



HAL
open science

Urban Transportation Road Network Problems : Models, Methods and Application

Yipeng Huang

► **To cite this version:**

Yipeng Huang. Urban Transportation Road Network Problems : Models, Methods and Application. Infrastructures de transport. Université de Technologie de Troyes, 2018. English. NNT : 2018TROY0016 . tel-03563657

HAL Id: tel-03563657

<https://theses.hal.science/tel-03563657>

Submitted on 9 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse
de doctorat
de l'UTT

Yipeng HUANG

**Urban Transportation Road
Network Problems:
Models, Methods and Application**

Spécialité :
Optimisation et Sécurité des Systèmes

2018TROY0016

Année 2018



THESE

pour l'obtention du grade de

DOCTEUR de l'UNIVERSITE DE TECHNOLOGIE DE TROYES

Spécialité : OPTIMISATION ET SURETE DES SYSTEMES

présentée et soutenue par

Yipeng HUANG

le 28 mai 2018

Urban Transportation Road Network Problems: Models, Methods and Application

JURY

M. G. GONCALVES	PROFESSEUR DES UNIVERSITES	Président
M. C. DUHAMEL	MAITRE DE CONFERENCES	Directeur de thèse
M. L. GOUVEIA	PROFESSOR CATEDRATICO	Examineur
M. B. GRABOT	PROFESSEUR DES UNIVERSITES	Rapporteur
M. O. PÉTON	PROFESSEUR IMT - HDR	Rapporteur
M. C. PRINS	PROFESSEUR DES UNIVERSITES	Examineur
Mme A. C. SANTOS	MAITRE DE CONFERENCES - HDR	Directrice de thèse

Personnalité invitée

M. F. GUÉNIN	DIRECTEUR REGULATION TRAFIC TCM
--------------	---------------------------------

Acknowledgement

First, I would like to express my sincere thankfulness to my supervisors, Andréa Cynthia SANTOS and Christophe DUHAMEL, not only for their marvelous guidance to my research with immense intelligence and patience, but also for their companionship every time I got difficulties and for teaching me the philosophy of life. During these years of study, I always feel honored to spend my time with them, either for work or for leisure. Personally, I have really noticed my progression in terms of the graph theory, algorithms in general, the optimization, as well as the line of reasoning and the style of living.

I also thank to Olivier PETON, Bernard GRABOT, Louis F. GOUVEIA, Christian PRINS and Frédéric GUENIN for accepting to be part of the jury and for their valuable time and suggestions to improve this research.

Likewise, I thank all my fellows at UTT for the stimulating discussions, for their advises to my thesis as well as my life, and for all the fun we have had in the last three years, especially to Amadeu Almeida COCO, Omar JAAFOR, Alexandre Xavier MARTINS, Celso SAKURABA, Elyn SOLANO CHARRIS, Xixi WANG and Yuchen ZHAO. They are all warriors in various domains and made me feel that I was not lonely in this long way of research.

I want also to thank to all LOSI professors and the administrative team at UTT for their support directly or indirectly in my Ph.D. Especially, thanks Bernadette ANDRE, Véronique BANSE, Pascale DENIS, Thérèse KAZARIAN and Isabelle LECLERCQ for all your help.

Last but not least, I would like to thank my wife and my parents, for their love, kindness and full support from every aspect.

Contents

	Page
Contents	v
List of Figures	ix
List of Tables	xii
List of Algorithms	xiii
Glossary	xvi
1 Introduction	1
1.1 Document structure	5
1.2 List of publications	6
1.2.1 Submitted articles	6
1.2.2 International conference with indexed acts	6
1.2.3 International conference with extended abstracts	7
1.2.4 International conference abstracts	7
1.2.5 Others	7
2 State of the art	9
2.1 Related researches on graph theory	9
2.2 Network design and applications	11
2.2.1 Theoretical studies and surveys	11
2.2.2 Applications and case studies on NDP	13
3 Formal definitions and mathematical formulations	15
3.1 Notations	15
3.2 Road networks problems with disruptions and connecting requirements	16
3.3 Unidirectional road networks problems	17

3.4	Multidirectional road networks problems	22
3.5	Computational experiments and numerical results	25
4	Methods for solving single- and bi-objective URND and MRND	31
4.1	An orientation algorithm	31
4.2	Single-objective optimization	35
4.2.1	BRKGA	36
4.2.2	ILS	40
4.3	Bi-objective optimization	44
4.3.1	An ϵ -constraint method for bi-URND and bi-MRND	46
4.3.2	Bi-objective BRKGA adaptation for bi-URND and bi-MRND	47
4.3.3	Bi-objective ILS adaptation for bi-URND and bi-MRND	48
4.3.4	NSGA-II	49
4.4	Computational experiments and numerical results	50
4.4.1	Parameter tuning	51
4.4.2	Results for single-objective URND and MRND	53
4.4.3	Bi-objective optimization	62
4.4.4	An application to a real urban network	67
4.4.5	Conclusion	71
5	Application : Optimal traffic Deviation System	75
5.1	Requirements analysis	76
5.1.1	Functional requirements	76
5.1.2	Nonfunctional requirements	78
5.2	ODS architecture	80
5.2.1	Human Computer Interactions	82
5.2.2	Database	88
5.2.3	Computing core	89
5.3	Addressing technological gaps	90
6	Concluding remarks and future works	93
	Appendices	97
	Appendices	99
	A Résumé étendu en français	99
	B Tables of parameter tuning results	123

C	Figures of Pareto fronts of ϵ-constraint method, BRKGA, ILS and NSGA-II	131
D	Figures of Pareto fronts of BRKGA, ILS and NSGA-II	141
	Bibliography	155

List of Figures

	Page
3.1 An example network with one blockage addressed by URND	18
3.2 Modeling a deviation.	20
3.3 Values for flow-based variables f and y for p_1	21
3.4 Values for flow-based variables f and y for p_2	21
3.5 An example network with one blockage addressed by MRND	23
3.6 2-directed multigraphs reductions: 1 direction	24
3.7 2-directed multigraphs reductions: 2 directions	24
4.1 Examples of the strong orientation algorithm	33
4.2 Orientation process breakdown for Figure 4.1-(b)	34
4.3 Orientation impossible to generate	34
4.4 \mathcal{N}_1 move in cycle $\{(2, 3), (3, 4), (4, 2)\}$	42
4.5 \mathcal{N}_2 move at node 4	42
4.6 \mathcal{N}_3 move at arc $(1, 2)$	43
4.7 justification=centering	47
4.8 Time-to-target plots of BRKGA and ILS for URND c_1 objective	57
4.9 Time-to-target plots of BRKGA and ILS for MRND c_1 objective	60
4.10 Pareto fronts on 5x5-b1 (bi-URND)	64
4.11 Pareto fronts on 5x5-b2-100 (bi-MRND)	65
4.12 Pareto fronts on 8x8-b6 (bi-URND)	66
4.13 Pareto fronts on 8x8-b1-75 (bi-URND)	68
4.14 Graph of partial Troyes downtown road network with 24 nodes	68
4.15 Graph of entire Troyes downtown road network with 58 nodes	68
4.16 Graph of partial Troyes downtown road network with 102 nodes	68
4.17 Pareto fronts for the troyes-v24-b2 instance	72
4.18 Pareto fronts for the troyes-v58-b6 instance	72
5.1 System architecture of ODS	81

5.2	Mockup of home screen with organization of major activities	83
5.3	Mockup of deviation management with a map view and list of blocking events	84
5.4	Conception of responsive map views with themes of day and night	84
5.5	Example of map tiles with map projection and tile coordinates	85
5.6	ODS logo	85
5.7	Palette of colors used in ODS	86
5.8	Fonts used in ODS	86
5.9	An example of deviations visualization on the map view	87
5.10	An example of intuitive comparison of deviations	87
5.11	Visualization of traffic signals on the map view	88
5.12	ODS database schema	89
A.1	Exemples de l'heuristique d'orientation des graphes	109
A.2	System architecture of ODS	118
C.1	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1	131
C.2	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2	132
C.3	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b4	132
C.4	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b6	133
C.5	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-25	133
C.6	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-50	134
C.7	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-75	134
C.8	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-100	135
C.9	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-25	135
C.10	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-50	136
C.11	Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-75	136

C.12 Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-100	137
C.13 Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b1	137
C.14 Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b2	138
C.15 Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b4	138
C.16 Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b6	139
D.1 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1	141
D.2 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2	142
D.3 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b4	142
D.4 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b6	143
D.5 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b1	143
D.6 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b2	144
D.7 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b4	144
D.8 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b6	145
D.9 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b1	145
D.10 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b2	146
D.11 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b4	146
D.12 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b6	147
D.13 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-25	147
D.14 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-50	148
D.15 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-75	148
D.16 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-100	149
D.17 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-25	149
D.18 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-50	150
D.19 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-75	150
D.20 Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-100	151
D.21 Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b1	151
D.22 Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b2	152
D.23 Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b4	152
D.24 Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b6	153

List of Tables

	Page
3.1 Characteristics of generated simple graphs	28
3.2 Characteristics of generated multigraphs	28
3.3 CPLEX results of small and medium simple instances (URND)	29
3.4 CPLEX results of small and medium multigraph instances (MRND)	30
4.1 Characteristics of generated larger simple graphs	52
4.2 Characteristics of generated larger multigraphs	52
4.3 Predefined possible values of the parameters	53
4.4 Chosen parameter values	53
4.5 Results of single-objective ILS parameter tuning on SD_I	54
4.6 Results of bi-objective BRKGA parameter tuning on R_B , with other parameters fixed: $SD_B = 2$, $I_B = 100$ and $J_B = 12$	54
4.7 c_1 minimization for URND (I)	55
4.8 c_2 minimization for URND (I)	56
4.9 c_1 minimization for MRND (I)	58
4.10 c_2 minimization for MRND (I)	59
4.11 c_1 minimization for URND (II)	61
4.12 c_2 minimization for URND (II)	61
4.13 c_1 minimization for MRND (II)	62
4.14 c_2 minimization for MRND (II)	62
4.15 Performance comparison of Pareto fronts for bi-URND (I)	63
4.16 Performance comparison of Pareto fronts for bi-MRND (I)	64
4.17 Performance comparison of Pareto fronts for bi-URND (II)	66
4.18 Performance comparison of Pareto fronts for bi-MRND (II)	67
4.19 Characteristics of Troyes network based instances	69
4.20 c_1 minimization on troyes-v24 instances	70
4.21 c_2 minimization on troyes-v24 instances	70

4.22	c_1 minimization on troyes-v58 and troyes-v102 instances	70
4.23	c_2 minimization on troyes-v58 and troyes-v102 instances	71
4.24	Performance comparison of Pareto fronts on troyes-v24 instances	72
4.25	Performance comparison of Pareto fronts on troyes-v58 instances	72
A.1	Exemple des résultats de l'optimisation mono-objectif sur c_1	115
A.2	Exemple des résultats de l'optimisation bi-objectif	116
B.1	Preliminary results of single-objective BRKGA parameter tuning on SD_{brkga} , with other parameters fixed: $R_{brkga} = 1.45$, $I_{brkga} = 100$ and $J_{brkga} = 12$. . .	123
B.2	Preliminary results of single-objective BRKGA parameter tuning on R_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $I_{brkga} = 100$ and $J_{brkga} = 12$. . .	124
B.3	Preliminary results of single-objective BRKGA parameter tuning on I_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $J_{brkga} = 12$. . .	124
B.4	Preliminary results of single-objective BRKGA parameter tuning on J_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $I_{brkga} = 100$. .	125
B.5	Preliminary results of single-objective ILS parameter tuning on J_{ils}	125
B.6	Preliminary results of bi-objective BRKGA parameter tuning on SD_{brkga} , with other parameters fixed: $R_{brkga} = 1.45$, $I_{brkga} = 100$ and $J_{brkga} = 12$	126
B.7	Preliminary results of bi-objective BRKGA parameter tuning on I_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $J_{brkga} = 12$	126
B.8	Preliminary results of bi-objective BRKGA parameter tuning on I_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $I_{brkga} = 100$	127
B.9	Preliminary results of NSGA-II parameter tuning on SD_{nsgaii} , with other parameters fixed: $R_{nsgaii} = 1.45$, $I_{nsgaii} = 100$, $\rho_c = 0.3$ and $\rho_m = 0.03$	127
B.10	Preliminary results of NSGA-II parameter tuning on R_{nsgaii} , with other parameters fixed: $SD_{nsgaii} = 2$, $I_{nsgaii} = 100$, $\rho_c = 0.3$ and $\rho_m = 0.03$	128
B.11	Preliminary results of NSGA-II parameter tuning on I_{nsgaii} , with other parameters fixed: $SD_{nsgaii} = 2$, $R_{nsgaii} = 1.75$, $\rho_c = 0.3$ and $\rho_m = 0.03$	128
B.12	Preliminary results of NSGA-II parameter tuning on ρ_c , with other parameters fixed: $SD_{nsgaii} = 2$, $R_{nsgaii} = 1.75$, $I_{nsgaii} = 100$ and $\rho_m = 0.03$	129
B.13	Preliminary results of NSGA-II parameter tuning on ρ_m , with other parameters fixed: $SD_{nsgaii} = 2$, $R_{nsgaii} = 1.75$, $I_{nsgaii} = 100$ and $\rho_c = 0.3$. .	129

List of Algorithms

	Page
Algorithm 4.1 <i>StrongOrient</i> (G, W, u)	34
Algorithm 4.2 <i>Decode</i> (G, c)	38
Algorithm 4.3 <i>Decode-Repair-c₂</i> (G, c)	39
Algorithm 4.4 <i>BRKGA</i> (G, p, n)	40
Algorithm 4.5 <i>Construct</i> (G)	42
Algorithm 4.6 <i>VND</i> (g)	43
Algorithm 4.7 <i>RestartOrShake</i> (g, i, p)	44
Algorithm 4.8 <i>ILS</i> (G, p)	45
Algorithm 4.9 ϵ – <i>constraint</i> (G, ϵ)	48
Algorithm 4.10 <i>ArcOrient</i> (G, W)	51
Algorithm 4.11 <i>Simplify</i> (G)	69

Glossary

ACO Ant Colony Optimization.

AOF Aggregate Objective Function.

B&B Branch-and-Bound.

BKS Best Known Solutions.

BRKGA Biased Random Key Genetic Algorithm.

DFS Depth-First Search.

EA Evolutionary Algorithm.

GA Genetic Algorithm.

GIS Geographical Information System.

GRASP Greedy Randomized Adaptive Search Procedures.

GUI Graphical User Interface.

HCI Human Computer Interaction.

HV Hypervolume.

ILS Iterated Local Search.

LS Local Search.

MOEA Multi-Objective Evolutionary Algorithm.

MOGA Multi-Objective Genetic Algorithm.

MRND Multidirectional Road Network problem with Disruptions and connecting requirements.

NDP Network Design Problem.

NSGA Non-dominated Sorting Genetic Algorithm.

NSGA-II Non-dominated Sorting Genetic Algorithm II.

O-D Origin-Destination.

ODS Optimal traffic Deviation System.

OSM OpenStreetMap.

RKGA Random Key Genetic Algorithm.

RND Road Network problem with Disruptions and connecting requirements.

SCC Strongly Connected Components.

SNOP Strong Network Orientation Problem.

SPP Shortest Path Problem.

TS Tabu Search.

UI User Interface.

URND Unidirectional Road Network problem with Disruptions and connecting requirements.

UX User Experience.

VEGA Vector Evaluated Genetic Algorithm.

VND Variable Neighborhood Descent.

Chapter 1

Introduction

The steady growth of population in urban areas has a strong impact on the basic infrastructures of transportation networks, which are often close to saturation due to their limited capacity. Today, the urban population is already higher than the rural population around the world, but the concentration of population in urban areas is still increasing. The 2014 revision of the World Urbanization Prospects of the United Nations indicates that by 2050, the urban population will account for nearly 66% of total population (6.3 over 9.5 billion of people). This implies an increase of urban traffic flows. Hence, managing the daily traffic is an important and challenging issue, especially in rush hours. Expanding the networks by constructing new routes can be a solution, but it cannot be indefinitely carried out since geographical resources are often scarce, particularly in conurbations.

In recent years, serious modifications in the environment have been observed with significant ecological impacts, such as high concentrations of harmful particles in the air and reduction of agricultural areas resulting in less physical space available for production and livestock. In addition, maintaining urban network infrastructures requires increasing financial input. In this context, one of society's major challenges is to well manage urban transportation network infrastructures in a way to provide support for multiple daily trips for different purposes.

Paris is an example of a city with a high population density. It stands as a reference in the world due to its rich public transportation offer. In spite of that, Paris still needs further improvements in order to handle the traffic better in rush hours, with obvious difficulties for implementing solutions. Thus, several cities are instead trying to reduce the vehicle traffic flow, for instance by setting tolls, prohibited periods, motivating the use of bicycles, stimulating pedestrian habits and public transportation. Even planned cities face problems to managing their urban transportation network. For instance, Brasília, the planned capital of Brazil, was designed for 500 000 inhabitants and it was supposed to allow an easy traffic

flow without signal settings. However, 50 years later, this city contains about 2 million inhabitants and is known for its chaotic rush hour traffic. On an even larger scale, Beijing, the capital of China, has one of the largest and the most complicated urban transportation networks in the country, with 21.5 million citizens and about 5.5 million vehicles. As the political and economic center of China, Beijing also welcomes a large number of tourists every year, as well as many international conferences and visits. Although scalability and flexibility were considered in its road and highway design, the fast-growing demands of travel are leading to unbearable congestion, especially in rush hours and on the concentric rings of in-city highways. The geographical configuration can also be a complicating factor. Nice (France) and Rio de Janeiro (Brazil), settled between mountains and beaches, are examples where the geography is a barrier to managing the traffic.

The situation becomes even more complex whenever disruptions (blockages) occur on the road network, either predictable (road works, maintenance, social events, etc.) or not (accidents, bad weather conditions, disasters, etc.). Disruptions can block a number of road sections, thus reducing the traffic capacity and pass-rate in involved areas. They may also break travel paths between some locations on the network (loss of strong connectivity). Besides, they always generate more congestion, noise and air pollution. Although blockages disrupt a lot the urban road networks, they are usually not properly addressed from a global point of view in most of the cities, especially in medium-large ones. Therefore, innovating approaches should be studied so as to maintain a good level of network services.

Under this background, this thesis comprises two major goals. The one consists of an academic study on Road Network problem with Disruptions and connecting requirements (RND) and the other is dedicated to the software engineering (design and development) of Optimal traffic Deviation System (ODS). The investigation of RND is divided into two specific problems depending on the network structure: the Unidirectional Road Network problem with Disruptions and connecting requirements (URND) and the Multidirectional Road Network problem with Disruptions and connecting requirements (MRND). URND aims at addressing the problem in simpler downtown networks, where streets are likely one-way and simple digraphs can be used as an adequate modeling tool. On the other hand, MRND is designed to address the problem on general transportation networks by using multigraphs (*i.e.* graphs that can have multiple arcs between pairs of nodes), because streets can be two-way and can have multiple lanes. These two problems are formulated using a multi-flow mathematical model where the decision criteria involve the total travel distance between all pairs of nodes and the number of arc reversals. Reversing the direction of a road segment (an arc reversal) is thus allowed to restore the strong connectivity of the network when needed. Moreover, two heuristic methods, a Biased Random Key Genetic Algorithm (BRKGA) and

an Iterated Local Search (ILS), are proposed with specific adaptations to these two problems for both two objectives. The bi-objective versions of these two metaheuristics are also addressed and analyzed by comparing with an ϵ -constraint method and a Non-dominated Sorting Genetic Algorithm II (NSGA-II) in terms of the quality of Pareto fronts.

As an application of the research part, we develop the ODS to provide solutions for decision-makers in order to efficiently manage traffic flows and prevent any inaccessible locations on the network whenever disruptions occur. ODS is planned to be a Decision Support System to manage, schedule and compute deviations (alternate paths) to bypass the blocked routes on a road network. It is also a collaborative project with industry, academic researchers and the traffic control center of city Troyes in France. The industrial partner is a Chinese R&D company, Beijing YuanZhiTianCheng Technology Co. Ltd., and the academic researchers are from the University of Technology of Troyes (Troyes, France) and the Clermont-Auvergne University (Clermont-Ferrand, France). The Traffic Regulation Center of Troyes (Troyes, France) is also involved to provide data, support the development and to test the first deployments of the system, via the regional Innov'Action project "AGIR" (*systeme d'Aide à la décision pour la Gestion d'Interruptions Routières*) held by region Champagne-Ardenne (France). The major goal of this project is to run the developed application in a real context and to perform knowledge transfer from academies to the industry.

The core of ODS is a network solver to compute deviations for blocking events. A number of constraints have been identified, but only the most important ones are taken into account in this work which is supposed to be the first stage of the entire collaborative project AGIR. The first constraint is the strong connectivity of the network and the second one is the traffic load associated to the given disruption. For the former, it is obviously necessary to keep at least one path between all pairs of network points. Thus, the network solver first checks if a disruption breaks the strong connectivity. In such a case, it sends a warning and it proposes deviations using some arc reversals to restore it. For the latter constraint, it tries to avoid as much congestion as possible caused by redirecting a large traffic flow disrupted by a blockage to a route with a smaller traffic load capacity. Yet it can be the only solution in some situations. Hence, the solver does not block the computation of deviations but alerts instead the network manager that the capacity of the deviation may be insufficient. ODS also combines a Geographical Information System (GIS) based on "OpenStreetMap (OSM)" with the solver to obtain the network information from the targeted urban area, to display the computed solutions and to interact with the traffic manager. The roads are classified in several types which can be used to deduce the traffic load capacity by Zilske *et al.* (2011).

Other identified constraints and requirements are planned to be fulfilled gradually in

future stages. For instance, an accident can interrupt a route for a few hours, while tubing maintenance can take days or weeks. Thus, handling disruptions should be scheduled according to the importance and the emergency. In scientific terms, scheduling a set of deviations belongs to the NP-hard class of problems, while the real context raises some more difficulties, like dealing with large amount of data, sophisticated algorithms and integration of complex systems, among others. In addition, cars, buses and trucks have different sizes and functions, implying that a deviation adapted for one type of vehicle could not be used for others. A typical example is trucks employed for merchandise transportation whose weight can exceed the allowed capacity of a street. Another example is a deviation set for cars that cannot be used for long buses due to limitation in its turning radius. Furthermore, the alternating traffic can also be a good solution for some disruptions, but it is not yet evident to determine in which cases it could be applied, especially in narrow streets.

The deviations, apart from manually-designed ones, are computed according to one or more optimization criteria. The following objectives are considered: the bypass distance; the expected duration of travel among pairs of origins and destinations; the number of allowed modifications on the route directions; the number of signal settings and congestion. Such objectives can be handled individually, in priority order or even in a multi-criteria schema.

In a real urban context, several predictable and unpredictable disruptions may be considered per time period, which could generate conflicting decisions if addressed individually. Thus, a global vision of the network over time is necessary. ODS is being developed to support all of the mentioned requirements which have been defined together with the partners working in a real context. Its general goal is to manage and schedule deviations over time and it is expected to be integrated into the traffic control center of Troyes by 2018. A functional prototype has been first designed to validate the idea and workflow. The system in production is then developed for a professional use. Major features are mostly accomplished while fine tuning and optimization are still required in order to deliver progressively the final software. Currently, a number of modules are already working: (i) an interactive customized GIS based on the network of Troyes; (ii) a Graphical User Interface (GUI); (iii) computing deviations for a given blockage minimizing either the travel distance, the number of employed road segments or the number of traffic signals; (iv) network information administration and adjustment, especially the positions of traffic signals and traffic calming features; and (v) monthly monitoring of disruptions classified in three categories: (a) disruptions to handle, (b) deviations decided but not yet validated and (c) deviations validated.

1.1 Document structure

This document is globally organized around these two parts, *i.e.* the scientific part and the application one. The current chapter ends with the Section 1.2 listing the publications produced during this thesis and the conferences attended. Contributions ordered by chapters are as follows.

Chapter 2 gives a bibliographical survey in three related research fields: the graph theory, the Network Design Problem (NDP) as well as its applications in specific contexts, and algorithms. Basis of graph's strong connectivity has been investigated in previous works, either for simple graphs or for multigraphs. The orientation problem respecting strong connectivity was mentioned in some studies as well, referred as Strong Network Orientation Problem (SNOP). It is highly related to the class of RND. The arc reversal is another feature of RND, which is considered in some investigations. Besides, inspiration from NDP's point of view is also an important gateway to single- and bi-objective RND, especially in terms of mathematical models and frameworks for assignment problems, traffic signal location problems and toll setting problems. There are also many studies of NDP applied on real transportation networks.

Mathematical formulations are detailed in Chapter 3. Notations used in mathematical definitions are presented first in Section 3.1. In Section 3.2, RND is precisely defined based on the graph theory. Depending on the network structure, two problems are defined: the URND and the MRND. They rely on different underlying graphs, the one on simple graphs and the other on multigraphs. Section 3.3 gives the model of URND while Section 3.4 addresses the formulation for MRND. Specification of MRND model is also described by proposing the 2-directed multigraph and indexing multiple lanes. Experiments using the commercial solver CPLEX and their numerical results are developed in Section 3.5.

In Chapter 4, two approximate methods are proposed, a BRKGA and an ILS. Extending the Depth-First Search (DFS) on graphs, a random orientation algorithm is developed. It has been integrated in the two heuristics. Section 4.2 presents the two heuristics in their single-objective version. The basic principles of BRKGA are explained, such as the random keys, decoding, elitism, etc., as well as the adaptation introduced so as to handle both URND and MRND. The section also gives the details of ILS, including the constructive heuristic, local search operators and shaking. The bi-objective versions of these two methods are addressed in Section 4.3, especially the adaptations for constructing Pareto fronts. In addition, a NSGA-II method is adapted to bi-URND and bi-MRND since it is one of the most efficient method to compute approximate Pareto fronts. It will be used to measure the performance of our two dedicated metaheuristics. An exact method, the ϵ -constraint

method, is also developed in this section in order to analyze the bi-objective performance of the heuristics. Results of experiments on the two methods for both single- and bi-objective versions are presented in Section 4.4. For the former, a comparison is done with the results obtained using the proposed mathematical formulations. For the latter, the heuristic results are compared with the ones from the bi-objective ϵ -constraint method.

Chapter 5 is dedicated to the application in real context: the ODS. Its description is divided into three parts. The first corresponds to the requirements analysis; the second presents the architecture of the system, *i.e.* the Human Computer Interaction (HCI), the database and the network solver; the third raises a number of known gaps in terms of technology. Screenshots are given in order to illustrate the system's look-and-feel and the usage of the system is explained by scenarios.

Finally, Chapter 6 concludes this thesis with a summary of contributions and proposals of future works.

1.2 List of publications

Works of this thesis are progressively carried out to publications listed below. Especially, the software development of ODS belongs to the project TOAST (Tactical Optimization Strategies to Adapt Urban Transportation Networks), one of the laureates of the *Prix Européen d'Innovation Le Monde - Smart Cities* 2017. As an innovation idea, ODS participates as well to the TRAVISION 2016 student contest and overall ranked 13th out of 107 submissions from Europe.

1.2.1 Submitted articles

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Model and methods to address urban road network problems with disruptions*. Submitted to *International Transactions in Operational Research* (to appear). 2018.
- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Bi-objective methods for road networks problems with disruptions*. Submitted to *Annals of Operations Research* (to appear). 2018.

1.2.2 International conference with indexed acts

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Methods for solving road network problems with disruptions*. *Electronic Notes in Discrete Mathematics*, Volume 64. pp. 175-184, 2018.

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *A bi-objective model to address disruptions on unidirectional road networks*, in: *Proceedings of the 8th IFAC Conference on Manufacturing Modelling, Management and Control (MIM2016)*, 12 (49), pp. 1620-1625, Troyes, France, 2016.

1.2.3 International conference with extended abstracts

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Heuristiques pour le problème de reconfiguration des réseaux urbains suite à des interruptions routières*, in: *Actes du 19ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*. 2p. Lorient, France, 2018.
- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Disruptions management in multidirectional road networks*, in: *Proceedings of the 9th Triennial Symposium on Transportation Analysis (TRISTAN16)*, 4p. Oranjestad, Aruba, 2016.

1.2.4 International conference abstracts

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Managing disruptions in urban road networks for real contexts*, in: *6th INFORMS Transportation Science and Logistics Society Workshop*, Hong Kong, China, 2018.
- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Managing predictable and unpredictable disruptions on road networks*, in: *Int'l Conference on Intelligent Transportation Engineering and Smart City (ITESC 2017)*, Guilin, China, 2017.
- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Heuristics for the bi-objective Unidirectional Road Network Design Problem with Disruptions*, in: *Annual workshop of the EURO working group on Vehicle Routing and Logistics optimization (VeRoLog)*, p140. Nantes, France, 2016.

1.2.5 Others

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Optimal traffic Deviation System*. TRAVISIONS (Academic Student Competition), 23, Varsovie, Pologne, 2016.

Chapter 2

State of the art

Road networks have been intensively investigated in various domain over years. This chapter is dedicated to the literature review closely related to the problems addressed in this thesis in terms of graph theory, network design and real applications. According to the definition of RND, the strong connectivity and the arc reversal are two relevant graph features. Theoretical studies have been carried out on the SNOP, which are reviewed in Section 2.1. In terms of applications, NDP is also strongly related to RND. Therefore, other related contributions are surveyed in Section 2.2.

2.1 Related researches on graph theory

In the field of graph theory, there are several relevant topics for RND, such as the strong connectivity, orientation problem and reversal of arc direction. In the seminal work of Robbins (1939), the theorem stating the condition for a simple graph to admit a strong orientation is first presented. Motivated by the traffic control problem for modern cities, an orientable graph is defined as a graph which admits strong orientations. The orientability of a graph refers to the fact that, if any of its edges can be safely removed without breaking its connectedness, there will exist a strong orientation of the graph. Besides, an edge whose removal makes the resulting graph no longer connected corresponds to a bridge of the graph. Thus, only bridgeless graphs admit strong orientations. This theorem is commonly considered as the first key-stone for the strong orientation problem which has been studying since last decades.

Furthermore, multiple orientations can exist on an orientable graph but there should be metrics by which the orientations can be compared. To this extent, Chvátal & Thomassen (1978) propose a thorough study on the distance metric for strong orientations, which was not taken into account in earlier literature and which can be problematic when converting

a two-way network into a one-way network since some original paths may involve longer detours. A lower-bound, which is still the best one till now, is proposed for the radius of orientations and a classification of orientation problems with different distance constraints is presented. By limiting the radius (resp. diameter) of strong orientation on a given undirected graph, the authors prove that the problem becomes very difficult to solve and belongs to the class of NP-hard problems.

Also inspired by Robbins (1939), Boesch & Tindell (1980) extend the strong orientation problem to mixed multigraphs. The term “mixed” specifically indicates that undirected edges and directed arcs can co-exist on the given graph. Thus, a network containing both one-way and two-way streets can be modeled (an initial strong orientation exists). Strictly speaking, it also differs from multigraphs, consisting only of undirected edges, and multidigraphs, including only directed arcs. Turning a mixed multigraph into a multigraph simply involves undirecting all arcs. The authors propose and prove that a strong orientation of a mixed multigraph exists *iff* there is no bridges on its undirected multigraph. In a real context, this corresponds to a road segment whose orientation cannot be changed since it would leave some locations unreachable and isolated on the network.

Chung *et al.* (1985) is another extension of Robbins (1939) combining the study of Chvátal & Thomassen (1978) on distance and the study of Boesch & Tindell (1980) on mixed multigraphs. The major contributions are adapting the orientation radius lower-bound of Boesch & Tindell (1980) on mixed multigraphs and proposing a linear-time algorithm to check the orientability of a given graph. Authors prove two Lemmas following the conditions of the orientability proposed by Boesch & Tindell (1980). The first states that orienting an undirected edge following the direction of the cycle it belongs to does not violate the orientability of the given graph. The second shows that there should be at least one outgoing directed arc and one incoming directed arc between any two connected components. Based on these Lemmas, the orientability check algorithm is designed by: (i) extending a DFS (Tarjan (1972)) to orient the given graph and (ii) reachability checks on all vertices. Note that this algorithm is an adaptation of Tarjan’s Strongly Connected Components Algorithm developed in Tarjan (1972). Besides, using a DFS algorithm to orient simple graphs is also proposed by Roberts (1978).

A recent work by Conte *et al.* (2016) can be noticed as it has a considerable correlation to RND. It proposes an algorithm for enumerating all strong orientations in $\mathcal{O}(m)$ amortized time each for a given mixed multigraph with respectively m edges. Authors defined a specific graph transformation technique by merging a directed cycle to an auxiliary node such that undirected self-loops are formed. A strong orientation of the rest of the graph can be extended by assigning a direction to each self-loop to obtain a strong orientation of the entire graph.

Next, following the theorems of Robbins (1939) and Boesch & Tindell (1980), the *one-way cut* and *forcing cut* are defined depending on the status of an eventual bridge (whether or not directed towards the component such that the resulting graph is not strongly connected). By making use of the linear time algorithm of Italiano *et al.* (2012) for finding strong bridges, the algorithm proposed in this work orients all these bridges in a way that the rest of the graph is orientable. Then, the algorithm recurs with removal of an arbitrary edge and outputs every strong orientation found during the procedure.

The question of reversing the orientation of an arc has been investigated in graph theory, mainly to investigate the impact on some graph invariants. For instance, Ádám (1963) introduced a conjecture on the existence of an arc whose reversal decreases the number of elementary cycles in a digraph by at least one cycle. The conjecture has been validated on some class of graphs, for example, digraphs containing a 2-arc cycle. However, counterexamples are also found on some classes of graphs, by Jirásek (1987), Thomassen (1987) and Jirásek (2005). According to these works, the conjecture remains open on simple digraphs. More recently, Bogdanowicz (2011) shows that the reversal of any arc in a balanced directed loopless multigraph decreases the number of non-necessarily elementary cycles.

The impact of network disruptions on the Shortest Path Problem (SPP) in terms of travel time is addressed by Sever *et al.* (2013). Each arc is associated with a disruption probability. Thus the disruption state of each vulnerable arc is modeled by a random variable. A generic framework is proposed using a dynamic programming approach to evaluate both online and offline routing policies. In this Markov decision process, online information on disruption states is available or not. This investigation provides valuable insights for RND, even if our problem is more oriented towards a deterministic context in which disruptions are pre-defined in advance.

2.2 Network design and applications

As a problem based on the reorganization of urban road networks, RND is highly related to NDP. Studies are thus reviewed in this section in theoretical and application's points of view.

2.2.1 Theoretical studies and surveys

The work of Magnanti & Wong (1984) is considered as a key entry point to the NDP. A three-level classification of NDP is proposed: strategic, tactical and operational, which correspond respectively to investment decision-making (*i.e.* road construction or expansion), link modification (especially orientation problems) and facility location problems

(like traffic signal setting). Authors give a survey on many network design models specialized in various problems, either discrete or continuous. A generic model is developed to cover most of discrete network design problems with some specializations and variations. Optimization-based methods for solving NDP are also mentioned, such as Benders Decomposition, Branch and Bound and Linear Programming. Complexity, limitation in computations and heuristics for solving NDP are also presented, stating that medium-sized networks (50 nodes and 150 arcs) can be efficiently solved at that time by current heuristics. Nevertheless, handling larger instances still requires improvements on the proposed methods.

Boyce & Janson (1980) investigate a Discrete NDP minimizing total travel cost with respect to a given budget for network changes, especially operations concerning the capacity. Authors emphasize the use of discrete functions that appropriately satisfy transportation network design. Formulations are separately given for the objective function, the trip distribution problem and the assignment problem. Several problems combining the aforementioned formulations are presented. Moreover, computational results obtained by experiments of a simple 10-node network are shown and perspectives on a number of open questions are given for further studies.

Yang & Bell (1998) study a generic bi-level framework for modeling transportation planning and investment decision-making problems under a transportation network management system. Authors consider the NDP on an economic aspect, especially budget constraints on financial resources (investment). The goal of the lower level is to carry out the network flow pattern holding the equilibrium between traffic flow and the level of service of the network (demand-performance equilibrium) influenced by a given investment. It is passed as a decision variable to the upper-level model. Authors review some equilibrium models such as User Equilibrium and the logit-based Stochastic User Equilibrium. For the upper-level, the profit from investment is maximized in terms of social welfare for transport planners. It can handle different types of decision variables, either discrete, continuous or mixed.

The Generalized NDP is formally defined and investigated by Feremans *et al.* (2003), describing the generalization of NDP by vertex clustering and the optimization-based solution by an optimal subgraph subject to a set of constraints. Authors consider that a graph is divided into several groups of vertices denoted “clusters”. According to their statement, problems can be divided into spanning problems whose feasible solutions should cover all vertex subsets and non-spanning problems whose solutions do not involve all clusters of vertices. Complexity of different example problems are listed, showing that the generalization makes these problems harder than their classical versions in most of the cases. Authors then survey some previously studied generalized graph problems which can prove the concept of

the Generalized NDP, such as the generalized minimum spanning tree problem (by Myung *et al.* (1995)), the generalized traveling salesman problem (by Henry-Labordere (1969), Saskena (1970) and Srivastava *et al.* (1969)) and the generalized shortest path problem (by Li *et al.* (1995)).

More reviews on theoretical network design can be found in L. & Friesz (1983), Boyce (1984) and Friesz (1985), including models, methods and case studies for network equilibrium, design or aggregation problems. Furthermore, diving deeper in the context of urban transportation system, readers are referred to Cascetta (2001) and Farahani *et al.* (2013) for a summary of several mathematical models, such as traffic assignment to transportation networks, supply design models and transportation systems engineering for planning and evaluation. In this context, problems can be specialized in Urban Transportation Network Design Problem, a refined term interpreting transportation planning decision-making problems when traffic demand grows, by applying network changes and facility location setting under some given constraints. Other terms include Road Network Design Problem and (Public) Transit Network Design Problem.

SNOP (Santos *et al.* (2013)) is a problem highly related to RND. It aims at setting a strong orientation to an undirected graph $G_0 = (V, E)$. That is to say assigning a direction for every undirected edge $e \in E$ such that the resulting digraph should be strongly connected. The total distance of a given set of origin-destination trips ($R = \{(o_1, d_1), (o_2, d_2), \dots, (o_n, d_n)\} | (o_n, d_n) \in V \times V, 1 \leq n \leq |V|\}$) is minimized. Differing from SNOP, RND searches for re-orienting a given oriented digraph (the original network) in which a set of blocked arcs are removed, such that the graph remains strongly connected. If an arc is oriented in the opposite direction of the original one, it is considered as a reversal.

2.2.2 Applications and case studies on NDP

Miandoabchi & Farahani (2011) study the lane allocation problem, considering the reserve capacity concept (ability to allow additional flow). They propose a bi-level model following the widely-used framework of Yang & Bell (1998), in order to solve the lane addition and orientation problem with reserve capacity. In particular, lane allocation was considered in two variations according to the layout symmetry constraint on both sides of a two-way road. Two metaheuristics have been proposed: a Hybrid Genetic Algorithm and an Evolutionary Simulated Annealing. Experiments using several fictive networks indicate that the first one has a better performance.

Some works handle NDP in real contexts in different levels. For instance, Cantarella *et al.* (2006) investigate methods to deal with the UTNDP addressing the network topology and the signal settings, in order to reorganize the traffic. The signal settings are modeled using the

Method of Successive Averages on the Flow Averaging which had been proposed by Powell & Sheffi (1982). The authors propose heuristics, metaheuristics and hybrid methods: two Hill Climbing methods, a Genetic Algorithm (GA), a Simulated Annealing and a Tabu Search (TS). The first hybrid method combines the GA and the TS, while the second couples the TS and a Path Relinking procedure. Experiments are done with real networks of three small Italian towns: Villa San Giovanni with 45 km, 88 nodes and 225 arcs; Melito Porto Salvo with 23 km, 96 nodes and 189 arcs; and Barcellona Pozzo di Gotto with 81 km, 176 nodes and 510 arcs. Gallo *et al.* (2012) study the problem of improving the network infrastructure considering the investment cost in order to increase the flow capacity and the free-flow speed. The authors propose a metaheuristic based on the Scatter Search of Glover *et al.* (2003), where the assignment problem is solved by the Ant Colony Optimization method of Dorigo & Stützle (2004). Computational experiments are done with 8,700 km of the Campania regional road network composed of 262 road nodes, 764 road links.

It is important to mention that the works of Cantarella *et al.* (2006) and Gallo *et al.* (2012) deal with long-term issues of NDP. Thus, in this thesis, relevant contributions consist of handling both short-term and long-term decisions to improve urban network infrastructures. Especially, the core of ODS is able to manage disruptions for both short- and long-term purposes with the orientation problem in multigraphs. To the best of our knowledge, this problem has not been investigated in the literature either in theoretical and practical issues, or in the real context. The aforementioned theoretical and practical works will provide an initial base to build our methods for RND and to develop ODS.

Chapter 3

Formal definitions and mathematical formulations

In this chapter, mathematical formulations are presented for RND. First, the notations used in definitions and models are listed and explained in Section 3.1. Details and common definitions of RND are given in Section 3.2. Then in Sections 3.3 and 3.4, URND and MRND are formally defined and formulated by bi-objective mathematical models with respective examples illustrating several key concepts. Particularly, the latter section includes also the definition of the 2-directed multigraph, a specific multigraph used in MRND formulations. Finally, Section 3.5 shows computational experiments of the aforementioned formulations and provides analysis of their numerical results.

3.1 Notations

Notations used in this chapter as well as in the rest of this document are explained in this section. Especially, different types of graph are presented.

Let $G_0 = (V, E)$ be an undirected connected loopless graph with V the set of vertices and E the collection of edges. An unweighted edge is denoted by a pair of vertices i, j where $i, j \in V \times V$ such that $i, j = j, i$. In addition, a weighted edge is an edge (i, j) with a cost assigned denoted by γ_{ij} .

In addition, let $G_1 = (N_1, A_1)$ be a connected simple digraph. N_1 is the set of nodes and A_1 is the set of distinct arcs. An arc has an orientation denoted by an ordered pair (i, j) where $i, j \in N_1 \times N_1$, *i.e.* $(i, j) \neq (j, i)$. Note that any arc $a \in A_1$ is unique. Similarly to the definition of a weighted edge, the arc (i, j) can also be associated with a cost γ_{ij} . Suppose $B_1 \subset A_1$ be a set of disruptions containing arcs that can no longer be used for routing. $H_1 = (N_1, A'_1)$ is the underlying graph of G_1 on which blocked arcs are removed,

thus $A'_1 = A_1 \setminus B_1$.

Moreover, let $G_2 = (N_2, A_2)$ be a directed connected multigraph with N_2 the set of nodes and A_2 the collection of arcs. As mentioned previously, there can be more than one arc between a pair of nodes in a multigraph. Thus, an arc $a \in A_2$ does not have to be unique. Therefore, arcs between the same pair of node $i, j \in N_2 \times N_2$ should be associated with an index $k \in \mathbb{N}$ so as to be distinct from each other. Then, the representation of an arc becomes (i, j, k) . Likely to the aforementioned set of blockages $B_1, B_2 \subset A_2$ is the collection of disruptions on G_2 . $H_2 = (N_2, A'_2)$ is then defined as the underlying graph of G_2 that excludes blocked arcs $b \in B$, *i.e.* $A'_2 = A_2 \setminus B$.

3.2 Road networks problems with disruptions and connecting requirements

First, it is important to highlight the relation between the situation in practice and its modeling by means of graph theory. For instance, a road network must have a path allowing the access to any point of the networks. In graph theory, this characteristic relies on the so-called connectedness property for undirected graphs and the strong connectivity property for directed graphs. Moreover, changing the orientation of a lane corresponds to modifying the orientation of the corresponding arc, named here an arc reversal. One may note that arc reversals are likely to disturb users habits. That is why one of the objectives focused in this thesis is to minimize the number of arc reversals. As a consequence, as few modifications as possible are done on the network such that the strong connectivity is guaranteed. Last but not least, a disruption on the road networks is defined as an unavailable edge in undirected graphs or as an unavailable arc on digraphs. Other specific solutions could also be taken into account such as alternated traffic, lanes reduction and transforming a two-way street into two one-way lanes as it have been done in the “*Quai du Comte Henri*” in Troyes. These could be possible extensions to our work.

Roads (road segments) and crossroads (intersections) are the two topological components of a transportation network, which correspond respectively to edges and vertices in graph theory. Thus, modeling networks by graphs has been an obvious and efficient approach. However, roads can be either one-way or two-way, which should be represented differently on graphs in such a way that one-way streets are modeled by directed arcs and two-way streets are displayed by undirected edges. Moreover, a two-way street can even have multiple lanes in both directions and it does not seem evident if such a street is represented by an undirected edge. That is the reason why directed multigraphs are involved. Unlike simple graphs which can have at most one edge (arc) between a pair of vertices, there might be more

than one edge (arc) between two nodes so as to represent lanes of a road. The connecting requirement means the graph has to be strongly connected, *i.e.*, there exists at least one path between every pair of Origin-Destination (O-D) nodes. Since disruptions are treated as arcs removed from the given graph, they can potentially break the strong connectivity.

The main goal of RND is to find orientations of given networks in order for the networks to remain strongly connected in case of disruptions. Two objectives are considered as measures of orientations: (i) minimizing the total travel distance (c_1) and (ii) minimizing the number of arc reversals (c_2). To restore strong connectivity broken by disruptions, the direction of every arc is allowed to be reversed. However, an arc reversal may involve a lot of operational forces such as setting notice panels and keeping network users informed. It also perturbs drivers' habits which may further cause congestion in concerned areas and even more disruptions. Thus, it should be applied only when necessary and minimizing arc reversals is justified as mentioned at the beginning of this section.

Depending on the type of networks addressed, either unidirectional or multidirectional, two problems (URND and MRND) are then proposed and defined in the following two sections with their bi-objective versions being respectively bi-URND and bi-MRND.

3.3 Unidirectional road networks problems

URND particularly addresses historical downtown networks where roads are often one-way due to limited width. This is the case of the city center of Troyes and many other French cities. Layouts are often fixed in order to lead traffic flows in designed paths. However, finding bypasses is almost impossible without modifying the configuration of the network since blocking a one-way street can make some destinations unreachable, which must be avoided for network users. Furthermore, if a non-informed driver just arrives in front of a blocked road, he is willing to find a detour. But he cannot simply take a U-turn to get back to the previously passed intersection and should sometimes perform a long trip over a certain area. This increases the travel cost of some network users and often leads to traffic difficulties. Hence, managing disruptions can be very complicated from a network manager's point of view. The previously defined graph G_1 is used to model URND with nodes representing road intersections and arcs standing for road segments. Setting strong orientation minimizing all distances on undirected graphs is an NP-hard problem, formally proved by Burkard *et al.* (1999). In addition, the problem with the number of reversals minimization is polynomial Bang-Jensen & Gutin (2008). Consequently, URND is NP-hard with c_1 objective and it is polynomial with c_2 objective. Thus, NP-hardness can be deduced for the bi-objective version, bi-URND.

Figure 3.1 depicts an example of different orientations on an unidirectional road network with nine nodes 0, 1, 2, 3, 4, 5, 6, 7, 8 and one blocked arc (4, 5). Suppose G_1 models this network with the blockage illustrated in Figure 3.1-(a). It is no longer strongly connected due to the blocked arc. Figure 3.1-(b) is an infeasible orientation since the node 5 cannot be reached from other nodes, thus it is not the desired strong orientation. Figures 3.1-(c) and 3.1-(d) are two possible orientations. More precisely, a detour of the blockage (4, 5) can go through (4, 7), (7, 8) and (8, 5) in the orientation of Figure 3.1-(c), while another tour should be applied in Figure 3.1-(d) via (4, 1), (1, 2) and (2, 5).

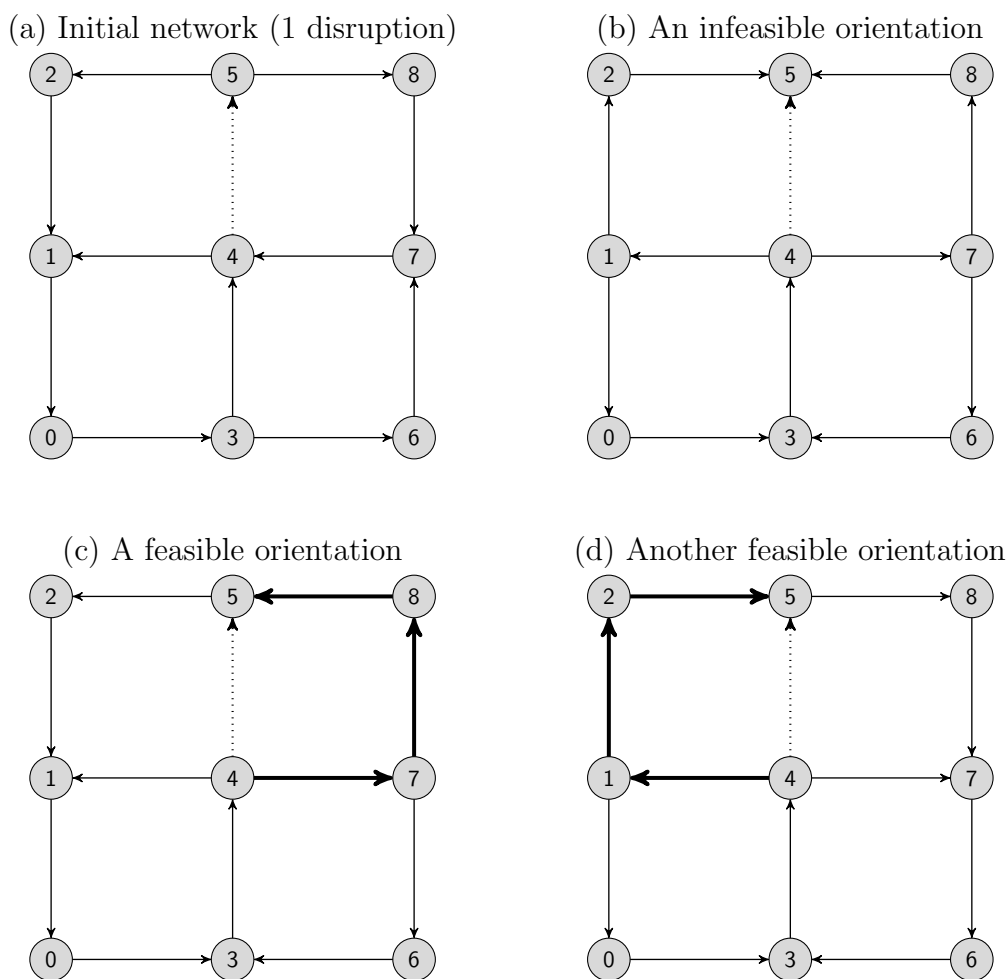


Figure 3.1: An example network with one blockage addressed by URND

Given $G_1 = (V_1, A_1)$, let $P = \{(o, d) | o, d \in V_1, o \neq d\}$ be the set of all O-D pairs in G_1 and let the B_1 be the list of the p blocked arcs (disruptions). All arcs $(i, j) \in B$ are preprocessed and removed from G_1 , leading to the modified graph $H_1 = (N_1, A'_1)$, as described in Section 3.1. Since the strong connectivity is the major concern in URND, it is ensured by forcing a unit flow for each O-D pair $(o, d) \in P$. The direction of arcs in A'_1 are allowed to be reversed. The two criteria to be minimized for a solution s are the total distance $c_1(s)$ and the number

of reversed arcs $c_2(s)$.

The decision variables $x_{ij} \in \{0, 1\}$ is used to represent the reversal state of the arc $(i, j) \in A'_1$. If it is reversed, $x_{ij} = 1$; $x_{ij} = 0$ otherwise. The flow of the path from o to d on the arc (i, j) is modeled by variables $f_{ij}^{od} \in \mathbb{R}$, taking its reversal state into account. For instance, assuming (i, j) is used to send flow from o to d , $f_{ij}^{od} > 0$ if the arc keeps its original direction and $f_{ij}^{od} < 0$ if the arc is reversed. Variables $y_{ij}^{od} \geq 0$ keep the amount of flow from o to d going through the arc (i, j) independently of its orientation.

