temps CPU nécessaires pour le calcul des solutions optimales et les déviation(%) des solutions par rapport à la meilleure borne inférieure calculées par CPLEX. Ce Tableau présente aussi le nombre de nœuds du graphe (#noeuds), le nombre total de personnes à évacuer par bus (#Bevac), le nombre total de personnes utilisant leurs voitures (#Vevac) et le nombre de bus (#bus).

#noeuds	#Bevac	#Vevac	#bus	#inst_opt	temps(s)			deviation(%)		
					min	avg	max	min	moy	max
12	60	15	1	2	48,126	120,45	226,71	0	0	0
20	80	25	3	18	446,02	1379,28	3577,71	0	0	0
20	100	40	3	11	561,82	2301,73	3555,09	0	0	0
25	40	10	2	20	299,89	709,17	1118,46	0	0	0
50	100	70	3	0	3600	3600	3600	100	100	100
55	120	53	3	0	3600	3600	3600	100	100	100

TABLE 3.2 – Les temps d'exécution et les déviations

Les résultats de ce Tableau montrent que pour les instances où $\#noeuds = 25$, $\#Bevac < 80$ et $\#Vevac < 25$, CPLEX est capable de calculer des solutions optimales. Notamment, dans le cas où #noeuds =12, CPLEX calcule des solutions optimales en 120,45s en moyenne. Pour les autres instances où $\#noeuds \geq 50$, CPLEX n'arrivent pas à calculer des solutions réalisables. Nous concluons que ce modèle est incapable de résoudre des instances de grande taille.

7.2 Instances réelles

Instance de la ville de Kaiserslautern

L'instance de la ville de Kaiserslautern a été fournie par les autorités responsables de la sécurité de la ville de Kaiserslautern. Le graphe de Kaiserslautern est constitué de 13284 nœuds et 32463 arcs. Le scénario de catastrophe retenu est une découverte d'une bombe dans le centre de la ville. Cela nécessite une évacuation d'une population sur un

périmètre de 500 mètres. Nous avons 6 points de rassemblement et 9 centres de secours. Nous disposons de 6 bus dont chacun a une capacité de 80 personnes. Le nombre de personnes à évacuer par bus est égale à 1250. Les spécificités de l'évacuation de la ville de Kaiserslautern est que les voitures se dirigent vers des sorties et non pas vers des centres de secours. Cette évacuation par voitures est gérée en amont par les algorithmes développés dans la tâche WP5. De plus, les valeurs de risque sur les arcs sont nulles car le réseau routier n'est pas endommagé.

Instances de la ville de Nice

Les scénarios de catastrophe dans la cas de la ville de Nice est un séisme Vésubie modéré de magnitude 6.4 et un sièsme Ligure de magnitude 6.9. Le graphe de la ville de Nice est constitué de 2323 nœuds et 5184 arcs. Vu que la ville de Nice est une ville touristique, nous considérons deux types de population, la population d'hiver (PopINSEE) et la population d'été (PopMAx). PopMax est constituée des habitants de Nice et des touristes. Il résulte quatre scénarios de données présentés dans le tableau 3.4. Le tableau 3.4 présente le nombre de nœuds (#noeuds), le nombre d'arcs (#arcs), le nombre total de personnes à évacuer par bus (#Bevac), le nombre total de personnes à évacuer par voitures (#Vevac), le nombre de bus (#bus), le nombre de centres de secours (#centre) et le nombre de points de rassemblement (#P_rass). Notons que chaque bus a une capacité égale à 60.

Ces données ont été fournies par Le BRGM, et calculées par des méthodes de simulation dans le but de déterminer les dommages causés par chaque séisme ([Lemoine et al., 2014]). Aussi, nous avons utilisé les sorties des algorithmes de la tâche WP5 qui sont les répartitions des piétons et les voitures sur les points de rassemblement.

Instance	#noeuds	#arcs	#Bevac	#Vevac	#bus	#centre	#P_rass
Vésubie PopINSEE	2323	5184	6025	1989	50	51	65
Vésubie PopMax	2323	5184	9019	2835	50	51	65
Ligure PopINSEE	2323	5184	25569	2391	60	65	65
Ligure PopMax	2323	5184	32191	2464	60	65	65

TABLE 3.3 – Instances de la ville de Nice

Les figures 3.10, 3.11, 3.12, 3.13 et 3.14 représentent respectivement les solutions des instances de Kaiserslautern, Vésubie PopINSEE, Vésubie Popmax, Ligure PopINSEE et Ligure Popmax, calculées par l'algorithme génétique (GA), et celles calculées par l'heuristique par décomposition (DH).

La figure 3.10 montre que les solutions calculées par les deux méthodes GA et DH sont faiblement non dominées. Ceci était prévisible car toutes les valeurs de risque sont nulles. Nous constatons aussi que les solutions calculées par l'algorithme génétique dominent celles calculées par l'heuristique par décomposition. Cette dominance est une conséquence de la prise en compte du critère de la durée lors du calcul des chemins par l'algorithme génétique. En outre, les deux méthodes sont capables de calculer plusieurs solutions en une minute.

Les figures 3.11 et 3.12 montrent que l'heuristique par décomposition calcule des so-

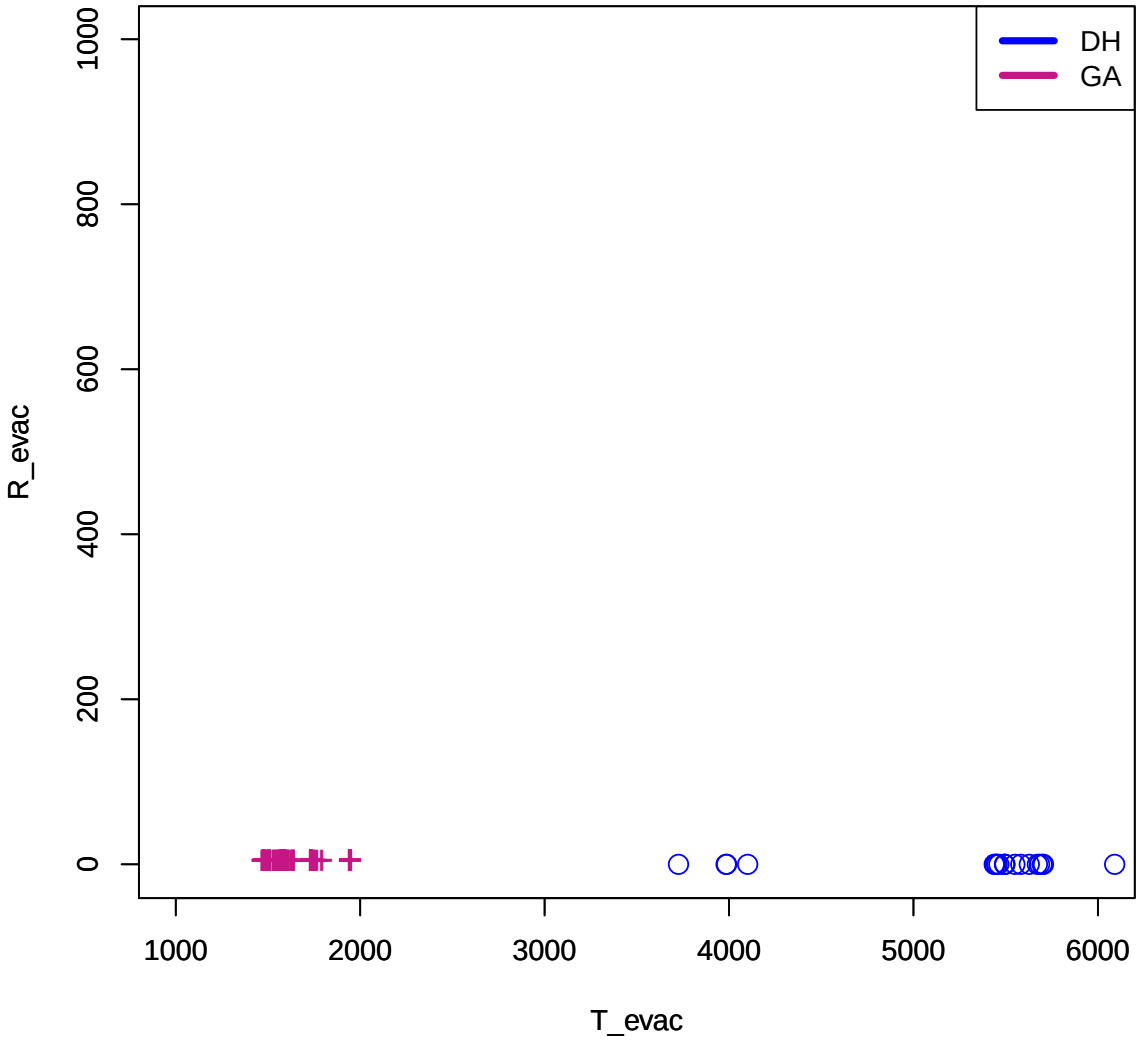


FIGURE 3.10 – Solutions de l'instance de Kaiserslautern

lutions faiblement non dominées. Ces solutions dominent une partie du front de Pareto calculé par l'algorithme génétique. Aussi, le nombre de solutions calculées par l'algorithme génétique est supérieur au nombre de solutions calculées par l'heuristique par décomposition.

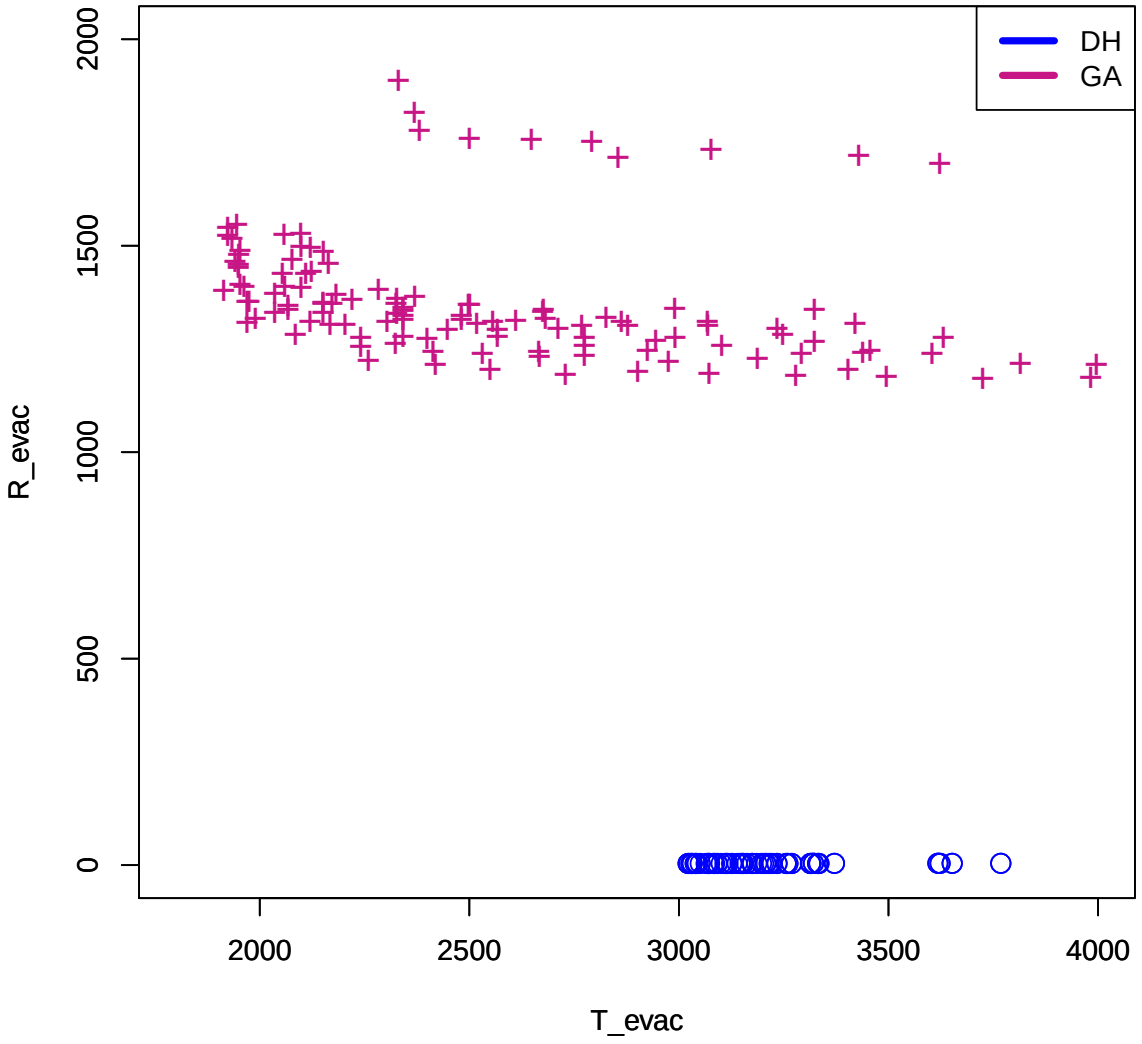


FIGURE 3.11 – Solutions de l'instance Vésubie PopINSEE

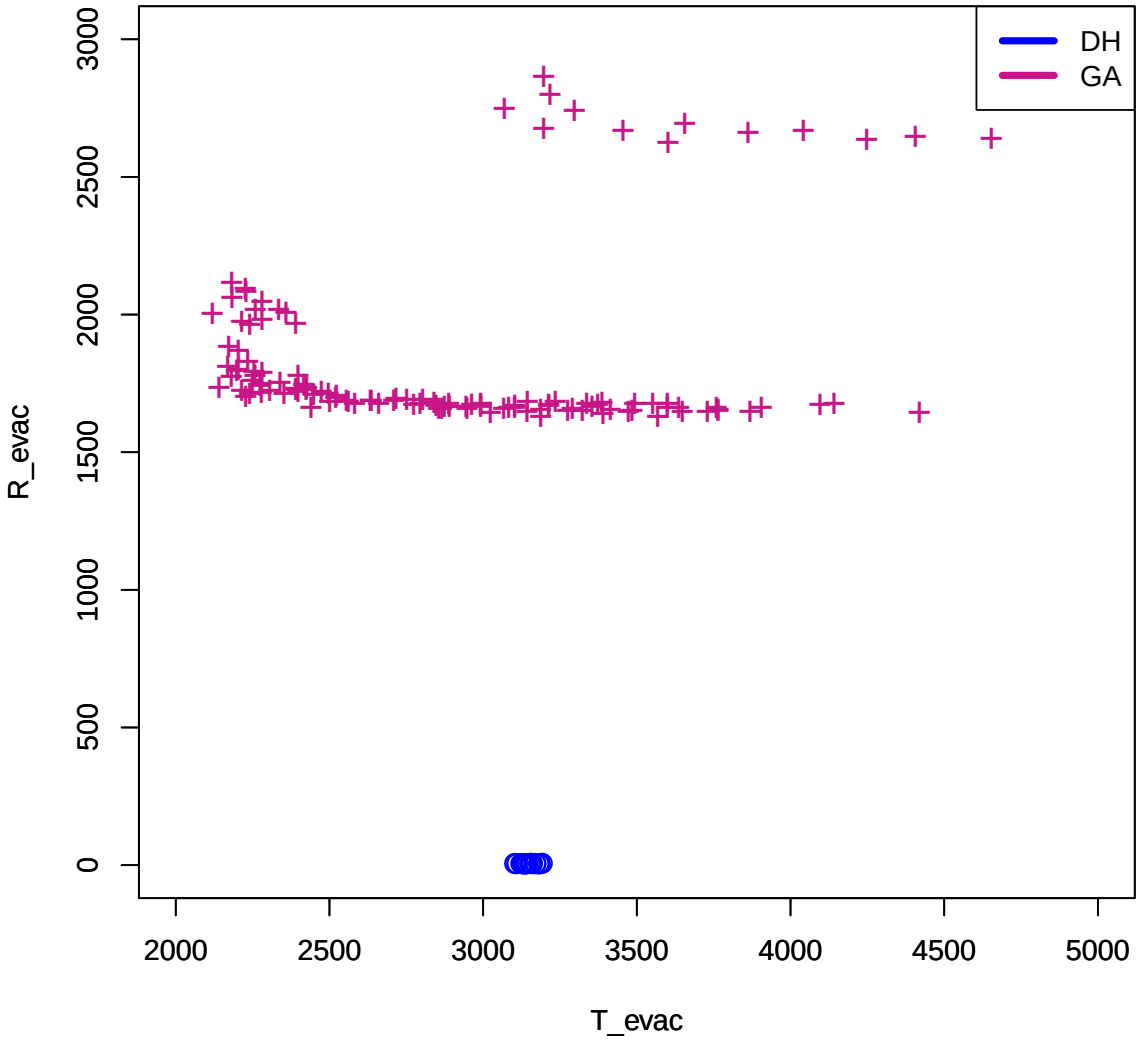


FIGURE 3.12 – Solutions de l'instance Vésubie PopMax

Les figures 3.13 et 3.14 montrent que, contrairement à l'algorithme génétique, l'heuristique par décomposition calcule peu de solutions pour les instances Ligure PopINSEE et Ligure PopMax. Cela est justifié par l'augmentation significative du nombre de personnes à évacuer par bus est par voiture. Pour l'instance Ligure PopINSEE, l'algorithme génétique et l'heuristique par décomposition sont complémentaires. Chacun d'eux calcule une partie de front de Pareto. Dans le cas de l'instance Ligure PopMax, l'heuristique par décomposition est capable de calculer des solutions dominant les solutions calculées par l'algorithme génétique.

