UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

"SEVERAL APPROACHES FOR THE TRAVELING SALESMAN PROBLEM"

TESIS

PRESENTADA POR

NANCY ARACELY ARELLANO ARRIAGA

PARA OBTENER EL GRADO DE
DOCTORADO EN INGENIERÍA DE SISTEMAS

FEBRERO 2019

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Doctorado en Ingeniería de Sistemas

Universidad de Málaga

Facultad de Ciencias Económicas y Empresariales

Doctorado en Economía y Empresa

Tesis doctoral en régimen de cotutela

Several approaches for the Traveling Salesman Problem

por

Nancy Aracely Arellano Arriaga

en opción al grado de

Doctor

Directores

Dra. Iris Abril Martínez Salazar	Dr. Julián Molina Luque
UANL, México	UMA, España

San Nicolás de los Garza, Nuevo León, 11 de febrero de 2019

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Doctorado en Ingeniería de Sistemas

Universidad de Málaga

Facultad de Ciencias Económicas y Empresariales

Doctorado en Economía y Empresa

Tesis doctoral en cotutela

Several approaches for the Traveling Salesman Problem

por

Nancy Aracely Arellano Arriaga

en opción al grado de

Doctor

Directores

Dra. Iris Abril Martínez Salazar — UANL, México

Dr. Julián Molina Luque — UMA, España

San Nicolás de los Garza, Nuevo León, 11 de febrero de 2019

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Doctorado en Ingeniería de Sistemas

Los miembros del Comité de Tesis recomendamos que la Tesis en régimen de cotutela «Several approaches for the Traveling Salesman Problem», realizada por el alumno Nancy Aracely Arellano Arriaga, con número de matrícula 1613608, sea aceptada para su defensa como opción al grado de Doctor en Ingeniería con especialidad en Ingeniería de Sistemas.

El Comité de Tesis

[illegible]

Dra. Iris Abril Martínez Salazar
Asesora UANL

[illegible]

Dr. Julián Molina Luque
Asesor UMA

[illegible]

Dra. Ada Margarita Álvarez Socarrás

Revisora

[illegible]

Dra. Satu Elisa Schaeffer

Revisora

[illegible]

Dra. Irma Delia García Calvillo

Revisora

Vo. Bo.

[illegible]

Dr. Simón Martínez Martínez

Subdirector de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, 11 de febrero de 2019

Para el mejor filósofo del mundo

Contents

Acknowledgements **xix**

Abstract **xxiii**

Resumen **xxvii**

Preface **xxx**

1 Introduction **1**

1.1 Problem description 2

1.2 State of the art 4

1.3 Hypothesis and justifications 5

1.4 Objectives 6

1.5 Scientific contribution 7

1.6 Relevance 7

1.7 Thesis structure 8

2 Theoretical framework **10**

2.1 Linear Optimization 12

2.1.1 General overview 13

2.1.2 Integer programming 15

2.2 Traveling Salesman Problem 17

2.3 m-Traveling Salesman Problem 24

2.4 Minimum Latency Problem 28

2.5 k-Minimum Latency Problem 37

2.6 Multi-objective optimization 42

2.7 Multi-objective distance-centered routing problems 47

2.7.1 Multi-objective Traveling Salesman Problem 47

2.7.2 Multi-objective m-Traveling Salesman Problem 48

2.8 Multi-objective latency-centered routing problems 48

2.8.1 Multi-objective Minimum Latency Problem 49

2.8.2 Multi-objective k-Minimum Latecy Problem 49

2.9 The simultaneous minimization of latency and distance in routing problems 50

3 Minimum Latency-Distance Problem **51**

3.1 Minimum Latency-Distance Problem: an introduction 52

3.2 MLDP: assumptions and mathematical formulation 54

3.3 Complexity of MLDP 61
3.4 Solution methods for MLDP 67
3.4.1 Exact method 68
3.4.2 Strategic Memetic Search Algorithm for MLDP 70
3.4.2.1 Initial population 72
3.4.2.2 Offsprings 72
3.4.2.3 Ranking 72
3.4.2.4 Neighborhood exploration 73
3.4.3 Evolutionary algorithm with Intelligent Local Search Algorithm for MLDP 74
3.4.3.1 Initial population 75
3.4.3.2 Elite set 76
3.4.3.3 Offsprings 76
3.4.3.4 Mutation 77
3.4.3.5 Neighborhood exploration 77
3.4.3.6 Neighborhood performance 80
3.4.4 Automatically adjusting evolutionary algorithm with Intelligent Local Search Algorithm for MLDP 83
3.4.4.1 Initial population 83
3.4.4.2 Elite set 83

3.4.4.3 Offsprings 85

3.4.4.4 Ranking and crowding scheme 85

3.4.4.5 Mutation 88

3.4.4.6 Neighborhood exploration 88

3.4.4.7 Neighborhood performance 90

3.4.4.8 Autocalibration of parameters 90

3.5 Experimentation 92

3.5.1 Trial instances 92

3.5.2 Measurements of the quality of the obtained fronts 93

3.5.3 Technical specifications 93

3.5.4 Parameter calibration 94

3.5.5 Computational experimentation 96

3.5.5.1 Small instances 96

3.5.5.2 Large instances 100

3.5.6 Conclusions 103

3.6 Conclusions and future research 103

4 k-Minimum Latency-Distance Problem 106

4.1 k-Minimum Latency-Distance Problem: an introduction 106

4.2 k-MLDP: assumptions and mathematical formulation 108

4.3 Complexity of k-MLDP 115
4.4 Solution methods for k-MLDP 120
4.4.1 Exact method 121
4.4.2 Strategic Memetic Search Algorithm for k-MLDP 122
4.4.2.1 Initial population 122
4.4.2.2 Offsprings 122
4.4.2.3 Ranking 123
4.4.2.4 Neighborhood exploration 123
4.4.3 Evolutionary algorithm with Intelligent Local Search Algorithm for k-MLDP 123
4.4.3.1 Initial population 123
4.4.3.2 Elite set 123
4.4.3.3 Offsprings 124
4.4.3.4 Mutation 124
4.4.3.5 Neighborhood exploration 124
4.4.3.6 Neighborhood performance 126
4.4.4 Automatically adjusting evolutionary algorithm with Intelligent Local Search Algorithm for k-MLDP 126
4.4.4.1 Initial population 127
4.4.4.2 Elite set 127

4.4.4.3 Offsprings 127

4.4.4.4 Ranking and crowding scheme 127

4.4.4.5 Mutation 127

4.4.4.6 Neighborhood exploration 128

4.4.4.7 Neighborhood performance 128

4.4.4.8 Autocalibration of parameters 128

4.5 Experimentation 128

4.5.1 Trial instances 128

4.5.2 Measurements of the quality of the obtained fronts 129

4.5.3 Technical specifications 129

4.5.4 Parameter calibration 129

4.5.5 Computational experimentation 133

4.5.5.1 Small instances 133

4.5.5.2 Large instances 139

4.5.6 Conclusions 144

4.6 Conclusions and future research 145

5 Conclusions **147**

5.1 Future research 148

List of Figures

2.1 Representation of a continuous polytope in a two-dimensional space. . 14

2.2 Representation of an integer polytope in a two-dimensional space. . . 16

2.3 Graphical example of a small TSP solution in an instance of ten vertices. 18

2.4 Synoptic table depicting several solution methods employed in solving combinatorial problems, among which routing problems are immersed. 20

2.5 Graphical example of an m-TSP instance. 24

2.6 The difference in the metrics to evaluate the objectives of the TSP and the MLP. To minimize the travel distance (see Figure 2.6a), only the sum of all travel times is considered. Figure 2.6b illustrates the contribution of every visited vertex in the final latency of the tour. . . 31

2.7 Graphical example of an small MLP solution in an instance of ten vertices. 32

2.8 Multi-level network used to formulate the MLDP. 33

2.9 Graphical example of an k-TRP instance. 37

2.10 Multi-level network used to formulate the k-TRP. 39

2.11 Basic definitions of Pareto optimality: two objective functions are being minimized; the Pareto front is drawn in red. 44

2.12 Ideal point (y^I) and nadir point (y^N) depicted in a two-objective space. Both objectives are being minimized. 45

3.1 Graphical example of an small MLDP solution in an instance of ten vertices. 55

3.2 Multi-level network used to formulate the MLDP. 57

3.3 Illustration of some particular instances of the MLDP, which were studied in the complexity proof. In Figure 3.3a, the general case is shown: in this case, service and travel times take all different arbitrary values. In Figure 3.3b, a simpler case is shown: all service times have constant values u, while all travel times are arbitrary. This particular case is a TSP instance. Lastly, in Figure 3.3c, another simpler case is shown: all service times are arbitrary and all travel times are equal to a constant u. This particular case is an MLP instance. 64

3.4 Reduction diagram from TSP to MLDP (both as decision problems). The main box in the figure depicts an algorithm to solve the TSP problem that takes as input an instance of TSP and determines whether the answer is “yes” or “no” to this instance. To prove the complexity of MLDP, a reduction algorithm must transform a TSP entry into a MLDP one. This MLDP input is then solved and receives a “yes” or “no” answer. As a whole, the process takes a TSP instance, reduces it into an MLDP instance, solves the reduced instance with an algorithm for MLDP, and responds correctly “yes” or “no” to the original instance. 66

4.1 Graphical example of an small k-MLDP solution in an instance of ten clients. 109

4.2 Multi-level network used to formulate the k-MLDP. 110

4.3 Illustration of some particular instances of the k-MLDP. In Figure 4.3a, the general case is shown: in this case, service and travel times all take arbitrary values. In Figure 4.3b, a simpler case is shown: all service times have a constant value u but travel times are arbitrary (a k-TSP instance). Lastly, in Figure 4.3c: all service times are arbitrary and all travel times are equal to a constant u (a k-MLP instance). . . 116

4.4 Reduction diagram from k-TSP to k-MLDP (both as decision problems). The main box in the figure depicts an algorithm to solve the k-TSP problem that takes as input an instance of k-TSP and determines whether the answer is "yes" or "no" to this instance. To prove the complexity of k-MLDP, a reduction algorithm must transform a k-TSP entry into a k-MLDP one. This k-MLDP input is then solved and receives a "yes" or "no". As a whole, the process takes a k-TSP instance, reduces it into a k-MLDP instance, solves the reduced instance with an algorithm for k-MLDP, and responds correctly "yes" or "no" to the original instance. 119

List of Tables

2.1 Pay-off table obtained from the ideal point. 46

3.1 EiLS offspring generation with $n = 10$ and $\varphi = 3$. The exchanged segments are highlighted in grey. 76

3.2 An example of $\mathcal{N}_2$ with $n = 10$ with $a = 1$, $b = 3$, and $c = 9$. 78

3.3 An example of $\mathcal{N}_3$ with $n = 10$ where $a = 3$, $\lfloor \xi \rfloor = 5$, and $b = 4$. . . . 78

3.4 An example of $\mathcal{N}_4$ with $n = 10$ where $a = 8$, $\lfloor \xi \rfloor = 4$, $b = 5$, and $c = 6$. 79

3.5 An example of $\mathcal{N}_8$ with $n = 10$, $a = 2$, and $b = 7$, showing all quantities involved in Equation (3.1), using integers for simplicity of the example. 80

3.6 An example of $\mathcal{N}_9$ with $n = 10$. 80

3.7 An example $\mathcal{N}_{10}$ with $n = 10$, $a = 7$, and $b = 3$. 82

3.8 An example $\mathcal{N}_{11}$ with $n = 10$ and $\lambda = 3$. 82

3.9 Metrics list used to measure the quality of the solutions found by our proposed algorithms. 93

3.10 Factorial design for parameter calibration for SMSA (left) and EiLS (right); n is the number of clients, # the number of points in the approximated front, and SCC the size of space covered. 95

3.11 Average CPU times obtained by our four proposed solution methods. These times are shown in averages per each set of instances and measured in seconds; n is the size of the instance sets solved. 97

3.12 Comparison among the metrics obtained per algorithm. These values are averages over the set of instances. 98

3.13 Average coverage of the approximated fronts obtained by the SMSA, EiLS, and AEiLS procedures. 99

3.14 Average CPU times obtained by our three approximation methods in larger instances. These times are shown in averages per each set of instances and measured in seconds. 100

3.15 Comparison among the metrics obtained per algorithm in larger instances. These values are averages over the set of instances. 101

3.16 Average coverage of the approximated fronts in larger instances. Averages obtained by the SMSA, EiLS and AEiLS. 102

4.1 Example of a single k-MLDP solution. This solution has 10 clients and $k = 3$ agents. The depot is denoted as v_0. 122

4.2 EiLS offspring generation with $n = 10$ and $k = 3$. The exchanged segments are highlighted in grey. 124

4.3 An example $\mathcal{N}_{13}$ with $n = 10$ and $k = 3$, where $a = 11$ and $b = 3$. . . 125

4.4 An example of $\mathcal{N}_{14}$ with $n = 10$ and $k = 3$, where $a = 1$, $b = 3$, and $c = 9$. 125

4.5 An example of $\mathcal{N}_{15}$ with $n = 10$ and $k = 3$, where $a = 12$, $b = 4$. . . . 126

4.6 An example of $\mathcal{N}_{16}$ with $n = 10$ and $k = 3$, where $a = 4$, $b = 12$. . . . 126

4.7 Factorial design for parameter calibration in SMSA; n is the size of the instance. For the SMSA algorithm we tested the size of the initial population (γ), the probability to mutate a solution (β), as well as number of iterations without success (I). 131

4.8 Factorial design for parameter calibration in EiLS; n is the size of the instance. For the EiLS algorithm we the size of the population (γ), the number of evaluations per module (ϕ), as well as the minimum success (α). 132

4.9 Average CPU times obtained by the four proposed solution methods. The times are averages over set of instances, measured in seconds. . . 134

4.10 Comparison among the metrics obtained per algorithm, averaged over the set of instances. 135

4.11 Comparison among the metrics obtained per algorithm, averaged over the set of instances. 136

4.12 Comparison among the metrics obtained per algorithm, averaged over the set of instances. 138

4.13 Average CPU times obtained by our four proposed solution methods in averages over each set of instances, measured in seconds. 140

4.14 Comparison among the metrics obtained per algorithm, in averages over each set of instances 141

4.15 Comparison among the metrics obtained per algorithm, in averages over each set of instances. 142

4.16 Comparison among the metrics obtained per algorithm, as averages over each set of instances 143

4.17 Comparison among the metrics obtained per algorithm, as averages over each set of instances 144

Acknowledgements

I would like to thank all those who made this dissertation possible. As my committee, friends, and family are Spanish speakers, I provide the acknowledgements in Spanish.

Esta investigación doctoral no sería posible sin el apoyo económico del Consejo Nacional de Ciencia y Tecnología (CONACyT). Agradezco el apoyo brindado bajo la beca número 446316, así como la confianza para brindarme una beca mixta por el período 2016–2017 para realizar una estancia doctoral en Málaga y poder acceder a la cotutela UMA-UANL. Agradezco igualmente a la Asociación Universitaria Iberoamericana de Postgrado (AUIP) por el apoyo económico brindado en el período 2017–2018 para finalizar mis estudios doctorales en la Universidad de Málaga. Conjuntamente ambos apoyos económicos han permitido que tenga una dedicación a tiempo completo para realizar este trabajo. Gracias a ambas instituciones.

Le agradezco a la Universidad Autónoma de Nuevo León (UANL), por haberme abierto nuevamente las puertas en esta casa de estudios. Gracias al Posgrado en Ingeniería de Sistemas (PISIS), a todos los investigadores pertenecientes al programa, y con especial atención le agradezco al grupo de investigación de la Dra. Iris Martínez por abrirme las puertas. Agradezco a la Universidad de Málaga (UMA), por haberme aceptado para unirme al programa de cotutela UMA-UANL. Agradezco a la Facultad de Ciencias Económicas y Empresariales por aceptarme, especialmente al departamento de Matemáticas.

Principalmente le agradezco a mi directora de tesis, la Dra. Iris Abril Martínez Salazar, por el apoyo y la atención brindada durante todos estos años de investigación conjunta. Gracias por confiar en mí y aceptar trabajar conmigo en el doctorado. Gracias por siempre hacerme sentir parte de tu grupo de trabajo, gracias por la comprensión siempre y por compartirme las divertidas anécdotas de sus bebés, sobretodo las de Victorín.

Gracias a mi co-director de tesis, el Dr. Julián Molina Luque por su cálido recibimiento en Málaga, por aceptar trabajar conmigo sin conocerme antes y romper desde el día uno esa barrera alumno-maestro que hay muchas veces en las universidades mexicanas. Gracias por el apoyo en todo, por escucharme cuando hay problemas y darme buenas ideas siempre. Ha sido un placer y un privilegio contar con su guía en la realización de esta tesis doctoral. Finalmente me gustaría añadir mis más sinceros agradecimientos a Jorge, a Julián y a María por su amabilidad y su atención.

Le agradezco al comité doctoral por sus acertadas correcciones y el tiempo invertido en la lectura de esta tesis. Comenzando con la Dra. Ada Margarita Álvarez Socarrás, a quien le agradezco por apoyarme de forma personal siempre. Por nunca dejar de ser mamá pollito conmigo y dejarme claro que aunque yo soy muy rara ella siempre está ahí. Gracias por terminar adoptándome más que académicamente hablando. Se le quiere mucho, a usted y al Dr Frank, aunque casi nunca se lo diga.

Gracias a la Dra. Satu Elisa Schaeffer, por la paciencia infinita que me tiene. Por enseñarme hasta como puntuar correctamente un texto y que la perfección académica nunca peca de exageración. Gracias por aceptar colaborar conmigo y con el resto del comité en las revisiones de esta tesis. Gracias por cada corrección y cada mejora. Definitivamente este texto no estaría tan bonito sin su ayuda. Gracias por tanto apoyo y por no dejar que mi negatividad propia pueda conmigo.

Finalmente, le agradezco a la Dra. Irma Delia García Calvillo por todo su apoyo, paciencia y buenos consejos. Es la que más años consecutivos me ha soportado y aún así siempre me recibe con una sonrisa y un abrazo sincero. Muchas gracias por haberme apoyado desde el inicio en mi curiosidad académica aún cuando nos costó que nos aceptaran trabajar juntas, pero sobre todo muchas gracias por seguir enseñándome cosas nuevas y apoyándome siempre.

De manera personal, le agradezco a mis amigos del posgrado por tantos buenos momentos. En especial al Dr. Juan Banda, por tanto. Al Dr. Luis Benavides, al MC. Daniel Mosquera y al MC. Eduardo García, así como al Dr. Fernando Elizalde, a la Dra. Cristina Maya, a la Dra. Nelly Hernández y al MC. Diego Hernández. Gracias por tantos buenos ratos y por cada plática amena.

Un afectuoso agradecimiento al departamento de matemáticas de la facultad de ciencias económicas y empresariales de la Universidad de Málaga. Muchas gracias a todos por el cálido recibimiento en ambas ocasiones que he estado ahí, muchas gracias por la amistad y por los buenos ratos. En especial le agradezco a la Dra. Laura Delgado, a la Dra. Ana Belén Ruiz, al Dr. Julian Molina y a la Dra. Beatriz Rodríguez, que me hicieron sentir siempre su amistad dentro y fuera de la rutina académica. Gracias también al Dr. Rafael Caballero, a la Dra. Trinidad Gomez, a la Dra. Analía Cano, a la Dra. Mercedes González, al Dr. Mariano Luque y al Dr. Ángel Torrico por hacerme sentir una más en el departamento durante toda mi estancia.

Gracias a la ME. Lilibeth Ruiz, a la Lic. Gisela Fuentes Zul y al Lic. Héctor Aguilar por seguir aquí, gracias por el apoyo, la amistad, el tiempo, las buenas vivencias y todo, ¡Los quiero mucho! Gracias a la MC. Myriam Herrera por la amistad y el apoyo. Gracias a la Dra. Laura Delgado, por enseñarme que las mejores amistades llegan de improviso: Gracias bonita por adoptarme, por hacerme compañía siempre y por escucharme y darme buenos consejos, ¡Te quiero mucho!

Le agradezco al MPhil. Fernando Berdún Palacios por enseñarme cosas nuevas todos los días, pero sobre todo le agradezco que apareciera y que me haga querer ser una mejor persona siempre. Gracias por tanto, mi corazón bonito. Y tú sigue pensando como loco, que no hay cosa más bonita que verte dando vueltas a detalles pequeñitos y curiosos que sólo tú notas. Gracias por tanta luz, por tanto y por todo. ¡Te quiero muchísimo, mi rubiecito guapo!

Finalmente, le agradezco a mis padres, María Arriaga y Elias Arellano, por todo. A mi preciosa hermana la Dra. Yolanda Arellano, por sacarnos a todos del hoyo y seguir cultivando su luz propia, ¡Te quiero mucho, Enis! Le agradezco a cada uno de ustedes su apoyo a las cosas raras que hago, las palabras de ánimo siempre que tengo mis momentos de duda y sobre todo, que aún con todo sigan ahí escuchándome con mis tonterías, ¡Los quiero mucho! Gracias también a mi Babu, a Neyma, Numy, Nina, Bruno, Pocky, Lili y Norman. Ya saben, por los pelitos en la ropa.

Finally, I would like to add an acknowledgement to the reader, thank you for your interest and your time in reading this.

Abstract

Nancy Aracely Arellano Arriaga.

Double Doctoral candidate to obtain the degrees of:

Doctor of Engineering with specialization in Systems Engineering, and

Doctor of Economics and Business

Universidad Autónoma de Nuevo León,
Facultad de Ingeniería Mecánica y Eléctrica,
in double doctoral program with
Universidad de Málaga,
Facultad de Ciencias Económicas y Empresariales.

Thesis Dissertation Title:

SEVERAL APPROACHES FOR THE TRAVELING SALESMAN PROBLEM

Number of pages: 189.

OBJECTIVES AND METHODOLOGY: This doctoral dissertation studies and characterizes of a combination of objectives with several logistic applications. This combination aims to pursue not only a company benefit but a benefit to the clients waiting to obtain a service or a product. In classic routing theory, an economic approach is widely studied: the minimization of traveled distance and cost spent to perform the visiting is an economic objective. This dissertation aims to the inclusion of the client in the decision-making process to bring out a certain level of satisfaction in the client set when performing an action.

We part from having a set of clients demanding a service to a certain company. Several assumptions are made: when visiting a client, an agent must leave from a known depot and come back to it at the end of the tour assigned to it. All travel times among the clients and the depot are known, as well as all service times on each client. This is to say, the agent knows how long it will take to reach a client and to perform the requested service in the client location. The company is interested in improving two characteristics: an economic objective as well as a service-quality objective by minimizing the total travel distance of the agent while also minimizing the total waiting time of the clients.

We study two main approaches: the first one is to fulfil the visits assuming there is a single uncapacitated vehicle, this is to say that such vehicle has infinite capacity to attend all clients. The second one is to fulfil the visits with a fleet of k uncapacitated vehicles, all of them restricted to an strict constraint of being active and having at least one client to visit. We denominate the single-vehicle approach the minimum latency-distance problem (MLDP), and the k-sized fleet the k-minimum latency-distance problem (k-MLDP).

As previously stated, this company has two options: to fulfil the visits with a single-vehicle or with a fixed-size fleet of k agents to perform the visits. We characterize both approaches, MLDP and k-MLDP, with several methodologies; both a linear and a non-linear mathematical formulation are proposed. Additionally, the design and implementation of an exact methodology to solve both linear formulations is implemented and with it we obtained exact results. Due to the large computation time these formulations take to be solved with the exact methodology proposed, we analyse the complexity each of these approaches and show that both problems are NP-hard.

As both problems are NP-hard, we propose three metaheuristic methods to obtain solutions in shorter computation time. Our solution methods are population based metaheuristics which exploit the structure of both problems and give good quality solutions by introducing novel local search procedures which are able to explore more efficiently their search space and to obtain good quality solutions in shorter computation time.

CONTRIBUTIONS AND CONCLUSIONS: Our main contribution is the study and characterization of a bi-objective problematic involving the minimization of two objectives: an economic one which aims to minimize the total travel distance, and a service-quality objective which aims to minimize of the waiting time of the clients to be visited. With this combination of objectives, we aim to characterize the inclusion of the client in the decision-making process to introduce service-quality decisions alongside a classic routing objective.

To the best of our knowledge, and excluding our own published research, a similar combination of objectives has not been studied and therefore the whole study in this dissertation is a direct contribution. We propose a set of instances inspired by those of Angel-Bello et al. (2013), with adjustments to create client locations and service times more similar to real-world settings. We propose a linear and a non-linear mathematical formulation for each problem, MLDP and k-MLDP, as well as three metaheuristic methodologies for each. The implementation, design, and calibration of each of these methods are considered a contribution of this dissertation as well.

Advisors:

Dra. Iris Abril Martínez Salazar

UANL (Mexico)

Dr. Julián Molina Luque

UMA (Spain)

Resumen

Nancy Aracely Arellano Arriaga.

Candidato para la obtención del doble grado en:

Doctor en Ingeniería con especialidad en Ingeniería de Sistemas, y

Doctor en Economía y Empresa

Universidad Autónoma de Nuevo León,
Facultad de Ingeniería Mecánica y Eléctrica,
en cotutela con
Universidad de Málaga,
Facultad de Ciencias Económicas y Empresariales.

Título del estudio:

SEVERAL APPROACHES FOR THE TRAVELING SALESMAN PROBLEM

Número de páginas: 189.

OBJETIVOS Y MÉTODO DE ESTUDIO: El objetivo de este trabajo es el planteamiento y estudio de dos problemas bi-objetivo los cuales se centran en el estudio de una combinación de objetivos que buscan tanto un beneficio para la empresa como un beneficio para los clientes que dicha empresa atiende. Esta tesis doctoral tiene como objetivo el estudio de la inclusión del cliente en el proceso de toma de decisiones, y así añadir un objetivo de calidad en el servicio a un problema clásico de ruteo. Se estudia el problema que tiene una compañía, la cual provee un servicio a un conjunto de clientes y la cual está obligada a asignar todos los clientes a uno o varios agentes para visitarles. Estos agentes deben partir de y volver a un depósito previamente establecido. Se conocen todos los tiempos de viaje entre clientes y los tiempos de servicio que los agentes se demoran en cada cliente, es decir, cada agente conoce con exactitud cuanto tiempo tardará en atender a cada cliente y en llegar a él. En este problema se busca simultáneamente optimizar dos características: minimizar la distancia total recorrida por el o los vehículos, así como minimizar el tiempo de espera de los clientes.