Proposition 3.1. *The variables f_{ij}^{od} as defined above are valid for the flow conservation constraints in URND.*

The proposed mathematical model is then as follows:

$$\min c_1 = \sum_{(o,d) \in P} \sum_{(i,j) \in A'_1} \gamma_{ij} y_{ij}^{od} \quad (3.1)$$

$$\min c_2 = \sum_{(i,j) \in A'_1} x_{ij} \quad s.t. \quad (3.2)$$

$$\sum_{l:(l,o) \in A'_1} f_{lo}^{od} - \sum_{l:(o,l) \in A'_1} f_{ol}^{od} = -1 \quad \forall (o, d) \in P \quad (3.3)$$

$$\sum_{l:(l,i) \in A'_1} f_{li}^{od} - \sum_{l:(i,l) \in A'_1} f_{il}^{od} = 0 \quad \forall (o, d) \in P, \forall i \notin \{o, d\} \quad (3.4)$$

$$-y_{ij}^{od} \leq f_{ij}^{od} \leq y_{ij}^{od} \quad \forall (o, d) \in P, \forall (i, j) \in A'_1 \quad (3.5)$$

$$-x_{ij} \leq f_{ij}^{od} \leq 1 - x_{ij} \quad \forall (o, d) \in P, \forall (i, j) \in A'_1 \quad (3.6)$$

$$x_{ij} \leq \sum_{(o,d) \in P} y_{ij}^{od} \quad \forall (i, j) \in A'_1 \quad (3.7)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A'_1 \quad (3.8)$$

$$f_{ij}^{od} \in \mathbb{R} \quad \forall (o, d) \in P, \forall (i, j) \in A'_1 \quad (3.9)$$

$$y_{ij}^{od} \geq 0 \quad \forall (o, d) \in P, \forall (i, j) \in A'_1 \quad (3.10)$$

The first objective function (3.1) aims at minimizing the total travel cost and the second objective function (3.2) minimizes the number of arc reversals. Constraints (3.3) and (3.4) are the flow conservation constraints using variables f_{ij}^{od} . Inequalities (3.5) link variables f_{ij}^{od} with variables y_{ij}^{od} in order to set the amount of flow. Constraints (3.6) link f_{ij}^{od} with x_{ij} for f_{ij}^{od} variables to take arcs' orientation into account. In addition, arcs not carrying flows do not need to be reversed, as set in constraints (3.7). Moreover, variables are defined from (3.8) to (3.10). Besides, this model contains $(2 \times |P| + 1) \times |A'_1|$ variables and $(|P|^2 + (4 \times |A'_1| - 1) \times |P| + 2 \times |A'_1|)$ constraints.

Note that, according to a recent improvement of the model, the relation among the variables x_{ij} , f_{ij}^{od} and y_{ij}^{od} can be enhanced by the constraints (3.11). It can be linearized into the inequalities (3.12) along with the constraints (3.5). The constraints (3.6) can thus be replaced by this definition.

$$f_{ij}^{od} = f_{ij}^{od} \times (1 - 2x_{ij}) \quad \forall (o, d) \in P, \forall (i, j) \in A'_1 \quad (3.11)$$

$$-2x_{ij} + y_{ij}^{od} \leq f_{ij}^{od} \leq 2 - 2x_{ij} - y_{ij}^{od} \quad \forall (o, d) \in P, \forall (i, j) \in A'_1 \quad (3.12)$$

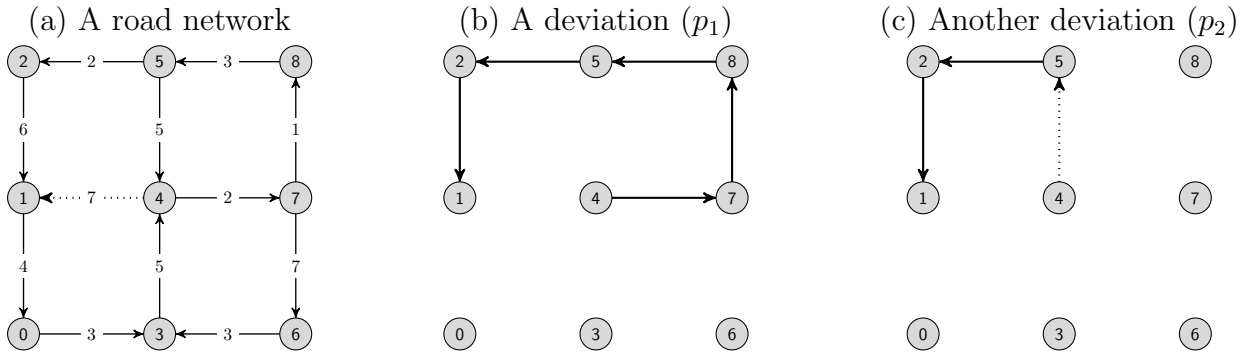
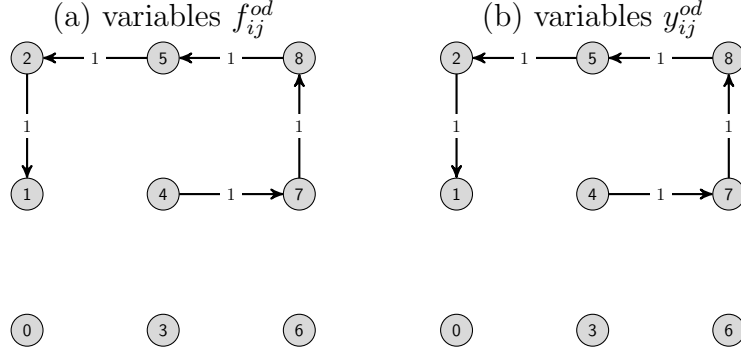
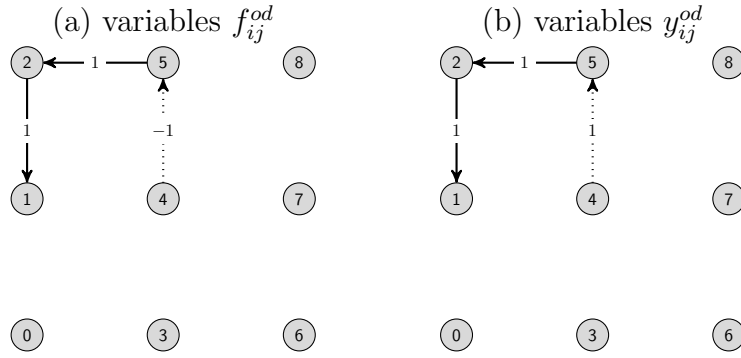


Figure 3.2: Modeling a deviation.

Taking the same orientation of the network of Figure 3.1, Figure 3.2-(a) illustrates the situation in which deviations for the blockage (4, 1) have to be found. The cost γ_{ij} is displayed for each arc $(i, j) \in H_1$. To route traffic flows from node 4 to node 1, the (unique) shortest path without any arc reversals is $p_1 = (4, 7, 8, 5, 2, 1)$ with a travel cost of 14, see Figure 3.2-(b). When arcs are allowed to be reversed, the shortest path is $p_2 = (4, 5, 2, 1)$ with cost 13, as shown in Figure 3.2-(c). The original arc (5, 4) is reversed, hence turned to be (4, 5). Thus, $c_1(p_1) = 14$, $c_2(p_1) = 0$, $c_1(p_2) = 13$, $c_2(p_2) = 1$. Figures 3.3 and 3.4 report the respective values for variables f_{ij}^{od} and y_{ij}^{od} for the 2 solutions presented in Figures 3.2-(b) and 3.2-(c), respectively. In this context, the variables $f_{ij}^{4,1}$ and $y_{ij}^{4,1}$ are null for every arc


 Figure 3.3: Values for flow-based variables f and y for p_1

 Figure 3.4: Values for flow-based variables f and y for p_2

$(i, j) \in A'_1$ without any flow, *i.e.* arcs not used in the path from 4 to 1. The difference between f and y only occurs for the flow on the arc $(5, 4)$ in the solution with arc reversal: $f_{5,4}^{4,1} = -1$ while $y_{5,4}^{4,1} = 1$.

Another simpler model can also be defined with only 2 sets of variables. It relies on the graph $G'_1 = (N'_1, A''_1)$ with $N'_1 = N_1$ and $A''_1 = A'_1 \cup \overline{A'_1}$, *i.e.* the set of all possible arc orientations. The set of arc reversals is extended to A''_1 : $x_{ij} \in \{0, 1\} \forall (i, j) \in A''_1$. The variables $f_{ij}^{od} \geq 0$ store the flow on the arc $(i, j) \in A''_1$ from the origin o to the destination d . Then, the alternate model is as follows:

$$\min c_1 = \sum_{(o,d) \in P} \sum_{(i,j) \in A''_1} \gamma_{ij} f_{ij}^{od} \quad (3.13)$$

$$\min c_2 = \sum_{(i,j) \in A''_1} x_{ij} \quad s.t. \quad (3.14)$$

$$\sum_{l:(l,i) \in A_1''} f_{li}^{od} - \sum_{l:(i,l) \in A_1''} f_{il}^{od} = \begin{cases} -1 & \text{if } i = o \\ 1 & \text{if } i = d \\ 0 & \text{otherwise} \end{cases} \quad \forall (o, d) \in P \quad (3.15)$$

$$f_{ij}^{od} \leq 1 - x_{ij} \quad \forall (o, d) \in P, \forall (i, j) \in A_1'' \quad (3.16)$$

$$x_{ij} + x_{ji} = 1 \quad \forall (i, j) \in A_1' \quad (3.17)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A_1'' \quad (3.18)$$

$$f_{ij}^{od} \geq 0 \quad \forall (o, d) \in P, \forall (i, j) \in A_1'' \quad (3.19)$$

3.4 Multidirectional road networks problems

MRND is a more general problem that addresses road networks closer to real situations where one-way and two-way streets coexist. Layouts can be diverse and there can also exist one-way streets with multiple lanes in one direction. The strong connectivity is easier to ensure in MRND than in URND since two-way streets can be used as a bi-directional bridge to connect two strongly connected components. Let the aforementioned multigraph $G_2 = (N_2, A_2)$ be used in the MRND model. Especially, the collection of arcs A_2 can contain multiple arcs corresponding to lanes of two-way streets. One-way streets can still belong to G_2 , which means that an arc $a = (n_3, n_4)$ ($a \in A'$, $(n_3, n_4) \in N_2 \times N_2$) can also be the unique instance between the nodes n_3 and n_4 . Similar to the complexity of URND (resp. bi-URND) presented in the previous section, MRND with c_1 objective is NP-hard while it is polynomial for c_2 . Therefore, bi-MRND is NP-hard as well.

Illustrative orientations are given in Figure 3.5. The network contains nine nodes $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ and a single disruption $(4, 1)$. Suppose G_2 models this network with the disruption as depicted in Figure 3.5-(a). Figure 3.5-(b) presents an infeasible solution in which the strong connectivity is broken for nodes 0, 1 and 2. Two feasible orientations are shown in Figures 3.5-(c) and 3.5-(d). Bypasses of the blocked arc $(4, 1)$ can be done by the path $(4, 3)$, $(3, 0)$ and $(0, 1)$ in Figure 3.5-(c) and by the path $(4, 5)$, $(5, 2)$ and $(2, 1)$ in Figure 3.5-(d).

The URND formulation is generalized to handle multigraphs. A graph reduction is first proposed to simplify multigraphs to 2-directed multigraphs defined as follows:

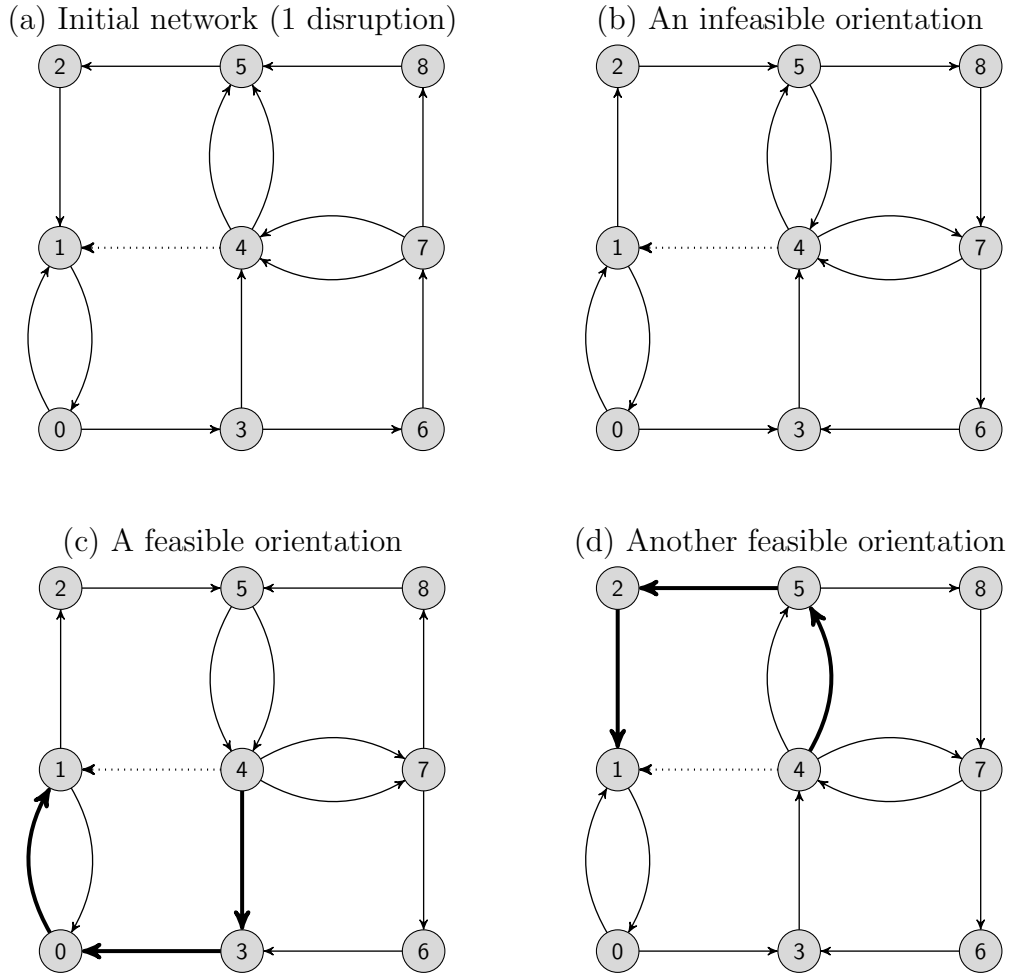


Figure 3.5: An example network with one blockage addressed by MRND

Definition 3.1. A 2-directed multigraph is a directed multigraph with at most two arcs between any pair of nodes.

Consequently, the following property holds for MRND.

Property 3.1. Any instance of MRND can be represented by a 2-directed multigraph.

Given two nodes $i, j \in N_2 \times N_2$ in the multigraph $G_2 = (N_2, A_2)$, assume $[l_1, l_2] \in \mathbb{N}^+ \times \mathbb{N}^+$ be the pair of two respective numbers of arcs in the two directions $i \rightarrow j$ and $j \rightarrow i$. The combination $[1, 0]$ respects the definition of a 2-directed multigraph since there is a single arc $(i, j, 0)$. On the other hand, $[0, 1]$ is also valid. Any $[l, 0]$ with $l > 1$ can be reduced to $[2, 0]$ with two arcs: $(i, j, 0)$ and $(i, j, 1)$. Symmetrically, any $[0, l]$ with $l > 1$ can be reduced to $[0, 2]$ with two arcs $(j, i, 0)$ and $(j, i, 1)$. More generally, a pair $\{i, j\}$ with $[l_1, l_2]$ where $l_1 > 1$ and $l_2 > 1$ can be reduced to $[1, 1]$ with an arc $(i, j, 0)$ and another arc $(j, i, 0)$. These reductions do not impact the computation of c_1 criterion. They also support the optimal values for the c_2 criterion.

Thus, G_2 can be reduced to a 2-directed multigraph and possible values for indices k of an arc $(i, j, k) \in A_2$ can be restricted to 2, *i.e.* $k \in \{0, 1\}$. Extended to the model in Section 3.3, decision variables are additionally indexed on k , *i.e.* $x_{ijk} \in \{0, 1\}$ for the reversal state of an arc (i, j, k) , $f_{ijk}^{od} \in \mathbb{R}$ for the flow passed on the arc (i, j, k) on the path from o to d and $y_{ij}^{od} \geq 0$ for the absolute amount of flow related to f .

Proposition 3.2. *The variables f_{ijk}^{od} are valid for the flow conservation constraints in MRND.*

Figures 3.6 and 3.7 depict the reduction cases described above. In the former, every graph in columns “a” and “b” can be reduced to the ones in cells “1-a” and “1-b”, respectively. In the latter, reductions can be made from right to left in every row and from bottom to top in every column, and every graph can be reduced to the one in the cell “1-c”. Note that these figures illustrate only graphs with up to 3 arcs in a direction. Graphs with more than 3 arcs in one direction between a pair of nodes also admit underlying reduced 2-directed multigraphs.

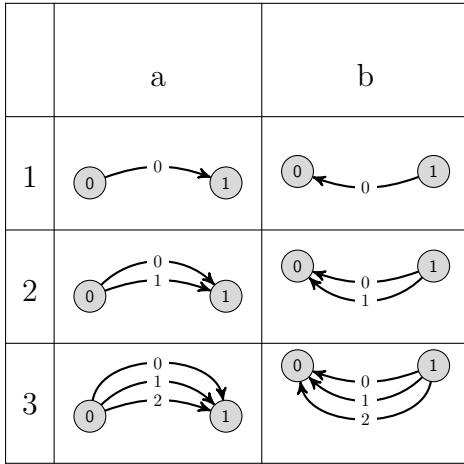


Figure 3.6: 2-directed multigraphs reductions: 1 direction

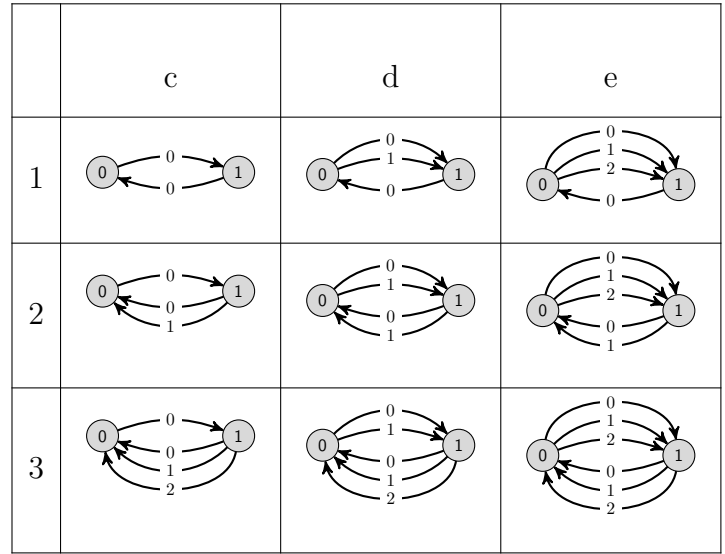


Figure 3.7: 2-directed multigraphs reductions: 2 directions

The proposed formulation for MRND is then as follows:

$$\min c_1 = \sum_{(o,d) \in P} \sum_{(i,j,k) \in A'_2} \gamma_{ijk} y_{ijk}^{od} \quad (3.20)$$

$$\min c_2 = \sum_{(i,j,k) \in A'_2} x_{ijk} \quad s.t. \quad (3.21)$$

$$\sum_{l:(l,o,k) \in A'_2} f_{lok}^{od} - \sum_{l:(o,l,k) \in A'_2} f_{olk}^{od} = -1 \quad \forall (o, d) \in P \quad (3.22)$$

$$\sum_{l:(l,i,k) \in A'_2} f_{lik}^{od} - \sum_{l:(i,l,k) \in A'_2} f_{ilk}^{od} = 0 \quad \forall (o, d) \in P, \forall i \notin \{o, d\} \quad (3.23)$$

$$-y_{ijk}^{od} \leq f_{ijk}^{od} \leq y_{ijk}^{od} \quad \forall (o, d) \in P, \forall (i, j, k) \in A'_2 \quad (3.24)$$

$$-x_{ijk} \leq f_{ijk}^{od} \leq 1 - x_{ijk} \quad \forall (o, d) \in P, \forall (i, j, k) \in A'_2 \quad (3.25)$$

$$x_{ijk} \leq \sum_{(o,d) \in P} y_{ijk}^{od} \quad \forall (i, j, k) \in A' \quad (3.26)$$

$$x_{ijk} \in \{0, 1\} \quad \forall (i, j, k) \in A' \quad (3.27)$$

$$f_{ijk}^{od} \in \mathbb{R} \quad \forall (o, d) \in P, \forall (i, j, k) \in A'_2 \quad (3.28)$$

$$y_{ijk}^{od} \geq 0 \quad \forall (o, d) \in P, \forall (i, j, k) \in A'_2 \quad (3.29)$$

The two objective c_1 and c_2 are defined by (3.20) and (3.21). The flow conservation constraints are formulated in (3.22) and (3.23). In addition, inequalities (3.24) and (3.25) model the relations between f and y and between x and f . Reversals of unused arcs are still forbidden as in the constraints (3.26). Finally, definition of variables are formulated in (3.27), (3.28) and (3.29). Moreover, this model consists of $(2 \times |P| + 1) \times |A'_2|$ variables and $(|P|^2 + (4 \times |A'_2| - 1) \times |P| + 2 \times |A'_2|)$ constraints.

3.5 Computational experiments and numerical results

Two major goals of the computational experiments on the mathematical models are the following: (i) to know their computational performance and limits; (ii) to identify the gaps to find optimal solutions, for instance, in terms of linear relaxation and how the branch-and-cut tree progresses towards the optimal solution proof. Some mathematical models lead to a very fast computation but they visit a significant number of nodes in the Branch-and-Bound (B&B) tree, while others require only a few nodes to be visited in the B&B tree in spite of a high running time. According to Gouveia & Magnanti (2003), multiflow models result generally in good linear relaxation. However, the combinatorial aspects as well as the number of variables and constraints do not allow solving very large instances.

Another relevant point concerns the objective functions. Two criteria are considered, but at this point each one is optimized independently of the other, following a single-objective approach. The bi-objective optimization is also considered but it will be presented later on in this manuscript.

Small- and medium-sized graphs with grid-based layout have been generated as instances for testing the proposed mathematical model, including 14 simple graphs and 24 2-directed multigraphs. All arc costs are set to 1. Disruptions are randomly generated, but a control procedure is also involved in order to guarantee that, when these disruptions are removed from an instance, the resulting graph is still orientable. For multigraph instances, multiple arcs are randomly added so as to stay within the 2-directed multigraph context. Instance characteristics are presented in Tables 3.1 and 3.2 for simple graph instances and multigraph instances, respectively. In these tables, instances are classified into small-sized instances (3x3 and 4x4) and medium-sized instances (5x5 and 6x6), reported respectively in the column “Group”. All instance names are reported in column “Name”. Columns “ n ”, “ m ” and “ p ” represent respectively the number of nodes, arcs and disruptions. In addition, the column “ $M\%$ ” reports the percentage of multiple arcs in multigraphs, as additional information in Table 3.2. Experiments are designed to solve all the generated instances within the proposed models for URND and MRND by the commercial solver CPLEX 12.6. They are done on a machine with an Intel CORE i7-4710MQ at 2.50 GHz, with 4 cores (8 threads) and 16 GB RAM under Windows.

Results are reported in Tables 3.3 and 3.4 for URND and MRND, respectively. In both two tables, instance names, c_1 value, c_2 value, CPLEX processing time in seconds, linear relaxation value and relative gap are presented by columns “Instance”, “ c_1 ”, “ c_2 ”, “t(s)”, “LR” and “Gap(%)”. Depending on the optimization objective (either c_1 or c_2), the columns marked with “*” report the optimal value (*i.e.* “ c_1^* ” or “ c_2^* ”, resp.).

For URND, an optimal value of c_1 sometimes implies a significant number of arc reversals. This is undesirable, even unpractical in the real context, because it involves a lot of changes in the network infrastructure as well as an unaffordable user experience for drivers (for instance the impact on driving habits which may cause troubles and congestion in the areas impacted by reversals). To the opposite, c_2 minimization restricts the number of necessary reversals while the total travel distance of the whole network is left free. This can also be problematic on a real unidirectional network since drivers should sometimes make much longer trips than before to reach their destinations. Traffic loads may also increase on some important paths. Results also indicate that c_1 minimization tends to be very time-consuming when instance becomes larger since the model uses two sets of 4-dimension variables. This can be justified by the sharp growth of the computational time and the globally high value of

relative integrality gap. In addition, 3 instances out of 4 of the medium-sized network with 36 nodes and 60 arcs cannot be solved to optimality in a two-hour limit (only lower bounds are given for these instances). However, c_2 minimization is relatively easier for CPLEX. The processing time increases mildly and all 6x6-* instances can be solved to optimality in a reasonable amount of time. On an instance, CPLEX generally uses less time when more disruptions happen. One possible reason may be that blocked arcs are removed from the given instance before the computation. In other words, there are less arcs to handle.

Regarding MRND, according to the results, it seems that CPLEX solves such problem faster than URND. In addition, respecting our generated instance, the more multiple arcs an instance has, the less time CPLEX uses to solve MRND on it. Based on our proposition of 2-directed multigraph, this is probably because 2 arcs (at most) between a pair of nodes o and d can be turned into bi-directional ($a_1 = (o, d)$ and $a_2 = (d, o)$) that simplifies the paths from o to d and from d to o , while in URND, an arc $a_0 = (o, d)$ cannot be served in the path from d to o . The same situation can also be found in c_2 minimization, that, comparing to URND, CPLEX uses less time to solve multigraphs. Besides, the number of disruptions does not seem to have a significant impact on the objective values and the processing time.

Name	n	m	p
3x3-b1	9	12	1
3x3-b2			2
4x4-b1	16	24	1
4x4-b2			2
4x4-b4			4
4x4-b6			6
5x5-b1	25	40	1
5x5-b2			2
5x5-b4			4
5x5-b6			6
6x6-b1	36	60	1
6x6-b2			2
6x6-b4			4
6x6-b6			6

Table 3.1: Characteristics of generated simple graphs

Name	n	m	$M\%$	p
3x3-b1-25	9	24	15	25
3x3-b1-50			18	50
3x3-b1-75			21	75
3x3-b1-100			24	100
3x3-b2-25	9	24	15	25
3x3-b2-50			18	50
3x3-b2-75			21	75
3x3-b2-100			24	100
4x4-b1-25	16	48	30	25
4x4-b1-50			36	50
4x4-b1-75			42	75
4x4-b1-100			48	100
4x4-b2-25	16	48	30	25
4x4-b2-50			36	50
4x4-b2-75			42	75
4x4-b2-100			48	100
5x5-b1-25	25	100	50	25
5x5-b1-50			60	50
5x5-b1-75			70	75
5x5-b1-100			80	100
5x5-b2-25	25	100	50	25
5x5-b2-50			60	50
5x5-b2-75			70	75
5x5-b2-100			80	100
6x6-b1-25	36	120	75	25
6x6-b1-50			90	50
6x6-b1-75			105	75
6x6-b1-100			120	100
6x6-b2-25	36	120	75	25
6x6-b2-50			90	50
6x6-b2-75			105	75
6x6-b2-100			120	100

Table 3.2: Characteristics of generated multigraphs

Instance	c_1 minimization					c_2 minimization				
	c_1^*	c_2	t(s)	LR	Gap(%)	c_1	c_2^*	t(s)	LR	Gap(%)
3x3-b1	246	2	0.27	212	13.82	248	0	0.06	0	0
3x3-b2	284	2	0.16	264	7.04	284	2	0.05	2	0
4x4-b1	912	22	22.40	786	13.82	960	0	0.44	0	0
4x4-b2	968	5	11.34	836	13.64	1060	0	0.42	0	0
4x4-b4	1392	8	3.99	1216	12.64	1442	2	0.51	2	0
4x4-b6	1304	7	1.72	1088	16.56	1388	6	0.39	6	0
5x5-b1	2638	14	414.62	2292	13.12	3388	0	3.84	0	0
5x5-b2	2748	16	373.96	2404	12.52	3960	0	3.51	0	0
5x5-b4	3032	23	182.74	2636	13.06	3844	1	3.70	1	0
5x5-b6	3116	4	253.86	2672	14.25	3322	2	3.28	2	0
6x6-b1	≤ 6250	-	7200	5506	11.90	7770	2	20.39	2	0
6x6-b2	≤ 6554	-	7200	5632	14.07	7636	1	21.25	1	0
6x6-b4	≤ 6820	-	7200	5956	12.67	7606	1	20.90	1	0
6x6-b6	7006	35	3894.85	6394	8.74	7814	3	18.78	3	0

Table 3.3: CPLEX results of small and medium simple instances (URND)

Instance	c_1 minimization					c_2 minimization				
	c_1^*	c_2	t(s)	LR	Gap(%)	c_1	c_2^*	t(s)	LR	Gap(%)
3x3-b1-25	204	6	0.21	198	2.94	246	0	0.08	0	0.00
3x3-b1-50	186	5	0.13	186	0.00	246	0	0.08	0	0.00
3x3-b1-75	172	6	0.13	172	0.00	242	0	0.09	0	0.00
3x3-b1-100	148	7	0.11	148	0.00	200	0	0.08	0	0.00
3x3-b2-25	250	7	0.19	242	3.20	250	2	0.08	2	0.00
3x3-b2-50	194	3	0.22	190	2.06	208	2	0.06	2	0.00
3x3-b2-75	166	6	0.22	164	1.20	198	1	0.08	1	0.00
3x3-b2-100	152	5	0.09	152	0.00	174	1	0.08	1	0.00
4x4-b1-25	832	5	4.23	768	7.69	862	0	0.23	0	0
4x4-b1-50	764	12	1.78	750	1.83	818	0	0.25	0	0
4x4-b1-75	716	8	1.20	708	1.12	780	0	0.23	0	0
4x4-b1-100	652	7	0.45	652	0	712	0	0.19	0	0
4x4-b2-25	850	12	2.92	804	5.41	972	0	0.25	0	0
4x4-b2-50	764	14	2.00	752	1.57	890	0	0.25	0	0
4x4-b2-75	720	12	0.70	720	0	798	0	0.25	0	0
4x4-b2-100	652	10	0.53	652	0	730	0	0.27	0	0
5x5-b1-25	2406	27	56.69	2270	5.65	2936	0	1.45	0	0
5x5-b1-50	2296	12	12.67	2226	3.05	2814	0	1.36	0	0
5x5-b1-75	2158	19	7.75	2142	0.74	2458	0	1.31	0	0
5x5-b1-100	2024	20	3.65	2024	0	2408	0	1.19	0	0
5x5-b2-25	2450	12	41.09	2318	5.39	2868	0	1.33	0	0
5x5-b2-50	2260	26	23.90	2194	2.92	2676	0	1.37	0	0
5x5-b2-75	2126	16	8.10	2114	0.56	2468	0	1.33	0	0
5x5-b2-100	2036	20	3.81	2036	0	2430	0	1.19	0	0
6x6-b1-25	Out of memory			5466	-	7636	1	26.40	1	0.00
6x6-b1-50				5396	-	6550	2	22.04	2	0.00
6x6-b1-75	5268	41	41.48	5268	0	6344	1	20.48	1	0.00
6x6-b1-100	5060	35	20.22	5060	0	5942	1	16.01	1	0.00
6x6-b2-25	Out of memory			5592	-	7308	1	20.28	1	0.00
6x6-b2-50				5478	-	6986	0	5.76	0	0.00
6x6-b2-75	5322	34	45.12	5322	0	6678	0	17.89	0	0.00
6x6-b2-100	5092	38	21.61	5092	0	6116	0	5.07	0	0.00

Table 3.4: CPLEX results of small and medium multigraph instances (MRND)

Chapter 4

Methods for solving single- and bi-objective URND and MRND

In this chapter, the proposed single- and bi-objective methods are presented. First of all, a DFS-based orientation algorithm is proposed and described in Section 4.1. It is used respectively as the decoding method and the constructive heuristic in BRKGA and ILS. Then, Section 4.2 presents the two approximate methods, BRKGA and ILS, for solving single-objective URND and MRND. In Section 4.3, an ϵ -constraint exact method is proposed to compute the Pareto-optimal fronts for bi-URND and bi-MRND, followed by bi-objective versions of BRKGA and ILS. Since these two proposed metaheuristics have been basically designed for single-objective optimization, an additional NSGA-II algorithm is also implemented, used as a reference method for bi-objective heuristic optimization. It is presented in Subsection 4.3.4. Finally, experiments on all these methods and their numerical results are shown in Section 4.4.

4.1 An orientation algorithm

Given an existing road network in which every link is available for carrying traffic flows, it is by definition strongly connected since every location should be accessible from any other location. However, as described in Chapter 1, disruptions on links can disable some arcs, hence probably turn some locations isolated, *i.e.* the road network may not be strongly connected anymore. This is the reason why a strong orientation algorithm should be developed on an already oriented network in order to set another orientation that makes the given network strongly connected.

In the case of URND, Roberts (1978) proposed an algorithm to build a strong orientation on a simple connected non-oriented graph $G_0 = (V, E)$. It uses the algorithm DFS (Cormen

et al. (2009)) to visit the vertices in V through edges in E . Thus, an edge can be oriented during this process. Assume two vertices $i, j \in N \times N$ and an edge $e = (i, j) \in E$. Suppose the algorithm tries to visit j from i through e . Then, e is oriented from i to j . In this way, every edge in E is oriented and an orientation of G_0 is thus constructed.

Proposition 4.1. *The orientation constructed by applying the algorithm proposed by Roberts (1978) on a simple connected undirected bridgeless graph (orientable simple graph) is strongly connected.*

Readers are referred to Roberts (1976) for the proof.

The extension of this algorithm is proposed in this work. First, it is adapted to 2-directed multigraphs for MRND. Second, the algorithm is parameterized in order to provide different strong orientations depending on the parameter.

For MRND, there can be several edges between a pair of vertices on the underlying undirected multigraph of an instance. However, DFS can visit a vertex only one time. We relax this constraint in such a way that a vertex can always be visited until all its related edges are oriented.

Proposition 4.2. *By using the relaxed visit mechanism, all the edges of a connected undirected bridgeless multigraph (orientable multigraph) are oriented.*

Proof. Given a connected non-oriented graph, the DFS browses all the vertices. By definition, it ends with a final backtrack to the root vertex. Besides, the backtrack in our DFS-based algorithm only occurs when all the edges related to a vertex is oriented. If a non-oriented edge e still exists while the final backtrack, the scan of non-oriented edges on the two ends of e is not completed and the backtrack should not have been triggered. This leads to a contradiction. \square

Proposition 4.3. *By using the relaxed visit mechanism, the resulting orientation on a connected undirected bridgeless multigraph (orientable multigraph) is strongly connected.*

Proof. Suppose the underlying simple graph of the given multigraph. It is strongly oriented by the algorithm. When constructing the orientation for the multigraph, the order of visits through its edges is not changed and these edges are oriented in the same way as the underlying graph. Thus, the resulting orientation is strongly connected. \square

This algorithm is determinist. Given a graph and a root vertex, it generates only one strong orientation. By running the algorithm with each n vertex of the graph as root, we can hence construct at most n strong orientations, whereas some of them might be identical. However, the DFS can provide more possibilities. In fact, the order of selecting adjacent

vertices to visit from the current vertex does not impact the strong connectivity of the resulting orientation, while different orders result in different configurations of the edges. Thus, varying this order increases the possibility to build different strong orientations.

In this work, the variation of the order is introduced by giving a weight to every vertex. When scanning from a current vertex, its available adjacent vertices (through a non-oriented edge) are sorted in descending order according to their weights. In this way, a different vector of weights might result in a different strong orientation. This also allows to have a randomized algorithm, if every weight is randomly generated with respect to a certain distribution $\mathcal{U}_{[0,1]}$.

Property 4.1. *Introducing this variation of the selecting order is valid for URND and MRND.*

Thus, we obtain the final algorithm detailed in Algorithm 4.1. As for the input arguments, it requires an undirected graph G (either simple for URND or multiple for MRND), a vector of vertex weights W and a root vertex u . Note that any vertex can be used as root. Here, the vertex with the highest weight is considered as the root vertex for the first call of the algorithm. As described in the previous paragraphs, the selecting order is defined by sorting the adjacent vertices in lines 1-2. Then, following this order, lines 3-8 examine every neighbor v of u and propagate a visit to v if the edge (u, v) is not yet oriented, see the line 4. More precisely, the line 5 sets the orientation for the edge (u, v) and the line 6 recursively calls the strong orientation algorithm with v as the new root vertex.

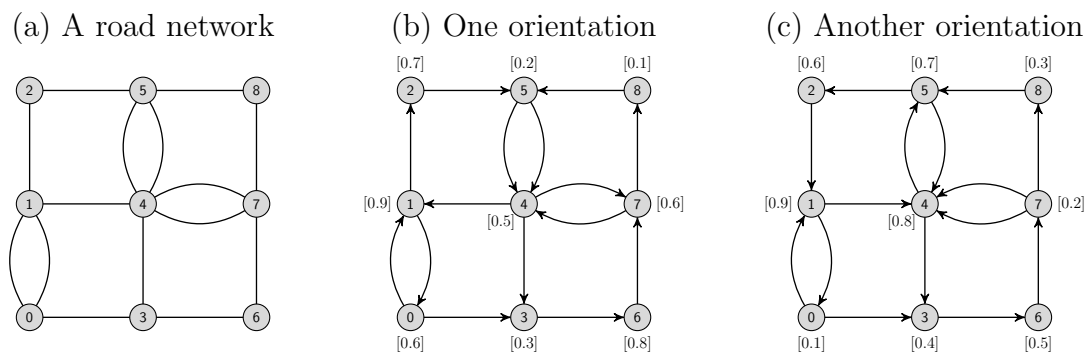


Figure 4.1: Examples of the strong orientation algorithm

An example is given in Figure 4.1. Figure 4.1-(a) shows the initial undirected network with 9 vertices and 15 edges. Suppose the following weights $\{0.6, 0.9, 0.7, 0.3, 0.5, 0.2, 0.8, 0.6, 0.1\}$ are assigned respectively to vertices $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ as depicted in Figure 4.1-(b). The algorithm starts at vertex 1, whose weight is the largest. From its neighbors $\{0, 2, 4\}$ (see Figure 4.1-(a)), vertex 2 is first visited because of its higher weight (0.7) and the edge (1, 2)

Algorithm 4.1 *StrongOrient*(G, W, u)

Input: $G = (V, E)$: undirected graph (simple or multiple)

W : vertices weight

$u \in V$: root vertex

Output: G with a strong orientation

- 1: $N(u) \leftarrow$ list of adjacent vertices of u
 - 2: sort $N(u)$ by descending order of weights in W
 - 3: **for** each $v \in N(u)$ **do**
 - 4: **if** $(u, v) \in E$ not oriented **then**
 - 5: set orientation from u to v
 - 6: *StrongOrient*(G, W, v)
 - 7: **end if**
 - 8: **end for**
-

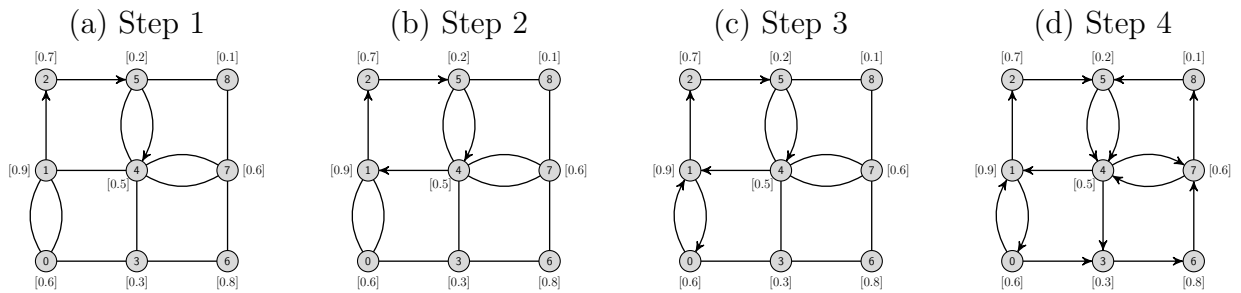


Figure 4.2: Orientation process breakdown for Figure 4.1-(b)

is oriented from 1 to 2. The algorithm continues from vertex 2 to vertex 5 and ends up with the orientation in Figure 4.1-(b). Figure 4.1-(c) reports another orientation obtained by applying the same algorithm with weights $\{0.1, 0.9, 0.6, 0.4, 0.8, 0.7, 0.5, 0.2, 0.3\}$.

However, there exist some strong orientations that cannot be constructed by the proposed algorithm. For instance, considering the strong orientation illustrated in Figure 4.3, there do not exist any weight vectors that allow its generation. As a consequence, this algorithm, even randomized by the random weights, may not cover the entire solution space. Thus, it could not generate the optimal solution for certain instances. However, this issue can be overcome by applying a Local Search (LS).

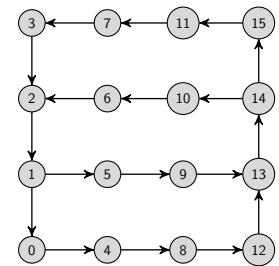


Figure 4.3: Orientation impossible to generate

This algorithm is used as a basic component in our proposed BRKGA and ILS. It is used as the decoding procedure and the constructive heuristic, respectively. Further details are explained in following sections in this chapter.

4.2 Single-objective optimization

Heuristics are a very classic way to quickly compute solutions for combinatorial optimization problems. The possible loss of optimality and the fact that poor or even infeasible solutions may be produced are compensated by the short CPU time consumed, whereas sometimes near-optimal solutions can also be found. The main drawback of heuristics is often the poor quality of the produced solution. Thus, LS procedures are designed to improve a given solution by iteratively performing modifications improving its quality, until no more improvement can be done. In addition, metaheuristics extend the LS principle by providing global guidance.

Given an initial solution s , a LS is a method for iteratively improving s by performing local changes. It stops whenever s cannot be improved anymore. Thus, s has become a local optimum s^* . Usually, a LS relies on one or several move operators, which define neighborhoods of the current solution, $\mathcal{N}_i(s), i = 1, 2, \dots, k$ (k is specified according to the needs). A neighborhood is the set of all solutions that can be obtained from s by performing a kind of moves. Since a LS is dependent to the input solution and on the moves, the quality of the output solution s^* can be poor. Thus, the community tries to optimize the use of the LS in order to enhance its performance to solve complex problems. An intuitive approach is to embed a LS into a multi-start method that calls the LS on a number of randomly generated starting solutions (random restart), for instance Greedy Randomized Adaptive Search Procedures (GRASP). In this way, the output solutions can be diverse and the one with the best fitness can be chosen as the final output solution. Furthermore, memory structures can be integrated in the procedure so as to bias the search towards poorly explored regions or regions of interest, for instance TS and Ant Colony Optimization (ACO). Embedding LS into global guidance mechanisms leads to the metaheuristics.

A metaheuristic is often a high level coordinator for a set of simple heuristics. It organizes the way these algorithms are called, along with some procedures to overcome their drawbacks, such as the restart, multistart, randomization and shaking procedures. Key metaheuristics include Simulated Annealing (Kirkpatrick (1984)), GRASP (Feo & Resende (1989) and Feo & Resende (1995)), Tabu Search (Glover (1989) and Glover (1990)), Variable Neighborhood Search (Mladenović & Hansen (1997)), Iterated Local Search (Lourenço *et al.* (2003)), etc. Population-based metaheuristics are also widely used, such as Scatter Search (Glover (1977)),

Genetic Algorithm (Goldberg (1989)) and ACO (Dorigo & Stützle (2004)).

In order to simplify the presentation of heuristic methods, we first present methods for single-objective RND first and then we show their adaptations to bi-objective optimization. Two proposed metaheuristics, a BRKGA and an ILS are studied respectively in Subsections 4.2.1 and 4.2.2. In both subsections, a brief state-of-the-art is first presented for the respective method, followed by the adaptation to the context of both single-objective URND and MRND.

4.2.1 BRKGA

GA (Goldberg (1989)) is an evolutionary algorithm that works on a population of individuals. Each individual is represented by a sequence of n genes, called a chromosome, for which a fitness value can be evaluated. A so-called generation consists of the creation of a new population by crossover and mutation operators. After a given number of generations, GA selects the fittest individual as the output of the algorithm following the specification of the given problem. At each generation, GA makes its population evolved by randomly selecting and combining individuals in order to generate new individuals called offsprings. Two individuals are selected as parents denoted A and B . This selection can follow a specific strategy, for instance, an individual with a better fitness is more probable to be selected. Then, the gene $i \in [1, n]$ of their child chromosome is copied arbitrarily from the gene i of either A or B . This procedure is called a crossover. Hence, new children compose the new population to make up the next generation. In addition, a random mutation is another fundamental component of a GA. It is applied to escape local optima and hence improve the diversity of the algorithm. Unlike the crossover, the mutation takes only one chromosome and changes its genes such that it may become different. Note that the mutation does not always take place. A probability can be given in order to decide whether a mutation happens or not in each generation.

One may note that some components of GA are general while others are problem-specific. For example, problems with different objective functions often have different ways to evaluate an individual, but the crossover always follows a common mechanism. A large part of the specification comes from different encodings, which are the way to represent an individual over different problems and implementations. Thus, using random keys as a common representation of an individual has been proposed by Bean (1994). This leads to a GA with random keys, referred as a Random Key Genetic Algorithm (RKGA). Such an approach is very generic and it can be applied to handle different combinatorial optimization problems. In a RKGA, every gene of a chromosome is a random key, *i.e.* a real value in the interval $[0, 1]$. Depending on the specification of the given problem, a vector of random keys should

be decoded into a feasible solution by calling a deterministic algorithm, *i.e.* a decoder, which includes also the logic to evaluate the fitness of the solution. In this way, the problem-specific part is totally isolated. The RKGA can hence be reused in different problems. This only requires an effort to implement specific decoders.

Regarding the classic mechanisms of the evolution, a RKGA keeps the principles of the crossover and the mutation with a slight difference to classical GAs. First, the elitism is applied in a RKGA. Depending on the fitness of the individuals, the population is divided into a small group of elite individuals with a better fitness, and a larger group of the other individuals who are less fit (called non-elites). All elite individuals of a generation are copied without any modification to the next generation. Note that newly created individuals may have better fitness than these copied elites. Therefore, a RKGA sorts the population to recompute the elite set before going to the next generation. Second, the mutation is no longer directly applied on individuals. When creating a new population, a given number of new individuals, called mutants, are introduced. Finally, besides the unchanged elites and the mutants, the crossover is used to generate only the remainder of the new population and the parents are randomly selected from the current population.

BRKGA has been proposed by Gonçalves & Resende (2011) and Resende (2012). It is another GA-based framework that extends RKGA. It inherits the major features of RKGA such as the random keys, the elitist population and the mechanism of mutants. The term “biased” refers to the crossover operator. In a BRKGA, to generate an offspring, a parent is always selected from the elite group and the other parent comes from the non-elite group. Moreover, during the crossover, the child has a higher probability to inherit a gene from the elite parent. In this way, the population tends to keep the characteristics from fittest parents over generations. In Gonçalves & Resende (2011), the authors show that a BRKGA converges much better than a classical GA without loss of diversity. Two RKGA are then compared to a BRKGA by using time-to-target plots and the latter shows a better performance. Parameter settings are also addressed by a set of experiments carried out on variations of the population size, the partition of elite individuals, the number of mutants and the elite inheritance probability. As a consequence, recommendations for the parameters configuration are given. Applications of BRKGA to telecommunication problems are presented in Gonçalves & Resende (2011) and Resende (2012). For each problem, authors give a brief definition, the way a solution is encoded by random keys, the heuristic in the decoder and several numerical results. Particularly, it is shown that a BRKGA can combine a LS in the decoder in order to improve solutions quality. Stefanello *et al.* (2017) propose using a BRKGA on toll setting problem for minimizing the traffic congestion. It aims at assigning tolls to critical streets and deciding their tariffs. The main goal is to reduce the

congestion and better distribute the traffic, since drivers tend to choose alternate routes in order to avoid paying tolls. Authors propose mathematical formulations and a BRKGA to solve the problem. Numerical results of experiments on the models as well as the BRKGA are shown and it is stated that the BRKGA produces high quality solutions.

4.2.1.1 BRKGA applied to URND and MRND

For the problems in this thesis, the randomized strong orientation algorithm (Algorithm 4.1) can be used as a decoder, since the vector of weights can be interpreted as a vector of random keys. Thus, a BRKGA is developed taking this algorithm as the basis of the decoding heuristic for both URND and MRND. A solution is an oriented graph (respectively G_1 and G_2) encoded in the BRKGA with random keys representing the weights on each vertex. Using the Algorithm 4.1 and set the vertex of highest random key as the input root vertex u , a chromosome of random keys is decoded into a solution. Then, depending on the optimization objective, the solution fitness is evaluated. An adapted Dijkstra's shortest paths algorithm (Dijkstra (1959)) on all vertices is used for c_1 objective. The c_2 objective is evaluated by counting the number of arcs whose directions are reversed comparing to the initial graph. This decoding procedure is detailed in Algorithm 4.2.

Algorithm 4.2 *Decode*(G, c)

Input: $G = (N, A)$: graph (simple or multiple) with initial orientation

c : a chromosome with random keys

Output: fitness of the orientation decoded from c

Variables: g : temporary graph

- 1: $g \leftarrow \text{removeOrientation}(G)$
 - 2: $u \leftarrow$ index of the highest random key in c
 - 3: $\text{StrongOrient}(g, c, u)$
 - 4: $f \leftarrow \text{fitness}(g)$
 - 5: **return** f
-

Since the chromosomes depend on random keys and are independent to the initial orientation of the given graph, the solutions decoded can show a large difference in the configuration. It has been observed that the resulting graphs often have a lot of arc reversals, which is not suitable when minimizing the c_2 objective in neither URND nor MRND. Thus, a repairing procedure is added to the decoding method, especially for c_2 . It aims at reducing as much as possible the arc reversals. According to the guideline of a BRKGA, a solution should always be feasible. Therefore, this procedure also ensures that the resulting graph

is strongly connected by using Tarjan's Strongly Connected Components (SCC) algorithm proposed in Tarjan (1972). Modifications on Algorithm 4.2 result the Algorithm 4.3 when optimizing the c_2 objective. Since the complexity of the Tarjan's algorithm is $\mathcal{O}(|N|+|A|)$ for a given graph $G = (N, A)$, the procedure in Lines 4-10 has a complexity of $\mathcal{O}(|N| \times |A| + |A|^2)$ in the worst case where all arcs $a \in A$ are reversed.

Algorithm 4.3 *Decode_Repair_* $c_2(G, c)$

Input: $G = (N, A)$: graph (simple or multiple) with initial orientation

c : a chromosome with random keys

Output: fitness of the orientation decoded from c

Variables: g : temporary graph

```

1:  $g \leftarrow \text{removeOrientation}(G)$ 
2:  $u \leftarrow$  index of the highest random key in  $c$ 
3:  $\text{StrongOrient}(g, c, u)$ 
4: for each reversed arc  $e$  in  $g$  do
5:   reverse  $e$  back to its initial direction
6:   run Tarjan's algorithm on  $g$ 
7:   if  $g$  has multiple strongly connected components then
8:     reverse  $e$ 
9:   end if
10: end for return  $\text{fitness}(g)$ 

```

With the proposed solution encoding and decoding, BRKGA can be used to solve both URND and MRND. This requires a graph with the initial orientation, the size of the population and the number of wanted generations. The graph can be either G_1 or G_2 , for URND and MRND, respectively. Algorithm 4.4 presents the process for solving an instance of the problems. The population is initialized with the given number of chromosomes in Line 1. Then, the population is evolved for the given number of generations in Lines 2-13. In each generation, the BRKGA decodes every chromosome in the population by the respective algorithm (Algorithm 4.2). Next, Lines 6-12 is the evolution procedure which is common for either URND and MRND. The BRKGA sorts the chromosomes by their fitness, partitions the population into an elite group and a non-elite group, initializes a new population, copies the elite chromosomes into the new population, generate mutants for the new population and generate offsprings of the current population to make up entirely the new population for the next generation. At the end of all generations, the best fitness is returned as the near-optimal solution found, shown in Line 14.