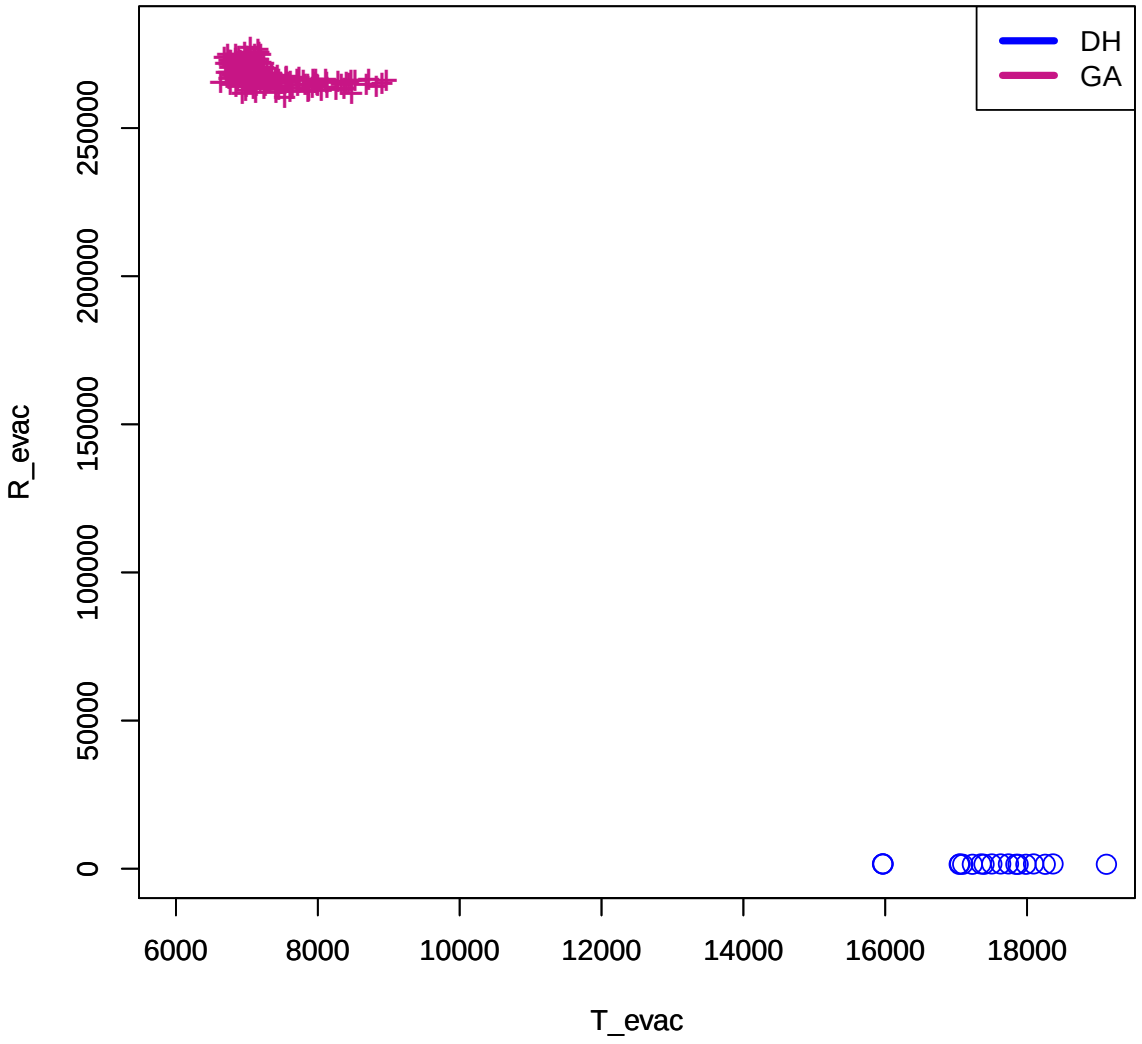


FIGURE 3.13 – Solutions de l'instance Ligure PopINSEE

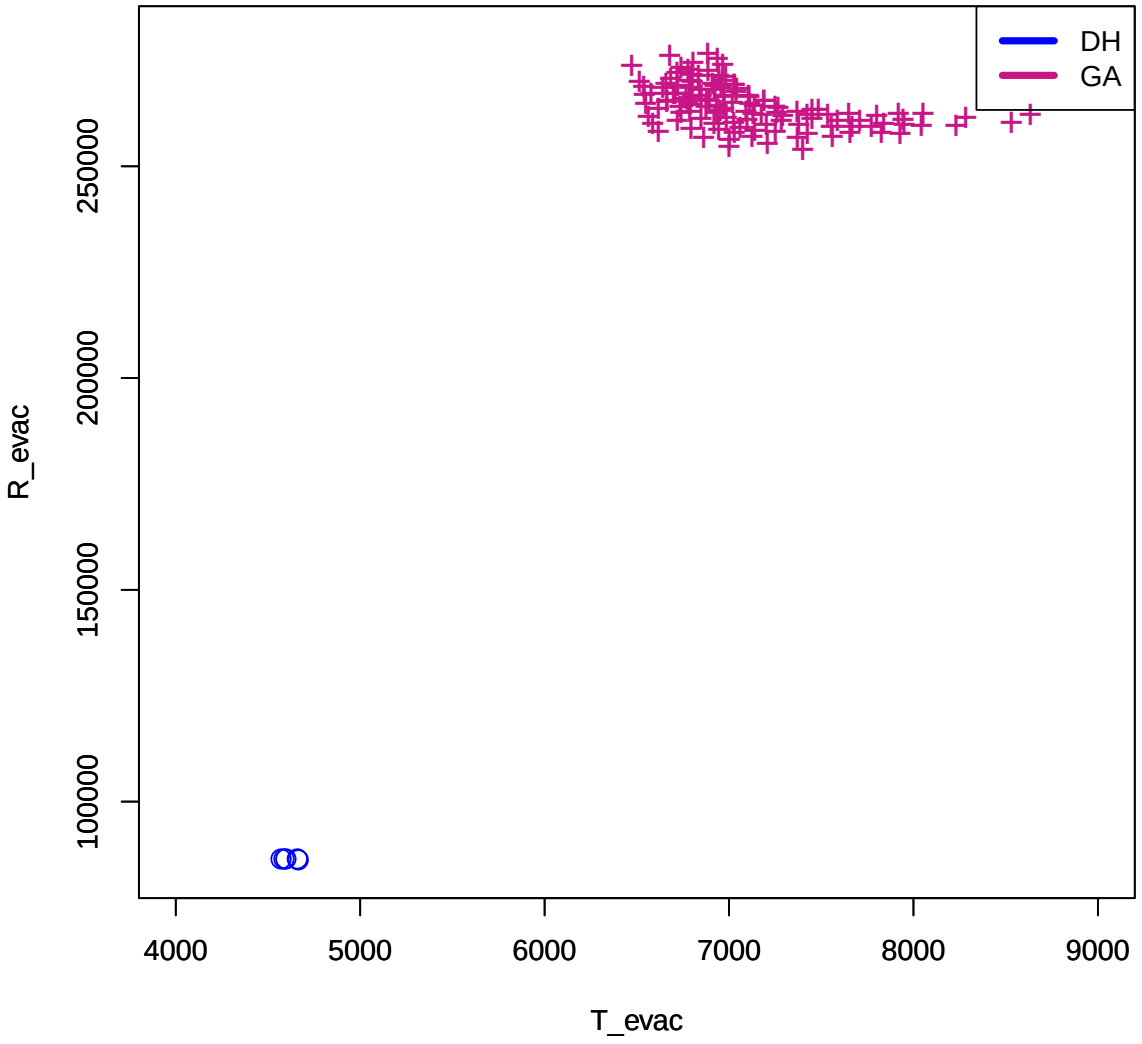


FIGURE 3.14 – Solutions de l'instance Ligure PopMax

Le Tableau 3.4 présente le nombre total de solutions constituant les front de Pareto calculés respectivement par l'algorithme génétique (GA(#Sol)) et l'heuristique par décomposition (DH(#Sol)), et le nombre de solutions nondominées (GA(#Sol_ND)) et (DH(#Sol_ND)) de ces fronts.

À travers les résultats présentés dans cette section, nous concluons que l'algorithme génétique est plus rapide que l'heuristique par décomposition. En revanche, l'heuristique par décomposition calcule un ensemble de solutions complémentaire du front de Pareto obtenu par l'algorithme génétique. Donc, nous pouvons constater que ces deux méthodes sont complémentaires. Notons qu'une amélioration de l'heuristique par décomposition est

Instance	GA		DH	
	#Sol	#Sol_ND	#Sol	#Sol_ND
Kaiserslautern	125	1	40	1
Vésubie PopINSEE	125	15	75	1
Vésubie PopMax	125	14	34	5
Ligure PopINSEE	125	5	20	4
Ligure PopMax	125	7	5	1

TABLE 3.4 – Solutions calculées par l'algorithme génétique et l'heuristique par décomposition

possible en échantillonnant le front de Pareto, c'est-à dire nous choisissons, à priori, des valeurs pour $T_{\text{evac}}^{\text{exp}}$. Cela permet à la fois d'étaler les solutions sur la frontière du front et de réduire le temps de calcul.

8 Conclusion du chapitre

Contrairement au chapitre précédent, ici nous avons considéré le problème de calcul des plans d'évacuation pour les bus et les voitures sur un réseau routier réel. L'objectif est de choisir les centres de secours à ouvrir afin de minimiser la date de fin d'évacuation, ainsi que le risque encouru durant l'évacuation. Pour la résolution, nous avons proposé en premier lieu une modélisation en nombres entiers. Pour cette modélisation, les résultats expérimentaux ont montré qu'elle est incapable de résoudre des instances pour un nombre de nœuds supérieure à 50. Pour cette raison, nous avons développé en second lieu un algorithme génétique et une heuristique par décomposition permettant de résoudre des instances réelles de la ville de Nice et de la ville de Kaiserslautern.

Une autre approche qui nous a paru intéressante pour résoudre le problème GEP est de développée un algorithme inspiré du comportement des fourmis. Ce type d'algorithme a été utilisé pour résoudre essentiellement les problèmes d'évacuation des piétons ou des voitures (voir [Forcael et al., 2014], [Zong et al., 2010]) et a montré son efficacité dans le cas d'une évacuation des piétons (voir [Ndiaye et al., 2014b]). Pour les idées de base des algorithmes de fourmis le lecteur peut consulter les travaux de [Dorigo and Stützle, 2004].

Dans ce qui suit, nous décrirons brièvement comment nous avons adapté ce type d'heuristiques pour prendre en compte les spécificités de notre problème. Comme pour la méthode par décomposition, nous supposons que l'évacuation se fait en premier lieu par voitures et elle sera suivie par l'évacuation des piétons par bus. Notons que, les centres de secours sont choisis en utilisant le programme dynamique résolvant un problème de sac à dos multidimensionnel, comme expliqué dans la section 5.1. L'approche utilisée pour l'énumération du front de Pareto est $\epsilon-$contrainte. Nous cherchons à minimiser la somme des risques pour une date de fin d'évacuation fixée.

Dans cet algorithme, une fourmi représente un bus ou une voiture. Donc chaque colonie construit une solution qui est l'établissement d'un plan d'évacuation pour les voitures et les bus. Pour évacuer une fourmi vers un centre de secours, une décision doit être prise à chaque nœud. Tout d'abord, nous commençons à déterminer l'ensemble des nœuds

possibles qu'une fourmi pourrait prendre. Vu que la durée d'évacuation est fixée T_{evac}^{exp}, il est nécessaire de vérifier que la fourmi ne s'engage pas sur un nœud qui ne lui permettra pas de rejoindre une destination avant cette date.

Nous associons une quantité de phéromones à chaque arc (i, j) qui mène à une destination d. Cette quantité de phéromone permet de définir la probabilité qu'une fourmi choisisse le nœud j pour atteindre la destination d. Cette destination peut être un centre de secours lors d'une évacuation d'une fourmi bus ou une fourmi voiture. Dans le cas d'une fourmi de type bus contrainte à retourner vers un point de rassemblement pour évacuer des personnes, cette destination est un point de rassemblement. Notons que ces valeurs sont initialisées à 1 au début de l'algorithme.

La partie importante de cet algorithme est la phase d'apprentissage du déplacement des fourmis. Cette phase permet, d'une part, aux fourmis de déterminer les chemins les plus sûrs respectant la condition imposée sur la date de fin d'évacuation. D'autre part, elle assure la convergence de la matrice des phéromones vers des valeurs stables. Pour assurer la convergence de cette matrice, plusieurs itérations sont nécessaires. Le but de cette phase est d'obtenir des quantités de phéromones proportionnelles au risque d'un chemin depuis sa source vers sa destination. Ces chemins les moins risqués auront des valeurs importantes de phéromones. Donc il n'y a pas un seul chemin qui sera mis en valeur mais un ensemble de chemins pour pallier les possibilités d'embouteillages.

Chapitre 4

Problème d'ordonnancement et de localisation "ScheLoc"

Dans ce chapitre, nous nous intéressons à l'étude d'un problème d'ordonnancement et de localisation (*ScheLoc Problem*). Dans la section 2, nous présentons un état de l'art sur les problèmes d'ordonnancement et de localisation. Dans la section 3, nous introduisons formellement le problème d'ordonnancement et de localisation. Dans la section 4, nous présentons une modélisation mathématique de ce problème et nous proposons plusieurs bornes inférieures. Ensuite, la section 5 est consacrée à la résolution approchée en utilisant des heuristiques. Dans la section 6, nous présentons des méthodes de recherche locale pour améliorer les solutions obtenues par ces heuristiques. Les méthodes proposées dans les deux dernières sections sont inspirées du problème de clusterisation (*Clustering problem*). La section 7 est dédiée aux résultats expérimentaux. Enfin, ce chapitre se termine par une conclusion.

Les résultats des travaux présentés dans ce chapitre sont le fruit d'une collaboration avec Corinna Hessler doctorante à l'Université de Kaiserslautern (Allemagne) et membre du projet DSS_Evac_Logistic.

1 Contribution du chapitre

Dans ce chapitre, nous avons étudié le problème de localisation d'un ensemble de machines identiques et d'ordonnancement des travaux de ces machines. L'objectif est de minimiser la date de fin d'ordonnancement. À notre connaissance ce problème n'a pas été abordé. Pour la résolution de ce problème, nous avons proposé un modèle mathématique et des heuristiques de clusters. Ces travaux ont fait l'objet de deux communications dans des conférences internationales [Hessler and Deghdak, 2015a, Hessler and Deghdak, 2015b] et qui seront soumis prochainement à une revue internationale [Hessler and Deghdak, 2015c].

2 État de l'art

Les problèmes d'ordonnancement et de localisation (ScheLoc) combinent deux problèmes très étudiés dans en Recherche Opérationnelle : le problème d'ordonnancement et le problème de localisation. La résolution des problèmes d'ordonnancement et de localisation implique deux niveaux de décision ; la localisation des machines qui est une décision stratégique (sur le long terme) et l'ordonnancement des travaux sur ces machines qui est une décision opérationnelle (sur le court terme). Ces deux décisions sont souvent interdépendantes et le traitement séparé des deux décisions nous donne une solution approchée du problème complet. En réalité, plusieurs applications nécessitent la considération simultanée des deux problèmes. Nous prenons l'exemple cité dans [Kalsch, 2009]. Dans un port, une cargaison de conteneurs doit être chargée sur des navires. Dans cette application, il faut trouver les meilleures positions pour les navires sur les quais (localisation) et gérer le processus de chargement des conteneurs correspondants aux navires (ordonnancement). Traditionnellement, on résout d'abord le problème de localisation puis le problème d'ordonnancement. Cependant, la solution optimale du problème complet n'est pas forcément la solution optimale pour les deux problèmes. Quand une approche intégrée est proposée, les meilleures positions pour les navires et les séquences de chargement des conteneurs sont calculées simultanément.

Le problème d'ordonnancement-localisation (*ScheLoc*) a été introduit par [Hennes, 2005]. [Hennes, 2005] a étudié le problème de localisation d'un ensemble de machines sur un réseaux et l'ordonnancement des travaux sur ces machines. Le but est de minimiser la date de fin d'ordonnancement (*N-ScheLoc*). Ensuite [Hamacher and Hennes, 2007] ont proposé des algorithmes polynomiaux pour le cas spécial d'une seule machine placée à une position quelconque dans un réseau. [Elvikis et al., 2008] sont les premiers qui ont abordé le problème d'ordonnancement à une machine localisée sur un plan (*P-ScheLoc*). Les travaux sont ordonnancés sur cette machine afin de minimiser la date de fin du dernier travail. Ils ont proposé des algorithmes polynomiaux pour la résolution de ce problème.

Ensuite [Kalsch, 2009] a défini le problème *ScheLoc* Universel. Dans ce problème, plusieurs machines doivent être localisées sur un espace de localisation et les travaux doivent être ordonnancés sur ces machines afin de minimiser une fonction universelle. Cette fonction est supposée croissante des dates de fin des travaux. [Kalsch and Drezner, 2010] ont proposé des méthodes pour résoudre le cas spécial à une seule machine du ScheLoc Universel. Par contre, aucun algorithme n'est proposé pour le cas de plusieurs machine. Récem-

ment, [Kaufmann, 2014] a proposé un algorithme polynomiale pour résoudre le problème *ScheLoc* Universel à une machine où la préemption des travaux est autorisée.

Dans ce chapitre, nous nous intéressons à l'étude du problème d'ordonnancement et de localisation où la préemption des travaux n'est pas autorisée. Ce problème porte sur la localisation d'un ensemble fini de machines identiques sur un espace de localisation (plan, réseau, discret) dans le but de définir les meilleures localisations pour les machines et d'ordonnancer un ensemble des travaux sur ces machines. Ce problème est un cas particulier du problème introduit par [Kalsch, 2009] et la fonction objectif est la minimisation la date de fin d'ordonnancement. Ce problème sera noté par *DPMM ScheLoc* (*Discrete Parallel Machine Makespan scheduling and location*).

Le problème étudié dans ce chapitre est le problème *DPMM ScheLoc*. Ce dernier est $\mathcal{NP}$-difficile puisque deux de ses sous-problèmes le sont. Nous nous intéressons dans ce chapitre principalement au développement d'heuristiques. Ces heuristiques ont pour but de choisir des localisations pour les machines, et d'ordonnancer les travaux sur ces machines afin de minimiser la date de fin d'ordonnancement. Chaque travail du sous-ensemble des travaux placés sur une machine doit être transféré depuis sa localisation initiale vers la localisation de cette machine. Cette durée de déplacement correspond dans les problèmes d'ordonnancement à une date de début au plut tôt. Il évident que la date de début au plut tôt est dépendante de la distance entre la localisation des travaux et la localisation des machines. Par conséquent, la date de fin d'ordonnancement est aussi dépendante de l'affectation des travaux sur ces machines. Vu que *DPMM ScheLoc* et p-median ont pour but de minimiser une fonction de coût qui est dépend des distances entre les localisations des demandes (travaux) et les localisations possibles pour les activités (machines), alors, le problème de localisation que nous considérons ici est plus proche du problème p-median que du problème $p-$center.

Il existe plusieurs états de l'art sur les algorithmes de résolution du problème p-median ([Reese, 2006], [Mladenović et al., 2007], [Tansel et al., 1983], etc). Parmi les heuristiques de résolution proposées pour le problème p-median, certaines sont ceux basées sur la résolution de problème de clusterisation. Ce type d'heuristique a été proposé initialement par [Maranzana, 1964]. Le principe de ces heuristiques est de partitionner un ensemble d'objets en des ensembles disjoints en tenant compte de la similarité entre les objets du même ensemble. Dans le contexte de localisation, le problème de clusterisation consiste à trouver les p centres des clusters et d'affecter des demandes à ces clusters, de telle sorte que la somme des distances entre la localisation des demandes et le centre de cluster soit minimale. Nous renvoyons le lecteur intéressé à [Xu and Wunsch, 2005] pour un état de l'art sur les algorithmes de cluster. Nous avons adapté ce type d'heuristiques pour la résolution du problème *DPMM ScheLoc*, en intégrant les données d'ordonnancement dans les phases de calcul des centres de cluster (localisation des machines) et d'affectation des travaux à ces clusters. Ensuite, nous calculons l'ordonnancement optimal des travaux affectés à chaque cluster.

À première vue, le problème *DPMM ScheLoc* peut être considéré comme un sous-problème du problème d'évacuation général du chapitre 3 en ne considérant que les deux critères qui sont la date de fin d'évacuation et le nombre de centres à ouvrir. En effet, les centres de secours sont les machines, les travaux sont les groupes de personne associés

à chaque point de rassemblement et les déplacements entre les centres de secours et les points de rassemblement sont assurées par bus ou par voitures. De plus, les durées de déplacement sont calculées par le plus court chemin, tandis que, ces durées sont calculées dans le problème d'évacuation général sur le graphe réel. Mais en réalité, ces deux problèmes présentent des dissimilitudes : *DPMM ScheLoc* suppose que les localisations des machines (les centres de secours) ne sont pas définis et l'un de ses buts est de définir leurs positions effectives. Alors que dans le problème d'évacuation général, les centres de secours sont positionnés et le but est de choisir parmi ces centres de secours, les centres réellement opérationnels. Aussi, contrairement au problème d'évacuation général, le problème *DPMM ScheLoc* permet d'appréhender l'ordonnancement des travaux (les groupes de personnes) sur les machines (les centres de secours). Dans un contexte d'évacuation, cet ordonnancement permet de tenir compte, par exemple, de l'état de santé des personnes arrivant aux centres de secours qui nécessitent souvent des examens médicaux et psychologiques.

Le problème d'évacuation général vise à calculer la date de fin d'évacuation des personnes, donc plusieurs groupes d'évacués arrive même temps à un centre de secours, pour le problème *DPMM ScheLoc* nous considérons le séquencement des travaux sur mes machines (centres de secours) afin de minimiser leurs date de fin. Ce séquencement est un aspect important dans le cas des problème d'ordonnancement où certains groupe de personnes nécessite des examens médicaux et psychologique. Donc l'intérêt théorique par rapport au projet est de s'intéresser au séquencement des groupes des personnes lors de leurs arrivées au centre de secours.

3 Définition du problème

Nous considérons le problème de localisation et d'ordonnancement à machines parallèles. Ce problème peut être défini comme suit : Étant donné un espace de localisation (plan, réseau, discret), un ensemble $\mathcal{M} = \{1, \ldots, m\}$ de localisations possibles pour placer les machines, un nombre de p machines identiques parallèles, un ensemble $\mathcal{N} = \{1, \ldots, n\}$ de localisations des travaux sur cet espace, et une matrice de distances $D \in \mathbb{R}^{n \times m}$ avec $D(i,k) = dist(i,k)$, et $dist(i,k)$ la distance entre la localisation du travail $i \in \mathcal{N}$ et la localisation d'une machine $k \in \mathcal{M}$. De plus, un travail i est défini par sa localisation, sa durée opératoire p_i et sa date de début au plut tôt r_{ik} s'il est placé sur une machine de localisation k. La date de début au plut tôt dépend des données suivantes : la distance $dist(i,k)$, la date à partir de laquelle le travail i est disponible sur sa localisation initiale, notée σ_i et la vitesse de déplacement de ce travail, notée ν_{ik}, depuis sa localisation vers une localisation d'une machine k. Ainsi, la date de début au plus tôt r_{ik} du travail i est calculée comme suit :

$$r_{ik} = \sigma_i + \tau_{ik} dist\left(a_i, v_k\right),$$

où $\tau_{ik} = 1/\nu_{ik}$.

Le problème *DPMM ScheLoc* consiste à localiser p machines à partir de l'ensemble $\mathcal{M}$ et d'ordonnancer tous les travaux $i \in \mathcal{N}$ sur ces machines. L'ordonnancement des travaux doit respecter les dates r_{ik} de début au plus tôt. L'objectif est de minimiser $C_{\max} = \max\{C_i | i \in \mathcal{N}\}$ où C_i est la date de fin du travail i.

Pour mieux illustrer cette définition, nous utilisons l'exemple suivant.