Para lograr esto, proponemos dos problemas: *the minimum latency-distance problem* (MLDP), en el cual un vehículo de capacidad infinita visita a todos los clientes, y *k-minimum latency-distance problem* (k-MLDP), en el cual se cuenta con una flotilla de k vehículos para visitar a todos los clientes con la restricción de que todos los vehículos deben estar activos y tener al menos a un cliente asignado. Para ambos problemas, se propone un modelo lineal y un modelo no lineal, así como el diseño e implementación de una metodología exacta para ambas formulaciones lineales. Debido al excesivo tiempo de cómputo, se analiza la complejidad de ambos problemas propuestos y se comprueba que ambos son NP-duros. Se proponen tres métodos metaheurísticos basados en técnicas evolutivas, los cuales son capaces de brindar soluciones de buena calidad en tiempos de cómputo pequeños. Estos tres métodos exploran la estructura de cada uno de los problemas e introducen técnicas de búsqueda local novedosas, las cuales son capaces de brindar efectivamente soluciones de buena calidad en tiempos de cómputo pequeños.

CONTRIBUCIONES Y CONCLUSIONES: La contribución principal de este trabajo se centra en el estudio de una problemática bi-objetivo que optimiza simultáneamente dos aspectos importantes para una compañía centrada en el servicio o distribución: la minimización de la distancia recorrida por el o los vehículos al hacer la visita a los clientes y optimizar así un objetivo económico, así como minimización del tiempo total de espera de los clientes que esperan por el servicio que se les brindará y así, consolidar los beneficios que tiene la inclusión del cliente en la toma de decisiones de una empresa centrada en actividades logísticas y de servicio.

La revisión de literatura mostró que no existen trabajos similares, excluyendo las publicaciones y presentaciones obtenidas con nuestra investigación, por lo tanto el estudio presentado en este trabajo es una contribución directamente. Se propone además un conjunto de instancias inspiradas en las ya reportadas previamente por Angel-Bello et al. (2013) pero con características más apegadas a la realidad. Se proponen una formulación lineal y una formulación no lineal para cada uno de los problemas estudiados, MLDP y k-MLDP, así como el diseño e implementación de tres metaheurísticas novedosas las cuales también son consideradas una contribución directa de este trabajo.

Firmas de los asesores:

Dra. Iris Abril Martínez Salazar	Dr. Julián Molina Luque
UANL (México)	UMA (España)

Preface

This doctoral dissertation is submitted to obtain a double doctoral degree in the joint doctorate agreement between the Autonomous University of Nuevo León and the University of Málaga. The research described herein was conducted under the supervision and guidance of Professor Iris Martínez Salazar and Professor Julián Molina, in the Department of Mechanical and Electrical Engineering at Autonomous University of Nuevo León and the Department of Economy (Mathematics) at University of Málaga respectively.

This is, to the best of our knowledge, all original work and particularly it focusses in the study of a routing problem with a single-vehicle or a k-sized fleet. This is achieved by studying a trade-off among an economic and a service-quality decision: to minimize the distance traveled by the vehicles to fulfil the visits and simultaneously to minimize the total waiting time of the clients. This research is divided into two major parts:

- the Minimum Latency-Distance Problem (MLDP): studies the case when this company only has a single-vehicle tour with infinite capacity to satisfy the demand of the clients, and
- the k-Minimum Latency-Distance Problem (k-MLDP): studies a generalization of MLDP, this is to say, a more realistic approach which explores the employ-

ment of a fleet of k vehicles to satisfy the demand of the clients. Each of these vehicles must be active by having at least one client assigned to each one of the vehicles and all tours must be disjoint.

Each of these two problems is studied and reported in an individual chapter which contains all the information needed before hand, the research done and the reported contributions we offer. Each chapter contains a literature review, the original research, as well as conclusions and future work insights.

The scientific production generated as a result of this dissertation includes:

- Technical report:
  1. N. A. Arellano-Arriaga, J. Molina, S. E. Schaeffer, A. M. Álvarez-Socarrás, and I. A. Martínez-Salazar. Complexity of MLDP. Technical Report arXiv:1809.02656 [cs.CC], arXiv.org, Sept. 2018.
- Publication:
  1. N. A. Arellano-Arriaga, J. Molina, S. E. Schaeffer, A. M. Álvarez-Socarrás, and I. A. Martínez-Salazar. A bi-objective study of the minimum latency problem. *Journal of Heuristics*, 2019. Accepted for publication.
  2. N. A. Arellano-Arriaga, A. M. Álvarez-Socarrás, and I. A. Martínez-Salazar. A sustainable bi-objective approach for the minimum latency problem. In E. Alba, F. Chicano, and G. Luque, editors, *Smart Cities*, volume 10268 of *Lecture Notes in Computer Science*, pages 11–19, Cham, Switzerland, 2017. Springer. doi: 10.1007/978-3-319-59513-9_2.
- International conferences.

1. Aug 2017. Arellano-Arriaga, N.A., Martinez-Salazar, I.A., A bi-objective routing problem: Two variants. First International Workshop on Artificial Intelligence and Optimization (WAIO). Monterrey, Mexico.
2. June 2017. Arellano-Arriaga, N.A., Alvarez-Socarras, A.M., Martinez-Salazar, I.A., A sustainable bi-objective approach for the minimum latency problem. International Conference on Smart Cities. Malaga, Spain.
3. Oct 2014. Arellano-Arriaga, N.A., Alvarez-Socarras, A.M., Un enfoque biobjetivo para el problema del reparador. XVII Latin-Iberian American Conference on Operations Research (CLAIO). Monterrey, Mexico.

- National conferences.

1. Sept 2016. Arellano-Arriaga, N.A., Martinez-Salazar, I.A., Sobre la complejidad del k-MLDP. 2do Encuentro Nacional de Matemáticas Aplicadas. Arteaga, Coahuila, México.
2. Jun 2016. Arellano-Arriaga, N.A., Martinez-Salazar, I.A., k-MLDP: complejidad y modelación. IV Jornada de Optimización. San Nicolás de los Garza, NL, México.
3. May 2015. Arellano-Arriaga, N.A., Martinez-Salazar, I.A., Un enfoque bi-objtivo al problema de múltiples reparadores. III Jornada de Optimización. San Nicolás de los Garza, NL, México.
4. Dec 2014. Arellano-Arriaga, N.A., Alvarez-Socarras, A.M., Martinez-Salazar, I.A., Estudio explorativo sobre la complejidad computacional de un enfoque bi-objetivo al problema del reparador. II Jornada de Optimización. San Nicolás de los Garza, NL, México.

This work was funded by the following grants:

- CONACyT, the Mexican National Council for Science and Technology, under grant 446316.
- AUIP, the Asociación Universitaria Iberoamericana de Postgrado, for the scholarship granted for the academic year 2017–2018.

Chapter 1

Introduction

Businesses are economic systems in which goods and services are exchanged in order to earn a profit in a regular basis. These goods can be exchanged for money or for other goods. For these systems to be profitable, they require an investment and enough customers to be reliable. These customers are colloquially referred to as *clients*.

In this doctoral dissertation, we study an hypothetical case in which a certain company in charge of visiting a set of clients to provide them a service is interested in improving their routing assignation by considering not only the economic aspects of the business to improve profitability but simultaneously, to improve the quality of the service given to improve the satisfaction of the clients waiting to have their demand fulfilled. We divide this study in two major branches: when the company has a single-agent to fulfill the visiting, as well as when the company has k agents for doing so.

The following sections in this chapter introduce the general description of both of the problems this dissertation focuses on, as well as the motivation and objectives of this work.

1.1 Problem description

Routing theory refers to the determination of an optimal set of cycles, or *tours*, within a network to establish a way to reach one point from another under certain circumstances. This area of study has many applications: the delivery of goods from a depot to a set of clients or the application of routing to determine the optical sequence in an electrical drilling device, etcetera.

In particular, this dissertation studies a problem which arises in the logistic processes and activities of a certain distribution company in charge of providing a set of clients with a service or a delivery. This company is interested in improving the assignation of their clients into tours but considering not only the economic aspects of the business to improve profitability, but simultaneously, to improve the quality of the service given to improve the satisfaction of the clients waiting to have their demand fulfilled.

To achieve this, the company aims to simultaneously deal with two conflictive objectives: to visit all the clients by traveling the minimum distance to ensure that the cost this routing represents for the company is minimum, as well as the maximization of the client satisfaction, through the minimization of the latency of the routing, to ensure a competitive quality in the service provided to the clients. We propose to involve the client to be considered in the decision-making process, to turn a classic routing problem into a client-centered one.

We study this bi-objective proposal in two major branches: when the company has only a single vehicle available to fulfill the visiting. Such vehicle has infinite capacity, this is to say it can visit every client without exceeding its capacity. In this single-vehicle approach we assume the depot's location is known, as well as the exact locations of all the clients. We therefore know with certainty the time it takes the agent to travel from every client's location to the depot and vice versa as well as all times it takes to travel from one client to another. The main goal is to design the most suitable way of visiting all clients within this set, departing from the depot and returning to it at the end of the workday, in such a way that the total travel distance as well as the total waiting time of the clients in the tour are minimized. We call this problem *the Minimum Latency-Distance Problem* (MLDP).

The second case is when the company has a fleet of k vehicles to fulfill the visits to the clients. All these k vehicles have infinite capacity, this is to say that every one of them can visit as many clients as are assigned to them without exceeding their capacity. In this fleet-vehicle approach, we have to satisfy certain conditions: all k vehicles must be active and have at least one client to visit. Every client must be visited once, by only one agent; this gives a disjoint set of k tours to fulfill the demands of the clients. Some assumptions made are the beforehand knowledge of the locations of each client and the depot, meaning we know with certainty how long it will take for every agent to travel to and service each client. The main goal is to design a disjoint set of k tours for visiting all clients, each of them departing from the depot and returning to it at the end of the workday, in such a way that the total travel distance as well as the total waiting time of the clients in the tour are minimized. We call this problem the *k-Minimum Latency-Distance Problem* (k-MLDP).

By studying this bi-objective proposal, regardless of the number of agents to make the visits, we aim to combine an economic and a service-quality objective to integrate in the decision-making process both parts related: the company responsible for distributing its product or services and the clients to be attended by the agents, by minimizing their total waiting time to be served.

1.2 State of the art

MLDP as well as k-MLDP minimize two objectives: distance and latency of one or a set of k disjoint tours, respectively. Our single-vehicle approach, MLDP, is the combination of the objectives optimized by the *Traveling Salesman Problem* (TSP) and of the *Minimum Latency Problem* (MLP). Applications for both raise from classic routing for TSP (Applegate et al., 2007; Gutin and Punnen, 2006; Mjirda et al., 2017), all the way to humanitarian logistics for MLP (Blum et al., 1994; Ferrer et al., 2016; García et al., 2002; Kovács and Spens, 2007, 2009; Lucena, 1990; Stephenson, 2017; Tomasini et al., 2009; Vargas et al., 2017). Both of these single-objective problems are NP-hard separately (Afrati et al., 1986; Papadimitriou, 1994) and hence their combination is expected to be NP-hard as well.

The fleet-sized approach, k-MLDP, is the combination of the objectives optimized by the k-Traveling Salesman Problem (k-TSP) and the k-Traveling Repairman Problem (k-TRP). Applications for both raise from routing a set of agents to attend a set of clients (Cordeau et al., 2007; Dantzig and Ramser, 1959; Gansterer and Hartl, 2018; Khodabandeh et al., 2017; Laporte, 1992; Modares et al., 1999; Toth and Vigo, 2014) for k-TSP, to the minimization of the arrival of a set of agents to certain destinations (Bektas, 2006; Fakcharoenphol et al., 2003; Onder et al., 2017) for k-TRP. These single-objective problems have been proven NP-hard as well (Ausiello et al., 2000; Fakcharoenphol et al., 2003), and therefore a combination of their objectives is expected to be NP-hard, too.

The bi-objective approach we propose combines two objectives which have not been simultaneously studied in a routing context. Nonetheless, considering a single-vehicle tour, they have been researched in contexts where the goal is the measurement of the latent time an specific growth takes, depending on the distance such growth it is measured from as in medicine (Bair et al., 2003; Fischer et al., 1987; Salami et al., 2003) or network applications (Borah et al., 2017; Depuy et al., 2001; He et al., 2015; Kim and Na, 2017; Madhyastha et al., 2006; Patterson, 2004; Sarddar et al., 2010, 2011). These applications are seen similarly when considering a fleet of vehicles, as in (Frye, 1995; Kara et al., 2007; Kim et al., 2012; Malarky, 2013; Mangharam et al., 2007).

To the best of our knowledge, excluding our own research, this combination of objectives has not been applied under routing conditions nor considering the client set in the decision-making for deciding an optimal routing. This dissertation aims to study the characterization of such trade-off to simultaneously define a more client-centered routing approach to offer to companies aiming to improve their profits and the satisfaction of their clients.

1.3 Hypothesis and justifications

A realistic approach for real-life problems involves the simultaneous optimization of several conflicting objectives. This dissertation aims to preserve the profit of a company in charge of providing a service or a delivery to a set of clients, as well as to improve the quality of the service given by such company by minimizing the total waiting time of the clients waiting to be served. This is a relevant problematic in contexts where both client service and company profit are priorities. We assign the same importance to both: distance-dependent aspects such as fuel cost and to quality-of-service aspects such as the waiting time of the clients to be attended.

As mentioned before, to the best of our knowledge and excluding our own research, a bi-objective approach with these characteristics has not been studied in a routing context previously. This bi-objective approach aims to the improvement of several aspects in classic routing approaches such as the competitive position of the company as well as the client satisfaction by considering a client-centered approach.

This dissertation aims to simultaneously optimize two objective functions previously proven NP-hard and presents a formal proof and justification to the employment of metaheuristic methods to solve both problematics, MLDP and k-MLDP, due to their complexity. This research proposes mathematical formulations to characterize them, along with the complexity study and the proposal of several techniques to obtain the Pareto fronts of both.

1.4 Objectives

This doctoral dissertation has the following objectives:

- Study of a bi-objective problem which, excluding our own research, does not exist in current literature. This problem combines an economic and a service-quality objective to formulate a more client-centered routing approach and is studied in two different problems: a single-vehicle problem, which we call the Minimum Latency-Distance problem (MLDP), and a k-vehicle problem, which we call the k-Minimum Latency-Distance problem (k-MLDP).
- Mathematical formulation for, MLDP and k-MLDP, alongside with the design, implementation and calibration of an exact solution method.
- Study of the computational complexity of both problems, also under particular trivial instances.
- Design, implementation, calibration, and testing of several metaheuristic solution methods for, MLDP and k-MLDP.

1.5 Scientific contribution

The main contributions of this dissertation are the following:

- Study of a combination of objectives leading to a more client-centered routing approach to offer to companies interested in simultaneously improving their profits and the satisfaction of their clients.
- Linear and non-linear mathematical formulations for this combination of objectives, considering one or several vehicles.
- Study of the computational complexity these bi-objective problems have, including particular trivial instances.
- Design of novel local search procedures to employ in the design of metaheuristic methods to solve MLDP and k-MLDP.

1.6 Relevance

We propose a bi-objective problem that considers two routing objectives, an economic objective and a service-quality objective, in order to preserve the profit of a company in charge of providing a service or a delivery to a set of clients, as well as to improve the quality of the service given by such company by minimizing the total waiting time of the clients waiting to be served.

This is a relevant combination of objectives in contexts where client service and company profit are priorities; we assign the same importance to both distance-dependent aspects such as fuel cost and to quality-of-service aspects such as the waiting time of the clients to be attended. This bi-objective approach aims to the improvement of several aspects in classical routing approaches improving the competitive position of the company as well as the client satisfaction by creating a client-centered routing approach.

1.7 Thesis structure

As previously stated, we study and characterize a bi-objective problem involving latency and distance under two main scenarios: a single-vehicle approach and a k-vehicle approach. In this dissertation, the research is divided similarly, according to the theoretical framework to the research. With this in mind, the structure of this dissertation is as follows.

Chapter 2 describes the basic concepts for the study of our proposed bi-objective problems, beginning with the description of basic Linear Optimization theory for single-objective problems moving to a short introduction of Integer Programming in single-objective theory. Due to the fact that our proposed problems, MLDP and k-MLDP, combine four important problems of routing theory, this chapter progresses to a brief introduction of all the four single-objective problems in Sections 2.2 – 2.5. Each of these sections undertakes the following structure: the considered single-objective problem is defined, several applications are discussed and a classical mathematical formulation is described. Lastly, a short summary of solution methods reported in existing literature is provided. As our main objective is the study of two bi-objective problems, Section 2.6 gives a brief introduction to multi-objective optimization. In Sections 2.7 and 2.8 we give a summarized state-of-the-art of multi-objective approaches that involve any of the four previous key problems, the combinations of which yield MLDP and k-MLDP. Lastly, Section 2.9 gives a brief introduction to the specific problems we aim to study in this thesis.

Chapter 3 is dedicated to the Minimum Latency-Distance Problem (MLDP). In Section 3.1, a description of MLDP is given. Assumptions for MLDP and two mathematical formulations proposed for it are reported in Section 3.2. Section 3.3 details the analysis of the computational complexity of MLDP and presents a proof that MLDP is an NP-hard problem. The exact methodology to solve MLDP is detailed in Section 3.4.1. We propose several heuristic methodologies to solve it in Sections 3.4.2, 3.4.3, and 3.4.4. The calibration of parameters, the description of a set of instances we propose to recreate real-life conditions, as well as the experimentation and comparison of results of the proposed methods are shown in Section 3.5. Conclusions and future lines of research for MLDP are given in Section 3.6.

Similarly, Chapter 4 is dedicated to the k-Minimum Latency-Distance Problem (k-MLDP). Section 4.1 gives a description of k-MLDP. Assumptions considered and two mathematical formulations for k-MLDP are reported in Section 4.2. Section 4.3 details the analysis of the computational complexity of k-MLDP and presents a proof that k-MLDP is an NP-hard problem. The exact methodology to solve k-MLDP is detailed in Section 4.4.1. Similarly, we propose three heuristics to solve it in Sections 4.4.2, 4.4.3, and 4.4.4. Section 4.5 includes the calibration of parameters, the comparison among the methods and the experimentation results. Lastly, conclusions and future lines of research for k-MLDP are given in Section 4.6.

Finally, Chapter 5 provides a discussion to summarize the conclusions obtained in exploring this combination of objectives, regardless of the number of vehicles used, and proposes future lines of research.

Chapter 2

Theoretical framework

Distribution and delivery companies face the problem of finding efficient tours to visit their clients in such a way that economic costs are at a minimum. This is the idea on which vehicle routing is based: to fulfill the demand of a set of clients while keeping the cost of visiting every one of them as low as possible. This objective is known as the minimization of the travel *distance* of the tour and it is the main objective of the *Traveling Salesman Problem*. With one vehicle available for the visits (Applegate et al., 2007; Gutin and Punnen, 2006; Mjirda et al., 2017), as well as with a fleet of several vehicles available for this purpose (Khodabandeh et al., 2017; Laporte, 1992; Modares et al., 1999), it is one of the most studied combinatorial problems in literature, aiming to guarantee a company-friendly solution (Applegate et al., 2007; Cordeau et al., 2007; Dantzig and Ramser, 1959; Gansterer and Hartl, 2018; Gutin and Punnen, 2006; Khodabandeh et al., 2017; Laporte, 1992; Mjirda et al., 2017; Modares et al., 1999; Toth and Vigo, 2014).

In recent years, humanitarian logistic has studied a different approach for vehicle routing by *not* taking into account the economic cost of the tour visiting a set of locations but instead by considering the waiting time spent at every one of the locations to be visited (Kovács and Spens, 2007, 2009; Ortuño et al., 2013; Stephenson, 2017; Thomas and Mizushima, 2005; Tomasini et al., 2009). The minimization of the waiting time is an objective known as *latency*, and to visit a set of clients to fulfill their demand with the goal of keeping the client waiting time as short as possible is the main objective of the *Minimum Latency Problem* (Blum et al., 1994; Chaudhuri et al., 2003; García et al., 2002; Lucena, 1990; Nagarajan and Ravi, 2008). With one vehicle (Agnihothri, 1988; García et al., 2002; Van Ee and Sitters, 2018) or with several vehicles in a fleet (Bektas, 2006; Fakcharoenphol et al., 2003; Onder et al., 2017), latency is a client-friendly objective employed to guarantee quality of the service given by the company.

Vehicle routing as such deals with a single-objective decision: to visit a set of clients at the minimum economic cost (Applegate et al., 2007; Dantzig and Ramser, 1959; Toth and Vigo, 2014). To minimize the latency of a tour is to define a tour as a single-objective task: to visit a set of clients and ensure the minimum waiting time for them to get their service (Blum et al., 1994; Lucena, 1990; Nagarajan and Ravi, 2008). Nonetheless, real-life situations bring the need to take more than one objective into consideration. The involvement of several objectives leads to the employment of multi-objective optimization to make better decisions and to consider more adequate trade-offs (Ehrgott, 2005).

This dissertation studies a bi-objective problem which arises in the logistic processes and activities of distribution companies in charge of providing a set of clients with a service or a delivery. Such companies deal with two conflicting objectives: to visit all the clients by traveling the minimum distance (to ensure the cost this tour represents for the company is the minimum) and the maximization of the clients satisfaction (through the minimization of the latency of the tour, to ensure a competitive quality in the service provided to the clients). We study two main branches: a single-vehicle approach and a multi-vehicle approach.

In this chapter, we present a summary of fundamental concepts that uphold this dissertation. A general overview of linear optimization is presented in Section 2.1. As stated before, this dissertation focuses on the study of the combination of two different objectives: the minimization of the traveled distance of the tour and the minimization of the waiting time of the clients. Sections 2.2 and 2.4 show the state of the art over each single-objective and single-vehicle problem to help us build the basis for the study of our bi-objective and single-vehicle proposal; the basis for the study of our bi-objective and multi-vehicle approach along with the state of the art is shown in Sections 2.3 and 2.5, where each single-objective and multi-vehicle problem is studied.

We present a brief introduction to multi-objective optimization in Section 2.6. Similarly to the single-objective versions we discuss the study of both related objectives, latency and distance, but combined with other objectives previously reported in literature. Sections 2.7.1 and 2.8.1 provide a general overview of the state of the art of several bi-objective problems in a single-vehicle modality for both of the objectives considered in this dissertation. Sections 2.7.2 and 2.8.2 present a general overview of existing literature regarding bi-objective approaches for a multi-vehicle approach, for both of the objectives considered in this research. Lastly, Section 2.9 gives a short introduction to the main research this dissertation focuses on.

2.1 Linear Optimization

Operations Research is the application of mathematical methodologies to the study, analysis, and resolution of problems that involve complex systems. This area of study is applied to optimize business operations and other day-to-day problems as routing, backpacking, and scheduling, among others. It helps in company decision-making such as resource allocation, production scheduling to improve the development of production schedules, supply chain management, as well as the organization and the distribution of products (Ahuja et al., 1993; Dantzig, 1963; Frederick and Lieberman, 1982; Taha, 1992).

Linear Optimization, also known as *Linear Programming* (LP), is an operations research technique used to depict complex relationships through linear functions to find their optimal solution. It can be understood as a mathematical technique to state general goals in order to find the best solution to achieve an improvement in a particular problem considering a certain set of constraints and resources (Dantzig, 1963). Real day-to-day systems are often much more complex, but when simplified to linear relations. This method achieves the best outcome, either maximizing or minimizing, a linear system of equations. As the goal is to identify the best possible course of action, this area of study attempts to resolve the conflicts of interest among the components of the organization in such a way that the solution is the best for the organization as a whole, it provides the company with an understandable conclusion of a problem and gives tools to the decision maker to help him decide a course of action.

2.1.1 General overview

Linear Optimization searches for the best solution, referred as an *optimal solution*, for the problem or system of constraints under consideration. Linear Programming is a technique to optimize a linear objective function, subject to linear equality and linear inequality constraints that represents a *convex polytope* (also known as *convex cover*) which is defined as the intersections of the spaces in which each linear inequality is defined. The objective function to optimize is a linear function defined on this polytope and if there exist a point at which this function has the optimal value, which could be the smallest or the largest depending on the optimizing criterion, then such a point is called an *optimum*.

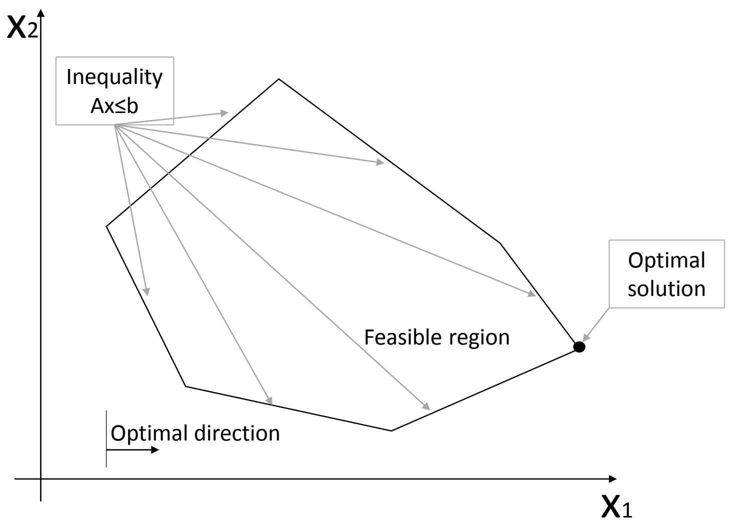


Figure 2.1: Representation of a continuous polytope in a two-dimensional space.