Algorithm 4.4 *BRKGA*(G, p, n)

Input: $G = (N, A)$: graph (simple or multiple) with initial orientation p : population size n : number of generations**Output:** fitness of G with incumbent near-optimal orientation**Variables:** P : population P' : temporary population c : a chromosome

```
1:  $P \leftarrow$  generate  $p$  chromosomes (vectors of random keys)
2: for  $i \leftarrow 1$  to  $n$  do
3:   for  $c \in P$  do
4:     fitness( $c$ )  $\leftarrow$  Decode( $G_1, c$ )
5:   end for
6:   sort  $P$  by fitness
7:    $P \leftarrow \{E|\bar{E}\}$  ▷ partition  $P$  into elite ( $E$ ) and non-elite ( $\bar{E}$ ) individuals
8:    $P' \leftarrow E$  ▷ copy elite individuals into the new population  $P'$ 
9:    $P' \leftarrow P' \cup M$  ▷ copy generated mutants ( $M$ ) into  $P'$ 
10:   $P' \leftarrow P' \cup C$  ▷ copy chromosomes ( $C$ ) generated by crossover into  $P'$ 
11:   $P \leftarrow P'$ 
12: end for
13: return current best fitness
```

4.2.2 ILS

In the context of optimizing the use of a LS, Lourenço *et al.* (2003) first formally defined the ILS which follows the guideline of a random shaking (biased sampling) better than a random restart (random sampling). The main purpose is to take a single input solution and repeat a LS for a certain number of iterations. This high-level procedure coordinates several operators and the LS in order to obtain a solution of a better fitness than the one given by running the LS itself for a single time. According to the authors, ILS is divided into four major independent components: the generation of an initial solution, the LS, the acceptance criterion and the perturbation (shaking). The initial solution is often generated by a constructive heuristic. Its quality can be mediocre so that the algorithm tries to improve it. The LS then combines several operators that push the current solution to its neighborhoods so as to find other solutions which might better fit. In the end of the LS, the solution is

stable in a local optimum and cannot be further improved. Thus, the LS is trapped into the local optima. A perturbation procedure is then applied at each iteration, trying to generate a close-enough neighbor solution such that calling the LS on it will lead to an exploration of new regions of the solution space. Before passing to a next iteration, the incumbent solution is evaluated by an acceptance criterion which decides whether it will be used as the new input to the next iteration or not. In most applications, the acceptance criterion is defined by the improvement on the objective function. This means if the current solution has a better fitness, it will replace the input solution and be passed to the next iteration. The authors claim that the choice of the criterion is important and other conditions may also have a good performance.

Applications of ILS on different optimization problems are presented in Lourenço *et al.* (2003) as well as in Lourenço *et al.* (2010), such as the Traveling Salesman Problem, the Quadratic Assignment Problem, Vehicle Routing Problems, scheduling problems and the MAX-SAT problem. It is claimed that, depending on the given problem, sophisticated components of a ILS and problem-specific fine tuning are necessary to reach a better performance.

4.2.2.1 ILS applied to URND and MRND

In our single-objective ILS intended to address URND and MRND, the DFS-based orientation algorithm is used as the constructive heuristic to generate the initial solution as shown in Algorithm 4.5. To comply to its input requirements, the set of weights is randomly generated in the similar way of BRKGA. Then, the graph's orientation is removed by simply making all its directed arcs undirected. Finally, the resulting undirected graph is strongly oriented by applying the strong orientation algorithm. Thus, the initial solution is feasible by construction.

As for the LS, a Variable Neighborhood Descent (VND) from Mladenović & Hansen (1997) is used. Given a predefined set of neighborhood structures $\{\mathcal{N}_i, i = 1, 2, \dots, l\}$, the VND starts with an initial solution s_0 and \mathcal{N}_1 . When an improving solution s' is found in the neighborhood $\mathcal{N}_k(s)$ of the current solution s , the method moves to s' according to the first improvement search strategy and resumes to the first neighborhood, *i.e.* $k \leftarrow 1$. Otherwise, the search continues on the next neighborhood ($k \leftarrow k + 1$) of s , until all the l neighborhoods have been scanned. Adapted from Santos *et al.* (2013), three neighborhoods are developed. A move in \mathcal{N}_1 reverses the orientation of all arcs in a cycle found in the given network, as illustrated in Figure 4.4. A move in \mathcal{N}_2 reverses all arcs incident to a node, as shown in Figure 4.5. A move in \mathcal{N}_3 reverses one arc, see Figure 4.6. The strong connectivity is ensured by moves in \mathcal{N}_1 , while it is not in \mathcal{N}_2 and \mathcal{N}_3 . Hence Tarjan's Algorithm for SCC is used

Algorithm 4.5 *Construct*(G)

Input: $G = (N, A)$: graph (simple or multiple) with initial orientation

Output: an orientation of G_2

Variables: g : temporary graph

c : a set of random weights in $[0, 1]$

- 1: $g \leftarrow \text{removeOrientation}(G)$
 - 2: $c \leftarrow \text{generate } |N| \text{ random weights}$
 - 3: $u \leftarrow \text{index of the highest weight in } c$
 - 4: $\text{StrongOrient}(g, c, u)$
 - 5: **return** g
-

as feasibility check to accept only strongly connected neighbors. Disconnected neighbors are rejected and no repair procedure is performed. The detailed procedure is referred in Algorithm 4.6.

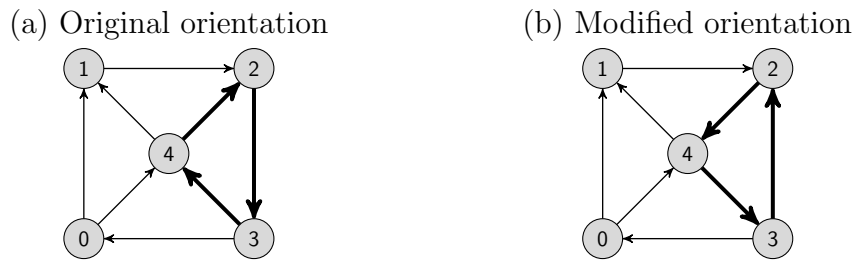


Figure 4.4: \mathcal{N}_1 move in cycle $\{(2, 3), (3, 4), (4, 2)\}$

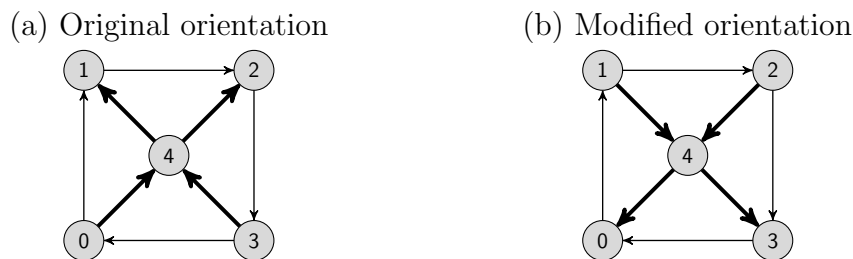


Figure 4.5: \mathcal{N}_2 move at node 4

In the perturbation component of the ILS described in Algorithm 4.7, a random restart mechanism and two shaking functions are developed. The random restart approach generates a brand-new set of weights, then re-orientates the graph by calling the DFS-based orientation algorithm. This provides diversity to ILS even if it breaks the connection between the

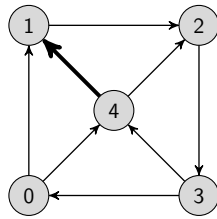
Algorithm 4.6 $VND(g)$ **Input:** $g = G = (N, A)$: multigraph with an orientation**Output:** an orientation of G **Variables:** g' : temporary graph

```

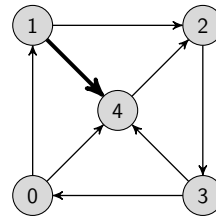
1: repeat
2:    $g' \leftarrow \mathcal{N}_1(g)$ 
3:   if  $\text{fitness}(g') < \text{fitness}(g)$  then
4:      $g \leftarrow g'$ 
5:   else
6:      $g' \leftarrow \mathcal{N}_2(g)$ 
7:     if  $\text{fitness}(g') < \text{fitness}(g)$  then
8:        $g \leftarrow g'$ 
9:     else
10:       $g' \leftarrow \mathcal{N}_3(g)$ 
11:      if  $\text{fitness}(g') < \text{fitness}(g)$  then
12:         $g \leftarrow g'$ 
13:      end if
14:    end if
15:  end if
16: until  $g$  is not improving
17: return  $g$ 

```

(a) Original orientation



(b) Modified orientation

Figure 4.6: \mathcal{N}_3 move at arc $(1, 2)$

current solution and the new one. Following a procedure finding all the cycles of the current orientation, the first shaking reverses arcs direction on the largest cycle which contains one or several arcs belonging to other cycles as well. Such a cycle can be called a dependent cycle. The second shaking changes completely the current orientation of the graph by reversing all arcs direction, which helps to escape local optima. In order to determine which perturbation should be used in an iteration of the ILS, two thresholds on the number of iterations without

improvement are defined to divide the ILS into three stages. A specific perturbation is hence associated with each stage. According to calibrations, the partition is as follows: let i be the current number of iteration without improvement on the incumbent solution and p be the max number of iterations without improvement allowed, such that $0 \leq i \leq p$ and the ILS stops if $i = p$; the random restart takes effect when $0 \leq i \leq p/3$; the first shaking is applied when $p/3 \leq i \leq 2p/3$; in the last iterations ($2p/3 < i < p$), the second shaking is used.

Algorithm 4.7 *RestartOrShake*(g, i, p)

Input: $g = G = (N, A)$: multigraph with an orientation

i : current number of iteration without improvement

p : max number of iteration without improvement

Output: an orientation of G

Variables: g' : temporary graph

- 1: **if** $0 \leq i \leq p/3$ **then return** Construct(g)
 - 2: **else if** $p/3 \leq i \leq 2p/3$ **then return** CycleReverse(g)
 - 3: **else return** AllArcReverse(g)
 - 4: **end if**
-

According to the architecture given by Lourenço *et al.* (2003) and Lourenço *et al.* (2010), the dedicated components are coordinated in the proposed ILS given by Algorithm 4.8. Line 1 initializes the counter of iterations without improvement. The initial solution is generated and then optimized by a LS in Lines 2-3. Next, Lines 4-13 repeats the procedure of the perturbation (Line 5), the LS (Line 6) and the acceptance criterion (Lines 7-12) under the stopping condition depending on the number of iterations without improvement specified in Line 4. Since the acceptance follows a greedy criterion, the incumbent solution is the one with the best fitness returned as the output of the metaheuristic in Line 14.

4.3 Bi-objective optimization

Many real-world operations research problems cannot be handled by optimizing a single objective. This is especially the case in decision making systems where many criteria are often considered simultaneously. That is why multiobjective optimization is widely addressed in the literature. Reviews and surveys can be found in Chankong & Haimes (1983), Chinchuluu & Pardalos (2007), Ehrgott & Gandibleux (2000), Zopounidis & Pardalos (2010) and Zhou *et al.* (2011).

In most of the multiobjective optimization problems, a single solution is not a global

Algorithm 4.8 $ILS(G, p)$

Input: $G = (N, A)$: multigraph with initial orientation
 p : max number of iteration without improvement
Output: fitness of G with incumbent near-optimal orientation
Variables: g, g' : temporary graph
 i : current number of iteration without improvement

- 1: $i \leftarrow 0$
- 2: $g \leftarrow \text{Construct}(G)$
- 3: $g \leftarrow \text{VND}(g)$
- 4: **while** $i < p$ **do**
- 5: $g' \leftarrow \text{RestartOrShake}(g, i, p)$
- 6: $g' \leftarrow \text{VND}(g')$
- 7: **if** $\text{fitness}(g') < \text{fitness}(g)$ **then**
- 8: $g \leftarrow g'$
- 9: $i \leftarrow 0$
- 10: **else**
- 11: $i \leftarrow i + 1$
- 12: **end if**
- 13: **end while**
- 14: **return** $\text{fitness}(g)$

optimum since the objectives are often in conflict. An optimal solution for one of the objectives may have a poor quality for other objectives. Furthermore, many solutions that are optimal for different single objectives can be found, but it is hard to say that one is better or worse than another from a general viewpoint. Thus, solving a multiobjective optimization problem is more about finding a set of solutions. The concept has first been proposed by Pareto (1906) and the solution set is named as the Pareto-optimal set or Pareto front. In the literature, solutions on a Pareto front are said to be “noninferior”, “nondominated” or “Pareto-optimal”. The relation of dominance is well formulated in many studies. For instance, in bi-URND and bi-MRND, given two solutions s_1 and s_2 , s_1 dominates s_2 either if (i) $f_{c_1}(s_1) < f_{c_1}(s_2)$ and $f_{c_2}(s_1) \leq f_{c_2}(s_2)$ or if (ii) $f_{c_1}(s_1) \leq f_{c_1}(s_2)$ and $f_{c_2}(s_1) < f_{c_2}(s_2)$. If such a s_1 does not exist, s_2 is nondominated.

Determinist methods for solving multiobjective optimization problems are also well studied, such as the weighting method (Zadeh (1963)) and the ϵ -constraint method (Haimes *et al.* (1971) and Chankong & Haimes (1983)). However, it is often very expensive to find

all nondominated solutions of a given problem, especially with a large number of criteria. Even identifying whether a solution belongs to the Pareto-optimal set is NP-hard. Hence, heuristics should be investigated in order to find good quality Pareto front in a reasonable time. Among others, Evolutionary Algorithm (EA) are intuitive for multiobjective purposes because they rely on populations, similar to finding a set of solutions. Important reviews and investigations are referred to Kalyanmoy (2001) and Zitzler & Thiele (1998).

4.3.1 An ϵ -constraint method for bi-URND and bi-MRND

The bi-objective version for URND and MRND models, resp. bi-URND and bi-MRND, are solved to optimality by means of the ϵ -constraint method. It is based on the enumeration strategy proposed by Haimes *et al.* (1971). This method is especially intuitive whenever at least one of the objective functions take values on \mathbb{Z} . As a consequence, its value may be turned into a constraint of the problem while the other ones are optimized. This means the search space is constrained by successively optimizing one of the objectives and considering the new solution value as a new bound. The procedure stops when the sub-problem becomes infeasible. Examples of applications for the ϵ -constraint method are found in Bérubé *et al.* (2009) and De Sousa *et al.* (2015).

In the case of bi-URND, in order to compute the Pareto-optimal front \mathcal{F} , its extreme points Z_1 and Z_2 are first computed by optimizing the single objectives (3.1) and (3.2) respectively, within a mixed integer linear programming by CPLEX. The values $c_2 = \overline{c_2}$ and $c_2 = \underline{c_2}$ of arc reversals in respectively Z_1 and Z_2 define the interval of variation on c_2 for solutions in \mathcal{F} . Since $c_2 \in \mathbb{N}$, \mathcal{F} can be computed by solving the modified problem $M(\kappa)$ in which objective function (3.2) is turned into a constraint $c_2(x) \leq \kappa$, *i.e.*

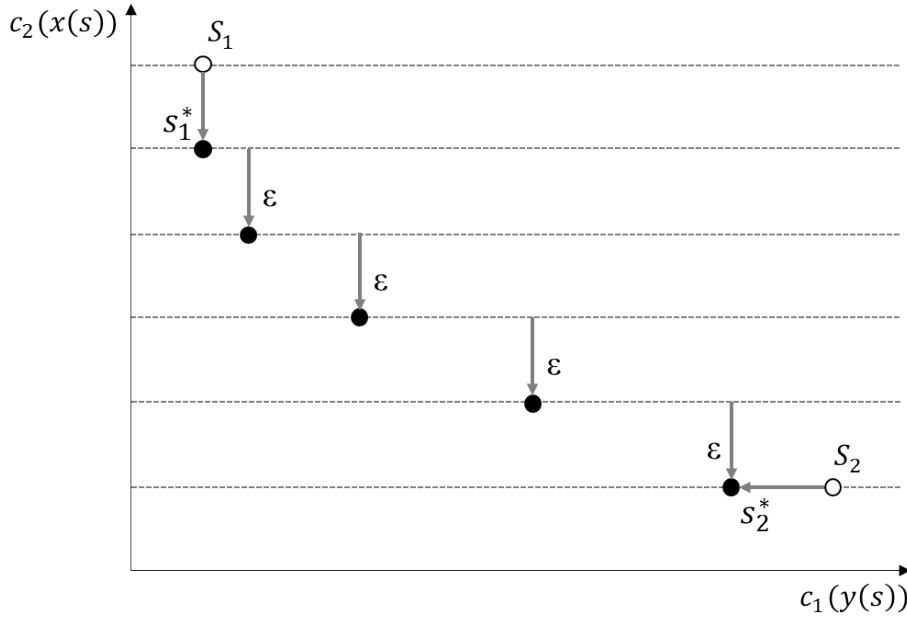
$$\sum_{(i,j) \in A'_1} x_{ij} \leq \kappa \quad (4.1)$$

Every integer value $\kappa \in [\underline{c_2}, \overline{c_2}]$ can be investigated using the ϵ -constraint method since $c_2(x) \in \mathbb{N}$.

The same approach is applied for bi-MRND, whereas the objective function (3.21) is turned into the following constraint:

$$\sum_{(i,j,k) \in A'_2} x_{ijk} \leq \kappa \quad (4.2)$$

Figure 4.7 illustrates the schema of ϵ -constraint method applied on bi-URND and bi-MRND. The extreme points s_1 and s_2 are computed by optimizing respectively c_1 and


 Figure 4.7: Schema of ϵ -constraint method

c_2 criteria by a MILP solver. One may note that by solving the subproblem (4.1) or (4.2) with $\kappa = c_2(s_1)$, we might obtain a solution s_1^* such that $s_1^* \prec s_1$ with $c_1(s_1^*) = c_1(s_1)$ and $c_2(s_1^*) < c_2(s_1)$. Then, by iteratively reducing κ by a predefined value of ϵ (in the case of bi-URND and bi-MRND, $\epsilon = 1$), the subproblem is solved in MILP until the solution s_2^* , $c_2(s_2^*) = c_2(s_2)$ and $c_1(s_2^*) \leq c_1(s_2)$, is found. Details of this adapted method can be found in Algorithm 4.9.

4.3.2 Bi-objective BRKGA adaptation for bi-URND and bi-MRND

BRKGA is originally designed for single-objective combinatorial optimization. Adaptations need to be done for solving bi-URND and bi-MRND. To do so, a solution pool is created in the proposed BRKGA to store solutions. To construct a Pareto front for a given instance, two single-objective BRKGA optimizations are done for the respective c_1 and c_2 objectives. All solutions found during these two processes are stored in the pool, including intermediate solutions obtained in the repairing procedure mentioned in Subsection 4.2.1. After the two independent minimization procedures, a dominance check is done using all solutions in the pool. Those not dominated by any other are inserted into the Pareto front returned as the result of the bi-criteria optimization. Thus, our BRKGA for the bi-objective version of the problems consists in two calls of the BRKGA, including in each call the predefined number of generations, followed by a merge and a filter on the two sets of solutions.

Algorithm 4.9 ϵ – constraint(G, ϵ)

Input: $G = (N, A)$: graph (simple or multiple) with initial orientation

ϵ : step of constraint variation

Output: Pareto-optimal front of G

```

1:  $s_{c_1}^* = (c_1^*, \bar{c}_2) \leftarrow$  optimize  $G$  for  $c_1$  objective
2:  $s_{c_2}^* = (\bar{c}_1, c_2^*) \leftarrow$  optimize  $G$  for  $c_2$  objective
3:  $f \leftarrow \{s_{c_1}^*, s_{c_2}^*\}$ 
4:  $p \leftarrow (c_1^*, \bar{c}_2)$ 
5: while  $c_2(p) > c_2^*$  do
6:    $\bar{c}_2 \leftarrow c_2(p)$ 
7:    $p \leftarrow \min\{c_1(p) | c_2(p) \leq \bar{c}_2 - \epsilon, p \in P\}$ 
8:    $f \leftarrow f \cup \{p\}$ 
9: end while
10: return  $f$ 
    
```

4.3.3 Bi-objective ILS adaptation for bi-URND and bi-MRND

Similar to BRKGA, ILS was not originally designed for multi-objective optimization. In order to deal with both bi-URND and bi-MRND, an Aggregate Objective Function (AOF) of c_1 and c_2 objectives is used in the proposed ILS as presented in Equation (4.3). Given a solution for URND or MRND, the objective values of c_1 and c_2 are evaluated and normalized by $f_{c_1}(s)$ and $f_{c_2}(s)$, respectively. Given a varying parameter $\alpha \in [0, 1]$, they are aggregated into one objective value $f_\alpha(s)$, which can be used as the evaluation criterion in the components of our ILS. The AOF replaces the evaluation evaluation not only in a bi-criteria optimization, but also when solving single-objective URND and MRND. Running the ILS for several times with different α give a set of nondominated solutions, hence defining an approximation of the Pareto-optimal front. Besides, calling the ILS with $\alpha = 1$, the objective value of c_2 is ignored and the c_1 objective is minimized; to optimize the c_2 objective, it is sufficient to set $\alpha = 0$.

$$\min_{s \in S} f_\alpha(s) = \alpha f_{c_1}(s) + (1 - \alpha) f_{c_2}(s) \quad \forall \alpha \in [0, 1] \quad (4.3)$$

Thus, similar to BRKGA, the bi-objective adaptation of ILS consist in running ILS several times and merging/filtering the sets of solutions.

4.3.4 NSGA-II

EA have been proved to be efficient for solving combinatorial optimization problems. A number of implementations and sophisticated adaptations are widely applied in different research areas. Since multi-objective optimization problems arise over the years, extension of EAs to Multi-Objective Evolutionary Algorithm (MOEA) gains researchers' attention and efforts have been given such as in Kalyanmoy (2001) and Zitzler & Thiele (1998). In particular, Multi-Objective Genetic Algorithm (MOGA) is well studied in Deb *et al.* (2002), Fonseca & Fleming (1993), Horn *et al.* (1994), Santos *et al.* (2014) and Srinivas & Deb (1994).

Goldberg (1989) suggests the nondominated sorting based on the ranking mechanism can be integrated in a GA for a multi-objective optimization. At each generation, a nondominance check is first performed on all individuals according to their objective values. The individuals nondominated by the remaining individuals of the current population are assigned to the current rank (starting from 1) and removed from the current population. Then, the rest of the population is scanned for another nondominance check. By repeating the process of nondominance check, rank assignment and individual removal, the sorting terminates when all solutions are given a rank.

The Non-dominated Sorting Genetic Algorithm (NSGA) has been hence proposed by Srinivas & Deb (1994) by using the ranking mechanism. In addition, a sharing method with a niche formation technique is applied in order to maintain the diversity of the population and the distribution of nondominated solutions. The other mechanisms of the metaheuristic follow the guideline of a classical GA, such as the crossover and the mutation. Regarding simulation experiments, three test problems are solved by the proposed NSGA and a Vector Evaluated Genetic Algorithm (VEGA), another MOGA developed in the thesis of Schaffer (1984). By using a performance measure defined by Deb (1989) for evaluating the distribution of a Pareto front, authors claim that the NSGA performs generally better than the VEGA, except several cases where the sharing parameter is not well set.

However, drawbacks of NSGA have been identified by the community, including the high complexity due to the implementation of the nondominated sorting, the lack of an elitist mechanism and the dependence on the sharing parameter. In Deb *et al.* (2002), these disadvantages are well explained and an enhanced approach is proposed, named NSGA-II. This improved method integrates a fast nondominated sorting with a lower complexity. The main idea is, for every solution, to memorize the number of solutions dominating it and to store all the solutions dominated by it. The time complexity is proved to be $\mathcal{O}(MN^2)$ where M is the number of objectives and N is the population size, but this new approach requires a slightly higher storage space of $\mathcal{O}(N^2)$. NSGA-II also replaces the sharing function by a

crowded-comparison operator which guarantees a good solution distribution on the Pareto front as well. Instead of sharing the fitness, a crowding distance is computed for every solution to indicate whether it is located in a crowded region of the current rank or not. The selection of the GA is then guided by comparing the crowding distances in a way that solutions in less crowded areas are preferred. In the literature, NSGA-II is compared to other two MOEA on four different problems and it has generally a better performance.

4.3.4.1 NSGA-II applied to bi-URND and bi-MRND

Since the proposed BRKGA and ILS are adapted to the bi-objective optimization, it is valuable to compare them with a metaheuristic especially designed for multiobjective purposes. Thus, the NSGA-II is also developed. It applies a generic fast nondominated sort with a ranking calculation, a crowded-comparison operator with a crowding distance calculation, the classical crossover and a bitwise mutation. However, by carrying out some preliminary experiments, the strong orientation algorithm proposed in Section 4.1 seemed not fit for NSGA-II. A different chromosome encoding has been hence designed. It also takes a set of random weights $w \in [0, 1]$ as input but they are associated with arcs. If an arc possesses a weight $w < 0.5$, it is oriented from the vertex with a lower label to the other end and vice versa. Readers are referred to Algorithm 4.10 for details. This procedure has a lower computational complexity of $\mathcal{O}(m)$ (m is the number of arcs), but the orientation it proposes cannot be guaranteed to be strongly connected. In such a case, some shortest paths do not exist and a high-enough cost is used instead. This corresponds to a partial penalization of an infeasible solution. Nonetheless, according to the results of a testing implementation, every orientation is kept in the final population of the NSGA-II as it contributes to the performance of the final Pareto front.

4.4 Computational experiments and numerical results

Four computational experiments have been performed using the proposed metaheuristics so as to test their performance in terms of the quality of the solutions and the processing time. In the first experiment, parameter tuning has been carried out for the proposed metaheuristics, BRKGA, ILS and NSGA-II. The second experiment consists of the single-objective optimization on the generated instances. The small- and medium-sized instances, referred respectively in Tables 3.1 and 3.2, are solved by BRKGA and ILS so as to analyze their quality of solutions comparing to the optima from CPLEX. Two additional sets of medium- and large-sized instances (presented in Tables 4.1 and 4.2) that CPLEX cannot solve in a reasonable amount of time or memory. Tests on these larger instances are done

Algorithm 4.10 *ArcOrient*(G, W)

Input: $G = (V, E)$: undirected graph (simple or multiple) W : arcs weight**Output:** G with an orientation

```
1: for each  $e = (u, v) \in E$  such that  $u < v$  do
2:   if  $W[e] < 0.5$  then
3:     set orientation from  $u$  to  $v$ 
4:   else
5:     set orientation from  $v$  to  $u$ 
6:   end if
7: end for
```

only using the proposed metaheuristics. Third, the bi-objective optimization by the proposed methods has been tested on some of the generated instances. The last experiment aims at solving instances extracted from the real road network of Troyes, which is further explained in Subsection 4.4.4. Note that the machine used for running all experiments is the same one previously described in Section 3.5. The ϵ -constraint method is coded and compiled in Java version 1.7 using the “IBM ILOG CPLEX Concert” API library. All the metaheuristics are coded in C++ and compiled by Visual C++ version 11.0 (in “Visual Studio 2012”).

4.4.1 Parameter tuning

A tuning phase is essential before getting into the experiments in order to figure out good settings for the proposed metaheuristics. In every calibration experiment, the worst, the best and the average solution values and average running time in seconds are used as performance indicators. Table 4.3 reports all parameters tested in the calibration experiments. For each method and each configuration, one parameter varies and the others were fixed. In BRKGA, the key parameters to be tuned are the seed of the random number generator (SD_B), the multiplier of the number of nodes (R_B) used to deduce the population size, the maximal number of generations (I_B) and the number of generations without profit (J_B). In ILS, the tuning is performed on the seed of the random generator (SD_I) and the number of iterations without profit (J_I). Note that, since ILS relies on α variation for bi-objective optimization, the pace of variation (Δ_I) needs also to be tuned. In NSGA-II, the settings concern the seed of the random number generator (SD_N), the multiplier of the number of nodes (R_N) used to deduce the population size, the maximal number of generations (I_N), the probability of

Name	n	m	p
7x7-b1			1
7x7-b2	49	84	2
7x7-b4			4
7x7-b6			6
8x8-b1			
8x8-b2	64	112	2
8x8-b4			4
8x8-b6			6
9x9-b1			
9x9-b2	81	144	2
9x9-b4			4
9x9-b6			6
10x10-b1			
10x10-b2	100	180	2
10x10-b4			4
10x10-b6			6
11x11-b1			
11x11-b2	121	220	2
11x11-b4			4
11x11-b6			6
12x12-b1			
12x12-b2	144	264	2
12x12-b4			4
12x12-b6			6

Table 4.1: Characteristics of generated larger simple graphs

Name	n	m	$M\%$	p		
7x7-b1-25		105	25			
7x7-b1-50	49	126	50	1		
7x7-b1-75					147	75
7x7-b1-100					168	100
7x7-b2-25						105
7x7-b2-50	49	126	50	2		
7x7-b2-75					147	75
7x7-b2-100					168	100
8x8-b1-25						140
8x8-b1-50	64	168	50	1		
8x8-b1-75					196	75
8x8-b1-100					224	100
8x8-b2-25						140
8x8-b2-50	64	168	50	2		
8x8-b2-75					196	75
8x8-b2-100					224	100
9x9-b1-25						180
9x9-b1-50	81	216	50	1		
9x9-b1-75					252	75
9x9-b1-100					288	100
9x9-b2-25						180
9x9-b2-50	81	216	50	2		
9x9-b2-75					252	75
9x9-b2-100					288	100

Table 4.2: Characteristics of generated larger multigraphs

crossover (ρ_{cN}) and the probability of mutation (ρ_{mN}).

Tests are launched on a set of samples chosen from the generated instances detailed previously, including 6 instances: 3x3-b1, 4x4-b2, 5x5-b4, 3x3-b2-25, 4x4-b1-75 and 5x5-b1-50. The strategy for testing a parameter is similar to a unidimensional optimization, that is varying one parameter at a time among several predefined possible values with other parameters fixed. Then, the value with better results are used to fix this parameter and the process is repeated on another parameter until a so far best settings is determined. In this

study, 3 possible values are chosen for every parameter reported in Table 4.3.

BRKGA				ILS			NSGAI				
SD_B	R_B	I_B	J_B	SD_I	J_I	Δ_I	SD_N	R_N	I_N	ρ_{cN}	ρ_{mN}
0	1.25	50	6	2	6	0.075	2	1.25	50	0.3	0.03
2	1.45	100	12	4	12	0.1	4	1.45	100	0.6	0.1
8	1.75	150	24	8	24	0.15	8	1.75	150	0.9	0.3

Table 4.3: Predefined possible values of the parameters

For example, Table 4.5 presents the preliminary results obtained by tuning the parameter SD_I in ILS for c_1 objective. The additional column “ $\delta_{c_1}\%$ ” represents the margins of the objective values comparing to the global optima computed by CPLEX (*cf.* Section 3.5). The other parameter J_I is fixed and SD_I is varied among the predefined values. Generally, the performance of ILS is better when setting $SD_I = 4$ in terms of the processing time and the secondary value c_2 . Another example is given on the tuning of R_B in BRKGA for bi-objective optimization, as shown in Table 4.6. The column “ $\delta_{HV}\%$ ” compares the resulting Hypervolume (HV) (Zitzler & Thiele (1998)) to the exact Pareto-optimal fronts computed by ϵ -constraint method. The value 1.45 for R_B saves the computational time in general, without losing too much quality of the fronts. After the calibration, the sets of parameter values in Table 4.4 are chosen for the following experiments in this work. Other numerical results in this experiment can be found in Appendix B.

BRKGA				ILS			NSGAI				
SD_B	R_B	I_B	J_B	SD_I	J_I	Δ_I	SD_N	R_N	I_N	ρ_{cN}	ρ_{mN}
0	1.45	100	12	4	12	0.1	2	1.75	100	0.3	0.03

Table 4.4: Chosen parameter values

4.4.2 Results for single-objective URND and MRND

Tables 4.7 and 4.8 present respectively the results for the URND minimizing c_1 and c_2 on instances from Tables 3.1 and 4.1, with the number of nodes less than 50. In the following, results for the MRND minimizing c_1 and c_2 are listed respectively in Tables 4.9 and 4.10 that are related to multigraph instances referred in Tables 3.2 and 4.2 (also with the number of nodes less than 50). In order to compare the results in an intuitive way, the numerical results for solving URND and MRND models on CPLEX solver are copied from respective tables reported in Section 3.5, keeping the defined notation of columns. In addition, for both heuristics (BRKGA and ILS), columns “ c_1 ”, “ c_2 ” and “t(s)” provide respectively the total

Instance	Settings											
	$SD_I = 2, J_I = 6$				$SD_I = 4, J_I = 6$				$SD_I = 8, J_I = 6$			
	c_1^*	c_2	t(s)	$\delta_{c_1}\%$	c_1^*	c_2	t(s)	$\delta_{c_1}\%$	c_1^*	c_2	t(s)	$\delta_{c_1}\%$
3x3-b1	246	2	0.14	0	246	2	0.14	0	246	2	0.13	0
4x4-b2	968	17	0.60	0	968	17	0.52	0	968	17	0.67	0
5x5-b4	3032	13	3.86	0	3032	13	2.67	0	3032	13	4.29	0
3x3-b2-25	250	7	0.15	0	250	7	0.14	0	250	2	0.14	0
4x4-b1-75	716	8	1.61	0	716	7	1.22	0	716	7	1.28	0
5x5-b1-50	2296	15	7.65	0	2296	15	4.10	0	2296	22	4.81	0
Best	246	2	0.14	0	246	2	0.14	0	246	2	0.125	0
Worst	3032	17	7.647	0	3032	17	4.103	0	3032	22	4.805	0
Average	1251.33	10.33	2.34	0.00	1251.33	10.17	1.46	0.00	1251.33	10.50	1.89	0.00

 Table 4.5: Results of single-objective ILS parameter tuning on SD_I

Instance	Settings											
	$R_B = 1.25$				$R_B = 1.45$				$R_B = 1.75$			
	Q	HV	t(s)	$\delta_{HV}\%$	Q	HV	t(s)	$\delta_{HV}\%$	Q	HV	t(s)	$\delta_{HV}\%$
3x3-b1	2	0.00	0.98	0	2	0.00	1.25	0	2	0	1.19	0
4x4-b2	4	0.50	4.98	0	4	0.50	9.07	0	4	0.495	10.36	0
5x5-b4	9	0.81	47.41	10	9	0.82	28.34	9	10	0.754	51.18	8
3x3-b2-25	1	0.00	0.91	0	1	0.00	1.05	0	1	0	1.28	0
4x4-b1-75	8	0.65	5.62	0	8	0.65	6.80	1	8	0.647	8.14	1
5x5-b1-50	15	0.72	31.23	10	14	0.73	39.05	11	14	0.756	43.32	6
Best	15	0	0.905	0	14	0	1.046	0	14	0	1.185	0
Worst	1	0.805	47.408	0.104	1	0.818	39.047	0.109	1	0.756	51.184	0.075
Average	6.50	0.45	15.19	0.03	6.33	0.45	14.26	0.03	6.50	0.44	19.25	0.02

 Table 4.6: Results of bi-objective BRKGA parameter tuning on R_B , with other parameters fixed: $SD_B = 2$, $I_B = 100$ and $J_B = 12$

travel distance, the number of reversals and the processing time in seconds. Furthermore, columns “ c_1^* ” and “ c_2^* ” rely on optimum values for c_1 and c_2 objective function, respectively.

Results in Tables 4.7 and 4.8 indicate that ILS performs generally better than BRKGA. On 4x4 and 5x5 instances, ILS found 9 out of 10 global optima when minimizing c_1 (in Table 4.7) and 10 out of 10 global optima when minimizing c_2 (in Table 4.8); in the same time, BRKGA found 7 out of 10 global optima when minimizing c_1 (in Table 4.7) and 8 out of 10 global optima when minimizing c_2 (in Table 4.8). On 6x6 instances, CPLEX solved only 1 out of 4 instances in the time limit when minimizing c_1 and both BRKGA and ILS are able to find the global optimum. When minimizing c_2 , CPLEX can easily solve all four instances of 6x6 to optimality and the two heuristics are also able to find the global optima. As for the instances of 7x7, it is not even possible to read the LP files for neither c_1 nor c_2 objective

Instance	CPLEX					BRKGA			ILS		
	c_1^*	c_2	t(s)	LR	Gap(%)	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
3x3-b1	246	2	0.27	212	13.82	246	2	0.18	246	2	0.15
3x3-b2	284	2	0.16	264	7.04	284	2	0.18	284	2	0.09
4x4-b1	912	22	22.40	786	13.82	912	3	0.82	912	1	0.47
4x4-b2	968	5	11.34	836	13.64	968	5	1.17	968	17	0.56
4x4-b4	1392	8	3.99	1216	12.64	1392	8	0.60	1392	8	0.42
4x4-b6	1304	7	1.72	1088	16.56	1388	6	0.58	1304	7	0.28
5x5-b1	2638	14	414.62	2292	13.12	2668	23	2.42	2638	25	4.16
5x5-b2	2748	16	373.96	2404	12.52	2748	16	2.64	2758	11	2.28
5x5-b4	3032	23	182.74	2636	13.06	3032	23	2.68	3032	13	2.82
5x5-b6	3116	4	253.86	2672	14.25	3140	28	3.35	3116	4	1.64
6x6-b1	≤ 6250	-	7200	5506	11.90	6244	22	8.60	6394	36	8.57
6x6-b2	≤ 6554	-	7200	5632	14.07	6356	32	7.70	6470	24	7.71
6x6-b4	≤ 6820	-	7200	5956	12.67	6780	35	10.94	6854	9	8.34
6x6-b6	7006	35	3894.85	6394	8.74	7006	19	7.29	7006	35	6.14
7x7-b1	Out of memory					13282	30	37.32	13344	43	49.90
7x7-b2						13592	35	33.10	13688	29	23.68
7x7-b4						14206	49	31.64	14214	30	24.80
7x7-b6						14824	45	24.54	14494	50	34.29

Table 4.7: c_1 minimization for URND (I)

due to the memory limit, whereas the two metaheuristics could finish the computation on all the instances. We can observe that, in c_1 optimization, BRKGA found better solutions than ILS on 3 out of 4 instances and ILS performs better on the other instance. When minimizing c_2 , the two methods resulted in similar objective values for 3 out of 4 instances, while ILS found a solution with a better c_2 value than BRKGA in the other instance.

As for time consumption, BRKGA and ILS have a much better performance than CPLEX in the harder problem c_1 . CPLEX optimizes c_2 much more quickly than c_1 since the former is a polynomial problem while the latter is NP-hard. It can even solve instances with up to 36 nodes (6x6 instances) when minimizing c_2 while c_1 cannot be optimized in 2h limit. Comparing the two heuristics, the running time is generally in the same scale with a slight difference in some instances. ILS is less time-consuming than BRKGA in c_1 minimization on a number of instances and it is not too sensitive to instance size. This is due to the fact that ILS addresses only one solution per iteration. However, BRKGA solved the problems for c_2 globally faster than ILS (see Table 4.8), because the local search operators in ILS tend to reverse arcs' direction, which is hard to control and does not comply with the arc reversals minimization. Thus, a post-processing is added to ILS in order to reverse back some reversed arcs and to reduce the value of c_2 for URND.

Instance	CPLEX					BRKGA			ILS		
	c_1	c_2^*	t(s)	LR	Gap(%)	c_1	c_2^*	t(s)	c_1	c_2^*	t(s)
3x3-b1	248	0	0.06	0	0	248	0	0.11	248	0	0.09
3x3-b2	284	2	0.05	2	0	284	2	0.09	284	2	0.05
4x4-b1	960	0	0.44	0	0	960	0	0.48	960	0	0.77
4x4-b2	1060	0	0.42	0	0	1060	2	0.52	1060	0	0.33
4x4-b4	1442	2	0.51	2	0	1538	3	0.33	1442	2	0.38
4x4-b6	1388	6	0.39	6	0	1388	6	0.30	1388	6	0.21
5x5-b1	3388	0	3.84	0	0	3388	0	1.72	3388	0	2.50
5x5-b2	3960	0	3.51	0	0	3980	2	1.20	3960	0	2.39
5x5-b4	3844	1	3.70	1	0	3844	1	1.01	3844	1	2.03
5x5-b6	3322	2	3.28	2	0	3732	2	0.86	3322	2	1.85
6x6-b1	7770	2	20.39	2	0	8180	2	2.75	7770	2	10.77
6x6-b2	7636	1	21.25	1	0	7636	1	3.09	7636	1	9.69
6x6-b4	7606	1	20.90	1	0	7606	1	2.31	7606	1	8.78
6x6-b6	7814	3	18.78	3	0	8522	3	3.40	7814	3	11.54
7x7-b1	Out of memory					15312	2	8.10	15684	2	34.83
7x7-b2						16700	1	12.28	16650	1	18.92
7x7-b4						17240	2	6.93	17638	2	15.46
7x7-b6						18034	4	10.64	18044	2	15.94

 Table 4.8: c_2 minimization for URND (I)

Tests were also carried out using the strategy of the time-to-target plot (tttplot), proposed by Aiex *et al.* (2002) and Aiex *et al.* (2007), in order to compare the robustness of the proposed metaheuristics. The idea of tttplots is to compare the ability of the methods to obtain a certain target value following 200 runs with different random seeds. In addition, the selected target value should not be too easy nor too difficult to reach. Thus, the selection of target values is done so as to allow conclusions in these experiments. Here, for each instance (Table 3.1 for URND and 3.2 for MRND), we fixed a small deviation from the optimum value to obtain the target values. The axes x and y indicate respectively the running time and the probability to reach the target value. For each method, the more the curve is straight, the higher probability the method has to reach rapidly the target value. Example plots obtained by minimizing c_1 objective for URND are illustrated in Figure 4.8. The instances from Table 3.1 are used in these experiments. Results indicate that ILS seems to perform globally better than BRKGA for URND as it reaches target values quicker.

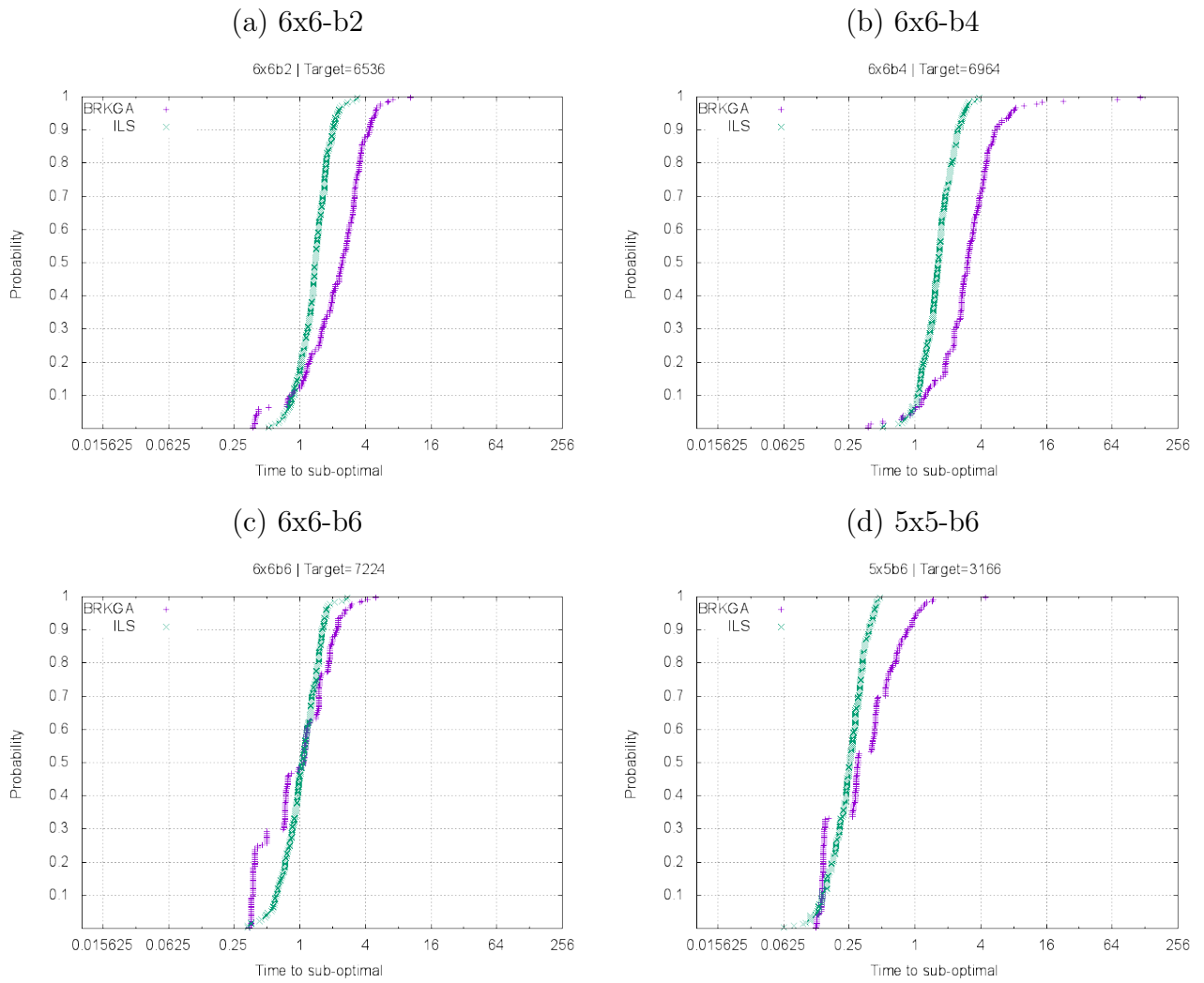


Figure 4.8: Time-to-target plots of BRKGA and ILS for URND c_1 objective

Instance	CPLEX					BRKGA			ILS		
	c_1^*	c_2	t(s)	LR	Gap(%)	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
3x3-b1-25	204	6	0.21	198	2.94	204	6	0.20	204	6	0.09
3x3-b1-50	186	5	0.13	186	0.00	186	5	0.22	186	5	0.10
3x3-b1-75	172	6	0.13	172	0.00	172	6	0.22	172	6	0.11
3x3-b1-100	148	7	0.11	148	0.00	148	7	0.27	148	7	0.14
3x3-b2-25	250	7	0.19	242	3.20	250	2	0.19	250	2	0.13
3x3-b2-50	194	3	0.22	190	2.06	194	3	0.22	194	3	0.11
3x3-b2-75	166	6	0.22	164	1.20	166	5	0.23	166	5	0.12
3x3-b2-100	152	5	0.09	152	0.00	152	5	0.27	152	5	0.13
4x4-b1-25	832	5	4.23	768	7.69	832	5	0.72	832	14	0.63
4x4-b1-50	764	12	1.78	750	1.83	764	7	0.81	764	7	0.50
4x4-b1-75	716	8	1.20	708	1.12	716	7	0.89	716	7	0.91
4x4-b1-100	652	7	0.45	652	0	652	7	1.01	652	7	0.54
4x4-b2-25	850	12	2.92	804	5.41	850	6	0.70	850	6	0.69
4x4-b2-50	764	14	2.00	752	1.57	764	6	0.78	764	6	0.50
4x4-b2-75	720	12	0.70	720	0	720	8	0.89	720	8	0.53
4x4-b2-100	652	10	0.53	652	0	652	8	1.00	652	8	0.89
5x5-b1-25	2406	27	56.69	2270	5.65	2406	10	2.11	2406	10	2.23
5x5-b1-50	2296	12	12.67	2226	3.05	2296	22	2.36	2296	15	3.35
5x5-b1-75	2158	19	7.75	2142	0.74	2158	19	2.64	2158	16	4.05
5x5-b1-100	2024	20	3.65	2024	0	2024	20	2.92	2024	20	1.73
5x5-b2-25	2450	12	41.09	2318	5.39	2450	24	5.15	2498	13	2.19
5x5-b2-50	2260	26	23.90	2194	2.92	2264	19	2.90	2264	21	2.34
5x5-b2-75	2126	16	8.10	2114	0.56	2126	16	2.62	2126	16	4.48
5x5-b2-100	2036	20	3.81	2036	0	2036	20	2.92	2036	20	2.41
6x6-b1-25	Out of memory			5466	-	5954	34	6.29	5952	37	6.29
6x6-b1-50	Out of memory			5396	-	5536	46	7.46	5526	23	6.01
6x6-b1-75	5268	41	41.48	5268	0	5268	40	10.34	5280	31	13.43
6x6-b1-100	5060	35	20.22	5060	0	5060	35	7.43	5060	35	4.78
6x6-b2-25	Out of memory			5592	-	5986	22	8.81	5962	23	5.92
6x6-b2-50	Out of memory			5478	-	5436	26	11.84	5434	27	17.17
6x6-b2-75	5322	34	45.12	5322	0	5322	33	6.76	5338	40	9.65
6x6-b2-100	5092	38	21.61	5092	0	5092	38	7.36	5092	38	4.56
7x7-b1-25	Out of memory					12614	52	36.52	12636	51	59.99
7x7-b1-50	Out of memory					11914	46	38.72	11868	50	153.13
7x7-b1-75	Out of memory					11318	53	29.19	11318	53	88.41
7x7-b1-100	Out of memory					11000	53	16.57	11000	53	26.38
7x7-b2-25	Out of memory					12642	48	43.18	12596	32	86.69
7x7-b2-50	Out of memory					11988	55	56.30	11936	53	76.88
7x7-b2-75	Out of memory					11530	54	34.99	11520	49	85.24
7x7-b2-100	Out of memory					11092	51	16.37	11092	51	26.19

 Table 4.9: c_1 minimization for MRND (I)

Instance	CPLEX					BRKGA			ILS		
	c_1^*	c_2	t(s)	LR	Gap(%)	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
3x3-b1-25	246	0	0.08	0	0	246	0	0.13	246	0	0.23
3x3-b1-50	246	0	0.08	0	0	246	0	0.14	246	0	0.37
3x3-b1-75	242	0	0.09	0	0	242	0	0.17	242	0	0.47
3x3-b1-100	200	0	0.08	0	0	200	0	0.20	200	0	0.62
3x3-b2-25	250	2	0.08	2	0	250	2	0.12	250	2	0.14
3x3-b2-50	208	2	0.06	2	0	214	2	0.14	208	2	0.22
3x3-b2-75	198	1	0.08	1	0	210	2	0.31	198	1	0.56
3x3-b2-100	174	1	0.08	1	0	176	1	0.19	194	1	0.37
4x4-b1-25	862	0	0.23	0	0	856	2	0.41	862	0	1.08
4x4-b1-50	818	0	0.25	0	0	818	0	0.52	818	0	1.37
4x4-b1-75	780	0	0.23	0	0	780	0	0.95	780	0	1.87
4x4-b1-100	712	0	0.19	0	0	712	0	0.67	712	0	1.65
4x4-b2-25	972	0	0.25	0	0	972	0	0.42	972	0	0.90
4x4-b2-50	890	0	0.25	0	0	890	0	0.47	890	0	1.36
4x4-b2-75	798	0	0.25	0	0	798	0	0.58	798	0	1.70
4x4-b2-100	730	0	0.27	0	0	730	0	0.67	730	0	1.89
5x5-b1-25	2936	0	1.45	0	0	2936	0	1.61	2936	0	3.34
5x5-b1-50	2814	0	1.36	0	0	2814	0	1.25	2814	0	5.01
5x5-b1-75	2458	0	1.31	0	0	2458	0	1.78	2458	0	6.82
5x5-b1-100	2408	0	1.19	0	0	2408	0	1.70	2408	0	9.00
5x5-b2-25	2868	0	1.33	0	0	2868	0	1.37	2868	0	3.35
5x5-b2-50	2676	0	1.37	0	0	2676	0	1.84	2676	0	5.43
5x5-b2-75	2468	0	1.33	0	0	2468	0	2.40	2468	0	6.66
5x5-b2-100	2430	0	1.19	0	0	2430	0	1.69	2430	0	8.56
6x6-b1-25	7636	1	26.40	1	0	7636	1	2.29	7636	1	12.62
6x6-b1-50	6550	2	22.04	2	0	6550	2	6.47	6550	2	16.90
6x6-b1-75	6344	1	20.48	1	0	6344	1	6.24	6344	1	24.57
6x6-b1-100	5942	1	16.01	1	0	5942	1	3.78	5982	1	30.06
6x6-b2-25	7308	1	20.28	1	0	7308	1	2.81	7308	1	9.66
6x6-b2-50	6986	0	5.76	0	0	6986	0	5.02	6986	0	15.66
6x6-b2-75	6678	0	17.89	0	0	6678	0	4.77	6678	0	39.05
6x6-b2-100	6116	0	5.07	0	0	6116	0	3.71	6116	0	32.28
7x7-b1-25	Out of memory					16008	0	12.61	16008	0	27.38
7x7-b1-50						18398	0	10.65	18398	0	42.90
7x7-b1-75						14044	0	14.12	14044	0	66.29
7x7-b1-100						13728	0	17.96	13728	0	81.99
7x7-b2-25	Out of memory					15242	0	13.78	15242	0	27.41
7x7-b2-50						15376	3	14.38	16100	0	44.65
7x7-b2-75						15448	1	17.44	15448	1	64.18
7x7-b2-100						15146	0	15.52	15146	0	81.79

 Table 4.10: c_2 minimization for MRND (I)

For the experimentation on multigraph instances shown in Tables 4.9 and 4.10, ILS found 25 out of 28 global optima on instances from 3x3 to 6x6 that can be solved by CPLEX when minimizing c_1 . It solves to optimality all 32 instances solvable by CPLEX when minimizing c_2 . BRKGA performs better for c_1 , finding 27 out of 28 global optima. It is a little bit worse for c_2 , finding 30 out of 32 global optima. With the growth of multiple arcs on instances with the same number of nodes, it seems that the problems are getting easier for CPLEX since it tends to set the bi-direction configuration for two arcs between a pair of nodes. However, the more multiple arcs an instance has, the higher the time consumed by the two heuristics. They seem to be sensitive to the increase of the arc number. There is little gap between BRKGA and ILS in terms of computational time in c_1 minimization, but ILS is slightly more time-consuming than BRKGA when minimizing c_2 on most of the instances.