Exemple 8 *Soit le graphe $G = (V, E)$ représenté dans la figure 4.1 où V est l'ensemble des nœuds et E l'ensemble des arcs. Nous supposons que $\mathcal{N} = \mathcal{M} = V$, c'est-à-dire : sur chaque nœud du graphe un travail est localisé sur chaque nœud du graphe. Les machines seront localisées sur un sous-ensemble de V.*

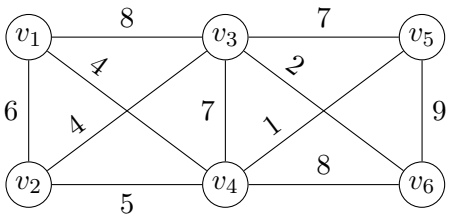


FIGURE 4.1 – Graphe $G = (V, A)$

Nous prenons le nombre de machines $p = 2$. Soit le vecteur $P = (6, 1, 2, 4, 5, 3)$ qui représente les durées opératoires des travaux. La matrice de distances $D \in \mathbb{R}^{n \times m}$ est calculée par le plus court chemin dans ce graphe.

$$D = \begin{pmatrix} 0 & 6 & 8 & 4 & 7 & 12 \\ 6 & 0 & 4 & 5 & 8 & 6 \\ 8 & 4 & 0 & 7 & 7 & 2 \\ 4 & 5 & 7 & 0 & 1 & 8 \\ 7 & 8 & 7 & 1 & 0 & 9 \\ 12 & 6 & 2 & 8 & 9 & 0 \end{pmatrix}$$

Posons $\sigma_2 = 1$, $\sigma_4 = 2$, et $\sigma_i = 0$ pour $i \in \{1, 3, 5, 6\}$, $\nu_{41} = 2$, $\nu_{61} = 1/2$ et $\nu_{ik} = 1$ pour $i \in \mathcal{N}/\{1, 3, 5, 6\}$ et $k \in \mathcal{M}$. Les dates de début au plus tôt sont calculées à partir des distances comme suit : $r_{2k} = dist(2, k) + 1$ pour tout $k \in \mathcal{M}$, $r_{41} = \frac{1}{2}dist(4, 1) + 2 = 4$, $r_{4k} = dist(4, k) + 2$ pour $k \neq 1$, $r_{61} = 2dist(6, 1) = 24$, et $r_{ik} = dist(i, k)$ pour tout autre couple (i, k). Nous obtenons la matrice des dates de début au plus tôt suivante :

$$R = \begin{pmatrix} 0 & 6 & 8 & 4 & 7 & 12 \\ 7 & 1 & 5 & 6 & 9 & 5 \\ 8 & 4 & 0 & 7 & 7 & 2 \\ 4 & 7 & 9 & 2 & 3 & 10 \\ 7 & 8 & 7 & 1 & 0 & 9 \\ 24 & 6 & 2 & 8 & 9 & 0 \end{pmatrix}$$

Supposons que nous localisons les machines sur les nœuds v_1 et v_5. Par la suite, les travaux doivent être ordonnancés sur ces machines. Dans cet exemple, un travail est affecté à une machine sur laquelle la date de début au plut tôt est minimale. Donc, les travaux 1 et 2 sont affectés à la machine localisée sur v_1. Quant aux travaux restants, ils sont affectés à la machine localisée sur v_5. Pour ces affectations, nous calculons l'ordonnancement optimal sur chaque machine en ordonnançant les travaux selon l'ordre croissant de leur date de début au plut tôt. Ce qui donne l'ordonnancement présenté dans la figure 4.2 :

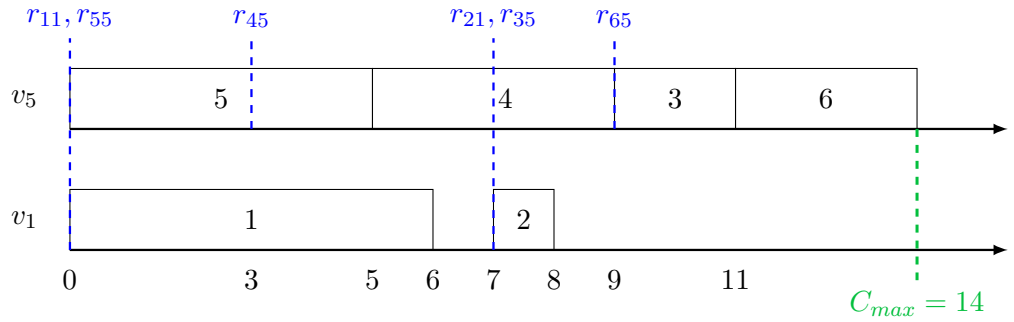


FIGURE 4.2 – Ordonnancement des travaux sur les machines localisées sur v_1 et v_5

La date de fin de l'ordonnancement est donc $C_{\max} = 14$. *Celle date n'est pas optimale vu que l'affectation du travail 4 à la localisation* v_1 *donne une valeur* $C_{\max} = 12$.

Cet exemple montre que le traitement du problème de localisation indépendamment du problème d'ordonnancement peut donner des solutions qui ne sont pas optimales. Par conséquent, dans les sections suivantes, nous présentons deux approches exacte et heuristique pour calculer des solutions intégrées.

4 Formulation mathématique et bornes inférieures

Pour résoudre le problème *DPMM ScheLoc* à l'optimalité, nous utilisons une formulation mathématique en nombres entiers. De plus, afin de mesurer la qualité des heuristiques, nous proposons des bornes inférieures pour des instances de grande taille.

4.1 Formulation mathématique

Puisque le problème *DPMM ScheLoc* combine le problème de localisation et le problème d'ordonnancement, la modélisation (IP) que nous proposons ici combine aussi les deux formulations de ces deux problèmes. Pour cette fin, nous introduisons les variables d'ordonnancement C_{max}, x_{ik} et y_{ij}^k, $\forall i \in \mathcal{N}$, $\forall j \in \mathcal{N}$ et $\forall k \in \mathcal{M}$. La variable x_{ik} est égale à 1, si le travail i est ordonnancé sur une machine placée sur la localisation k et 0 sinon. La variable y_{ij}^k est égale à 1, si le travail i est le prédécesseur direct du travail j sur la machine k et 0 sinon. Les variables de localisation sont notées z_k : la variable z_k égale à 1, si une machine est placée sur la localisation $k \in \mathcal{M}$, et 0 sinon. De plus, nous avons besoin de deux travaux fictifs, le travail 0 et le travail $n+1$. Leurs dates de début au plus tôt et leurs durées opératoires sont égales à 0 sur toutes les machines. Soit M une borne supérieure du critère C_{max}, par exemple, $M = \sum_{i \in \mathcal{N}} p_i + \max_{k \in \mathcal{M}} \{r_{ik}\}$. Le problème *DPMM ScheLoc* est formulé comme suit :

$$\min C_{max} \tag{4.1}$$

$$\text{s.c. } C_i \geq \sum_{k\in\mathcal{M}} r_{ik}x_{ik} + p_i, \qquad \forall i \in \mathcal{N} \tag{4.2}$$

$$C_i \geq C_j + p_i - M\left(1 - y_{ji}^k\right), \qquad \forall i,j \in \mathcal{N}, k \in \mathcal{M} \tag{4.3}$$

$$C_{max} \geq C_i, \qquad \forall i \in \mathcal{N} \tag{4.4}$$

$$\sum_{j\in\mathcal{N}\cup\{n+1\}} \sum_{k\in\mathcal{M}} y_{ij}^k = 1, \qquad \forall i \in \mathcal{N}\cup\{0\} \tag{4.5}$$

$$\sum_{i\in\mathcal{N}\cup\{0\}} \sum_{k\in\mathcal{M}} y_{ij}^k = 1, \qquad \forall j \in \mathcal{N}^{\cup}\{n+1\} \tag{4.6}$$

$$y_{0i}^k \leq x_{ik}, \qquad \forall i \in \mathcal{N}, k \in \mathcal{M} \tag{4.7}$$

$$y_{ij}^k \leq \frac{1}{2}\left(x_{ik} + x_{jk}\right), \qquad \forall i,j \in \mathcal{N}, k \in \mathcal{M} \tag{4.8}$$

$$\sum_{k\in\mathcal{M}} z_k \leq p \tag{4.9}$$

$$\sum_{j\in\mathcal{N}} y_{0j}^k \leq z_k, \qquad \forall k \in \mathcal{M} \tag{4.10}$$

$$\sum_{j\in\mathcal{N}} x_{jk} \leq nz_k, \qquad \forall k \in \mathcal{M} \tag{4.11}$$

$$\sum_{k\in\mathcal{M}} x_{ik} = 1, \qquad \forall i \in \mathcal{N} \tag{4.12}$$

$$C_0 = 0 \tag{4.13}$$

$$C_i \geq 0, \qquad \forall i \in \mathcal{N} \tag{4.14}$$

$$y_{ii}^k = 0, \qquad \forall i \in \mathcal{N}, k \in \mathcal{M} \tag{4.15}$$

$$x_{ik}, y_{ij}^k, z_k \in \{0,1\}, \qquad \forall i,j \in \mathcal{N}, k \in \mathcal{M}. \tag{4.16}$$

L'objectif (4.1) est la minimisation de la date de fin d'ordonnancement. Les contraintes (4.2)- (4.8) et(4.13) - (4.15) sont des contraintes sur l'ordonnancement des travaux. Les contraintes (4.9) sont des contraintes sur la localisation des machines. Tandis que les contraintes (4.10) - (4.12) sont des contraintes de liaison entre les variables d'ordonnancement et les variables de localisation.

Les contraintes (4.2) assurent qu'un travail ne peut pas être ordonnancé avant sa date de début au plut tôt. Les contraintes (4.3) assurent qu'un travail n'est ordonnancé qu'après la date de fin de son prédécesseur. Les contraintes (4.4) permettent de calculer la date de fin d'ordonnancement. Pour assurer l'unicité du prédécesseur et du successeur d'un travail

$i \in \mathcal{N}$, nous avons besoin des contraintes (4.5) et (4.6). Les contraintes (4.7) stipulent qu'un seul travail est ordonnancé en première position sur une machine k. Les contraintes (4.8) assurent que le prédécesseur et le successeur d'un travail sont placés sur la même machine.

Les contraintes (4.9) assurent que le nombre de localisations utilisées est égal à p. Les contraintes (4.10) forcent, d'une part, qu'un seul ordonnancement soit placé sur une seule machine et que, d'autre part, les machines soient placées sur des localisations choisies réellement. Les contraintes (4.11) assurent que les travaux sont ordonnancés sur les machines où elles sont réellement localisées. Les contraintes (4.12) stipulent que chaque travail est placé sur une machine. Les contraintes (4.13) - (4.16) sont des contraintes d'intégrité des variables.

Le nombre de variables de cette formulation est de l'ordre $\mathcal{O}(n^2m)$ et le nombre de ses contraintes est de l'ordre $\mathcal{O}(n^2m)$. À part pour des instances de petite taille, cette formulation est hors de portée des solveurs commerciaux (cf. la section 7).

4.2 Bornes inférieures

Afin de mesurer la qualité des heuristiques, nous suggérons plusieurs bornes inférieures pour le problème *DPMM ScheLoc*. La borne la plus intuitive est la borne inférieure du problème d'ordonnancement à machines parallèles avec minimisation de la date de fin d'ordonnancement, $P_m||C_{\max}$. Ce problème est un cas particulier du problème *DPMM ScheLoc* où $D = \mathbf{0}$, $\sigma_i = 0$, $\forall i \in \mathcal{N}$ et $\nu_{ik} = 1$ pour tout $\forall i \in \mathcal{N}$ et $\forall k \in \mathcal{M}$. La borne inférieure la plus connue pour le problème $P_m||C_{\max}$ est donnée par :

$$C_{\max} \geq \frac{\sum_{i \in \mathcal{N}} p_i}{p}. \tag{4.17}$$

avec $r_{ik} = 0, \forall i \in \mathcal{N}$ et $\forall k \in \mathcal{M}$. Cette borne n'est pas la meilleure des bornes du problème *DPMM ScheLoc*, car la majorité des r_{ik} sont strictement positifs. Pour cette raison, nous ajoutons à cette borne la date au plut tôt minimale :

$$C_{\max} \geq \frac{\sum_{i \in \mathcal{N}} p_i}{p} + \min_{k \in \mathcal{M}} \min_{n \in \mathcal{N}} r_{ik}. \Longleftrightarrow LB1 = \frac{\sum_{i \in \mathcal{N}} p_i}{p} + \min_{k \in \mathcal{M}} \min_{n \in \mathcal{N}} r_{ik}$$

Pour améliorer ces bornes dans le cas où le nombre de travaux est égal au nombre de localisations disponibles ($\mathcal{N} = \mathcal{M}$), la borne inférieure de la date de fin du travail i ordonnancé sur une machine placée sur sa localisation est $LB(C_i) = p_i$. Dans le cas contraire, une borne inférieure de sa date de fin est $LB(C_i) = r_i + p_i$. Donc, une borne inférieure du critère est $C_{\max} \geq \max\{LB(C_i)\}$. De plus, nous supposons que les machines sont placées sur les p localisations où les travaux i ont la plus grande valeur de $r_i + p_i$. Soit K' l'ensemble des travaux correspondants. Pour les travaux restants, nous obtenons $LB(C_i) = r_i + p_i$. Nous pouvons borner inférieurement la date de fin d'ordonnancement par :

$$C_{\max} \geq \max_{i \notin K'}\{r_i + p_i\} \Longleftrightarrow LB2 = \max_{i \notin K'}\{r_i + p_i\}.$$

Enfin, nous proposons une borne inférieure qui dépend de la date de début au plus tôt minimale et la structure des dates de début au plus tôt. Pour cela, nous définissons la charge d'une machine k par $l(k) = \sum_{i \in \mathcal{N}} p_i p$. Pour chaque $k \in \mathcal{M}$, nous résolvons à l'optimalité le problème d'ordonnancement de tous les travaux sur une machine (règle ERD) jusqu'à ce que $l(k)$ soit atteinte. Soit $C_{\max}(k)$ la date de fin d'ordonnancement d'un sous-ensemble de travaux sur la machine k. Nous obtenons la borne inférieure suivante :

$$C_{\max} \geq \min_{k \in \mathcal{M}} \{C_{\max}(k)\} \Longleftrightarrow LB3 = \min_{k \in \mathcal{M}} \{C_{\max}(k)\}. \tag{4.18}$$

Pour montrer effectivement que $C_{\max} \geq \min_{k \in \mathcal{M}} \{C_{\max}(k)\}$ est une borne inférieure au problème *DPMM ScheLoc*, nous procédons de la façon suivante : soit une solution réalisable du problème, alors il existe au moins une machine qui a une charge $l(k) \geq \frac{\sum_{i \in \mathcal{N}} p_i}{p}$. Dans le cas contraire, certains travaux ne seront pas placés. Puisque nous utilisons la règle ERD pour ordonnancer les travaux sur la machine k, alors pour $l(k) \geq \frac{\sum_{i \in \mathcal{N}} p_i}{p}$(solution optimale), la date d'ordonnancement minimale de la machine k est $C_{\max}(k)$. Soit la localisation k' telle que $k' = \arg\min_{k \in \mathcal{M}} \{C_{\max}(k)\}$. Si une machine est placée sur la localisation k' dans une solution optimale, et cette localisation vérifie $l(k') \geq \frac{\sum_{i \in \mathcal{N}} p_i}{p}$, alors 4.18 est une borne inférieure. Sinon, il existe une machines k'' qui vérifie $l(k'') \geq \frac{\sum_{i \in \mathcal{N}} p_i}{p}$ avec $C_{\max}(k'') \geq C_{\max}(k')$. Donc, (4.18) est réellement une borne inférieure.

Pour les instances où p est légèrement plus grand ou égal à n, nous proposons la borne inférieure suivante :

$$C_{\max} \geq \max_{i \in \mathcal{N}} \{\min_{k \in \mathcal{M}} r_{ik} + p_i\} \Longleftrightarrow LB4 = \max_{i \in \mathcal{N}} \{\min_{k \in \mathcal{M}} r_{ik} + p_i\}.$$

La borne inférieure du problème *DPMM ScheLoc* est $LB = \max\{LB1, LB2, LB3, LB4\}$.

5 Heuristiques de cluster

Dans cette section nous nous intéressons à la résolution approchée de notre problème. Nous proposons des heuristiques de cluster qui permettent de fournir de bonnes solutions en un temps raisonnable. La résolution du problème *DPMM ScheLoc* revient à résoudre les trois sous-problèmes : (1) définir les différentes localisations des machines, (2) affecter les travaux à ces machines (3) ordonnancer chaque affectation sur la machine correspondante. Une fois les sous-problèmes (1) et (2) résolus, le troisième sous-problème est résolu en temps polynomiale selon la règle ERD ($1|r_i|C_{max}$). Par conséquent, la résolution approchée *DPMM ScheLoc* se ramène à la résolution des sous-problèmes (1) et (2). Afin d'obtenir des solutions de bonnes qualités, nous intégrons les données d'ordonnancement lors du choix des localisations pour les machines et aussi lors des affectations des travaux sur ces machines. Nous proposons une approche séquentielle en utilisant le problème de clusterisation pour résoudre les sous-problèmes (1) et (2). De manière formelle, un problème de clusterisation est défini par les données suivantes : un ensemble de localisations possibles $\mathcal{M}$, un ensemble $\mathcal{N}$ de localisations des demandes d_i pour chaque $i \in \mathcal{N}$, et une

fonction de distance $dist\,(i,k)$ pour $i \in \mathcal{N}$ et $k \in \mathcal{M}$. Le but est de trouver un sous-ensemble $\mathcal{C} \subset \mathcal{M}$ de cardinal p appelé centre de clusters et une affectation des demandes $i \in \mathcal{N}$ aux centre des clusters $C_k \in \mathcal{C}$ afin de minimiser $\sum_{C_k \in \mathcal{C}} \sum_{i \in C_k} d_i dist\,(i,k)$. Dans le cas où les clusters ont une capacité suffisamment grande (*Uncapacited cluster*), chaque localisation de la demande i est affectée, indépendamment des valeurs d_i, au centre du cluster C_k le plus proche.

Les sous-problèmes (1) et (2) sont étroitement liés au problème de clusterisation avec des capacités. Comme nous cherchons à minimiser la date de fin d'ordonnancement, cette date n'est déterminée que lorsque les différentes localisations des machines sont choisies et les travaux sont affecter sur ces machines. Pour ces raisons, les affectations des demandes aux centres des clusters et les capacités des clusters sont indéfinies. Par conséquent, les sous-problèmes (1) et (2) joints peuvent être assimilés à un problème de clusterisation sans capacité. Dans ce cas, le centre d'un cluster correspond à une localisation d'une machine et le critère d'affectation des travaux (demandes) est modifié afin de tenir compte des données d'ordonnancement.

Nous considérons trois types d'heuristiques pour résoudre le problème de clusterisation :

1. D'abord, nous choisissons les centres des clusters, ensuite, nous affectons les travaux aux clusters.
2. D'abord, nous construisons p clusters de travaux, ensuite, nous affectons pour chaque cluster une localisation d'une machine (centre du cluster).
3. Nous choisissons itérativement un centre de cluster et nous affectons à ce centre les travaux jusqu'à l'obtention de p clusters.

Pour chaque type d'heuristique, nous définissons les critères pour les décisions de localisation et de construction des clusters. Dans les sections suivantes, nous présentons en détails ces trois types d'heuristique et les critères associés.

5.1 Localisation et cluster (LC)

Dans ce type d'heuristique, les p centres des clusters sont construits en ajoutant une machine à chaque cluster. Ensuite, nous ajoutons les travaux à ces clusters. Nous proposons les différents critères suivant pour le choix des centres des clusters (localisation des machines) :

1. Les localisations sont choisies aléatoirement.
2. Sélectionner les k localisations des machines tel que $\max_{i \in \mathcal{N}} r_{ik} + p_i$ soit minimale.
3. Sélectionner les k localisations des machines tel que $\sum_{i \in \mathcal{N}} r_{ik}$ soit minimale.
4. Sélectionner les k localisations des machines tel que r_{i^*k} soit minimale, pour $i^* \in \arg\max_{i \in J} \left\{ \sum_{l=1}^{k-1} r_{il} \right\}$.