Linear systems are expressed in canonical form as shown in Equation (2.1),

$$\max Z = Cx \tag{2.1}$$

$$\text{s.t. } Ax \leq b,$$

$$x \geq 0,$$

where x is a vector representing the decision variables of the system of equations, this is to say, x represents the values to be determined: C and b are vectors of known coefficients which denote, respectively, the costs and the resources that cannot be exceeded in the solution; A is a matrix of coefficients of the constraints that represents the available resources. The set of constraints define a polyhedron, convex or concave, that represents the feasible region of the problem. Such polyhedron can be *infeasible* if two or more constraints of the constraint set cannot be satisfied jointly and an optimal solution does not exist. In Linear Programming, if feasible, such polyhedron is continuous and the optimal solution is found in a corner of it (Taha, 1992). A graphical representation of such a polytope is shown in Figure 2.1. Notice that the inequalities in the constraint set define the polytope and limit the feasible region of the problem. This example is of a feasible region in a two-dimensional space.

2.1.2 Integer programming

An special case of Linear Programming problems is defined when all decision variables, or some decision variables, of the problem are integers. A *integer problem* keeps the same qualities of a continuous linear problem and only differs by adding the restriction that the solution must be integer in at least one variable. By adding constraints of integer variables, the feasible region is discretized and the convex cover is formed by points instead of the full polytope of a continuous case. A graphical example of such a polytope is shown in Figure 2.2. Note that, similarly to the continuous case, each inequality in the constraint set limits the polytope, but only the points define the feasible region. The example is of a feasible region in a two-dimensional space.

Formally, the canonical form of an integer problem is defined in Equation (2.2),

$$\max Z = Cx \tag{2.2}$$

$$\begin{aligned} \text{s.t. } Ax &\leq b, \\ x &\geq 0, \\ x &\in \mathbb{Z}^n, \end{aligned} \tag{2.3}$$

where x is a vector representing the decision variables among which there are one or more integer variables, as stated in Equation (2.3). The vector C and b represent, respectively, all costs and resources that cannot be exceeded, defined as the known coefficients in the problem. The matrix A of coefficients of the constraints represents the available resources.

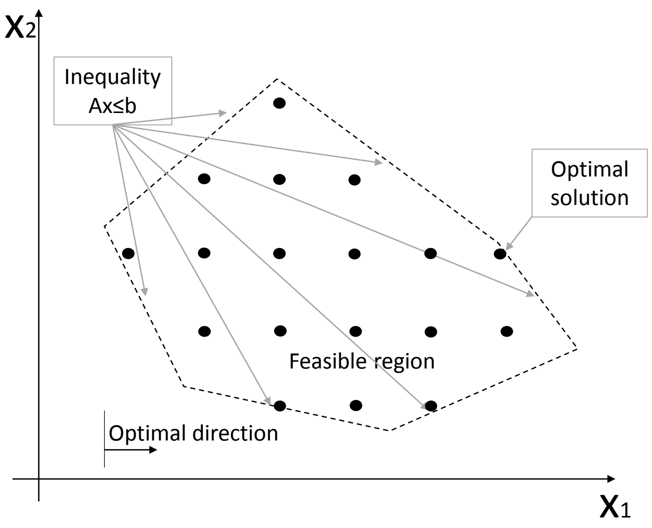


Figure 2.2: Representation of an integer polytope in a two-dimensional space.

In Operational Research, there exist many combinatorial integer problems. In general, integer problems are NP-complete (Garey and Johnson, 1990; Papadimitriou, 1994). Some examples of combinatorial integer problems include the following:

- *Knapsack problems* that aim to the maximization of the total value obtained by a selection of items with certain weight and value to be carried in a knapsack without exceeding the limit weight. This is an integer problem because the items must be included whole, not in halves or parts.
- *Assignment problem* that aims to match an equal amount of agents and tasks, where the assignment of a specific task to an specific agent has a known cost. Each agent can only have one task assigned and each task only can be assigned to one agent. The objective is to minimize the cost of the assignment.
- *Routing problems* in which a set of one or more agents are in charge of providing a service or a product to a set of clients. Each client must be visited only once.

Distances and travel times are known beforehand. The objective is to minimize the total travel distance of the tours.

All these problems have solutions that are integer and therefore are considered linear integer problems. This dissertation studies a bi-objective routing problem to simultaneously minimize two conflictive objectives: distance and latency. In particular, we focus in the characterization of this combination of objectives under two different scenarios: a single agent and a fleet of agents to perform the visiting. In this research, we propose several solution methodologies for such a combination of objectives. Sections 2.2, 2.4, 2.3, and 2.5 provide the basis of the routing theory involved in the formal characterization of our research.

2.2 Traveling Salesman Problem

As previously mentioned, in the introduction part of this chapter, we study a bi-objective problem that simultaneously optimizes two conflictive objectives: total travel distance and total waiting time (or latency) in two different scenarios: in a single tour or in a set of tours of a fleet of vehicles.

This section as well as Section 2.3 briefly discuss the distance objective and the two key distance-related problems we employ throughout this dissertation. The minimization of the total travel distance is an economic decision taken by a company eager to keep the cost of visiting the clients as low as possible. In a routing context, this describes the objective of the well-studied Traveling Salesman Problem.

The Traveling Salesman Problem (TSP) is the principal problem in vehicle-routing theory (Bellmore and Nemhauser, 1968; Dantzig et al., 1954; Dantzig and Ramser, 1959; Flood, 1956; Kruskal, 1956). This problem aims to the minimization of the total travel distance of a single-vehicle, or single-agent, tour. Such an agent departs from a known depot to visit a set of clients, the exact locations and all inter-travel times known beforehand, restricted to visit each of the clients exactly once and to return to the depot at the end of the tour. Figure 2.3 shows a graphical representation of a single-vehicle tour to visit a set of ten clients. Each grey arrow depicts a possible connection among clients and the depot whereas the darker lines denote the tour followed by the agent.

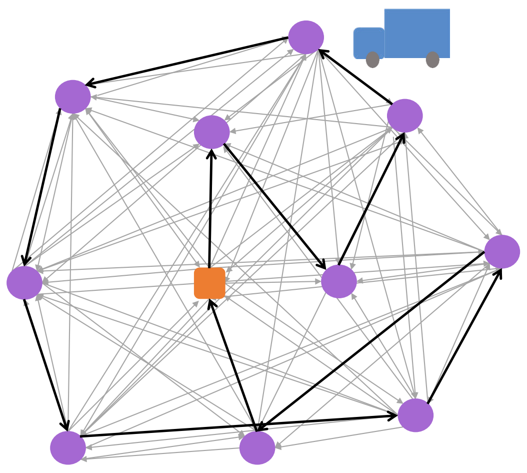


Figure 2.3: Graphical example of a small TSP solution in an instance of ten vertices.

TSP is a combinatorial widely studied problem (Applegate et al., 2007; Cordeau et al., 2007; Dantzig and Ramser, 1959; Gansterer and Hartl, 2018; Gutin and Punnen, 2006; Hoffman et al., 2013; Khodabandeh et al., 2017; Laporte, 1992, 2006; Mjirda et al., 2017; Modares et al., 1999; Toth and Vigo, 2014), mostly due to the simplicity with which it is stated and the complexity it has (Papadimitriou, 1994). TSP is used for a wide range of applications, some examples are the following:

- Delivery: the most intuitive application of the TSP, from school-bus routing (Bowerman et al., 1995; Laporte et al., 1995; Newton and Thomas, 1969; Park and Kim, 2010), geographic information system (Curtin et al., 2014; Jung et al., 2006), drone assistance (Murray and Chu, 2015), to GPS improvements

(Camara and Loureiro, 2000; Cook, 2012; Dare and Saleh, 2000; Jung et al., 2006), this is clearly a well-known reference for this problem.

- Drilling, which refers to the minimization of the total numbers of moves in a computer controlled drill (Grötschel et al., 1991; Lin and Kernighan, 1973; Onwubolu, 2004).

- Computer wiring (Lenstra and Kan, 1975), refers to the minimization of the wire used to link computer components. The components have an established position and the goal is to join them all with the minimum amount of wire, starting and ending in non-specified component.

Several formulations of this problem have been reported, such as those of Laporte (1992); Miller et al. (1960); Orman and Williams (2007). A classic formulation for this problem is defined on a *complete directed graph* $G = (V, A)$, where A denotes the *arc* set and $V=\{v_1, v_2, v_3, \ldots, v_n\}$ is the *vertex* set; n is the number of vertices the agent must visit. The matrix $C = t_{ij}$ represents the costs to travel from client i to client j. Binary decision variables x_{ij} are defined as,

$$x_{ij} = \begin{cases} 1, & \text{if arc from vertex } i \text{ to vertex } j \text{ is included in the tour,} \\ 0, & \text{otherwise.} \end{cases}$$

With these decision variables, the mathematical formulation for the TSP is defined as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij} \tag{2.4}$$

subject to

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} x_{ij} = 1, \ \forall j \tag{2.5}$$

$$\sum_{\substack{j=1 \\ i \neq j}}^{n} x_{ij} = 1, \ \forall i \tag{2.6}$$

$$\sum_{\substack{(i,j) \in A \\ i \in S \\ j \in V \setminus S}} x_{ij} \geq 1, \ \forall S \subset V \tag{2.7}$$

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

where Equation (2.4) depicts the TSP objective of minimizing the total travel distance of the vehicle. The sets of Equations (2.5) and (2.6), both of cardinality n, guarantee that the agent visits every vertex only once. To guarantee that no solution contains subtours, the set of Equations (2.7) with cardinality 2^n is used. Lastly, Equation (2.8) corresponds to the integer nature of the variables. In general, the number of solutions of an instance of TSP is $n!$, where n denotes the size of the instance (Applegate et al., 2007; Laporte, 2006; Papadimitriou, 1994).

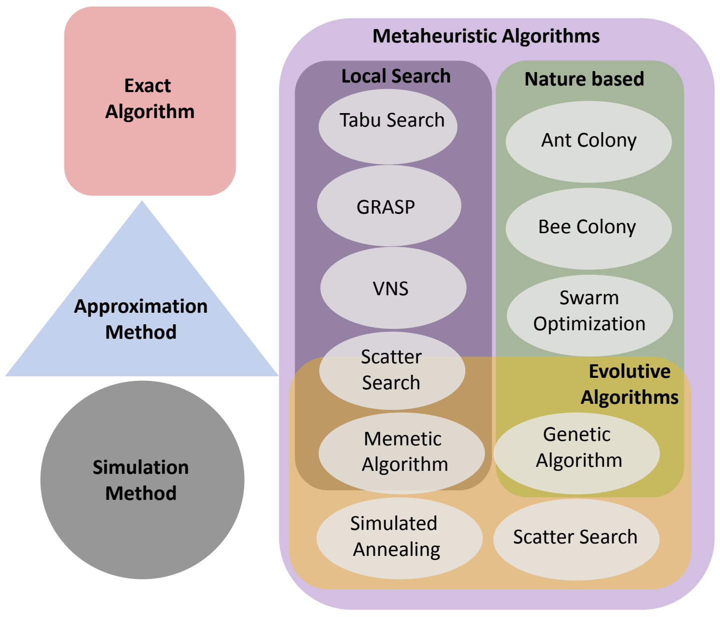


Figure 2.4: Synoptic table depicting several solution methods employed in solving combinatorial problems, among which routing problems are immersed.

Due to its complexity (Papadimitriou, 1994), there exist not only improvements to the known mathematical formulations, but heuristic algorithms to produce good quality solutions faster. A general graphical depiction of some approximated techniques for the TSP is shown in Figure 2.4, where several solution approaches are summarized. The most representative methodologies employed to solve TSP include.

- *Approximation methods* that attempt to find approximate solutions with a certainty of the distance between the approximated solution and the exact one. Examples of works using approximation methods include Arora (1998); Asadpour et al. (2010); He and Xiang (2017); Karp (1977); Karpinski et al. (2015), among others. The principal difference between an approximation algorithm and a heuristic or a simulation is that an approximation algorithm involves a mathematical proof to certify the distance between the approximated and the exact solution (Johnson, 1974).
- *Simulation methods*, such as:
  - Monte Carlo simulations (Chen and Aihara, 1995; Hammersley, 1960; Lee and Choi, 1994; Powley et al., 2012; Vanderbilt and Louie, 1984; Wong and Liang, 1997) and Markov simulations (Martin et al., 1992; Ulmer et al., 2017).
- *Heuristic methods* are techniques to quickly produce near optimal solutions for NP-hard problems such as method deals with a trade-off, losing the assurance of optimality by minimizing computation time (Silver et al., 1980; Zanakis et al., 1989). There exist several heuristic types employed for solving TSP:
  - Constructive heuristics that part from constructing good quality solutions based on the structure of the problem as in Bentley (1992); Gendreau et al. (1992); Lin and Kernighan (1973); Rosenkrantz et al. (1977). A

disadvantage of such heuristics is the greediness creating apparently good solutions in the beginning while being stuck with expensive decisions later on due to the local blindness during the construction process.

  - Local-search heuristics that take a solution constructed beforehand and disturb it with the goal of improving it by examining neighborhoods (Freisleben and Merz, 1996; Gu and Huang, 1994; Johnson, 1990; Johnson and McGeoch, 1997; Paquete and Stützle, 2003; Voudouris and Tsang, 1999).
  - Combined heuristics that combines both of the above types (Gendreau et al., 1992).

- *Metaheuristics* that are algorithms defined as higher-level procedures, designed to employ a heuristic to provide sufficiently good solutions (Gendreau and Potvin, 2010; Glover and Kochenberger, 2006). There exist several types of metaheuristics, including:
  - Nature colony optimization, based in natural systems such as ant colonies (Bianchi et al., 2002; Chen and Chien, 2011; Dorigo and Gambardella, 2016, 1997; Dorigo et al., 2006; Duan and Yu, 2007; Junjie and Dingwei, 2006), bee colony (Hu and Zhao, 2009; Karaboga and Gorkemli, 2011; Marinakis et al., 2011; Teodorovic et al., 2006; Wong et al., 2008, 2010) or swarm optimization (Chen and Chien, 2011; Clerc, 2004; Lope and Coelho, 2005; Marinakis and Marinaki, 2010; Tasgetiren et al., 2007; Wang et al., 2003). The main idea of such procedures is to recreate animal behaviors.
  - Simulated annealing, inspired on annealing in metallurgy, aims to approximate the global optimum of a given function in a large search space (Chen and Chien, 2011; Malek et al., 1989; Randelman and Grest, 1986;

Vanderbilt and Louie, 1984), among others.

- Local-search-improvement metaheuristics are algorithms based on the idea of avoiding any local search heuristic being hindered at a local optimum. These algorithms include tabu search (Fiechter, 1994; Gendreau et al., 1998; Knox, 1994; Malek et al., 1989; Misevičius, 2004; Thamilselvan and Balasubramanie, 2009; Toth and Vigo, 2003), variable neighborhood search (Burke et al., 2001; Carrabs et al., 2007; Da Silva and Urrutia, 2010; Hemmelmayr et al., 2009; Hernández-Pérez et al., 2009; Mladenović and Hansen, 1997) as well as guided local search procedures (Kilby et al., 1999; Lope and Coelho, 2005; Voudouris and Tsang, 1999).
- Greedy randomized adaptive search procedure, better known as GRASP, that constructs an initial solution by iterating a greedy constructive algorithm and sequentially applies iterative improvements through a local search. Some examples are found in the research of Hernández-Pérez et al. (2009); Marinakis et al. (2005)
- Genetic algorithms take inspiration in evolutionary theory, these algorithms are based on the idea of having a set of initial solutions and iterating it under certain conditions to improve the population (Ahmed, 2010; Choi et al., 2003; Duan and Yu, 2007; Freisleben and Merz, 1996; Grefenstette et al., 1985; Krasnogor and Smith, 2000; Larranaga et al., 1999; Merz and Freisleben, 1997; Potvin, 1996; Thamilselvan and Balasubramanie, 2009; Ulder et al., 1990).

TSP is considered a company-centered problem as it minimizes the travel distance, which can be seen as the minimization of costs for the company in charge of providing the service to the clients. In the rest of this dissertation we discuss some other routing problems and the multi-objective approach we aim for. Section

2.3 gives a general overview of a natural generalization of TSP: multi-agents routing. Section 2.6 provides a general overview of bi-objective problems involving the minimization of travel distance.

2.3 m-Traveling Salesman Problem

A natural generalization of TSP comes from not being restricted to having just a single-vehicle with infinite capacity to visit the clients, but instead, employing a *fleet* of vehicles with infinite capacity to attend the requests of the client set. This generalization must satisfy certain conditions: every vehicle in the fleet must have at least one assigned client. Each vehicle must not execute subtours. This problem is known as the multiple Traveling Salesman Problem or as the m-Traveling Salesman Problem (m-TSP) (Bektas, 2006), where the m denotes the size of the fleet to perform the visits. A graphical representation of this problem is shown in Figure 2.5. Its applications include

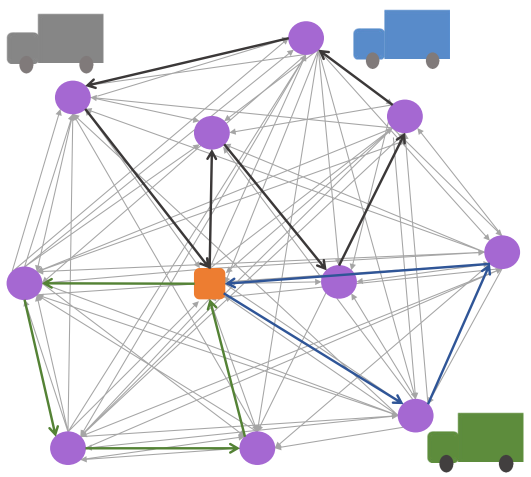


Figure 2.5: Graphical example of an m-TSP instance.

- Routing of vehicles, such as school buses or delivery trucks. The objective is to obtain a tour for each available vehicle, while having a fixed depot from where all vehicles must part and arrive at the end of their tours. These approaches aim to minimize the total travel distance by all buses. Examples include works of Bektaş and Elmastaş (2007); Clarke and Wright (1964); Li and Fu (2002); Min (1989); Park and Kim (2010); Schittekat et al. (2006, 2013); Tang et al. (2000)

- Door-to-door-querist, when a set of querists interview citizens at their homes, each interviewer must leave from a fixed point and have a set of homes to visit (Gilbert and Hofstra, 1992; Kergosien et al., 2009; Sipahioglu et al., 2008)

- Satellite system navigation. In global navigation satellite system (GNSS) surveying networks, the satellites transmit signals for navigation purposes (Groves, 2013) and determine the geographical position of unknown points above Earth by satellite equipment. Such application aims to find the best order of sessions for the receivers to allocate such unknown points and when having several receivers it can be formulated as an m-TSP (Saleh and Chelouah, 2004).

- Crew scheduling: a delivery company has set of agents to deliver items or goods to a set of clients, each agent must leave from a central depot and come back to it at the end of its tour (Lenstra and Kan, 1975; Raff, 1983; Svestka and Huckfeldt, 1973; Zhang et al., 1999).

By adding capacity constraints, m-TSP is transformed into several more realistic types of vehicle-routing problems (VRP). Literature for TSP is extensive, but m-TSP has not received the same amount of attention (Bektas, 2006). Several formulations for m-TSP have been reported in literature (Bektas, 2006; Gavish, 1976; Henry, 1981; Hong and Padberg, 1977; Kara and Bektas, 2006; Kergosien et al., 2009; Rao, 1980; Svestka and Huckfeldt, 1973). A classic mathematical formulation is defined on a complete directed graph $G = (V, A)$, where A denotes the arc set and $V{=}\{v_1, v_2, v_3, \dots, v_n\}$ is the vertex set and v_0 is the depot from where all vehicles part and come back at the end of their tours. A matrix $C = t_{ij}$ represents the cost it has to travel from client i to client j; the total of agents in the fleet is denoted as m. Binary decision variables x_{ij} are defined as,

$$x_{ij} = \begin{cases} 1, & \text{if the arc from vertex } i \text{ to vertex } j \text{ is used on the tour,} \\ 0, & \text{otherwise.} \end{cases}$$

With these decision variables, the m-TSP mathematical formulation is defined as follows,

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij} \tag{2.9}$$

$$\text{subject to}$$

$$\sum_{j=2}^{n} x_{0j} = m \tag{2.10}$$

$$\sum_{i=1}^{n} x_{j0} = m \tag{2.11}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \ \forall j = 2, \dots, n \tag{2.12}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \ \forall i = 2, \dots, n \tag{2.13}$$

$$\sum_{j \in S} x_{ij} \geq |S| - \{1, \ \forall S \subset V \setminus \{1\}, S \neq \emptyset \tag{2.14}$$

$$x_{ij} \in \{0, 1\}, \tag{2.15}$$

where Equation (2.9) describes the m-TSP objective of minimizing the total traveled distance of the m-vehicle fleet. Equations (2.10) and (2.11) express that exactly m vehicles are active by ensuring that all m agents parts from the depot and come back at it at the end of the tour. Equations (2.12) and (2.13) guarantee that each client is visited once by the agent assigned, and Equation (2.14), guarantees that no tour has subtours in it. Lastly, Equation (2.15) corresponds to the integer nature of the variables.

According to Garey and Johnson (1990), K-TSP retains the computational complexity of TSP: it also is NP-hard. Figure 2.4 in Section 2.2 depicts several solution methods. The most representative methodologies employed for solving m-TSP include:

- *Approximation methods* that aim to obtain approximated solutions but with a certainty of the distance between the optimal solution and the approximated one (Gavish and Srikanth, 1986; Jonker and Volgenant, 1988; Laporte, 1992).
- *Simulation methods* that attempt to imitate the theoretical behavior of the m-TSP, such as neural network (Modares et al., 1999).
- *Heuristic methods* are algorithms to quickly obtain and optimal solutions (Silver et al., 1980; Zanakis et al., 1989), such as:
  - Constructive heuristics that exploit the structure of the problem to construct good quality solutions (Gilbert and Hofstra, 1992).
  - Local-search heuristics disturb a solution with the goal of improving it by searching local neighborhoods (Shokouhi rostami et al., 2015)
  - Combined heuristics (Gilbert and Hofstra, 1992)
- *Metaheuristics* are higher-level procedures designed to employ a heuristic to provide good solutions (Gendreau and Potvin, 2010; Glover and Kochenberger, 2006), such as:

– Nature colony optimization, based on natural systems such as ant colony or swarm optimization (Ghafurian and Javadian, 2011; Junjie and Dingwei, 2006; Sofge et al., 2002).

– Simulated annealing (Geng et al., 2011; Nallusamy et al., 2010).

– Local-search-improvement heuristics (Nallusamy et al., 2010; Shokouhi rostami et al., 2015).

– Genetic algorithms (Carter and Ragsdale, 2006; Fogel, 1990; Király and Abonyi, 2011; Sofge et al., 2002; Tang et al., 2000; Zhang et al., 1999; Zhao et al., 2008)

2.4 Minimum Latency Problem

In this dissertation, two main objectives are explored: the minimization of the total travel distance of a vehicle or a fleet of vehicles and the minimization of the total waiting time of the clients.

To minimize the total travel distance is to pursue an economic objective for the company. We aim to incorporate a client-oriented objective in a routing decision by minimizing the waiting time of the clients in the tour. A waiting period is known as *latency*.

The *Minimum Latency Problem* MLP, also known as the *Traveling Repairman Problem* TRP and as the *Delivery Man Problem* DMP (Blum et al., 1994), aims to serve as quick as possible a set of clients waiting to be given a service provided by a company in charge of a single agent assigned to make the visiting. As in the TSP, an agent must visit each client only once, restrained to part from and return to a known fixed depot, knowing all exact locations of the clients and therefore all travel times. As Blum et al. (1994) states, at first glance, TSP and MLP may seem similar problems but in reality this is not the case; The main differences are,

- Objectives: two problems is that they do not pursue the same goals. MLP is a client-centered problem which aims to minimize the time each client awaits for a service despite the cost the tour may have (Afrati et al., 1986; Blum et al., 1994; García et al., 2002).
- Service times: for MLP the agent must also know the service time required to attend each client.
- Metric: latency and distance are computed completely differently (Blum et al., 1994).

Let us define MLP on a complete directed graph $G = (V, A)$, where A denotes the arc set and V=$\{v_0, v_1, v_2, v_3, \ldots, v_n\}$ represents the vertex set. v_0 denotes the depot and $v_1, \ldots, v_n$ the clients to visit. A non-negative matrix $C = c_{i,j}$ associated. Let us denote the service time at client i as s_i and the travel time from vertex i to vertex j as t_{ij}. The total waiting time is defined by the time the agent takes to reach the client's location. By considering the time the agent spent with the previous clients before arriving to client i and considering all the traveling times before reaching i,

$$c_{i,j} = s_i + t_{ij}. \tag{2.16}$$

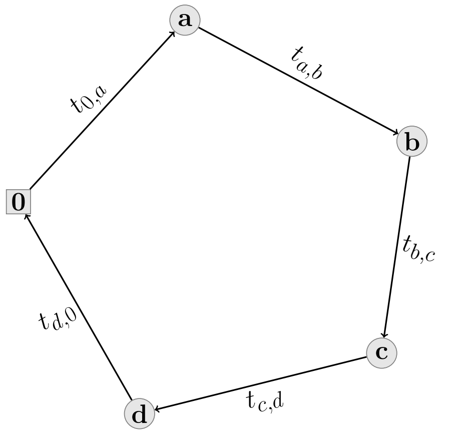
A feasible solution for MLP is a permutation of the clients, with the depot in the first position. Latency at vertex i is the length of the path traveled by the agent from the depot to i,

$$\ell_i = \sum_{j=1}^{i} c_{j-1,j}. \tag{2.17}$$

The total latency of the tour is therefore,

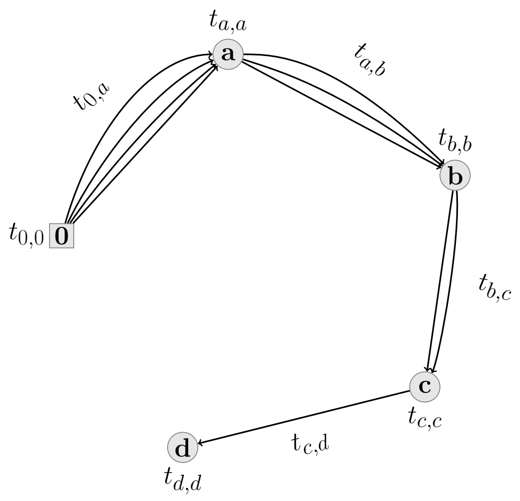
$$\begin{aligned} L &= \sum_{i=1}^{n} \ell_i \\ &= \sum_{i=1}^{n} (n-i+1)c_{i-1,i}. \end{aligned} \tag{2.18}$$

Afrati et al. (1986); Angel-Bello et al. (2013); Angel-Bello et al. (2013); Blum et al. (1994); Gouveia and Voß (1995); Lucena (1990); Nagarajan and Ravi (2008); Orman and Williams (2007); Picard and Queyranne (1978), among others, highlight a *contribution* of the position of the clients in the tour, which means, each arc will contribute a certain weight in the final calculation of the total latency of the tour. This contribution is considered a dragged weight throughout the tour until the visiting is over. Figure 2.6, illustrates graphically the difference in the calculation of the metrics in the latency (MLP) and the distance (TSP) objectives. Notice Figure 2.6a illustrates the straightforward calculation of the distance objective. Figure 2.6b, exemplifies the importance in the order of the clients in the tour due to the local contribution of each arc and the impact in the total latency of the tour.