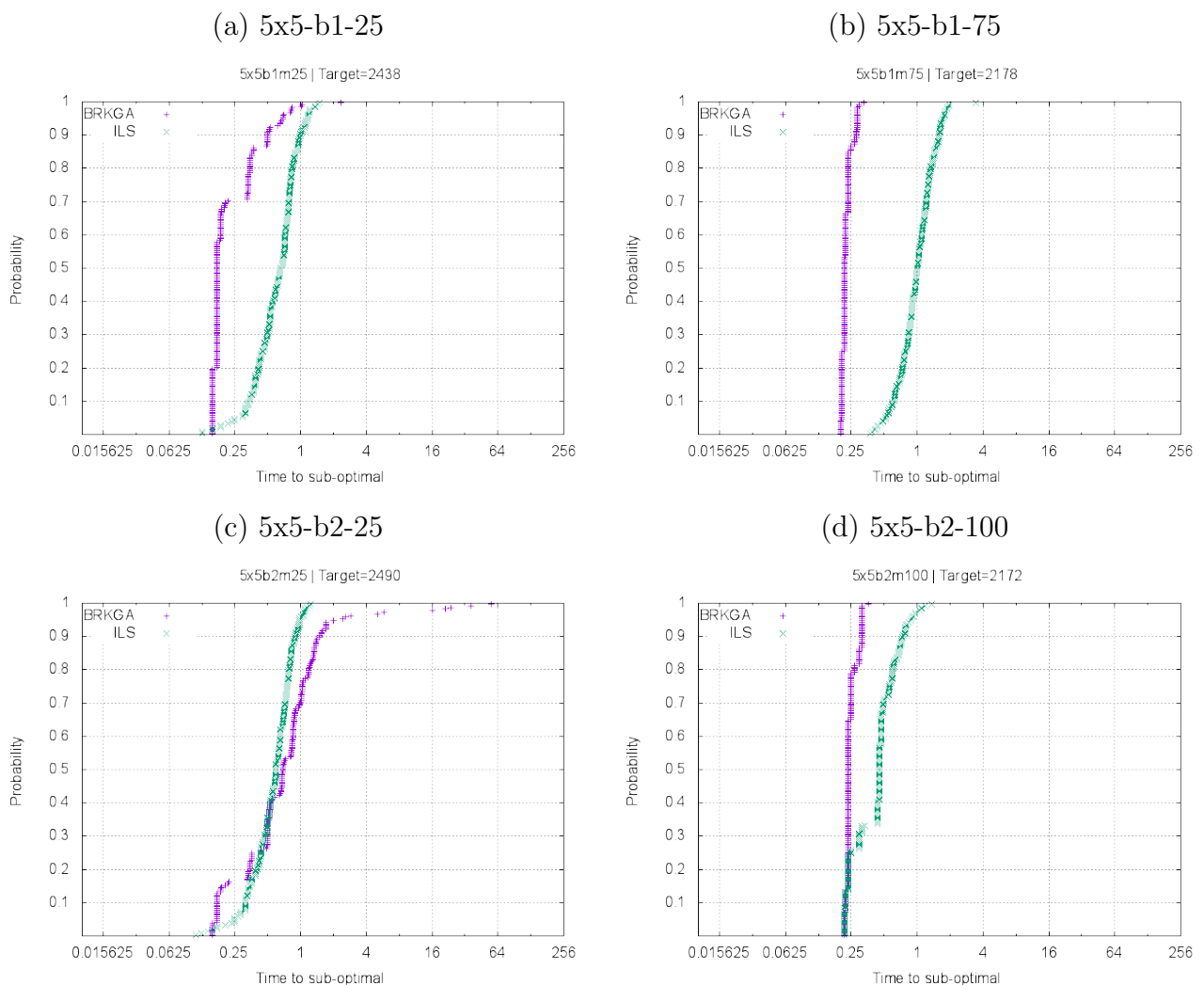


Figure 4.9: Time-to-target plots of BRKGA and ILS for MRND c_1 objective

The tttplots presented in Figure 4.9 illustrate the behavior of the two metaheuristics for

MRND. Instances from Table 3.2 are used in these experiments. Results indicate that, on most of the instances, BRKGA performs better than ILS, while ILS leads BRKGA slightly on a few instances, for instance as shown in Figure 4.9-(c).

Tables 4.11 and 4.12 refer to the instances from Table 4.1 with the number of nodes greater than 50 (8x8, 9x9, 10x10, 11x11 and 12x12 instances). Moreover, Tables 4.13 and 4.14 refer to the instances from Table 4.2 with the number of nodes greater than 50 (8x8 and 9x9 instances). These instances cannot be solved by CPLEX due to memory issues. Hence, experiments were done using BRKGA and ILS. Results of c_1 optimization for URND reported in Table 4.11 indicate that ILS performs much better than BRKGA in terms of solution quality, since BRKGA found lower or identical objective values than ILS on only 4 out of 20 instances, whereas the processing time is very similar for both methods. When minimizing c_2 , much more time is elapsed by ILS to find better or equal objective values than BRKGA on 12 out of 20 instances, as presented in Table 4.12. In addition, ILS has also a better solutions quality than BRKGA on multigraph instances, but its time consumption is even higher, either for optimizing c_1 or c_2 . Details are referred to Tables 4.13 and 4.14. In general, ILS performs better on most of the instances, while BRKGA takes the lead on a few instances for c_2 objective or for MRND.

Instance	BRKGA			ILS		
	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
8x8-b1	25868	61	64.57	25728	61	57.44
8x8-b2	25868	33	62.29	25970	61	69.92
8x8-b4	26686	65	109.79	26532	30	47.19
8x8-b6	27674	63	57.56	26988	50	137.56
9x9-b1	45762	88	189.76	45550	51	142.58
9x9-b2	45412	68	255.22	45714	60	130.65
9x9-b4	47362	89	226.87	46876	82	179.14
9x9-b6	47474	75	194.81	46634	61	250.47
10x10-b1	77378	84	258.32	76478	98	222.34
10x10-b2	76244	72	528.75	76908	88	395.16
10x10-b4	78468	92	552.88	77496	100	343.53
10x10-b6	78510	63	767.77	77702	71	421.89
11x11-b1	122672	106	1187.38	121660	89	864.46
11x11-b2	121830	123	1421.04	122278	89	912.95
11x11-b4	123798	98	1001.62	122578	88	1058.84
11x11-b6	126970	104	811.03	124494	108	718.79
12x12-b1	188368	159	1680.56	187300	134	1460.52
12x12-b2	190872	122	677.76	187384	119	1105.15
12x12-b4	188164	122	2989.76	187690	111	1527.31
12x12-b6	195228	126	791.19	188124	116	1613.10

 Table 4.11: c_1 minimization for URND

(II)

Instance	BRKGA			ILS		
	c_1	c_2^*	t(s)	c_1	c_2^*	t(s)
8x8-b1	48850	3	13.98	46930	1	37.72
8x8-b2	55668	0	24.32	55668	0	39.97
8x8-b4	53130	6	8.83	48826	4	38.72
8x8-b6	40622	5	19.64	42914	3	32.74
9x9-b1	60446	0	36.43	60446	0	91.07
9x9-b2	60188	0	51.04	60188	0	91.98
9x9-b4	65440	5	33.96	66466	5	70.76
9x9-b6	59572	3	34.49	61360	0	77.39
10x10-b1	100160	0	17.98	100160	0	136.63
10x10-b2	100604	0	59.25	100604	0	143.40
10x10-b4	102988	1	79.61	102988	1	143.52
10x10-b6	112602	2	79.90	112602	2	157.92
11x11-b1	184828	3	88.86	184736	1	276.94
11x11-b2	187024	1	93.37	185134	1	275.28
11x11-b4	186314	1	26.01	186314	1	279.04
11x11-b6	186312	5	34.80	186692	1	272.38
12x12-b1	245874	1	186.83	245946	1	464.90
12x12-b2	245720	3	233.83	246060	1	441.26
12x12-b4	251400	1	246.42	251400	1	429.66
12x12-b6	252998	4	176.00	253366	2	419.81

 Table 4.12: c_2 minimization for URND

(II)

Instance	BRKGA			ILS		
	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
8x8-b1-25	24544	64	60.25	24122	44	179.18
8x8-b1-50	23300	72	87.69	23144	63	137.64
8x8-b1-75	22142	69	76.78	22084	65	300.18
8x8-b1-100	21568	74	33.62	21568	74	39.73
8x8-b2-25	24502	69	74.29	24248	56	158.84
8x8-b2-50	23238	61	70.59	23176	75	245.70
8x8-b2-75	22262	70	105.14	22258	65	165.41
8x8-b2-100	21612	69	33.49	21612	69	55.15
9x9-b1-25	43310	61	213.25	42370	95	990.45
9x9-b1-50	41106	101	319.88	41180	101	798.19
9x9-b1-75	39876	87	213.14	39870	85	440.53
9x9-b1-100	38952	97	64.58	38952	97	93.30
9x9-b2-25	43520	66	205.17	43324	71	344.37
9x9-b2-50	41770	81	68.08	41254	88	958.42
9x9-b2-75	39946	81	198.28	40030	87	514.05
9x9-b2-100	38992	92	64.40	38992	92	90.76

 Table 4.13: c_1 minimization for MRND

(II)

Instance	BRKGA			ILS		
	c_1	c_2^*	t(s)	c_1	c_2^*	t(s)
8x8-b1-25	28814	3	18.33	29036	2	54.09
8x8-b1-50	29188	0	29.53	29188	0	97.11
8x8-b1-75	28696	0	42.23	28696	0	143.72
8x8-b1-100	25988	0	12.81	25988	0	184.83
8x8-b2-25	29808	2	12.40	29808	2	57.00
8x8-b2-50	29628	3	23.82	30704	0	101.53
8x8-b2-75	27932	0	26.93	27932	0	136.61
8x8-b2-100	27484	1	28.38	28126	1	169.15
9x9-b1-25	53256	0	45.74	53256	0	124.25
9x9-b1-50	53350	0	56.05	53350	0	198.09
9x9-b1-75	48388	0	90.68	48388	0	270.24
9x9-b1-100	49930	0	21.23	49930	0	382.88
9x9-b2-25	49496	1	39.41	49496	1	116.06
9x9-b2-50	49922	1	63.98	49756	1	180.80
9x9-b2-75	48952	0	76.29	48952	0	264.37
9x9-b2-100	46962	0	20.98	46962	0	359.64

 Table 4.14: c_2 minimization for MRND

(II)

4.4.3 Bi-objective optimization

As previously claimed, BRKGA and ILS were not initially designed for multi-objective optimization. Experiments are hence necessary to validate the adaptations to these methods by comparing their performance for computing Pareto fronts with the classical multi-objective frameworks, ϵ -constraint method and NSGA-II. Due to the long running time of these method on large instances, only the instances from 3x3 to 8x8 are tested in this work.

In order to analyze the quality of Pareto fronts produced by the proposed methods, three performance metrics are used. The first metric is the number of Pareto solutions in the first front. The second metric is the hypervolume (Zitzler & Thiele (1998)) that represents the size of the area dominated by the given front in the solution space. The last one is the spacing (Veldhuizen & Lamont (2000) and Zitzler *et al.* (2003)) which allows to analyze the distribution of the given Pareto front.

Tables 4.15 and 4.16 report respectively the results of bi-URND and bi-MRND obtained on the instances with up to 25 nodes from Tables 3.1 and 3.2. Aside from the notations defined in previous tables, the columns “ Q ”, “ HV ” and “ SP ” present respectively the number of solutions, the HV, a well-known performance measure for Pareto front approximation Zitzler & Thiele (1998) (the higher the better), and the spacing proposed by Veldhuizen &

Lamont (2000) and Zitzler *et al.* (2003) (the lower the better) which allows to analyze the distribution of the given Pareto front. However, due to the memory limit, the ϵ -constraint method cannot handle instances larger than 5x5. Thus, for every instance with the number of nodes greater than 25, nondominated solutions among the solutions found by the three metaheuristics are grouped to construct a set of *Best Known Solutions (BKS)*, denoted “*BKS*” in Tables 4.17 and 4.18. It can hence be used as the basis for the relative comparison of their performance. Readers are referred to Appendices C and D for more figures of Pareto fronts.

Instance	ϵ -constraint				BRKGA				ILS				NSGA-II			
	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)
3x3-b1	2	0.00	0.00	0.30	2	0.00	0.00	1.64	2	0.00	0.00	1.35	2	0.00	0.00	8.20
3x3-b2	1	0.00	0.00	0.14	1	0.00	0.00	1.50	1	0.00	0.00	0.78	1	0.00	0.00	6.85
4x4-b1	2	0.67	0.00	2.34	3	0.07	0.46	21.65	2	0.67	0.00	8.46	2	0.67	0.00	24.09
4x4-b2	4	0.50	0.46	24.07	4	0.50	0.46	16.23	4	0.50	0.46	6.30	4	0.50	0.46	24.73
4x4-b4	5	0.72	0.83	11.59	5	0.72	0.83	8.38	4	0.63	0.82	4.49	4	0.71	0.89	20.22
4x4-b6	2	0.00	0.00	2.28	1	0.00	0.00	6.63	2	0.00	0.00	2.12	2	0.00	0.00	17.66
5x5-b1	15	0.89	0.89	3769.28	12	0.80	1.78	87.32	9	0.87	3.97	42.28	11	0.80	0.94	62.21
5x5-b2	13	0.90	0.87	1835.61	13	0.87	0.86	87.99	7	0.88	3.90	30.45	11	0.90	0.65	64.15
5x5-b4	11	0.90	0.66	1021.64	10	0.84	0.59	77.75	8	0.89	1.27	25.69	7	0.84	4.24	59.34
5x5-b6	2	0.67	0.00	308.13	3	0.05	0.26	69.61	2	0.67	0.00	20.58	2	0.67	0.00	48.95

Table 4.15: Performance comparison of Pareto fronts for bi-URND (I)

Results in Table 4.15 indicate that ILS can approximate Pareto fronts with high quality for bi-URND on 3x3, 4x4 and 5x5 instances. This is confirmed by the tiny optimality gap on HV (comparing with the Pareto-optimal fronts constructed by the ϵ -constraint method) on 9 out of 10 instances. The spacing values seem also reasonable for a non-evolutionary metaheuristic without any functions that ensure the solutions distribution. Since a linear aggregation of the two objective functions is applied in ILS, most of its solutions found on the fronts are supported, whereas a few non-supported solutions can also be reported thanks to the post-processing integrated in the method. For the other two metaheuristics, BRKGA and NSGA-II, they can construct Pareto fronts with more solutions than ILS because of the use of populations, especially on 5x5 instances. The quality of their fronts are also good but they are slightly worse than ILS in general. Moreover, NSGA-II allows to construct Pareto fronts with a very well spread on these instances. Starting from 6x6 instances, the ϵ -constraint method ran out of time due to too many variables and constraints in the mathematical model. All three metaheuristics can approximate the Pareto-optimal fronts in a reasonable amount of time. Figure 4.10 illustrates the Pareto fronts found by these four methods on the 5x5 instance with 1 disruption, respectively.

Moreover, Table 4.16 reports the results for bi-MRND on 3x3, 4x4 and 5x5 instances, all

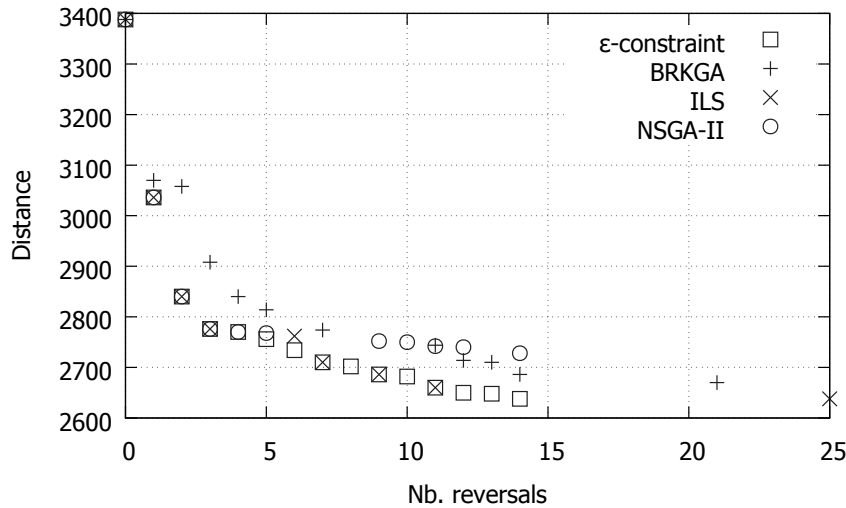


Figure 4.10: Pareto fronts on 5x5-b1 (bi-URND)

Instance	ϵ -constraint				BRKGA				ILS				NSGA-II			
	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)
3x3-b1-25	5	0.66	0.87	0.90	5	0.66	0.87	2.57	5	0.66	0.87	1.98	4	0.36	1.55	10.17
3x3-b1-50	6	0.61	0.21	0.86	6	0.51	0.15	3.03	5	0.60	0.40	2.91	6	0.61	0.21	13.04
3x3-b1-75	7	0.66	0.32	1.06	7	0.57	0.20	3.19	5	0.63	0.85	3.59	7	0.66	0.32	14.84
3x3-b1-100	8	0.71	0.21	1.11	8	0.68	0.16	3.56	4	0.62	1.44	4.22	9	0.69	0.37	19.44
3x3-b2-25	1	0.00	0.00	0.20	1	0.00	0.00	2.00	1	0.00	0.00	1.51	1	0.00	0.00	9.05
3x3-b2-50	2	0.00	0.00	0.31	2	0.00	0.00	2.48	2	0.00	0.00	2.12	2	0.00	0.00	11.97
3x3-b2-75	5	0.56	0.17	1.03	5	0.56	0.17	3.15	4	0.53	0.45	3.23	5	0.56	0.17	15.35
3x3-b2-100	5	0.60	0.12	0.55	5	0.54	0.11	3.21	4	0.56	0.40	2.87	5	0.60	0.12	18.66
4x4-b1-25	5	0.76	0.45	13.71	5	0.73	0.43	16.06	4	0.72	0.93	11.53	6	0.40	3.68	31.36
4x4-b1-50	8	0.75	0.05	13.85	8	0.74	0.05	19.02	5	0.72	0.81	19.14	8	0.74	1.37	44.15
4x4-b1-75	8	0.73	0.07	7.52	8	0.73	0.07	22.16	5	0.69	0.81	21.53	10	0.70	0.09	55.38
4x4-b1-100	8	0.59	0.06	4.95	8	0.59	0.06	22.08	4	0.46	1.93	16.63	8	0.59	0.06	68.16
4x4-b2-25	6	0.48	0.39	11.26	7	0.44	0.06	13.90	4	0.44	0.10	7.96	6	0.48	0.39	32.29
4x4-b2-50	7	0.72	0.12	9.22	7	0.66	0.08	17.38	6	0.70	0.46	14.70	7	0.69	0.72	41.96
4x4-b2-75	9	0.61	0.08	8.07	9	0.61	0.08	20.84	5	0.53	1.27	21.15	9	0.61	0.08	55.33
4x4-b2-100	9	0.81	0.08	6.51	9	0.81	0.08	23.81	4	0.75	1.78	21.54	10	0.76	2.81	67.52
5x5-b1-25	7	0.62	0.35	118.44	11	0.38	0.21	75.54	9	0.41	0.65	56.08	11	0.38	0.34	91.12
5x5-b1-50	13	0.91	0.26	150.56	13	0.88	0.63	104.94	12	0.91	1.14	76.39	20	0.83	1.90	127.05
5x5-b1-75	17	0.85	0.13	142.26	17	0.81	0.13	120.21	8	0.81	2.47	108.58	12	0.79	3.44	155.70
5x5-b1-100	21	0.84	0.15	101.59	21	0.83	0.14	122.09	11	0.82	1.55	121.37	26	0.78	1.27	190.65
5x5-b2-25	13	0.72	0.17	477.25	10	0.63	0.55	109.15	7	0.69	1.36	50.93	10	0.70	0.43	89.37
5x5-b2-50	14	0.88	0.17	240.56	17	0.81	0.59	121.68	10	0.86	3.91	96.56	14	0.87	0.52	121.40
5x5-b2-75	17	0.87	0.13	145.08	17	0.83	0.14	133.40	12	0.86	0.69	102.21	17	0.80	3.74	153.97
5x5-b2-100	21	0.79	0.17	135.24	21	0.76	0.13	121.68	11	0.78	1.34	106.22	21	0.72	0.92	182.46

Table 4.16: Performance comparison of Pareto fronts for bi-MRND (I)

the three metaheuristics have some issues on particular instances. It is clear that ILS has a relatively mediocre performance on 3x3 and 4x4 instances. Since there are usually a very small number of solutions on the Pareto-optimal fronts, the loss of one or two nondominated

solutions by ILS can result in a significant gap in terms of HV. When the instance size is getting larger, this cause has less impact and the quality of the fronts found by ILS is getting better. NSGA-II performs very well on 3x3 and 4x4 instances with only a slight quality decrease on 5x5 instances. BRKGA has a relatively stable behavior throughout the size of instances. On every set, it can construct Pareto fronts with high quality on most of the instances while the performance gap is noticeable on the others. Especially, BRKGA performs very well in terms of the spread, with leading spacing values on 22 out of 24 instances. An example illustrating the Pareto fronts built by all four methods can be found in Figure 4.11 for the 5x5 instance with 100% multiple arcs and 2 disruptions.

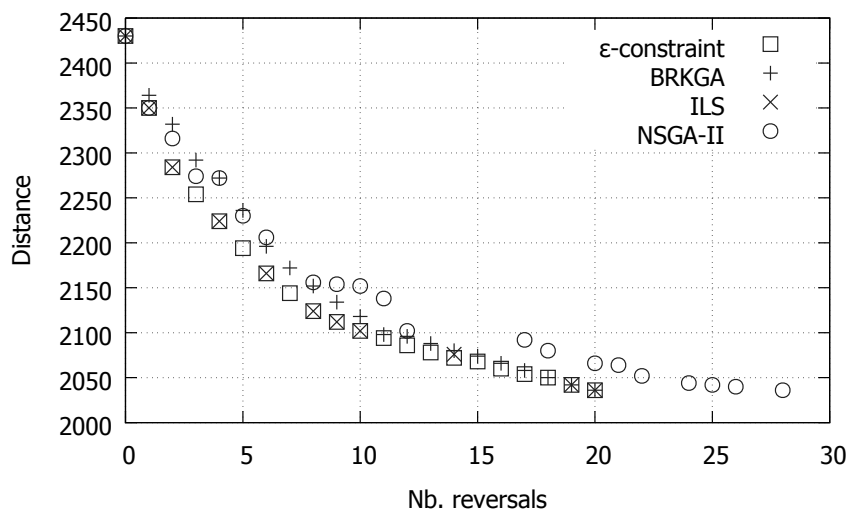


Figure 4.11: Pareto fronts on 5x5-b2-100 (bi-MRND)

Regarding the bi-URND on 6x6, 7x7 and 8x8 instances, results of the three metaheuristics are shown in Table 4.17. BRKGA and NSGA-II have a globally similar performance comparing to BKS. They can find good quality fronts with a large quantity of solutions. Besides, the spread performance of these two methods are very well and generally better than ILS. In particular, NSGA-II consumes less time than the other two methods. On the other hand, the Pareto fronts found by ILS are very close to BKS on these instances but with minor gaps in terms of solution numbers. However, for bi-MRND with results reported in Table 4.18, ILS starts to struggle when the problem size is getting larger, particularly on the 8 instances of 8x8. Sometimes, it spends more than two hours to construct a Pareto front while BRKGA and NSGA-II can finish most of the tasks in less than one hour. In terms of the front quality, ILS still has a significant lead in front of the other two metaheuristics. Figure 4.12 illustrates the Pareto fronts constructed by the three metaheuristics on the 8x8 instance with 6 disruptions.

For bi-MRND with respect to results in Table 4.18, ILS continues to perform well with

Instance	BKS		BRKGA				ILS				NSGA-II			
	Q	HV	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)
6x6-b1	12	0.88	17	0.82	2.59	716	10	0.88	0.18	133	11	0.78	2.30	161
6x6-b2	15	0.78	17	0.68	1.21	485	13	0.76	2.12	126	14	0.62	3.16	161
6x6-b4	9	0.77	10	0.70	0.67	332	6	0.77	1.47	119	9	0.76	0.67	144
6x6-b6	10	0.72	10	0.48	1.65	438	8	0.68	1.26	88	5	0.67	0.11	137
7x7-b1	15	0.78	16	0.61	1.09	1369	11	0.74	2.30	703	8	0.74	0.82	368
7x7-b2	19	0.83	17	0.76	1.75	1276	14	0.83	1.98	739	20	0.58	1.82	367
7x7-b4	7	0.91	13	0.75	4.13	1100	7	0.91	1.50	587	15	0.83	0.80	346
7x7-b6	14	0.87	24	0.74	3.56	1179	9	0.85	6.05	701	13	0.72	3.70	362
8x8-b1	17	0.94	33	0.87	2.40	4556	15	0.93	3.80	2158	23	0.82	4.63	732
8x8-b2	15	0.92	22	0.88	1.76	3474	10	0.86	5.61	2024	28	0.87	1.97	716
8x8-b4	18	0.93	26	0.84	1.26	4894	17	0.91	3.45	1898	21	0.83	1.79	677
8x8-b6	18	0.94	26	0.85	3.96	2840	17	0.94	1.20	1727	32	0.80	1.22	655

Table 4.17: Performance comparison of Pareto fronts for bi-URND (II)

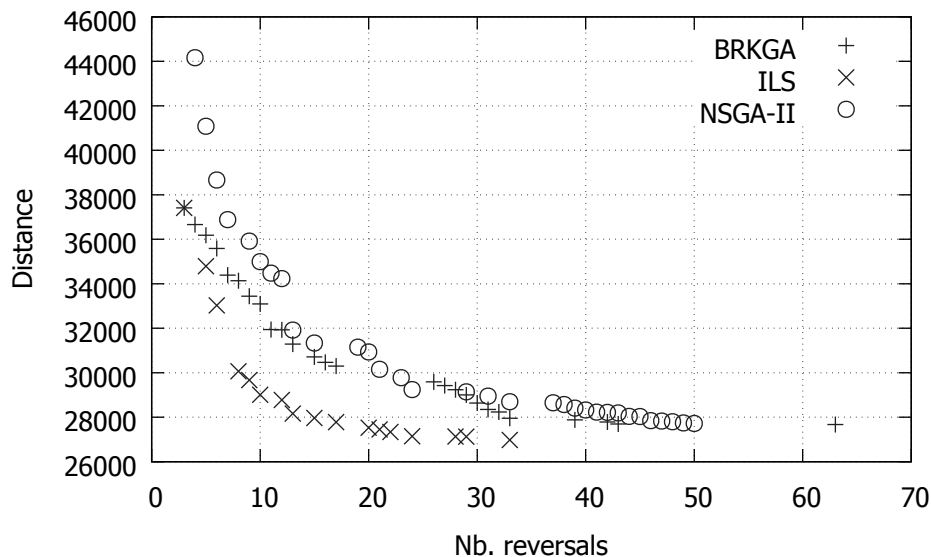


Figure 4.12: Pareto fronts on 8x8-b6 (bi-URND)

a very comparable quality of fronts to BKS. However, the drawback of a high running time is getting more significant, compared to the other two metaheuristics. BRKGA has a good performance on several instances such as 6x6-b1-100, 6x6-b2-100 and 7x7-b2-100. This particularly refers to the instances with a high percentage of multiple arcs. NSGA-II still runs with a stable performance by finding good quality Pareto fronts in a very short time. By extending its processing duration, there may be ample room for improvement. By testing the proposed methods on this kind of large generated instances, it can hence be expected that real (small or medium scale) networks can be handled to address both bi-URND and bi-MRND. In terms of the solutions distribution, BRKGA had better spacing on 19 out of 24 instances than ILS and NSGA-II, while ILS also found 3 fronts out of 24 instances that

Instance	BKS		BRKGA				ILS				NSGA-II			
	Q	HV	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)
6x6-b1-25	16	0.84	18	0.80	0.54	591	12	0.83	4.08	224	21	0.82	0.79	226
6x6-b1-50	15	0.85	22	0.80	0.35	551	11	0.84	0.94	374	21	0.75	1.85	301
6x6-b1-75	19	0.83	29	0.74	0.35	632	16	0.82	2.19	518	28	0.75	1.12	377
6x6-b1-100	29	0.84	35	0.81	0.17	518	14	0.83	2.52	586	35	0.75	2.54	459
6x6-b2-25	17	0.89	19	0.85	1.94	874	16	0.89	1.32	219	20	0.79	4.05	226
6x6-b2-50	18	0.72	28	0.62	0.18	678	12	0.70	2.32	391	24	0.65	0.50	320
6x6-b2-75	26	0.79	34	0.74	0.19	512	22	0.78	1.08	594	32	0.72	0.95	396
6x6-b2-100	24	0.81	39	0.77	0.17	512	11	0.79	3.65	542	40	0.75	0.80	478
7x7-b1-25	22	0.87	31	0.78	1.14	1837	21	0.86	3.82	1725	24	0.73	2.05	534
7x7-b1-50	24	0.90	38	0.84	1.08	1105	22	0.89	4.98	2205	35	0.81	2.32	683
7x7-b1-75	32	0.90	50	0.84	0.53	1276	22	0.89	8.65	3604	37	0.82	1.75	854
7x7-b1-100	39	0.88	54	0.82	0.30	671	20	0.87	5.62	3884	41	0.77	2.60	1046
7x7-b2-25	19	0.84	26	0.67	1.93	1683	17	0.84	1.40	1442	27	0.67	1.59	535
7x7-b2-50	17	0.87	38	0.79	1.59	1119	16	0.87	2.34	3036	35	0.78	1.56	675
7x7-b2-75	24	0.92	46	0.85	0.55	1817	21	0.91	2.23	3741	35	0.83	1.47	861
7x7-b2-100	31	0.90	52	0.86	0.70	635	18	0.89	3.70	4243	33	0.79	2.61	994
8x8-b1-25	15	0.91	31	0.71	3.17	5029	14	0.90	4.29	4199	22	0.68	4.56	1078
8x8-b1-50	26	0.89	40	0.75	3.96	5143	23	0.88	5.75	8196	35	0.77	2.29	1408
8x8-b1-75	30	0.90	61	0.79	0.67	8000	29	0.90	2.12	9476	31	0.74	2.86	1718
8x8-b1-100	37	0.88	75	0.81	0.19	1989	22	0.87	4.81	12378	48	0.72	2.69	2042
8x8-b2-25	19	0.85	35	0.67	2.77	3772	16	0.84	2.63	3959	25	0.71	3.76	1133
8x8-b2-50	24	0.92	43	0.81	1.98	2974	23	0.92	5.42	9468	36	0.74	3.23	1388
8x8-b2-75	40	0.90	55	0.82	1.38	5823	33	0.90	9.37	11697	44	0.77	3.36	1732
8x8-b2-100	33	0.92	69	0.85	0.35	1761	19	0.92	5.74	10958	41	0.77	2.83	2069

Table 4.18: Performance comparison of Pareto fronts for bi-MRND (II)

are better distributed than the other two methods. NSGA-II performs surprisingly worse than BRKGA and ILS using this metric on these instances. Figure 4.13 illustrates the fronts found by the three methods on the 8x8 instance with 100% multi-arcs and 1 disruption.

4.4.4 An application to a real urban network

In order to take a step into the real context of urban network management, we also extracted real geographical data (from OSM) for the urban network of Troyes¹. Several instances are then created onto this basis that can hence be tested by the proposed metaheuristics for both (bi-)URND and (bi-)MRND. Disruptions are selected on a random basis but also in a way to ensure that the resulting graph allows strong orientations to be computed, *i.e.* does not have bridges. We consider that there is a travel demand for every O-D pair. Thus, there should be a path between every pair of nodes. Experiments were performed on the same machine described at the beginning of this section.

The original dataset from OSM consists of 4261 nodes and 614 ways including redundant

¹A medium city in France with about 61,000 inhabitants in 2015

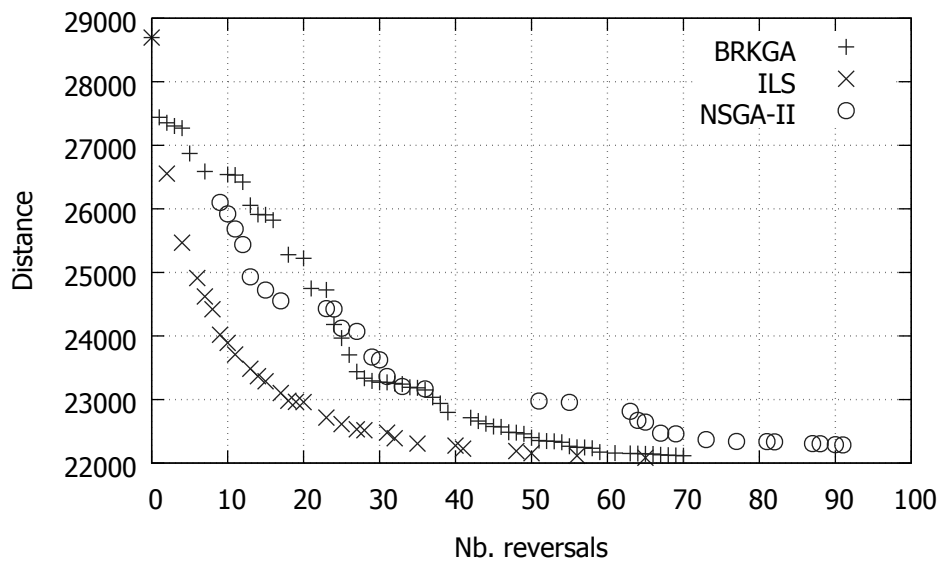


Figure 4.13: Pareto fronts on 8x8-b1-75 (bi-URND)

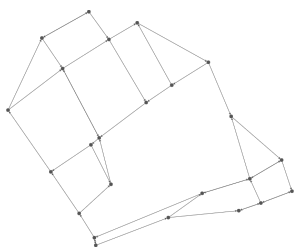


Figure 4.14: Graph of partial Troyes downtown road network with 24 nodes

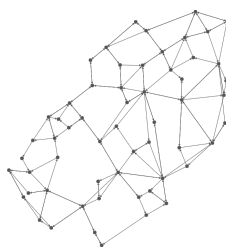


Figure 4.15: Graph of entire Troyes downtown road network with 58 nodes

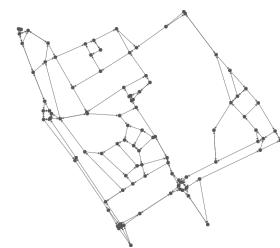


Figure 4.16: Graph of partial Troyes downtown road network with 102 nodes

information like buildings, squares and internal ways in parking lots. Besides, ways are represented by lists of node references instead of edge sequences. Therefore, a post-processing procedure is done to clean up the data and to reformulate the graph structure fitting our needs. Figure 4.14 depicts the extracted multigraph modeling a small block around the cathedral with 24 nodes and 43 arcs. The second extraction presented in Figure 4.15 is from the entire downtown network that uses a simplification heuristic (see Algorithm 4.11) to reduce the graph size in terms of nodes and arcs. It contains 58 nodes and 155 arcs. The western half part of the Troyes downtown network surrounded by the Seine is converted into the third multigraph instance with 102 nodes and 193 arcs, as illustrated in Figure 4.16. Details of the instances are reported in Table 4.19.

In the single-objective optimization, experiments for both c_1 and c_2 objective are carried out on all instances listed below. The behavior of the three methods are also similar to what

Algorithm 4.11 *Simplify*(G)**Input:** $G = (N, A)$: graph (simple or multiple) with initial orientation**Output:** Simplified G

- 1: **for** arc $a = (o, d) \in A_2$ such that $distance(a) < 50$ **do**
- 2: create new node Π with the location in the middle of a
- 3: add Π to N
- 4: link all adjacent nodes of o and d to Π
- 5: delete all the adjacency relations of o and d
- 6: delete a from A
- 7: delete o and d from N
- 8: **end for**
- 9: **return** G

Name	n	m	p
troyes-v24-b1			1
troyes-v24-b2	24	43	2
troyes-v24-b4			4
troyes-v24-b6			6
troyes-v58-b1			1
troyes-v58-b2	58	155	2
troyes-v58-b4			4
troyes-v58-b6			6
troyes-v102-b1			1
troyes-v102-b2	102	193	2
troyes-v102-b4			4
troyes-v102-b6			6

Table 4.19: Characteristics of Troyes network based instances

is described in 4.4.2. First, CPLEX, BRKGA and ILS are used to solve the small instances of troyes-v24. Tables 4.20 and 4.21 contain the results obtained. CPLEX is still able to handle these instances for minimizing both c_1 and c_2 . When minimizing c_1 , BRKGA is a bit faster than ILS in terms of computational time, but the quality of solutions is slightly worse since only 2 out of 4 global optima are found whereas ILS finds all the four. As for c_2 minimization, both BRKGA and ILS can find all 4 global optima in a reasonable amount of time. However, ILS runs much longer than either CPLEX or BRKGA. Second, since CPLEX cannot handle the larger instances of troyes-v58 and troyes-v102 (respectively larger than the generated 7x7 and 10x10 instances), only the two proposed metaheuristics are tested with results reported in Tables 4.22 and 4.23. While consuming much more time, ILS leads

BRKGA in terms of the solution quality in c_1 optimization, whereas BRKGA can find the same objective values as ILS on 7 out of 8 instances when minimizing c_2 .

Instance	CPLEX					BRKGA			ILS		
	c_1^*	c_2	t(s)	LR	Gap(%)	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
troyes-v24-b1	210913	21	12.90	202149	4.16	215719	21	2.50	210913	7	5.91
troyes-v24-b2	214475	18	30.36	195456	8.87	218821	19	2.43	214475	11	5.12
troyes-v24-b4	225282	19	35.45	202986	9.90	225282	19	2.14	225282	8	3.85
troyes-v24-b6	246259	10	40.30	210191	14.65	246259	17	1.70	246259	10	3.78

Table 4.20: c_1 minimization on troyes-v24 instances

Instance	CPLEX					BRKGA			ILS		
	c_1	c_2^*	t(s)	LR	Gap(%)	c_1	c_2^*	t(s)	c_1	c_2^*	t(s)
troyes-v24-b1	286280	0	0.97	0	0	286280	0	0.97	286280	0	4.79
troyes-v24-b2	275796	1	2.03	1	0	275796	1	1.01	275796	1	7.21
troyes-v24-b4	289461	1	1.87	1	0	289461	1	0.80	289461	1	4.45
troyes-v24-b6	350430	1	1.92	1	0	350430	1	0.84	350430	1	4.56

Table 4.21: c_2 minimization on troyes-v24 instances

Instance	BRKGA			ILS		
	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
troyes-v58-b1	2082930	24	50.31	2061660	43	127.20
troyes-v58-b2	2079430	24	53.52	2064860	43	119.33
troyes-v58-b4	2094690	13	72.12	2072890	41	112.18
troyes-v58-b6	2141170	20	77.10	2108600	49	215.11
troyes-v102-b1	6535240	79	490.82	6452160	47	824.71
troyes-v102-b2	6622340	33	466.19	6469860	46	840.28
troyes-v102-b4	6718200	75	594.92	6601810	39	948.22
troyes-v102-b6	6895030	45	499.74	6643740	51	694.14

Table 4.22: c_1 minimization on troyes-v58 and troyes-v102 instances

Following the same testing strategy, ϵ -constraint method and the other three metaheuristics BRKGA, ILS and NSGA-II are used to generate Pareto fronts on small instances of troyes-v24 while the larger instances are tested only by the metaheuristics. Table 4.24 shows the comparison of the bi-objective optimization performance of the four methods. The exact method consumes a relatively longer time compared to the similar

Instance	BRKGA			ILS		
	c_1	c_2^*	t(s)	c_1	c_2^*	t(s)
troyes-v58-b1	2172160	0	23.90	2172160	0	37.88
troyes-v58-b2	2184720	0	24.87	2184720	0	37.17
troyes-v58-b4	2185640	2	13.99	2199210	0	39.76
troyes-v58-b6	2259330	0	16.61	2259330	0	42.96
troyes-v102-b1	7372890	0	73.05	7372890	0	156.72
troyes-v102-b2	7390380	0	64.13	7390380	0	154.19
troyes-v102-b4	7600060	0	56.75	7600060	0	155.61
troyes-v102-b6	7677250	0	57.34	7677250	0	153.24

Table 4.23: c_2 minimization on troyes-v58 and troyes-v102 instances

instance set of 5x5 multigraphs. The reason might be that the real networks are more complex than the generated ones with real arc distances though the network size stays small. The metaheuristics constructed high quality fronts. Among them, BRKGA has the best performance. However, unlike the behavior observed on the results for bi-MRND instances, BRKGA does not ensure a good spread of the fronts on these real instances. ILS got high quality Pareto fronts in terms of both hypervolume and spacing within a much shorter time than 5x5 bi-MRND instances. As for the Pareto fronts produced by NSGA-II, they are generally worse than bi-BRKGA and bi-ILS in terms of the hypervolume, whereas the quality of solutions distribution is leading. On the instances with 58 nodes with results reported in Table 4.25, ILS still has a good solutions quality along with a huge time consumption. It can find Pareto fronts very close to the best known fronts combining the solutions found by all the metaheuristics. In addition, BRKGA dominates the other two metaheuristics on the Pareto fronts spread and it performs generally better than NSGA-II in terms of HV. Since NSGA-II uses much less time than the other two methods, it should be tested in the future whether its performance can be enhanced by running a longer time. Example of combined Pareto fronts can be found in Figures 4.17 and 4.18 for troyes-v24-b2 and troyes-v58-b6 instances, respectively. Readers are also referred to Appendices C and D for more images.

4.4.5 Conclusion

Under the single-objective context (URND and MRND) with respect to c_1 and c_2 criteria, numerical experiments are performed on a set of generated instances up to 144 nodes and 264 arcs for URND and up to 81 nodes and 288 arcs for MRND. Methods are also applied on instances extracted from the real network of the French Troyes city, Troyes, with 58 nodes

Instance	ϵ -constraint				BRKGA				ILS				NSGA-II	
	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV
troyes-v24-b1	8	0.86	0.36	108.48	9	0.85	4.81	81.66	6	0.85	0.83	68.05	8	0.86
troyes-v24-b2	11	0.86	0.34	263.28	12	0.85	1.49	99.65	7	0.84	0.79	61.07	13	0.78
troyes-v24-b4	7	0.88	0.35	99.86	8	0.87	2.21	97.15	7	0.86	0.48	51.73	13	0.66
troyes-v24-b6	9	0.86	0.47	290.96	9	0.83	2.17	51.33	8	0.85	0.56	45.76	8	0.84

Table 4.24: Performance comparison of Pareto fronts on troyes-v24 instances

Instance	BKS		BRKGA				ILS				NSGA-II			
	Q	HV	Q	HV	SP	t(s)	Q	HV	SP	t(s)	Q	HV	SP	t(s)
troyes-v58-b1	17	0.90	15	0.77	1.04	2230	17	0.90	1.10	4754	21	0.50	7.35	1183
troyes-v58-b2	16	0.94	19	0.84	1.14	1553	15	0.94	20.29	3413	27	0.61	8.21	1205
troyes-v58-b4	17	0.88	14	0.76	2.00	2525	15	0.88	2.22	3696	13	0.40	8.08	1155
troyes-v58-b6	20	0.91	24	0.81	0.76	2730	19	0.91	1.08	3121	26	0.48	5.18	1138

Table 4.25: Performance comparison of Pareto fronts on troyes-v58 instances

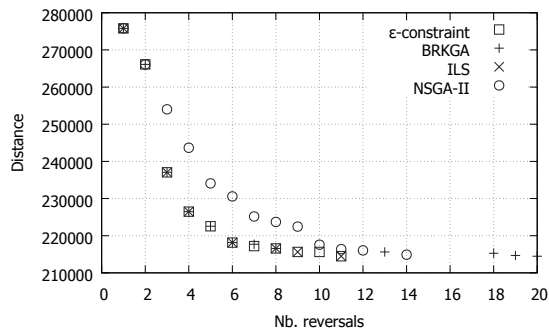


Figure 4.17: Pareto fronts for the troyes-v24-b2 instance

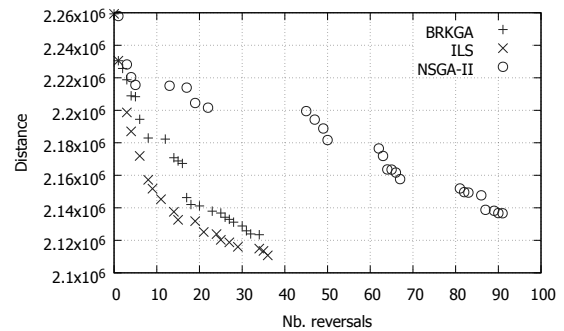


Figure 4.18: Pareto fronts for the troyes-v58-b6 instance

and 155 arcs. Results show that the limit for CPLEX on URND instances is around 25 nodes and 40 arcs. For MRND, its limit is around 36 nodes and 120 arcs. On the other hand, the metaheuristics are able to solve instances with more nodes and arcs. Comparing to BRKGA, ILS seems more robust for both URND and MRND, even when the number of disruptions increases.

As for bi-objective optimization, computational tests are performed on grid instances with up to 64 nodes and 112 arcs for bi-URND and up to 64 nodes and 224 arcs for bi-MRND. The four methods are also applied on instances built from the real road network of Troyes. Results show that ILS is able to produce high quality Pareto fronts in both bi-URND and bi-MRND. In addition, it is less dependent on the network configuration, either the topology

or the number of disruptions. Yet, BRKGA is a solid alternative, even if its results stay a bit behind. NSGA-II, in its current setting, does not converge well towards the optimal or best Pareto front. It would be relevant to develop the ILS for other problems in order to check if such results confirm.

Chapter 5

Application : Optimal traffic Deviation System

Urban traffic management is a complicated issue in the real world due to the growing demand of travels and the insufficient capacity of networks, as stated earlier in this document. When disruptions happen on a critical link of a road network, it is important for traffic manager to react rapidly and adequately in order to maintain the circulation functioning, which may impact concerned areas. Based on the road network of Troyes city in France, there are over 8000 planned disruptions in 2014, nearly 22 per day. Managing this network is even more difficult because the downtown city is a historical center where roads have a low capacity and are almost one-way. A great challenge is to choose alternate paths for every disruption. Since drivers willing to pass a disrupted road are sent to another road which has its original traffic load. Thus, a bad choice may cause congestion or an uncomfortable experience of the drivers. Therefore, ODS is designed to help the decision-making of traffic managers from different viewpoints and in a short time. Under the meta-project TOAST, the design and development of the first ODS prototype are launched in 2014 thanks to the collaboration with *PC-Régulation* of Troyes. The corresponding work in Subsection 1.2.5 is ranked 13th out of 107 projects in TRA-Vision 2016 Student competition. By a financial support from the Innov'Action project “*AGIR*”, the further research and development have been carried out.

A functional prototype is firstly designed to justify the idea and workflow and the system in production is then developed oriented to professional uses. Major features are mostly accomplished while fine tuning and maintenance are still required in order to deliver progressively the real product. As the main functionality of ODS, the deviations are computed according to one or more optimization criteria. The following objectives are focused: the bypass distance, the number of road sections and the number of traffic signals.

These objectives are handled individually. A multi-criteria schema can also be imagined, whereas it is not the priority in this stage. Currently, a number of modules are already working: (i) an interactive customized GIS based on the network of Troyes; (ii) a GUI; (iii) computing deviations for a given blockage minimizing either the travel distance, the number of employed road segments or the number of traffic signals; (iv) network information administration and adjustment, especially the positions of traffic signals and traffic calming features; and (v) monthly monitoring of disruptions.

5.1 Requirements analysis

As a decision-support system, ODS is supposed to help decision makers (probably a hierarchy team with leaders, managers and operators, etc.) in daily tasks of disruptions management. Its key feature is the automatic computation of deviations while users can always get involved by manually create and edit deviations. The system is able to retrieve the planned blocking events from the given databases of a city hall, while unpredictable disruptions such as traffic accidents can also be manually added to the system. Moreover, information is presented to users in an intuitive way, including the blocking events, deviations and network elements, such that a user can really focus on different aspect of making the decision instead of searching different information everywhere in the system. Regarding the engineering, ODS is developed and maintained under several mature application standards in the Web context following the Scrum for the agile software development.

5.1.1 Functional requirements

The functional requirements of ODS concern mostly the features proposed to users for managing blocking events as well as the network information. They may involve every layer in the software design: the presentation to end users, application and services, business logic and data persistence. The end users of ODS are decision makers in general, referred as “USER” in this report. Moreover, there may be different user roles, such as traffic engineers (non-administrator) and administrator. They have specific needs when using ODS. Hence, the requirements are also addressed from their viewpoints. Considering the background of the collaborations and to concentrate on the key features of ODS, several requirements are based on the existing services of the road network of Troyes.

- ODS must be able to connect to the disruption databases of the city hall of Troyes, to retrieve the information of blocking events and corresponding contacts (such as a road maintenance company, a removers firm, the organization of a parade, etc.).

- ODS must be able to store and update the data gotten from the disruption database.
- ODS must be able to visualize disruptions and deviations on an intuitive map of the road network of Troyes.
- USER must be able to create unpredictable disruptions by offering its label, starting and ending dates, a contact (if exists) and the blocked road segments.
- USER must be able to edit a disruption by modifying its contact and ending date (*i.e.* to extend its duration).
- USER must be able to launch an automatic computation of deviations for every disruption.
- USER must be able to edit a deviation by changing its path.
- USER must be able to create deviations for a disruption by indicating detour paths.
- Non-administrator USER must be able to select deviation(s) for a disruption and submit to administrator.
- Administrator USER must be able to validate or reject the deviation selection for a disruption.
- ODS must be able to consider the strong connectivity of the network impacted by disruptions and to propose adapted solutions for different types of vehicle: car and truck (bus is another type but it is not considered at this moment due to possible complex changes of bus stops).
- ODS can propose deviations with insufficient capacity, but a warning must be highlighted to USER.
- ODS must be able to compute optimal deviations with respect to several criteria: the minimization of the distance, the number of segments and the number of traffic signals (more criteria are also identified and planned for future stages).
- ODS must be able to classify disruptions according to their status: (i) not handled, (ii) handled but not yet validated, (iii) handled but rejected and (iv) validated.
- Within a calendar, ODS must be able to illustrate the unhandled disruptions according to their starting dates, so that USER shall be able to have a vision of the tasks.

- ODS must be able to retrieve and store the network information from OpenStreetMap, such as the locations of traffic signals, stop signs and speed reducers.
- USER must be able to monitor the locations of traffic signals, stop signs and speed reducers.
- USER must be able to add to or remove from ODS a traffic signal, a stop sign or a speed reducer by offering the location on the map.
- ODS must be able to perform a qualitative analysis of deviations according to the network information, for instance in terms of air pollution, noise or driving experience.
- ODS should store user activity data and perform statistical analysis (probably by some machine learning approaches) in order to enhance the strategies for computing deviations.
- ODS should connect to the database of Gertrude's system, deployed in the traffic control center of Troyes, in order to obtain real-time traffic flow data. In the case of addressing unpredictable disruptions, these data are especially important so as to take into account the real-time network status.

5.1.2 Nonfunctional requirements

The analysis of nonfunctional requirements is mainly divided into five parts: the usability, reliability, performance, supportability and interfaces. The main purpose is to clarify the needs of the User Experience (UX) and different aspects in the development, deployment and maintenance of ODS.

5.1.2.1 Usability

- ODS must be able to synchronize regularly the newly created blocking events from remote databases.
- USER must be able to synchronize the newly created disruptions on demand.
- ODS must be able to indicate the starting and ending point of deviations, with respective brief descriptions of the paths.
- ODS must be able to show all the performance values, either quantitative or qualitative, of each deviation, in order that USER has an intuitive vision.
- ODS must be able to warn whenever a deviation has an insufficient capacity.