Dans le premier critère, le choix des localisations est arbitraire, les données d'ordonnancement sont prise en compte lors de la phase d'affectation des travaux sur les clusters. Une meilleure façon de choisir les localisations, consiste à intégrer ces données dans la phase de localisation des machines, comme c'est le cas pour les critères 2, 3 et 4. Dans le deuxième critère, nous calculons une borne inférieure de la date de fin de chaque travail sur chaque machine. Nous choisissons les localisations de sorte que le maximum de

cette borne inférieure soit minimal. Ce critère, sélectionne les localisations les plus proches des travaux dont les durées opératoires sont grandes. Ceci évite l'arrivée tardive de ces travaux sur une machine causant une augmentation de la date de fin d'ordonnancement, notamment, si cette machine été inactive avant. L'inconvénient de ce critère est que les localisations sont éloignées des travaux ayant des petites durées opératoires ; la date de fin d'ordonnancement augmente en conséquence si plusieurs de ces travaux arrivent simultanément sur une machine. Pour remédier ce problème, nous proposons le troisième critère : nous choisissons les localisations qui minimisent la somme des dates de début au plus tôt de tous les travaux. Ce critère, permet de choisir uniquement les localisations pour les machines qui sont proches de tous les travaux. Donc, plusieurs travaux arrivent tôt sur une machine, ce qui réduit son temps d'inactivité. L'inconvénient commun des critères 1, 2 et 3 est que la localisation des machines se fait simultanément. Donc, une meilleure façon pour localiser les machines est de choisir les localisations séquentiellement, pour tenir compte des décisions antérieures. Cela est fait à l'aide du quatrième critère. Dans ce critère, nous choisissons aléatoirement la première localisation. Supposons $k-1$ localisations choisies pour les machines, nous calculons la somme des dates de début au plus tôt pour chaque travail i sur les $k-1$ localisations. Ensuite, la localisation k est choisie proche du travail i^* ayant une somme maximale des dates de début au plus tôt. Cela qui conduit à choisir une nouvelle localisation proche du travail le plus éloigné des localisations déjà sélectionnées. En revanche, nous pouvons avoir parmi les localisations choisies celles qui sont proche d'un travail et éloignées de tous les autres. Ce problème sera résolu dans les versions 2 et 3 des heuristiques de cluster (cf. les sections 5.2 et 5.3).

Une fois que les localisations des machines ont été sélectionnées, chaque machine est placée dans un cluster sur l'une de ces localisations. Nous devons alors affecter et ordonnancer les travaux sur ces machines. Pour ce faire, nous utilisons la règle ERD modifiée : un travail est ordonnancé sur une machine libre, si sa date de début au plut tôt sur cette machine est minimale. Cette règle a deux avantages : elle permet d'intégrer les données d'ordonnancement lors de l'affectation des travaux aux clusters, et de calculer un ordonnancement optimal sur chaque machine en ordonnançant les travaux selon l'ordre croissant de leur date de début au plus tôt.

Les étapes principales de cette heuristique sont présentées dans l'Algorithme 13

Algorithme 13 Localisation et cluster (LC)

Entrées: Une instance du problème *DPMM ScheLoc*

1: Sélectionner les p localisations en utilisant un des critères (1-4)
2: Ordonnancer les travaux en utilisant la règle ERD modifiée

Sorties: Un ensemble de p localisations des machines et un ordonnancement sur chaque machine

Pour chacun des quatre critères du choix des p localisations, nous associons les versions LC1, LC2, LC3 et LC4 de cette heuristique. Ces notations seront utilisées dans la section 7.

Exemple 9 *Soit une instance du problème* DPMM ScheLoc *présentée dans la figure 4.3. Nous disposons de trois localisations possibles L1, L2 et L3 pour les machines. Nous supposons avoir deux machines $p = 2$ à placer sur ces localisations. Nous cherchons à placer les travaux J1, J2,..., J7 sur les deux machines afin que le C_{max} soit minimale. Sachant que les localisations de machines et de travaux sont identifiées par leurs coordonnées dans le plan $\mathbb{R}^2$.*

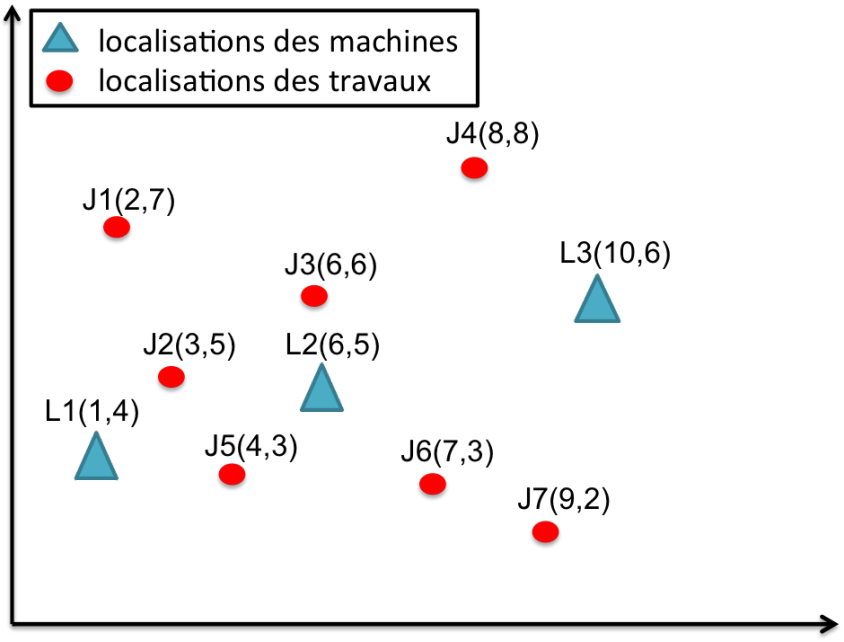


FIGURE 4.3 – Exemple d'une instance du problème *DPMM ScheLoc*

Soit le vecteur $P = (2, 3, 6, 4, 5, 3, 1)$ qui représente les durées opératoires des travaux. Nous supposons que tous les travaux sont présents sur leurs localisations à $t = 0$. Alors, la matrice des dates de début au plutôt $R \in \mathbb{R}^{n \times m}$ est calculée par la distance euclidienne entre les cordonnées des localisations de machines et les cordonnées des localisations de travaux :

$$R = \begin{pmatrix} 6.3 & 3.2 & 5 \\ 2.2 & 3 & 7.1 \\ 5.4 & 1 & 4 \\ 8.1 & 3.6 & 2.8 \\ 3.2 & 2.8 & 6.7 \\ 6.1 & 2.2 & 4.2 \\ 8.2 & 4.2 & 4.1 \end{pmatrix}$$

Nous utilisons l'heuristique LC2 pour calculer une solution du problème DPMM ScheLoc. *D'abord, nous calculons pour chaque localisation d'une machine k, les valeurs $\max_{i \in \mathcal{N}} r_{ik} + p_i$, nous obtenons les valeurs respectivement 36.4, 21,3 et 37,9 pour les localisations L1, L2*

et L3. Donc les localisations choisies pour placer les deux machines sont L1 et L2. Ensuite, nous ordonnançons les travaux sur les machines comme suit : un travail est ordonnancé sur une machine libre si ce travail a la plus petite valeur de la date de début au plutôt sur cette machine. L'ordonnancement obtenu par cette version d'heuristique est présenté dans la figure 4.4 et $C_{max} = 14$.

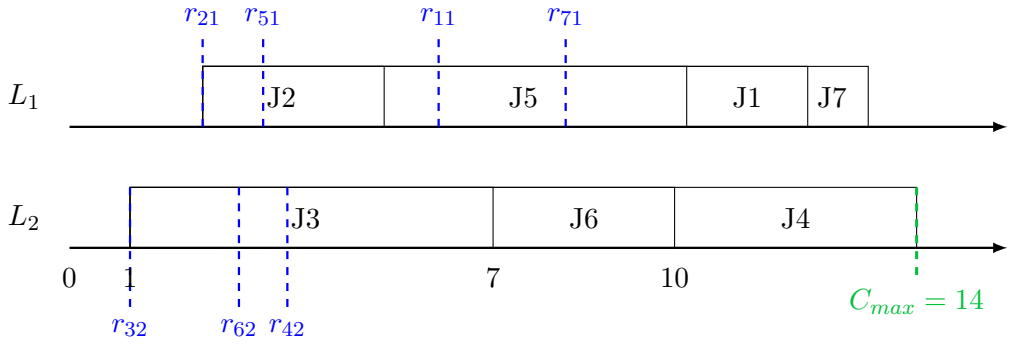


FIGURE 4.4 – Solution du problème *DPMM ScheLoc* calculée par LC2

5.2 Cluster et localisation (CL)

Dans cette version de l'heuristique, et contrairement à la première, nous construisons p clusters de travaux. Ensuite, nous affectons les localisations des machines (centre de cluster) à ces clusters. Dans l'étape de construction des p clusters de travaux, les dates de début au plus tôt sont inconnues, puisque les localisations des machines sont indéfinies. Par conséquent, cette construction est basée uniquement sur la distance entre les localisations des travaux. Comme ces distances ne font pas partie des données du problème *DPMM ScheLoc*, nous distinguons trois cas. Dans le cas discret, les instances sont générées aléatoirement sans aucune structure sur les distances. Pour ces instances nous ne pouvons pas calculer de solutions. Dans le cas où l'espace de localisation est un réseau (cf. example 8, page 145), les distances entre les localisations des travaux se fait selon le plus court chemin. Enfin, lorsque l'espace de localisation est un plan, les distances entre les localisations des travaux est obtenues à l'aide de la distance euclidienne.

Pour construire les p clusters, nous affectons un travail de départ à chaque cluster, selon l'un des trois critères suivants :

1. Choisir aléatoirement les p premiers travaux.
2. Soit G le centre de gravité de toutes les localisations des travaux. Sélectionner les p travaux i qui maximisent $dist(G, i)$.
3. Soit G_i le centre de gravité d'un cluster $i = \{1, \ldots, p-1\}$. Nous construisons le prochain cluster $i+1$ en maximisant $dist(G_i, i+1)$.

Le premier critère choisi aléatoirement les p premiers travaux des p clusters. Le deuxième critère choisi les p premiers travaux simultanément en se basant sur les distances entre leurs localisations et le centre de gravité de toutes les localisations des travaux. Ce critère permet de mettre les travaux éloignés du centre de gravité dans des clusters différents. En

revanche, les travaux voisins (proches) et éloignés du centre de gravité peuvent être dans des clusters différents. Pour palier ce problème, nous proposons le troisième critère où un cluster $i+1$ est choisi itérativement à partir du cluster i, de sorte qu'il soit éloigné du centre de gravité du cluster i. Une fois que les p clusters sont construits, les travaux j restants sont successivement ajoutés au cluster dont le centre de gravité est proche de la localisation du travail j.

L'avantage de cette heuristique est qu'à partir des clusters de travaux construits, nous pouvons souvent calculer des localisations pour les machines sur lesquelles l'ordonnancement des travaux est optimal. Certaines précautions doivent être prisent, notamment, si plusieurs clusters ont sur la même localisation des dates de fin minimales. Dans ce cas et afin d'éviter de tester toutes les combinaisons des p clusters sur les m localisations, nous appliquons une heuristique pour affecter une localisation à un cluster de travaux : pour chaque cluster, nous calculons la date de fin d'ordonnancement sur chaque localisation. Ensuite, nous trions les localisations selon l'ordre croissant de leur date de fin d'ordonnancement. S'il existe une localisation k qui est optimale pour plus d'un cluster, nous identifions le cluster ayant la deuxième plus grande date de fin d'ordonnancement sur $k+1$ et nous l'affectons à la localisation k. Nous considérons ensuite les clusters restants avec leur localisation respective ayant la deuxième plus grand date de fin. Nous poursuivons le procédé pour les clusters qui sont en compétition sur les mêmes localisations, jusqu'à l'affectation d'une localisation à un cluster. Ensuite, les travaux de chaque cluster sont ordonnancés selon la règle ERD, sur la machine placée sur la localisation affectée à ce cluster.

Le pseudo-code de l'heuristique est donné par l'Algorithme 14.

Algorithme 14 Cluster et Localisation (CL)

Entrées: Instance du problème *DPMM ScheLoc*

1: Déterminer les p clusters des travaux en utilisant l'un des critères 1-3
2: Pour chaque travail restant j, affecter ce travail à un cluster dont le centre de gravité est proche de la localisation du travail j
3: Pour chaque cluster, calculer la date de fin d'ordonnancement sur chaque localisation
4: Affecter chaque cluster à la localisation qui donne une date de fin d'ordonnancement minimale
5: **Si** plusieurs clusters sont affectés à la même localisation **alors** :
6: Affecter le cluster qui a la plus grande valeur de la date de fin d'ordonnancement sur la prochaine localisation
7: Déplacer tous les autres clusters à la localisation qui a la prochaine grande date de fin d'ordonnancement
8: **finsi**
9: Répéter l'étape 4 jusqu'à à ce que chaque localisation soit affectée à un cluster

Sorties: Un ensemble de p localisations des machines et un ordonnancement sur chaque machine

Selon les trois critères du choix des p clusters des travaux, nous avons trois versions notées CL1, CL2 et CL3 de cette heuristique. Ces notations seront utilisées dans la section

7.

Exemple 10 *Reprenons l'exemple 9 de la page 152. Nous cherchons à calculer une solution réalisable pour le problème* DPMM ScheLoc *en utilisant l'heuristique CL2. Tout d'abord, nous calculons les cordonnées du centre de gravité* G *des localisations des travaux. Nous obtenons G=(5.5,4.8). Puis, nous construisons deux clusters de travaux C_1 et C_2, en calculant la distance entre toutes les localisations des travaux et le point G. Les deux premiers localisation des travaux les plus éloignés du G seront placés dans des clusters différents. Alors J1 est affecté à C_1 et J7 est affecté à C_2. Ensuite, nous ajoutons itérativement les travaux restants à ces deux clusters comme suit : un travail est ajouté a un cluster si la distance entre sa localisation et le centre de gravité de ce cluster est minimale. Enfin, on obtient deux clusters C_1=$\{J1, J2, J3, J4\}$ et C_2=$\{J5, J6, J7\}$.*

Une fois que les deux clusters de travaux sont construit, cherchons maintenant à placer les machines sur les localisations qui minimisent la date de fin d'ordonnancement. En d'autre terme, affecter une machine à chaque cluster de travaux et ordonnancer ces travaux sur cette machine. Pour ce faire, nous calculons l'ordonnancement des clusters sur les localisations possibles des machines. Le Tableau 4.1 présente les dates de fin d'ordonnancement des travaux de clusters sur chacune de ces localisations.

	L1	L2	L3
C_1	17.4	16	17.8
C_2	12.2	9.2	13.2

TABLE 4.1 – Les différentes dates de fin des travaux de clusters sur les localisations L1, L2 et L3

D'après le Tableau 4.1, la localisation L2 donne une date de fin d'ordonnancement minimale pour C_1 et C_2. Puisque C_2 a une date de fin minimale sur L_2 et qu'une localisation doit contenir une seule machine sur laquelle sont ordonnancés les travaux d'un seul cluster. Alors, nous cherchons pour C_2 la prochaine localisation qui lui procure un C_{max} minimal. Cette localisation est la localisation L1. Donc, les machines seront placées sur les localisations L1 et L2 et les travaux des clusters C_1 et C_2 seront ordonnancés respectivement sur les machines placées sur L2 et L1. La solution obtenue par cette heuristique est présentée dans la figure 4.5 et la date de fin de l'ordonnancement est $Cmax = 16$.

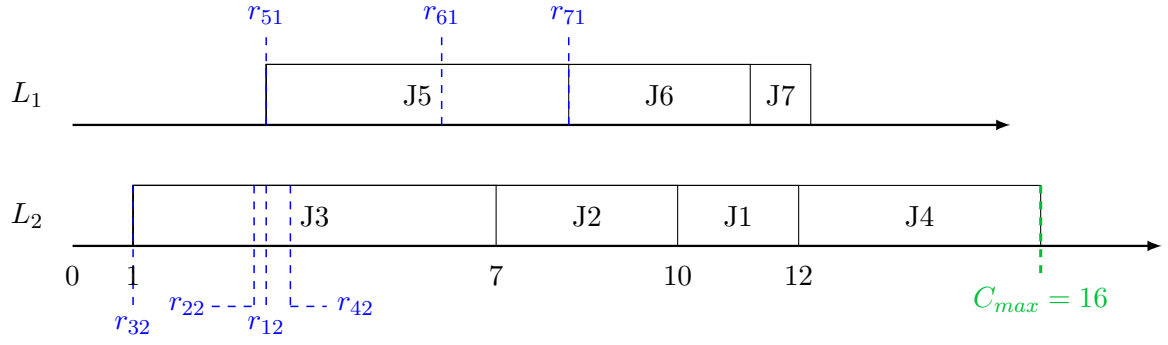


FIGURE 4.5 – Solution du problème *DPMM ScheLoc* calculée par CL2

5.3 Sélection itérative des clusters et des localisations (ICL)

Les heuristique proposées dans les sections 5.1 et 5.2 résolvent séquentiellement les problèmes de localisation et d'affectation. Dans cette section, nous proposons une troisième heuristique pour résoudre alternativement les problèmes de localisation et d'affectation.

Cette heuristique fonctionne de la façon suivante. Nous commençons par un seul cluster qui contient tous les travaux. Pour ce cluster, nous calculons une localisation optimale pour une machine. Ensuite, nous choisissons un ensemble de travaux qui seront déplacés depuis ce cluster vers un autre cluster. Nous calculons une nouvelle localisation optimale pour ce cluster. Nous réitérons ce procédé jusqu'à l'obtention de p clusters.

Il existe plusieurs possibilités pour choisir les localisations des machines et les affectations des travaux. Commerçons par les choix possibles de localisations :

1. Sélectionner la prochaine localisation k qui minimise la date de fin d'ordonnancement sur tous les travaux qui ne sont pas encore affectés.
2. Sélectionner la prochaine localisation k pour une machine, telle que celle-ci minimise la date de fin d'ordonnancement des $\lfloor n/p \rfloor$ travaux ayant les plus petites dates de début au plut tôt sur cette machine.

Le premier critère sélectionne une localisation adéquate pour tous les travaux non affectés, tandis que le deuxième critère minimise uniquement la date de fin d'ordonnancement des $\lfloor n/p \rfloor$ travaux proches de la localisation de cette machine. Ce dernier critère, suppose que les travaux sont répartis équitablement sur chaque machine.

Pour l'étape d'affectation des travaux, nous utilisons les critères suivants :

1. Considérons un ordonnancement optimal pour tous les travaux non affectés sur la machine k. Nous supprimons le premier travail j dans cet ordonnancement ayant $C_j > r_{j,k} + p_j$ et nous calculons la nouvelle date de fin de cette machine, en décalant à gauche tous les travaux i vérifiant $C_i > C_j$. Nous continuons ce procédé jusqu'à suppression de tous les travaux j ayant $C_j > r_{j,k} + p_j$.
2. Affecter les $\lfloor n/p \rfloor$ travaux sur la machine placée sur la localisation k, sur laquelle ces travaux ont les plus petites valeurs de dates de début au plus tôt.

Le premier critère permet de supprimer tous les travaux débutant après leur date de début au plus tôt depuis un cluster. Le but de ce critère est de calculer un ordonnancement sur chaque machine pour lequel la majorité de ses travaux ont des dates de fin minimales. Ce critère peut conduire à une mauvaise répartition de charges entre les machines. Pour cela, nous introduisons le deuxième critère qui construit des clusters de même taille.

Maintenant, nous combinons un critère d'affectation avec un critère de localisation, nous obtenons les trois heuristiques suivantes :

1. critère de localisation 1, critère d'affectation 1 (ICL1),
2. critère de localisation 1, critère d'affectation 2 (ICL2),
3. critère de localisation 2, critère d'affectation 2 (ICL3).

Les étapes de cette heuristique sont présentées dans l'algorithme 15.

Algorithme 15 Sélection itérative des clusters et des localisations (ICL)

Entrées: Une instance du problème *DPMM ScheLoc*

1: Sélectionner une localisation k pour une machine en utilisant un des critères de localisation 1 et 2
2: **Si** moins de p localisations ont été choisies **alors**
3: Affecter les travaux à la localisation k en utilisant le critère d'affectation 1 ou 2.
4: Aller à l'étape 1.
5: **finsi**
6: **Si** le nombre de machines placées est inférieur à p **alors**
7: Affecter tous les travaux restants à la localisation k.
8: **finsi**

Sorties: Localisation des p machines et un ordonnancement sur chaque machine

L'inconvénient de cette heuristique est que les travaux qui restent sont affectés à la dernière localisation choisie pour la dernière machine. Notamment, le nombre de travaux restants peut être assez grand avec le critère d'affectation (1). Ce problème peut être résolu en équilibrant les charges des machines pour des localisations et des affectations définies. Pour cela, dans la section 6.1, nous introduisons une procédure d'équilibrage de charge.

Selon les trois critères du choix des p clusters et des localisations, nous avons trois versions notées ICL1, ICL2 et ICL3 de cette heuristique. Ces notations seront utilisées dans la section 7.

Exemple 11 *Reprenons l'exemple 9. Nous souhaitons calculer une solution réalisable pour le problème DPMM ScheLoc en utilisant ICL2.*

La première localisation choisie pour la première machine est L2 car elle donne un C_{max} minimal égal à 25, en ordonnançant les travaux J1, J2,..., J7 (les travaux qui ne sont pas encore affectés). Ensuite, nous affectons à cette machine les $\lfloor n/p \rfloor = \lfloor 7/2 \rfloor = 3$ travaux qui ont des dates de début au plutôt minimales sur cette machine. Alors, à cette machine, nous affectons J3, J6 et J5. Pour les travaux restants, il faut tout d'abord choisir la localisation. Pour cela, nous calculons la date de fin des travaux sur les localisations L1 et L3. Les dates de fin d'ordonnancement sur L1 et L3 sont respectivement $C_{max}(L1) = 13.5$ et $C_{max}(L3) = 12.8$. Donc, la localisation L3 sera choisie pour placer la deuxième machine sur laquelle les travaux seront ordonnancés J4, J7, J1 et J2. La solution obtenue par ICL2 est présentée dans la figure 4.6 et la date de fin d'ordonnancement est $C_{max} = 15$.