(a) Distance objective D

$D = t_{0,a} + t_{a,b} + t_{b,c} + t_{c,d} + t_{d,0}$



(b) Latency objective L,

$L = \ell_0 + \ell_a + \ell_b + \ell_c + \ell_d$,

$\ell_i = t_{i,i} + t_{i,j},\ \ell_0 = 0$

$L = 0 + (t_{a,a} + t_{0,a}) + ((t_{a,a} + t_{0,a}) + t_{b,b} + t_{a,b}) + \ldots + ((((t_{a,a} + t_{0,a}) + (t_{b,b} + t_{a,b})) + (t_{c,c} + t_{b,c}) + (t_{d,d} + t_{c,d}))$

$L = 4(t_{a,a} + t_{0,a}) + 3(t_{b,b} + t_{a,b}) + 2(t_{c,c} + t_{b,c}) + (t_{d,d} + t_{c,d})$

Figure 2.6: The difference in the metrics to evaluate the objectives of the TSP and the MLP. To minimize the travel distance (see Figure 2.6a), only the sum of all travel times is considered. Figure 2.6b illustrates the contribution of every visited vertex in the final latency of the tour.

Note that in Figure 2.6a, the returning arc is necessarily considered due to the definition of TSP and on Figure 2.6b this returning arc is not considered. This is due to the definition of MLP: the latency aimed to minimize is at the clients. The vehicle must return to the depot at the end of the tour, but that final arc is not considered in the total latency of the tour. Figure 2.7 shows a graphical representation of a single-vehicle tour in charge of visiting a set of ten clients while minimizing the total latency. Each grey arrow depicts a known possible connection among all clients and the depot. In this example, no returning arc is depicted due to MLP definition; the darker lines indicate the tour the agent follows to visit the clients.

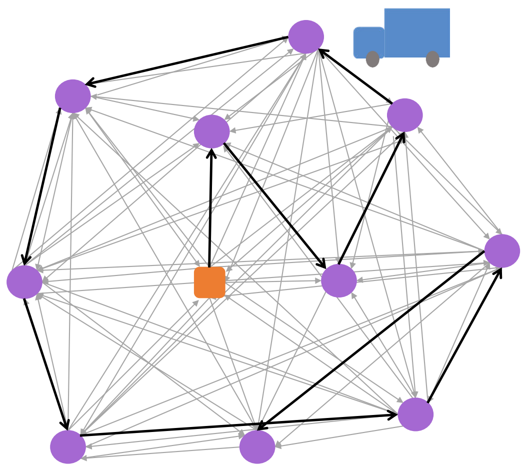


Figure 2.7: Graphical example of an small MLP solution in an instance of ten vertices.

The interest in the MLP has grown in the Operations Research area (Blum et al., 1994; Chaudhuri et al., 2003; García et al., 2002; Lucena, 1990). Example applications include:

- Humanitarian logistic: the delivery and supply of essential provisions during natural disasters or emergencies is a direct application of the minimization of the latency due to the importance in providing essential items to the victims of a disaster in the shortest possible time (Caunhye et al., 2012; Ferrer et al., 2016; Kovács and Spens, 2007, 2009; Leiras et al., 2014; Overstreet et al., 2011; Stephenson, 2017; Tomasini et al., 2009; Vargas et al., 2017).

- Medicine: the study of delays between an impulse and the time it takes action in the human body (Darcis et al., 2017; Goldfarb et al., 2018; Goodwill, 1965; John and Picton, 2000; Kohara et al., 2000; Littner et al., 2005; Madsen

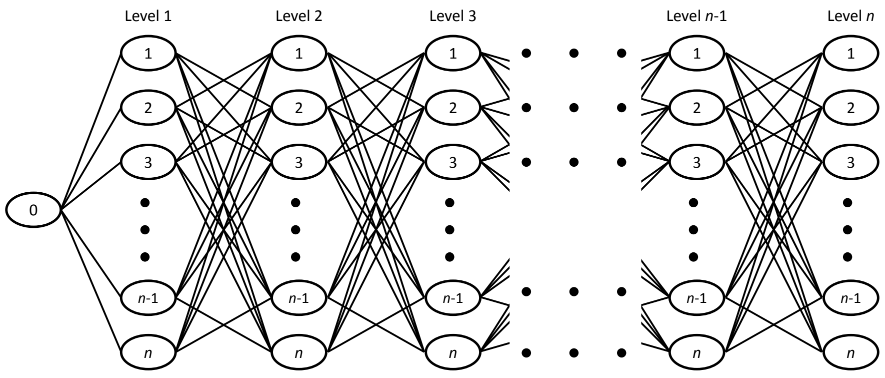


Figure 2.8: Multi-level network used to formulate the MLDP.

et al., 2018; Paniagua-Soto et al., 2013; Saito et al., 2018; Squires et al., 1975; Sternberg et al., 1978).

- Telecommunications: minimization of the delay in real-time connections such as wireless networks (Crowcroft et al., 2015; Grout et al., 2007; Gummadi et al., 2002) and communication systems (Crowcroft et al., 2015; Joo Ghee et al., 2009; Kao et al., 2017; Lichtsteiner et al., 2008; Lu et al., 2007; Przybył, 2018; Schurgers et al., 2002).
- Routing: improvement of the response time in a client-centered approach (Bock, 2015; Bulhoes et al., 2018; Das et al., 2018; Tsitsiklis, 1992).

In a routing context, research of latency has lead to the development of several mathematical formulations to characterize it (Angel-Bello et al., 2013; Gouveia and Voß, 1995; Méndez-Díaz et al., 2008; Picard and Queyranne, 1978; Sarubbi et al., 2008). Among them, the formulation with the best performance is reported by Angel-Bello et al. (2013). Their formulation is based on a multi-level network, shown in Figure 2.8, which has $n + 1$ levels. Such network is inspired on a previous work of Picard and Queyranne (1978) for a time-dependent TSP.

By definition, solutions to MLP are permutations of all vertices belonging to V with the depot in the first position. On each level $\ell_i, i > 0$ of the network, a vertex v_i is present: a solution corresponds to a path that visits exactly one vertex per level. No vertex can be selected on multiple levels.

$$x_{i,k} = \begin{cases} 1, & \text{if vertex } i \text{ is selected on level } k \text{ on the network,} \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{i,j}(k) = \begin{cases} 1, & \text{if vertex } j \text{ on level } k+1 \text{ follows vertex } i \text{ on level } k, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, MLP is formulated as:

$$\min\ F_1 = n\sum_{i=1}^{n} c_{0i}x_i^1 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(n-k)c_{ij}y_{i,j}(k) \tag{2.19}$$

subject to

$$\sum_{k=1}^{n} x_{i,k} = 1,\ i = 1,2,\dots,n \tag{2.20}$$

$$\sum_{i=1}^{n} x_{i,k} = 1,\ k = 1,2,\dots,n \tag{2.21}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{i,j}(k) = x_{i,k},\ i = 1,2,\dots,n,\ k = 1,2,\dots,n-1 \tag{2.22}$$

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{j,i}(k) = x_{i,k+1},\ i = 1,2,\dots,n,\ k = 1,2,\dots,n-1 \tag{2.23}$$

$$x_{i,k} \in \{0,1\},\ i = 1,2,\dots,n,\ k = 1,2,\dots,n \tag{2.24}$$

$$y_{i,j}(k) \geq 0,\ i = 1,2,\dots,n,\ j = 1,2,\dots,n, j \neq i,\ k = 1,2,\dots,n. \tag{2.25}$$

The minimization of the total waiting time of the clients is defined by Equation (2.19). Equation (2.20) guarantees that each vertex occupies a single position in any feasible solution. Equation (2.21) guarantees that each position is occupied by no more than one vertex in any feasible solution. Equation (2.22) ensures that only one arc leaves from position k, exactly from the vertex taking that position, and Equation (2.23) imposes that only one arc at a time can arrive to position $k+1$, exactly to the vertex occupying that position. Equations (2.20) to (2.23) require that all possible solutions for the MLDP can be represented on the multi-level network previously defined. Equations (2.24) and (2.25) correspond to the nature of the variables. Note that variables $y_{i,j}(k)$ are previously defined as binary but they are handled as continuous variables in the model. This is achieved by Equations (2.22) and (2.23), which guarantee that only one arc leaves from position k and arrives exactly to the vertex occupying the position $k+1$. These pair of constraints ensures that $y_{i,j}(k)$ is either one or zero.

MLP is NP-hard (Afrati et al., 1986). Figure 2.4 reflects the state of the art in approximation techniques used in combinatorial problems, such techniques are also used for MLP. Solution methods for MLP include:

- *Approximation methods*: to find approximate solutions with a certainty of the distance between the approximated solution and the exact one (Archer and Williamson, 2003; Archer et al., 2008; Arora and Karakostas, 1999; Frederickson and Wittman, 2012; Garg, 1996; Goemans and Kleinberg, 1998).
- *Simulation methods* that attempt to imitate the behavior of MLP, such as Monte Carlo simulations (Agnihothri, 1988).
- *Heuristic methods* to quickly produce near optimal solutions (Silver et al., 1980; Zanakis et al., 1989), such as
  - Constructive heuristics that construct good quality solutions by using the structure of the problem (Frederickson and Wittman, 2012; Ma et al.,

2015).

  - Local-search heuristics that disturb a solution with the goal of improving it by searching local neighborhoods (Bjelić and Popović, 2015).
  - Combined heuristics (Bjelić and Popović, 2015).

- *Metaheuristics* are higher-level procedures that employ a heuristic to provide good solutions (Gendreau and Potvin, 2010; Glover and Kochenberger, 2006) such as:
  - Nature colony optimization (Shyu et al., 2004).
  - Simulated annealing (Ma et al., 2015; Moshref-Javadi and Lee, 2016).
  - Local-search-improvement metaheuristics, such as tabu search (Dewilde et al., 2013), variable neighborhood search (Bjelić et al., 2013; Moshref-Javadi and Lee, 2016; Salehipour et al., 2011), and guided local search procedures (Vargas et al., 2017).
  - GRASP (Ferrer et al., 2016; Salehipour et al., 2011; Silva et al., 2012).
  - Genetic algorithms (Bang and Nghia, 2010).

Published research for MLP is not as extensive as the research available for TSP. It ranges from mainly approximation techniques (Archer and Williamson, 2003; Archer et al., 2008; Arora and Karakostas, 1999; Garg, 1996; Goemans and Kleinberg, 1998) to heuristics and metaheuristics (Salehipour et al., 2011; Silva et al., 2012).

This problem brings a new point of view to classical routing problems by giving priority to the clients waiting to be attended. This problem has a client-centered objective important to improve quality in the service given by a company.

2.5 k-Minimum Latency Problem

Similarly to TSP, a natural generalization of MLP comes from not being restricted to having just a single agent but to employ a fleet of vehicles instead. Every vehicle in the fleet must have at least one assigned client, each active vehicle must not execute sub-tours and every client can only be visited once. This problem is known as the *Multiple Repairman Problem* or as the *k-Traveling Repairman Problem* (k-TRP) (Fakcharoenphol et al., 2003; Nucamendi-Guillén et al., 2016; Onder et al., 2017).

In the multi-vehicle latency approach, k-TRP, each one of the k agents is in charge of a subset of clients. Similarly than in the one-tour approach, each agent knows beforehand the exact location of each client, the service time spent on each client, and the travel times among them. Similarly, the k tours in k-TRP do not contemplate returning arcs to the depot and only count the latency at the clients. A graphical representation of k-TRP instance is shown in Figure 2.9.

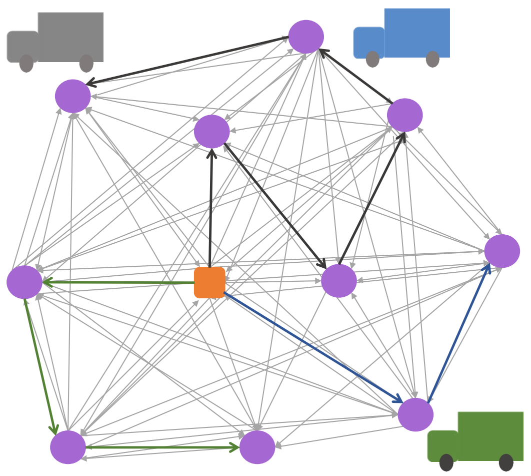


Figure 2.9: Graphical example of an k-TRP instance.

A k-TRP solution can be computed as k single tours. The latency is computed per tour and all k resulting latencies are summed to obtain the total latency of the fleet. In Figure 2.9 the grey arcs depict possible connections among clients and depot whereas the darker lines represent the arcs traveled by each agent in the tour, differing in color, not considering the return arcs to the depot as it does not affect at the objective.

Applications of the k-TRP include:

- Routing: a fleet of vehicles fulfills a service to a set of clients (Avci and Avci, 2017; González and Rivera, 2015; Ha Bang, 2018; Martin and Salavatipour, 2016).
- Humanitarian logistics: as in post-disaster resilience to schedule multiple teams of repairmen to fix damaged facilities (Dikbyk, 2017; Ma et al., 2015).

There exist several formulations in literature for k-TRP (Luo et al., 2014; Nucamendi-Guillén et al., 2016; Onder et al., 2017); a multi-level formulation with good computational results is reported in the works of Angel-Bello et al. (2017); Nucamendi-Guillén et al. (2016). To formulate k-TRP, let us declare a complete directed graph $G = (V, A)$, where A denotes the arc set and $V=\{v_0, v_1, v_2, v_3, \ldots, v_n\}$ represents the vertex set, v_0 denotes the depot, and $v_1, \ldots, v_n$ the clients to visit. This multiple repairman problem has k vehicles to fulfill the client requests and is restricted to having all k agents active. Therefore, a bound on the number of clients assigned to a client is,

$$N = n - k + 1. \tag{2.26}$$

Equation (2.26) characterizes the worst-case scenario for each agent, which is that all other agents only have one single client assigned. A non-negative matrix $C = c_{ij}$ represents the total cost it takes the agent to commence service at client i after client j. Figure 2.10 represents the $N + 1$ multilevel network used to characterized

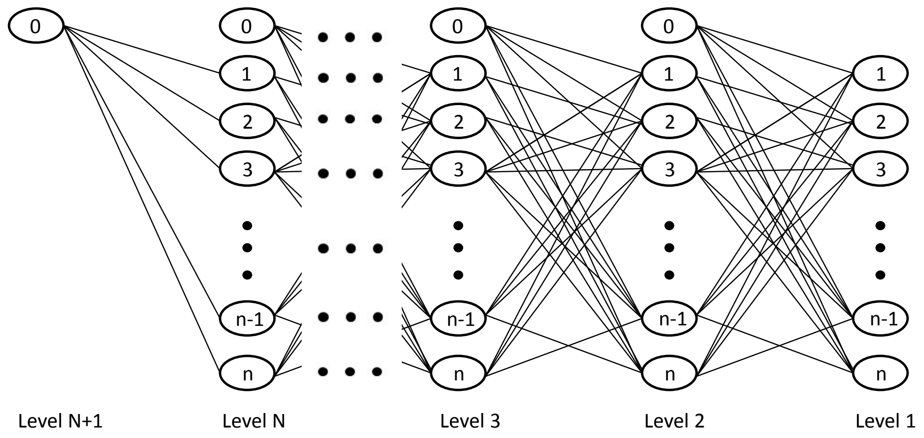


Figure 2.10: Multi-level network used to formulate the k-TRP.

this problem mathematically: level one consists of a copy of all vertices associated with clients $v_1, v_2, \ldots, v_n$, levels 2,3,...,N consist of a copy of all vertices belonging in the set V, and level $N+1$ represents the depot.

By definition, a feasible solution for k-TRP is a set of k paths starting in the depot and visiting vertices until all k agents have visited all vertices belonging to V. In this network, each of the k paths to express a feasible solution can be successfully represented by starting each path in a copy of the depot and visiting vertices in the lower levels of the network. As stated by definition of k-TRP, only one agent can visit each client, this is to say a client i can only belong to one path. Each path should only visit one vertex on each level below the starting level; if a path starts in vertex l, then this path has $l-1$ vertices associated. With this, decision variables are defined as:

$$x_{i,l} = \begin{cases} 1, & \text{if client } i \text{ is visited in position } l \text{ in the path,} \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{ij}(l) = \begin{cases} 1, & \text{if client } i \text{ in position } l+1 \text{ is followed by client } j \text{ in the} \\ & \text{position } l \text{ in the path,} \\ 0, & \text{otherwise.} \end{cases}$$

With these variables, the k-TRP formulation is defined as follows,

$$\min\ F_1 = \sum_{j=1}^{n} c_{0j} \sum_{l=1}^{N} l y_{0,j}(k) + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} \sum_{l=1}^{N-1} l y_{i,j}(l) \tag{2.27}$$

subject to

$$\sum_{l=1}^{N} x_{i,l} = 1,\ i = 1, 2, \ldots, n \tag{2.28}$$

$$\sum_{i=0}^{n} x_{i,1} = k \tag{2.29}$$

$$\sum_{l=1}^{N} \sum_{j=1}^{n} y_{0,j}(l) = k \tag{2.30}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} y_{i,j}(l) = x_{i,l+1},\ i = 1, 2, \ldots, n;\ l = 1, 2, \ldots, N-1 \tag{2.31}$$

$$y_{0,j}(l) + \sum_{\substack{i=1 \\ i \neq j}}^{n} y_{i,j}(l) = x_{j,l},\ j = 1, 2, \ldots, n;\ l = 1, 2, \ldots, N-1 \tag{2.32}$$

$$y_{0,j}(N) = x_{j,N},\ j = 1, 2, \ldots, n \tag{2.33}$$

$$x_{i,l} \in \{0, 1\},\ i = 1, \ldots, n;\ l = 1, \ldots, N \tag{2.34}$$

$$y_{i,j}(l) \geq 0,\ i = 1, \ldots, n;\ j = 1, \ldots, n;\ l = 1, \ldots, N-1. \tag{2.35}$$

The minimization of the total waiting time of all the clients visited by the fleet of vehicles is defined by Equation (2.27). Equation (2.28) guarantees that there can only be one active vertex per level, meanwhile Equation (2.29) requires that each agent must visit at least a client, to ensure all k vehicles are active. Equation (2.30) ensures all k vehicles are active, Equation (2.31) guarantees that only from the vertices active in level $l + 1$, arcs can leave. Equation (2.32) guarantees that from level l in the network can only arrive arcs to the active vertices in such level and Equation (2.33) ensures that if a vertex in level N is active, level $N + 1$ must be connected to the vertex active in level N. Equations (2.34) and (2.35) denotes the nature of the variables. Note that variables $y_{i,j}(l)$ are previously defined as binary but they are handled as continuous variables in the model. This is achieved because binary variables $x_{i,l}$ determine the active vertices in the network and this simplifies finding the k shortest paths from the active vertex in level l through the active vertices in the intermediate levels. This is an uni-modular problem and, therefore, there is no need to define $y_{i,j}(l)$ as binary variables.

Due to the complexity of k-TRP (Sitters, 2014), several approximation methods and heuristics have been reported, such as:

- *Approximation methods* that find approximate solutions with a certainty of the distance between the approximated solution and the exact one (Fakcharoenphol et al., 2003; Martin and Salavatipour, 2016; Sitters, 2014).
- *Heuristic methods* that quickly produce near optimal solutions. Some examples of heuristics for k-TRP include:
  - Constructive heuristics that construct good quality solutions by using the structure of the problem (Ma et al., 2015).
  - Local-search heuristics that take a solution and disturb it with the goal of improving it (Avci and Avci, 2017; González and Rivera, 2015).
  - Combined heuristics that combine both heuristics types above.

- *Metaheuristics* are procedures designed to employ a heuristic to provide good solutions (Gendreau and Potvin, 2010; Glover and Kochenberger, 2006). Examples of metaheuristics for k-TRP include:
  - Simulated annealing, (Ma et al., 2015).
  - Local-search-improvement metaheuristics such as variable neighborhood search (Ha Bang, 2018), and guided local search procedures (Avci and Avci, 2017; González and Rivera, 2015; Nucamendi-Guillén et al., 2016).
  - GRASP, as in Ha Bang (2018)

2.6 Multi-objective optimization

Problems that consider a set of constraints while optimizing a single objective are the main pillar of Operations Research (Ahuja et al., 1993; Frederick and Lieberman, 1982; Hillier and Lieberman, 1986; Taha, 1992). In general, a single-objective problem is solved when an optimum solution is found and labeled as the *best* in a set of possible solutions. All problems mentioned in Sections 2.2, 2.3, 2.4, and 2.5 are basic routing problems, each with a single objective function to optimize.

Solving real-life situations implies the challenge of taking decisions which involves the optimization of several objectives simultaneously. A realistic approach to real-life problems tends to involve several conflicting objectives and the challenge of optimizing them all, leads to the redefinition of *best* solution. Decision-making under multiple objectives can no longer be defined under a single-objective preference and this leads to the employment of multi-criteria or *multi-objective optimization*.

This area of study is mainly applied in field of science where optimal decisions are made under a trade-off among two or more conflictive objectives (Deb et al., 2016; Ehrgott, 2005; Sengupta, 2016). Some examples of a conflict in the objectives are found in problems that aim to optimize contradictory objectives such as the classic example or minimizing the cost of a vehicle while maximizing the comfort or performance (Ehrgott, 2005). A multi-objective problem has no single solution that simultaneously optimizes all objectives, this is to say, there exists a grand amount of "optimal solutions" (Ehrgott, 2005; Romero, 1993). A solution is considered as a trade-off, as a decision among the *best* solutions among a set of *alternatives* that need a defined criteria to rank them according to which the quality of the alternatives is measured.

A multi-objective problem has the following general structure:

$$\textit{optimize } f(X) = [f_1(x), \ldots, f_i(x), \ldots, f_q(x)] \tag{2.36}$$

$$\text{subject to:}$$
$$x \in X,$$

where Equation (2.36) denotes the search of efficient solutions for the $f_i(x)$ objectives simultaneously optimized, X is the feasible set, x the decision variables vector and $f(X)$ denotes the image of the feasible set. Multi-objective optimization determines a subset, colloquially called trade-off, of the feasible set of solutions for which every belonging is a good solution in Pareto terms.

Definition 2.6.1. Pareto optimal. A feasible solution $x^* \in X$ is *efficient* or *Pareto optimal* if there is no other solution $x \in X$ such that $f(x) \leq f(x^*)$.

Definition 2.6.2. Non-dominated point. If a solution x^* is satisfies the definition of efficient solution, $f(x^*)$ is called a *non-dominated point*.

Definition 2.6.3. Dominance. If two solutions x_1 and x_2 belong to X, and $f(x_1) \leq f(x_2)$, it is said that x_1 dominates x_2 and $f(x_1)$ dominates $f(x_2)$.

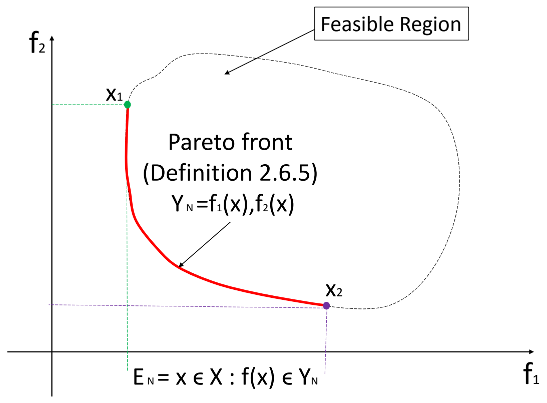


Figure 2.11: Basic definitions of Pareto optimality: two objective functions are being minimized; the Pareto front is drawn in red.

Definition 2.6.4. Efficient set. The set of all Pareto optimal solutions $x^* \in X$ is called *efficient set* and is denoted by X_E.

Definition 2.6.5. Non-dominated set. All non-dominated points $Y_N : y^* = f(x^*) \in Y,$ where $x^* \in X_E$ is called the non-dominated set or *Pareto front.*

The above definitions are given in minimization terms of $f(x)$, and are depicted in Figure 2.11, where two objective functions are shown. Note that both objectives are being minimized and the Pareto front is depicted in red. The efficient set X_E (Definition 2.6.4), is formed by all non-dominated points that their images belong to the non-dominated set or Pareto frontier. In this particular example, two example points are shown in green and purple, and their image is directly depicted in the front. The non-dominated region is Y_N (Definition 2.6.5), drawn in red.

A Pareto front has two important bounds to define the range in which non-dominated points can belong to; these bounds are used as reference points to several solution methods: *the ideal point*, or lower bound, and *the nadir point*, or the upper bound. Note that a Pareto-optimal solution is a vector of all values of a point in all objectives.

The *ideal point*, is the solution vector which includes the optimal values of all objective functions, simultaneously optimized. In general, such solution corresponds to a non-existing solution, due to the nature of a multi-objective problem in which conflict among objectives exists.

Definition 2.6.6. Ideal point. The *ideal point* $y^I = (y_1^I, \dots, y_q^I)$ is given by

$$y_k^I = \min_{x \in X} f_k(x) = \min\ _{y \in Y} y_k. \tag{2.37}$$

Similarly, the *nadir point* is the solution vector which includes the worst value of all objective functions.