- Administrator USER must be able to compare selected deviations for a disruption in an intuitive way, for instance a polar graph that combines the performance values of every deviations.
- USER must be able to switch the look-and-feel of the map between a “day” theme and a “night” theme, like what does a GPS navigation device.
- ODS must provide a user manual.

5.1.2.2 Reliability

- Automatic unit tests must be carried out for the components of the computing core of ODS with a code coverage $\geq 80\%$.
- Automatic integration tests must be carried out for the modules of Web services of ODS.

5.1.2.3 Performance

- Data synchronization must not block the main process of ODS.
- Automatic deviation computation must be performed in a reasonable time.
- Web client of ODS should be responsive to different size of viewport.
- USER must have a smooth loading view whenever a relatively long procedure is executed.

5.1.2.4 Supportability

- Web client of ODS must support modern browsers (of recent versions) as many as possible.
- Web services of ODS must be independent to operating systems and able to deploy on any platform with a Docker container support.

5.1.2.5 Interfaces

- ODS must have an independent module of data access able to make connections to different heterogeneous databases or services of traffic information.
- Web services of ODS must be able to support mobile applications under authorization.

5.2 ODS architecture

Architecture is the high level coordination of every single part of the system, which shows clearly the goals of the system and provides guidelines to the implementation. On one hand, it should classify the modules considering the current requirements such that the features are divided into small parts of functionality instead of being centralized in some domains. This decentralization is already a mature concept in software engineering that helps to simplify the software development and maintenance as well as the project management. On the other hand, the architecture of a software should also anticipate the potential extensions in the future. In practice, choosing a balanced vision is always challenging. If the architecture includes too many needs that are not important at the beginning of the design, it will make the implementation much more complicated because one should consider a lot when developing a small function and cannot focus on key elements. In turn, it is hard to apply any concept of agile project management because every development cycle would take a long time and any delivery cannot be ensured. It is called an *over-design* or *over-engineering*. If the future updates are not well foreseen in the architecture, problems may arise when adding new features since there might be some concerned functionality which would have been but was not extracted to an independent component or module. This will surely increase the amount of work and may involve an additional refactoring which is often difficult and expensive.

The architecture of ODS concentrates on the major features that can prove its concept and provides (at this moment) the extensibility only on computing solvers and data sources. Figure 5.1 illustrates the design of the entire system.

ODS is divided into four layers. The presentation layer comprises generally, just as its name implies, all the modules concerning the User Interface (UI) where ODS interacts with the end users. The logic layer consists of different deviation computing cores. It is related to the presentation layer via Web services in the service layer to collect required data from user, to performance the computation and to return the results shown to user. Data modules are classified in the data access layer. They are in charge of collecting data inside or outside ODS, converting the data when needed, storing the data and abstracting the fundamental data manipulations such as to create, read, update and delete (CRUD) data. This layer may provide data supports to the solvers for computation or directly to the UI for visualization via Web services.

Beyond these four layers of functional modules, there are three supplementary elements: the activity metadata is composed mainly by the log across the four basic layers for debugging and tuning uses; the security configuration protects the Web services and confidential data from being abused; the tests ensures the functioning of the modules of ODS which play a

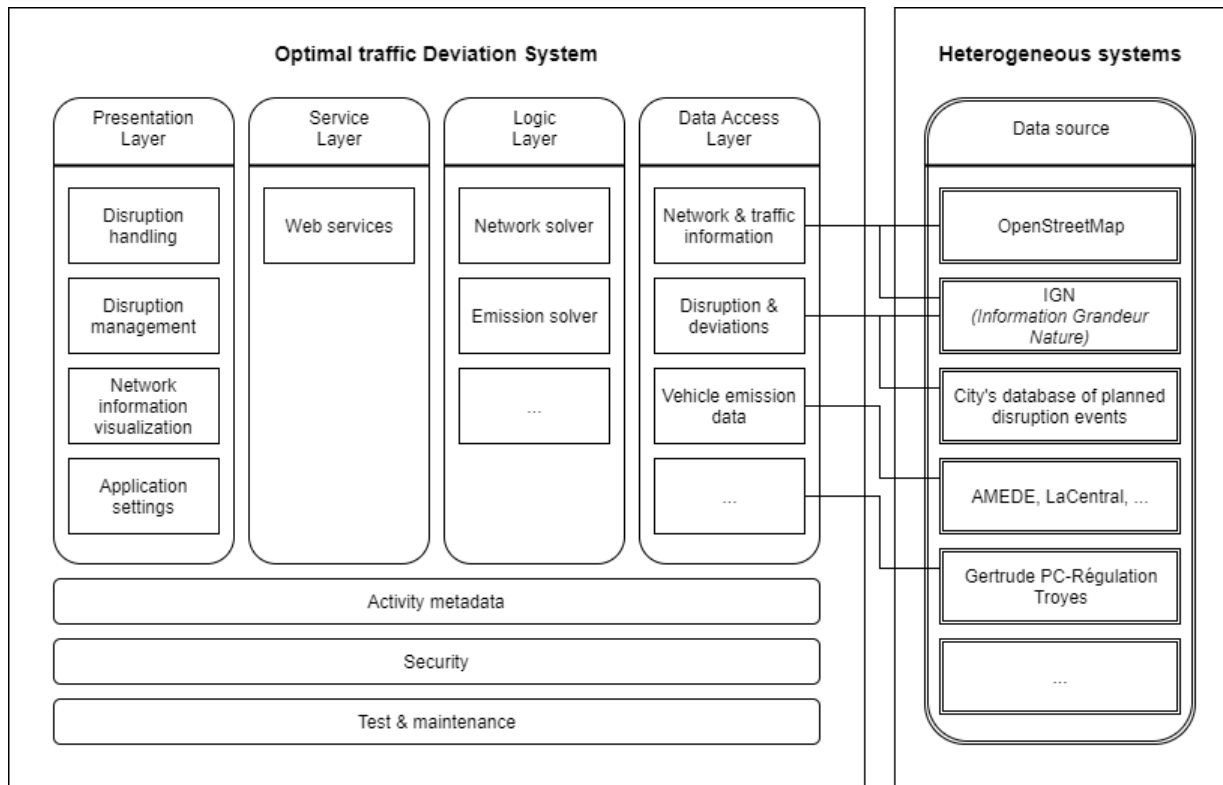


Figure 5.1: System architecture of ODS

key role in the development, deployment and refactoring.

In addition, the architecture of ODS considers the connection to heterogeneous systems, which are mostly drivers data sources at this first stage of design. As described previously, ODS needs network data from OpenStreetMap and disruption data from the databases of Troyes's city hall. To complete the geographical data, especially the data of addresses, it is also possible to integrate the databases of IGN (*Institut National de l'information Géographique et forestière* or *Information Grandeur Nature*), a hyper organization of geographical information of France. Moreover, the qualitative analysis of deviations may require additional information, for instance, the emission of carbon dioxide of vehicles according to the type (vehicle producer and engine model), the distribution and usage of different types of vehicles in the city, etc.

Note that implementation details are not presented in the architecture, which will be described in the next three subsections regarding the human-computer interaction, database and computing core. Technologies and practices applied in the development will be discussed.

5.2.1 Human Computer Interactions

HCI investigates in general the way that people exchange information with artifacts on computers in order to collect required information from users for some computational purposes and to show the results back to users. From a software's point of view, HCI is mainly represented by UI and UX. The former is a concrete field about what and how things are presented to users, while the latter focuses on making UI intuitive and facilitating the usage of the product. All the concerns should concentrate on making the software simple to use so that users can easily understand what is proposed to solve their problems.

As for ODS, its end users are mostly urban traffic managers, who are supposed to have a professional background of the road network and disruption management while probably being confused by theory of graph and algorithms to compute alternate paths, to check the strong connectivity or to calculate the CO_2 emission. Thus, UI developers should centralize most of the input and output operations on a map view such that users can have a clear vision on their problems and the proposed solutions. There are four major aspects to address HCI of ODS:

1. General layout and feature organization: to study and propose the appropriate workflow.
2. Cartography: to integrate an interactive view of OSM.
3. UI element: to design the logo, icon set, color scheme and fonts.
4. Visualization: to carry out the methods for showing information such as disruptions, deviations and traffic elements.

5.2.1.1 General layout

According to the requirements related to the interaction between users and ODS, a preliminary study is done by designing some conceptual screens of the GUI on mockups. It is proposed to a few target users in *PC-Régulation* of Troyes helping to validate the general layout regarding their antecedent workflow of disruption management. As the base of HCI which should be respected during implementation, the mockups are also confirmed by concerning actors of ODS comprising the developers team and the project supervisors.

Figure 5.2 is the designed home page of ODS, on which four major activities are classified: to create deviations, to supervise disruptions and their deviations, to configurer the application (settings) and to consult the user manual. A sidebar of shortcuts is also considered and settled on the left of the screen. As a UX consideration, it can be folded up

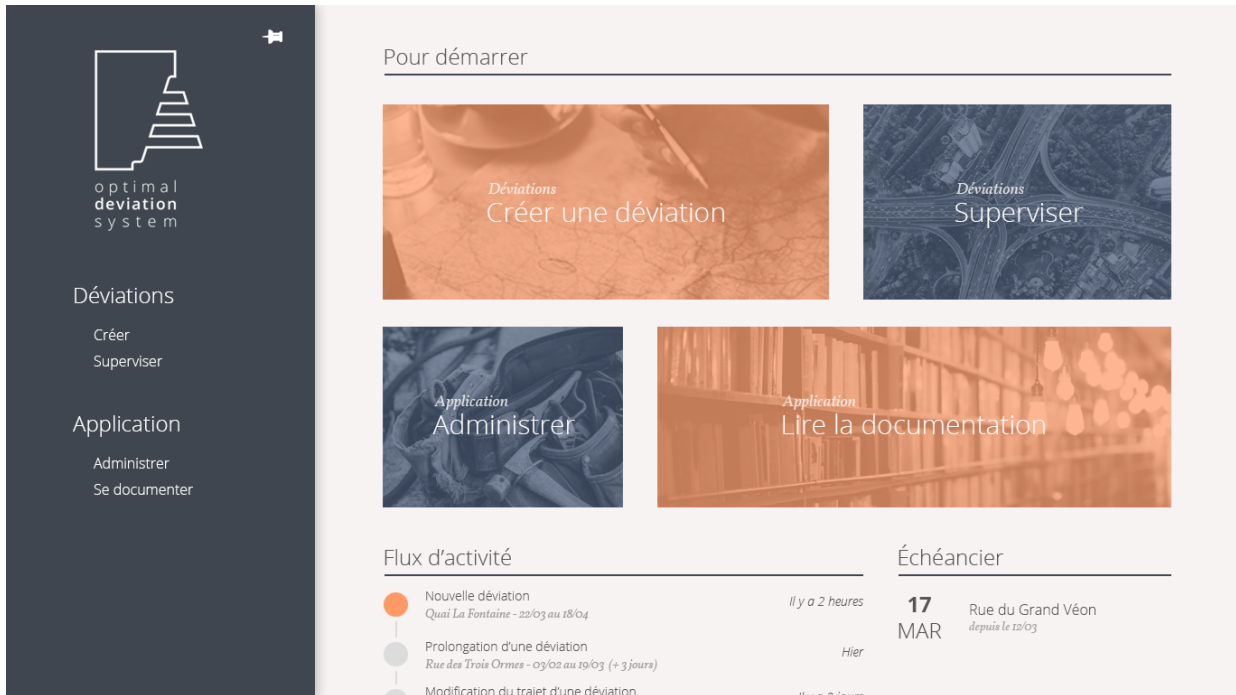


Figure 5.2: Mockup of home screen with organization of major activities

to save the space for the activity view and allows from deep inside an activity to navigate to another activity. Moreover, a block of summaries is also planned on bottom right of this main screen, in order for multiple users to have a quick preview of actions performed by colleagues and of urgent disruptions requiring to be handled rapidly.

Besides, the general layout of deviation management screen is also proposed, as shown in Figure 5.3. The main feature is the map view with markers indicating the locations of the ongoing blocking events which have already been handled with validated deviations. There is also a list of details to the right with, for each disruption, the street name it concerns and a brief summary of its deviations. Details are shown when selecting a disruption either from the map or from the list.

5.2.1.2 Cartography

As the most important component of ODS, the cartography is used to exchange a large amount of information with users related to disruptions, deviations and network elements. It is associated with not only a map view on UI, but also the underlying network data for computational purposes. The current cartography solution in ODS is based on OSM since it is an open source GIS having a stable database with public access and a integration-friendly ecosystem. It also serves a Web application with a number of features such as searching a place and routing between several locations. In this application, the map tiles technology

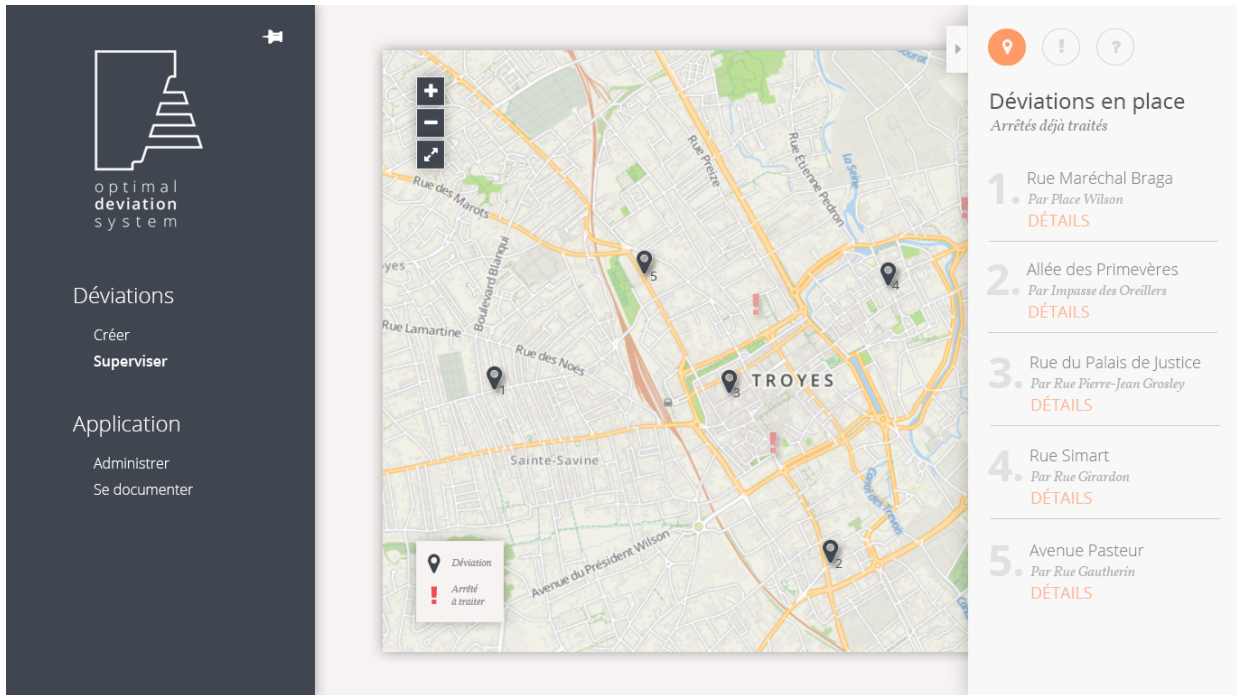


Figure 5.3: Mockup of deviation management with a map view and list of blocking events

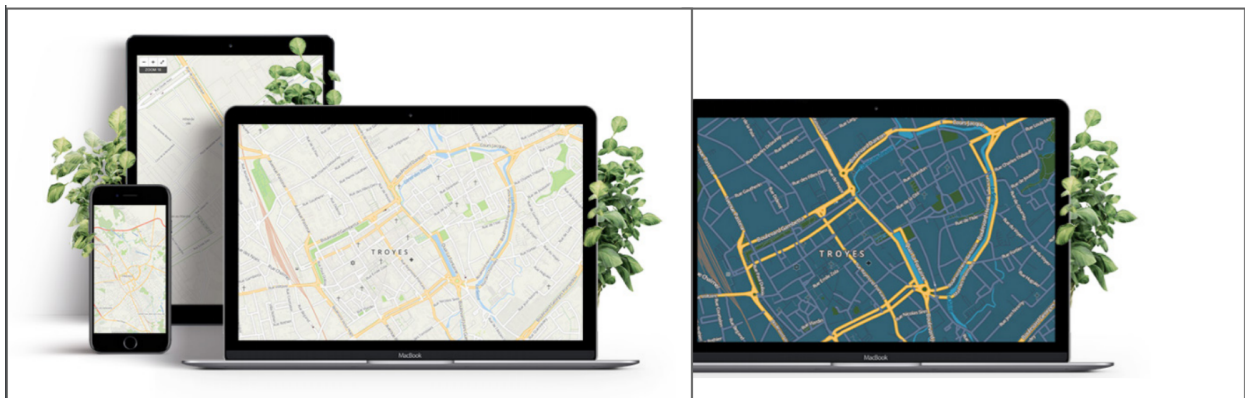


Figure 5.4: Conception of responsive map views with themes of day and night

(cf. Figure 5.5) is applied for showing the map. Its main idea is that the whole map of the Earth can be divided into small images of a high resolution that can be loaded in parallel. Besides, only necessary tiles are loaded depending on the current view spot of the user and on the level of zoom in order to avoid wasting time and memory. Furthermore, it is possible to generate our own tiles using the Python library *MBUtil* by providing decoration scripts in *CartoCSS* on *Shapefiles* of network information. These tiles can hence be distributed within ODS for all types of uses. However, unlike the other famous GIS *Google Maps*, OSM does not have an official support for a ready-to-use map component (Web and mobile widgets are available for *Google Maps*). To integrate a map view similar to the application, one can use the third-party Javascript library called *Leaflet.js* which support the dynamic loading of

map tiles.

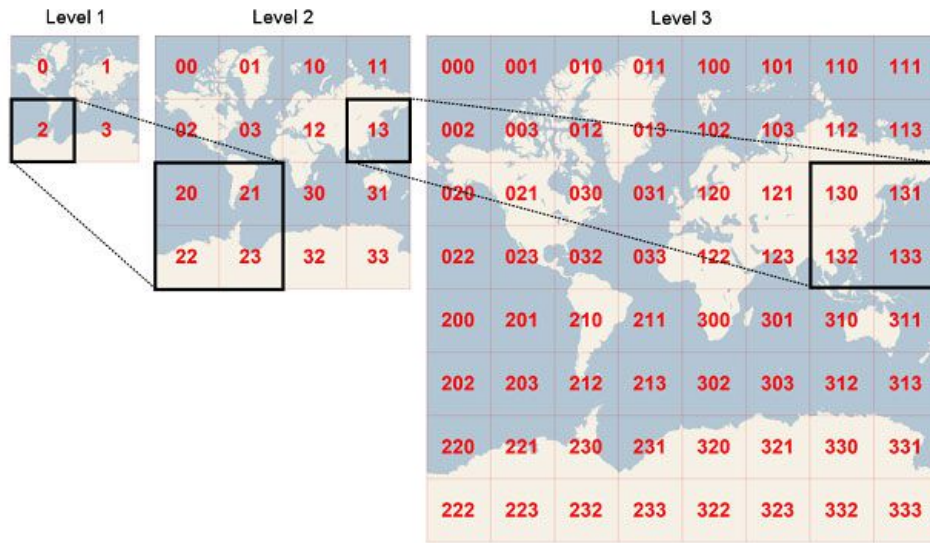


Figure 5.5: Example of map tiles with map projection and tile coordinates¹

5.2.1.3 UI elements

Apart from the global conception, another important goal of a GUI is to express the visual identity of the corresponding application by some concrete UI elements like the logo, icons, colors and fonts. The combination of all these details should not only be esthetic, but also fit the target software with reflection of its purposes.



Figure 5.6: ODS logo

The design of the first ODS logo, as shown in Figure 5.6, is inspired by traffic cones which are often placed in the area of blocking road works. The main illustration is a closed line draws the stroke of a traffic cone, which can also represent a blocked street to the left and its deviation to the right (which tends sometimes to be complicated with many curves in downtown cities).

The color scheme (see Figure 5.7) is established with a special care to avoid conflicting with the themes of the cartography. Also with an additional consideration, two free fonts (see Figure 5.8), OpenSans and Vollkorn are selected mainly for their elegance and adaptability across platforms.

¹Credit to MSDN with origin at <https://msdn.microsoft.com/en-us/library/bb259689.aspx>



Figure 5.7: Palette of colors used in ODS



Figure 5.8: Fonts used in ODS

5.2.1.4 Visualization

The visualization is an efficient tool in HCI. The principle is to present information on the GUI in a simple and intuitive way so that users can understand what is requested and what is responded by the software. In a complex system, the visualization is not an easy task since there is often a huge amount of data to be shown at a time.

With the aforementioned technologies of the cartography, the map view of ODS is able to highlight paths with custom decorations, such as the color, the line style and the line width. Figure 5.9 illustrates the screen visualizing a deviation of a blockage. The track of the disruption is drawn by dotted lines in red color. The path of the deviation is presented with green lines and two markers to indicate its starting and ending intersections. Text description is also available for every deviation route. In addition, all deviations for the current blockage is listed below the map view with of each a summary of performance criteria: the dedicated vehicle type, whether it has an insufficient capacity, the distance, the number of intersections and the number of traffic signals. Switching deviations in this list triggers rendering of its path on the map.

As determined in the requirement analysis, the administrator should validate deviations selected by agents of traffic management. This feature also involves the visualization since users need to compare relevant deviations for decision making. According to the current criteria taken into account: the distance, the number of intersections and the number of traffic signals, a radar chart in three scales is presented in such a way that the pros and cons of every deviation can be intuitively perceived, as shown in Figure 5.10. Moreover, a score between (0, 1) is calculated for each deviation by aggregating the three normalized objective values. It can also be a comparison measure of the global performance of each deviation.

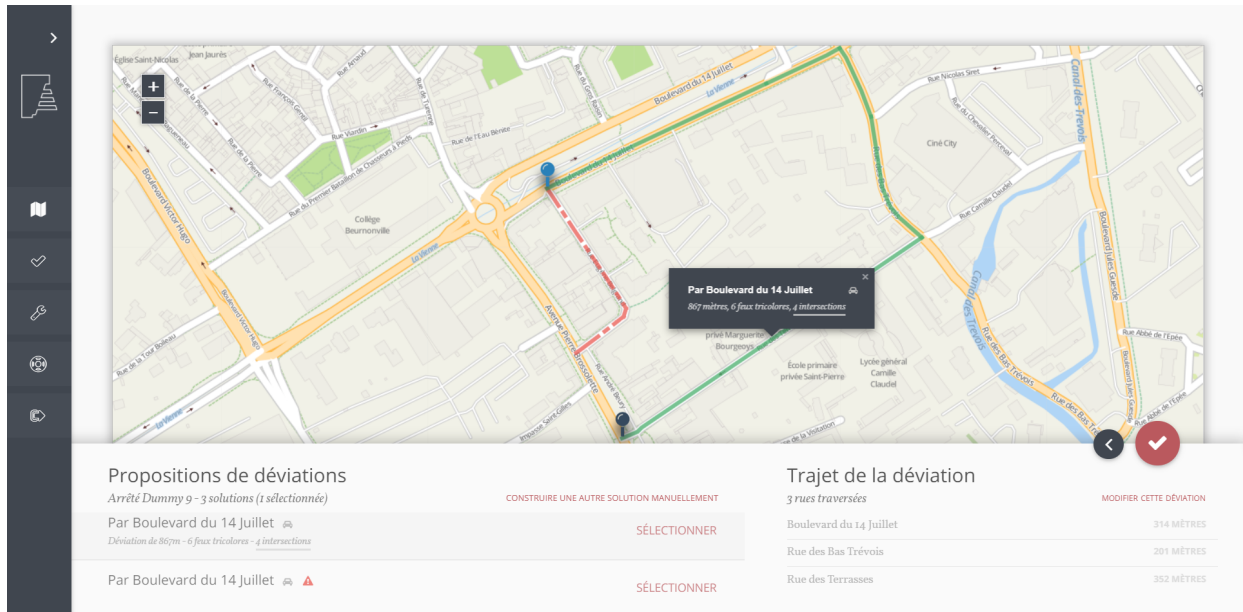


Figure 5.9: An example of deviations visualization on the map view

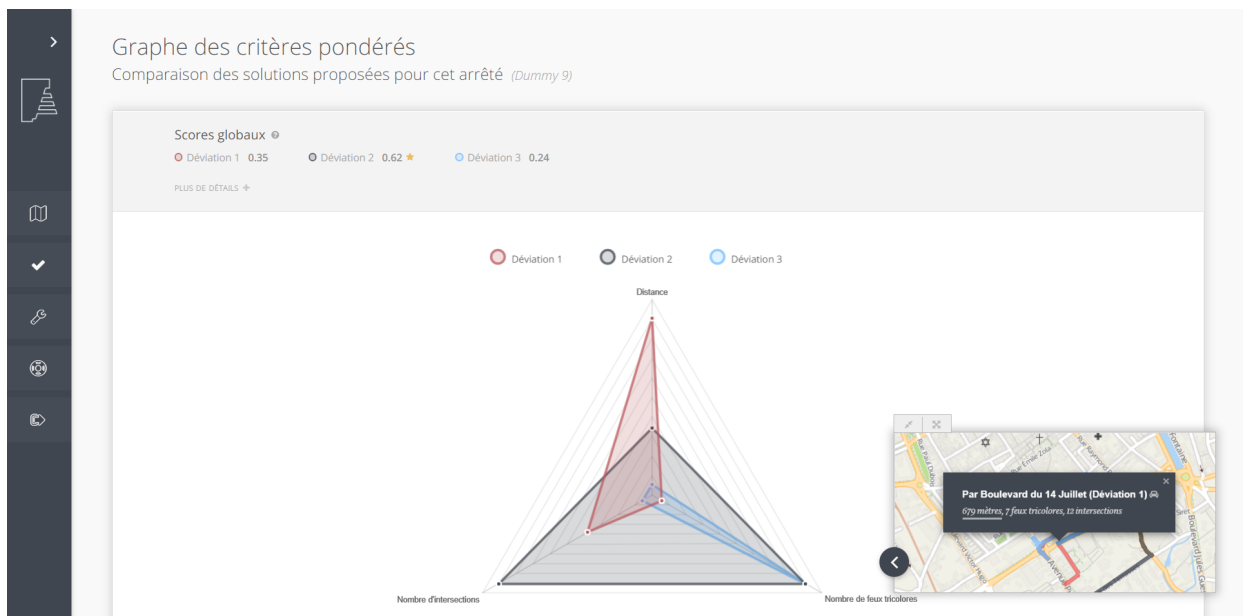


Figure 5.10: An example of intuitive comparison of deviations

The visualization concerns the network elements as well, especially for showing the traffic signals, the stop signs and the speed bumps. Originally, these objects are used for qualitative analysis of deviations. Meanwhile, since the information of locations are collected from OSM, some locations may be incorrect or even missing. Thus, network managers are also involved not only to have a global vision of the locations but also to maintain the data. They can add or delete a traffic element just by clicks on the interactive map. Figure 5.11 is a screenshot of the way the traffic signals are visualized on the map view.

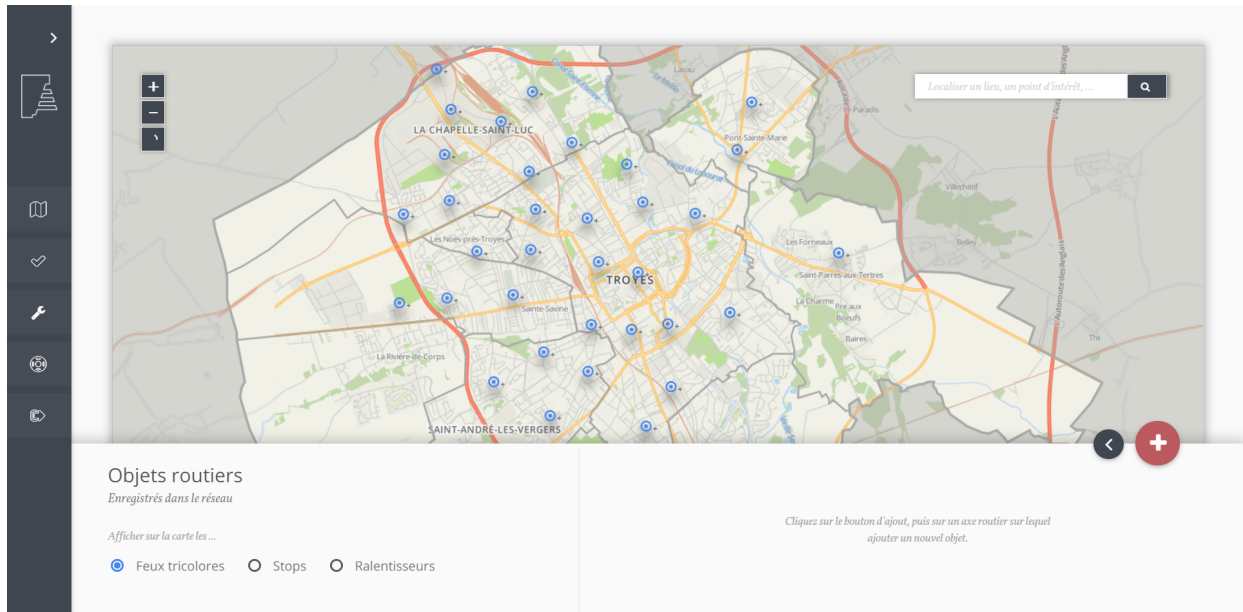


Figure 5.11: Visualization of traffic signals on the map view

5.2.2 Database

The goal of ODS database is to store the network information, the disruptions and the computational results (deviations). Since some of these data are obtained from heterogeneous systems, their structure should be adapted especially for the uses in the computing core.

The geographical network data of Troyes are collected from OSM. They are stored in the database for the visualization use in HCI. Furthermore, OSM also provides additional network information, such as street names, the speed limit and traffic signals, which are useful for ODS. However, the data objects “Node” and “Way” cannot be directly converted into graph structures for computing purposes. Hence, a pre-processing module has been developed in order to cut ways into arcs and merge transitive nodes. Then, the resulting data are stored separately.

Since ODS is connected to the database GEODP of the Troyes city hall, the disruption data can be obtained. In this database, the geographical information is not specified for disruptions. Instead, the impacted area is represented by addresses. Thus, ODS needs to identify the geolocation of every disruption in order to fit computational requirements. To do so, another module is developed using the Jaro-Winkler distance (Winkler (1990)) to analyze the text similarity (Gomaa & Fahmy (2013)) between the addresses and street names in the data from OSM.

With the analysis and specifications, the structure of ODS database is conceived as illustrated in Figure 5.12. The diagram is constructed using the classical entity-relationship model. Objects are classified into three domains:

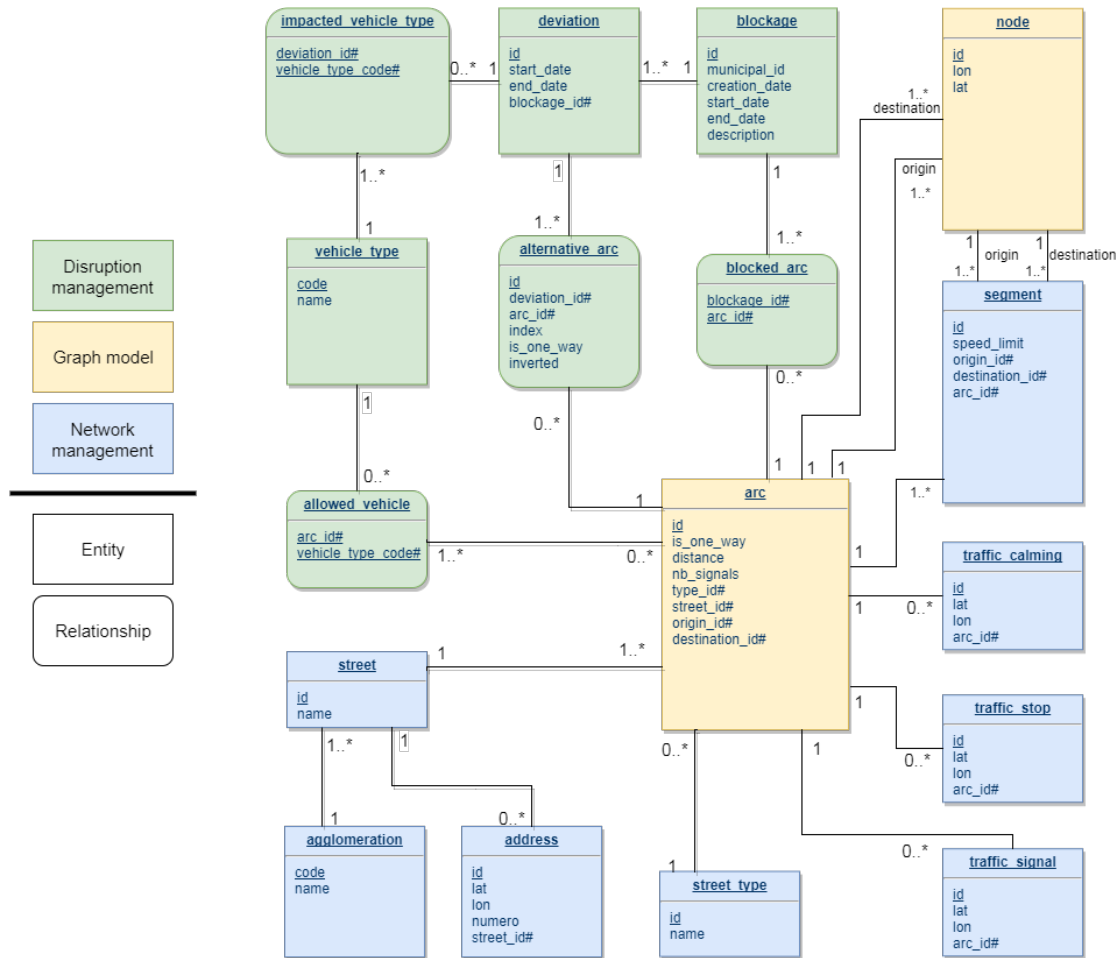


Figure 5.12: ODS database schema

Another long term issue is the data update from heterogeneous systems. Updating disruptions from the database of GEODP is not too problematic because the major change of the database is adding new disruptions and a regular polling from ODS can accomplish the task. However, since OSM is open source, contributions from its community may change the network. Especially when a node is deleted, it is not very obvious to adapt our database. Till the moment of redaction, the problem is yet left to be handled and searching an update solution is still in progress.

5.2.3 Computing core

The computing core of ODS supports the major functionality for proposing deviations to decision makers. By the aforementioned system architecture, it consists of various solvers and belongs to the business layer. The major component is the network solver which receives disruptions information and returns several deviations combining the context of the given network. It is designed with a loose external coupling to other modules of ODS, especially

in the routines for the deviation computation. Required data are always passed through public interfaces and the calculation works on any formatted data. Thus, the solver is highly reusable when the network context changes and its side effect to other features of ODS is minimized.

Given the graph modeling the concerned network and disruptions information, the network solver applies the algorithms studied in the scientific part of this thesis. They are adapted to the optimization objectives defined for ODS: the detour distance, the number of sections used in the deviation and the number of traffic signals. Hence, the algorithms use different arc costs corresponding to these criteria. Moreover, for each disruption, an origin node and a destination node are also given to the network solver in order to simplify the calculation. Besides, specific constraints for different vehicle types should be taken into account by the solver, for instance the height, the weight and the angle of turning. Variations and extensions of the algorithms are thus considered, whereas their application is still in progress.

By the way, a multi-layer graph model can also be applied to handle the specific arc constraints of distinct vehicle types. A dedicated subgraph can be created for each vehicle type with only the arcs that it is allowed. Hence, algorithms can use different layers of the graph to calculate deviations destined to different vehicle types. However, there are several problems left to be addressed for this idea, for instance the storage and subgraphs identification. Therefore, this feature is still conceptional.

The concepts explained in this subsection mainly concern the network solver for the deviation computation. Other solvers are also planned to be developed and to be integrated gradually in the computing core as plug-ins or post-processing modules, for instance the solver to compute pollution indicators for a given deviation. As the system grows, there is a strong possibility that the requirements change. The architecture will hence evolve in order to adapt to the needs, especially on the business layer.

5.3 Addressing technological gaps

One major technical challenge concerning the ODS system is to obtain accurate traffic data, especially to reproduce the real flow of vehicles. With the technological evolution since last decades, sensors become the most important weapon to collect real-time information from a city, especially on an urban road network. However, those data may not be available in every network. In fact, some cities have a lot of modern sensors that can cover almost every single intersection of the network. They allows not only to count the number of vehicles on different road sections by different timescale, but also to track the path of every vehicle

or to deduce the congestion level at any given intersection, as it is the case in Beijing, China. Another example is Lyon, France, with a large number of sensors for vehicle counting, weather detection and pollution measurement. Troyes also possesses its own digital sensor network for traffic control purposes, but it covers only some of the critical intersections of the downtown city and it is not possible to access any traffic data of suburbs that contains important links as well. This is due to financial or environmental limits (insufficient investment or lack of public area). Thus, constraints and criteria based on real flow models are difficult to integrate in ODS for this moment.

Another issue is the consideration of the public infrastructure. There are many blockages in critical bi-direction road sections where detouring the traffic can be very complicated and can change driver experience a lot. According to the workflow of the manual design of deviations, alternating traffic is often a solution facing this situation. Usually, it is employed when one direction is completely blocked and drivers in the two directions share the other direction left open. In the original idea before the development of ODS, it is planned to be taken into account to propose deviations. However, it is yet difficult for the network solver to determine whether alternating traffic can be applied and to choose the positions to place temporary signals. Furthermore, disruptions can impact the public transport system on urban transportation networks such as bus and tram. Sometimes, bus lines have to be periodically modified. In this case, deviation managers should work with public transport engineers in order to figure out the modified routes and the places of additional bus stops to deploy. The implementation of such a process automated by ODS needs also a great effort, since it requires investigations on the modeling, algorithms and infrastructural information systems.

Last but not least, there is an important technological gap concerns the congestion. Some studies have tried to anticipate congestion by using prediction based models, the traffic simulation, mathematical equilibrium strategies, etc. (Institut national de la statistique et des études économiques (France) & Berthier (1998) and Buisson & Lesort (2010)). However, the congestion is still a phenomenon for further researches since some obvious solutions can produce even more congestion. We can mention the paradox of Braess (1968) in which a route is added to a network infrastructure and it generates more congestion, instead of improving the traffic. For now, we are planning to bypass this drawback by doing simulations of traffic flow to take into account different scenarios.

Chapter 6

Concluding remarks and future works

In the context of disruption management, we investigate the reconfiguration of urban road networks as the first part of this thesis. With the development of urban areas all over the world, this problem becomes critical since there are growing concerns in various domains, for instance the increasing traffic load facing the insufficient network capacity. The situation is more complex when there are disruptions on the network. They often interrupt the traffic at a certain level or even break the paths between some locations. Modification of the network layout by reversing arc directions is hence taken into account in this work as the breakthrough to the problem, since it may overcome the issue of broken paths and eventually reduce travel costs of drivers. Therefore, RND is properly defined based on the reversals of the road direction to rearrange a given transportation network. Depending on network properties, it is divided into two problems. URND handles particularly one-way road networks which is often the case in historical centers, and MRND can treat general networks mixing one-way and two-way streets. With respect to the concerned criteria, the total distance of all O-D pairs and the number of arc reversals, both single- and bi-objective optimization are considered (resp. bi-URND and bi-MRND).

This thesis involves many related fields of research such as the graph theory in terms of orientation, the arc reversals and the NDP, for which relevant studies have been carried out for decades resulting in the respective theories and methods. Using graphs to model road network, bi-objective mathematical formulations are developed as the starting point to address these two problems by introducing the constraints of disruptions and reversals to classical multi-flow based models. In addition, 2-directed multigraph is introduced as an efficient way to simplify the multigraphs in bi-MRND. By applying an ϵ -constraint method, these models are able to handle the problems in bi-objective version. Next, two metaheuristics, a BRKGA and an ILS are proposed to handle single-objective URND and MRND together with a DFS-based orientation heuristic. Following their functionality, they

can find high quality solutions in a very short time. Adaptations are then introduced in both methods in order to address bi-URND and bi-MRND. Moreover, a classical bi-objective framework NSGA-II is also used with respect to the specifications of bi-URND and bi-MRND.

Experiments were performed using simulated and realistic instances in order to validate the models and the proposed metaheuristics. Numerical results indicate that the models are able to solve the problems (using the commercial solver CPLEX) on small sized instances and on a few medium sized instances with up to 36 nodes and 60 arcs, whereas it is difficult for them to confront with larger instances. The metaheuristics have generally good performance and they can handle large instances with up to (not limited) 144 nodes and 264 arcs in URND, and with up to (not limited) 81 nodes and 288 arcs in MRND. According to the numerical results, ILS performs globally better than BRKGA for URND, whereas BRKGA works better than ILS for MRND. Moreover, an encouraging result was obtained with the ILS adapted for bi-URND and bi-MRND, since it performs better than the well known bi-objective metaheuristic NSGA-II. Thus, this idea could be applied for other problems in order to check if such results also verify.

Besides, the methods studied in this work are tested on instances generated from the real urban network of Troyes, France. Graphs are extracted from the geographical data on which random disruptions are added. Specific techniques are used to restrict the size of the graphs that are grouped into three levels: (i) 24 nodes and 43 arcs; (ii) 58 nodes and 155 arcs; (iii) 102 nodes and 193 arcs. The smallest instances can be solved by CPLEX for MRND and by ϵ -constraint method for bi-MRND, while the metaheuristics are run on larger instances. The methods are able to solve all these instances and their behaviors are quite similar comparing to the results obtained from the constructed instances.

The second part of this work consists of the design and development of ODS, a decision-support system that helps to compute alternate paths for disruptions and to manage the concerned road network. In the background of the collaboration with the traffic control department of Troyes city hall, the requirements of ODS as well as its architecture are determined and validated in order to guide the following development. The major features of ODS comprise the visualization of disruptions on a map view, the computation of deviations and the management of network elements such as the traffic signals, the stop signs and the speed reducers. The development of the software are divided into three phases concerning the HCI for interacting with users, the database for integrating information and the computing core for calculating deviations according to various constraints and criteria. During the thesis, my contribution to ODS relied on the design of the software architecture and the development of the computing core. Furthermore, the first version of the final product is released that combines the most important functionality and optimization criteria, for

instance, minimizing the detouring distance, the number of sections and the number of traffic signals. Other elements are planned to be integrated progressively in ODS in future stages of development.

A number of possibilities are opened for future works. Firstly, a polynomial time algorithm to minimize the number of arc reversals is still an open issue. Finding such an algorithm may help to handle bi-URND and bi-MRND faster. Then, there is still room to improve the performance of the metaheuristics, especially on the computation time of ILS. Also, the solution quality of BRKGA and NSGA-II might be enhanced by allowing more generations, while it involves much deeper tuning. In addition, more realistic constraints and criteria are being studied for the application on general networks, such as the traffic capacity, multi-modal flows and the scheduling of planned disruptions, in order to better approximate the real contexts of RND problems. As for ODS, feature tests should be organized in the traffic control center of Troyes, in order to validate the functionality of its first version. Several aspects can also be considered in the future, such as to develop other algorithms with different criteria (travel time, gaz emission, etc.), to integrate more data sources, to address real-time traffic load and to explore a machine learning procedure taking into account the activities of traffic managers, so as to improve the computation of deviations.

Appendices

Appendix A

Résumé étendu en français

Problèmes du réseau de transport urbain: modèles, méthodes et application

Chapitre 1 - Introduction

L'augmentation de la population dans des zones urbaines a un fort impact sur les infrastructures de réseaux de transport, qui risquent souvent d'être saturées. Ceci implique aussi une croissance importante du flux de trafic urbain. Or, les réseaux routiers ne peuvent pas être étendus indéfiniment. L'influence de la concentration de population devient de plus en plus problématique et la situation devient encore plus compliquée lorsqu'il y a des interruptions sur des axes, telles que des arrêtés municipaux ou des accidents de véhicules. Ces interruptions perturbent la circulation dans les secteurs concernés, voire cassent des chemins entre certains points. Il est donc important de trouver des approches efficaces pour les gérer au niveau stratégique et tactique.

Dans cette thèse, nous abordons le problème décrit ci-dessus sous deux angles. Le premier consiste en une recherche scientifique sur le « Road Network problem with Disruptions and connecting requirements » (RND). Le second est axé sur le développement d'un système d'aide à la décision pour le traitement des arrêtés. Nous autorisons la possibilité de renverser le sens de circulation d'une rue à sens unique pour pouvoir restaurer la connexion entre tous les points du réseaux et éventuellement réduire le coûts de déplacement pour des conducteurs.

RND est un problème d'optimisation combinatoire avec deux critères : la distance totale entre toute paire de sommets et le nombre de rues dont le sens de circulation est renversé. En fonction de la nature du réseau traité, soit unidirectionnel soit multidirectionnel, RND peut être spécialisé en deux sous-problèmes : URND et MRND. Le premier modélise

des graphes simples pouvant couvrir les cas des centres-villes historiques. De son côté, MRND utilise une modélisation des réseaux par multigraphes dans lesquels des rues à double sens peuvent être définies. Les modèles mathématiques sont proposés pour ces deux problèmes en se basant sur le modèle de multi-flot. Ensuite, deux métaheuristiques sont développées pour l'optimisation mono-objectif, un « Biased Random Key Genetic Algorithm » (BRKGA) et un « Iterated Local Search » (ILS). Des adaptations sont réalisées pour ces deux méthodes afin de pouvoir traiter les deux problèmes en bi-objectif (respectivement bi-URND et bi-MRND). Une méthode exacte multi-objectif « ϵ -constraint method » et une métaheuristique multi-objectif « Non-dominated Sorting Genetic Algorithm II » (NSGA-II) sont également développées. L'analyse de la performance des méthodes est basée sur la comparaison des résultats numériques provenant des expérimentations.

La partie applicative de cette thèse est consacrée à la conception et au développement du logiciel « Optimal traffic Deviation System » (ODS), un système d'aide à la décision pour gérer des arrêtés et calculer automatiquement des déviations correspondantes. À l'aide d'une représentation graphique de la cartographie, des arrêtés et déviations peuvent être visualisés. De plus, les ressources de différents composants sont gérées par la base de données. Celle-ci relie des systèmes hétérogènes comme OpenStreetMap et GEODP, contenant des informations géographiques et les enregistrements des arrêtés municipaux. Plusieurs contraintes et critères d'optimisation sont pris en compte par les algorithmes dédiés permettant de calculer des déviations. Ce travail s'inscrit dans le cadre du projet TOAST (Tactical Optimization Strategies to Adapt Urban Transportation Networks), en collaboration avec le PC-Régulation de trafic de la Ville de Troyes et l'entreprise de R&D Beijing YuanZhiTianCheng Co. Ltd. Il fait partie aussi du projet AGIR (système d'Aide à la décision pour la Gestion d'Interruptions Routières), porté par le PC-Régulation de Troyes et financé par la région Champagne-Ardenne. Ce projet a pour but de réaliser ODS et de tester son fonctionnement dans un contexte réel.

Structure du document

À la fin du chapitre d'introduction se trouve une liste des contributions réalisées au cours des années de doctorat. Le reste du document s'organise en quatre parties : la revue de littérature sur les travaux liés au RND, la définition et modélisation de bi-URND et bi-MRND, les méthodes pour résoudre les problèmes et la réalisation d'ODS. Les points clés de chaque chapitre sont soulignés ci-dessous : le chapitre 2 présente la bibliographie en relation avec deux domaines de recherche : la théorie des graphes et la conception de réseau (Network Design Problem (NDP)). Les études qui nous intéressent le plus concernent l'orientation des graphes, le renversement d'arcs ainsi que les modèles, méthodes

et applications dans le contexte des réseaux routiers. Elles sont essentielles non seulement pour la recherche sur RND mais également pour la conception des méthodes de résolution dans ODS. Bi-URND et bi-MRND sont formellement définis dans le chapitre 3 avec leur modélisation par graphe. Les modèles reposent sur la notion classique de multi-flot. De plus, nous définissons un multigraphe particulier, le 2-directed multigraph, afin de réduire les multigraphes généraux en graphes contenant au plus deux arcs entre chaque paire de sommets. Pour tester la pertinence des modèles, des instances abstraites sont générées et les expérimentations reposent sur le solveur commercial CPLEX. Ensuite, des métaheuristiques sont proposées pour pouvoir traiter des instances de taille plus grande, que les modèles n'arrivent pas à résoudre. Tout d'abord, nous adaptions l'algorithme de parcours en profondeur (Depth-First Search (DFS)) pour proposer une heuristique d'orientation des graphes. Elle est modifiée pour pouvoir générer davantage de solutions initiales différentes. Elle est ensuite utilisée dans les deux métaheuristiques BRKGA et ILS initialement conçues pour l'optimisation mono-objectif. Leur principe est expliqué ainsi que les adaptations apportées pour pouvoir passer en version bi-objectif. Toujours dans le cas de l'optimisation bi-critère, sont proposées une méthode exacte utilisant ϵ -constraint et la métaheuristique NSGA-II, très utilisée dans la recherche opérationnelle multi-critère. La discussion se termine par les résultats numériques des expérimentations sur plusieurs ensembles d'instances de tailles différentes, y compris des données provenant du réseau de transport de Troyes. Le dernier chapitre du mémoire consiste en la présentation détaillée d'ODS, depuis la phase de conception jusqu'au développement. Suite à l'analyse des besoins établie au début du projet et validée par l'ensemble des acteurs, l'architecture du système est décrite. Cette dernière est ensuite analysée d'une manière plus technique, notamment l'Interface Homme-Machine (IHM), la base de données et le moteur de calcul. Toutefois, il existe encore de nombreux problèmes techniques dont la solution restent à être définie de façon précise. Finalement, ce mémoire se conclut par des remarques et des discussions sur tous ces sujets concernés, ainsi que des perspectives pour des travaux dans le futur.

Liste des contributions

Les publications réalisées au cours de la thèse sont listées ci-dessous. En particulier, ODS a été classé 13^{ème} parmi 107 projets européens dans la compétition étudiante de TRAVISION 2016. De plus, le projet TOAST a été classé à la seconde place du Prix Européen d'Innovation Smart Cities 2017.

Article soumis à des revues internationales

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Model and methods to address road network problems with disruptions and an application to urban contexts*. Submitted to an international journal with impact factor. 2018.

Congrès internationaux avec actes indexés

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Methods for solving road network problems with disruptions*. Electronic Notes in Discrete Mathematics, Volume 64, pp. 175-184, 2018.
- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *A bi-objective model to address disruptions on unidirectional road networks*, in: *Proceedings of the 8th IFAC Conference on Manufacturing Modelling, Management and Control (MIM2016)*, 12 **49**, pp. 1620-1625, Troyes, France, 2016.

Congrès internationaux avec résumés étendus

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Disruptions management in multidirectional road networks*, in: *Proceedings of the 9th Triennial Symposium on Transportation Analysis (TRISTAN16)*, 4p. Oranjestad, Aruba, 2016.
- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Heuristiques pour le problème de reconfiguration des réseaux urbains suite à des interruptions routières*, in: *Actes du 19ème congrès de la société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF)*. 2p. Lorient, France, 2018.

Congrès internationaux avec communications orales

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Heuristics for the bi-objective Unidirectional Road Network Design Problem with Disruptions*, in: *Annual workshop of the EURO working group on Vehicle Routing and Logistics optimization (VeRoLog)*, p140. Nantes, France, 2016.
- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Managing predictable and unpredictable disruptions on road networks*, in: *Int'l Conference on Intelligent Transportation Engineering and Smart City (ITESC 2017)*, Guilin, China, 2017.

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Managing disruptions in urban road networks for real contexts*, in: *6th INFORMS Transportation Science and Logistics Society Workshop*, Hong Kong, China, 2018.

Autres

- Yipeng Huang, Andréa Cynthia Santos, Christophe Duhamel, *Optimal traffic Deviation System*. TRAVISIONS (Academic Student Competition), 23, Varsovie, Pologne, 2016.