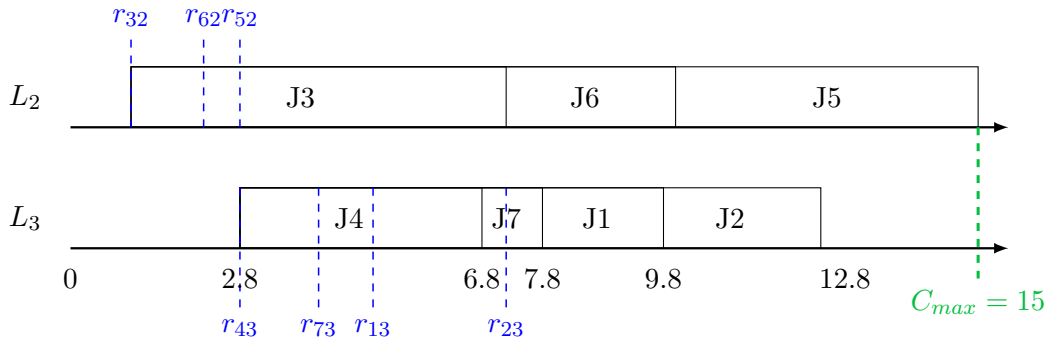


FIGURE 4.6 – Solution du problème *DPMM ScheLoc* calculée par ICL2

6 Recherche locale

6.1 Équilibrage de la charge

Comme il a été déjà mentionné dans la section précédente, les charges des machines peuvent être mal réparties. Ce problème peut apparaitre pour les heuristiques précédentes que nous avons proposées. Pour cette raison, nous procédons à une recherche locale pour équilibrer la charge sur les machines en réaffectant certains travaux.

La procédure commence par une solution initiale réalisable du problème *DPMM ScheLoc*. Ensuite, nous sélectionnons la machine qui donne la date de fin d'ordonnancement. Nous essayons de réaffecter à une autre machine le dernier travail j placé sur cette machine. Ceci est possible en déterminant de manière optimale l'emplacement du travail j sur chaque machine en utilisant la règle ERD. Si pour certaines machines k, cela conduit à une amélioration de la valeur de la date de fin alors, nous réitérons en mettant à jour la solution initiale. Les détails de cette procédure sont présentés dans l'Algorithme 16.

Algorithme 16 Équilibrage de la charge

Entrées: Instance du problème *DPMM ScheLoc* et une solution réalisable

1: Déterminer la machine k qui a la date de fin maximale.
2: Sélectionner le travail j qui est ordonnancé à la dernière position sur la machine k.
3: Pour l'ensemble $l = \{1, \ldots, p\}$ des localisations choisies pour les machines, ordonnancer j dans la position $l \neq k$ selon la règle ERD
4: **Si** il existe une machine $l \neq k$ qui minimise la date de fin d'ordonnancement de la solution, et si le travail j est ordonnancé sur la machine l **alors** :
5: Supprimer le travail j de la machine k et le placer sur la machine l selon la règle ERD.
6: Aller à l'étape 1
7: **Sinon**, fin de l'équilibrage de charge.
8: **finsi**

Sorties: Une solution réalisable

6.2 Réaffectation et localisation (LS)

L'inconvénient des heuristiques de cluster est que les décisions de localisation et d'affectation sont irréversibles. Mais à un degré moindre lorsque la procédure d'équilibrage de charge est utilisée : les possibilités de modifier les mauvaises décisions sont très limitées. Pour contourner ce problème, nous proposons, une recherche locale qui alterne le choix des localisations et des clusters. Le voisinage d'une solution est calculée en combinant les deux heuristiques localisation et cluster (LC, section 5.1) et cluster et localisation (CL, section5.2). Cette recherche s'arrête lorsque un optimum local ou bien un nombre maximal d'itérations est atteint.

Cette recherche locale commence par une solution réalisable du problème *DPMM ScheLoc* (obtenue par les heuristiques) où nous gardons les localisations choisies sur lesquelles les machines sont déjà placées. Ensuite, nous calculons comme dans l'heuristique localisation et cluster (LC), une nouvelle affectation des travaux selon la règle ERD. Si la nouvelle solution améliore la solutions courante alors nous mettons à jour cette solution.

Dans l'étape suivante, comme dans l'heuristique cluster et localisation, nous choisissons un nouveau ensemble de localisations. Si la solution obtenue est meilleure que la solution courante, nous recommençons la procédure avec un nouvel ensemble de localisations. Dans le cas contraire, la recherche s'arrête et une solution qui est au moins meilleur que la solution initiale est retournée. L'algorithme 17 synthétise les itérations de cette recherche locale.

Algorithme 17 réaffectation et localisation (LS)

Entrées: Une instance du problème *DPMM ScheLoc*, une solution réalisable $S = (X, A)$ avec X machines choisies et A affectations

1: Pour les localisations X nous calculons des affectations A' avec la règle ERD modifiée
2: **Si** A' améliore la date de fin d'ordonnancement, **alors** mettre $A = A'$
3: Pour les affectations A, calculer de nouvelles localisations X' avec la procédure de tri de la section 5.2
4: **Si** X' améliore la date de fin d'ordonnancement, **alors** mettre $X = X'$. **Sinon** fin de l'algorithme

Sorties: Solution réalisable

7 Expérimentations

7.1 Implémentation et Instances

Dans cette section, nous nous focalisons sur les tests et la comparaison des méthodes développées dans ce chapitre. Ces méthodes sont codées en C++ et python et exécutées sur un PC 16-cœurs Intel Xeon E5-2670 avec une fréquence 2.60 GHz, 96 GB RAM et Ubuntu 12.04. Le solveur mathématique utilisée est ILOG CPLEX 12.6 avec un nombre de cœurs égal à 5.

Les méthodes proposées dans ce chapitre sont testées sur quatre jeux d'instances. Les deux premiers Set 1 et Set 2 sont générés aléatoirement (espace discret) : les instances où il n'y a aucune structure sur les distances. L'ensemble Set 1 est constitué de 50 instances de petite taille avec $n \in \{1, \ldots, 30\}$, $m \in \{1, \ldots, 10\}$ et $p \in \{1, \ldots, 8\}$. L'ensemble Set 2 est constitué de 450 instances de grande taille avec $n \leq 300$, $m \leq 60$ et $p \leq 50$. Pour le troisième ensemble d'instances Set 3, 350 graphes sont générés aléatoirement dont 25 de petite taille. Ces graphes sont générées en posant $V = \mathcal{N} = \mathcal{M}$ avec $n \leq 300$, $m \leq 300$ et $p \leq 35$. Quant aux distances, elles sont calculées par un algorithme de plus court chemin. Le dernier ensemble Set 4 contient 600 instances avec $n \leq 300$, $m \leq 300$ et $p \leq 35$, dont 50 instances de petite taille. Les localisations des machines et des travaux sont générées aléatoirement dans un plan $\mathbb{R}^2$. Chaque localisation est identifiée par ses cordonnées sur ce plan. Les distances sont calculées en utilisant la distance euclidienne.

La formulation mathématique est testée sur les instances de petite taille des ensembles Set 1, Set 3 et Set 4. Tandis que les heuristiques de cluster sans l'équilibrage de charge, avec l'équilibrage de charge et la procédure de réaffectation et localisation (LS) sont testées sur toutes les instances appartenant aux ensembles Set 1, Set 2, Set 3 et Set 4. Nous ajoutons à ces heuristiques deux nouvelles versions de la procédure de réaffectation et localisation (LS). La première est aléatoire et prend comme solution initiale une solution générée aléatoirement. Cette version sera notée LS-Rand. La deuxième prend comme solution initiale la meilleure solution calculée par les différentes heuristiques de cluster avec l'équilibrage de charge. Les solutions de cette recherche locale seront notées LS-Best.

Pour chaque instance, chaque version aléatoire (LC1, LC4, CL1 et LS-Rand) est exécutée 10 fois et sa moyenne est stockée.

Dans le cas d'instance de petite taille, nous comparons les solutions approchées avec celles de la formulation mathématique résolue par le solveur CPLEX auquel une limite de temps de 15 minutes est imposée. Les résultats de ces tests sont présentés dans la section 7.2 suivante.

Pour les instances de grande taille, nous comparons les solutions avec les bornes inférieures (cf. section 4.2). Les résultats des comparaisons sont donnés dans la section 7.3.

7.2 Comparaison avec CPLEX

Dans un premier temps, nous présentons dans le Tableau 4.2 les résultats obtenus par CPLEX et les temps d'exécution. Chaque ligne de ce tableau correspond respectivement à l' ensemble des instances de petite taille de Set 1, Set 3 et Set 4. Les colonnes de ce tableau représentent respectivement le nombre d'instances de chaque ensemble (#Instances), la déviation moyenne entre la solution et la borne inférieure calculées par CPLEX (déviation(moy)), le nombre d'instances résolues à l'optimalité (#Optimal) et le temps moyen des instances résolues à l'optimalité (Opt_temps(s)).

Pour l'ensemble Set 1, qui contient 50 instances, nous distinguons deux cas. Dans le premier cas, nous avons 20 instances avec $n \leq 10$, CPLEX résout à l'optimalité toutes ces instances. Pour le deuxième cas, nous avons 30 instances avec $n > 10$, CPLEX résout uniquement 6 instances à l'optimalité. Dans les deux cas, le temps d'exécution moyen de

	#Instances	déviation(moy)	#Optimal	Opt_temps(s)
Set 1	50	28.6%	26	35.9s
Set 3	25	47.6%	2	271.85s
Set 4	50	51.3%	2	466.7s

TABLE 4.2 – Résultats obtenus par CPLEX et les temps d'exécution

CPLEX est de 35.9 secondes. La déviation moyenne sur toutes les instances est de 28% .

Pour les instances de l'ensemble Set 3 où $n \in \{10, \ldots, 20\}$, CPLEX résout uniquement 2 instances à l'optimalité avec un temps moyen de 271,85 secondes. L'augmentation du temps d'exécution de CPLEX pour résoudre ces instances par rapport aux instances de Set 2, est dû au nombre assez grand des localisations, même pour les instances avec $n = 10$ travaux. Ceci explique aussi le nombre petit d'instances résolues à l'optimalité.

L'ensemble Set 4 contient 50 instances avec $n \in \{10, \ldots, 20\}$. CPLEX calcule la solution optimale pour deux instances en un temps moyen de 466,7 secondes. Pour les mêmes raisons que dans la cas des instances de l'ensemble Set 3, nous constatons une augmentation du temps de calcul et un nombre de solutions optimales calculées très petit. La déviation moyenne est de 51.3%.

Dans le Tableau 4.3, nous présentons les comparaisons des heuristiques avec CPLEX. Pour cela, nous utilisons les pourcentage de la déviations moyennes de ces heuristiques par rapport aux solutions correspondantes obtenues par CPLEX. Dans ce tableau nous utilisons pour chaque instance, la meilleure solution obtenue par chaque type d'heuristique (LC : localisation et cluster, CL : cluster et localisation, ICL : sélection itérative des clusters et des localisations). Dans la section 7.3, nous donnons plus de détails sur les comparaisons entre les différentes versions de chaque heuristique. Pour la la procédure de réaffectation et localisation (LS), nous considérons la version LS-Rand et trois autres versions LS-LC, LS-CL et LS-ICL. Ces trois dernières prennent respectivement comme solution initiale, les solutions des heuristiques LC, CL, ICL. Rappelons que dans la section 5.2 l'heuristique CL n'est pas testée sur les instances où l'espace de localisation est aléatoire.

Puisque le temps d'exécution des heuristiques et les méthodes de recherche locale est négligeable (quelques secondes), nous nous focalisons sur la qualité des solutions calculées.

	LC	CL	ICL	LS-LC	LS-CL	LS-ICL	LS-Rand
Set 1	11.7	-	17.5	5.5	-	5.7	5.1
Set 3	13.4	21.1	17.1	8.2	9.4	7.4	3.4
Set 4	7.5	19.5	15.8	6.6	8.7	8.7	3.0

TABLE 4.3 – Comparaison entre les solutions des heuristiques et les meilleures solutions obtenues par CPLEX

Le Tableau 4.3 montre que toutes les heuristiques de clusters LC, CL et ICL calculent en moyenne des solutions éloignées pour un pourcentage entre 7% et 21% de la solution optimale. Quand la procédure de réaffectation et localisation est appliqué la déviation moyenne est de 3 à 9%. Les résultats du tableau montrent aussi, que la meilleure perfor-

mance est obtenue avec LS-Rand (3% à 6%). Ceci est justifié par le faite que la solution initiale de LS-Rand est obtenue après 10 exécutions.

Nous concluons que les solutions obtenues par les heuristiques ne sont pas loin des meilleures solutions obtenue par CPLEX (maximum 21%). Pour la procédure de réaffectation et localisation ce dernier pourcentage décroit à 10% au maximum. Donc ces heuristiques sont un bon compromis entre la qualité des solutions et le temps d'exécution.

7.3 Comparaison des heuristiques

Pour les instances de grande taille de Set 2, Set 3 et Set 4, nous évaluons la qualité des heuristiques en comparant ces solutions aux bornes inférieures (cf. section 4.2).

Le Tableau 4.4 présente la déviation des solutions obtenues par les heuristiques avec la procédure d'équilibrage de charge par rapport aux bornes inférieures. La dernière colonne Best-CH représente pour chaque instance, la déviation de la meilleure solution obtenue par les heuristiques (LC, CL, ICL) par rapport à la meilleure borne inférieure.

	LC1	LC2	LC3	LC4	CL1	CL2	CL3	ICL1	ICL2	ICL3	Best-CH
Set 2	17.9	18.2	18.4	17.8	-	-	-	22.9	24.3	25.8	12.3
Set 3	16.0	21.2	20.8	19.4	18.1	25.2	30.1	21.2	20.2	24.0	9.5
Set 4	19.6	24.5	22.8	22.4	19.2	23.4	34.5	17.6	18.4	21.1	11.5

TABLE 4.4 – Déviation moyenne (%) des solutions des heuristiques par rapport aux meilleurs bornes inférieures

À travers les résultats du tableau 4.4, nous pouvons constater que les solutions obtenues par les différentes versions d'heuristiques sont comparables : il n'existe pas une version qui calcule les meilleures solutions pour toutes les instances. Plus précisément, l'heuristique LC est la meilleure heuristique pour les instances de l'ensemble Set 2 et l'ensemble Set 3, tandis que ICL est la meilleure heuristique dans le cas des instances de l'ensemble Set 4. L'heuristique CL est légèrement plus mauvaise que LC et ICL dans le cas des instances de l'ensemble Set 3. Les versions aléatoires LC1, LC4 et CL1 sont meilleures que les heuristiques de leurs versions déterministes respectives.

Pour analyser l'amélioration que la procédure d'équilibrage de charge apporte sur les solutions des heuristiques, nous présentons dans le Tableau 4.5 la déviation des solutions obtenues sans l'équilibrage de charge par rapport aux meilleures bornes inférieures.

	LC1	LC2	LC3	LC4	CL1	CL2	CL3	ICL1	ICL2	ICL3
Set 1	18.6	19.8	19.7	19.0	-	-	-	735.6	175.0	173.7
Set 3	17.8	22.7	21.7	20.5	42.7	56.5	64.3	715.5	104.6	120.9
Set 4	22.9	27.3	24.5	25.2	40.5	50.8	70.4	783.0	116.8	128.0

TABLE 4.5 – Déviation moyenne (%) des heuristiques sans l'équilibrage de charge par rapport aux bornes inférieures.

Une comparaison entre les résultats des Tableaux 4.5 et 4.4 montre des améliorations importantes sur les solutions obtenues par les heuristiques CL et l'heuristique ICL1, en

utilisant la procédure de l'équilibrage de charge. La construction des clusters sans la prise en compte des durées opératoires est la raison principale de cette amélioration dans le cas des heuristiques de type CL. Dans le cas de heuristique ICL1, la procédure d'équilibrage de charge permet la répartition des charges entre les machines.

La méthode d'équilibrage de charge permet de réduire à 22% au maximum les déviations pour toutes les heuristiques (Tableau 4.4). Aussi cette recherche locale permet de réduire significativement les déviations de l'heuristique ICL. Nous constatons aussi que ces heuristiques sont capables de faire de bons choix de localisation, puisque les modifications de l'équilibrage de charge portent uniquement sur les ordonnancements sur les machines. Notamment pour LC où les améliorations des solutions sont comprises entre 1 % et 3%.

Notons que le temps d'exécution de la procédure d'équilibrage de charge est de quelques secondes : elle est donc appliquée après chaque heuristique.

Enfin, pour analyser l'amélioration apportée par la procédure de réaffectation et localisation sur les solutions initiales, nous présentons dans le Tableau 4.6 la déviation moyenne de la procédure de réaffectation et localisation en prenant comme solutions de départ les solutions obtenues par les heuristiques. Pour les heuristiques non déterministes, la solution de départ est la moyenne sur les solutions obtenues après 10 exécutions. La dernière colonne LS-Best représente pour chaque instance la déviation de la meilleure solution obtenue par les différentes versions de la procédure de réaffectation et localisation (LS-LC, LS-CL et LS-ICL) par rapport à la meilleure borne inférieure.

	LC1	LC2	LC3	LC4	CL1	CL2	CL3	ICL1	ICL2	ICL3	LS- Rand	LS- Best
Set 1	14.1	14.5	14.6	14.2	-	-	-	14.3	15.3	15.5	17.7	10.7
Set 3	13.0	14.8	15.7	14.3	13.8	16.9	17.0	15.1	13.9	16.1	9.7	8.2
Set 4	13.7	15.4	16.2	12.8	14.1	15.3	16.3	13.5	13.8	14.8	10.0	8.9

TABLE 4.6 – Déviation moyenne(%) des solutions de la procédure de réaffectation et localisation par rapport aux meilleurs bornes inférieures

Une comparaison des résultats du Tableau 4.6 avec les résultats du Tableau 4.4 montre que la procédure de réaffectation et localisation améliore significativement les solutions obtenues par les heuristiques. Cette amélioration est comprise entre 4% et 18%. Nous constatons aussi que l'amélioration est d'autant plus importante que la déviation de la solution initiale est très petite (cf. tableau 4.4). Pour les instances des ensembles Set 3 et Set 4, la procédure de réaffectation et localisation pour les heuristique non-déterministes donne de meilleures solutions. Cela est dû aux 10 exécutions de ces heuristiques, permettant de couvrir une plus grande partie de l'espace de recherche.

Les résultats des heuristiques de clusters et des méthodes de recherche locale montrent des bon compromis entre la qualité des solutions (déviation moyenne de 3 % maximum, par rapport à la solution obtenue par CPLEX) et les temps d'exécution.

8 Conclusion du chapitre

Dans ce chapitre, nous avons abordé le problème d'ordonnancement et de localisation (*DPMM ScheLoc*). Ce problème a plusieurs applications dans la vie réelle (cf. section 2). Nous avons proposé une formulation mathématique pour résoudre des instances de petite taille et des heuristiques inspirées du problème de clusterisation. Ces heuristiques résolvent les sous-problèmes : localisation des machines, affectation des travaux et ordonnancement de ces travaux sur ces machines.

Afin d'améliorer les solutions obtenues par ces heuristiques, nous avons développé une procédure d'équilibrage de charge et une procédure de réaffectation et localisation. Les temps d'exécution de toutes les heuristiques de cluster sont négligeables. Pour cela, une approche possible est d'exécuter pour chaque instance, toutes ces heuristiques en gardant la meilleure solution trouvée. Cette solution est loin de la solution optimale de 7% à 22%. Néanmoins, si la procédure de réaffectation et localisation utilise cette solution comme solution initiale, l'écart devient de l'ordre de 2%.

Chapitre 5

Intégration au Projet DSS_Evac_Logistic

Ce dernier chapitre est consacré à la présentation du logiciel d'aide à la décision nommé *Visual Flow*. Il s'agit d'un outil qui aidera à préparer des évacuations à grande échelle, de planifier et de mener des actions en cas d'une catastrophe.

Dans un premier temps, nous présentons dans la section 1 le contexte et le cycle de développement de ce logiciel. Ensuite, la section 2 explique l'architecture du *Visual Flow*. Enfin, dans la section 3, nous présentons brièvement les méthodes intégrées dans *Visual Flow* et développées dans cette thèse.

1 Contexte et cycle de développoment du *Visual Flow*

Le but de notre projet est d'établir un logiciel ergonomique qui soit un support d'aide à la décision (DSS, Decision Support System) et qui permet aux décideurs de choisir dans une base de données des plans d'évacuation. Le logiciel développé durant le projet DSS_Evac_Logistic est *Visual Flow*. Pour le développement de *Visual Flow*, nous avons opté pour une approche incrémentale de gestion de projet selon un modèle itératif. Cette approche est une succession des itérations suivantes : définition des scénarios de catastrophe, identification des objectifs à atteindre, modélisation et développement des méthodes de résolution (Figure 5.1). Ce type de cycle de développement présente, d'une part, l'avantage d'impliquer l'ensemble des acteurs sur tout le projet et, d'autre part, d'aboutir le plus rapidement possible à la livraison des versions intermédiaires permettant d'être évaluées par les utilisateurs finaux.