Definition 2.6.7. Nadir point $y^N = (y_1^I, \dots, y_q^I)$ is given by

$$y_k^N = \max_{x \in X_E} f_k(x) = \max\ _{y \in Y_N} x_k. \tag{2.38}$$

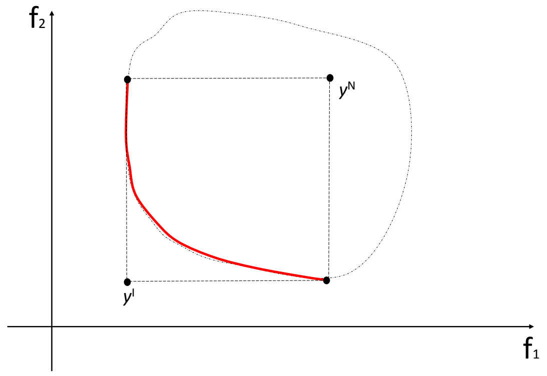


Figure 2.12: Ideal point (y^I) and nadir point (y^N) depicted in a two-objective space. Both objectives are being minimized.

Figure 2.12 shows an example a bi-objective problem which aims to minimize both objectives. The ideal point is that in which both objectives simultaneously have the minimum value and the nadir point is that worst-case scenario. In general, the nadir point is not easy to obtain. Several heuristics have been proposed to estimate it (Bechikh et al., 2010; Deb and Miettinen, 2008, 2010; Deb et al., 2006; Korhonen et al., 1997; Metev and Vassilev, 2003), but one basic estimation of the nadir point is by using *pay-off tables*.

On a multi-objective problem which optimizes p objectives simultaneously, the pay-off table is obtained by solving each of those p objectives in a single-objective approach, each of them subject to the original set of constraints of the problem. The ideal point is formed by all those optimal solutions, $y_k^I, = \min_{x \in X} f_k(x^k), \forall k = 1, \ldots, p$. By having the ideal point, the pay-off table is computed by evaluating each optimal solution on each objective. These evaluations can be shown in a table in which the diagonal is formed by the ideal point previously found (Table 2.1).

Table 2.1: Pay-off table obtained from the ideal point.

	x^1	x^2	$\ldots$	x^{p-1}	x^p
f_1	$f_1(x^1) = y_1^I$	$f_1(x^2)$	$\ldots$	$f_1(x^{p-1})$	$f_1(x^p)$
f_2	$f_2(x^1)$	$f_2(x^2) = y_2^I$	$\ldots$	$f_2(x^{p-1})$	$f_2(x^p)$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
f_{p-1}	$f_{p-1}(x^1)$	$f_{p-1}(x^2)$	$\ldots$	$f_{p-1}(x^{p-1}) = y_{p-1}^I$	$f_{p-1}(x^p)$
f_p	$f_p(x^1)$	$f_p(x^2)$	$\ldots$	$f_p(x^{p-1})$	$f_p(x^p) = y_p^I$

In this dissertation, we employ a pay-off setting to establish a range in which the Pareto front is located and find each point belonging to it. Multi-objective optimization has several techniques to obtain a front (Ehrgott, 2005). Classic methods are

- Weighted sum: transform the set of objectives into a weighted sum (Ehrgott, 2005), where each weight denotes the importance of each criterion to optimize (Athan and Papalambros, 1996; Kim and de Weck, 2005; Marler and Arora, 2010; Smith et al., 1982).

- ε-constraint: solve one objective subject to the original set of constraints plus the rest of the objectives constraint to bounds defined as ε (Aghaei et al., 2011; Bérubé et al., 2009; Laumanns et al., 2006; Mavrotas, 2009).

Heuristics and metaheuristics are also applied to obtain approximations to the Pareto front. Most approaches are iterative: only a single solution is obtained on each iteration and thus multiple runs are needed to find a front. An example of such techniques are genetic-based heuristics (Coello Coello, 1999; Fonseca and Fleming, 1995; Fonseca et al., 1993; Rey Horn et al., 1994; Zitzler, 1999; Zitzler and Thiele, 1998, 1999). The most commonly used metaheuristic is called NSGA-II (Deb et al., 2000).

2.7 Multi-objective distance-centered routing problems

We study the simultaneous minimization of the total travel distance and the total waiting time. In this section, each objective is studied in combination with others.

2.7.1 Multi-objective Traveling Salesman Problem

The Traveling Salesman Problem (TSP) has been extended with several additional objectives:

- Scheduling: to minimize the setup costs of a production line defined as a TSP tour (Tang et al., 2000).

- To minimize the traveled distance while maximizing the collected profits of a tour (Bérubé et al., 2009; Feillet et al., 2005; Jozefowiez et al., 2008).
- To minimize the total traveled distance and maximize the benefit obtained by the company has been studied in Angus (2007); Samanlioglu et al. (2008).
- To minimize the traveled cost and distance of a set of clients to be visited while minimizing the length of the tour (Elaoud et al., 2010; Jaszkiewicz and Zielniewicz, 2009; Paquete and Stützle, 2003, 2009; Paquete et al., 2004).
- To minimize the traveled distance and maximize the equalization of the vehicle travel times (Sutcliffe and Board, 1990).
- To minimize the travel cost and distance of a set of clients to be visited (Lust and Teghem, 2010; Peng et al., 2009).

2.7.2 Multi-objective m-Traveling Salesman Problem

Similarly several multi-objective approaches exist for m-TSP:

- To minimize the weighted sum of the total traveling costs of all salesmen and the highest traveling cost of any single salesman (Shim et al., 2012a,b).
- To upgrade the production of iron and steel companies by formulating their production lines as m-TSP instances while minimizing setup costs simultaneously (Tang et al., 2000).

2.8 Multi-objective latency-centered routing problems

The minimization of the total waiting time of the clients combination with others in literature. This section introduces such multi-objective problems.

2.8.1 Multi-objective Minimum Latency Problem

Main applications of the Minimum Latency Problem MLP are focused on humanitarian logistics:

- Minimization of the sum of the waiting time of affected areas in a disaster while simultaneously minimizing the sum of the expected value and the variance of the total cost of the relief chain as studied in (Bozorgi-Amiri et al., 2013).
- Minimization of the sum of the total waiting time of disaster areas while measuring time of response, equity of the distribution or reliability and security of the operation tours (Vitoriano et al., 2011).
- Minimizing the total waiting time of an earthquake disaster place while optimizing emergency relief conditions (Najafi et al., 2013).

2.8.2 Multi-objective k-Minimum Latecy Problem

Research for multi-objective multi-vehicle minimization of latency includes emergency areas and the improvement of sustainability of companies:

- Minimize the total latency of a set of tours while minimizing costs and enviromental emissions while maximizing responsiveness of the network (Ebrahimi, 2018).
- Minimize the total cost of the routing while considering both: transportation costs and penalty costs of not delivering commodities to their destination while simultaneously minimizing the number and waiting time of injured people who have not been served (Tavakkoli-Moghaddam et al., 2016).

2.9 The simultaneous minimization of latency and distance in routing problems

Mono-objective decisions are taken by considering an objective function subject to a set of limiters in the resources to be considered. Multi-objective optimization takes several objectives and optimizes them simultaneously, all subject to a set of constraints to take into account. We present a bi-objective problem that arises in the context of logistic activities of distribution-and-service companies which specialize in customer service by attending requests of product delivery and/or maintenance services. We seek a trade-off between the company performing the service and the client receiving it, by simultaneously minimizing distance and latency of a tour, or a set of tours, under certain assumptions.

Chapter 3 introduces our bi-objective problem for a single tour whereas Chapter 4 shows our bi-objective problem for a fleet of vehicles.

Chapter 3

Minimum Latency-Distance Problem

This bi-objective problem arises in logistic activities of a company in charge of providing a set of clients with a service or a delivery. The company deals with two conflicting objectives: to visit all the clients by traveling the minimum distance to ensure the cost this tour represents for the company is the minimum, as well as the maximization of the clients satisfaction, on terms of the minimization of the latency of the tour, to ensure a competitive quality in the service provided to the clients. In this chapter we introduce our bi-objective proposal, the assumptions we make to formulate it mathematically, as well as the solution methods we propose to obtain Pareto fronts for it.

Section 3.1 introduces our proposed problem, Section 3.2 shows the mathematical formulation as well as the assumptions considered for it. The computational complexity of our proposed problem is shown in Section 3.3, and solution methods are shown in Sections 3.4.1 – 3.4.4. Parameter calibration and computational experimentation are found in Section 3.5. Lastly, conclusions for this chapter are given in Section 3.6.

3.1 Minimum Latency-Distance Problem: an introduction

The Traveling Salesman Problem (TSP) is a classic combinatorial problem that aims to minimize the total travel distance of a vehicle in charge of visiting a set of clients (Applegate et al., 2007; Gutin and Punnen, 2006; Laporte, 1992). There is also a growing interest in studying the minimization of the waiting time (latency) of all clients to be visited in a single-vehicle tour (Blum et al., 1994; Chaudhuri et al., 2003; García et al., 2002; Lucena, 1990). Recently, the minimization of latency has been studied particularly in humanitarian logistics to minimize the waiting time of disaster areas to be provided with basic necessities in the shortest possible time (Ferrer et al., 2016; Kovács and Spens, 2007, 2009; Stephenson, 2017; Tomasini et al., 2009; Vargas et al., 2017). As a consequence of the literal definition of latency[1], the study of the minimization of the waiting time can also be found in other areas such as Medicine (Darcis et al., 2017; Littner et al., 2005; Paniagua-Soto et al., 2013; Squires et al., 1975; Sternberg et al., 1978) and Telecommunications (Crowcroft et al., 2015; Grout et al., 2007; Gummadi et al., 2002; Joo Ghee et al., 2009; Kao et al., 2017; Lichtsteiner et al., 2008; Lu et al., 2007; Schurgers et al., 2002).

To define our bi-objective approach, let us consider a company that provides a service to a set of clients. Such company has only a single agent to fulfil the visit. This agent departs from a known depot and returns to it at the end of the day after visiting each of the clients exactly once. The company goal is to satisfy the requests of the clients as quickly to minimize the total waiting time over the client set, while the agent travels a short distance to perform the visiting. With such delimiters, we propose a bi-objective problem called the *Minimum Latency-Distance Problem*

[1]Delay between the moment a request for an action is made and the moment that action is executed. Definition according to `https://www.collinsdictionary.com/dictionary/english/latent-time`

(MLDP), to simultaneously optimize both objectives.

MLDP arises in the context of logistic activities of companies that offer delivery or maintenance services. Its essence is the simultaneous optimization of a combination of two classic routing objectives that have not been previously studied together in a routing context. By optimizing both, the total travel distance as well as the total waiting time, MLDP ensures a competitive quality in the service provided to the clients and by definition, satisfies all conditions of TSP and MLP.

The objectives pursued by MLDP are considered relevant in contexts where both client service and company profit are priorities. It takes into account distance-dependent quantities such as fuel cost simultaneously with quality-of-service aspects such as the waiting time of the clients. In other words, MLDP takes into consideration an economic objective to preserve the profit of the company as well as a customer-centered objective to improve the quality of the service by minimizing the total waiting time of each client.

To the best of out knowledge, a bi-objective approach of this type has not been studied in a routing context, although the combination of objectives has been applied in several medical (Bair et al., 2003; Fischer et al., 1987; Salami et al., 2003) and network applications (Borah et al., 2017; Depuy et al., 2001; He et al., 2015; Kim and Na, 2017; Madhyastha et al., 2006; Patterson, 2004; Sarddar et al., 2010, 2011), both aiming to measure a specific growth depending on the distance it is measured from.

To consider these two objectives under a routing context helps achieving a more client-centered approach to classical routing. The main motivation of such combination of objectives is to provide an equilibrium between two important stakeholders in the decision-making process: the client and the company. This bi-objective approach aims to improve several aspects in the classical routing approach that does not incorporate a-quality-of-the-service factor taking it into consideration helps to improve the competitive position of the company as well as the client satisfaction (Clark et al., 1992; Oliver, 1987).

In Section 2.4, differences between both objectives were discussed. Section 3.2 shows how we deal with these differences and the assumptions made to formulate this problem mathematically.

3.2 MLDP: assumptions and mathematical formulation

To formally define the MLDP, let us consider a company that provides a service to a set of clients through a single agent. This agent departs from a known depot and returns to it at the end of the day, visiting each of the clients exactly once. The main goal of this single-vehicle tour is the minimization of the total travel distance of the agent together with the minimization of the total waiting time of the clients waiting to be served. As established in Sections 2.2 and 2.4 one of the differences in the latency and distance objectives is the inclusion of *service times* when computing latency.

To obtain a formulation for the MLDP, we assume that the duration of each service, the exact distance between the depot and the clients, as well as the distance between all the clients are known with certainty. The direct proportionality of travel time to distance traveled as well as the sufficient capacity of the agent to attend all the clients in a single tour are assumed as well. Note that because of the assumption of linearity between travel time and travel distance, we refer to the two indistinctly in our formulation.

As mentioned in Section 3.1, MLDP deals with the optimization of two well-known routing objectives that have several differences, one being that the two objectives are measured with a different metric (Blum et al., 1994). In MLDP, these differences are considered when computing the two objective values. Figure 3.1 shows a graphical representation of a single-vehicle tour in charge of visiting a set of ten vertices by minimizing the total latency and distance of the tour. Each grey arrow depicts the possible connections among all clients and the depot. Similarly to Figure 2.6 in Section 2.4, when considering latency, no returning arc is included by MLP definition, and therefore the returning arc is depicted as a dashed line to show that it is only considered by MLDP in the distance objective by definition of TSP.

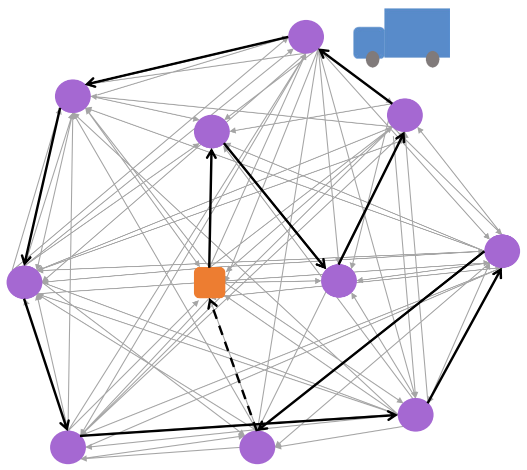


Figure 3.1: Graphical example of an small MLDP solution in an instance of ten vertices.

As mentioned in Sections 2.2 and 2.4, there exist several reported formulations in literature for both objectives (Angel-Bello et al., 2013; Gouveia and Voß, 1995; Laporte, 1992; Méndez-Díaz et al., 2008; Miller et al., 1960; Orman and Williams, 2007; Picard and Queyranne, 1978; Sarubbi et al., 2008), nonetheless Angel-Bello et al. (2013) state that it is inconvenient to employ a formulation designed for the minimization of the distance of a tour to handle an objective of minimizing the total latency along the tour.

To model MLDP, we part from a model developed for the objective of latency, into which we incorporate the distance objective. The best-performing models reported for latency minimization are those of Angel-Bello et al. (2013); we base our MLDP model on their reported "Model A".

Let us consider a directed and complete graph $G = (V, A)$. Let A be the arc set and V a vertex set $V = \{v_0, v_1, v_2, \ldots, v_n\}$, where v_0 represents the depot and the remaining vertices represent n clients. Each client $i \in I$ has associated a *service time* s_i, which refers to the time the agent takes to fulfil the service to the client. Each arc $(v_i, v_j) \in A$ has associated a *travel time* from client i to j, defined as t_{ij}, which represents the distance from client i to client j by considering the distance and travel time linearity assumption discussed before.

By defining a matrix $\mathbf{T}$ that contains for each pair (v_i, v_j) a weight $t_{i,j} \geq 0$ as the travel time from vertex v_i to vertex v_j, and where the diagonal element $t_{i,i}$ represents the service time s_i of client i, a *cost matrix* $\mathbf{C}$ is computed as follows:

$$c_{i,j} = t_{i,i} + t_{i,j}. \tag{3.1}$$

To formulate MLDP, similar to how Angel-Bello et al. (2013) formulate MLP, our mathematical model is based on a multi-level network with $n + 1$ levels inspired by one previously proposed by Picard and Queyranne in 1978, which was designed for a time-dependent travelling salesman problem. Such a network is depicted in figure 3.2 and it is helpful in representing an MLDP tour but with the exception of the returning arc traveled by the agent which, by definition of TSP, is only considered in the calculations of the total traveled distance for the tour.

In this multi-level network, on each level $\ell_i, i > 0$, all vertices v_i are represented. Note that an MLDP solution is a permutation of the n clients with the depot in the first position, like the solutions of the TSP and MLP. Consequently, any solution can

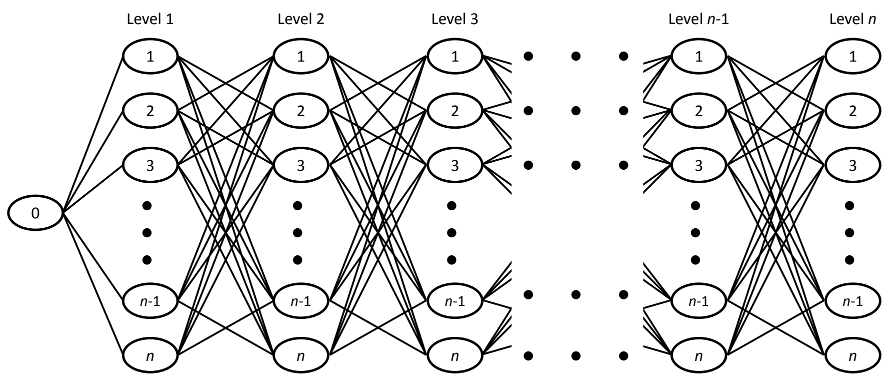


Figure 3.2: Multi-level network used to formulate the MLDP.

be represented on this multilevel network having a single vertex active in each level, with no repetitions between levels. These facts entail to define the decision variables $x_{i,k}$ and $y_{i,j}(k)$ as

$$x_{i,k} = \begin{cases} 1, & \text{if vertex } i \text{ is selected on level } k \text{ on the network,} \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{i,j}(k) = \begin{cases} 1, & \text{if vertex } j \text{ on level } k+1 \text{ follows vertex } i \text{ on level } k, \\ 0, & \text{otherwise.} \end{cases}$$

MLDP is linearly formulated as follows,

$$\min\ F_1 = n\sum_{i=1}^{n} c_{0i}x_{i,1} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(n-k)c_{ij}y_{i,j}(k) \tag{3.2}$$

$$\min\ F_2 = \sum_{i=1}^{n} c_{0i}x_{i,1} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}y_{i,j}(k) + \sum_{i=1}^{n}c_{i0}x_{i,n} \tag{3.3}$$

subject to

$$\sum_{k=1}^{n} x_{i,k} = 1,\ i = 1, 2, \ldots, n \tag{3.4}$$

$$\sum_{i=1}^{n} x_{i,k} = 1,\ k = 1, 2, \ldots, n \tag{3.5}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} y_{i,j}(k) = x_{i,k},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1 \tag{3.6}$$

$$\sum_{\substack{j=1 \\ j \neq i}}^{n} y_{j,i}(k) = x_{i,k+1},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n-1 \tag{3.7}$$

$$x_{i,k} \in \{0, 1\},\ i = 1, 2, \ldots, n,\ k = 1, 2, \ldots, n \tag{3.8}$$

$$y_{i,j}(k) \geq 0,\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, n, j \neq i,\ k = 1, 2, \ldots, n. \tag{3.9}$$

The minimization of the total waiting time of the clients is defined by Equation (3.2). In Equation (3.3) lays the minimization of the total traveled distance and therefore, the minimization of the cost this tour represents to the company. Note that in Equation (3.2) the arc denoting the return of the vehicle to the depot is not considered, meanwhile this returning arc has to be considered on Equation (3.3). The reason for this is that we do not consider the returning of the vehicle to the depot as client's waiting time.

Equation (3.4) guarantees that each vertex occupies a single position in any feasible solution. Equation (3.5) guarantees that each position is occupied by no more than one vertex in any feasible solution. Equation (3.6) ensures that only one arc leaves from position k, exactly from the vertex taking that position and Equation (3.7) imposes that only one arc at a time can arrive to position $k + 1$, exactly to the vertex occupying that position. The nature of Equations (3.4) to (3.7) lays in that all possible solutions for the MLDP can be represented on the previously defined multi-level network.

Lastly, Equations (3.8) and (3.9) correspond to the nature of the variables. Note that variables $y_{i,j}(k)$ are previously defined as binary but they are handled as continuous variables in the model. This is achieved by Equations (3.6) and (3.7), which guarantee that only one arc leaves from position k and arrives exactly to the vertex occupying the position $k+1$. These pair of constraints assures $y_{i,j}(k)$ takes a value of one or a value of zero. This formulation consists of n^2 binary variables, $n^3 - 2n^2 + n$ real variables, and $2n^2$ constraints.

This formulation can be expressed in non-linear terms, under the same parameters and in the same multi-level network. This is achieved by representing the selection of vertices over the network with boolean decision variables $x_{i,k}$,

$$x_{i,k} = \begin{cases} 1, & \text{if } v_i \text{ is selected on } \ell_k, \\ 0, & \text{otherwise,} \end{cases} \tag{3.10}$$

where both i and k take values from $1, \ldots, n$. Note that the depot v_0 forms ℓ_0, meaning no selection is made there as every feasible MLDP tour initiates at the depot. Additionally, we write as an auxiliary notation

$$f_{i,j}(k) = x_{i,k} x_{j,k+1}, \tag{3.11}$$

meaning that $f_{i,j}(k) = 1$ if and only if v_i is selected on ℓ_i and v_j is selected on ℓ_{i+1}, and zero otherwise. With the above notation, the travel distance from the depot to the first client is defined as

$$\mathcal{T}_0 = \sum_{i=1}^{n} c_{0,i} x_{i,1}, \tag{3.12}$$

the travel distance from the last client to the depot is defined as

$$\mathcal{T}_n = \sum_{i=1}^{n} c_{i,0} x_{i,n}. \tag{3.13}$$

Similarly, as each of all the n clients will have to wait for the agent to reach the first client, we write

$$\mathcal{W}_0 = n \sum_{i=1}^{n} c_{0,i} x_{i,1}. \tag{3.14}$$

The complete MLDP formulation is defined as follows,

$$\min\ F_1 = \mathcal{W} = \mathcal{W}_0 + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}(n-k)c_{i,j}f_{i,j}(k) \tag{3.15}$$

$$\min\ F_2 = \mathcal{T} = \mathcal{T}_0 + \mathcal{T}_n + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{j=1}^{n}c_{i,j}f_{i,j}(k) \tag{3.16}$$

subject to

$$\forall k : \sum_{i=1}^{n} x_{i,k} = 1, \tag{3.17}$$

$$\forall i : \sum_{k=1}^{n} x_{i,k} = 1, \tag{3.18}$$

$$x_{i,k} \in \{0,1\}, \forall i, \forall k. \tag{3.19}$$

In this non-linear formulation, due to the introduction of the multiplicative terms $f_{i,j}(k)$, Equation (3.15) expresses the objective to minimize the total waiting time over the set of clients, as each client has to wait through the services of all the previous clients and the intermediate travel times. Equation (3.16), denotes the minimization of total travel distance, from the depot and back to the depot, after visiting all the clients. Equation (3.17) ensures that only one vertex is selected per level in such network and Equation (3.18) makes sure that each vertex is selected exactly once in the network. Lastly, Equation (3.19) denotes the nature of the variables.

Note that the depot is *not* waiting for a service and that the time it takes to return from the last client to the depot is not considered part of the total latency of the clients, although it is included in the calculation of the total travel distance in Equation (3.16). By denoting the resulting $n \times n$ matrix of decision variables by $\mathbf{X}$, and in terms of the above definitions, MLDP can be stated as finding an assignment to $\mathbf{X}$ that minimizes both Equations (3.16) and (3.15) and satisfies both Equations

(3.17) and (3.18). This non-linear formulation consists of $\mathcal{O}(n^2)$ decision variables and $\mathcal{O}(n)$ constraints.

Section 3.3 analyzes the computational complexity of our proposed problem, and Sections 3.4.1 – 3.4.4 describe our solution approaches for it.

3.3 Complexity of MLDP

In this section, we study the computational complexity of the MLDP. We begin with the description of the required definitions, along with the description of several special cases of this problem.

According to Garey and Johnson (1990), a *problem*, which usually possesses several parameters or variables with unspecified values, is a general question to be answered. A problem is described by two properties:

1. a general description of all the parameters belonging to the problem,
2. the properties a solution must satisfy to answer the problem.

A *instance* of a problem is a set of specific values of all parameters describing a problem and an *algorithm* is an step-by-step procedure to solve a problem, the most desirable outcome is finding the most efficient algorithm for solving a problem (Garey and Johnson, 1990).

Computational theory is an area of study that aims to classify problems according to their inherent difficulty. A problem is classified as inherently difficult if no matter what algorithm is used to solve it; its solution requires a significant amount of resources to obtain a solution. The idea that a problems *seem* difficult is not enough, a theoretic proof should exist. This is what a computational analysis is for: to formalize the idea of difficult-to-solve by proving that a mathematical model of the problem requires such an amount of resources to solve that it becomes impractical. Analysis of the complexity of a problem should not be confused with an analysis of algorithms, despite these two being closely related (Garey and Johnson, 1990; Papadimitriou, 1994): classifying a problem as difficult means it cannot be solved efficiently with restricted resources no matter which algorithm is used.