Chapitre 2 - État de l'art

D'après la définition générale précédente, RND concerne les réseaux routiers, qui font l'objet d'études depuis de nombreuses décennies. Plusieurs domaines de recherche sont connexes à l'étude de RND, notamment la théorie des graphes et le NDP. En termes de graphe, les notions sur la forte connexité, l'orientation et le renversement d'arcs sont importantes. De même, les modèles, les méthodes et les applications de NDP sont nécessaires pour guider notre recherche. Ce chapitre sert ainsi à détailler les travaux scientifiques se rapportant à nos thèmes de recherche.

Dans le cadre de l'orientation des graphes, le théorème de Robbins (Robbins (1939) est fondateur de la recherche sur les propriétés de forte connexité dans les graphes. L'auteur prouve qu'un graphe simple non-orienté admet une orientation fortement connexe ssi il reste connexe en retirant n'importe quelle arête. Ce graphe est orientable s'il existe une forte orientation. En prenant la définition d'un pont (*bridge*) signifiant une arête dont l'enlèvement casse la connexité du graphe, le théorème peut aussi s'énoncer comme: un graphe est orientable ssi il n'existe pas de ponts.

Ensuite, Chvátal & Thomassen (1978) étendent la recherche théorique sur le problème d'orientation des graphes en considérant une contrainte de distance, mesurée par le rayon ou le diamètre du graphe induit. Dans cette étude, une borne inférieure est proposée pour le rayon des orientations fortes. Les auteurs prouvent aussi que trouver une orientation fortement connexe de diamètre 2 est un problème NP-difficile.

De manière plus générale, l'orientation des multigraphes mixtes est étudiée dans Boesch & Tindell (1980). Il s'agit de traiter des graphes contenant à la fois des arcs et des arêtes tout en respectant le principe de multigraphe, c'est-à-dire qu'il peut y avoir plus d'un arc ou d'une arête entre chaque paire de sommets. Le théorème de Robbins est ainsi étendu tel que des orientations fortes existent pour un multigraphe mixte ssi il n'y a pas de ponts dans le graphe résultant après avoir supprimé l'orientation du graphe initial.

Il est aussi nécessaire de mentionner les travaux davantage algorithmiques sur le problème

d'orientation de graphes, notamment celui de Tarjan (1972), de Roberts (1978) et de Chung *et al.* (1985). Des algorithmes en temps linéaire sont proposés pour fortement orienter soit des graphes simples soit des multigraphes mixtes. Généralement, ils s'attachent à l'algorithme de DFS et de recherche de composantes fortement connexes, qui sont centraux dans le domaine. En particulier, Conte *et al.* (2016) propose un algorithme pour énumérer toutes les orientations fortement connexes d'un multigraphe mixte de m arcs et arêtes en temps amorti de $\mathcal{O}(m)$ pour le passage d'une orientation à la suivante. L'idée basique est de trouver des ponts dans des sous-graphes ainsi qu'une direction pour chacun respectant la forte connexité du graphe. Par ailleurs, les auteurs utilisent une technique spéciale basée sur des boucles pour transformer efficacement d'une orientation en une autre.

L'utilisation du renversement d'arcs est en fait relativement peu étudiée. Il y a cependant des travaux essentiels comme la conjecture de Ádám (1963) sur l'existence d'un arc dont le renversement réduit le nombre de cycles élémentaires dans un graphe simple orienté. Néanmoins, plusieurs contre-exemples sur des graphes particuliers sont trouvés par Jirásek (1987), Thomassen (1987) et Jirásek (2005). Récemment, l'étude de Bogdanowicz (2011) indique que, dans un multigraphe équilibré, orienté et sans boucles, n'importe quel arc renversé engendre la diminution des cycles élémentaires non-nécessaires.

Au niveau de NDP, l'étude Magnanti & Wong (1984) est considérée comme un point d'entrée dans le domaine. Elle propose une classification du problème en trois niveaux: stratégique, tactique et opérationnel, ce qui correspondent aux différents points de vue pour la prise de décision. Les auteurs révisent aussi des problèmes divers selon le type de contraintes, avec un modèle générique proposé pour le NDP discret. Ensuite, les méthodes d'optimisation sont aussi détaillées en indiquant que les heuristiques de l'époque sont capables de traiter des instances dont la taille monte jusqu'à 50 sommets et 150 arcs.

Boyce & Janson (1980) se consacre spécialement au NDP discret en minimisant le coût total de déplacement, ce qui est en relation avec l'un des objectifs de RND. Les modèles mathématiques sont proposés avec des expérimentations effectuées sur une instances de 10 sommets. De plus, Yang & Bell (1998) étudie un modèle générique à deux niveaux pouvant couvrir de nombreux problèmes de prise de décision dans le contexte de NDP. Il peut surtout être appliqué pour des problèmes considérant la planification soumise à des budgets d'investissement. Ensuite, Feremans *et al.* (2003) étudie une généralisation de NDP en utilisant le regroupement des sommets afin de classifier des problèmes spécifiques en deux catégories: des problèmes de couverture dont les solutions utilisent tous les groupes de sommets et des problèmes opposés, de non-couverture. Des exemples généralisant les problèmes connus sont présentés. D'autres travaux sur le NDP tels que L. & Friesz (1983), Boyce (1984), Friesz (1985), Cascetta (2001) et Farahani *et al.* (2013), dans lesquelles

plusieurs aspects théoriques et applicatifs sont détaillés, existent.

Particulièrement, le problème d'orientation forte du réseau (Strong Network Orientation Problem (SNOP)) est corrélé avec RND dans notre contexte. L'étude de Santos *et al.* (2013) est notamment importante. Elle consiste à la définition du problème et de nombreuses métaheuristiques sont proposées. Cependant, au lieu de chercher à orienter un réseau non-orienté tel que SNOP fait, RND se concentre plutôt à reconfigurer et améliorer des réseaux avec leur orientation existante.

Comme NDP est fortement lié aux problèmes rencontrés dans la vie réelle de la gestion des réseaux routiers, il y a aussi des recherches applicatives. Par exemple, dans le contexte urbain, NDP est étudié dans Cantarella *et al.* (2006) en considérant le problème de localisation et configuration des feux de circulation. Plusieurs méthodes indépendantes et hybrides sont proposées, notamment différentes combinaisons de l'Algorithme Génétique (GA) avec d'autres métaheuristiques telles que Tabu Search (TS). Les expérimentations sont menées sur des instances réelles provenant de trois régions italiennes. Une autre étude avec des tests sur des réseaux réels est faite par Gallo *et al.* (2012). Notons que ces études se focalisent surtout sur le long terme alors que RND porte sur le réaménagement des réseaux de court terme et de long terme. C'est ainsi la contribution générale de notre thèse.

Chapitre 3 - Définition formelle des problèmes et modèles mathématiques

Dans ce chapitre, RND est formellement défini avec respectivement les modèles mathématiques pour bi-URND et bi-MRND. Ces derniers sont testés via le solver CPLEX sur des instances générées. Tout d'abord, nous présentons les notations utilisées dans le reste du document, particulièrement pour les définitions de différents types de graphe.

Supposons $G_0 = (V, E)$ un graphe simple connexe non-orienté sans boucles. V est l'ensemble des sommets et E est l'ensemble des arêtes. Une arête non pondérée est notée par (i, j) où $i, j \in V \times V$ et $(i, j) = (j, i)$. Pour une arête pondérée, un coût γ_{ij} est associé.

$G_1 = (N_1, A_1)$ est un graphe simple connexe orienté avec l'ensemble des noeuds N_1 et l'ensemble des arcs A_1 . Un arc non pondéré est représenté par une paire de noeuds (i, j) où $i, j \in N_1 \times N_1$ et $(i, j) \neq (j, i)$. Un poids γ_{ij} peut aussi être associé à un arc. Étant donné l'ensemble des blocages $B_1 \subset A_1$, $H_1 = (N_1, A'_1)$ est le sous-graphe de G_1 après avoir enlevé les arcs bloqués, $A'_1 = A_1 \setminus B_1$.

Enfin, un multigraphe connexe orienté est défini par $G_2 = (N_2, A_2)$ pour lequel les ensembles de noeuds et de multi-arcs sont dénotés N_2 et A_2 . Par définition, un multigraphe peut admettre plusieurs arcs entre chaque paire de noeuds. Ainsi, les arcs multiples entre

deux noeuds $i, j \in N_2 \times N_2$ sont indexés avec un entier $k \in \mathbb{N}$. La représentation d'un arc devient donc $(i, j, k) \in A_2$. Similaire à la réduction de G_1 à H_1 , $H_2 = (N_2, A'_2)$ est le sous-graphe de G_2 avec $A'_2 = A_2 \setminus B$, *i.e.* après avoir retiré les arcs bloqués.

Définitions de RND, d'URND et de MRND

Il est important d'abord de souligner la modélisation d'un réseau de transport par graphe. Dans un réseau routier, il existe au moins un chemin entre toute paire de points. Cette propriété correspond à la connexité d'un graphe non-orienté et à la forte connexité d'un graphe orienté. De plus, la rue (un tronçon) et l'intersection sont les deux éléments topologiques essentiels dans un réseau de transport. On peut les représenter par un sommet (noeud) et une arête (un arc) dans un graphe. Néanmoins, une rue peut avoir plusieurs voies et sa circulation peut être à sens unique ou à double sens. L'utilisation du multigraphe est donc naturelle pour bien distinguer le sens et les voies.

L'objectif de RND est de trouver une orientation pour un réseau de transport en assurant qu'il soit fortement connexe, en prenant les interruptions en compte. Deux critères de minimisation sont pris en compte: (i) la distance totale entre toute paire de noeuds (c_1) et (ii) le nombre d'arcs renversés (c_2). Le renversement d'arcs est admis comme une opération pour restorer la forte connexité (le cas échéant) et éventuellement réduire la distance totale.

URND s'adresse aux réseaux des rues lorsqu'elles sont seulement à sens unique. C'est souvent le cas dans un centre-ville ou dans un centre historique, où la gestion pratique des interruptions peut rencontrer des difficultés. Un tronçon bloqué peut impacter plusieurs autres axes puisqu'il faut souvent emprunter un chemin de déviation beaucoup plus long que celui prévu. En prenant le graphe simple G_1 pour modéliser URND, l'objectif c_1 est prouvé NP-difficile dans Burkard *et al.* (1999). La minimisation de c_2 est-elle polynomiale d'après Bang-Jensen & Gutin (2008). Par conséquent, bi-URND est NP-difficile. Supposons l'ensemble des paires origine-destination (paires O-D) $P = \{(o, d) | o, d \in V_1, o \neq d\}$ et l'ensemble B_1 des p arcs bloqués. Comme indiqué précédemment, H_1 est le sous-graphe de G_1 obtenu par suppression des arcs bloqués. Nous proposons alors le modèle mathématique pour URND ci-dessous:

$$\min c_1 = \sum_{(o,d) \in P} \sum_{(i,j) \in A'_1} \gamma_{ij} y_{ij}^{od} \quad (\text{A.1})$$

$$\min c_2 = \sum_{(i,j) \in A'_1} x_{ij} \quad s.t. \quad (\text{A.2})$$

$$\sum_{l:(l,o) \in A'_1} f_{lo}^{od} - \sum_{l:(o,l) \in A'_1} f_{ol}^{od} = -1 \quad \forall (o,d) \in P \quad (\text{A.3})$$

$$\sum_{l:(l,i) \in A'_1} f_{li}^{od} - \sum_{l:(i,l) \in A'_1} f_{il}^{od} = 0 \quad \forall (o,d) \in P, \forall i \notin \{o,d\} \quad (\text{A.4})$$

$$-y_{ij}^{od} \leq f_{ij}^{od} \leq y_{ij}^{od} \quad \forall (o,d) \in P, \forall (i,j) \in A'_1 \quad (\text{A.5})$$

$$-x_{ij} \leq f_{ij}^{od} \leq 1 - x_{ij} \quad \forall (o,d) \in P, \forall (i,j) \in A'_1 \quad (\text{A.6})$$

$$x_{ij} \leq \sum_{(o,d) \in P} y_{ij}^{od} \quad \forall (i,j) \in A'_1 \quad (\text{A.7})$$

$$x_{ij} \in \{0,1\} \quad \forall (i,j) \in A'_1 \quad (\text{A.8})$$

$$f_{ij}^{od} \in \mathbb{R} \quad \forall (o,d) \in P, \forall (i,j) \in A'_1 \quad (\text{A.9})$$

$$y_{ij}^{od} \geq 0 \quad \forall (o,d) \in P, \forall (i,j) \in A'_1 \quad (\text{A.10})$$

Notons que les variables $x_{ij} \in \{0,1\}$ indiquent si l'arc (i,j) est renversé ($x_{ij} = 1$) ou pas ($x_{ij} = 0$). Le flot passant par un arc (i,j) pour le chemin d'une paire (o,d) est dénoté $f_{ij}^{od} \in \mathbb{R}$. Il prend aussi le renversement en compte. Si (i,j) est renversé et utilisé pour le chemin entre (o,d) , alors $f_{ij}^{od} = -1$; si l'arc n'est pas renversé mais utilisé, $f_{ij}^{od} = 1$; si le chemin n'utilise pas l'arc, nous aurons $f_{ij}^{od} = 0$. Les variables $f_{ij}^{od} \geq 0$ sont les valeurs absolues de f , signifiant le montant de flot, indépendant du renversement d'arc. Les objectifs c_1 et c_2 sont décrits par des fonctions objectif (A.1) et (A.2). Les contraintes de conservation de flot sont définies par les équations (A.3) et (A.4). Les inégalités (A.5), (A.6) et (A.7) sont les relations entre variables avec les définitions de ces dernière dans (A.8), (A.9) et (A.10).

Le multigraphe G_2 est utilisé pour modéliser MRND, qui concerne plutôt les réseaux plus généraux dans lesquels des rues à double sens et à sens unique peuvent co-exister. Bi-MRND est également NP-difficile. Son modèle est similaire à celui d'URND, sauf que les variables sont à cinq dimensions avec l'index k ajouté dans la représentation d'un multi-arc, comme défini au début du chapitre. Toutefois, nous montrons que, considérant uniquement les critères c_1 et c_2 , tout multigraphe peut être réduit en un 2-directed multigraph dans lequel il y a au plus deux multi-arcs entre chaque paire de noeuds. Ainsi, l'index k est restreint, soit $k \in \{0,1\}$.

Les expérimentations de ces modèles sont ensuite effectuées. Le but est de tester leur fonctionnement, de connaître leurs limites et d'identifier les marges en termes de relaxation linéaire pour trouver des solutions optimales. Les deux objectifs sont testés indépendamment dans cette partie alors que l'optimisation bi-objectif est traitée plus tard dans ce mémoire.

Les instances de petite et moyenne taille ont été générées pour les tests. Ce sont des graphes en grille pour lesquels tous les arcs ont un coût de 1. Les arcs bloqués sont sélectionnés de façon aléatoire, en garantissant cependant que les graphes résultants sont toujours orientables. Selon le type d'instances, 14 graphes simples et 24 multigraphes 2-orientés sont créés. Pour URND, le modèle rencontre des difficultés sur l'objectif c_1 . Ceci est confirmé par des marges assez grandes au niveau de la relaxation linéaire. Aussi, le temps de calcul croît rapidement avec l'augmentation de taille d'instance et CPLEX n'arrive pas à résoudre 3 parmi 4 instances avec 36 noeuds et 60 arcs dans une limite de 2 heures. En revanche, la minimisation du nombre d'arcs renversés ne pose pas de problème pour le modèle. Les marges sont faibles et le temps d'exécution est raisonnable. Pour MRND, les résultats sont favorables pour c_1 et c_2 sauf que la résolution de certaines instances moyennes "6x6" est limitée par un blocage lié à la taille des programmes linéaires mixtes générés.

Chapitre 4 - Méthodes pour résoudre URND et MRND en mono- et bi-objectif

Le chapitre est dédié aux méthodes exactes et approchées proposées pour résoudre URND et MRND en mono- et bi-objectif.

Tout d'abord, nous présentons une heuristique d'orientation des graphes reposant sur l'algorithme DFS. L'utilisation du DFS pour orienter des graphes a été proposée initialement par Roberts (1978) et l'algorithme conçu dans cette thèse est une extension de son idée. À partir d'un sommet de départ, le parcours en profondeur a pour but de visiter tous les sommets en utilisant les relations d'adjacence. Chaque sommet visité est marqué comme "trouvé" et un sommet dont tous les voisins sont visités est marqué comme "complètement visité". Une visite signifie la propagation d'un sommet i à un sommet voisin j en passant par l'arête (i, j) . Cette dernière peut alors être orientée selon le sens de la visite. De plus, un sommet est marqué comme complètement visité si toutes les arêtes liées sont orientées. Un sommet déjà visité peut être visité de nouveau, tant qu'il n'est pas complètement visité. Cependant, au lieu de terminer quand tous les sommets sont complètement visités, comme dans DFS original, la condition d'arrêt de notre algorithme est que toutes les arêtes soient orientées. La complexité reste donc $\mathcal{O}(n + m)$ où les nombres de sommets et d'arêtes sont n et m respectivement. Par ailleurs, pour piloter les sens de visite, l'heuristique admet un

vecteur de poids associés aux sommets. Au moment de choisir le prochain sommet à visiter parmi plusieurs candidats, celui avec un poids plus haut sera sélectionné. Ceci permet de générer davantage d'orientations forte, selon le vecteur de poids. Cet algorithme est intégré à la fois dans BRKGA et dans ILS. La Figure A.1 montre des exemples de l'exécution de cette heuristique. Pour un multigraphe non-orienté avec 9 sommets et 15 arêtes, en prenant le vecteur de poids comme $\{0.6, 0.9, 0.7, 0.3, 0.5, 0.2, 0.8, 0.6, 0.1\}$ correspondant aux sommets 0-8, l'orientation résultant est présentée dans Figure A.1-(b). Étant donné un ensemble de poids différents, la Figure A.1-(c) affiche une autre orientation trouvée.

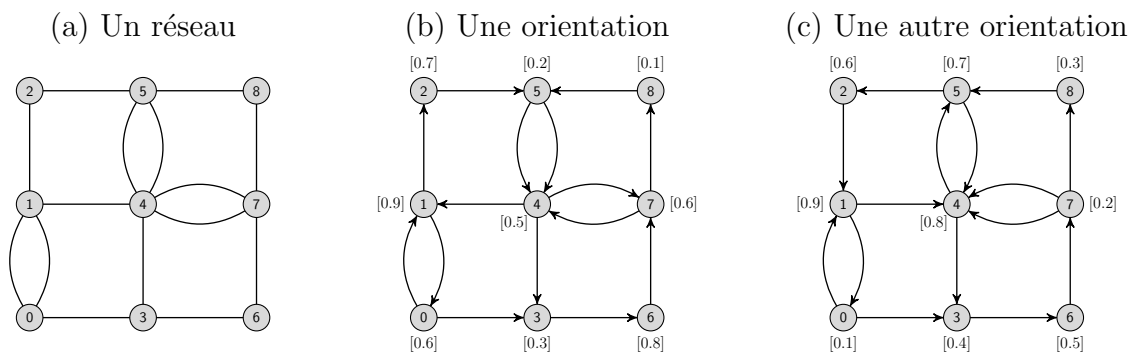


Figure A.1: Exemples de l'heuristique d'orientation des graphes

Optimisation mono-objective

L'intérêt des méthodes heuristiques pour l'optimisation combinatoire est leur simplicité et leur efficacité. Elles sont souvent capables de trouver des solutions de bonne qualité dans un temps limité. Les applications existent pratiquement dans tous les domaines de la Recherche Opérationnelle. Une métaheuristique, de son côté, est une stratégie de pilotage globale de la recherche de solutions dans l'espace des solutions. À ce titre, elle peut coordonner plusieurs heuristiques ou encore contrôler leurs paramètres d'exécution. Par conséquent, un heuristique hérite généralement des avantages des heuristiques simples pour résoudre des problèmes avec peu de consommation du temps (un peu plus quand même), tout en réduisant leurs inconvénients, notamment la faible qualité des solutions. C'est la raison pour laquelle des métaheuristiques sont développées dans notre travail. Afin de faciliter la présentation, nous commençons par les méthodes en mono-objectif puis nous passons ensuite à la résolution pour bi-URND et bi-MRND.

BRKGA

BRKGA est une métaheuristique proposée par Gonçalves & Resende (2011) et Resende (2012). C'est une variante de l'Algorithme Génétique, lequel repose sur l'évolution d'une population d'individus. La particularité de cette métaheuristique est surtout l'encodage générique des chromosomes, l'utilisation de l'élitisme, le biais et l'ajout des mutants. Premièrement, chaque individu est encodé par un vecteur de clés aléatoires dans l'intervalle $[0, 1]$. Il doit être décodé pour obtenir une solution selon les spécifications du problème. Ensuite, selon l'évaluation des solutions, la population des individus est ainsi divisée en deux groupes: l'un des élites avec des meilleures performances et le reste composé des non-élites. Au passage d'une génération à une autre, les solutions élites sont toujours gardées. De plus, pour générer de nouveaux individus, le croisement se fait toujours avec un chromosome dans les élites et un autre dans les non-élites. C'est le principe de l'évolution biaisée. Finalement, BRKGA n'applique pas une vraie mutation comme dans les implémentations classiques de l'Algorithme Génétique. À la place, à la fin de chaque génération, une partie de la nouvelle population est composée par des chromosomes arbitraires appelés mutants. Notons que toutes les opérations dans BRKGA sont indépendantes du problème donné, sauf la partie pour décoder des chromosomes en solutions spécifiques. Ainsi, c'est une métaheuristique assez générique pouvant être appliquée dans des domaines de recherche divers. Des applications spécifiques sont détaillées notamment dans Stefanello *et al.* (2017).

Avec l'heuristique d'orientation des graphes présentée précédemment, il est possible de considérer le vecteur des clés aléatoires comme un vecteur des poids associés aux noeuds d'un graphe. Ainsi, un chromosome de BRKGA peut être décodé en une orientation d'un graphe donné. Ensuite, pour évaluer la valeur c_1 , nous appliquons l'algorithme de Dijkstra (Dijkstra (1959)), adapté pour calculer la somme des distances des plus courts chemins entre chaque paire de noeud. De plus, en comparant l'orientation de la solution avec l'orientation initiale du graphe, le nombre d'arcs renversés peut être compté (objectif c_2). Cependant, d'après des tests préliminaires, BRKGA a tendance à trouver des solutions qui renversent de nombreux arcs, ce qui n'est pas désirable dans la gestion réelle des réseaux. De ce fait, un post-processing est intégré dans le décodage des chromosomes pour essayer d'annuler des arcs renversés tout en garantissant que l'orientation résultante soit fortement connexe.

ILS

L'idée d'une recherche locale (Local Search) est de prendre une seule solution initiale et d'examiner ses voisins définis par des changements spécifiques. Souvent dans le cadre d'une approche gloutonne, la procédure sélectionne une des voisins améliorant, se déplace

dessus, puis recommence, tant qu'une amélioration est possible. Cette méthode intuitive a été améliorée pour s'adapter à l'optimisation des problèmes complexes, notamment par Lourenço *et al.* (2003) qui propose la recherche locale itérée (ILS). D'après les auteurs, ILS consiste en quatre parties: la génération de la solution initiale, la recherche locale, le critère d'acceptation et la perturbation. Une heuristique constructive peut être appliquée pour générer une solution initiale. Sa qualité peut être médiocre, le reste de la métaheuristique l'améliorera. Des opérateurs de la recherche locale servent ensuite à examiner les voisins de la solution courante pour trouver une solution meilleure. Puis, si cette solution remplit la condition d'acceptation, elle devient la nouvelle solution courante. Enfin, une procédure perturbe cette solution pour générer une solution proche et se projeter dessus. Les quatre parties se répètent jusqu'à la condition d'arrêt atteinte. Grâce à la coordination de ces petites fonctions, ILS permet surtout d'éviter de rester bloqué dans des optima locaux. Des applications se trouvent dans Lourenço *et al.* (2003) et Lourenço *et al.* (2010).

Dans notre ILS destinée à URND et à MRND, la génération de la solution initiale se fait par l'heuristique d'orientation des graphes. Un vecteur des poids aléatoires est généré pour satisfaire au prérequis d'entrée de l'algorithme. Notre recherche locale reprend la stratégie de la descente à voisinages variables (Variable Neighborhood Descent (VND)) introduite par Hertz & Mittaz (2001). Trié selon la complexité de chaque type de mouvement, un ensemble de trois voisinages $\{\mathcal{N}_i, i = 1, 2, 3\}$ sont définis pour ILS. Un mouvement de \mathcal{N}_1 cherche à renverser un cycle dont la taille est supérieure à 2, afin d'éviter un retournement des arcs bi-directionnels pour une paire de noeuds. \mathcal{N}_2 contient des mouvements qui renversent tous les arcs associés à un noeuds. Un mouvement de \mathcal{N}_3 renverse un arc tout en respectant la forte connexité du graphe. Au niveau du critère d'acceptation, une solution meilleure que la solution courante est acceptée, en fonction du critère d'optimisation. Finalement, la perturbation est divisée en deux phases selon le nombre d'itérations passées. La première phase cherche à renverser le plus grand cycle dépendant dont les arcs appartiennent à d'autres cycles. La seconde phase renverse tous les arcs. Notons que la stratégie de multi-start est aussi utilisée pour générer d'autres solutions initiales et répéter les autres procédures. Ainsi, la diversité est renforcée.

Optimisation bi-objectif

Nombre de problèmes pratique dans l'industrie considèrent plusieurs critères, notamment pour des problèmes de prise de décision. De nombreuses travaux et synthèses sur ce sujet existent, notamment Chankong & Haimes (1983), Chinchuluu & Pardalos (2007), Ehrgott & Gandibleux (2000), Zopounidis & Pardalos (2010) et Zhou *et al.* (2011). Comme les critères sont en conflit dans la plupart des cas, il existe très rarement une unique solution

optimales. En revanche, dans le cadre d'optimisation multi-objectif, la résolution d'un problème s'effectue par la recherche d'un ensemble de solutions telles que aucune n'est dominée par une autre. En empruntant la notion de dominance de Pareto (Pareto (1906)), cet ensemble est appelé front de Pareto et les solutions sont dites Pareto-optimales. En considérant les objectifs c_1 et c_2 pour bi-URND et bi-MRND, une solution s_2 est dominée par s_1 si une des deux conditions suivantes est remplie: (i) $f_{c_1}(s_1) < f_{c_1}(s_2)$ et $f_{c_2}(s_1) \leq f_{c_2}(s_2)$; (ii) $f_{c_1}(s_1) \leq f_{c_1}(s_2)$ et $f_{c_2}(s_1) < f_{c_2}(s_2)$.

Méthode ϵ -constraint

La méthode ϵ -constraint est une méthode exacte pour construire le front de Pareto optimal, proposée par Haimès *et al.* (1971). Elle s'applique particulièrement bien aux problèmes dont au moins un des critères est défini sur \mathbb{Z} . Pour l'optimisation bi-critère, le principe est de transformer un critère entier en une contrainte tout en optimisant l'autre objectif. Progressivement, cette contrainte est modifiée par une solution trouvée et l'optimisation continue jusqu'au moment où toutes les solutions possibles ont été générées. Le front de Pareto se construit ainsi par l'identification itérative de ces solutions.

Pour bi-URND et bi-MRND, l'objectif c_2 prend des valeurs entières non-négatives. Il peut ainsi être transformé en une contrainte ϵ :

$$\sum_{(i,j) \in A'} x_{ij} \leq \kappa. \quad (\text{A.11})$$

D'où, κ prend initialement la valeur c_2 de la solution optimale pour la minimisation de c_1 . La contrainte s'actualise par la nouvelle solution calculée et ainsi de suite.

Adaptation BRKGA au bi-objectif pour bi-URND et bi-MRND

BRKGA n'est initialement pas une méthode d'optimisation multi-objectif. Des adaptations sont donc nécessaires afin qu'elle puisse fournir des fronts de Pareto pour bi-URND et bi-MRND. Pour ce faire, un pool de solutions est intégré dans la métaheuristique. Il sert à stocker toutes les solutions non-dominées par les autres solutions courantes. La résolution consiste en deux procédures d'optimisation indépendantes, pour c_1 puis c_2 . Au cours de chaque optimisation, on essaie d'insérer toutes les solutions dans le pool, en vérifiant la dominance. Finalement, les solutions du le pool sont triées et constituent comme un front de Pareto approché.

Adaptation ILS au bi-objectif pour bi-URND et bi-MRND

De la même manière, ILS est initialement destinée à l'optimisation mono-objectif. Afin de l'adapter à bi-URND et bi-MRND, une fonction objectif agrégée est utilisée dans l'évaluation des solutions. Elle est présentée ci-dessous:

$$\min_{s \in S} f_\alpha(s) = \alpha f_{c_1}(s) + (1 - \alpha) f_{c_2}(s) \quad \forall \alpha \in [0, 1] \quad (\text{A.12})$$

Étant donné une valeur de $\alpha \in [0, 1]$, ILS retourne une solution qui est la meilleure au sens de cette fonction agrégée. En variant α pour un certain nombre de valeurs, un ensemble de solutions peut être identifiées pour construire un front de Pareto. Évidemment, le résultat final est fourni après avoir vérifié la non-dominance de toutes les solutions et éliminé les solutions dupliquées.

NSGA-II

Grâce à l'utilisation des populations, les algorithmes évolutionnaires sont considérés comme l'une des meilleures approches pour calculer des fronts de Pareto approchés pour des problèmes multi-objectifs. Parmi toutes les méthodes à population, l'algorithme génétique a fait l'objet de nombreux travaux d'adaptation au multi-objectif. Récemment, NSGA-II a été proposée par Deb *et al.* (2002), sur la base du tri de non-dominance pour l'évolution de la population. Il repose sur deux opérateurs importants, le ranking et la crowding-distance, qui servent respectivement à classer des solutions selon des niveaux de front qu'elles appartiennent et à équilibrer la distribution des solutions sur chaque front. Pour le premier, une approche de tri rapide de non-dominance est utilisée. Pour chaque solution, NSGA-II enregistre le nombre de solutions qui la dominent et l'ensemble des solutions dominées par elle. De ce fait, toutes les solutions ayant aucune solutions la dominant appartiennent au premier front. Ensuite, ces solutions sont éliminées et les solutions dominées par elles sont actualisées. Les solutions restantes n'ayant aucune solutions qui les dominent composent alors le deuxième front. Ainsi suite, toutes les solutions peuvent être classées et la complexité de cette procédure est $\mathcal{O}(MN^2)$ avec M le nombre d'objectifs et N la taille de population. Le second opérateur permet surtout de guider la sélection au cours de l'évolution de la population. Pour chaque solution, une distance par rapport aux solutions voisines sur le front est calculée pour identifier si elle se trouve dans un espace peuplé de nombreuses solutions ou non. Les solutions dans les zones les moins peuplées sont priorisées afin de bien distribuer les points sur le dernier front. Notons que NSGA-II hérite aussi des idées principales de l'algorithme génétique, telles que le croisement pour générer les descendants et la mutation pour diversifier la population.

La plupart des fonctions de NSGA-II sont génériques, ce qui convient aux besoins de bi-URND et bi-MRND. L'adaptation majeure est réalisée surtout pour l'encodage des

solutions ainsi que leur évaluation. Pour un graphe de m arcs, chaque chromosome de notre NSGA-II est composé de m gènes booléens (allèle $\in \{0, 1\}$), chacun indiquant l'état de renversement d'un arc par rapport à son état initial dans le graphe entré. Par conséquent, une orientation décodée peut être non fortement connexe mais elle sera pénalisée. D'après les tests préliminaires, toutes les solutions sont gardées, mais les solutions infaisables recevront une pénalisation par l'évaluation. Elles seront donc moins favorisées par rapport aux solutions fortement connexes.

Résultats numériques

Le but des expérimentations sur les métaheuristiques proposées est de tester leur performance en termes de qualité des solutions et de temps d'exécution. Les aspects mono- et bi-objectif de URND et MRND sont analysés. En plus des instances traitées par les modèles via CPLEX, nous avons généré d'autres instances de taille plus grande, surtout pour les métaheuristiques. Également, un ensemble d'instances construites à partir du réseau routier de la ville de Troyes est proposé afin d'étudier le comportement des méthodes sur la résolution des problèmes proches de la réalité.

Comme chaque métaheuristique est dépendante d'un certain nombre de paramètres, une phase de calibration est réalisée pour trouver de bonnes configurations. Dans BRKGA, les paramètres importants sont la racine aléatoire, le ratio entre la taille de population et le nombre de noeuds du graphe entré, le nombre maximal de générations et le nombre de générations sans amélioration. Dans ILS, la calibration est faite sur la racine aléatoire et le nombre d'itérations sans amélioration. Dans NSGA-II, le paramétrage s'effectue sur la racine aléatoire, le ratio entre la taille de population et le nombre d'arcs du graphe entré, le nombre maximal de générations, la probabilité de croisement et la probabilité de mutation. Sur un échantillon d'instances représentatives prédéfini, chaque paramètre de chaque méthode varie parmi trois valeurs choisies a priori. De plus, chaque test est lancé 10 fois pour obtenir des valeurs objectif moyennes.

Pour URND et MRND mono-objectif, BRKGA et ILS traitent d'abord les instances résolues par CPLEX, puis les instances plus grandes. Les résultats sur les petites instances indiquent que ILS a une meilleure performance que BRKGA en trouvant 9 optima sur 10 instances pour c_1 et 10/10 pour c_2 . Avec un niveau similaire de temps de calcul, BRKGA trouve l'optimum pour 7 instances parmi 10 pour c_1 et 8/10 pour c_2 . Au niveau des grandes instances, les deux métaheuristiques peuvent trouver des solutions dans un temps assez raisonnable sur les instances avec jusqu'à 144 noeuds et 264 arcs. La qualité des solutions trouvées par ILS est généralement meilleure que celle de BRKGA. Cependant, le temps d'exécution d'ILS s'élève plus rapidement avec l'augmentation de la taille d'instance,

particulièrement dans l’optimisation de c_2 . Le comportement de ces deux méthodes est similaire sur les instances réelles. ILS est globalement plus performante que BRKGA au niveau de la qualité avec un temps de calcul beaucoup plus haut. Un extrait des résultats numériques est présenté dans la Table A.1.

Instance	CPLEX					BRKGA			ILS		
	c_1^*	c_2	t(s)	LR	Gap(%)	c_1^*	c_2	t(s)	c_1^*	c_2	t(s)
5x5-b4	3032	23	182.74	2636	13.06	3032	23	2.68	3032	13	2.82
5x5-b6	3116	4	253.86	2672	14.25	3140	28	3.35	3116	4	1.64
6x6-b4	≤ 6820	-	7200	5956	12.67	6780	35	10.94	6854	9	8.34
6x6-b6	7006	35	3894.85	6394	8.74	7006	19	7.29	7006	35	6.14
12x12-b4	LP read OOM					188164	122	2989.76	187690	111	1527.31
12x12-b6						195228	126	791.19	188124	116	1613.10
troyes-v102-b4	LP read OOM					6718200	75	594.92	6601810	39	948.22
troyes-v102-b6						6895030	45	499.74	6643740	51	694.14

Table A.1: Exemple des résultats de l’optimisation mono-objectif sur c_1

Pour bi-URND et bi-MRND, la notion d’hypervolume (Zitzler & Thiele (1998)) est empruntée pour mesurer la performance des métaheuristiques. La méthode exacte ϵ -constraint peut trouver les fronts de Pareto optimaux pour les petites instances de bi-URND jusqu’à 25 noeuds et 40 arcs. Pour les petites instances de bi-MRND, elle peut aller jusqu’à 25 noeuds et 80 arcs. Les trois méthodes approchées sont capables de trouver des fronts approximatifs. En général, BRKGA a une performance légèrement pire que les deux autres méthodes sur les instances de bi-URND alors qu’elle fonctionne très bien sur les instances de bi-MRND. De son côté, ILS trouve des fronts proches des optima pour bi-URND, mais elle rencontre des difficultés pour bi-MRND. De plus, la performance de NSGA-II est aussi bonne pour bi-URND que pour bi-MRND. Pour les instances que la méthode ϵ -constraint ne peut pas traiter, les solutions non-dominées trouvées par l’ensemble des trois méthodes sont combinées pour simuler un front de Pareto optimal, noté *BKS* (*Best Known Solutions*). Ceci est surtout fait pour construire une base de comparaison pour l’analyse. Sur ces instances, ILS a la meilleure performance parmi les trois métaheuristiques. Les fronts qu’elle retourne sont très proches de BKS dans la plupart des cas. Des exemples se trouvent dans Table A.2.

Chapitre 5 - Application: Optimal traffic Deviation System

La gestion du trafic urbain est très compliquée dans le monde réel, souvent à cause d’une capacité du réseau insuffisante qui ne correspond pas à la demande du déplacement

Instance	ϵ -constraint			BRKGA			ILS			NSGA-II		
	Q	HV	t(s)	Q	HV	t(s)	Q	HV	t(s)	Q	HV	t(s)
5x5-b1	15	0.89	3769.28	12	0.80	87.32	9	0.87	42.28	11	0.80	62.21
5x5-b2	13	0.90	1835.61	13	0.87	87.99	7	0.88	30.45	11	0.90	64.15
6x6-b1	<i>12</i>	<i>0.88</i>	<i>(BKS)</i>	17	0.82	716	10	0.88	133	11	0.78	161
6x6-b2	<i>15</i>	<i>0.78</i>	<i>(BKS)</i>	17	0.68	485	13	0.76	126	14	0.62	161
8x8-b1	<i>17</i>	<i>0.94</i>	<i>(BKS)</i>	33	0.87	4556	15	0.93	2158	23	0.82	732
8x8-b2	<i>15</i>	<i>0.92</i>	<i>(BKS)</i>	22	0.88	3474	10	0.86	2024	28	0.87	716
troyes-v58-b1	<i>17</i>	<i>0.90</i>	<i>(BKS)</i>	15	0.77	2230	17	0.90	4754	21	0.50	1183
troyes-v58-b2	<i>16</i>	<i>0.94</i>	<i>(BKS)</i>	19	0.84	1553	15	0.94	3413	27	0.61	1205

Table A.2: Exemple des résultats de l'optimisation bi-objectif

des habitants. La situation devient encore plus complexe quand certains axes critiques sont perturbés par des interruptions routières. Par exemple, dans le réseau de transport de la ville de Troyes, il y a eu environ 8000 arrêtés municipaux en 2014, gérés manuellement par des ingénieurs de trafic urbain, soit 22 problèmes par jour. Un des défis est de trouver des déviations pour chaque arrêté ayant lieu au centre-ville historique, dans lequel il existe de nombreuses rues à sens-unique. Dans ce contexte, la collaboration avec le Poste Central de Régulation de trafic de Troyes Champagne Métropole a été établie pour développer ODS, un système d'aide à la décision pour gérer les arrêtés et proposer le calcul des déviations adaptées à chaque catégorie de véhicule. Ce projet a été réalisé dans le cadre du projet TOAST avec un support de financement par le projet Innov'Action AGIR de la région Champagne-Ardenne.

La fonctionnalité principale d'ODS est de calculer les déviations sur la base de l'optimisation d'un seul critère. Pour le moment, 3 critères d'optimisation sont intégrés dans le système de production: la distance du détour, le nombre de segments empruntés et le nombre de feux tricolores rencontrés. D'autres objectifs ont aussi été prévus, telles que le nombre de changements de sens de circulation à effectuer. Notons que cette dernière dépend des opérations de renversement et de mise en double sens des arcs, qui sont en cours de réalisation. De plus, le calcul multi-objectif sera aussi pris en compte dans le futur. Pour l'instant, un module de visualisation multi-critère est déjà implanté.

Analyse des besoins

Dans le cadre de la conception du système, l'analyse des besoins est d'abord réalisée en considérant des flux de travail des utilisateurs et des aspects pratiques du développement. Dans ce cadre, un cahier des charges complet a été établi pour diriger la mise en place du système et le suivi de l'avancement. Les principes sont expliqués dans le reste de cette

section.

La première partie de cette phase de conception est l'analyse fonctionnelle. Il s'agit surtout des besoins concernant des cas d'utilisation ainsi que des fonctionnalités dérivées. ODS doit d'abord récupérer les informations nécessaires des arrêtés municipaux. La ville de Troyes possède une base de données dans laquelle les blocages sont stockés et mis à jour régulièrement. ODS doit donc se connecter à cette base et gérer les éventuelles mises à jour et informations partielles ou erronées. Au niveau de l'interface graphique, ODS doit fournir la visualisation des arrêtés et des déviations possibles de façon intuitive. Par exemple, une vue cartographique combine des informations géographiques et des objets de trafic du réseau. Comme ODS est un logiciel d'aide à la décision, la proposition semi-automatique des solutions tout en permettant l'intervention des décideurs est l'idée centrale. Les intervenant doivent ainsi être capable d'agir dans le processus de gestion des arrêtés et de conception des déviations par des opérations manuelles, telles que la création, modification et suppression des éléments correspondants. ODS tient compte également de la structure hiérarchique de l'administration avec le rôle des ingénieurs de trafic qui traitent des arrêtés et le rôle des administrateurs qui valident des décisions. Le flux de travail ainsi que le cycle de vie des arrêtés et des déviations doivent être conçus.

Également dans le cahier des charges, il est indispensable de préciser les besoins d'un point de vue de la qualité du logiciel. Plusieurs aspects non-fonctionnels sont concernés afin de garantir le bon déroulement du développement, notamment l'utilisabilité, la fiabilité, la performance, le support et les interfaçages. En particulier, ODS doit mettre des données à jour de manière régulière pour maintenir la cohérence des informations. Les utilisateurs doivent disposer d'une documentation pour répondre aux renseignements basiques concernant l'utilisation d'ODS. Au niveau du développement du logiciel, les tests unitaires et les tests d'intégrations sont nécessaires pour garantir la maintenabilité. Les algorithmes implémentés pour le calcul des déviations doivent être optimisés afin de minimiser le temps de réponse et l'utilisation de la mémoire vive. Il est aussi important de considérer l'extensibilité du système, surtout dans l'aspect de la base de données, pour faciliter l'intégration de systèmes hétérogènes dans le futur.

Architecture du système

L'architecture est toujours critique dans la conception d'un système. D'un côté, elle doit répondre à toute demande spécifiée par le cahier des charges tout en restant extensible pour faciliter l'intégration de nouvelles fonctionnalités. De l'autre côté, elle ne doit pas prendre trop de facteurs divers en compte afin de simplifier la conception.

L'architecture d'ODS est divisée en quatre couches pour décentraliser et modulariser

des fonctions, en considérant l'intégration des systèmes hétérogènes comme la base de données des arrêtés. Elle est illustrée dans la Figure A.2. La couche de présentation est destinée particulièrement aux utilisateurs. Elle contient tous les modules sur l'Interface Homme-Machine (IHM). La couche de services sert à établir le lien entre l'IHM interagissant avec les utilisateurs et le calcul en arrière-plan d'ODS. Elle consiste surtout en des Web services interfaçables par la couche de présentation. En effet, il est possible de changer l'IHM du système sans toucher aux autres parties. Ensuite, la couche logique est surtout composée des moteurs de calcul pour la proposition des déviations. Il y a en particulier le solveur pour calculer des chemins des déviations, mais il peut y avoir aussi d'autres moteurs pour évaluer les déviations selon des critères spéciaux, tels que la pollution de l'air, la pollution sonore, la satisfaction des usagers, etc. La dernière couche est prévue pour la manipulation des données ainsi que la connexion directe avec des bases de données hétérogènes. En dessous de ces couches modulaires, nous avons aussi conçu des composantes globales permettant de réaliser la journalisation, d'assurer la sécurité et de faciliter la maintenance d'ODS.

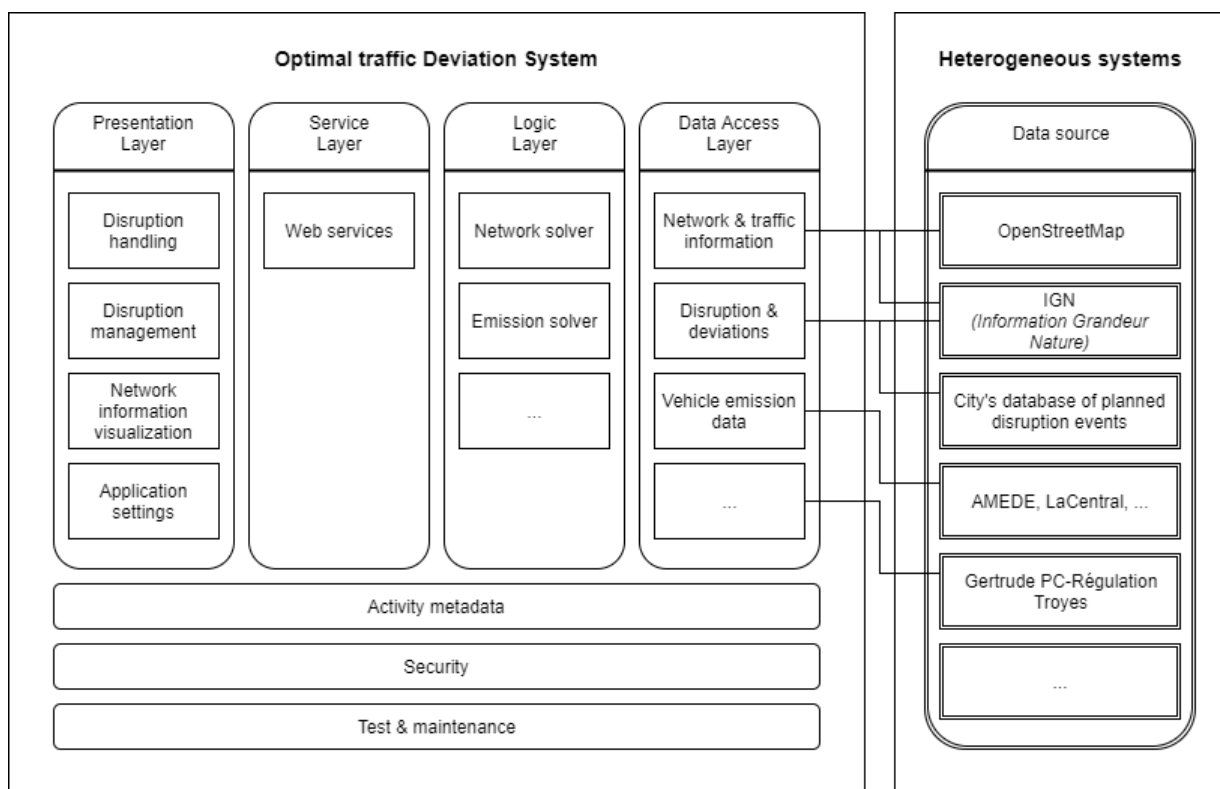


Figure A.2: System architecture of ODS

IHM

L'IHM d'un système informatique est le point direct d'interaction avec les utilisateurs. Elle se charge de collecter des entrées d'utilisateurs et d'afficher les résultats des calculs

effectués par le coeur du système. Les utilisateurs cibles d'ODS sont principalement les gestionnaires de trafic. Généralement, ils connaissent bien la problématique du réseau de transport de Troyes. En revanche, des connaissances académiques peuvent leur paraître complexes, comme la théorie des graphes, les algorithmes, le calcul des émissions, etc. Ainsi, les informations présentées par l'interface d'ODS doivent être centralisées sur une carte numérique et être les plus intuitives possible, afin d'éviter d'éventuelles difficultés de compréhension.

En général, la conception de l'IHM d'ODS consiste en quatre missions: la mise en forme générale, la cartographie, les éléments du design et la visualisation. Dans un premier temps, nous avons conçu la disposition des objets sur l'écran général du calcul des déviations. Il est composé d'une carte numérique au milieu, d'un espace de descriptifs à droite et d'un menu sur la barre latérale à droite contenant des accès rapides aux fonctionnalités. Ensuite, à l'aide du système d'informations géographiques OpenStreetMap, l'intégration d'une cartographie numérique est réalisée, avec les études préliminaires des techniques de manipulation de la carte. Puis, la palette de l'ensemble des couleurs et les polices de caractère à utiliser dans ODS sont définies. Finalement, nous avons conçu les approches pour visualiser des chemins de déviation, les critères d'évaluation et les dispositifs matériels du trafic (feux, panneaux de stop, ralentisseurs, etc.). Toutes les décisions prises dans cette phase ont été documentées pour piloter le développement de l'IHM.

Base de données

Suite à l'analyse des besoins, il paraît nécessaire d'avoir notre propre base de données dans ODS pour rassembler l'ensemble des informations concernant le réseau du trafic, les interruptions et les déviations et le graphe pour effectuer des calculs. Cependant, les bases de données hétérogènes ont des structures de données différentes qui peuvent aussi entrer en conflit avec notre manière de modélisation. Ainsi, un module de traitement des données doit être mis en place afin de convertir et nettoyer les données collectées dans un format compatible avec les modèles de données conçus pour la gestion des interruptions.

Les informations du réseau de transport proviennent d'OpenStreetMap. Il fournit des données géographiques concernant non seulement des axes routiers, mais aussi des bâtiments, parkings, voies ferroviaires, délimitations cadastrales et territoriales, etc. Ainsi, les données inutiles pour notre application sont éliminées. En outre, la modélisation du réseau par OpenStreetMap n'est pas parfaitement orientée graphe. Par exemple, une rue dans OpenStreetMap est souvent composée de plusieurs segments pour des raisons topologiques, alors que ces segments ne correspondent pas aux arcs d'un point de vue graphe car il peut y avoir des noeuds de transit. Pour simplifier le graphe utilisé dans nos calculs, il faut convertir

les rues en arcs, en distinguant les vraies intersections et en fusionnant les noeuds de transit.

Les informations des arrêtés municipaux sont aussi collectées à partir d'une base de données hétérogène. Elle contient des informations de base telles que les identifiants, les secteurs impactés, la date de début et la date de fin. Néanmoins, d'après une étude approfondie, sa structure est relativement obsolète et problématique, ce qui complique la conversion des données. Le problème le plus dur est que les informations géographiques des arrêtés ne sont pas renseignées. En revanche, les zones impactées par des arrêtés sont précisées par des adresses postales. Par conséquent, nous sommes obligés d'établir une correspondance entre les adresses et les arcs. Heureusement, OpenStreetMap dispose aussi des informations sur des adresses, avec des nuances sur le format par rapport à celui dans la base des arrêtés. Ainsi, un algorithme de similarité textuelle est implémenté en utilisant la distance de Jaro-Winkler.

Après avoir déterminé et résolu tous ces points de défis techniques, la structure de la base de données d'ODS a été définie et documentée par un diagramme de domaines entité-association.

Moteur de calcul

Le coeur du système est composé par les moteurs de calcul. Parmi ceux-ci, le moteur clé est le solveur du réseau permettant de proposer des chemins de déviation pour une interruption donnée selon des critères d'optimisation. D'autres solveurs pourront être ajoutés pour l'évaluation supplémentaire des déviations proposées.

Étant donné le graphe modélisant le réseau et les informations de l'interruption à gérer, le solveur du réseau applique les algorithmes vus dans la partie académique adapté aux critères d'optimisation prédéfinis pour calculer des déviations. Principalement, les algorithmes sont utilisés dans le solveur en considérant des coûts différents associés aux arcs, soient la distance réelle, le nombre de segments et le nombre de feux de circulation. Pour simplifier les calculs, un noeuds d'origine et un noeuds de destination sont précisés pour chaque interruption.

En outre, comme le type de véhicule est considéré, il y a souvent des contraintes particulières pour des poids lourd, telles que la hauteur, le poids et l'angle de giration. Des variations et extensions ont donc été réalisées pour prendre en compte ces contraintes. Cependant, l'application de ces algorithmes est en cours de finition, donc n'est pas délivrée dans la version courante d'ODS.

D'autres solveurs ont été prévus, notamment pour l'évaluation qualitative des déviations selon le niveau de pollution. Ils seront intégrés dans le coeur comme plugins, en respectant certaines contraintes d'interfaçage, surtout pour leurs entrées et sorties.

Défis technologiques

Le défi majeur pour ODS est de collecter des données précises de circulation. Des capteurs sont des outils efficaces pour ce faire. Ils permettent surtout de compter le nombre de véhicules selon des échelles de temps pour déduire le flot réel sur ces axes. Toutefois, à cause des contraintes budgétaires, il n'y a des capteurs que sur les intersections importantes dans le centre-ville de Troyes. Par conséquent, il est difficile d'avoir les critères d'optimisation dépendants du flot réel, notamment la congestion sur l'intégralité du réseau.

Un autre défi s'agit de modéliser l'infrastructure du réseau de transport. Il est surtout essentiel pour la mise en circulation alternée d'une rue partiellement bloquée et la mise en double sens pour évacuer le flot. De plus, pour prendre en compte des déviations pour les bus, il est nécessaire de procéder à des études sur les lignes de bus et la manière de déterminer la disposition des stations d'arrêt temporaires.