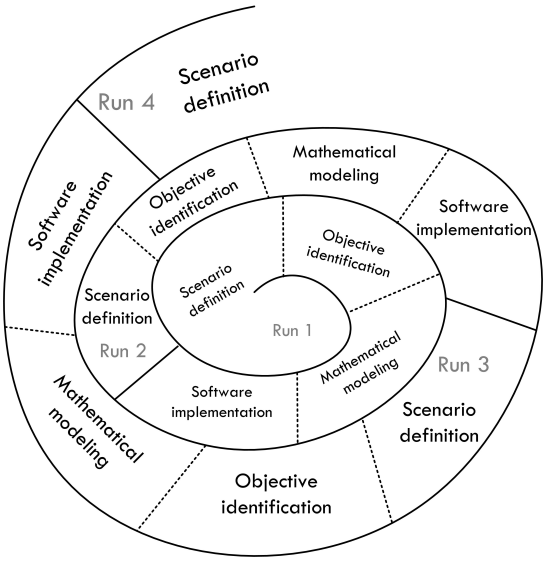


FIGURE 5.1 – Cycle de développement du Logiciel *Visual Flow* ([T'kindt et al., 2011])

Comme nous l'avons vu dans le chapitre 1, plusieurs partenaires ont participé au développement du logiciel *Visual Flow*. Dans ce qui suit, nous expliquons, le rôle de chaque partenaire dans le cycle de développement.

Tâche 1 : Definition of Scenarios and Outcomes

BRGM et CITERES (coté français) et les autorités responsables de la sécurité de Kaiserslautern (coté Allemand) ont définies respectivement les scénarios de catastrophe plausibles pour la ville de Nice et pour la ville de Kaiserslautersn. De plus, ils ont fournis une estimation des dommages causés par ces catastrophes : le nombre de personnes à évacuer, les endroits potentiels des centres de secours, et les localisations des points de rassemblement. Le BRGM a aussi fourni les résultats d'évaluation de l'impact des catastrophes sur le réseau routier et les bâtis.

Ces données ont servis comme support menant à la conception du *Visual Flow*.

Tâche 2 : Evac-Location

Les algorithmes développés dans ce “working package" ou tâche, permettent de choisir les centres de secours réellement opérationnels. Ces travaux font partie de la thèse de Philipp Hessler (Optimization Research Group, Université de Kaiserslautern, Allemagne). Les sorties de ces algorithmes sont utilisées comme des entrées des tâches WP3, WP4 et WP5.

Tâche 3 : Evac-EvalTraffic

Cette tâche a été réalisée par “Optimization Research Group" (Université de Kaiserslautern, Allemagne) et le BRGM. Les sorties de cette tâche sont les différents paramètres du réseau routier qui sont dépendants des scénarios de la catastrophe : les capacités maximales de chaque route, le temps de parcours de chaque route, la suppression de certaines routes, la réorientation des voies à double sens en voies à sens unique, etc.

Tâche 5 : Evac-Solve

Plusieurs algorithmes sont proposés pour le calcul des chemins des flots individuels (piétons et voitures) depuis leurs habitations vers les points de rassemblement. Ces travaux font partie de la thèse d'Ismaila Abderhmanne Ndiaye (Équipe Ordonnancement et Conduite, Laboratoire LI, Tours).

Tâche 4 : Evac-Scheduling

Cette tâche fait l'objet de cette thèse et utilise les sorties des étapes précédentes. Notamment, la répartition des piétons et les voitures sur les points de rassemblement. Le but est d'ordonnancer les opérations d'évacuation des voitures et des piétons utilisant les bus pour atteindre les centres de secours. Plusieurs algorithmes ont été développés dans cette tâche, dont ceux qui permettent de résoudre simultanément les problèmes d'ordonnancement et de localisation. Ces algorithmes peuvent donc même être utilisé pour WP2.

Tâche 6 : Evac-MSDN

Comme les problèmes considérés dans ce projet sont souvent multicritère, les algorithmes proposés dans cette tâche permettent de construire pour ces problèmes un échantillon de l'ensemble des optima de Pareto. Cet échantillon doit être suffisamment important pour être représentatif, et suffisamment petit pour permettre au décideur de faire un choix en comparant les solutions. Ces travaux font partie de la thèse de Tobias Kuhn (Optimization Research Group, Université de Kaiserslautern, Allemagne).

Tâche 7 : Evac-Simulate

La société CERVVAL (France) a développé un outil de simulation pour simuler et évaluer les plans d'évacuation calculés.

Tâche 8 : Evac-Software

La société INFORMS (Allemagne) a développé *Visual Flow* qui permet aux utilisateurs finaux de gérer des situations de crise. Ce logiciel intègre les algorithmes développés dans les tâches précédentes et propose une interface unifiée pour la gestion d'évacuation. Notons aussi que cette tâche a été réalisée en collaboration avec Flavien Audin, ingénieur d'étude à l'équipe d'ordonnancement et conduite, Laboratoire LI, Tours.

2 Architecture

L'architecture du logiciel *Visual Flow* est présentée dans la Figure 5.2. Il est principalement composé de trois modules.

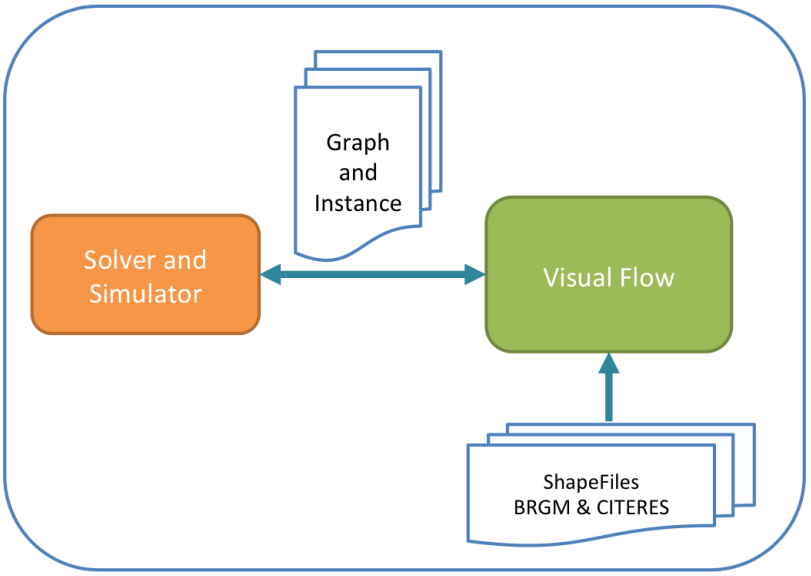


FIGURE 5.2 – Architecture générale de *Visual Flow*

Le module Visual Flow. Ce module est constitué de plusieurs composants :
- Le composant de visualisation d'une solution calculée.
- Le composant faisant les statistiques sur les solutions.
- Le composant Spider-Web comparant les solutions calculées.

Le module Solver. Ce module est constitué des exécutables des méthodes développées dans ce projet, à savoir :
- Les méthodes déterminant les localisations des centres de secours.
- Les méthodes de calcul des chemins des piétons et voitures vers les points de rassemblement.
- Les méthodes de calcul des plans d'évacuation par bus.

– Les méthodes de calcul des plans d'évacuation par bus et par voitures intégrant la décision de localisation des centres de secours.

Le module Simulator. Ce module permet de simuler le déroulement d'un plan d'évacuation calculé par les méthodes du module Solver. Ce module permet aussi de valider ce plan en prenant en compte l'occupation initiale du réseau routier ainsi que le comportement des foules en état de panique, perturbant le déroulement de l'évacuation organisée.

L'intéraction entres ces modules se fait à l'aide de fichier (.XML). Ces fichiers contiennent pour chaque scénario de catastrophe, les instances de la ville de Nice et de Kaiserslautern. Aussi, pour chaque instance, le module Visual Flow charge son fichier "Shape file" correspondant aux dommages sur les routes du réseau.

3 Implémentation

L'utilisateur final doit suivre plusieurs étapes pour pouvoir calculer un ensemble de plans d'évacuation (Figure 5.3), commençant par l'étape de chargement d'un fichier d'instance (Step 1), et le chargement de l'impact sur le réseau routier associé à ce fichier (Step 2). Ensuite, dans le Step 3, il obtient les centres de secours qui seront réellement utilisés. Par la suite, il peut lancer une exécution séquentielle des algorithmes développés dans WP4 et WP5 (Step 4). Une fois un ensemble de plans calculé, l'utilisateur peut visualiser les déplacements des bus, des voitures et le diagramme de Gantt d'un plan d'évacuation. Aussi, il a le choix de simuler ce plan (Step5). Le logiciel fourni aussi la possibilité aux utilisateurs finaux de comparer toutes les solutions calculées à l'aide d'un Spider-Web pour un fichier d'instance. Comme mentionné précédemment, plusieurs méthodes développées dans cette thèse sont intégrés à *Visual Flow*. Plus précisément les méthodes suivantes :

Les méthodes calculant des plans d'évacuation pour les bus :

– Les heuristiques de liste et la matheuristique, présentées dans le chapitre 2, servent à résoudre le problème d'évacuation par bus monocritère. La figure (5.4) présente une capture d'écran d'une solution calculée par les heuristiques de liste.
– Les heuristiques de liste et la matheuristique, présentées dans le chapitre 2, calculent un ensemble de solutions de front Pareto pour le problème de bus bicritère.

Les méthodes de calcul des plans d'évacuation pour les bus et les voitures :

– Ces méthodes sont présentées dans le chapitre 3 et résolvent le problème multicritère d'évacuation par bus et par voiture. Ce dernier problème intègre aussi le problème de localisation des centres de secours. Les deux méthodes intégrées à *Visual Flow* sont l'algorithme génétique et l'heuristique par décomposition. La figure (5.5) présente une capture d'écran d'une solution calculée par l'algorithme génétique. Le Spider-Web permet de comparer les plans calculés selon les critères suivants : la date de fin d'évacuation, la durée moyenne d'attente sur chaque point de rassemblement, la durée moyenne d'évacuation, etc.

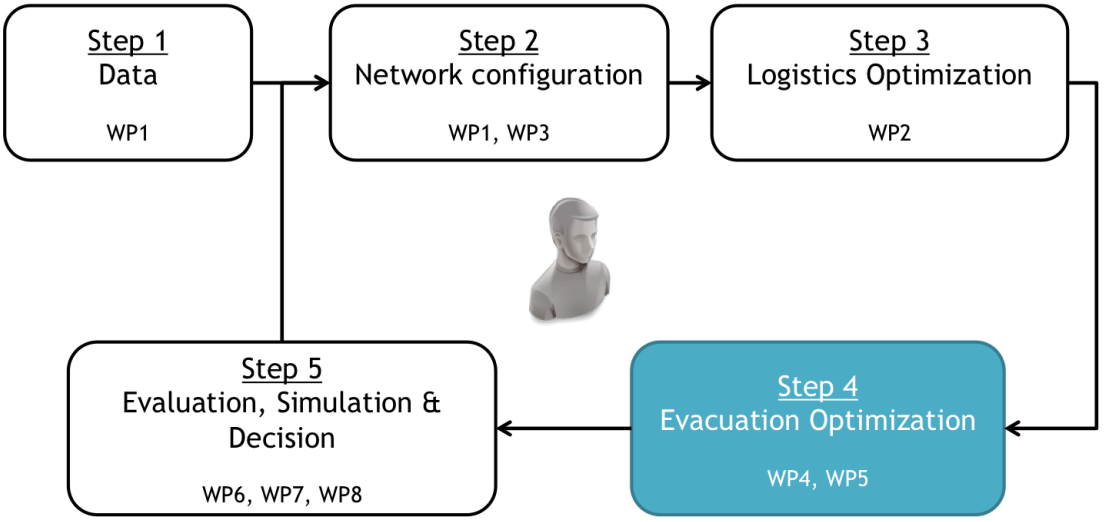


FIGURE 5.3 – Étapes pour le calcul des plans d'évacuation *Visual Flow*

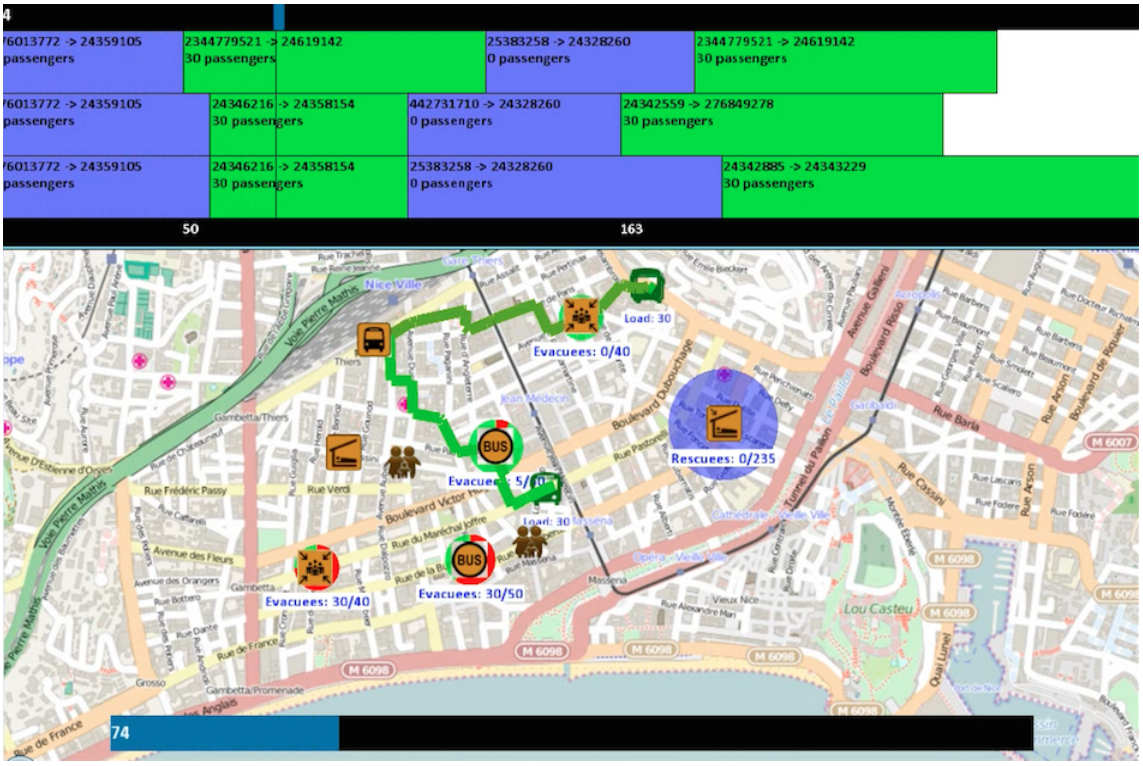


FIGURE 5.4 – Visualisation d'une évacuation par bus calculée par les heuristiques de liste

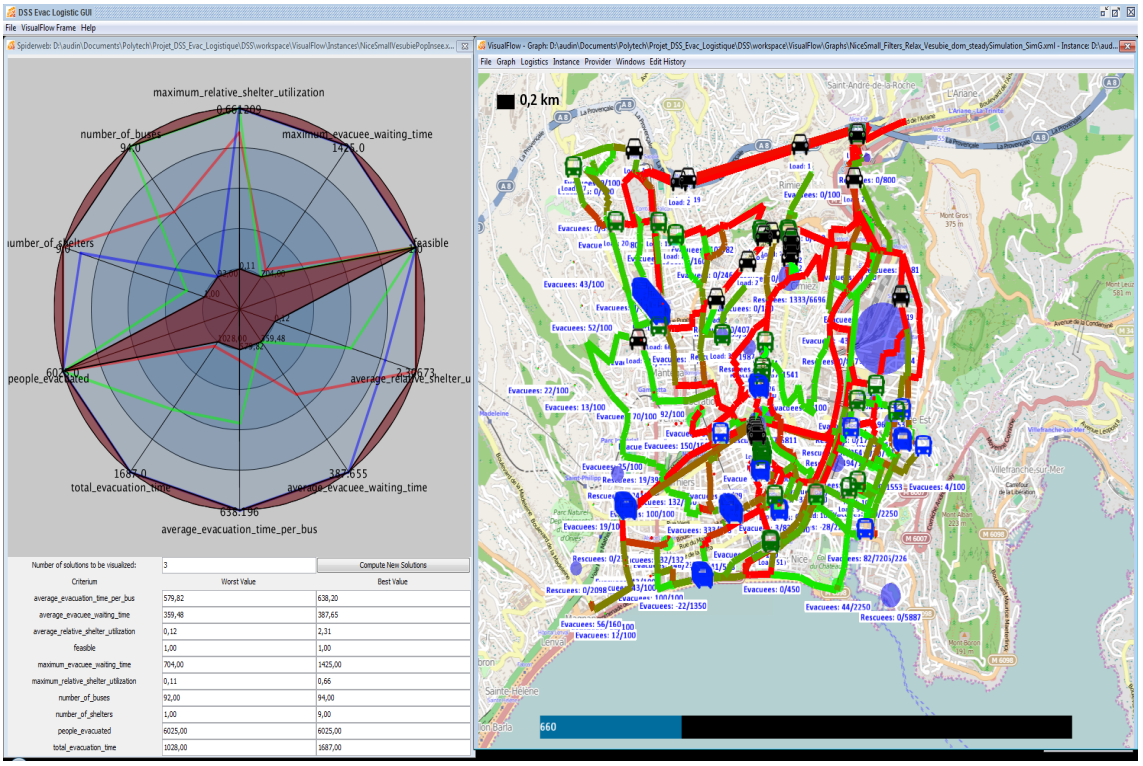


FIGURE 5.5 – Visualisation d'une évacuation par bus et voiture calculée par l'algorithme génétique

4 Conclusion du chapitre

Dans ce chapitre, nous avons présenté le logiciel d'aide à la décision *Visual Flow* permettant aux décideurs finaux de calculer des plans d'évacuation en se basant sur des scénarios pré-calculés et sur leurs expériences sur le terrain. Nous avons présenté le cycle de développement de *Visual Flow* et les différents partenaires qui ont participé au développement de cet outil. Ensuite, nous avons présenté l'architecture de ce logiciel.

Dans la dernière section de ce chapitre, nous avons présenté le fonctionnement de *Visual Flow* et une description succincte des méthodes développées dans cette thèse et intégrées au logiciel.

Conclusion

Les travaux réalisés durant cette thèse portent sur les problèmes d'évacuation à grande échelle après une catastrophe majeure. Cette thèse s'inscrit dans le cadre du projet franco-allemand DSS_Evac_Logistic. Nous prenons comme cas d'études les villes de Nice (France) et de Kaiserslautern (Allemagne). L'objectif est de calculer des plans d'évacuation macroscopiques pour assurer l'évacuation des personnes depuis la zone sinistrée vers des endroits plus sûrs. Pour ce faire, nous avons considéré les critères suivants : la date de fin d'évacuation, la somme des risques encourus durant l'évacuation et le nombre de centres de secours à ouvrir.

Dans le **chapitre 1**, nous avons d'abord rappelé les problèmes d'évacuation existants dans la littérature ainsi que les approches et les méthodes utilisées pour résoudre ces problèmes. L'objectif de cet état de l'art est de cerner les problèmes d'évacuation abordés dans la littérature pour positionner les problèmes d'évacuation abordés dans cette thèse. Ce chapitre se termine par une présentation succincte du projet DSS_Evac_Logistic et des problèmes d'évacuation considérés dans cette thèse.

Dans le **chapitre 2**, nous avons considéré trois versions du problème d'évacuation par bus avec approximation du réseau routier. Dans la première version qui est monocritère (MBEP), nous cherchons à calculer un plan d'évacuation en minimisant la date de fin d'évacuation. Tout d'abord, nous avons proposé deux modélisations mathématiques pour résoudre à l'optimalité des instances de petite taille. En plus, par adjonction de quelques inégalités valides à l'une de ces modélisations, nous restreignons l'espace des solutions. Puis, nous avons développé un algorithme de prétraitement pour réduire le nombre de variables. Ce travail a fait l'objet d'une communication à une conférence nationale ([Deghdak et al., 2013b]) et à deux conférences internationales ([Deghdak et al., 2013a, Deghdak et al., 2014c]). Pour résoudre des instances de grande taille, nous avons proposé plusieurs versions d'heuristiques de liste et une matheuristique pour améliorer les solutions obtenues par ces heuristiques. Ces méthodes approchées ont fait l'objet d'une communication nationale ([Deghdak et al., 2014b]) et d'une communication internationale ([Deghdak et al., 2014a]). Ces travaux ont été publié dans la revue Journal of Scheduling ([Deghdak et al., 2015b]).