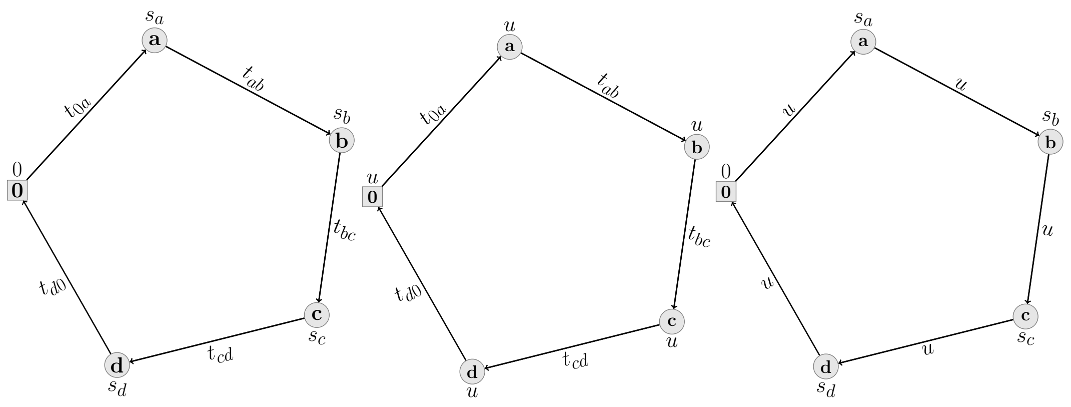
In general, the goal of an optimization problem O_σ is to find the best solution according to a set of constraints that describes the environment of the problem. Each optimization problem is associated to a decision problem D_σ which differs from O_σ in the sense that the objective is treated as a constraint. Given a bound B for the constraint representing the objective of O_σ, the answer to the decision problem is whether or not an assignment exists that satisfies both the original constraints and the new constraint with the bound B. Hence the decision problem D_σ has only two possible outcomes: yes or no. If the decision problem D_σ is proven NP-complete, then the corresponding optimization problem O_σ is NP-hard (Garey and Johnson, 1990; Papadimitriou, 1994). The theory of NP-completeness is designed to be applied to decision problems of optimization problems.

A *complexity class* is defined by several parameters such as non-determinism and restrictions on the amount of computational resources available (namely time and space) (Papadimitriou, 1994). NP stands for *non-deterministic polynomial time* meaning that the class is defined by bounding the execution time polynomially under non-deterministic computation: given an input for a problem D_σ, an “oracle” can guess a correct solution in polynomial time. To prove the inclusion of any problem D_σ in the class NP it suffices to demonstrate that any given solution of D_σ can be verified as a valid solution for D_σ in polynomial time. Both of the single-objective optimization problems that are merged into MLDP are NP-hard (Afrati et al., 1986; Papadimitriou, 1994). In this section, we demonstrate that also MLDP itself is NP-hard.

To demonstrate the complexity of MLDP, the demonstration is subject to the same assumptions as the model presented in the previous section: we consider one single agent with no capacity restrictions that visits all of the clients in a single tour. This requires efficiently reducing a known NP-complete problem D_σ to our problem (Garey and Johnson, 1990). We first define the decision problem D_{MLDP} associated to MLDP, then show that MLDP belongs to NP, and finally establish that an efficient reduction from a known NP-complete problem to our problem exists.

By definition, the MLDP consists in finding a tour which visits all clients, leaves from an established depot and returns to it, while minimizing the total latency of all the clients and the total traveled distance of an uncapacitated vehicle. To prove the complexity of MLDP, several cases were revised to prove that complexity holds in several "easy" instances, Figure 3.3 depicts some particular cases in which the complexity for MLDP holds. Nonetheless the special case when constant travel times and constant service times exist is trivial, and not shown in Figure 3.3, as in this case any visit order would be optimal. The general case, which refers to a more real-life situation for the agent, considers arbitrary non-zero traveling and service times, (see Figure 3.3a). We also consider some special cases of this problem (see Figures 3.3b, 3.3c) to ensure that in simpler and more arbitrary cases the complexity holds.

Note that in the case depicted in Figure 3.3b, when service times are a constant but travel times are arbitrary non-negative values, the problem is NP-hard as it is a TSP instance. The case depicted in Figure 3.3c, in which travel times are a constant but service times are arbitrary, is NP-hard as it is an MLP instance. In this particular study we present the general case with non-zero and arbitrary values for both travel and service times (see Figure 3.3a). In a previous study, Nucamendi-Guillén et al. (2016) discuss latency objective and propose Equation (3.20), where t_{ij} refers to



(a) arbitrary service and travel times (b) constant service times, arbitrary travel times (c) arbitrary service times, constant travel times

Figure 3.3: Illustration of some particular instances of the MLDP, which were studied in the complexity proof. In Figure 3.3a, the general case is shown: in this case, service and travel times take all different arbitrary values. In Figure 3.3b, a simpler case is shown: all service times have constant values u, while all travel times are arbitrary. This particular case is a TSP instance. Lastly, in Figure 3.3c, another simpler case is shown: all service times are arbitrary and all travel times are equal to a constant u. This particular case is an MLP instance.

travel time among clients i and j, as well as s_i and s_j refers to their corresponding service times.

$$\frac{1}{2}\sum_{k=1}^{n-1}(n-k)\sum_{i=0}^{n}\sum_{j=0}^{n} t_{ij} + (s_i + s_j). \tag{3.20}$$

By using Equation (3.20), we redefine non-zero travel and service times for computing the bounds used to define the decision problem D_{MLDP} associated to MLDP.

The corresponding decision problem of MLDP is the following: given the costs of Equation (3.1), an upper bound $\mathcal{T} \leq \Theta$ to to the total travel time of Equation (3.3), and an upper bound $\mathcal{W} \leq \Omega$ to the total latency of Equation (3.2), is there an assignment $\mathbf{X}$ that satisfies all the original constraints as well as the upper bounds set on the objectives?

As stated by Garey and Johnson (1990), in order to show that MLDP belongs to NP it is necessary to establish that an efficient reduction from a known NP-complete problem to our problem exists. Theorem 1 proves the reduction of MLDP to TSP.

Theorem 1. TSP *reduces efficiently to* MLDP.

Proof. We first establish that the feasibility of a given $\mathbf{X}$ can be verified in polynomial-time: each $\mathbf{X}$ captures a permutation that starts at the depot and the permutation can be efficiently recovered from the assignment matrix $\mathbf{X}$ as a visit sequence $v^{(0)}, v^{(1)}, \ldots, v^{(n)}$ where $v^{(0)}$ is always the depot and the visits from v^1 to $v^{(n)}$ correspond to the clients. We need three accumulator variables, one to measure time along the tour, one to count the total travel time, and the third to count the total latency. We also need an array of n binary variables, one per each client, to verify that each one is properly visited. We proceed in the visiting order, denoting the source by i and the destination by j adding the value of $t_{i,j}$ to both the time accumulator and the travel-time accumulator, then adding the service time $t_{i,i}$ to the time accumulator, and then the current value of the time accumulator to the latency accumulator, marking the client j as visited. If at the end of the permutation, all n binary variables are one, the travel-time accumulator respects its upper bound Θ, and the latency accumulator respects its upper bound Ω, the solution is feasible. As the verification is a polynomial procedure, $D_{\text{MLDP}} \in \mathsf{NP}$.

To establish NP-completeness, we reduce the Traveling Salesman Problem (TSP) in its decision version to D_{MLDP}, as illustrated in Figure 3.4. In the decision version of TSP, the input is a cost matrix $\mathbf{C}$ for the travel costs between n clients (with the diagonal elements being zero) together with an upper bound to the total cost of the tour, $\mathcal{C}$, and the question is whether a permutation over the set of clients exists that produces a tour with the sum of costs of the segments not

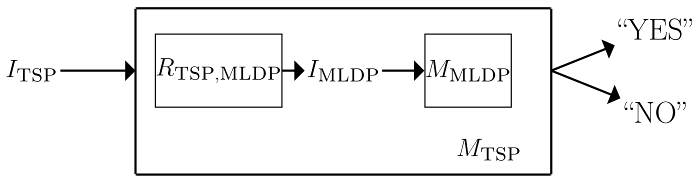


Figure 3.4: Reduction diagram from TSP to MLDP (both as decision problems). The main box in the figure depicts an algorithm to solve the TSP problem that takes as input an instance of TSP and determines whether the answer is "yes" or "no" to this instance. To prove the complexity of MLDP, a reduction algorithm must transform a TSP entry into a MLDP one. This MLDP input is then solved and receives a "yes" or "no" answer. As a whole, the process takes a TSP instance, reduces it into an MLDP instance, solves the reduced instance with an algorithm for MLDP, and responds correctly "yes" or "no" to the original instance.

exceeding $\mathcal{C}$.

In order to transform the input for the decision version of the TSP into an input for D_{MLDP}, we will use the elements of $\mathbf{C}$ as $\mathcal{T}$. As $\mathbf{C}$ indicates all travel times among the clients, by considering all service times as zero we remain with $\mathcal{T} = \mathbf{C}$. We set $\Theta = \mathcal{C}$, and only need to compute efficiently an adequate value for the latency bound Ω. We do this in terms of the worst-case costs, computing from the cost matrix $\mathbf{C}$ the largest cost per each row (that is, as if the agent at each client chose the most expensive segment to continue the tour) in $\mathcal{O}(n^2)$ time, and then sort these worst-case segments from largest to smallest in $\mathcal{O}(n \log n)$ time to construct an upper bound to the worst-case latency (the service times in TSP are the diagonal elements of the cost matrix that are all zero): the worst-case total latency is the sum of the cumulative sums of the sorted list of costs and we use this as Ω. Hence, D_{MLDP} responds "yes" to the transformed input if and only if the decision version of TSP would respond "yes" for the original input. $\square$

As stated before, this demonstration is subject to the same assumptions considered in the mathematical model presented in Section 3.2: we consider one single agent with no capacity restrictions that visits all of the clients in a single tour. This requires efficiently reducing a known NP-complete problem D_σ to our problem (Garey and Johnson, 1990). We first define the decision problem D_{MLDP} associated to MLDP, then show that MLDP belongs to NP, and finally establish that an efficient reduction from a known NP-complete problem to our problem exists. After proving that MLDP reduces efficiently to TSP, Theorem 2 discusses the complexity proof of MLDP.

Theorem 2. D_{MLDP} *is* NP-*complete.*

Corollary 2.1. MLDP *is* NP-*hard.*

Proof. By Theorem 1 and Corollary 2.1, it is proven that MLDP is at least as difficult as TSP. Thus, as stated in Corollary 2.1, D_{MLDP} is NP-complete and therefore MLDP is NP-hard. □

We have shown that our bi-objective problem is difficult to solve efficiently with exact algorithms. By establishing its complexity we are therefore justified to propose solving methods, not guaranteed to be optimal, to obtain solutions with less computation time. The following sections discuss our proposed algorithms for MLDP.

3.4 Solution methods for MLDP

MLDP is an NP-hard problem and therefore exact methods are only expected to solve very small instances efficiently, making heuristic approaches a necessity. In this section we discuss the design and implementation of our proposed procedures to obtain Pareto fronts for a set of MLDP instances.

Section 3.4.1 discusses obtaining exact MLDP Pareto fronts of these instances, whereas Sections 3.4.2, 3.4.3, and 3.4.4 describe three heuristic approaches to obtain approximations of the Pareto fronts.

3.4.1 Exact method

MLDP, as previously defined, is a problem which deals with the minimization of two objective functions: latency and distance of a tour. As shown in Section 2.6, the solution methods for multi-objective problems are different from solution methods for obtaining a single-objective optimal solution.

A well-known classic technique to obtain an exact solution of multi-objective problems is the ϵ-constraint method (Ehrgott, 2005), as mentioned in Section 2.6. Nonetheless, the ϵ-constraint has some aspects to improve. Mavrotas (2009) developed a technique called Augmecon which guarantees Pareto optimality in the pay-off table of the obtained solution, as well as in the generation process, parting from the idea of the classic ϵ-constraint to constraint one objective to the rest of them. In particular, Mavrotas and Florios (2013) developed an improvement of Augmecon called *Augmecon2*, and while it keeps the same idea of solving a single-objective problem constrained to the rest of the objectives, it exploits the information from the slack variables in every iteration, leading to a significant reduction on computation time as well as the avoidance of redundant iterations with the goal of performing an iterative lexicographic optimization in the objectives. Augmecon treats all the constrained objectives similarly, while Augmecon2 applies them in an order of importance. The general model for this technique is:

$$\begin{aligned}
&\text{Optimize } f_1(x) + \epsilon(S_2/r_2 + 10^{-1} \times S_3/r_3 + \ldots + 10^{-(p-2)} \times S_w/r_w) \\
&\text{subject to} \\
&\qquad f_2(x) - S_2 = e_2 \\
&\qquad f_3(x) - S_3 = e_3 \\
&\qquad \vdots \\
&\qquad f_w(x) - S_w = e_w \\
&\qquad x \in S, S_i \in \mathbb{R}^+.
\end{aligned} \tag{3.21}$$

In this model, in each iteration, $e_2, \ldots, e_w$ are the right-hand side parameters from the grid points of the w objective functions to optimize. Each of these objectives has a range that denotes the number of equidistant grid points it solves. In Equation (3.21), ranges are denoted by $r_2, \ldots, r_w$, surplus variables are denoted by $S_2, \ldots, S_w$, and $\epsilon \in [10^{-6}, 10^{-3}]$.

For our implementation of this algorithm, designed to obtain exact solutions for MLDP, the design was formulated for our linear formulation described in Section 3.2. We keep the distance objective of Equation (3.3) as the main objective and solve the problem subject to the original constraints along with the latency objective of Equation (3.2); in our case, this produces a single-objective problem which minimizes the distance of the tour constrained to its total latency.

The order of the objectives could be reversed, but according to Angel-Bello et al. (2013); Blum et al. (1994), as well as the complexity analysis of the latency objective (Afrati et al., 1986), the alternative is expected to be more time consuming. As ϵ-constraint, Augmecon2 uses a step size to construct a grid and to find the Pareto points. In our case, we define this step size as $\delta = 1/2n$, where n denotes the instance size. Note that this method is able to find optimal solutions, in terms of efficiency, but it may not find non-supported points in the Pareto front; in other words, this technique may not find *all* the points in the true Pareto front.

As both of the single-objective problems that MLDP may reduce to in this manner are also NP-complete (Afrati et al., 1986; Papadimitriou, 1994), the asymptotic complexity of computing a Pareto front with an ϵ-constraint is exponential in the size of the instance, which in this case is measured in terms of the number of clients. Therefore, a metaheuristic approach is justified and desirable. Sections 3.4.2, 3.4.3 and 3.4.4 describe three metaheuristic algorithms to solve this bi-objective problem.

3.4.2 Strategic Memetic Search Algorithm for MLDP

MLDP is an NP-hard problem. Therefore, a heuristic approach is justified and desirable. This section introduces an efficient but heuristic alternative we designed and implemented: a *memetic algorithm* to approximate the Pareto fronts for larger instances. An algorithm of this type is an evolutionary, population-based approach that employs local improvement procedures as mutation processes.

We propose an elitist multi-objective evolutionary algorithm that involves using *crowding distance* of a solution that estimates the density of solutions in the front surrounding that solution as a means of promoting diversity. This algorithm is based in the classic NSGA-II algorithm, proposed by Deb et al. (2000). In the following subsections, each step and the adaptations made in the design of this strategic memetic search algorithm, hereinafter called as SMSA, are explained in detail. The general SMSA algorithm is in Algorithm 1.

Algorithm 1 Strategic Memetic Search Algorithm (SMSA).

Input: integers $\gamma > 0$ and $\chi_{\max} > 0$.

Output: a set of solutions P.

```
1: $\chi := 0$
2: generate a population $P$ of size $\gamma$ (Section 3.4.2.1)
3: repeat
4:     generate a set of offsprings $Q$ of size $\gamma$ from $P$ (Section 3.4.2.2)
5:     rank $P \cup Q$ into fronts $F_0, F_1, \ldots$ (Section 3.4.2.3)
6:     $P' = \emptyset$
7:     for each front $F_i$ in increasing order of $i$ do
8:         for each solution $s \in F_i$ do
9:             with probability $\beta$, mutate $s$ with Algorithm 2 (Section 3.4.2.4)
10:        end for
11:        $P' := P' \cup F_i$
12:        if $|P'| > \gamma$ then
13:            break the loop
14:        end if
15:    end for
16:    $P^* :=$ the best $\gamma$ solutions in $P'$ by crowding distance
17:    if $P = P^*$ then
18:        $\chi := \chi + 1$ (no change in the population)
19:    else
20:        $\chi := 0$
21:    end if
22:    $P := P^*$
23: until $\chi = \chi_{\max}$
24: return $P$
```

3.4.2.1 Initial population

In our SMSA implementation, the generation of the initial population of size γ (step two) consists of 60% of random solutions and 40% of solutions obtained using the *cheapest insertion* heuristic (Rosenkrantz et al., 1974). From the latter, half of them are obtained using a distance metric and half using a latency metric.

3.4.2.2 Offsprings

In our implementation, the selection of each parent is made through a *binary tournament*. After selecting both parents, we apply a combination method based on the *OX crossover* introduced by Davis (1985), described in detail by Oliver et al. (1987). Prins (2004) modified this method so that the combination of the parents produces up to eight different offsprings; we repeatedly apply this modified version until a desired number of unique solutions are obtained.

3.4.2.3 Ranking

The *ranking process* used in SMSA is that described by Deb et al. (2000): in order to identify a dominating solution, it is iteratively compared to the other solutions to identify which ones it dominates, storing for each solution the set of solutions that dominate it as well as the set of solutions that it dominates. These two sets facilitate the placement of each solution into fronts F_h such that all solutions in front F_h dominate those in F_{h+1}. Each front is then internally sorted by dispersion, from best to worst. This procedure has asymptotic complexity in $\mathcal{O}(w\gamma^3)$, where w refers to the number of objectives involved and γ refers to the size of the population (Deb et al., 2000).

3.4.2.4 Neighborhood exploration

SMSA is a memetic algorithm, meaning that the mutation process is based on a local search. Each solution in the population is mutated with probability β, the value of which is a parameter.

We use a 2-OPT neighborhood (Croes, 1958): given a solution tour, select a random non-empty segment of the tour, reverse and reinsert it into the tour in *reverse* order. Having performed a 2-OPT movement, we compute the fitness of the neighboring solution thus created. We alternate between the two objective functions, using Equation (3.3) to compute fitness $f_i(\circ)$ when $i = 0$ in Algorithm 2), and Equation (3.2) when $i = 1$. Each objective is used for I consecutive iterations, after which the objective function to use as $f_i(\circ)$ is switched. This process is repeated R times. We make no attempt to save time by making adjustments to the fitness of the original solution, but instead re-evaluate the objective function entirely when computing the fitness of a neighbor.

Algorithm 2 SMSA mutation process.

INPUT: a solution s; integers $R > 0$ and $I > 0$.

OUTPUT: a (possibly mutated) solution s.

```
1: select $i \in \{0, 1\}$ uniformly at random
2: for $R$ repetitions do
3:     for $I$ iterations do
4:         $s' := 2\text{-OPT}(s)$
5:         if $f_i(s') \leq f_i(s)$ then
6:             $s := s'$ (keep the better one)
7:         end if
8:     end for
9:     if $i = 0$ then
10:        $i := 1$ (switch to $f_1$)
11:    else
12:        $i := 0$ (switch to $f_0$)
13:    end if
14: end for
15: return $s$
```

Section 3.5.4 discusses the calibration of parameters mentioned in this section. Section 3.5.5 discusses and compares results obtained with our proposed heuristic.

3.4.3 Evolutionary algorithm with Intelligent Local Search Algorithm for MLDP

Due to the complexity of MLDP, it is desirable to approximate Pareto fronts. In addition to the memetic algorithm of Section 3.4.2, we propose a second evolutionary algorithm to approximate the Pareto fronts.

We call this algorithm as *Evolutionary algorithm with Intelligent Local Search* (EiLS). It is based on the Pareto ranking scheme with crowding distance proposed by Deb et al. (2000) in their NSGA-II procedure but uses an efficient random-shake crossover to produce offsprings and an intelligent local search over various neighborhoods as a substitute for mutation. A general overview of EiLS is given in Algorithm 3.

Algorithm 3 Evolutionary Algorithm with Intelligent Local Search (EiLS).

INPUT: integers $\gamma > 0$, $\varepsilon > 0$ and $\kappa > 0$, real-valued $\alpha > 0$.

OUTPUT: a set of solutions P.

```
1: $\forall h \in [1,12] : \mathcal{S}(h) := 0, \mathcal{I}(h) := 0, \mathcal{C}(h) := 0$ (Section 3.4.3.6)
2: generate an initial population $P$ of size $\gamma$ (Section 3.4.3.1)
3: repeat
4:     evaluate the fitness of each solution in $P$        }
5:     place the $\varepsilon$ best solutions into $\mathcal{E}$      } (Section 3.4.3.2)
6:     create $\mathcal{K} := \kappa$ couples from $\mathcal{E}$         }
7:     compute the *offsprings* $\mathcal{F}$ of the couples in $\mathcal{K}$ (Section 3.4.3.3)
8:     $P :=$ *mutations* done with Algorithm 5 on $P \cup \mathcal{F}$
9: until $\max_h \mathcal{S}(h) < \alpha$
10: return P
```

3.4.3.1 Initial population

By definition of MLDP, solutions are represented as a permutation of the clients $(\pi_1, \pi_2, \ldots, \pi_n)$, where π_i represents the ith client visited along the tour. All initial solutions are created generating random permutations of the n clients. The depot is not permuted but instead prepended to all permutations, hence producing only feasible solutions.

3.4.3.2 Elite set

We rank the solutions into fronts $F_0, F_1, \ldots$ as described in Section 3.4.2.3 and construct an *elite set* $\mathcal{E}$ containing the best ε solutions (i.e., starting with F_0, then continuing with F_1, and so forth). We then select κ couples of two distinct parents from $\mathcal{E}$ using binary tournament with replacement (i.e., a single solution may form part of multiple couples).

3.4.3.3 Offsprings

Each couple selected from the elite set $\mathcal{E}$ produces two offsprings. First, one of the parents is chosen uniformly at random (p_1) and copied to the first offspring (o_1). Once copied, we apply the following perturbation: the positions of two non-overlapping tour segments of size φ are exchanged in the tour; the value of $\varphi \in [2, \lfloor 0.3n \rfloor]$ is chosen uniformly at random (see Table 3.1). This procedure is then repeated *independently* for the second parent p_2 to produce a second offspring o_2.

Table 3.1: EiLS offspring generation with $n = 10$ and $\varphi = 3$. The exchanged segments are highlighted in grey.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
p_i	v_2	v_3	v_4	v_1	v_5	v_6	v_8	v_{10}	v_9	v_7
o_i	v_8	v_{10}	v_9	v_1	v_5	v_6	v_2	v_3	v_4	v_7

3.4.3.4 Mutation

Each solution in the current population undergoes an intelligent local search that locally optimizes a *compromise function* with random weights as proposed by Molina et al. (2007). The construction of this compromise function is described in Algorithm 4. Its purpose is to aggregate the original objectives using weights to diversify the resulting Pareto front.

Algorithm 4 EiLS compromise-function construction.

INPUT: none.

OUTPUT: an objective function.

```
1: select $t \in \{\text{True}, \text{False}\}$ uniformly at random
2: if $t = \text{True}$ then
3:     select uniformly at random one of the objective functions as such
4: else
5:     generate a random linear combination of the objective functions
6: end if
7: return an objective function (chosen or constructed)
```

3.4.3.5 Neighborhood exploration

We employ twelve different neighborhoods $\mathcal{N}_h$ for the mutation process in EiLS. Once a neighborhood is selected, it is explored until a maximum number of consecutive iterations with no improvement is reached. It is an adjustable parameter ϕ; we define this maximum as a multiple of γ.

Several neighborhood definitions are given in terms of *closeness* $\lfloor \xi \rceil$ that limits how many steps away a selected position may be from another position, counting the separation along the permutation Π that represents the solution, allowing moves both backward and forward. Experimental tuning of closeness showed no specific value to be clearly superior on all instances. Hence, instead of fixing it, we duplicate the neighborhood structure with two different values for $\lfloor \xi \rfloor$: $\mathcal{N}_3$ and $\mathcal{N}_4$ use $\xi = 0.5n$ whereas $\mathcal{N}_6$ and $\mathcal{N}_7$ use $\xi = 0.8n$.

- $\mathcal{N}_1$: Two positions a and b, $a \neq b$, are chosen uniformly at random and their contents are exchanged.
- $\mathcal{N}_2$: Three positions a, b, and c are chosen uniformly at random and their contents are exchanged by moving the client of a to b, that of b to c, and that of c to a (see Table 3.2).

Table 3.2: An example of $\mathcal{N}_2$ with $n = 10$ with $a = 1$, $b = 3$, and $c = 9$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
Π	v_7	v_8	v_6	v_5	v_1	v_2	v_3	v_4	v_{10}	v_9
$\mathcal{N}_2(\Pi)$	v_{10}	v_8	v_7	v_5	v_1	v_2	v_3	v_4	v_6	v_9

- $\mathcal{N}_3$: A position a is chosen uniformly at random and its client is exchanged with another position $b \neq a$ with $\xi = 0.5n$ (see Table 3.3).

Table 3.3: An example of $\mathcal{N}_3$ with $n = 10$ where $a = 3$, $\lfloor \xi \rfloor = 5$, and $b = 4$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
Π	v_2	v_23	v_4	v_1	v_5	v_6	v_8	v_{10}	v_9	v_7
$\mathcal{N}_3(\Pi)$	v_2	v_3	v_1	v_4	v_5	v_6	v_8	v_{10}	v_9	v_7

- $\mathcal{N}_4$: A position a is chosen uniformly at random, and then two other positions b and c are chosen (all three positions being distinct) are at most $\xi = 0.5n$

steps away from a. The positions of a, b, and c are exchanged using the same scheme as in $\mathcal{N}_2$ (see Table 3.4).