Chapitre 6 - Conclusion et perspectives

Dans cette thèse, nous avons étudié des problèmes de réaménagement du réseau de transport urbain en cas d'interruptions routières. Une recherche scientifique a été réalisée sur les problèmes URND et MRND en version mono- et bi-objectif. Les modèles mathématiques sont proposés pour traiter ces problèmes sur des instances de petite taille. Deux métaheuristiques d'optimisation mono-objectif, BRKGA et ILS, sont développées pour résoudre des instances plus grandes. Elles sont ensuite adaptées au cas bi-objectif. Par ailleurs, dans la résolution bi-critère, une méthode exacte ϵ -constraint et une métaheuristique classique NSGA-II sont aussi proposées. Les expérimentations pour les modèles et méthodes sont effectuées sur des instances générées ou extraites du réseau routier de Troyes. Au niveau de l'optimisation mono-objectif, les modèles sont capables de résoudre des instances jusqu'à 36 noeuds et 60 arcs en URND et 25 noeuds et 80 arcs en MRND. BRKGA et ILS sont assez performantes pour des instances allant jusqu'à 144 noeuds et 264 arcs en URND et 81 noeuds et 288 arcs en MRND. Pour l'optimisation bi-objectif, la méthode ϵ -constraint peut trouver des fronts Pareto-optimaux pour les petites instances et les trois métaheuristiques permettent de traiter les grandes instances. Parmi ces méthodes approchées, ILS offre généralement une bonne performance en termes de la qualité de solutions. En revanche, elle consomme beaucoup plus de temps d'exécution que les autres méthodes.

La partie applicative de la thèse concerne la conception et le développement d'ODS, un système d'aide à la décision pour gérer des interruptions routières et pour proposer automatiquement des déviations. L'analyse des besoins est d'abord effectuée avec un cahier des charges validé par tous les acteurs du projet. L'architecture du système est ensuite

définie avec des points techniques identifiés sur trois aspects: l'IHM, la base de données et les moteurs de calcul. Le développement du logiciel s'est déroulé au sein d'une équipe, avec la participation ponctuelle d'étudiants. La première version du produit final est sortie en milieu de l'année 2017.

De nombreuses possibilités sont laissées ouvertes pour des travaux futurs. Par exemple, même si le problème d'orientation des graphes en minimisant le nombre de renversements est polynomial, l'algorithme en temps polynomial n'est pas encore trouvé pour le résoudre. De plus, d'éventuelles améliorations peuvent être apportées dans les métaheuristiques, particulièrement sur le temps de calcul d'ILS et la qualité de solutions de BRKGA et de NSGA-II. En outre, d'autres contraintes réalistes peuvent être rajoutées, telles que la capacité des axes et le flot multi-modal, afin d'approcher le contexte réel des réseaux de transport urbains.

Appendix B

Tables of parameter tuning results

Instance	Settings											
	$SD_{brkga} = 0$				$SD_{brkga} = 2$				$SD_{brkga} = 8$			
	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$
3x3-b1	246	2	0.211	0	246	2	0.188	0	246	2	0.156	0
4x4-b2	968	5	1.461	0	988	8	0.681	2	968	5	0.874	0
5x5-b4	3032	23	4.721	0	3032	23	2.618	0	3056	9	2.262	1
3x3-b2-25	250	2	0.188	0	250	2	0.204	0	250	2	0.172	0
4x4-b1-75	716	7	0.858	0	716	7	1.001	0	716	7	0.858	0
5x5-b1-50	2296	16	2.309	0	2296	16	2.719	0	2296	16	2.293	0
Best	246	2	0.19	0	246	2	0.19	0	246	2	0.16	0
Worst	3032	23	4.72	0	3032	23	2.72	2	3056	16	2.29	1
Average	1251.33	9.17	1.62	0	1254.67	9.67	1.24	0	1255.33	6.83	1.10	0

Table B.1: Preliminary results of single-objective BRKGA parameter tuning on SD_{brkga} , with other parameters fixed: $R_{brkga} = 1.45$, $I_{brkga} = 100$ and $J_{brkga} = 12$

Instance	Settings											
	$R_{brkga} = 1.25$				$R_{brkga} = 1.45$				$R_{brkga} = 1.75$			
	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$
3x3-b1	246	2	0.125	0	246	2	0.188	0	246	2	0.172	0
4x4-b2	968	17	0.499	0	988	8	0.681	2	988	8	0.780	2
5x5-b4	3032	23	2.184	0	3032	23	2.618	0	3058	22	3.697	1
3x3-b2-25	250	2	0.140	0	250	2	0.204	0	250	2	0.187	0
4x4-b1-75	716	7	0.734	0	716	7	1.001	0	716	7	1.060	0
5x5-b1-50	2296	15	1.982	0	2296	16	2.719	0	2296	16	2.776	0
Best	246	2	0.125	0	246	2	0.188	0	246	2	0.172	0
Worst	3032	23	2.184	0	3032	23	2.719	2	3058	22	3.697	2
Average	1251.33	11.00	0.94	0	1254.67	9.67	1.235	0	1259.00	9.50	1.445	0

Table B.2: Preliminary results of single-objective BRKGA parameter tuning on R_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $I_{brkga} = 100$ and $J_{brkga} = 12$

Instance	Settings											
	$I_{brkga} = 50$				$I_{brkga} = 100$				$I_{brkga} = 150$			
	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$
3x3-b1	246	2	0.171	0	246	2	0.188	0	246	2	0.192	0
4x4-b2	988	8	0.577	2	988	8	0.681	2	988	8	0.787	2
5x5-b4	3032	23	2.278	0	3032	23	2.418	0	3032	23	2.646	0
3x3-b2-25	250	2	0.171	0	250	2	0.204	0	250	2	0.287	0
4x4-b1-75	716	7	0.889	0	716	7	1.001	0	716	7	1.058	0
5x5-b1-50	2296	16	2.403	0	2296	16	2.519	0	2296	16	2.793	0
Best	246	2	0.171	0	246	2	0.188	0	246	2	0.192	0
Worst	3032	23	2.403	2	3032	23	2.519	2	3032	23	2.793	2
Average	1254.67	9.67	1.08	0	1254.67	9.67	1.169	0	1254.67	9.67	1.294	0

Table B.3: Preliminary results of single-objective BRKGA parameter tuning on I_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $J_{brkga} = 12$

Instance	Settings											
	$J_{brkga} = 6$				$J_{brkga} = 12$				$J_{brkga} = 24$			
	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$
3x3-b1	246	2	0.078	0	246	2	0.188	0	246	2	0.296	0
4x4-b2	988	8	0.312	2	988	8	0.681	2	988	8	1.154	2
5x5-b4	3032	23	1.482	0	3032	23	2.618	0	3032	23	3.915	0
3x3-b2-25	250	2	0.093	0	250	2	0.204	0	250	2	0.343	0
4x4-b1-75	716	7	0.484	0	716	7	1.001	0	716	7	1.653	0
5x5-b1-50	2296	16	1.263	0	2296	16	2.719	0	2296	15	4.414	0
Best	246	2	0.078	0	246	2	0.188	0	246	2	0.296	0
Worst	3032	23	1.482	2	3032	23	2.719	2	3032	23	4.414	2
Average	1254.67	9.67	0.62	0	1254.67	9.67	1.235	0	1254.67	9.50	1.963	0

Table B.4: Preliminary results of single-objective BRKGA parameter tuning on J_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $I_{brkga} = 100$

Instance	Settings											
	$SD_{ils} = 4, J_{ils} = 6$				$SD_{ils} = 4, J_{ils} = 12$				$SD_{ils} = 4, J_{ils} = 24$			
	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$	c_1^*	c_2	t(s)	$\delta(c_1)\%$
3x3-b1	246	2	0.14	0	246	2	0.23	0	246	2	0.44	0
4x4-b2	968	17	0.52	0	968	17	1.55	0	968	5	2.15	0
5x5-b4	3032	13	2.67	0	3032	13	3.82	0	3032	13	6.18	0
3x3-b2-25	250	7	0.14	0	250	7	0.25	0	250	7	0.44	0
4x4-b1-75	716	7	1.22	0	716	7	2.28	0	716	7	3.57	0
5x5-b1-50	2296	15	4.10	0	2296	15	6.97	0	2296	15	14.76	0
Best	246	2	0.14	0	246	2	0.234	0	246	2	0.437	0
Worst	3032	17	4.103	0	3032	17	6.973	0	3032	15	14.758	0
Average	1251.33	10.17	1.46	0	1251.33	10.17	2.52	0	1251.33	8.17	4.59	0

Table B.5: Preliminary results of single-objective ILS parameter tuning on J_{ils}

Instance	Settings											
	$SD_{brkga} = 0$				$SD_{brkga} = 2$				$SD_{brkga} = 8$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0	1.34	0	2	0	1.25	0	2	0	0.86	0
4x4-b2	4	0.495	13.20	0	4	0.495	9.07	0	4	0.495	8.47	0
5x5-b4	7	0.802	46.37	11	9	0.818	28.34	9	8	0.83	27.82	8
3x3-b2-25	1	0	1.13	0	1	0	1.05	0	1	0	1.08	0
4x4-b1-75	8	0.647	7.11	1	8	0.647	6.80	1	8	0.647	6.66	1
5x5-b1-50	14	0.726	37.67	10	14	0.727	39.05	11	17	0.74	39.83	9
Best	14	0	1.131	0	14	0	1.046	0	17	0	0.858	0
Worst	1	0.802	46.368	11	1	0.818	39.047	11	1	0.83	39.826	9
Average	6.00	0.45	17.80	4	6.33	0.45	14.26	3	6.67	0.45	14.12	3

Table B.6: Preliminary results of bi-objective BRKGA parameter tuning on SD_{brkga} , with other parameters fixed: $R_{brkga} = 1.45$, $I_{brkga} = 100$ and $J_{brkga} = 12$

Instance	Settings											
	$I_{brkga} = 50$				$I_{brkga} = 100$				$I_{brkga} = 150$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0	1.11	0	2	0	1.15	0	2	0	1.16	0
4x4-b2	4	0.495	8.13	0	4	0.495	8.07	0	4	0.495	8.14	0
5x5-b4	9	0.818	28.00	9	9	0.818	27.64	9	9	0.818	27.94	9
3x3-b2-25	1	0	1.04	0	1	0	1.05	0	1	0	1.05	0
4x4-b1-75	8	0.647	6.65	1	8	0.647	6.70	1	8	0.647	6.72	1
5x5-b1-50	14	0.727	38.91	11	14	0.727	38.06	11	14	0.727	39.06	11
Best	14	0	1.044	0	14	0	1.046	0	14	0	1.045	0
Worst	1	0.818	38.907	11	1	0.818	38.057	11	1	0.818	39.063	11
Average	6.33	0.45	13.97	3	6.33	0.45	13.78	3	6.33	0.45	14.01	3

Table B.7: Preliminary results of bi-objective BRKGA parameter tuning on I_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $J_{brkga} = 12$

Instance	Settings											
	$J_{brkga} = 6$				$J_{brkga} = 12$				$J_{brkga} = 24$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0	0.66	0	2	0	1.25	0	2	0	1.92	0
4x4-b2	4	0.495	5.34	0	4	0.495	9.07	0	4	0.495	17.10	0
5x5-b4	8	0.858	34.62	5	9	0.818	28.34	9	12	0.811	77.53	10
3x3-b2-25	1	0	0.64	0	1	0	1.05	0	1	0	2.37	0
4x4-b1-75	8	0.651	3.56	0	8	0.647	6.80	1	8	0.647	13.26	1
5x5-b1-50	12	0.705	21.06	12	14	0.727	39.05	11	16	0.711	76.86	12
Best	12	0	0.64	0	14	0	1.046	0	16	0	1.919	0
Worst	1	0.858	34.616	12	1	0.818	39.047	11	1	0.811	77.532	12
Average	5.83	0.45	10.98	3	6.33	0.45	14.26	3	7.17	0.44	31.51	4

Table B.8: Preliminary results of bi-objective BRKGA parameter tuning on I_{brkga} , with other parameters fixed: $SD_{brkga} = 2$, $R_{brkga} = 1.45$ and $I_{brkga} = 100$

Instance	Settings											
	$SD_{nsgaii} = 2$				$SD_{nsgaii} = 4$				$SD_{nsgaii} = 8$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0	6.76	0	2	0	5.82	0	2	0	5.69	0
4x4-b2	3	0.339	17.90	32	4	0.495	18.53	0	2	0.295	18.17	40
5x5-b4	8	0.761	45.51	7	7	0.781	46.60	4	6	0.783	44.34	4
3x3-b2-25	1	0	7.16	0	1	0	7.74	0	1	0	7.39	0
4x4-b1-75	8	0.651	41.45	0	8	0.8	42.34	8	10	0.727	40.95	7
5x5-b1-50	13	0.753	91.20	0	13	0.835	93.34	8	15	0.827	94.32	8
Best	13	0	6.761	0	13	0	5.819	0	15	0	5.694	0
Worst	1	0.761	91.197	32	1	0.835	93.335	8	1	0.827	94.318	40
Average	5.83	0.42	35.00	6	5.83	0.49	35.73	3	6.00	0.44	35.14	10

Table B.9: Preliminary results of NSGA-II parameter tuning on SD_{nsgaii} , with other parameters fixed: $R_{nsgaii} = 1.45$, $I_{nsgaii} = 100$, $\rho_c = 0.3$ and $\rho_m = 0.03$

Instance	Settings											
	$R_{nsgaii} = 1.25$				$R_{nsgaii} = 1.45$				$R_{nsgaii} = 1.75$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0.00	4.90	0	2	0.00	6.76	0	2	0	6.91	0
4x4-b2	4	0.50	14.59	0	3	0.34	17.90	32	3	0.495	22.15	0
5x5-b4	5	0.70	38.11	14	8	0.76	45.51	7	5	0.831	58.77	8
3x3-b2-25	1	0.00	6.55	0	1	0.00	7.16	0	1	0	8.49	0
4x4-b1-75	8	0.74	33.63	5	8	0.65	41.45	0	8	0.638	53.81	2
5x5-b1-50	13	0.74	75.55	2	13	0.75	91.20	0	15	0.786	121.15	7
Best	13	0	4.899	0	13	0	6.761	0	15	0	6.911	0
Worst	1	0.741	75.551	14	1	0.761	91.197	32	1	0.831	121.15	8
Average	5.50	0.45	28.89	3	5.83	0.42	35.00	6	5.67	0.46	45.21	3

Table B.10: Preliminary results of NSGA-II parameter tuning on R_{nsgaii} , with other parameters fixed: $SD_{nsgaii} = 2$, $I_{nsgaii} = 100$, $\rho_c = 0.3$ and $\rho_m = 0.03$

Instance	Settings											
	$I_{nsgaii} = 50$				$I_{nsgaii} = 100$				$I_{nsgaii} = 150$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0	3.43	0	2	0	6.91	0	2	0	11.08	0
4x4-b2	3	0.495	10.75	0	3	0.495	22.15	0	4	0.495	33.63	0
5x5-b4	6	0.794	27.41	3	5	0.831	58.77	8	10	0.81	86.18	1
3x3-b2-25	1	0	4.26	0	1	0	8.49	0	1	0	12.79	0
4x4-b1-75	9	0.707	26.30	9	8	0.638	53.81	2	8	0.651	81.17	0
5x5-b1-50	18	0.835	61.65	8	15	0.786	121.15	7	12	0.731	182.32	6
Best	18	0	3.432	0	15	0	6.911	0	12	0	11.076	0
Worst	1	0.835	61.652	9	1	0.831	121.15	8	1	0.81	182.318	6
Average	6.50	0.47	22.30	3	5.67	0.46	45.21	3	6.17	0.45	67.86	1

Table B.11: Preliminary results of NSGA-II parameter tuning on I_{nsgaii} , with other parameters fixed: $SD_{nsgaii} = 2$, $R_{nsgaii} = 1.75$, $\rho_c = 0.3$ and $\rho_m = 0.03$

Instance	Settings											
	$\rho_c = 0.3$				$\rho_c = 0.6$				$\rho_c = 0.9$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0	6.91	0	2	0	6.88	0	2	0	7.47	0
4x4-b2	3	0.495	22.15	0	3	0.295	22.93	40	4	0.495	23.09	0
5x5-b4	5	0.831	58.77	8	6	0.783	55.74	4	8	0.814	54.52	0
3x3-b2-25	1	0	8.49	0	1	0	8.72	0	1	0	8.52	0
4x4-b1-75	8	0.638	53.81	2	8	0.767	53.32	1	8	0.6	53.46	21
5x5-b1-50	15	0.786	121.15	7	13	0.736	119.31	3	13	0.741	120.73	6
Best	15	0	6.911	0	13	0	6.879	0	13	0	7.472	0
Worst	1	0.831	121.15	8	1	0.783	119.309	40	1	0.814	120.728	21
Average	5.67	0.46	45.21	3	5.50	0.43	44.48	8	6.00	0.44	44.63	5

Table B.12: Preliminary results of NSGA-II parameter tuning on ρ_c , with other parameters fixed: $SD_{nsgaii} = 2$, $R_{nsgaii} = 1.75$, $I_{nsgaii} = 100$ and $\rho_m = 0.03$

Instance	Settings											
	$\rho_m = 0.03$				$\rho_m = 0.1$				$\rho_m = 0.3$			
	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$	Q	HV	t(s)	$\delta(HV)\%$
3x3-b1	2	0	6.91	0	2	0	6.85	0	2	0	7.72	0
4x4-b2	3	0.495	22.15	0	3	0.495	22.75	0	4	0.495	23.57	0
5x5-b4	5	0.831	58.77	8	6	0.783	55.05	4	6	0.775	57.28	5
3x3-b2-25	1	0	8.49	0	1	0	8.72	0	1	0	8.49	0
4x4-b1-75	8	0.638	53.81	2	8	0.722	53.54	1	8	0.651	51.70	0
5x5-b1-50	15	0.786	121.15	7	14	0.822	118.92	3	13	0.859	119.26	1
Best	15	0	6.911	0	14	0	6.849	0	13	0	7.722	0
Worst	1	0.831	121.15	8	1	0.822	118.919	4	1	0.859	119.262	5
Average	5.67	0.46	45.21	3	5.67	0.47	44.30	1	5.67	0.46	44.67	1

Table B.13: Preliminary results of NSGA-II parameter tuning on ρ_m , with other parameters fixed: $SD_{nsgaii} = 2$, $R_{nsgaii} = 1.75$, $I_{nsgaii} = 100$ and $\rho_c = 0.3$

Appendix C

Figures of Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II

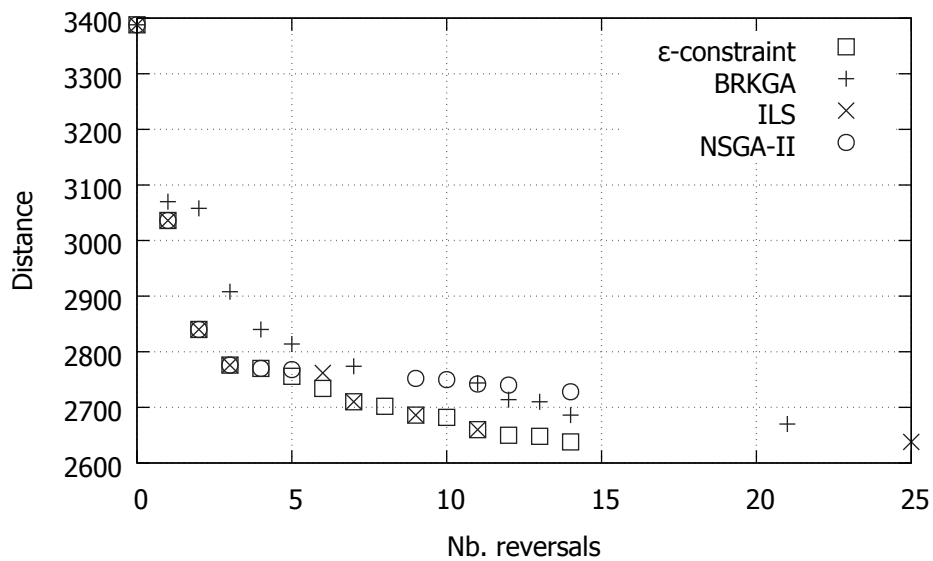


Figure C.1: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1

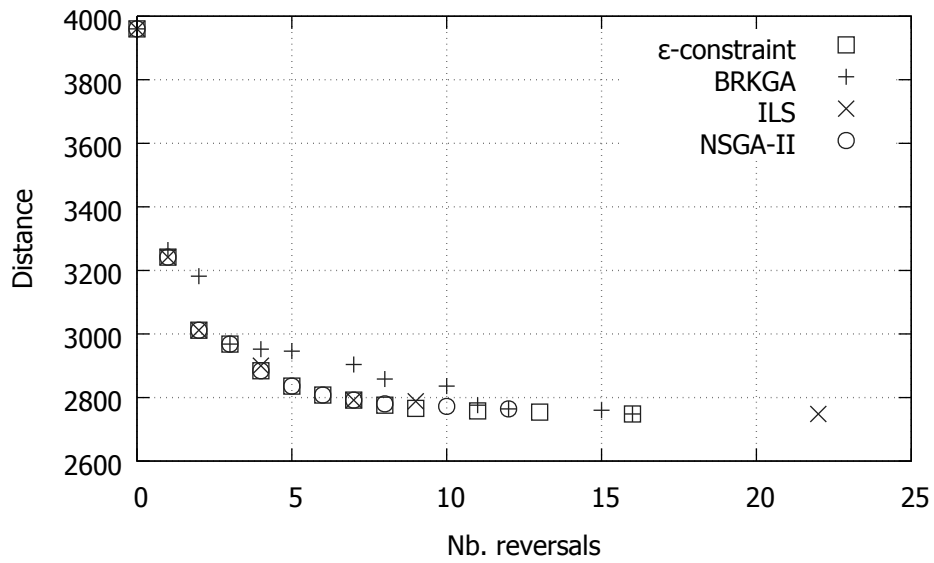


Figure C.2: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2

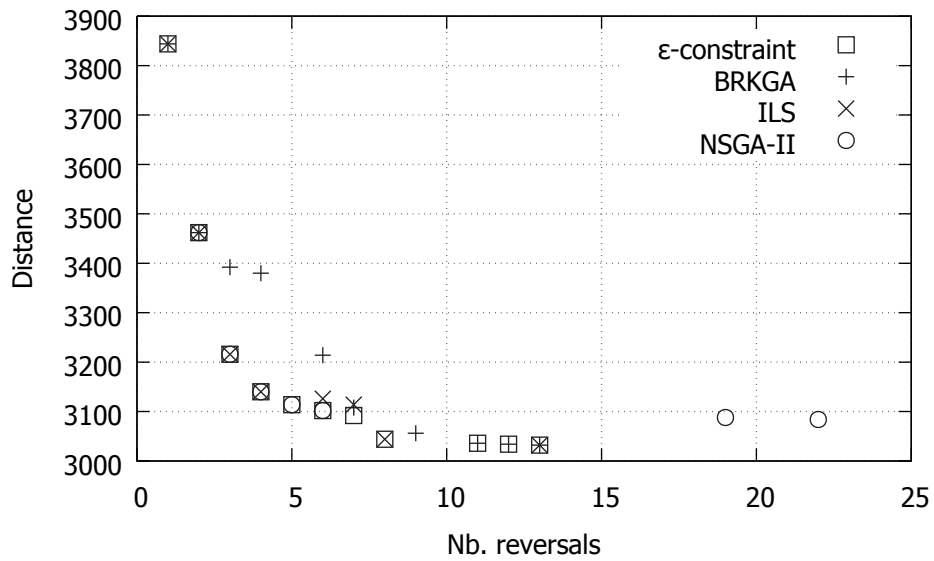


Figure C.3: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b4

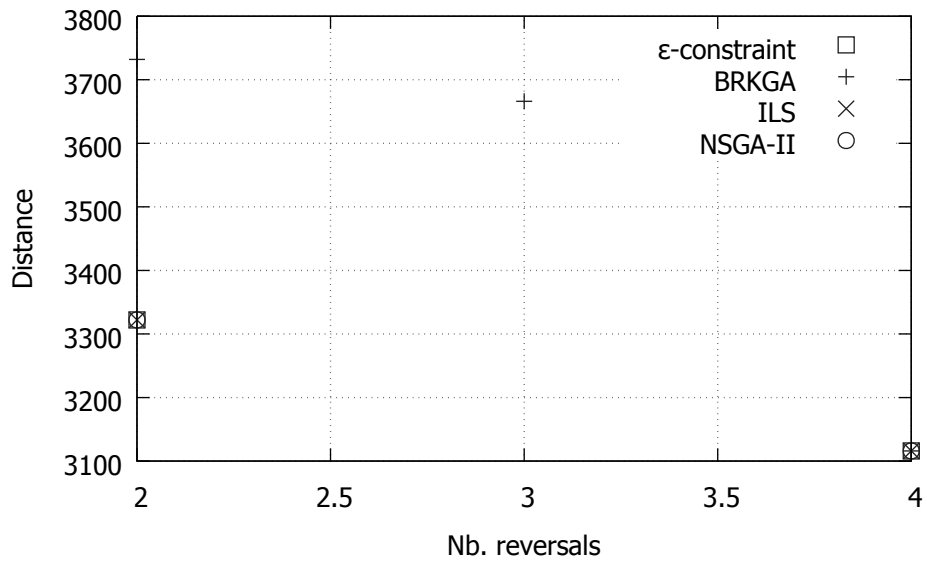


Figure C.4: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b6

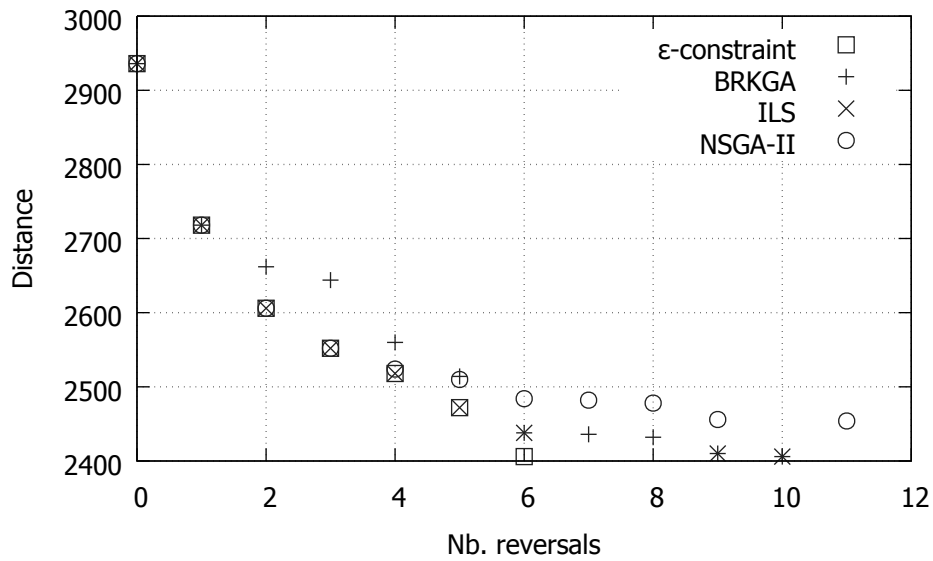


Figure C.5: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-25

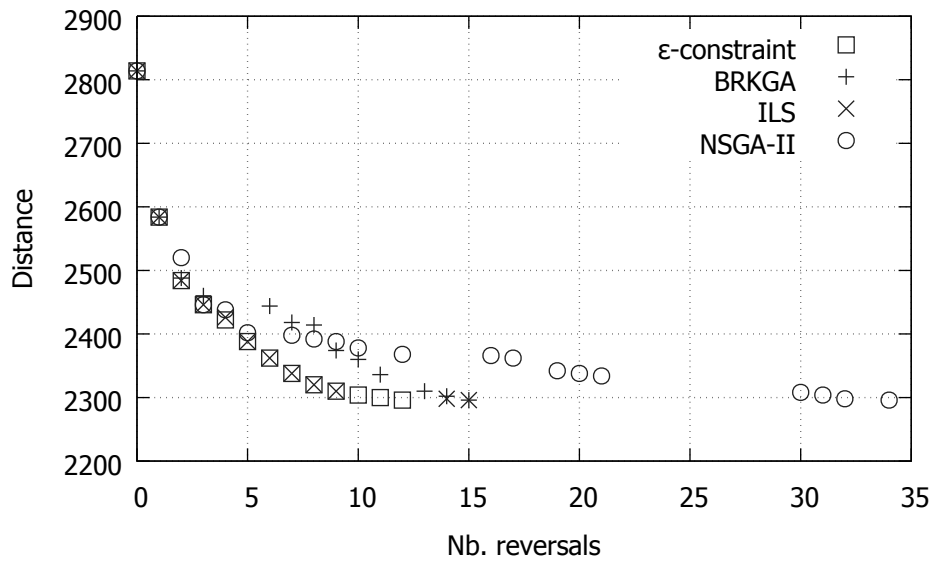


Figure C.6: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-50

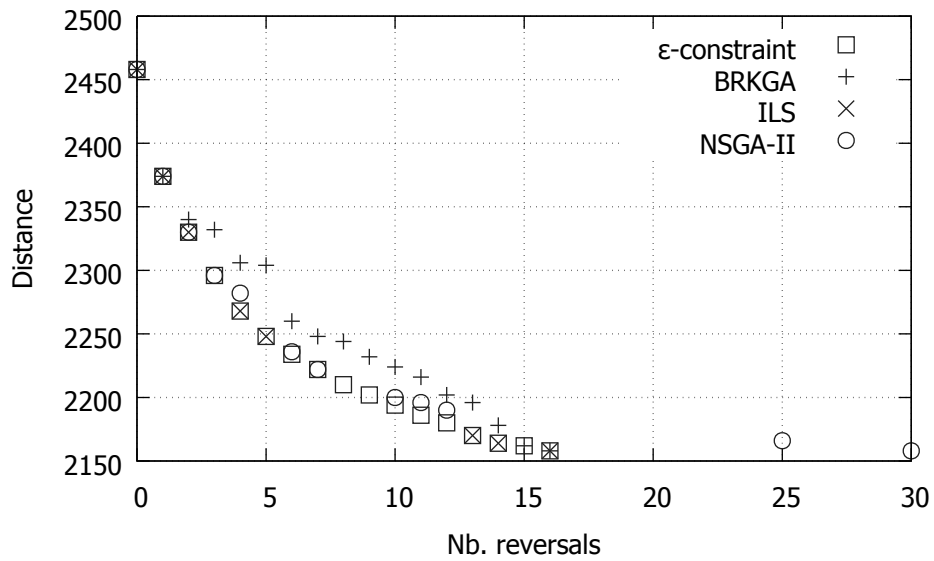


Figure C.7: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-75

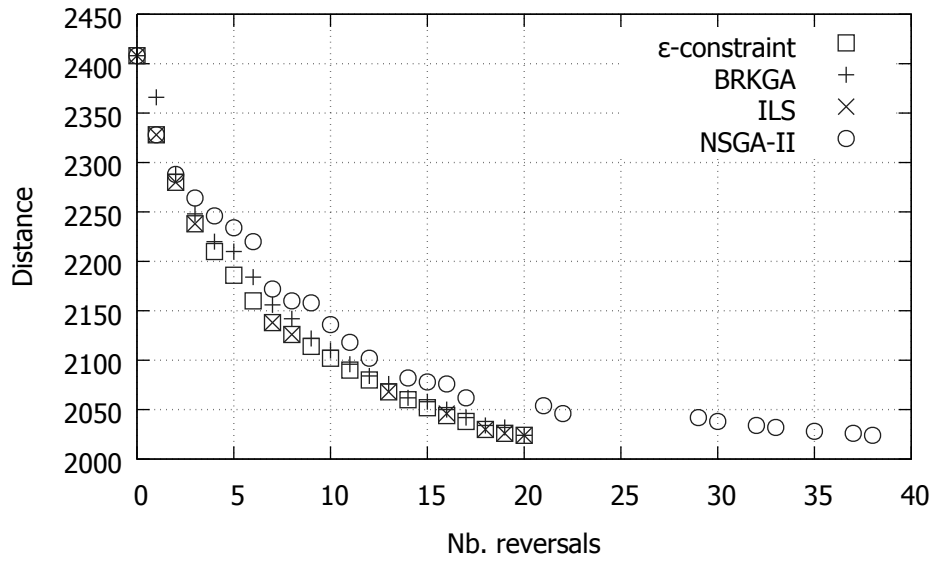


Figure C.8: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b1-100

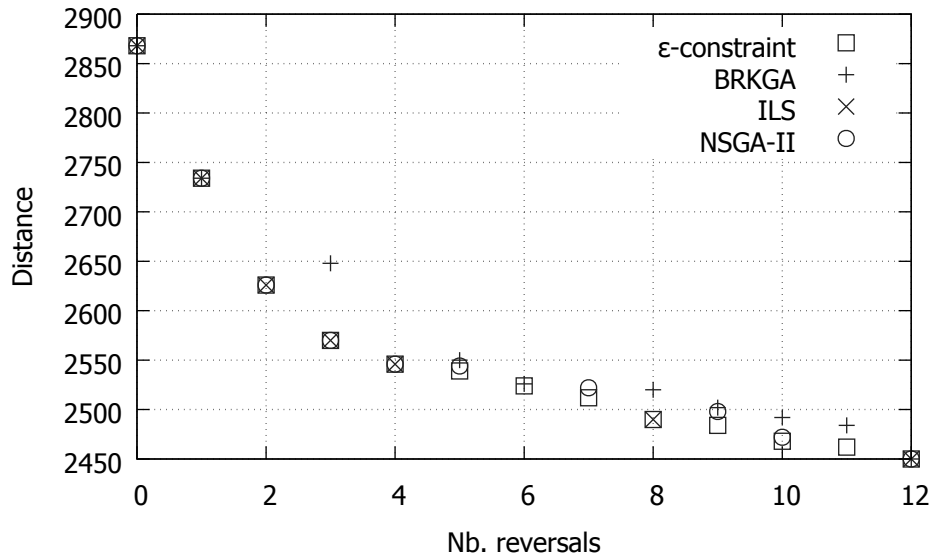


Figure C.9: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-25

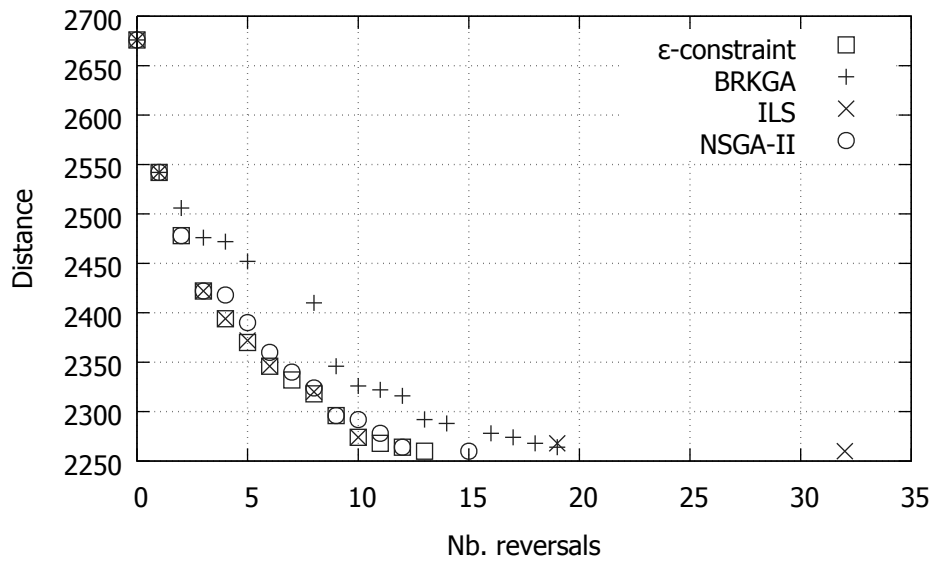


Figure C.10: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-50

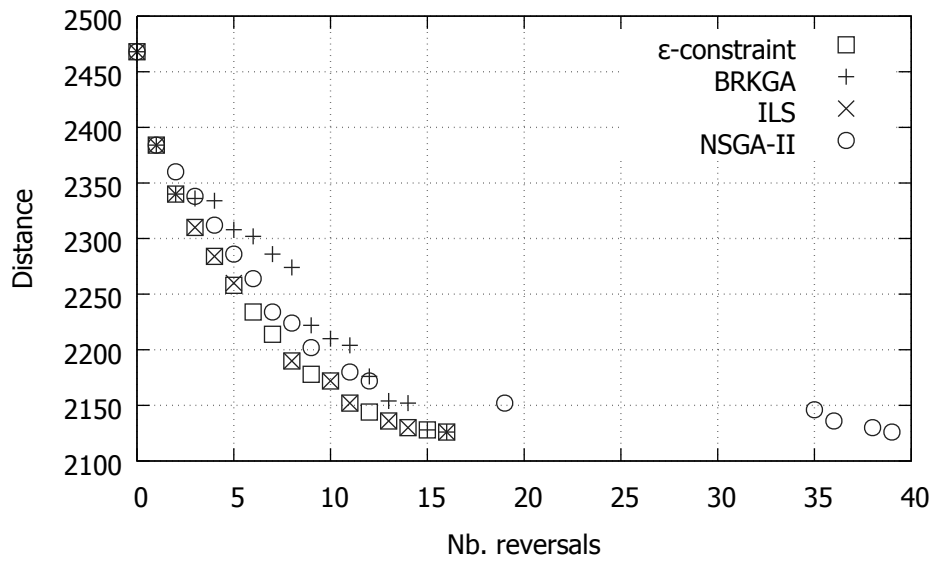


Figure C.11: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-75

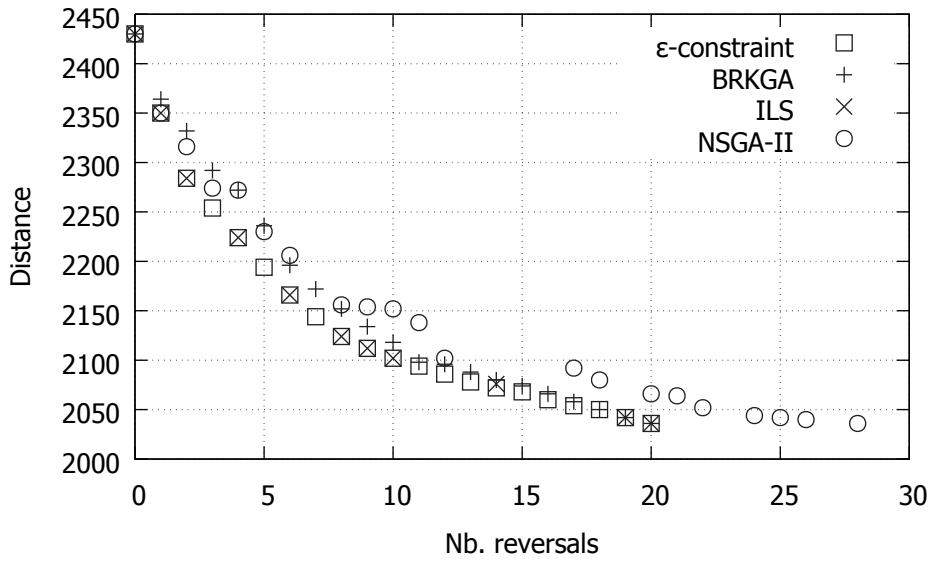


Figure C.12: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance 5x5-b2-100

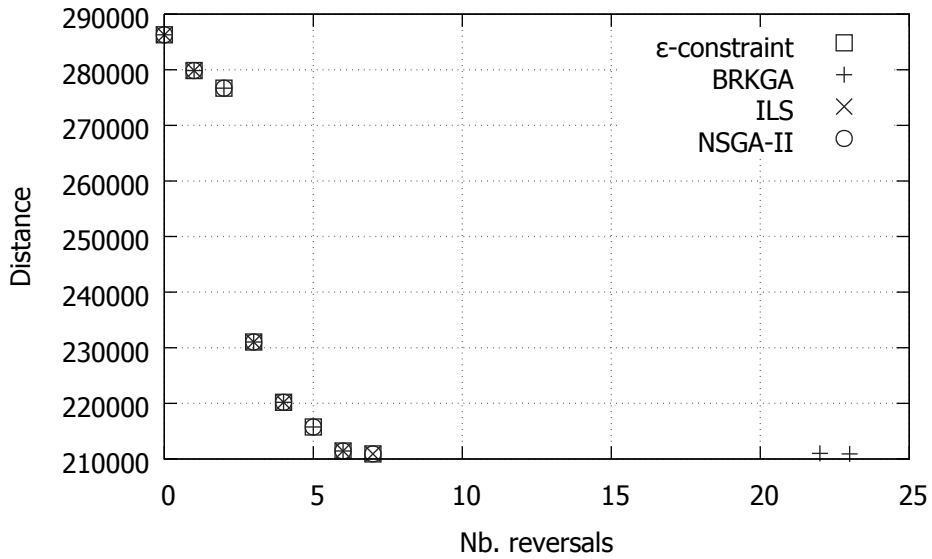


Figure C.13: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b1

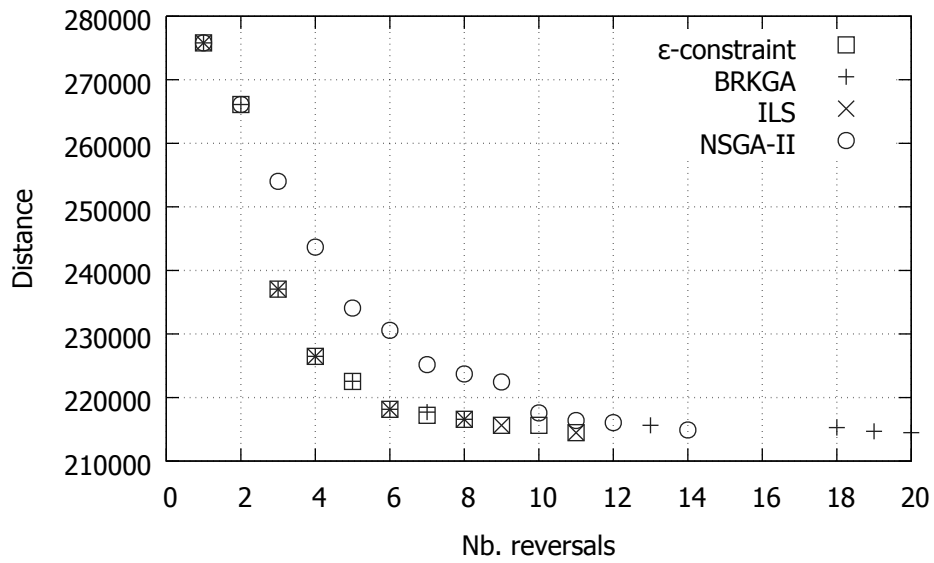


Figure C.14: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b2

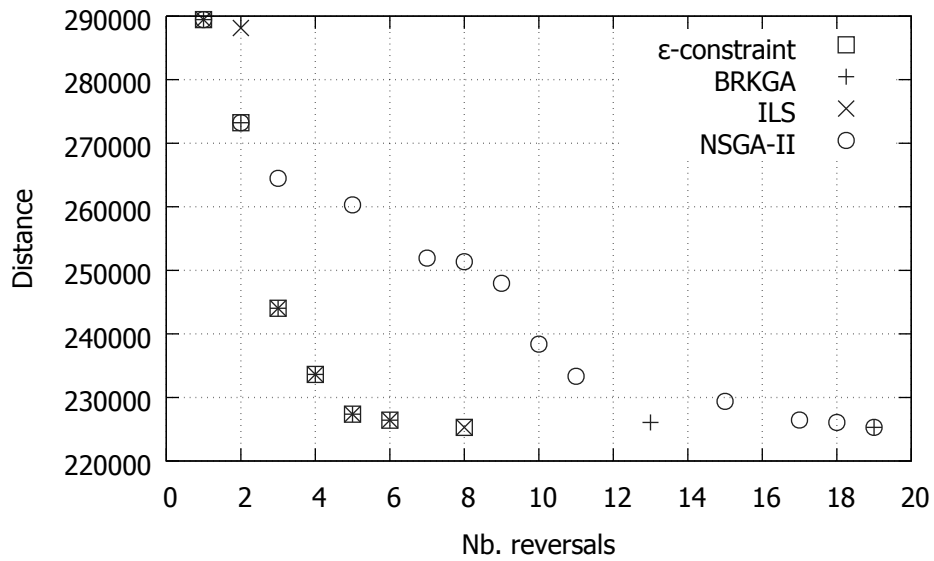


Figure C.15: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b4

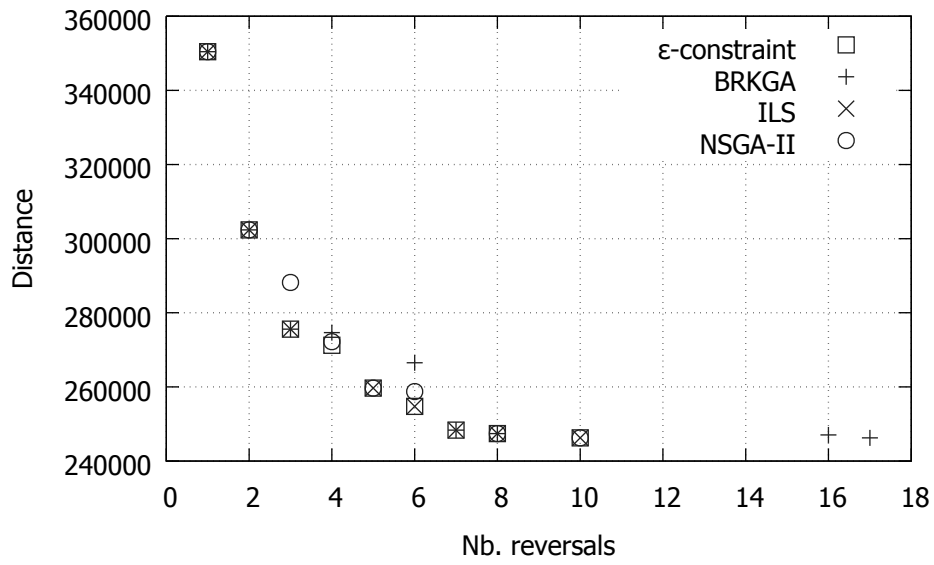


Figure C.16: Pareto fronts of ϵ -constraint method, BRKGA, ILS and NSGA-II on the instance troyes-v24-b6