Dans la deuxième version, nous nous sommes intéressés à l'étude de la version robuste du problème d'évacuation par bus (RBEP). Le but est double : (1) trouver une solution qui nécessite des modifications modérées pour qu'elle soit réalisable pour n'importe quel scénario de donnés , et (2) minimiser la détérioration de la date de fin dans le pire des cas. Pour résoudre ce problème, nous avons proposé des méthodes exactes et des méthodes

approchées basées sur la programmation mathématique. Ce travail a été réalisé en collaboration avec Marc Goerigk (Optimization Research Group, Université de Kaiserslautern, Allemagne), membre du projet DSS_Evac_Logistic et a fait l'objet d'une publication dans une revue internationale ([Goerigk et al., 2015]).

Enfin, dans la troisième version, nous avons étudié le problème bicritère d'évacuation par bus où les deux critères à minimiser sont la date de fin d'évacuation et la somme des risques encourus durant l'opération d'évacuation. Afin de résoudre ce problème à l'optimalité, nous avons proposé une procédure par séparation et évaluation (B&B) et une modélisation mathématique. Puisque ces deux méthodes sont incapables de résoudre des instances de grande taille, nous avons proposé deux heuristiques de liste et une matheuristique. La solution initiale de la matheuristique est la meilleure solution calculée par les heuristiques de liste. Ce travail a été présenté dans une conférence internationale ([Deghdak et al., 2015a]).

Dans le **chapitre 3**, nous avons présenté le problème d'évacuation par bus et voitures. Le but est de déterminer la date de début d'évacuation, les centres de secours de rattachement et les routes menant à ces centres. Les critères à minimiser sont le nombre de centres à ouvrir, la date de fin d'évacuation et la somme des risques. Nous avons proposé une modélisation mathématique capable de résoudre des instances de petite taille. Pour des instances de grande taille, nous avons développé un algorithme génétique et une méthode par décomposition. Ces travaux ont donné lieu à une communication dans une conférence national ([Deghdak and T'kindt, 2015b]), une communication dans une conférence internationale ([Deghdak and T'kindt, 2015a]) et à une publication dans une revue internationale ([Goerigk et al., 2014a]).

Le **chapitre 4** est consacré à l'étude du problème d'ordonnancement et de localisation *DPMM ScheLoc*. Le but est de chercher à placer un ensemble de machines sur des localisations et ordonnancer des travaux sur ces machines afin de minimiser la date de fin d'ordonnancement. Nous avons suggéré une modélisation mathématique permettant de résoudre des instances de petite taille. En revanche, pour les instances de grande taille, nous avons développé plusieurs heuristiques de cluster, une méthode de post-optimisation et une recherche locale. Ces travaux ont été réalisé en collaboration avec Corrina Hessler (Optimization Research Group, Université de Kaiserslautern, Allemagne), membre du projet DSS_Evac_Logistique et ont fait l'objet de deux communications dans des conférences internationales ([Hessler and Deghdak, 2015a, Hessler and Deghdak, 2015b]) et seront soumis prochainement à une revue internationale ([Hessler and Deghdak, 2015c]).

Dans le **chapitre 5**, nous avons présenté *Visual Flow* qui a été développé durant ce projet. Il s'agit d'un logiciel d'aide à la décision permettant aux utilisateurs finaux de choisir un plan d'évacuation parmi les plans pré-calculés. La majorité des méthodes développées dans cette thèse, à savoir les méthodes calculant des plan d'évacuation pour les bus et les méthodes calculant des plans d'évacuation pour les bus et pour les voitures ont été intégrées au module Solver de *Visual Flow*.

Plusieurs perspectives de recherche sont envisagées à court et moyen terme :

– **Proposition de nouvelles inégalités valides.** Dans le chapitre 1, nous avons proposé des inégalités valides basées essentiellement sur les contraintes de précédence entre les opérations d'évacuation et sur l'élimination des symétries entre ces opéra-

tions. Cependant, il existe d'autres inégalités valides, par exemple, celles proposés dans [Sousa and Wolsey, 1992] et adaptées par [Berghman and Spieksma, 2015] pour un problème proche du problème d'évacuation par bus. Il serait intéressant d'adapter ces inégalités valides à notre problème et voir leurs influences sur notre modélisation.

- **Établissement de nouvelles bornes inférieures.** Dans la section 6.2.3 du chapitre 1, nous avons constaté que les bornes inférieures proposées ne sont pas très efficaces, notamment, la borne utilisée au premier niveau de l'arbre de recherche, calculée simplement en utilisant la relaxation linéaire du modèle IP. Nous allons chercher d'autres bornes rapides et efficaces.
- **Voisinage des matheuristiques.** Dans les sections 4.4.2 et 6.2.2 du chapitre 2, la solution courante de la matheuristque est mis à jour par une solution meilleure et voisine. Nous pouvons aussi élargir le voisinage en acceptant avec une probabilité une solution mauvaise et voisine de la solution courante.
- **Alternance entre l'évacuation par bus et l'évacuation par voitures.** Dans l'heuristique par décomposition (cf. section 5), nous avons supposé que l'évacuation par voitures se fait en premier lieu et sera suivie par l'évacuation par bus. Une autre approche consiste à alterner entre l'évacuation par bus et l'évacuation par voitures.
- **Évacuation par niveau.** Une autre stratégie d'évacuation par zones peut être utilisée en décomposant la zone sinistrée en plusieurs zones. La décomposition peut se faire par exemple selon les dommages causés par la catastrophe. Cette stratégie peut être appliquée pour la résolution des problèmes d'évacuation par bus MBEP, RBEP et BBEP, mais aussi pour le problème d'évacuation général GEP.

D'autres perspectives sont envisagées à long terme :

- **Considération d'autres critères d'optimisation.** Tout au long de cette thèse, nous avons supposé que les évacués suivent les consignes d'évacuation qui ont été fournies. Cette hypothèse peut être renforcée en introduisant un autre critère supplémentaire lors du calcul d'un plan d'évacuation. Par exemple, la minimisation de la durée moyenne d'attente des évacués sur chaque point de rassemblement. Cela permet d'éviter de longues attentes provoquant la dispersion des évacués vers d'autres points de rassemblement et entrainant une entrave de l'application du plan d'évacuation prédéfini.
- **Évacuation multimodale.** Dans cette thèse, nous avons considéré uniquement le problème d'évacuation par bus et voitures. Il serait intéressant de considérer à long terme le problème d'évacuation multimodale, en utilisant les bus, les voitures et les ferries. Notamment, pour les villes côtières telle que la ville de Nice où une évacuation par bateaux est possible. À notre connaissance, ce problème est inédit.

Bibliographie

[KLD Associates, 1984] KLD Associates (1984). Formulations of the dynev and i-dynev traffic simulation models used in ESF. Report prepared for Federal Emergency Management Agency, http ://www.kldcompanies.com.

[Achterberg, 2009] Achterberg, T. (2009). SCIP : Solving constraint integer programs. *Mathematical Programming Computation*, 1(1) :1–41.

[Aerde and Yagar, 1988] Aerde, M. and Yagar, S. (1988). Dynamic integrated freeway/traffic signal networks : A routing-based modelling approach. *Transportation Research Part A : General*, 22(6) :445 – 453.

[Afshar and Haghani, 2008] Afshar, A. and Haghani, A. (2008). Heuristic framework for optimizing hurricane evacuation operations. *Transportation Research Record*, 2089 :9–17.

[Aissi et al., 2009] Aissi, H., Bazgan, C., and Vanderpooten, D. (2009). Min–max and min–max regret versions of combinatorial optimization problems : A survey. *European Journal of Operational Research*, 197(2) :427 – 438.

[Barrett et al., 2000] Barrett, B., Ran, B., and Pillai, R. (2000). *Developing a dynamic traffic management modeling framework for hurricane evacuation.*

[Ben-Tal et al., 2009] Ben-Tal, A., Ghaoui, L. E., and Nemirovski, A. (2009). *Robust Optimization.* Princeton University Press, Princeton and Oxford.

[Ben-Tal and Nemirovski, 2000] Ben-Tal, A. and Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88 :411–424.

[Berghman and Spieksma, 2015] Berghman, L. and Spieksma, F. C. (2015). Valid inequalities for a time-indexed formulation. *Operations Research Letters*, 43(3) :268 – 272.

[Bish, 2011] Bish, D. R. (2011). Planning for a bus-based evacuation. *OR Spectrum*, 33(3) :629–654.

[Bish et al.,] Bish, D. R., Sherali, H. D., and Hobeika, A. G. Optimal evacuation planning using staging and routing. *Journal of the Operational Research Society*, 65(1) :124–140.

[Bretschneider, 2012] Bretschneider, S. (2012). *Mathematical Models for Evacuation Planning in Urban Areas.* Springer- Heidelberg New York Dordrecht London.

[Bretschneider and Kimms, 2012] Bretschneider, S. and Kimms, A. (2012). Pattern-based evacuation planning for urban areas. *European Journal of Operational Research*, 216(1) :57 – 69.

[Chamlet et al., 1982] Chamlet, L., Francis, R., and Saunders, P. (1982). Network models for building evacuation. *Fire Technology*, 18(1) :90–113.

[Chen et al., 2006] Chen, X., Meaker, J., and Zhan, F. (2006). Agent-based modeling and analysis of hurricane evacuation procedures for the florida keys. *Natural Hazards*, 38(3) :321–338.

[Chen and Chin, 1990] Chen, Y. L. and Chin, Y. H. (1990). The quickest path problem. *Computers & Operations Research*, 17(2) :153–161.

[Cheng and Sun, 2005] Cheng, M. and Sun, S. (2005). Two scheduling problems in group technology with deteriorating jobs. *Applied Mathematics-A Journal of Chinese Universities*, 20 :225–234.

[Cheng et al., 2007] Cheng, T. C. E., Kang, L. Y., and Ng, C. T. (2007). Due-date assignment and parallel-machine scheduling with deteriorating jobs. *Journal of The Operational Research Society*, 58 :1103–1108.

[Chiu, 2004] Chiu, Y. C. (2004). Traffic scheduling simulation and assignment for area-wide evacuation. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 537–542. IIEEE Intelligent Transportation Systems Conference, Washington D.C.

[Chiu et al., 2007] Chiu, Y.-C., Zheng, H., Villalobos, J., and Gautam, B. (2007). Modeling no-notice mass evacuation using a dynamic traffic flow optimization model. *IIE Transactions*, 39(1) :83–94.

[Choi et al., 1988] Choi, W., Hamacher, H., and Tufekci, S. (1988). Modeling of building evacuation problems by network flows with side constraints. *European Journal of Operational Research*, 35(1) :98 – 110.

[Church and Sexton, 2002] Church, R. and Sexton, R. (2002). *Modeling Small Area Evacuation : Can Existing Transportation Infrastructure Impede Public Safety ?* University of California Santa Barbara, National Center for Geographic Information and Analysis, Vehicle Intelligence and Transportation Analysis Laboratory.

[(COE) and (SWFRPC), 1979] (COE), C. o. E. and (SWFRPC), S. F. R. P. C. (1979). Lee county florida flood emergency evacuation plan.

[Coutinho-Rodrigues et al., 2012] Coutinho-Rodrigues, J., Tralhao, L., and Alcada-Almeida, L. (2012). Solving a location-routing problem with a multiobjective approach : the design of urban evacuation plans. *Journal of Transport Geography*, 22(0) :206 – 218.

[Daganzo, 1994] Daganzo, C. F. (1994). The cell transmission model : a dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B*, 28(4) :269—-287.

[Daganzo, 1995] Daganzo, C. F. (1995). The cell transmission model, part II : network traffic. *Transportation Research Part B*, 29(2) :79—-93.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) :182–197.

[Deghdak and T'kindt, 2015a] Deghdak, K. and T'kindt, V. (2015a). A decomposition heuristic for a bicriteria evacuation scheduling problem. In *Proceedings of Multidisci-*

plinary International Scheduling Conference : Theory and Applications, MISTA, vol.7, pages 704–707, Pragues, czech republic.

[Deghdak and T'kindt, 2015b] Deghdak, K. and T'kindt, V. (2015b). Une heuristique par décomposition pour la résolution d'un problème d'ordonnancement bicritère d'opérations d'évacuation. In *Proceedings de Recherche Opérationnelle et d'Aide à la Décision, vol.15*, Marseille, France.

[Deghdak et al., 2015a] Deghdak, K., T'Kindt, V., and Bouquard, J. (2015a). Enumeration of pareto optima for a bicriteria evacuation scheduling problem. In *ICORES 2015 - Proceedings of the 4rd International Conference on Operations Research and Enterprise Systems, ICORES, Lisbon, Portugal, Januray 10-12, 2015.*, pages 162–171.

[Deghdak et al., 2013a] Deghdak, K., T'kindt, V., and Bouquard, J.-L. (2013a). Evacuation scheduling of a town after a natural disaster : time-indexed formulations. In *Proceedings of Multidisciplinary International Scheduling Conference : Theory and Applications, MISTA, vol.6*, pages 680–683, Gent, Belgium.

[Deghdak et al., 2013b] Deghdak, K., T'kindt, V., and Bouquard, J.-L. (2013b). Ordonnancement de l'évacuation d'une ville lors d'une catastrophe naturelle. In *Proceedings de Recherche Opérationnelle et d'Aide à la Décision, ROADEF, vol.13*, Troyes, France.

[Deghdak et al., 2014a] Deghdak, K., T'kindt, V., and Bouquard, J.-L. (2014a). Heuristics for scheduling evacuation operations in case of natural disaster. In *Proceedings of Operational Research and Enterprise Systems, ICORES , vol.3*, pages 294 – 300, Angers, France.

[Deghdak et al., 2014b] Deghdak, K., T'kindt, V., and Bouquard, J.-L. (2014b). Heuristiques pour l'ordonnancement des opérations d'évacuation après une catastrophe naturelle. In *Proceedings de Recherche Opérationnelle et d'Aide à la Décision, ROADEF, vol.14*, Bordeaux, France.

[Deghdak et al., 2014c] Deghdak, K., T'kindt, V., and Bouquard, J.-L. (2014c). Solving scheduling evacuation operations by mathematical programming and preprocessing. In *Proceedings of Project Management and Scheduling, PMS, vol.14*, pages 72–75, Munich, Germany.

[Deghdak et al., 2014d] Deghdak, K., T'kindt, V., and Bouquard, J.-L. (2014d). État de l'art sur les problèmes d'ordonnancement avec des durées opératoires contrôlable.

[Deghdak et al., 2015b] Deghdak, K., T'kindt, V., and Bouquard, J.-L. (2015b). Scheduling evacuation operations. *Journal of Scheduling*, pages 1–12.

[Della Croce et al., 2011] Della Croce, F., Grosso, A., and Salassa, F. (2011). A matheuristic approach for the total completion time two-machines permutation flow shop problem. In Merz, P. and Hao, J.-K., editors, *Evolutionary Computation in Combinatorial Optimization*, volume 6622 of *Lecture Notes in Computer Science*, pages 38–47. Springer Berlin Heidelberg.

[Dial, 1969] Dial, R. B. (1969). Algorithm 360 : shortest-path forest with topological ordering [h]. *Communication the ACM*, 12(11) :632–633.

[Dorigo and Stützle, 2004] Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. Bradford Company, Scituate, MA, USA.

[Driebeek, 1966] Driebeek, N. J. (1966). An algorithm for the solution of mixed integer programming problems. *Management Science*, 12(7) :576–587.

[Dunn, 1992] Dunn, C. (1992). *Optimal Routes in GIS and Emergency Planning Applications*, volume 24. National Emergency Training Center.

[Elloumi et al., 1994] Elloumi, N., Haj-Salem, H., and Papageorgiou, M. (1994). Metacor : a macroscopic modelling tool for urban corridor. In *Urban Traffic Networks*. Triennal symposium on transportation analysis, Capri, Italy.

[Elvikis et al., 2008] Elvikis, D., Hamacher, H. W., and Kalsch, M. T. (2008). Simultaneous scheduling and location (scheloc) : The planar scheloc makespan problem. *Journal of Scheduling*, 12 :361–374.

[Erera et al., 2009] Erera, A., Morales, J., and Savelsbergh, M. (2009). Robust optimization for empty repositioning problems. *Operations Research*, 57(2) :468–483.

[Fahy, 1994] Fahy, R. F. (1994). Exit 89-an evacuation model for high rise buildings. *Fire Safety Science*, 4 :657–668.

[Fischetti and Monaci, 2013] Fischetti, M. and Monaci, M. (2013). Proximity search for 0-1 mixed-integer convex programming. Technical report, Università degli Studi di Padova.

[Forcael et al., 2014] Forcael, E., González, V., Orozco, F., Vargas, S., Pantoja, A., and Moscoso, P. (2014). Ant colony optimization model for tsunamis evacuation routes. *Computer-Aided Civil and Infrastructure Engineering*, 29(10) :723–737.

[Franzese and Han, 2001] Franzese, O. and Han, L. D. (2001). *A Methodology for the Assessment of Traffic Management Strategies for Large-Scale Emergency Evacuations*. ITS America, 11th Meeting. Miami Beach, Florida, USA.

[Gawiejnowicz, 2007] Gawiejnowicz, S. (2007). Scheduling deteriorating jobs subject to job or machine availability constraints. *European Journal of Operational Research*, 180(1) :472 – 478.

[Gawiejnowicz et al., 2006] Gawiejnowicz, S., Kurc, W., and Pankowska, L. (2006). Parallel machine scheduling of deteriorating jobs by modified steepest descent search. In Wyrzykowski, R., Dongarra, J., Meyer, N., and Waoniewski, J., editors, *Parallel Processing and Applied Mathematics*, volume 3911 of *Lecture Notes in Computer Science*, pages 116–123. Springer Berlin Heidelberg.

[Goerigk, 2012] Goerigk, M. (2012). *Algorithm Engineering in Robust Optimization*. PhD thesis. PHD thesis, University Gottingen.

[Goerigk et al., 2014a] Goerigk, M., Deghdak, K., and Hessler, P. (2014a). A comprehensive evacuation planning model and genetic solution algorithm. *Transportation Research Part E : Logistics and Transportation Review*, 71(0) :82 – 97.

[Goerigk et al., 2015] Goerigk, M., Deghdak, K., and T'Kindt, V. (2015). A two-stage robustness approach to evacuation planning with buses. *Transportation Research Part B : Methodological*, 78(0) :66 – 82.

[Goerigk and Grun, 2014] Goerigk, M. and Grun, B. (2014). A robust bus evacuation model with delayed scenario information. *OR Spectrum*, 36(4) :923–948.

[Goerigk et al., 2013] Goerigk, M., Grun, B., and Hessler, P. (2013). Branch and bound algorithms for the bus evacuation problem. *Computers & Operations Research*, 40(12) :3010 – 3020.

[Goerigk et al., 2014b] Goerigk, M., Grun, B., and Hessler, P. (2014b). Combining bus evacuation with location decisions : A branch-and-price approach. *Transportation Research Procedia*, 2(0) :783 – 791. The Conference on Pedestrian and Evacuation Dynamics 2014 (PED 2014), 22-24 October 2014, Delft, The Netherlands.

[Graat et al., 1999] Graat, E., Midden, C., and Bockholts, P. (1999). Complex evacuation : effects of motivation level and slope of stairs on emergency egress time in a sports stadium. *Safety Science*, 31(2) :127–141.

[Hamacher and Hennes, 2007] Hamacher, H. W. and Hennes, H. (2007). Integrated scheuling and location models : Single machine makespan problems. *Studies in Locational Analysis*, 16 :77–90.

[Hamacher and Tjandra, 2001] Hamacher, H. W. and Tjandra, S. A. (2001). Mathematical modeling of evacuation problems : A state of the art. *In Pedestrian and Evacuation Dynamics (Schreckinberg, M. and Sharma, S. D. eds)*, 1964 :227–266.

[Hamza-Lup et al., 2005] Hamza-Lup, G., Hua, K., and Peng, R. (2005). Applying e-transportation to traffic evacuation management under human-caused threats. In *The 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pages 666–673. http ://doi.ieeecomputersociety.org/10.1109/EEE.2005.34.

[Han and Yuan, 2005] Han, L. D. and Yuan, F. (2005). Evacuation modeling and operations using dynamic traffic assignment and most desirable destination approaches. *Transportation Research Board, 84th Annual Meeting*, (05) :2401–2406.

[Henn, 2001] Henn, V. (2001). *Information routière et affectation du trafic : vers une modélisation floue*. PhD thesis, Université de Saint-Étienne.

[Hennes, 2005] Hennes, H. (2005). *Integration of Scheduling and Location Models*. PhD thesis, University of Kaiserslautern, Allemagne.

[Hessler and Deghdak, 2015a] Hessler, C. and Deghdak, K. (2015a). The discrete parallel machine makespan scheduling-location problem. In *XXII EURO Working Group on Locational Analysis Meeting, EWGLA*, pages 61–62, Budapest, Hungarian.