Table 3.4: An example of $\mathcal{N}_4$ with $n = 10$ where $a = 8$, $\lfloor \xi \rfloor = 4$, $b = 5$, and $c = 6$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
Π	v_8	v_7	v_6	v_5	v_9	v_1	v_2	v_3	v_4	v_{10}
$\mathcal{N}_4(\Pi)$	v_8	v_7	v_6	v_5	v_3	v_1	v_9	v_2	v_4	v_{10}

- $\mathcal{N}_5$: A position $a > 1$ is selected uniformly at random and its content is exchanged with the immediately preceding position in Π.
- $\mathcal{N}_6$: Selections and exchanges as in $\mathcal{N}_3$, but with $\xi = 0.8n$.
- $\mathcal{N}_7$: Selections and exchanges as in $\mathcal{N}_4$, but with $\xi = 0.8n$.
- $\mathcal{N}_8$: A position a is chosen uniformly at random among those with $\max c_{i,j}$ (Equation 3.1) where v_i is the client in the preceding position and v_j is the client in the position in question. Then, another position $b \neq a$ is chosen uniformly at random and its contents are exchanged with a (see Table 3.5).
- $\mathcal{N}_9$: A position a is chosen uniformly at random and its contents are exchanged with a position $b \neq a$ chosen uniformly at random such that $b > \frac{n}{2}$ (see Table 3.6).
- $\mathcal{N}_{10}$: A position $\frac{n}{2} < a < n$ is selected uniformly at random. Then, a position $b > n - a$ is chosen uniformly at random. The contents of a and b are exchanged (see Table 3.7).

Table 3.5: An example of $\mathcal{N}_8$ with $n = 10$, $a = 2$, and $b = 7$, showing all quantities involved in Equation (3.1), using integers for simplicity of the example.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
$t_{i,i}$	1	4	3	2	1	5	2	4	3	2
$t_{i,j}$	2	5	3	4	6	2	6	1	3	5
$c_{i,j}$	3	9	6	6	7	7	8	5	6	7
Π	v_2	v_3	v_4	v_1	v_5	v_6	v_8	v_{10}	v_9	v_7
$\mathcal{N}_8(\Pi)$	v_2	v_8	v_4	v_1	v_5	v_6	v_3	v_{10}	v_9	v_7

Table 3.6: An example of $\mathcal{N}_9$ with $n = 10$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
Π	v_2	v_3	v_4	v_1	v_5	6	v_8	v_{10}	v_9	v_7
$\mathcal{N}_9(\Pi)$	v_2	v_3	v_4	v_9	v_5	v_6	v_8	v_{10}	v_1	v_7

- $\mathcal{N}_{11}$: An integer $\lambda \in [2, \lfloor 0.3n \rfloor]$ is chosen uniformly at random. Then, two *non-overlapping* tour segments of length λ are chosen uniformly at random and their contents are exchanged (see Table 3.8).
- $\mathcal{N}_{12}$: First, a position a is chosen uniformly at random. Then a position b such that $a < b$ is chosen uniformly at random. The contents of a and b are exchanged.

3.4.3.6 Neighborhood performance

As proposed by Molina et al. (2018), the order in which the $\mathcal{N}_h$ neighborhoods

Algorithm 5 Intelligent neighborhood selection in EiLS.

INPUT: a set of solutions P; integers $\phi > 0$ and $\tau > 0$.

OUTPUT: a (possibly improved) set of solutions P.

```
1: for each solution $s \in P$ do
2:     repeat
3:         $\eta := 0$ (a stall counter)
4:         select a neighborhood $\mathcal{N}_h$ with probability $\mathcal{S}(h)$
5:         repeat
6:             obtain a neighbor $s' := \mathcal{N}_h(s)$
7:             $\mathcal{C}(h) := \mathcal{C}(h) + 1$
8:             construct $f(\circ)$ using Algorithm 4
9:             if $f(s') \leq f(s)$ then
10:                $\mathcal{I}(h) := \mathcal{I}(h) + 1$
11:                $s := s'$ (replace the solution by the improved one)
12:                $\eta := 0$ (reset the stall counter)
13:            else
14:                $\eta := \eta + 1$ (increment the stall counter)
15:            end if
16:            $\mathcal{S}(h) := \mathcal{I}(h)/\mathcal{C}(h)$
17:        until $\eta = \phi|P|$
18:    until $\tau$ repetitions have been completed
19: end for
20: return $P$
```

Table 3.7: An example $\mathcal{N}_{10}$ with $n = 10$, $a = 7$, and $b = 3$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
Π	v_2	v_3	v_4	v_1	v_5	v_6	v_8	v_{10}	v_9	v_7
$\mathcal{N}_{10}\Pi$	v_2	v_3	v_4	v_1	v_5	v_6	v_7	v_{10}	v_9	v_8

Table 3.8: An example $\mathcal{N}_{11}$ with $n = 10$ and $\lambda = 3$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}
Π	v_7	v_8	v_6	v_5	v_1	v_2	v_3	v_4	v_{10}	v_9
$\mathcal{N}_{11}(\Pi)$	v_3	v_4	v_{10}	v_5	v_1	v_2	v_7	v_8	v_6	v_9

are applied directly depends on their *performance* during the execution of EiLS. The performance $\mathcal{S}(h)$ of $\mathcal{N}_h$ is measured in terms of the quantity of the improvement to the local solution using the current compromise function, computed as the ratio of calls that resulted in an improvement to the total number of calls to that neighborhood. On line 1 of Algorithm 3, we initialize all these quantities as zero. Note that the stopping condition of Algorithm 3 is a threshold to the highest $\mathcal{S}(h)$. Algorithm 5 selects a neighborhood with a probability proportional to its current performance (we used rejection sampling in our experiments, although roulette-wheel selection would also work). The process is repeated sequentially τ times for each solution of the current population.

Section 3.5.4 discusses the calibration of parameters mentioned in this section. Section 3.5.5 discusses and compares results obtained with all our metaheuristic proposals as well as in Section 3.5.6 conclusions of such experimentation are found.

3.4.4 Automatically adjusting evolutionary algorithm with Intelligent Local Search Algorithm for MLDP

Sections 3.4.2 and 3.4.3 proposed two evolutionary methods that require a manual calibration of parameters.

In this section, we propose a metaheuristic algorithm able to *auto-adjust* its parameters to provide good Pareto fronts approximations. It is based on EiLS (3.4.3) but it is able to auto-calibrate itself and employs a novel ranking, described in Section 3.4.4.4. We call this algorithm as *Automatically adjusting Evolutionary algorithm with Intelligent Local Search* (AEiLS).

As in EiLS, AEiLS has its own specific crossover and mutation operators. On one hand, the computationally heavy crossover (combining tours is non-trivial) is substituted by a random shake crossover to produce offsprings and an intelligent local search over various neighborhoods as a substitute for mutation. The general algorithm of this self-adjusting method can be found in Algorithm 6.

3.4.4.1 Initial population

Similarly to EiLS, all initial solutions in AEiLS are created generating random permutations of the n clients. The depot is not permuted but instead prepended to all permutations, hence producing only feasible solutions.

3.4.4.2 Elite set

We rank the solutions into fronts $F_0, F_1, \ldots$ as described in Section 3.4.2.3 but employing a different crowding scheme (Section 3.4.4.4). We construct an *elite set* $\mathcal{E}$ containing the best ε solutions (i.e., starting with F_0, then continuing with

Algorithm 6 Automatically adjusting Evolutionary algorithm with Intelligent Local Search (AEiLS).

INPUT: integers $\gamma > 0$, $\varepsilon > 0$ and $\kappa > 0$.

OUTPUT: a set of solutions P.

```
1: $\forall h \in [1, 12] : \mathcal{S}(h) := 0, \mathcal{I}(h) := 0, \mathcal{C}(h) := 0, \alpha := 0$ (Section 3.4.4.7)
2: generate an initial population $P$ of size $\gamma$ (Section 3.4.4.1)
3: calculate success counters (Section 3.4.4.3)
4: **repeat**
5:     evaluate the fitness of each solution in $P$   }
6:     place the $\varepsilon$ best solutions into $\mathcal{E}$    } (Section 3.4.4.2)
7:     create $\mathcal{K} := \kappa$ couples from $\mathcal{E}$   }
8:     compute the *offsprings* $\mathcal{F}$ of the couples in $\mathcal{K}$ (Section 3.4.4.3)
9:     $P :=$ *mutations* done with Algorithm 5 (Section 3.4.3.6), on $P \cup \mathcal{F}$
10:    recalculate $\alpha$ according to a success ratio (Section 3.4.4.8)
11: **until** $\max_h \mathcal{S}(h) < \alpha$
12: **return** P
```

F_1, and so forth). We then select κ couples of two distinct parents from $\mathcal{E}$ using binary tournament with replacement (i.e., a single solution may form part of multiple couples).

3.4.4.3 Offsprings

For AEiLS, the crossover operator is similar to the crossover process detailed in Section 3.4.3.3 for EiLS. This operator selects a couple of solutions and applies the same perturbation shown in Table 3.1 on Section 3.4.3.3. Note that the two individuals are processed independently. The main difference between the crossover process of EiLS and that of AEiLS is that every solution created and every generation formed have a series of counters to measure the success, in terms of improvement in the global objective, of each one (Algorithm 7).

Each solution $s \in \mathcal{E}$ has associated a counter of success cs_s, each generation i created has a counter Gen_i as well. Such counters are of great importance because each time an offspring is created and evaluated, if its evaluation directly improves the considered objective function in that specific iteration then the success ratio $\mathcal{S}(i)$ is increased for the two original solutions from which such offspring was created from and counters cs_i are increased. Such counters are employed to categorize the success of each explored neighborhood h as well as in the calculation of the mutation probability for each generation. As for the success of the generation, Gen_i is only increased if the improvement of the objective function value is global.

3.4.4.4 Ranking and crowding scheme

One of the biggest differences between EiLS and AEiLS is the crowding scheme. EiLS uses a classical ranking scheme and a classical fitness assignment by crowding distance just as proposed by Deb et al. (2000). AiELS, on the contrary, proposes

Algorithm 7 AEiLS success counters scheme.

INPUT: set of solutions P.

OUTPUT: P.

```
1: construct $f(\circ)$ using Algorithm 4
2: for each $h \in [1, 12]$ do
3:     $\text{Gen}_h := 0$
4:     $\text{Gen}_h := 0$
5:     $\mathcal{C}(h) := 0$
6:     for each $s \in \mathcal{E}$ do
7:         $\text{cs}_s := 0$, $\mathcal{S}(i) := 0$
8:         repeat
9:             if $f(s') \leq f(s)$ then
10:                $\text{cs}_s := \text{cs}_s + 1$
11:            else if $f(s') \leq f(s),\ s \in \mathcal{E}$ then
12:                $\text{cs}_s := \text{cs}_s + 1$
13:                $\text{Gen}_h := \text{Gen}_h + 1$
14:                $\mathcal{S}(i) := \mathcal{S}(i) + 1$
15:            end if
16:        until $\tau$ repetitions have been completed
17:    end for
18: end for
19: return $P$
```

a hybrid methodology by considering a classical ranking scheme which instead of a crowding scheme considers a new fitness assignment procedure based on the *maximum ratio success* per neighborhood explored. Once a population is created, the evaluations on each objective are obtained and the ranking scheme is ruled by the non-dominated fronts as done by Deb et al. (2000).

Our crowding measurement is based in counters of success cs_i associated to every solution belonging to the elite set $\mathcal{E}$ and to Gen_i as well, to compute the success of each created generation; such counters are described in Section 3.4.4.3. By considering the measurements of crowding as in Equation (3.22), we consider only the success ratio caused per solution or its generation in improvements of the objective. Here, improvement is considered in efficiency domination terms.

$$\mathrm{crowding}_i = \frac{\mathrm{cs}_i}{\mathrm{Gen}_i}. \tag{3.22}$$

Algorithm 8 Crowding scheme in EiLS.

INPUT: a set of solutions P.

OUTPUT: a set of solutions P ranked per crowding scheme.

```
1: for each solution $s \in P$ do
2:     for each generation $i$ do
3:         consider $\mathcal{C}(h)$, $\mathcal{S}(i)$, $\mathrm{cs}_s$ and $\mathrm{Gen}_i$
4:         calculate $\mathrm{crowding}_s$ with Equation (3.22)
5:     end for
6: end for
7: return $P$
```

By considering such crowing, there exists two population outcomes:

- solutions producing improvements are successful, meaning they are good solutions,

- such improving producing solutions generate good offsprings and those offsprings are the ones improving the considered objective.

By using the success ratio measurement, a time-consuming crowding such as the one proposed by Deb et al. (2000) is not needed and AEiLS is able to return only good quality solutions from each generation. As mentioned in Section 3.4.4.3, note that the success of a generation is only counted as such if in the whole generation the global objective selected by Algorithm 4 is improved.

3.4.4.5 Mutation

Each solution in the current population undergoes an intelligent local search that locally optimizes a *compromise function* with random weights as proposed by Molina et al. (2007). The construction of this compromise function was described previously in Algorithm 4. Its purpose is to aggregate the original objectives using weights to diversify the resulting Pareto front.

3.4.4.6 Neighborhood exploration

We employ twelve neighborhoods $\mathcal{N}_h$ for the mutation process in AEiLS. Once a neighborhood is selected, it is explored until a maximum number of consecutive iterations with no improvement is reached. It is an adjustable parameter ϕ; we define this maximum as a multiple of γ.

Several neighborhood definitions are given in terms of *closeness* $\lfloor \xi \rfloor$ that limits how many steps away a selected position may be from another position, counting the separation along the permutation Π that represents the solution, allowing moves both backward and forward. Experimental tuning of closeness showed no specific value to be clearly superior on all instances. Hence, instead of fixing it, we duplicate the neighborhood structure with two different values for $\lfloor \xi \rfloor$: $\mathcal{N}_3$ and $\mathcal{N}_4$ use $\xi = 0.5n$ whereas $\mathcal{N}_6$ and $\mathcal{N}_7$ use $\xi = 0.8n$. Instead of fixing it, we let the intelligent selection system choose the most suitable one, based on performance.

- $\mathcal{N}_1$: Two positions a and b, $a \neq b$, are chosen uniformly at random and their contents are exchanged.
- $\mathcal{N}_2$: Three positions a, b, and c are chosen uniformly at random and their contents are exchanged by moving the client of a to b, b to c, and c to a.
- $\mathcal{N}_3$: A position a is chosen uniformly at random and its client is exchanged with another position $b \neq a$ with $\xi = 0.5n$.
- $\mathcal{N}_4$: A position a is chosen uniformly at random, and then two other positions b and c are chosen (all three positions being distinct) are at most $\xi = 0.5n$ steps away from a. The positions of a, b, and c are exchanged using the same scheme as in $\mathcal{N}_2$.
- $\mathcal{N}_5$: A position $a > 1$ is selected uniformly at random and its content is exchanged with the immediately preceding position.
- $\mathcal{N}_6$: Selections and exchanges as in $\mathcal{N}_3$, but with $\xi = 0.8n$.
- $\mathcal{N}_7$: Selections and exchanges as in $\mathcal{N}_4$, but with $\xi = 0.8n$.
- $\mathcal{N}_8$: A position a is chosen uniformly at random among those with $\max c_{i,j}$ (Equation 3.1) where v_i is the client in the preceding position and v_j is the client in the position in question. Then, another position $b \neq a$ is chosen uniformly at random and its contents are exchanged with a.
- $\mathcal{N}_9$: A position a is chosen uniformly at random and its contents are exchanged with a position $b \neq a$ chosen uniformly at random such that $b > \frac{n}{2}$.

- $\mathcal{N}_{10}$: A position $\frac{n}{2} < a < n$ is selected uniformly at random. Then, a position $b > n - a$ is chosen uniformly at random. The contents of a and b are exchanged.

- $\mathcal{N}_{11}$: An integer $\lambda \in [2, \lfloor 0.3n \rfloor]$ is chosen uniformly at random. Then, two *non-overlapping* tour segments of length λ are chosen uniformly at random and their contents are exchanged.

- $\mathcal{N}_{12}$: First, a position a is chosen uniformly at random. Then a position b such that $a < b$ is chosen uniformly at random. The contents of a and b are exchanged.

3.4.4.7 Neighborhood performance

As proposed by Molina et al. (2018), the order in which the $\mathcal{N}_h$ neighborhoods are applied directly depends on their *performance* during the execution of AEiLS. The performance $\mathcal{S}(h)$ of $\mathcal{N}_h$ is measured in terms of the quantity of the improvement to the local solution using the current compromise function, computed as the ratio of calls that resulted in an improvement to the total number of calls to that neighborhood. The neighborhood performance is measured similarly as in EiLS (Algorithm 5) and the stopping condition of Algorithm 6 is a threshold to the highest $\mathcal{S}(h)$.

3.4.4.8 Autocalibration of parameters

Specifically for AEiLS, the performance is quantified by considering the success ratio per solution in the elite set $\mathcal{E}$ and by the generation as a whole. Such success is quantified in terms of the improvement of the local solution using the objective constructed by previously described Algorithm 4. Opposed to EiLS algorithm, each $\mathcal{N}_k$ neighborhood is explored until a maximum number of consecutive iterations with no improvement is reached. The difference in AEiLS is that α auto calibrates itself, depending on the number of consecutive successes obtained by the neighborhood exploration. The rule to iterate is given by a random number in the first iteration and after this iteration the performance of the neighborhoods $\mathcal{S}(k)$ is updated according to Algorithm 10 and Equation (3.23).

Algorithm 10 Success indicator per explored neighborhood

```
1: $\mathcal{I}(k)$ := the number of times the current solution was improved using $\mathcal{N}_k$ in any
   local exploration
2: $\mathcal{C}(k)$ := the number of times that $\mathcal{N}_k$ has been called so far
3: $\mathcal{S}(k) := \frac{\mathcal{I}(k)}{\mathcal{C}(k)}$
```

$$\alpha = \frac{\text{max\_evals}(\mathcal{N}_k)}{\mathcal{S}(k)} \tag{3.23}$$

As shown in Algorithm 5 (cf. Section 3.4.3.6), each neighborhood has a probability to be chosen and a local search is performed until a maximum number of consecutive iterations with no improvement is reached. After this, the performance of the neighborhoods is updated with respect to the improvements obtained. This is repeated sequentially, using the best solution of the preceding round as the starting solution of the following round, seven times for each individual of the current population with a probability of $\mathcal{S}(k)^{-1}$ to be mutated. Notice that by the way such mutation is defined, by counting the success of each module to know when to stop, it allows the algorithm to have more consecutive iterations when the success rate is higher. When success rate decreases, so do the iterations, and the module is replaced by another.

3.5 Experimentation

Thus far, we have presented four techniques to obtain Pareto of our Minimum Latency-Distance Problem (MLDP): Augmecon2, SMSA, EiLS, and AEiLS. In this section, we study and compare their performance.

3.5.1 Trial instances

To test our algorithms, we experiment on instances inspired by Angel-Bello et al. (2013), with adjustments to create client locations and service times more similar to real-world settings. This set of instances are inspired by those published and created by Angel-Bello et al. (2013), but with adjustments to create client locations and service times more similar to real-world settings (Arellano-Arriaga and Schaeffer, 2017). We first create c cities with uniform random coordinates in $(X, Y) \in \mathbb{R}$, $X, Y \sim \text{Unif}(0, 1)$, and place n clients, selecting for each client a city, randomly. Notice that each city is selected with a probability proportional to the current number of clients attached to it in order to create a population pattern similar to real-world cities.

Once city is randomly selected, the coordinates of the client as the coordinate of its city are computed with normally distributed offsets $\Delta X, \Delta Y \sim \text{Norm}(\mu, \sigma)$, where μ and σ are parameters. We compute the travel distances $t_{i,j}$ for $i \neq j$ as Euclidean distances between the clients with an exponential delay $\delta \sim \text{Exp}(\lambda_d)$ and the service times $t_{i,i} \sim \text{Exp}(\lambda_t)$, this to force the service time at the depot to zero, while locating the depot as if it were another client.

To test all algorithms we experiment with $n \in \{10, 16, 20, 32, 40, 64, 80, 128, 160, 256\}$ clients with $c = \lfloor \log_s n \rfloor$ cities, $\mu = 1/c$, $\sigma = 1/\sqrt{n}$, $\lambda_d = 1/5\sqrt{n}$, and $\lambda_s = 1/\sqrt{n}$, creating 35 instances of each size. Note that by multiplying all the service times by a constant, the relative balance between service and travel times can be adjusted without regenerating the instance.

3.5.2 Measurements of the quality of the obtained fronts

To compare the performance of all our proposed metaheuristics against the exact Pareto fronts obtained with the Augmecon2 algorithm, we consider five metrics to measure the quality of the solutions. These metrics are shown on Table 3.9.

Table 3.9: Metrics list used to measure the quality of the solutions found by our proposed algorithms.

Label	Metric	Reference
#	number of points	–
SCC	size of the space covered	Zitzler and Thiele (1999)
k-D	k-distance	Zitzler et al. (2001)
M_1^*	distance between the exact Pareto front and the approximated Pareto front	Zitzler (1999)
c(∘,∘)	coverage of the fronts	Zitzler and Thiele (1999)

3.5.3 Technical specifications

Our implementation of Augmecon2 was solved using the MILP solver ILOG CPLEX C++ Concert Technology. All our heuristic procedures, SMSA (Section 3.4.2), EiLS (Section 3.4.3), and AEiLS (Section 3.4.4), are implemented in C++. Every instance was solved in a Xenon® Intel® CPU E3-1245 v3 @ 3.40GHz with 16 GB of RAM.

3.5.4 Parameter calibration

To calibrate the parameters of the proposed solution methods, we treat all algorithms independently.

- Augmecon2 (Section 3.4.1): the only parameter to be fixed is the step it takes to obtain each point in the Pareto front. For our implementation this step is fixed as $\delta = 1/2n$, where n denotes the size of the instance.
- SMSA (Section 3.4.2): we vary the size of the population (γ in Algorithm 1), the probability to mutate a solution (β in Algorithm 1), as well as number of iterations (I in Algorithm 2). All this with a fixed $R = 5$.
- EiLS (Section 3.4.3): we set $\chi_{\max} = \gamma$, we fix $\tau = 7$, $\varepsilon = \lceil \gamma/2 \rceil$, and $\kappa = \lceil \varepsilon/2 \rceil$, and vary the size of the population (γ in Algorithm 3), the number of evaluations per module (ϕ in Algorithm 3) as well as the minimum success in the exploration of each neighborhood (α in Algorithm 3).
- AEiLS (Section 3.4.4): due to the auto calibration of parameters this algorithms calibrates itself. We fix the size of the Elite set to $\varepsilon = \lceil \gamma/2 \rceil$ and the number of couples to generate offsprings to $\kappa = \lceil \varepsilon/2 \rceil$.

These parameters were calibrated under a factorial design of experiments and the combination of parameters where the best quality in the approximated fronts was obtained was selected as the fixed combination for the metaheuristic.

This experimentation was made for our SMSA and EiLS metaheuristics. For each of the 35 replicas, we performed 10 executions, and for each of the 350 resulting fronts we computed the *number of points* (#) in the resulting front together with the *size of the space covered* (SCC) or hypervolume (Zitzler and Thiele, 1999).

Table 3.10: Factorial design for parameter calibration for SMSA (left) and EiLS (right); n is the number of clients, # the number of points in the approximated front, and SCC the size of space covered.

	SMSA					EiLS				
Label	P1	P2	P3	#	SCC	P1	P2	P3	#	SCC
1	$5n$	0.05	$n/5$	5.92	0.345	n	10	0.3	56.24	0.568
2	$5n$	0.05	$n/10$	5.25	0.329	n	10	0.5	60.09	0.581
3	$5n$	0.03	$n/5$	6.17	0.518	n	10	0.7	101.14	0.583
4	$5n$	0.03	$n/10$	6.53	0.533	n	13	0.3	103.50	0.585
5	**10n**	**0.05**	**n/5**	**7.19**	**0.553**	n	13	0.5	106.90	0.622
6	$10n$	0.05	$n/10$	6.23	0.544	n	13	0.7	145.59	0.640
7	$10n$	0.03	$n/5$	6.14	0.532	n	15	0.3	146.41	0.649
8	$10n$	0.03	$n/10$	5.05	0.482	n	15	0.5	147.75	0.652
9						n	15	0.7	152.04	0.660
10						n^2	10	0.3	157.56	0.663
11						$\mathbf{n^2}$	**10**	**0.5**	**164.48**	**0.672**
12						n^2	10	0.7	152.52	0.671
13						n^2	13	0.3	149.83	0.670
14						n^2	13	0.5	134.79	0.662
15						n^2	13	0.7	118.25	0.661
16						n^2	15	0.3	117.58	0.649
17						n^2	15	0.5	109.57	0.645
18						n^2	15	0.7	106.88	0.641
19						n^3	10	0.3	104.21	0.640
20						n^3	10	0.5	98.33	0.633
21						n^3	10	0.7	98.20	0.622
22						n^3	13	0.3	98.13	0.611
23						n^3	13	0.5	95.38	0.600
24						n^3	13	0.7	90.27	0.600
25						n^3	15	0.3	82.45	0.588
26						n^3	15	0.5	78.94	0.583
27						n^3	15	0.7	78.82	0.572

For both methods, a single parameter combination produced the best value for both metrics (i.e., the highest number of points in the front and the largest hypervolume). Table 3.10 shows the quality obtained with each combination of parameters, where for both algorithms n refers to the size of the instance. The best-performing combination for each method is indicated in boldface and highlighted in gray.

3.5.5 Computational experimentation

The free parameters of SMSA and EiLS were calibrated under a factorial design of experiments. The combination of parameters for which the best quality in the approximated fronts was obtained was selected as the fixed combination for each algorithm for a comparative experimentation, the results of also are discussed in two sets: small instances (sized up to 40 vertices) and large instances (sized up to 256 vertices), all of them in terms of average values for each size, over sets of 35 instances each. When possible, the same instance sets are solved by all four solution approaches.

3.5.5.1 Small instances

We consider as small instances those that were able to be solved with the exact methodology (Section 3.4.1) that was able to solve sets of up to 40 vertices. All the reported values of the metrics are in terms of averages over the sets of 35 instances of each size, ten executions per instance. We use the metrics reported on Table 3.9 (page 93) to compare the performance of our proposed algorithms:

CPU times elapsed with all methodologies are reported in Table 3.11.

Table 3.11: Average CPU times obtained by our four proposed solution methods. These times are shown in averages per each set of instances and measured in seconds; n is the size of the instance sets solved.

Instance	n	Augmecon2	SMSA	EiLS	AEiLS
MLDP_10	10	20.228	2.875	**0.22**	0.205
MLDP_16	16	179.722	6.443	**2.59**	3.275
MLDP_20	20	800.531	29.953	**13.29**	17.115
MLDP_32	32	26946.179	201.345	**99.44**	102.885
MLDP_40	40	452100.702	527.655	**203.73**	206.395

EiLS and AEiLS show a better time-performance than SMSA and Augmecon2 by obtaining solutions in roughly half the time than our SMSA algorithm. Nonetheless, EiLS had a better performance than the self-adjusting AEiLS.