Appendix D

Figures of Pareto fronts of BRKGA, ILS and NSGA-II

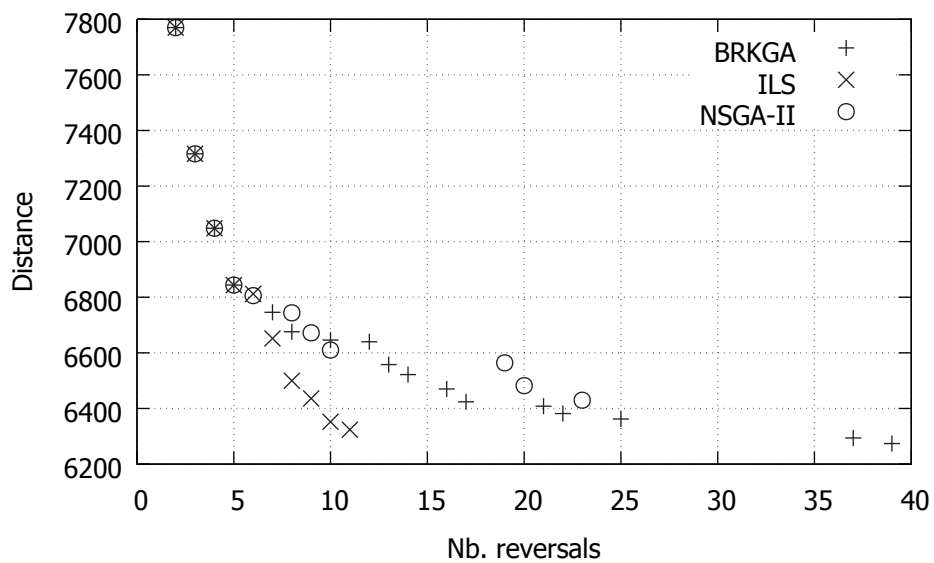


Figure D.1: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1

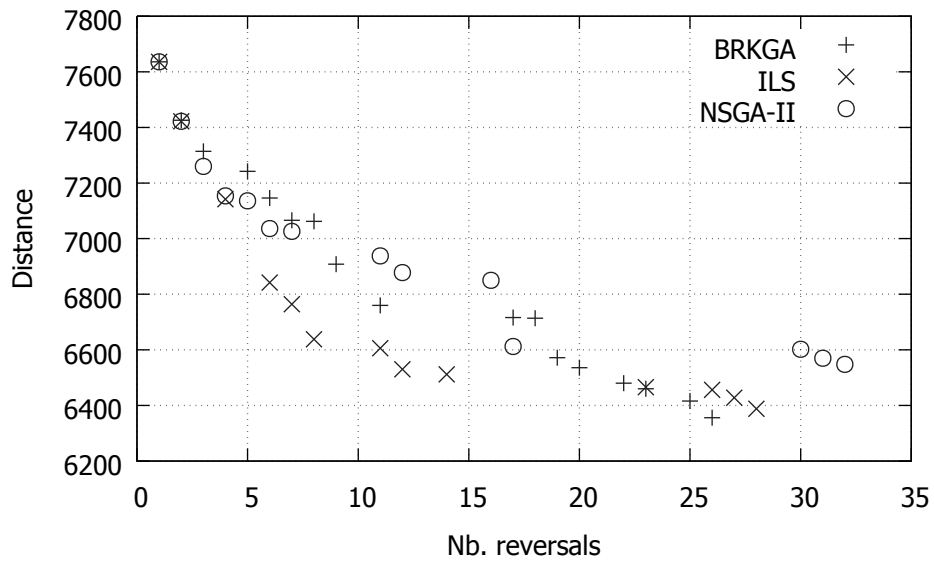


Figure D.2: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2

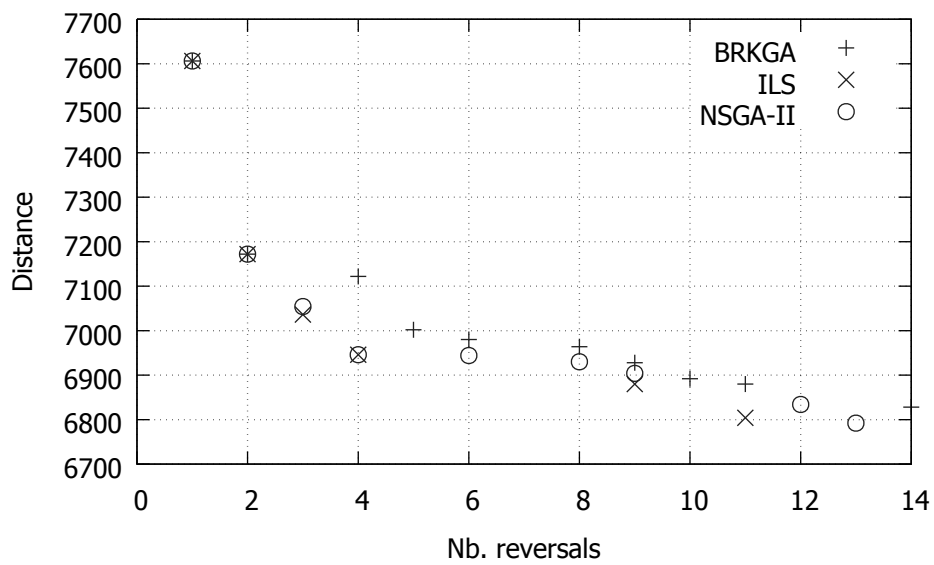


Figure D.3: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b4

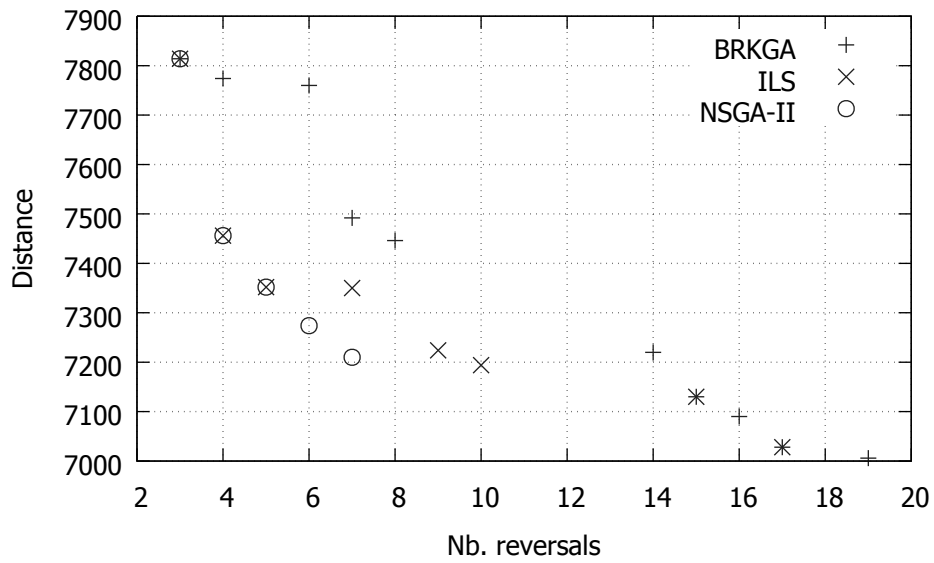


Figure D.4: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b6

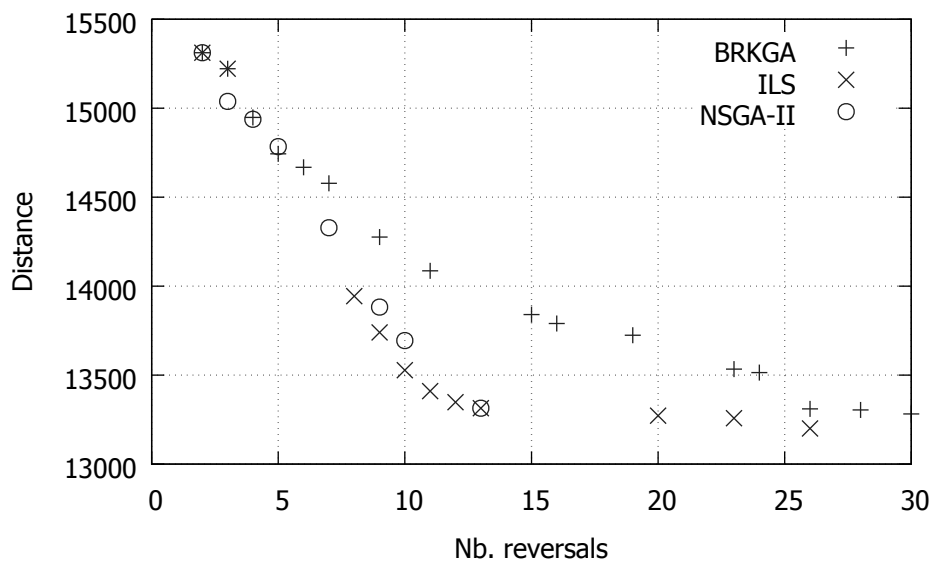


Figure D.5: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b1

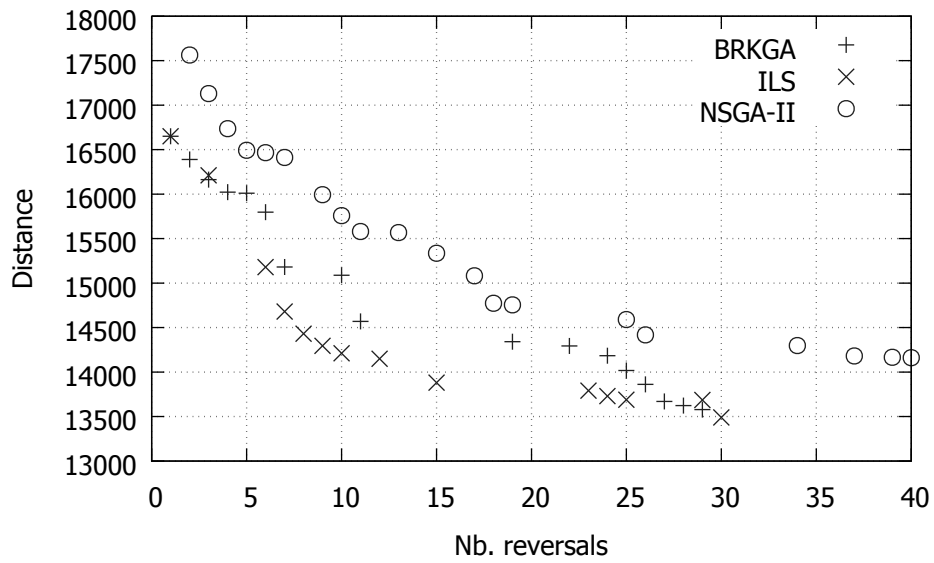


Figure D.6: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b2

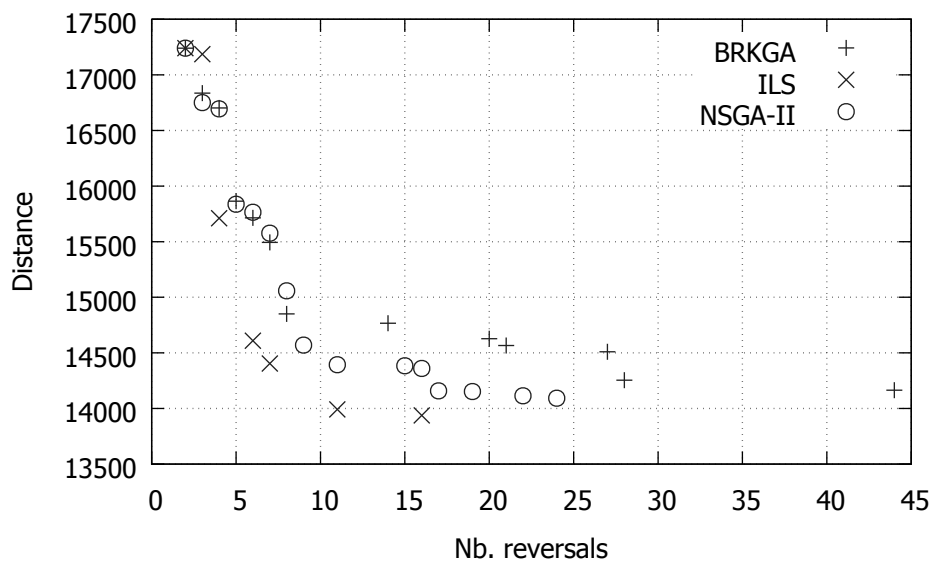


Figure D.7: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b4

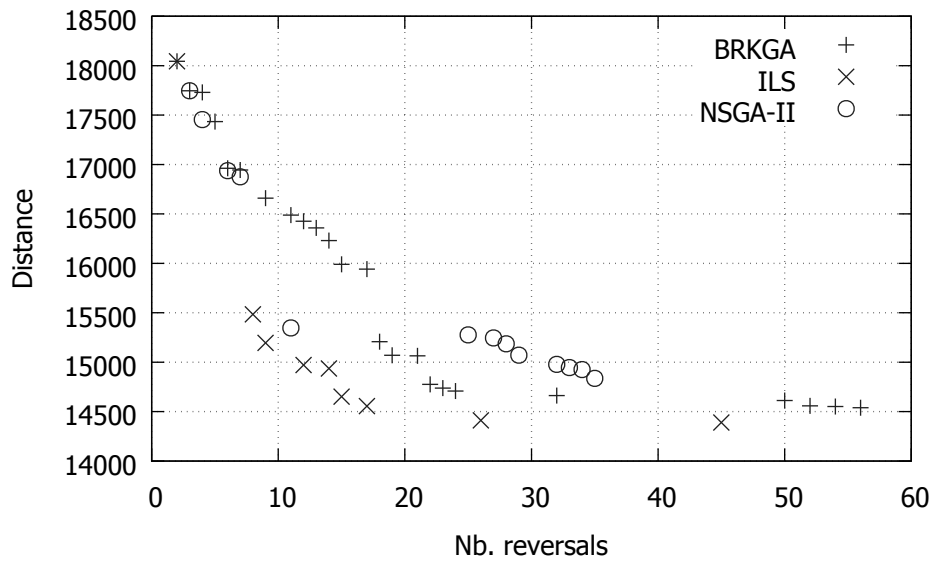


Figure D.8: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 7x7-b6

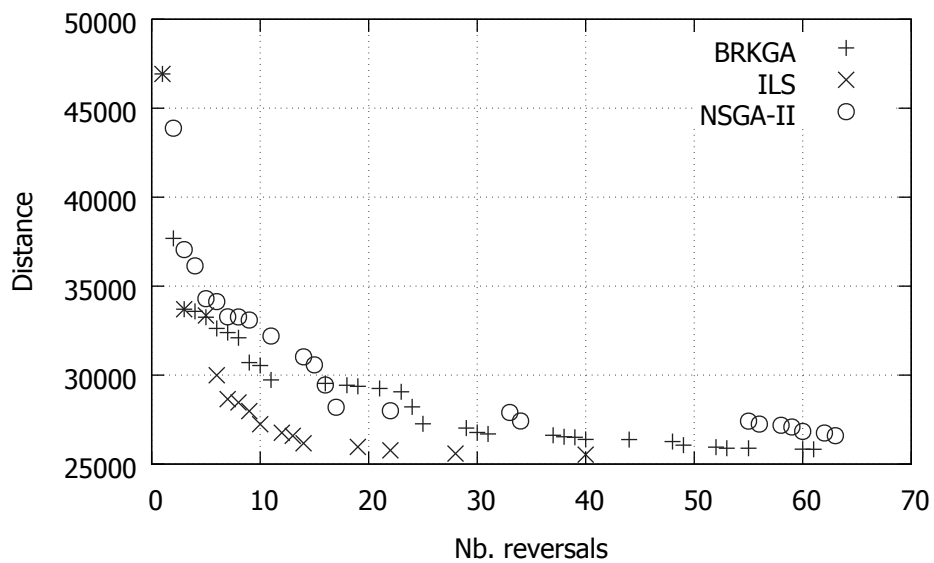


Figure D.9: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b1

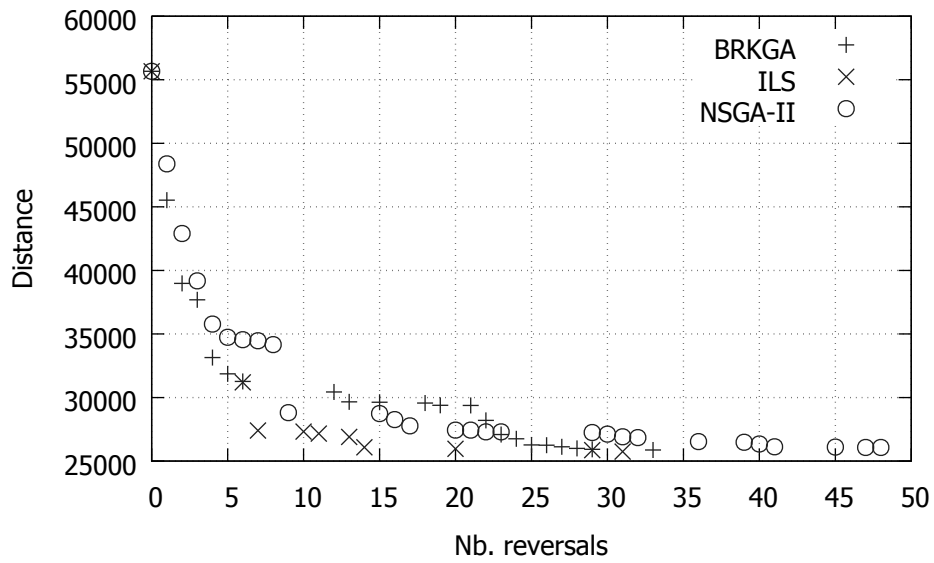


Figure D.10: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b2

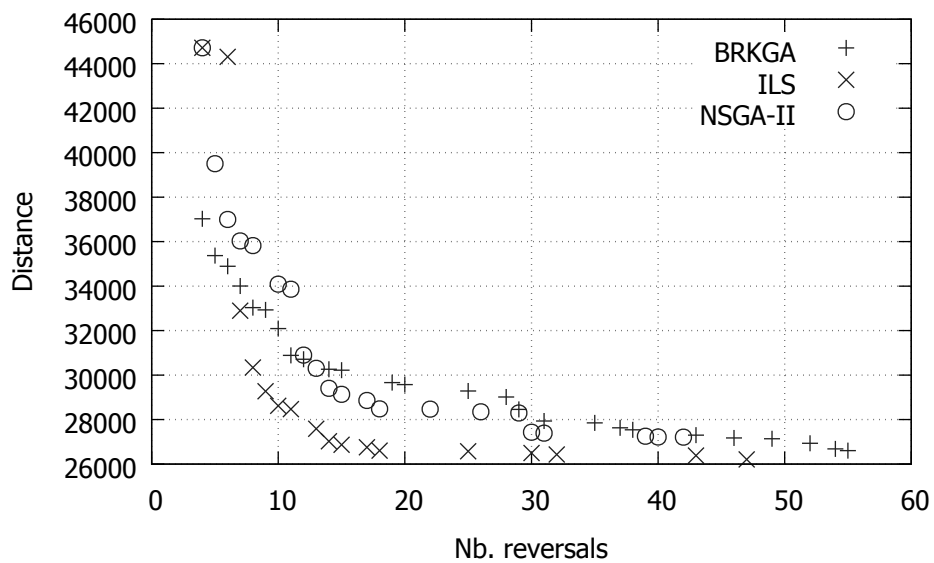


Figure D.11: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b4

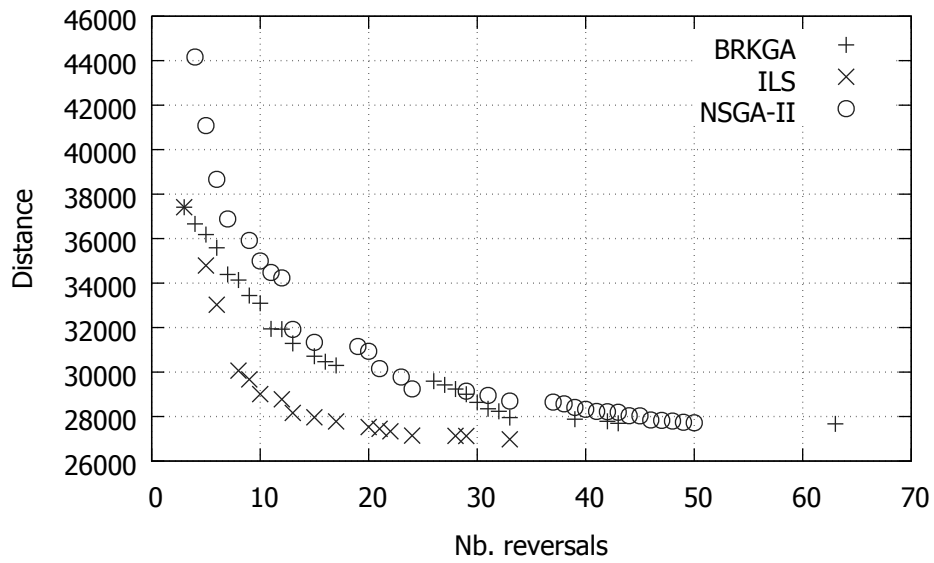


Figure D.12: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 8x8-b6

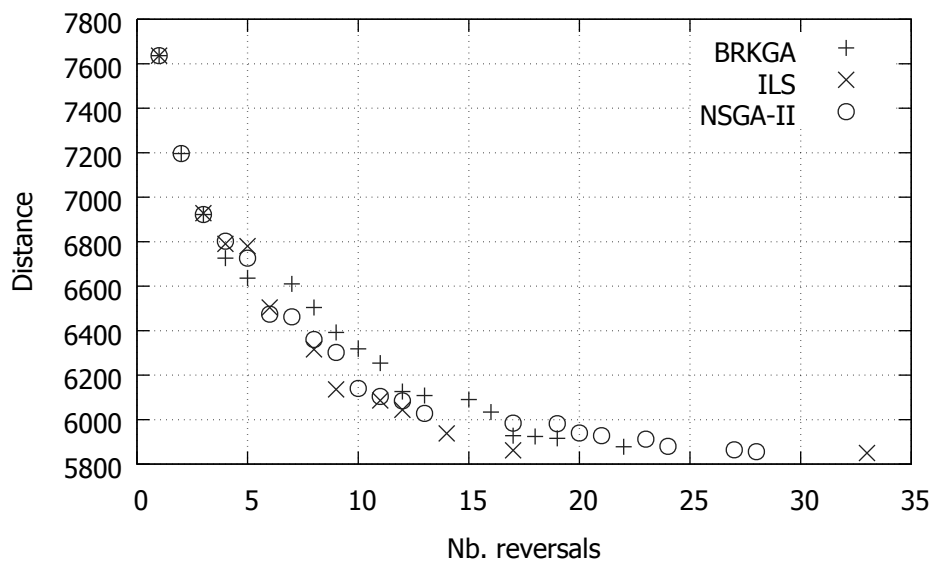


Figure D.13: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-25

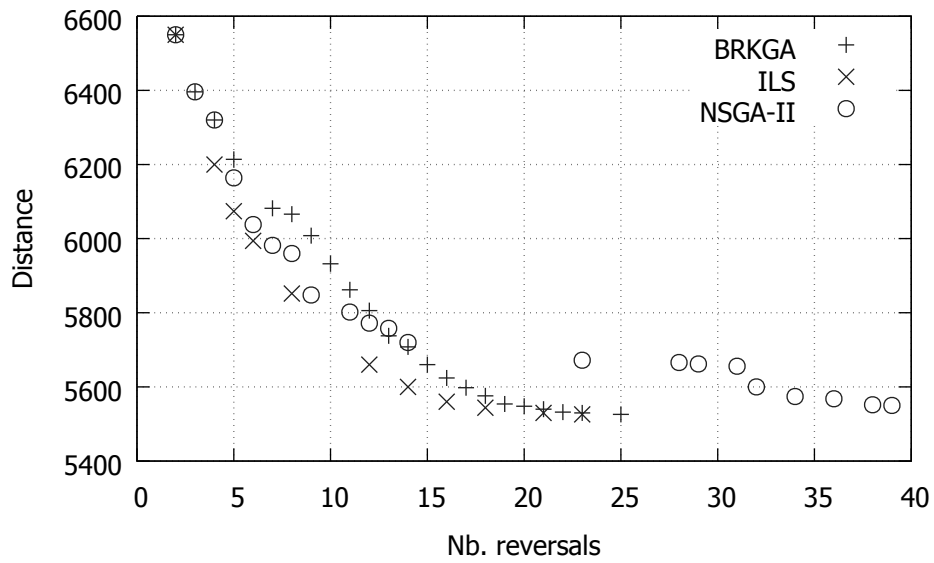


Figure D.14: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-50

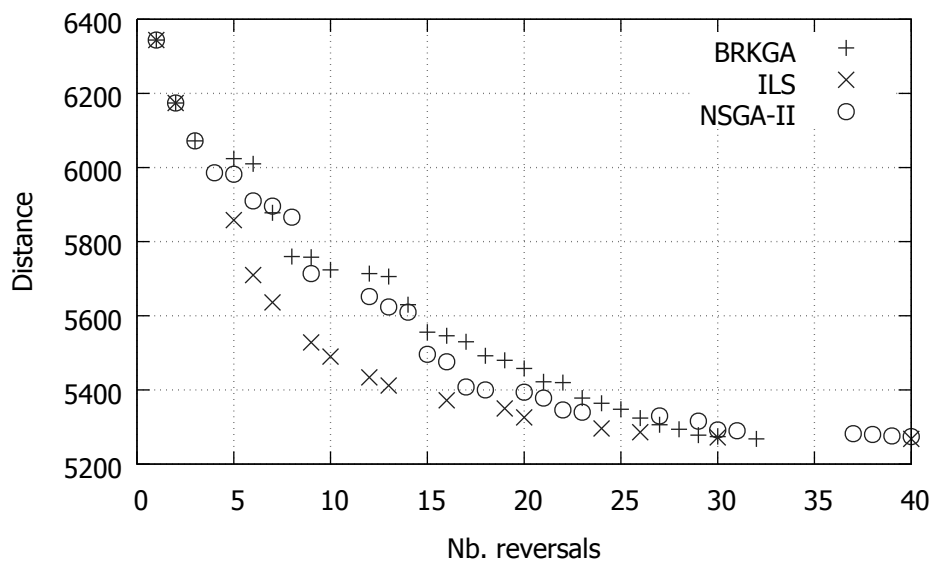


Figure D.15: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-75

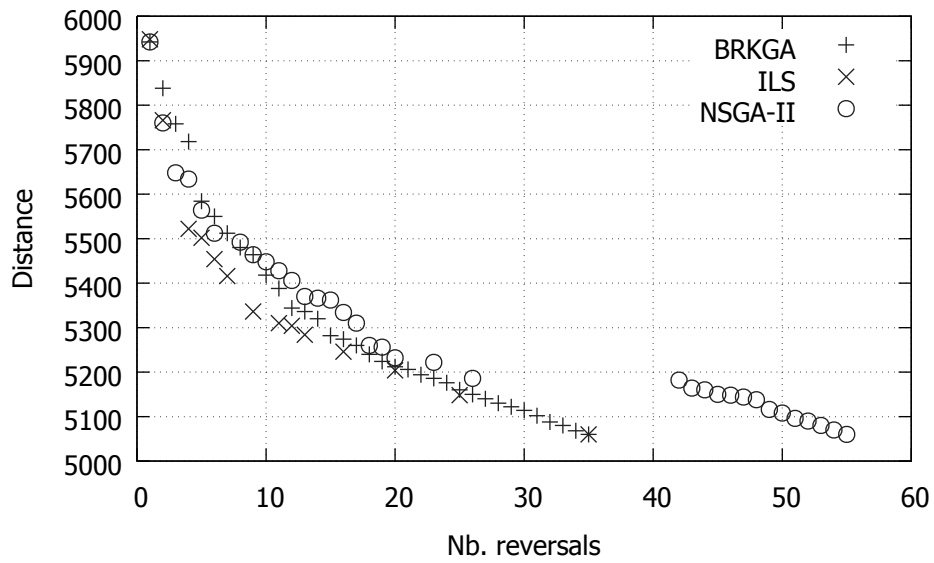


Figure D.16: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b1-100

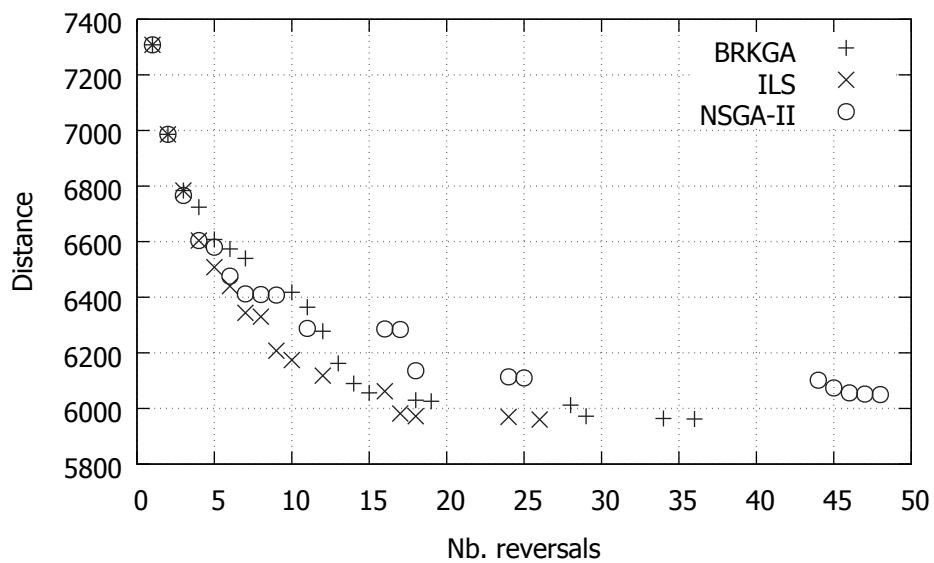


Figure D.17: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-25

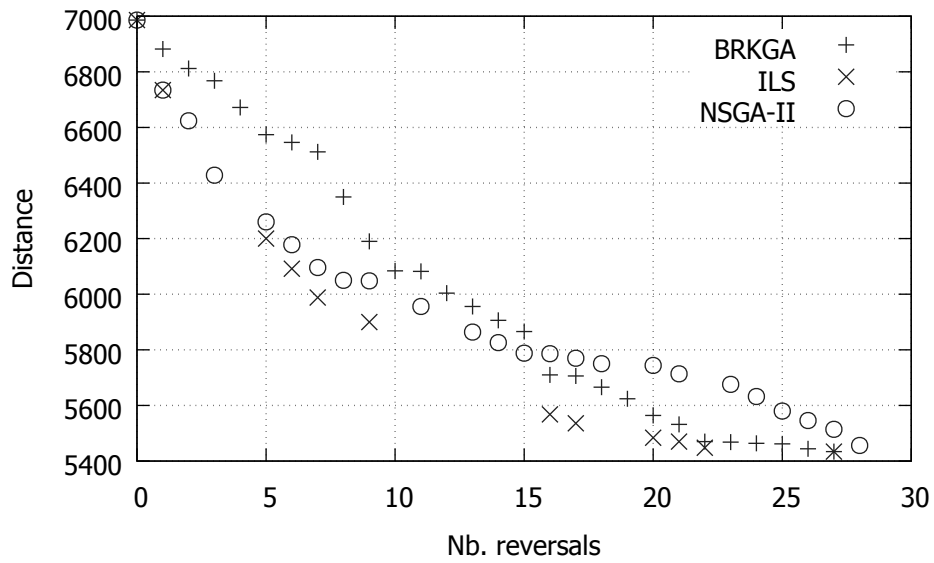


Figure D.18: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-50

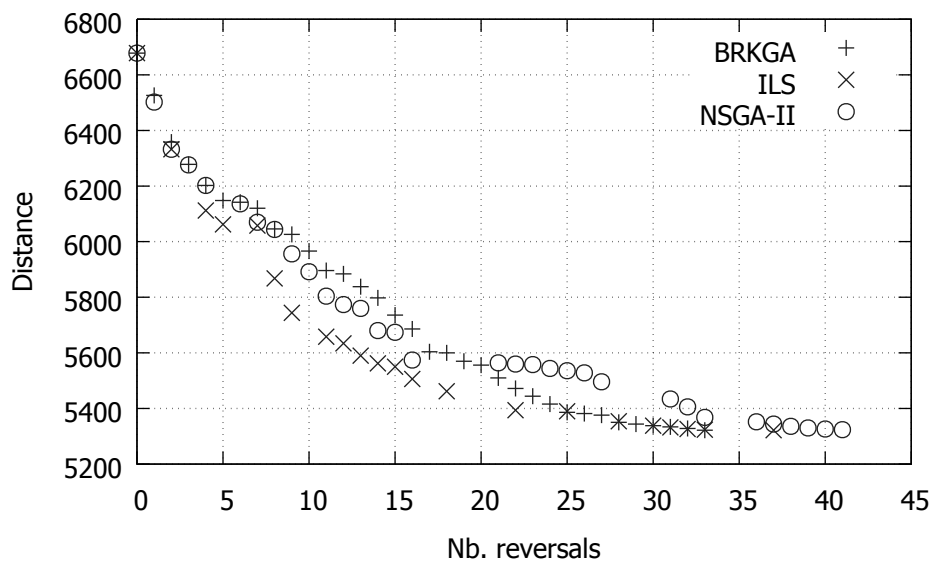


Figure D.19: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-75

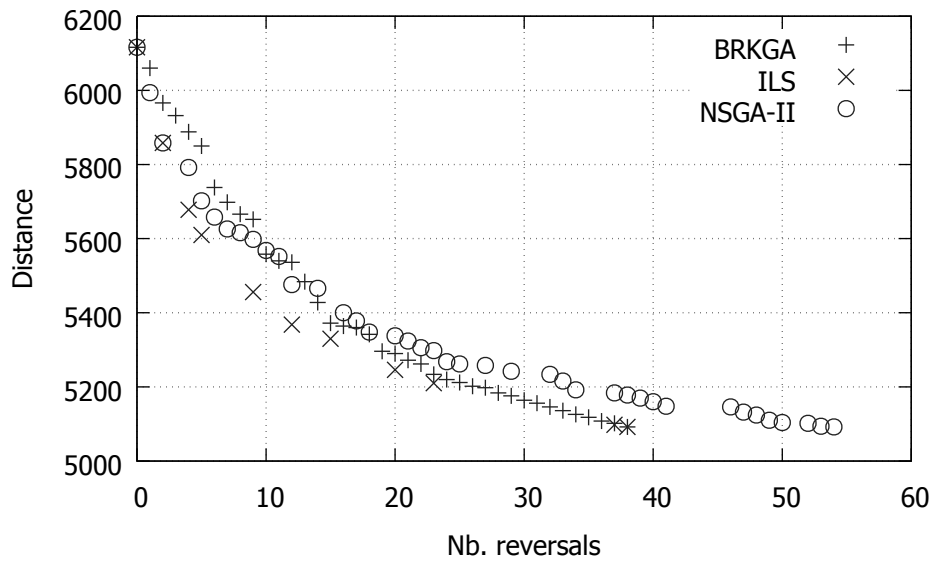


Figure D.20: Pareto fronts of BRKGA, ILS and NSGA-II on the instance 6x6-b2-100

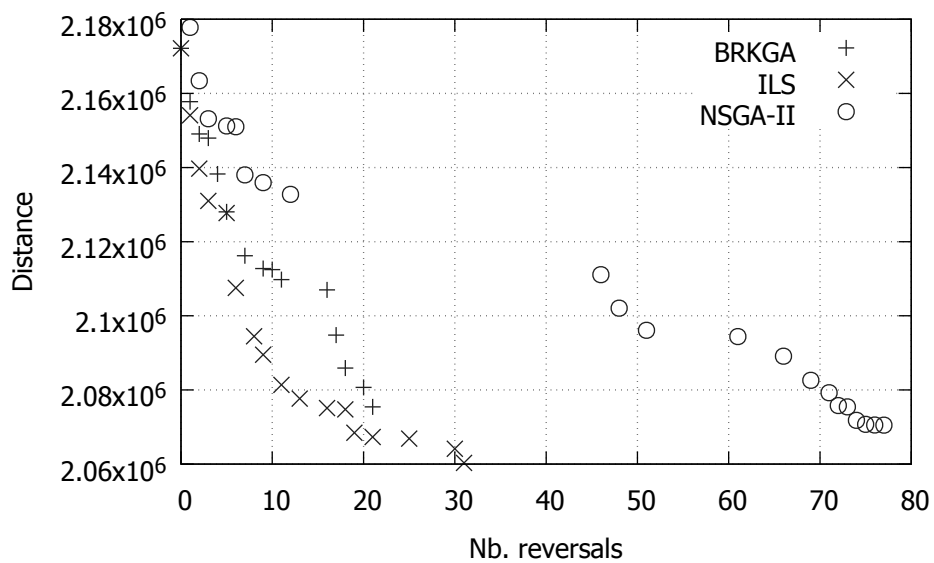


Figure D.21: Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b1

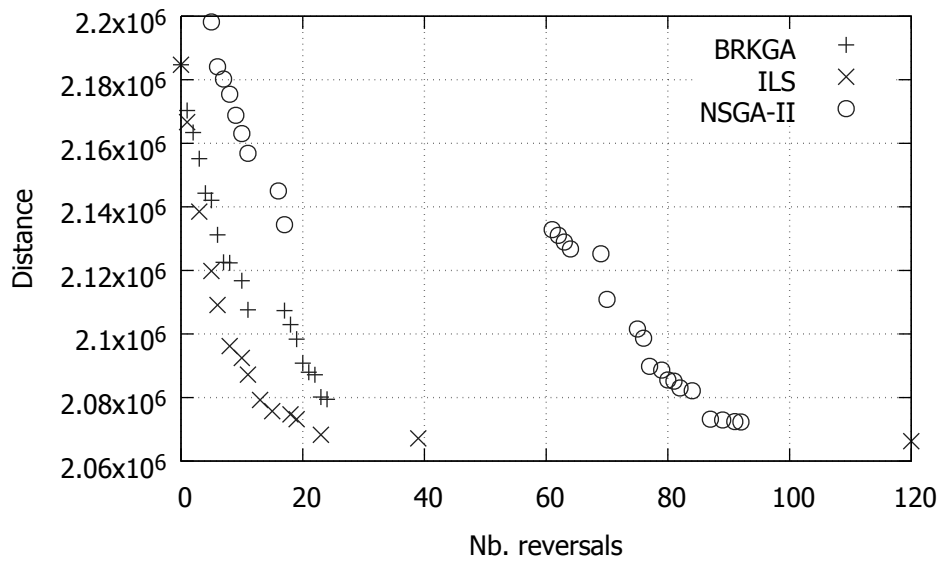


Figure D.22: Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b2

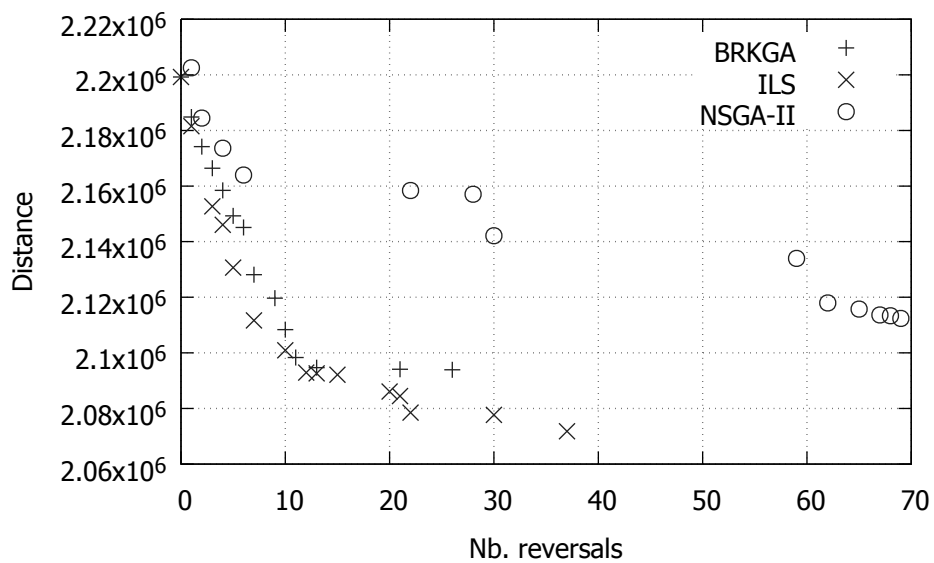


Figure D.23: Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b4

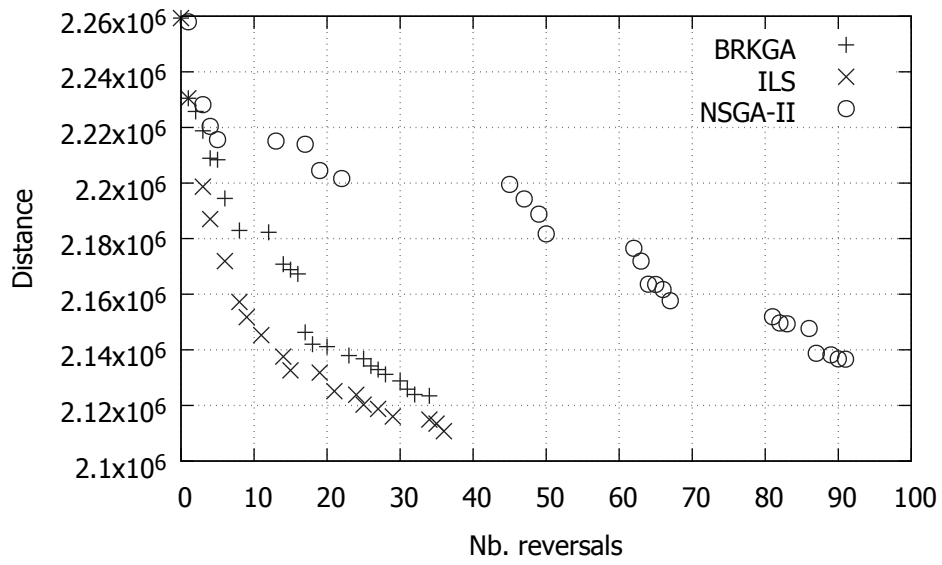


Figure D.24: Pareto fronts of BRKGA, ILS and NSGA-II on the instance troyes-v58-b6

Bibliography

- Ádám, A. 1963. Theory of graphs and its applications. *Page 157 of: Symp. Smolenice.*
- Aiex, Renata M., Resende, Mauricio G.C., & Ribeiro, Celso C. 2002. Probability Distribution of Solution Time in GRASP: An Experimental Investigation. *Journal of Heuristics*, **8**(3), 343–373.
- Aiex, Renata M., Resende, Mauricio G. C., & Ribeiro, Celso C. 2007. TTT plots: a perl program to create time-to-target plots. *Optimization Letters*, **1**(4), 355–366.
- Bang-Jensen, Jørgen, & Gutin, Gregory Z. 2008. *Digraphs: Theory, Algorithms and Applications*. 2nd edn. Springer Publishing Company, Incorporated.
- Bean, James C. 1994. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA Journal on Computing*, **6**(2), 154–160.
- Bérubé, Jean-François, Gendreau, Michel, & Potvin, Jean-Yves. 2009. An exact ϵ -constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research*, **194**(1), 39–50.
- Boesch, F., & Tindell, R. 1980. Robbins’s Theorem for Mixed Multigraphs. *The American Mathematical Monthly*, **87**(9), 716–719.
- Bogdanowicz, Zbigniew R. 2011. On arc reversal in balanced digraphs. *Discrete Mathematics*, **311**(6), 435–436.
- Boyce, D E. 1984. Urban transportation network-equilibrium and design models: recent achievements and future prospects. *Environment and Planning A*, **16**(11), 1445–1474.
- Boyce, D.E., & Janson, B.N. 1980. A discrete transportation network design problem with combined trip distribution and assignment. *Transportation Research Part B: Methodological*, **14**(1), 147–154.
- Braess, D. 1968. Über ein Paradoxon aus der Verkehrsplanung. *Unternehmensforschung*, **12**(1), 258–268.

Buisson, Christine, & Lesort, JB. 2010. *Comprendre le trafic routier. Méthodes et calculs*. Coll. Références. CERTU.

Burkard, R.E., Feldbacher, K., Klinz, B., & Woeginger, G.J. 1999. Minimum-cost strong network orientation problems: Classification, complexity, and algorithms. *Networks*, **33**(1), 57–70.

Cantarella, G.E., Pavone, G., & Vitetta, A. 2006. Heuristics for urban road network design: Lane layout and signal settings. *European Journal of Operational Research*, **175**(3), 1682–1695.

Cascetta, E. 2001. *Transportation Systems Engineering: Theory and Methods*. Boston, MA: Springer US.

Chankong, Vira, & Haimes, Yacov Y. 1983. *Multiobjective Decision Making: Theory and Methodology*. New York: North-Holland: Elsevier Science Publishing.

Chinchuluu, A., & Pardalos, P. M. 2007. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, **154**, 29–50.

Chung, F.R.K., Garey, M.R., & Tarjan, R.E. 1985. Strongly connected orientations of mixed multigraphs. *Networks*, **15**(4), 477–484.

Chvátal, V., & Thomassen, C. 1978. Distances in orientations of graphs. *Journal of Combinatorial Theory, Series B*, **24**(1), 61–75.

Conte, A., Grossi, R., Marino, A., Rizzi, R., & Versari, L. 2016. Directing Road Networks by Listing Strong Orientations. *Pages 83–95 of: VeliMäkinen, Puglisi, Simon J., & Salmela, Leena (eds), Combinatorial Algorithms - 27th International Workshop, IWOCA 2016, Helsinki, Finland, August 17-19, 2016, Proceedings*. Lecture Notes in Computer Science, vol. 9843. Springer.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. 2009. *Introduction to Algorithms*. 3rd edn. The MIT Press.

De Sousa, Ernando Gomes, Santos, Andréa Cynthia, & Aloise, Dario José. 2015. An exact method for solving the bi-objective Minimum Diameter-Cost Spanning Tree Problem. *RAIRO-Operations Research*, **49**(1), 143–160.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**(2), 182–197.

- Deb, Kalyanmoy. 1989. *Genetic Algorithms in Multimodal Function Optimization*. M.Phil. thesis, University of Alabama, Tuscaloosa, Alabama, USA.
- Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numer. Math.*, **1**(1), 269–271.
- Dorigo, Marco, & Stützle, Thomas. 2004. *Ant Colony Optimization*. Scituate, MA: MIT Press.
- Ehrgott, M., & Gandibleux, X. 2000. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, **22**, 425–460.
- Farahani, R.Z., Miandoabchi, E., Szeto, W.Y., & Rashidi, H. 2013. A review of urban transportation network design problems. *European Journal of Operational Research*, **229**(2), 281–302.
- Feo, Thomas A., & Resende, Mauricio G. C. 1995. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization*, **6**(2), 109–133.
- Feo, Thomas A, & Resende, Mauricio G.C. 1989. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, **8**(2), 67–71.
- Feremans, Corinne, Labbé, Martine, & Laporte, Gilbert. 2003. Generalized network design problems. *European Journal of Operational Research*, **148**(1), 1–13.
- Fonseca, Carlos M., & Fleming, Peter J. 1993. Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization. *Pages 416–423 of: Proceedings of the 5th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- Friesz, Terry L. 1985. Transportation network equilibrium, design and aggregation: Key developments and research opportunities. *Transportation Research Part A: General*, **19**(5), 413–427. Special Issue Transportation Research: The State of the Art and Research Opportunities.
- Gallo, M., D’Acierno, L., & Montella, B. 2012. A Meta-heuristic Algorithm for Solving the Road Network Design Problem in Regional Contexts. *Procedia - Social and Behavioral Sciences*, **54**(0), 84–95.
- Glover, Fred. 1977. Heuristics for integer programming using surrogate constraints. *Decision Sciences*, **8**(1), 156–166.

- Glover, Fred. 1989. Tabu Search — Part I. *ORSA Journal on Computing*, **1**(3), 190–206.
- Glover, Fred. 1990. Tabu Search — Part II. *ORSA Journal on Computing*, **2**(1), 4–32.
- Glover, Fred, Laguna, Manuel, & Marti, Rafael. 2003. *Scatter Search*. Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 519–537.
- Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st edn. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.
- Gomaa, Wael H, & Fahmy, Aly A. 2013. A survey of text similarity approaches. *International Journal of Computer Applications*, **68**(13).
- Gonçalves, José Fernando, & Resende, Mauricio G. C. 2011. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, **17**(5), 487–525.
- Gouveia, Luis, & Magnanti, Thomas L. 2003. Network flow models for designing diameter-constrained minimum-spanning and Steiner trees. *Networks*, **41**(3), 159–173.
- Haimes, Y., Lasdon, L., & Wismer, D. 1971. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-1**(3), 296–297.
- Henry-Labordere, A. L. 1969. The record balancing problem: A dynamic programming solution of a generalized traveling salesman problem. *RIBO*, **B-2**, 736–743.
- Hertz, Alain, & Mittaz, Michel. 2001. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. *Transportation science*, **35**(4), 425–434.
- Horn, J., Nafpliotis, N., & Goldberg, D. E. 1994. A niched Pareto genetic algorithm for multiobjective optimization. *Pages 82–87 of: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, vol. 1.
- Institut national de la statistique et des études économiques (France), Direction des études et synthèses économiques, & Berthier, J.P. 1998. *Congestion urbaine: un modèle de trafic de pointe à courbe débit-vitesse et demande élastique*. Document de travail / INSEE, Direction des études et synthèses économiques. INSEE.
- Italiano, Giuseppe F., Laura, Luigi, & Santaroni, Federico. 2012. Finding strong bridges and strong articulation points in linear time. *Theoretical Computer Science*, **447**, 74 – 84.

- Jirásek, Jozef. 1987. On a certain class of multidigraphs, for which reversal of no arc decreases the number of their cycles. *Commentationes Mathematicae Universitatis Carolinae*, **28**(1), 185–189.
- Jirásek, Jozef. 2005. Arc reversal in nonhamiltonian circulant oriented graphs. *Journal of Graph Theory*, **49**(1), 59–68.
- Kalyanmoy, Deb. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc.
- Kirkpatrick, Scott. 1984. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, **34**(5), 975–986.
- L., J.Enrique Fernandez, & Friesz, Terry L. 1983. Equilibrium predictions in transportation markets: The state of the art. *Transportation Research Part B: Methodological*, **17**(2), 155 – 172.
- Li, Wen-Jui, Tsao, H.-S. Jacob, & Ulular, Osman. 1995. The shortest path with at most l nodes in each of the series/parallel clusters. *Networks*, **26**(4), 263–271.
- Lourenço, Helena R., Martin, Olivier C., & Stützle, Thomas. 2003. *Iterated Local Search*. Boston, MA: Springer US. Pages 320–353.
- Lourenço, Helena R., Martin, Olivier C., & Stützle, Thomas. 2010. *Iterated Local Search: Framework and Applications*. Boston, MA: Springer US. Pages 363–397.
- Magnanti, T. L., & Wong, R. T. 1984. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science*, **18**(1), 1–55.
- Miandoabchi, Elnaz, & Farahani, Reza Zanjirani. 2011. Optimizing reserve capacity of urban road networks in a discrete Network Design Problem. *Advances in Engineering Software*, **42**(12), 1041 – 1050.
- Mladenović, N., & Hansen, P. 1997. Variable neighborhood search. *Computers & Operations Research*, **24**(11), 1097–1100.
- Myung, Young-Soo, Lee, Chang-Ho, & Tcha, Dong-Wan. 1995. On the generalized minimum spanning tree problem. *Networks*, **26**(4), 231–241.
- Pareto, V. 1906. *Manuale di economia politica*. Piccola biblioteca scientifica. Società Editrice Libreria.

Powell, Warren B., & Sheffi, Yosef. 1982. The Convergence of Equilibrium Algorithms with Predetermined Step Sizes. *Transportation Science*, **16**(1), 45–55.

Resende, Mauricio G. C. 2012. Biased random-key genetic algorithms with applications in telecommunications. *TOP*, **20**(1), 130–153.

Robbins, H.E. 1939. A theorem on graphs, with an application to a problem on traffic control. *The American Mathematical Monthly*, **46**, 281–283.

Roberts, F.S. 1976. *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems*. Prentice-Hall, Englewood Cliffs, NJ.

Roberts, F.S. 1978. *Graph theory and its applications to problems of society*. CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 29. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa.

Santos, A. C., Duhamel, C., & Prins, C. 2013. Heuristics for setting directions in urban networks. *Pages 1–4 of: The X Metaheuristics International Conference (MIC)*.

Santos, Andréa Cynthia, Lima, Diego Rocha, & Aloise, Dario José. 2014. Modeling and solving the bi-objective minimum diameter-cost spanning tree problem. *Journal of Global Optimization*, **60**(2), 195–216.

Saskena, J. P. 1970. Mathematical model of scheduling clients through welfare agencies. *Journal of the Canadian Operational Research Society*, **8**, 185–200.

Schaffer, James David. 1984. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms (Artificial Intelligence, Optimization, Adaptation, Pattern Recognition)*. Ph.D. thesis, Vanderbilt University, Nashville, TN, USA.

Sever, Derya, Dellaert, Nico, van Woensel, Tom, & de Kok, Ton. 2013. Dynamic shortest path problems: Hybrid routing policies considering network disruptions. *Computers & Operations Research*, **40**(12), 2852–2863.

Srinivas, N., & Deb, Kalyanmoy. 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evol. Comput.*, **2**(3), 221–248.

Srivastava, S. S., Kumar, S., Garg, R. C., & Sen, P. 1969. Generalized traveling salesman problem through n sets of nodes. *Journal of the Canadian Operational Research Society*, **7**, 97–101.

- Stefanello, F., Buriol, L. S., Hirsch, M. J., Pardalos, P. M., Querido, T., Resende, M. G. C., & Ritt, M. 2017. On the minimization of traffic congestion in road networks with tolls. *Annals of Operations Research*, **249**(1), 119–139.
- Tarjan, Robert. 1972. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, **1**(2), 146–160.
- Thomassen, Carsten. 1987. Counterexamples to *Ádám's* conjecture on arc reversals in directed graphs. *Journal of Combinatorial Theory, Series B*, **42**(1), 128–130.
- Veldhuizen, D. A. Van, & Lamont, G. B. 2000. On measuring multiobjective evolutionary algorithm performance. *Pages 204–211 of: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1.
- Winkler, William E. 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Pages 354–359 of: Proceedings of the Section on Survey Research*.
- Yang, H., & Bell, Michael G. H. 1998. Models and algorithms for road network design: a review and some new developments. *Transport Reviews*, **18**(3), 257–278.
- Zadeh, L. 1963. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, **8**(1), 59–60.
- Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N., & Zhang, Q. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, **1**(1), 32–49.
- Zilske, Michael, Neumann, Andreas, & Nagel, Kai. 2011. OpenStreetMap for traffic simulation. *Pages 126–134 of: Proceedings of the 1st European state of the map : OpenStreetMap conference*.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & da Fonseca, V. G. 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, **7**(2), 117–132.
- Zitzler, Eckart, & Thiele, Lothar. 1998. *Multiobjective optimization using evolutionary algorithms — A comparative case study*. Berlin, Heidelberg: Springer Berlin Heidelberg. Pages 292–301.
- Zopounidis, C., & Pardalos, P.M. 2010. *Handbook of Multicriteria Analysis*. Applied Optimization, vol. 103. Springer Berlin Heidelberg.

Yipeng HUANG

Doctorat : Optimisation et Sûreté des Systèmes

Année 2018

Problèmes du réseau de transport urbain : modèles, méthodes et application

L'augmentation du flux de trafic dans des zones urbaines a un fort impact sur les infrastructures de réseaux de transport. La situation devient encore plus compliquée lorsqu'il y a des interruptions qui perturbent la circulation. Dans cette thèse, deux objectifs majeurs sont ciblés : le premier consiste à l'étude du problème de conception de réseau routier avec des interruptions tout en assurant la forte connectivité du réseau (RND) ; le second porte sur le développement du système d'aide à la décision : Optimal traffic Deviation System (ODS). En fonction de la nature du réseau, RND est spécialisé en Unidirectionnel RND et Multidirectionnel RND. Des modèles mathématiques multi-flot sont proposés pour ces problèmes. De plus, des méta-heuristiques du type Biased Random Key Genetic Algorithm (BRKGA) et Iterated Local Search (ILS) sont développées pour les problèmes en versions mono-objectif. Des adaptations pour ces méthodes sont ensuite proposées pour traiter les problèmes en version bi-objectif. En outre, des méthodes multi-objectif classiques sont également développées, telles que la méthode ϵ -constraint et Non-dominated Sorting Genetic Algorithm II. Les résultats numériques indiquent que l'ILS dans les versions mono et bi-objectif produit les meilleurs résultats que les autres méthodes. Enfin, le système ODS est connecté à des systèmes hétérogènes de données géographiques et à la base des arrêtés municipaux de la ville de Troyes, incluant plusieurs contraintes, critères d'optimisation et algorithmes dédiés permettant de proposer des déviations.

Mots clés : optimisation combinatoire - recherche opérationnelle – métaheuristiques - génie logiciel – réseaux urbains.

Urban Transportation Road Network Problems: Models, Methods and Application

The increase of the traffic flow in urban areas has a strong impact on the infrastructures of transportation networks. The situation becomes more complex whenever disruptions occur on the road networks, which often reduce the traffic capacity. This thesis comprises two major goals. The first objective consists of an academic study on Road Network problems with Disruptions and connecting requirements (RND). The other one is dedicated to the software engineering of a decision-support system, named Optimal traffic Deviation System (ODS). Depending on the network structure, RND is specialized into two problems: Unidirectional RND and Multidirectional RND. Multi-flow mathematical models are proposed for both problems, i.e. URND and MRND. Moreover, a Biased Random Key Genetic Algorithm (BRKGA) and an Iterated Local Search (ILS) metaheuristics are developed for the mono-objective RND versions, and then adaptations of these methods are proposed for the URND and MRND bi-objectives. In addition, classical bi-objective methods such as Non-dominated Sorting Genetic Algorithm II and ϵ -constraint are also developed. Results indicate mono and bi-objective ILS overcome the other methods. The decision-making system ODS is connected to heterogeneous systems for collecting the geographical data and to get a set of disruptions from Troyes city. Moreover, several constraints and optimization criteria are taken into account in the dedicated algorithms in order to propose detouring solutions.

Keywords: combinatorial optimization - operations research - metaheuristics - software engineering – urban networks.

Thèse réalisée en partenariat entre :