[Hessler and Deghdak, 2015b] Hessler, C. and Deghdak, K. (2015b). The discrete parallel machine makespan scheduling-location problem. In *Proceedings of Multidisciplinary International Scheduling Conference : Theory and Applications, MISTA, vol.7*, pages 659–661, Pragues, czech republic.

[Hessler and Deghdak, 2015c] Hessler, C. and Deghdak, K. (2015c). Discrete parallel machine makespan scheloc problem. *Technical report, University of Kaiserslautern*. http ://nbn-resolving.de/urn/resolver.pl ?urn :nbn :de :hbz :386-kluedo-41297.

[HMM Associates, 1980] HMM Associates (1980). Evacuation time estimates for area near pilgrim station. Report prepared for Boston Edison Company, Boston.

[Hobeika and Jamei, 1985] Hobeika, A. and Jamei, B. (1985). *MASSVAC : A Model for Calculating Evacuation Times Under Natural Disasters*, volume 15. Conference on Computer Simulation in Emergency Planning, La Jolla, California.

[Hobeika and Kim, 1998] Hobeika, A. and Kim, C. (1998). Comparison of traffic assignments in evacuation modeling. *IEEE Transactions on Engineering Management*, 45(2) :192–198.

[Hobeika et al., 1994] Hobeika, A., Kim, S., and Beckwith, R. (1994). A decision-support system for developing evacuation plans around nuclear-power stations. *INTERFACES*, 24(5) :22 – 35.

[Hobeika, 2002] Hobeika, A. G. (2002). *TEDSS : A Software for Evacuating People around Nuclear Power Stations*, pages 688–695. 7th International Conference on Applications of Advanced Technologies in Transportation, ASCE, Reston, USA. http ://ascelibrary.org/doi/abs/10.1061/40632

[Hung and Chen, 1991] Hung, Y.-C. and Chen, G.-H. (1991). On the quickest path problem. In Dehne, F., Fiala, F., and Koczkodaj, W., editors, *Advances in Computing and Information-ICCI '91*, volume 497 of *Lecture Notes in Computer Science*, pages 43–46. Springer Berlin Heidelberg.

[Jaszkiewicz, 2004] Jaszkiewicz, A. (2004). Evaluation of multiple objective metaheuristics. In Gandibleux, X., Sevaux, M., Sörensen, K., and T'kindt, V., editors, *Metaheuristics for Multiobjective Optimisation*, volume 535, pages 65–89. Springer Berlin Heidelberg.

[Jayakrishnan et al., 1994] Jayakrishnan, R., Tsai, W. T., Prashker, J. N. ., and Rajadhyaksha, S. (1994). A faster path-based algorithm for traffic assignment. *Transportation Research Part C*, 2(34) :75–83.

[Jeng and Lin, 2006] Jeng, A. A. K. and Lin, B. M. T. (2006). A note on parallel-machine scheduling with deteriorating jobs. *Journal of the Operational Research Society*, 58(6) :824 – 826.

[Jenkins, 2000] Jenkins, L. (2000). Selecting scenarios for environmental disaster planning. *European Journal of Operational Research*, 121(2) :275 – 286.

[Ji et al., 2006] Ji, M., He, Y., and Cheng, T. (2006). Scheduling linear deteriorating jobs with an availability constraint on a single machine. *Theoretical Computer Science*, 362(1-3) :115 – 126.

[Kagaris et al., 1999] Kagaris, D., Pantziou, G. E., Tragoudas, S., and Zaroliagis, C. D. (1999). Transmissions in a network with capacities and delays. *Networks*, 33(3) :167–174.

[Kalsch and Drezner, 2010] Kalsch, M. and Drezner, Z. (2010). Solving scheduling and location problems in the plane simultaneously. *Computurs and Operations Research*, 37 :256–264.

[Kalsch, 2009] Kalsch, M. T. (2009). *Scheduling - Location (ScheLoc) Models, Theory and Algorithms*. PhD thesis, University of Kaiserslautern.

[Kaufmann, 2014] Kaufmann, C. (2014). A polynomial time algorithm for an integrated scheduling and location problem. In *Proceedings of the 14th International Conference on Project Management and Scheduling*, pages 124–128. https ://mediatum.ub.tum.de/doc/1200549/1200549.pdf.

[Kisko and Francis, 1985] Kisko, T. M. and Francis, R. L. (1985). Evacnet+ : A computer program to determine optimal building evacuation plans. *Fire Safety Journal*, 9(2) :211 – 220.

[Konak et al., 2006] Konak, A., Coit, D. W., and Smith, A. E. (2006). Multi-objective optimization using genetic algorithms : A tutorial. *Reliability Engineering & System Safety*, 91(9) :992–1007.

[Kongsomsaksakul et al., 2005] Kongsomsaksakul, S., Yang, C., and Chen, A. (2005). Shelter location-allocation model for flood evacuation planning. *Journal of the Eastern Asia Society for Transportation Studies*, 6 :4237–4252.

[Kulshrestha et al., 2014] Kulshrestha, A., Lou, Y., and Yin, Y. (2014). Pick-up locations and bus allocation for transit-based evacuation planning with demand uncertainty. *Journal of Advanced Transportation*, 48(7) :721–733.

[Kunnathur and Gupta, 1990] Kunnathur, A. S. and Gupta, S. K. (1990). Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. *European Journal of Operational Research*, 47(1) :56 – 64.

[Lemoine et al., 2014] Lemoine, A. et al. (2014). Pligurian earthquake : Seismic and tsunami scenario modeling, from hazard to risk assessment towards evacuations planning. In *Proceedings of the Second European Conference on Earthquake Engineering and Seismology*.

[Liebchen et al., 2009] Liebchen, C., Lübbecke, M., Möhring, R. H., and Stiller, S. (2009). The concept of recoverable robustness, linear programming recovery, and railway applications. In Ahuja, R. K., Möhring, R., and Zaroliagis, C., editors, *Robust and online large-scale optimization*, volume 5868 of *Lecture Note on Computer Science*, pages 1–27. Springer.

[Liu et al., 2006] Liu, Y., Lai, X., and Chang, G. (2006). Two-level integrated optimization system for planning of emergency evacuation. *Journal of Transportation Engineering*, 132(10) :800–807.

[Liu et al., 2005] Liu, Y., Zou, N., and Chang, G.-L. (2005). An integrated emergency evacuation system for real-time operations - a case study of ocean city, maryland under hurricane attacks. In *Intelligent Transportation Systems*, pages 464–469.

[Lovas, 1998] Lovas, G. G. (1998). Models of wayfinding in emergency evacuations. *European Journal of Operational Research*, 105(3) :371 – 389.

[Mahmassani and Peeta, 1995] Mahmassani, H. and Peeta, S. (1995). System optimal dynamic assignment for electronic route guidance in a congested traffic network. In Gartner, N. and Improta, G., editors, *Urban Traffic Networks*, Transportation Analysis, pages 3–37. Springer Berlin Heidelberg.

[Mahmassani, 2001] Mahmassani, H. S. (2001). Dynamic network traffic assignment and simulation methodology for advanced system management applications. *Networks and Spatial Economics*, 1(3-4) :267–292.

[Maranzana, 1964] Maranzana, F. E. (1964). On the location of supply points to minimize tranport costs. *Operational Research Quarterly*, 15 :261–270.

[Mitchell and Radwan, 2006] Mitchell, S. and Radwan, E. (2006). Heuristic priority ranking of emergency evacuation staging to reduce clearance time. *Transportation Research Record*, 1964 :219–228.

[Mladenović et al., 2007] Mladenović, N., Brimberg, J., Hansen, P., and Moreno Pérez, J. A. (2007). The p-median problem : A survey of metaheuristic approaches. *European Journal of Operational Research*, 179 :927–937.

[Mosheiov, 1994] Mosheiov, G. (1994). Scheduling jobs under simple linear deterioration. *Computers & Operations Research*, 21(6) :653 – 659.

[Mosheiov, 1996] Mosheiov, G. (1996). On shop scheduling with deteriorating jobs. Working Paper, School of Business Administration, University of Jerusalem.

[Ndiaye et al., 2014a] Ndiaye, I. A., Neron, E., and Jouglet, A. (2014a). Macroscopic evacuation plans for natural disasters : A lexicographical approach for duration and safety criteria : Lex((qjs) flow). *ORSPECTRUM*, submitted.

[Ndiaye et al., 2014b] Ndiaye, I. A., Neron, E., Linot, A., Monmarche, N., and Goerigk, M. (2014b). A new model for macroscopic pedestrian evacuation planning with safety and duration criteria. *Transportation Research Procedia*, 2(0) :486 – 494.

[Neron et al., 2011] Neron, E., T'kindt, V., Hamacher, H., foertser, E., Sarrhini, K., and Gasnier, B. (2011). Projet anr csosg.

[Ng and Waller, 2010] Ng, M. and Waller, S. T. (2010). Reliable evacuation planning via demand inflation and supply deflation. *Transportation Research Part E : Logistics and Transportation Review*, 46(6) :1086 – 1094.

[Opasanon and Miller-Hooks, 2009] Opasanon, S. and Miller-Hooks, E. (2009). The safest escape problem. *Journal of the Operational Research Society*, (60) :1749–1758.

[Peeta, 1994] Peeta, S. (1994). *System Optimal Dynamic Traffic Assignment in Congested Networks with Advanced Information Systems.* PhD thesis, University of Texas at Austin, USA.

[Peeta and Mahmassani, 1995a] Peeta, S. and Mahmassani, H. (1995a). Multiple user classes real-time traffic assignment for online operations : a rolling horizon solution framework. *Transportation Research Part C : Emerging Technologies*, 3(2) :83–98.

[Peeta and Mahmassani, 1995b] Peeta, S. and Mahmassani, H. (1995b). System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research*, 60(1) :81–113.

[Pidd et al., 1996] Pidd, M., de Silva, F., and Eglese, R. (1996). A simulation model for emergency evacuation. *European Journal of Operational Research*, 90(3) :413 – 419.

[Pisinger, 1995] Pisinger, D. (1995). *Algorithms for Knapsack Problems.* PhD thesis, University of Copenhagan, Denmark.

[Rathi and Solanki, 1993] Rathi, A. and Solanki, R. (1993). Simulation of traffic flow during emergency evacuations : a microcomputer based modeling system. In *Simulation Conference Proceedings, 1993. Winter*, pages 1250–1258.

[Reese, 2006] Reese, J. (2006). Solution methods for the p-median problem : An annotated bibliography. *NETWORKS*, 48 :125–142.

[Savelsbergh, 1994] Savelsbergh, M. W. P. (1994). Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6(4) :445–454.

[Sbayti et al., 2007] Sbayti, H., Lu, C.-C., and Mahmassani, H. S. (2007). Efficient implementation of method of successive averages in simulation-based dynamic traffic assignment models for large-scale network applications. *Transportation Research Record : Journal of the Transportation Research Board*, 2029 :22–30.

[Sbayti and Mahmassani, 2006] Sbayti, H. and Mahmassani, H. S. (2006). Optimal scheduling of evacuation operations. *Journal of the Transportation Research Board*, 1964 :238–246.

[Sheffi et al., 1982a] Sheffi, Y., Mahmassani, H., and Powell., W. B. (1982a). Evacuation studies for nuclear power plant sites : A new challenge for transportation engineers. *ITE Journal*, 51(6) :25–28.

[Sheffi et al., 1982b] Sheffi, Y., Mahmassani, H., and Powell, W. B. (1982b). A transportation network evacuation model. *Transportation Research Part A*, 16(3) :209–218.

[Sherali et al., 1991] Sherali, H. D., Carter, T. B., and Hobeika, A. G. (1991). A location-allocation model and algorithm for evacuation planning under hurricane/flood conditions. *Transportation Research Part B : Methodological*, 25(6) :439 – 452.

[Smith, 1956] Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2) :59–66.

[Sousa and Wolsey, 1992] Sousa, J. P. and Wolsey, L. A. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, 54(1-3) :353–367.

[Stiller, 2008] Stiller, S. (2008). *Extending concepts of reliability. Network creation games, real-time scheduling, and robust optimization.* PhD thesis, TU Berlin.

[Tanaev et al., 1994] Tanaev, V., Gordon, V., and Shafransky, Y. (1994). *Scheduling Theory : Single-Stage Systems.* Dordrecht Kluwer.

[Tansel et al., 1983] Tansel, B., Francis, R., and Lowe, T. (1983). State of the art - location on networks : A survey. part i : The p-center and p-median problems. *Management Science*, 29 :482–497.

[Taylor, 2003] Taylor, N. (2003). The CONTRAM dynamic traffic assignment model. *Networks and Spatial Economics*, 3(3) :297–322.

[T'kindt and Billaut, 2006] T'kindt, V. and Billaut, J.-C. (2006). *Multicriteria scheduling : theory, models and algorithms.* Springer-Verlag Berlin Heidelberg, 2nd edition.

[T'kindt et al., 2011] T'kindt, V., Hamacher, H., Neron, E., foertser, E., Sarrhini, K., and Gasnier, B. (2011). Decision support system for large-scale evacuation logistics : Joint research proposal.

[TSS-Transport Simulation Systems, 1980] TSS-Transport Simulation Systems (1980). Aimsun. http ://www.aimsun.com/wp/.

[Tufekci and Kisko, 1991] Tufekci, S. and Kisko, T. (1991). Regional evacuation modeling system (REMS) : A decision support system for emergency area evacuations. *Computers & Industrial Engineering*, 21(1-4) :89 –93.

[Tuydes, 2005] Tuydes, H. (2005). *Network Traffic Management under Disaster Conditions.* PhD thesis, Northwestern University, USA.

[Tuydes and Ziliaskopoulos, 2004] Tuydes, H. and Ziliaskopoulos, A. (2004). *Network Re-Design to Optimize Evacuation Contraflow*. Transportation Research Board, 83rd Annual Meeting.

[Tuydes and Ziliaskopoulos, 2006] Tuydes, H. and Ziliaskopoulos, A. (2006). Tabu-based heuristic approach for optimization of network evacuation contraflow. *Transportation Research Record : Journal of the Transportation Research Board*, 1964(1) :157–168.

[United States Coast Guard , 2006] United States Coast Guard (2006). Katrina history. http ://www.uscg.mil/History/katrina/katrinaindex.asp.

[University of Edinburgh, 1990] University of Edinburgh (1990). Paramics. http ://www.paramics-online.com/.

[Urbanik, 1978] Urbanik, T. (1978). Texas hurricane evacuation study. Texas Transportation Institute, College Station, TX.

[Visual Solutions Incorporated, 1989] Visual Solutions Incorporated (1989). Vissim. http ://www.vissim.com/.

[Wardrop, 1952] Wardrop, J. G. (1952). Some theoretical aspects of road traffic research. *Proceedings of Institute of Civil Engineers*, 2(1) :325–378.

[Wiedermann, 1974] Wiedermann, R. (1974). Simulation des strassenverkehrsflusses. In Schriftenreihe des Institutes fur Verkehrswesen des University Karlsruhe, Allemagne.

[Xu and Wunsch, 2005] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16 :645–678.

[Yamada, 1996] Yamada, T. . (1996). A network flow approach to a city emergency evacuation planning. *International Journal of Systems Science*, 27(10) :931–936.

[Yao et al., 2009] Yao, T., Mandala, S., and Chung, B. (2009). Evacuation transportation planning under uncertainty : A robust optimization approach. *Networks and Spatial Economics*, 9(2) :171–189.

[Yazici and Ozbay, 2007] Yazici, M. A. and Ozbay, K. (2007). Impact of probabilistic road capacity constraints on the spatial distribution of hurricane evacuation shelter capacities. *Transportation Research Record : Journal of the Transportation Research Board*, 2022(1) :55–62.

[Yildirimoglu and Geroliminis, 2013] Yildirimoglu, M. and Geroliminis, N. (2013). Dynamic traffic assignment with macroscopic fundamental diagrams. *13th Swiss Transport Research Conference, Monte Verità, Swissland*.

[Yuan et al., 2006] Yuan, F., Han, L. D., Chin, S.-M., and Hwang, H. (2006). Proposed framework for simultaneous optimization of evacuation traffic destination and route assignment. *Transportation Research Record : Journal of the Transportation Research Board*, 1964(1).

[Ziliaskopoulos and Mahmassani, 1993] Ziliaskopoulos, A. and Mahmassani, H. (1993). *A Time-dependent Shortest Path Algorithm for Real-time Intelligent Vehicle/highway Systems Applications*, volume 1408. Transportation Research Board.

[Ziliaskopoulos, 2000] Ziliaskopoulos, A. K. (2000). A linear programming model for the single destination system optimum dynamic traffic assignment problem. *Transportation Science*, 34 :37–49.

[Ziliaskopoulos and Mahmassani, 1996] Ziliaskopoulos, A. K. and Mahmassani, H. S. (1996). On finding least time paths considering delays for intersection movements. *Transportation Research Part B*, 30(5) :359–367.

[Zong et al., 2010] Zong, X., Xiong, S., Fang, Z., and Lin, W. (2010). Multi-objective ant colony optimization model for emergency evacuation. In *Natural Computation (ICNC), 2010 Sixth International Conference on*, volume 6, pages 2774–2778.

-Kaouthar Deghdak-

Étude et résolution de problèmes d'ordonnancement d'opérations d'évacuation

Résumé :

Les travaux présentés dans cette thèse, qui s'inscrivent dans le cadre du projet franco-allemand DSS_Evac_Logistic, visent à proposer des méthodes permettant de calculer des plans d'évacuation macroscopiques d'une ville lors d'une catastrophe majeure. Deux problèmes d'évacuations sont considérés dans cette thèse : le problème d'évacuation par bus et le problème d'évacuation par bus et voitures.

Le problème d'évacuation par bus a pour objectif de définir un plan d'évacuation afin de mettre à l'abri les évacués. Dans cette thèse, nous nous sommes intéressés à l'étude de trois versions du problème d'évacuation par bus. La première version est monocritère où nous cherchons à minimiser la date de fin d'évacuation. Puis, dans le second problème et afin d'assurer la sécurité des évacués, nous avons considéré une version bicritère qui généralise le cas monocritère, en incluant le risque encouru lors de l'évacuation des personnes. Les deux critères à minimiser sont la date de fin d'évacuation et le risque. La troisième version est une version robuste bicritère qui permet d'appréhender l'incertitude sur les données. Le but est de minimiser à la fois la date de fin d'évacuation et les modifications apportées sur une solution, de sorte qu'elle soit réalisable pour n'importe quel scénario de données. Pour résoudre ces problèmes d'évacuation par bus, nous avons proposé des méthodes exactes et des méthodes heuristiques.

Le second problème d'évacuation considéré par bus et voitures est multicritère. Il a pour but de définir pour chaque groupe de personnes, sa date de début d'évacuation, le centre de secours de rattachement, et le chemin menant à ce centre de secours. Nous cherchons à déterminer les centres de secours à ouvrir afin de minimiser la date de fin d'évacuation et l'exposition aux risques. Un modèle mathématique est proposé pour la résolution exacte des instances de petite taille de ce problème. Pour résoudre des instances de grande taille, des méthodes évolutionnaires et des méthodes basées sur le calcul de chemins multiobjectifs sont développées.

Dans cette thèse, toutes les méthodes proposées sont testées et validées sur des instances aléatoires et des instances réelles de la ville de Kaiserslautern du coté allemand et de la ville de Nice du coté français. Ce choix des villes a été imposé par le projet sur lequel s'inscrit cette thèse.

Abstract :

The work presented in this thesis, which is a part of the Franco-German project DSS_Evac_Logistic, aims at proposing methods to calculate macroscopic evacuation plans for mid-size towns after a tremendous disaster. Two evacuation problems have been tackled in this thesis : the bus evacuation problem and bus-and-vehicle evacuation problem.

The bus evacuation problem aims at calculating an evacuation plan to relocate evacuees outside the endangered area. In this thesis, we consider three versions of the bus evacuation problem. The first one is a monocriterion problem, where the objective is to minimize the maximum evacuation time. In order to guarantee the safety of evacuees, we have considered a bicriteria problem, which is a generalization of the monocriterion version, in which we take into consideration the risk exposure of the evacuees. Consequently, the bicriteria problem is solved by minimizing the total evacuation time and the risk. The third version is a bicriteria robust version because most of the planning data is subject to uncertainty. The goal is to minimize both the evacuation time and the vulnerability of the schedule that is subject to different evacuation circumstances. To solve all the versions of the bus evacuation problem, we have developed exact solutions based on mathematical formulation to address small instances and heuristic solutions to deal with larger instances.

The second evacuation problem studied in this thesis is a multi-criteria problem, which aims at defining the departure time, the route and the shelters for each group of people. Specifically, we would like to determine a set of appropriate shelters for hosting people while minimizing the total evacuation time and the risk exposure of the evacuees. For this purpose, a mathematical formulation has been proposed to deal with small instances. For larger instances, evolutionary methods and methods based on multi-objectives shortest path searches have been developed.

For validation and experiments, all the proposed methods have been tested using random instances and real instances of Kaiserslautern city in Germany and Nice city in France. The choice of these towns is imposed by the project in which this thesis takes place.