The performance of these proposed algorithms is reported in Table 3.12, divided into sections: each section corresponds to one of our proposed methods and each method contains four columns: # (average number of points obtained in the front), SCC (hypervolume), k-D (distance among the points obtained in the fronts), and M_1^* (distance among the exact fronts and the approximated ones). All reported values are in averages over the sets of 35 instances for each size.

Table 3.12 states the superiority of the fronts obtained by both: EiLS and AEiLS. These algorithms have the ability to employ a more sophisticated local search neighborhood selection which leads to the improvement in the density of the approximated Pareto fronts while keeping a good hypervolumen in the solutions. Among them both, there is a subtle superiority for EiLS over AEiLS in obtaining more points and getting a better coverage of the non-dominated points than our automatically adjusted version. The distance among the points within the obtained fronts is so small that it leads to dense approximated fronts, and the distance such fronts have

Table 3.12: Comparison among the metrics obtained per algorithm. These values are averages over the set of instances.

	Augmecon2				SMSA				EiLS				AEiLS			
n	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*
10	4.622	0.912	1.414	0	8.446	0.898	0.213	0.295	**35.218**	**0.949**	<**0.001**	0.016	32.274	0.921	<**0.001**	**0.013**
16	5.727	0.876	0.098	0	9.743	0.892	0.326	0.146	**91.422**	**0.938**	<**0.001**	0.023	90.969	0.929	<**0.001**	**0.018**
20	7.249	0.904	0.057	0	9.893	0.891	0.166	0.292	**138.261**	**0.935**	<**0.001**	0.013	137.261	0.912	<**0.001**	**0.009**
32	9.326	0.918	0.193	0	11.327	0.879	0.085	0.299	**201.319**	**0.921**	<**0.001**	0.055	199.847	0.919	<**0.001**	**0.032**
40	9.785	0.912	0.058	0	11.994	0.816	0.068	0.107	**592.368**	**0.921**	<**0.001**	0.068	589.992	0.919	<**0.001**	**0.071**

to the exact fronts is small as well.

The coverage of the fronts is presented in Table 3.13. In general, to measure the coverage of two fronts (A and B) is to measure the volume of points of front A that are dominated by front B: this means that either of both fronts that dominates more points is a better front. Table 3.13 allows us to highlight the superiority of both: EiLS and AEiLS algorithms over the SMSA because they both dominate a big volume of the points found by SMSA. Note that because Augmecon2 is able to find optimal solutions but it may not find non-supported points in the Pareto front, this leads to less points and less variability. Because of this we only report SMSA, EiLS and AEiLS coverages.

Table 3.13: Average coverage of the approximated fronts obtained by the SMSA, EiLS, and AEiLS procedures.

Instance	n	c(SMSA,EiLS)	c(SMSA,AEiLS)	c(EiLS,SMSA)	c(EiLS,AEiLS)	c(AEiLS,SMSA)	c(AEiLS,EiLS)
MLDP_10	10	<0.001	<0.001	**0.796**	0.004	0.792	0.001
MLDP_16	16	0.062	0.059	**0.832**	0.023	0.824	0.001
MLDP_20	20	0.183	0.181	**0.857**	0.033	0.849	0.002
MLDP_32	32	0.195	0.198	0.847	0.038	**0.852**	0.013
MLDP_40	40	0.197	0.198	0.914	0.038	**0.927**	0.018

With the results reported in Tables 3.12 and 3.13, it is clear that under small instances EiLS and AEiLS are able to provide good quality approximations to the Pareto fronts obtained by Augmecon2 but in shorter computational time. In small instances, we are unable to report a well defined superiority among both algorithms, due to the effectiveness of both and the apparent best performance of EiLS in smaller sized instances.

3.5.5.2 Large instances

Section 3.5.5.1 shows the experimentation obtained in small instances. We now report results for instances that the exact method was no longer able to solve. Each set contains 35 replicas (i.e., independent instances created with the same value of n) and we report results with $n \in \{64, 80, 128, 160, 256\}$, ten executions per instance of each method. Due to the fact that our implementation of an exact methodology is not able to solve instances of such sizes, the comparison is just between our three metaheuristic approaches: SMSA, EiLS, and AEiLS. All results are reported in averages obtained per set of instance.

Table 3.14: Average CPU times obtained by our three approximation methods in larger instances. These times are shown in averages per each set of instances and measured in seconds.

Instance	n	SMSA	EiLS	AEiLS
MLDP_64	64	627.283	301.873	**293.472**
MLDP_80	80	1002.374	379.572	**361.385**
MLDP_128	128	1473.287	683.958	**597.283**
MLDP_160	160	1924.739	935.284	**891.373**
MLDP_256	256	2423.624	1225.448	**999.374**

Table 3.14 shows the comparison in CPU time consumed in obtaining these fronts. Table 3.14 shows that both, EiLS and AEiLS have a better time-performance than SMSA in larger instances as well by only consuming about half the time SMSA takes to approximate the Pareto fronts. It is shown that also AEiLS have a slightly better CPU time than EiLS, above all, in the larger sizes tested.

Table 3.15: Comparison among the metrics obtained per algorithm in larger instances. These values are averages over the set of instances.

		SMSA			EiLS			AEiLS		
Instance	n	#	SCC	k-D	#	SCC	k-D	#	SCC	k-D
MLDP_64	64	17.372	0.8517	0.081	679.285	**0.9271**	<**0.001**	**731.371**	0.9226	<**0.001**
MLDP_80	80	21.273	0.8312	0.078	843.572	**0.9129**	<**0.001**	**893.362**	0.9112	<**0.001**
MLDP_128	128	38.371	0.8298	0.075	1085.274	0.9081	<**0.001**	**1142.383**	**0.9102**	<**0.001**
MLDP_160	160	50.282	0.8288	0.071	1483.753	0.8934	<**0.001**	**1512.248**	**0.9021**	<**0.001**
MLDP_256	256	75.813	0.8141	0.068	1812.935	0.8712	<**0.001**	**1873.372**	**0.8801**	<**0.001**

To evaluate the performance our three proposed metaheuristics have in larger instances, we divide Table 3.15 into three sections, one for each method, and each section containing four columns: #, SCC, k-D, and M_1^*. All reported values are in averages over the sets of 35 instances for each size. Table 3.15 shows a great difference in the volume of points obtained by EiLS and AEiLS against SMSA, and clearly show the superior ability of the former two to provide more populated fronts than SMSA. Contrary to the performance in small instances (cf. Section 3.5.5.1), in larger instances the latter being superior. EiLS and AEiLS use their intelligent neighborhood selection to improve the density of the approximated Pareto fronts while mantaining a good hypervolumen. As exact fronts could not be computed, we omit the M_1^* metric that would compare the approximated fronts to the exact ones.

Due to the MLDP constraints, the search space is smaller than in k-MLDP and therefore, it is expected for EiLS to perform better than AEiLS in small instances. Nonetheless, with larger instances, AEiLS has enough time to calibrate and produces slightly better results than EiLS.

Table 3.16: Average coverage of the approximated fronts in larger instances. Averages obtained by the SMSA, EiLS and AEiLS.

Instance	n	c(SMSA,EiLS)	c(SMSA,AEiLS)	c(EiLS,SMSA)	c(EiLS,AEiLS)	c(AEiLS,SMSA)	c(AEiLS,EiLS)
MLDP_60	60	0.201	0.193	0.919	0.001	**0.920**	0.001
MLDP_80	80	0.205	0.191	0.931	0.003	**0.937**	0.005
MLDP_128	128	0.219	0.212	0.944	0.007	**0.948**	0.010
MLDP_160	160	0.208	0.203	0.969	0.007	**0.970**	0.012
MLDP_256	256	0.213	0.205	0.971	0.012	**0.974**	0.043

The coverage of the fronts is reported in Table 3.16, for SMSA, EiLS and AEiLS: EiLS and AEiLS are superior to SMSA. Note that because Augmecon2 is able to find optimal solutions but it may not find non-supported points in the Pareto front, this leads to less points and less variability. Hence we only report SMSA, EiLS and AEiLS coverages. In larger instances, AEiLS has a slightly better performance than EiLS.

For solving larger instances, EiLS and AEiLS are able to provide good-quality approximations to the Pareto fronts in a short computational time and with a good diversity of the front.

3.5.6 Conclusions

Incorporating smart neighborhood exploration to approximate Pareto fronts of MLDP leads to better results than a classical multi-objective technique.

EiLS has a better performance than AEiLS in smaller instances: EiLS starts iterating at a good parameter combination while AEiLS has to adjust its parameters, resulting in poor solutions in the beginning. When instances are larger, AEiLS has enough time to self-adjust and slightly outperforms EiLS. Given enough time to iterate, the self-adjusting method calibrates itself appropriately.

3.6 Conclusions and future research

Our single-vehicle bi-objective problem, the Minimum Latency-Distance Problem (MLDP), involves minimizing the travel distance and the total latency of an assumed infinite-capacity tour. We propose a bi-objective approach which combines two well-known combinatorial problems with the goal to integrate the client as an active part in the decision-making process of a routing problem.

MLDP is an NP-hard problem (Section 3.3), which justifies the employment of approximated methods. We proposed a non-linear formulation for this problem, that can be solved to obtain the exact Pareto fronts (Sections 3.4 and 3.5).

We provide a set of new instances that take into account several adjustments to recreate real-world settings (Section 3.5.1), and were able to find the exact Pareto front of instances with ten to forty clients (Section 3.5.5.1). We compared three methods:

- Strategic Memetic Search Algorithm (SMSA), a metaheuristic based on the classic multi-objective NSGA-II algorithm,
- Evolutionary algorithm with Intelligent Local Search (EiLS), an intelligent neighborhood selection method, and
- Automatically adjusting Evolutionary algorithm with Intelligent Local Search (AEiLS), a self-adjusting intelligent neighborhood selection method.

EiLS and AEiLS, guided by an intelligent selection of a more appropriate neighborhood in the local search procedure, are superior to SMSA, in terms of density of the fonts, hypervolume percentage, computational time required, and other quality metrics.

Due to the characteristics of MLDP, an adequate parameter calibration can be obtained manually (Section 3.5.4), and hence, it is expected for EiLS to outperform AEiLS in small instances as AEiLS has to adjust its parameters during the execution. Nonetheless, for larger instances, AEiLS has enough time to self-adjust and slightly outperform EiLS.

As future work, we wish to incorporate real-world conditions in more detail to develop a more realistic mathematical formulation such as the introduction of *capacity* for each one of the vehicles. A stochastic version of this problem as well as the redefinition of the assumptions made for the linearity between traveling time and traveled distance are also worth considering. Chapter 4 considers a version of the MLDP with a fleet of vehicles instead of a single agent to carry out the visiting.

Chapter 4

k-Minimum Latency-Distance Problem

This chapter introduces a multi-vehicle result of the Minimum Latency-Distance Problem, this version considers the use of several vehicles of infinite capacity to perform the visiting. We call this problem the *k-Minimum Latency-Distance Problem* (k-MLDP).

Section 4.1 introduces this new problem, Section 4.2 shows the mathematical formulation as well as the assumptions considered for it. The computational complexity of our proposed problem is established in Section 4.3, and solution methods are shown in Sections 4.4.1 – 4.4.4. Parameter calibration and computational experimentation are reported in Section 4.5. Lastly, conclusions for this chapter are given in Section 4.6.

4.1 k-Minimum Latency-Distance Problem: an introduction

The Multiple Traveling Salesman Problem (m-TSP) is a classic combinatorial problem which aims to minimize the total traveled distance of a vehicle in charge of visiting a set of clients (cf. Section 2.3). The main application of this problem is found in vehicle routing (Bektaş and Elmastaş, 2007; Clarke and Wright, 1964; Li and Fu, 2002; Min, 1989; Park and Kim, 2010; Schittekat et al., 2006, 2013; Tang et al., 2000) or door to door visiting (Gilbert and Hofstra, 1992; Kergosien et al., 2009; Sipahioglu et al., 2008).

The employment of several vehicles in a fleet in charge of performing a service to a set of clients with the aim of minimizing the total waiting time of all the clients is known as the k-Traveling Repairman Problem (k-TRP, Section 2.5) Fakcharoenphol et al. (2003); Nucamendi-Guillén et al. (2016); Onder et al. (2017). The main application of such is in humanitarian logistic (Dikbyk, 2017; Ma et al., 2015) and routing (Avci and Avci, 2017; González and Rivera, 2015; Ha Bang, 2018; Martin and Salavatipour, 2016).

In both m-TSP and k-TRP, every vehicle in the fleet must have at least a client assigned. Each vehicle must not carry out sub-tours and each client can only be visited once.

Let us consider a company that provides a service to a set of clients with a fleet of k agents that all depart from a known depot and return to it at the end of the day. Each of the clients is visited exactly once. This company seeks to satisfy the requests of the clients as fast as possible by minimizing the total waiting time of the client set while the agents travel the minimum distance to perform the visiting. With such delimiters, we propose a bi-objective problem called the k-Minimum Latency-Distance Problem (k-MLDP) to simultaneously optimize both of these objectives.

To the best of out knowledge, a similar multi-vehicle approach has not been studied in a routing context, but has been seen in medical and telecommunications contexts (Frye, 1995; Kara et al., 2007; Kim et al., 2012; Malarky, 2013; Mangharam et al., 2007).

4.2 k-MLDP: assumptions and mathematical formulation

To formally define the k-MLDP, let us consider a company that provides a service to a set of clients through a fleet of vehicles. Such fleet has a fixed number of k vehicles and each one of them departs from a known depot and returns to it at the end of the day. All k vehicles must have at least one client assigned and it must be reassured that all clients have only an agent scheduled to visit them. The main goal is to find a combination of k active tours that fulfil the visiting of all clients while aiming to minimize the total travel distance of all agents as well as to minimize the total waiting time of the clients.

We assume that the duration of each service, the exact distance between the depot and the clients, as well as the distance between all the clients are known with certainty. The direct proportionality of travel time to distance traveled as well as the sufficient capacity of each and every one of the agents to attend all the clients it has assigned in its tour are assumed as well. Note that because of the assumption of linearity between travel time and travel distance, we refer to the two indistinctly.

k-MLDP deals with the optimization of two well-known routing objectives measured with a different metric (Blum et al., 1994). Figure 4.1 shows a graphical representation of a set of k tours in charge of visiting a set of ten vertices by minimizing the total latency and distance of it. Each grey arrow depicts the known possible connection among all clients and the depot.

Note that despite having k active agents to perform the service on each client, each of those tours is a single MLDP tour and therefore, no returning arcs to the depot are considered. In Figure 2.6 as well as in Figure 4.1, the returning arc is depicted as a dashed line to indicate that it is only considered when computing the distance objective.

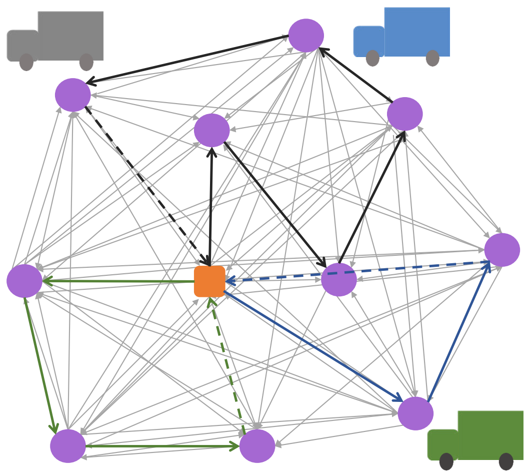


Figure 4.1: Graphical example of an small k-MLDP solution in an instance of ten clients.

To model k-MLDP, we part from a model developed for the objective of latency, into which we incorporate the distance objective; we base our formulation on the work of Nucamendi-Guillén et al. (2016). Let us consider a directed and complete graph $G = (V, A)$. Let A be the arc set and V a vertex set $V = \{v_0, v_1, v_2, \ldots, v_n\}$, where v_0 represents the depot and the remaining vertices represent the n clients to be visited. A set $I = \{v_1, v_2, \ldots, v_n\}$ is derived from set V and denotes the n clients. A total of k vehicles are in charge of fulfilling the visiting of the clients; all k vehicles must be active and have at least one client assigned to them. Due to this constraint being inflexible, a bound on the number of clients assigned to a client is defined as,

$$N = n - k + 1, \tag{4.1}$$

characterizing the worst-case scenario for each agent: all other agents only have one single client assigned.

Each client $i \in I$ has a *service time* s_i associated to it. This service time refers to the time the agent takes to fulfil the service to the client. Each arc $(v_i, v_j) \in A$ has associated a *travel time* from client i to j, defined as t_{ij}, which represents the distance from client i to client j.

A matrix $\mathbf{T}$ contains for each pair (v_i, v_j) a weight $t_{i,j} \geq 0$ (the travel time from vertex v_i to vertex v_j), and where the diagonal element $t_{i,i}$ represents the service time s_i of client i, a *cost matrix* $\mathbf{C}$ is computed with elements:

$$c_{i,j} = t_{i,i} + t_{i,j}. \tag{4.2}$$

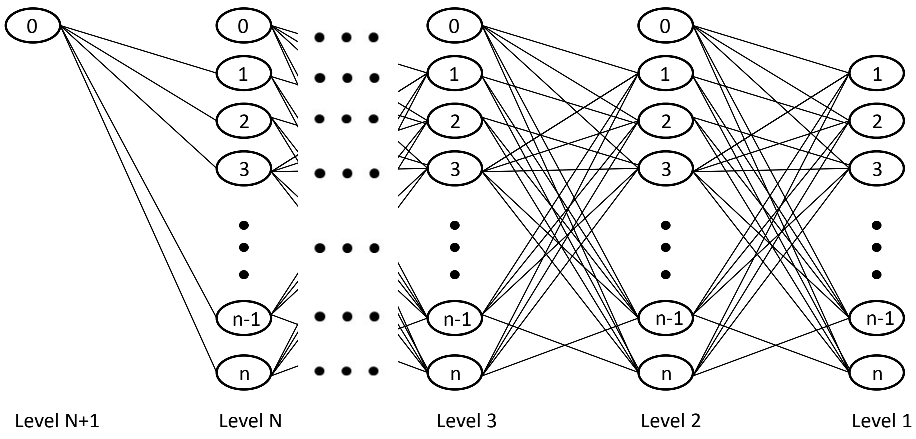


Figure 4.2: Multi-level network used to formulate the k-MLDP.

To represent k-MLDP, we base our formulation on a $N+1$ multi-level network, similarly to the k-TRP characterization of Nucamendi-Guillén et al. (2016). Such network is depicted in Figure 4.2: level one consists of a copy of all vertices associated with clients, $v_1, v_2, \ldots, v_n$. Levels $2, 3, \ldots, N$ are copies of V and level $N+1$ represents the depot. This network is helpful in representing a k-MLDP solution, this is to say, a set of k active and disjointed tours which start in the depot and in total, visit all clients.

A feasible solution of k-MLDP is a set of k paths starting at the depot and visiting vertices until the k agents have together visited all clients. A client i can only belong to one path. Each path should only visit a vertex in each level lower than the starting vertex level; if a path starts in vertex h, then this path has $h-1$ vertices associated to it.

Note than a k-MLDP solution only considers the returning arc of each of the k tours to the depot in the calculations of the total traveled distance for each tour. Hence we define the decision variables $x_{i,l}$ and $y_{i,j}(l)$ as

$$x_{i,l} = \begin{cases} 1, & \text{if vertex } i \text{ is active in level } l \text{ in the path,} \\ 0, & \text{otherwise,} \end{cases}$$

$$y_{i,j}(l) = \begin{cases} 1, & \text{if vertex } i \text{ in level } l+1 \text{ is followed by vertex } j \text{ in level } l, \\ 0, & \text{otherwise.} \end{cases}$$

k-MLDP is linearly formulated as follows,

$$\min\ F_1 = \sum_{j=1}^{n} c_{0j} \sum_{l=1}^{N} l y_{0,j}(k) + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} c_{ij} \sum_{l=1}^{N-1} l y_{i,j}(l) \tag{4.3}$$

$$\min\ F_2 = \sum_{j=1}^{n} c_{0j} \sum_{l=1}^{N} l y_{0,j(k)} + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} c_{ij} \sum_{l=1}^{N-1} l y_{i,j}(l) + \sum_{i=0}^{n} c_{i0} \sum_{l=0}^{N} y_{i,0}(l) \tag{4.4}$$

subject to

$$\sum_{l=1}^{N} x_{i,l} = 1,\ i = 1, 2, \ldots, n \tag{4.5}$$

$$\sum_{i=0}^{n} x_{i,1} = k \tag{4.6}$$

$$\sum_{l=1}^{N} \sum_{j=1}^{n} y_{0,j}(l) = k \tag{4.7}$$

$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{i,j}(l) = x_{i,l+1},\ i = 1,2,\ldots,n;\ l = 1,2,\ldots,N-1 \tag{4.8}$$

$$y_{0,j}(l) + \sum_{\substack{i=1\\ i\neq j}}^{n} y_{i,j}(l) = x_{j,l},\ j = 1,2,\ldots,n;\ l = 1,2,\ldots,N-1 \tag{4.9}$$

$$y_{0,j}(N) = x_{j,N},\ j = 1,2,\ldots,n \tag{4.10}$$

$$x_{i,l} \in \{0,1\},\ i = 1,\ldots,n;\ l = 1,\ldots,N \tag{4.11}$$

$$y_{i,j}(l) \geq 0,\ i = 1,\ldots,n;\ j = 1,\ldots,n;\ l = 1,\ldots,N-1 \tag{4.12}$$

The minimization of the total waiting time of all the clients visited by the fleet of vehicles is stated in Equation (4.3). Equation (4.4) states the minimization of the total travel distance by the fleet of vehicles. In Equation (4.3), the arcs denoting the return of the vehicles to the depot are not considered meanwhile all returning arcs have to be considered in Equation (4.4) as we do not consider the returning of the vehicles to the depot as client waiting time.

Equation (4.5) guarantees that there can only be one active vertex per level in the multilevel network shown in Figure 4.2, meanwhile Equation (4.6) requires that each agent must visit at least one client, to ensure all k vehicles are active. Hand to hand, Equation (4.7) ensures there are exactly k vehicles. Equation (4.8) ensures that from level $l+1$ can leave arcs from the nodes that are active at that level. Equation (4.9) impose that at level l can arrive arcs to nodes that are active at that level, and Equation (4.10) ensures that if a vertex on level N is active, level $N+1$ must be connected to that vertex.

Lastly, Equations (4.11) and (4.12) denote the nature of the variables. Note that variables $y_{i,j}(l)$ are defined as binary but they are handled as continuous variables in the model. This is achieved as the binary variables $x_{i,l}$ determine the active vertices in the network and this simplifies finding the k shortest paths from the active vertex in level l through the active vertices in the intermediate levels. This is to say, this is an uni-modular problem and therefore there is no need to define $y_{i,j}(l)$ as binary variables. This formulation consists of n^2N binary variables, $(n^3-2n^2+n)N$ real variables, and $(2n^2)N+2n$ constraints.

As MLDP, our k-MLDP formulation can be expressed in non-linear terms under the same parameters and in the same multi-level network. This is achieved by representing the selection of vertices over the network with boolean decision variables $x_{i,l}$,

$$x_{i,l} = \begin{cases} 1, & \text{if } v_i \text{ is selected on } \ell_l, \\ 0, & \text{otherwise,} \end{cases} \tag{4.13}$$

where i takes values from $1,\dots,n$, and l takes values from $1,\dots,N$. Note that the depot v_0 forms ℓ_{N+1}, meaning no selection is made there as every k-MLDP feasible tour initiates at the depot. Additionally, we write as an auxiliary notation

$$y_{i,j}(l) = x_{i,l+1}x_{j,l}, \tag{4.14}$$

meaning that $y_{i,j}(l)=1$ if and only if v_i is selected on ℓ_{i+1} and v_j is selected on ℓ_i, and zero otherwise. With the above notation, the travel distance from the depot to the first client is defined as

$$\mathcal{T}_0 = \sum_{j=1}^{n} c_{0j} \sum_{l=1}^{N} l y_{0,j}(l), \tag{4.15}$$

the travel distance from the last client to the depot is defined as

$$\mathcal{T}_n = \sum_{i=0}^{n} c_{i0} \sum_{l=0}^{N} y_{i,0}(l). \tag{4.16}$$

Similarly, as each of all the n clients will have to wait for the agent to reach the first client, we write

$$\mathcal{W}_0 = \sum_{j=1}^{n} c_{0j} \sum_{l=1}^{N} l y_{0,j}(l). \tag{4.17}$$

The complete k-MLDP formulation is defined as follows,

$$\min\ F_1 = \mathcal{W} = \mathcal{W}_0 + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} c_{ij} \sum_{l=1}^{N-1} l y_{i,j}(l) \tag{4.18}$$

$$\min\ F_2 = \mathcal{T} = \mathcal{T}_0 + \mathcal{T}_n + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} c_{ij} \sum_{l=1}^{N-1} l y_{i,j}(l) \tag{4.19}$$

subject to

$$\forall i : \sum_{l=1}^{N} x_{i,l} = 1 \tag{4.20}$$

$$\forall i : \sum_{i=0}^{n} x_{i,1} = k \tag{4.21}$$

$$\sum_{l=1}^{N} \sum_{j=1}^{n} y_{0,j}(l) = k \tag{4.22}$$

$$\forall i, \forall l : \sum_{\substack{j=1 \\ j\neq i}}^{n} y_{i,j}(l) = x_{i,l+1} \tag{4.23}$$

$$\forall j, \forall l : y_{0,j}(l) + \sum_{\substack{i=1 \\ i\neq j}}^{n} y_{i,j}(l) = x_{j,l} \tag{4.24}$$

$$\forall j : y_{0,j}(N) = x_{j,N} \tag{4.25}$$

$$\forall i, \forall l : x_{i,l} \in \{0, 1\}. \tag{4.26}$$

In this non-linear formulation, Equation (4.18) expresses the objective to minimize the total waiting time over the set of clients, as each client has to wait through the services of all the previous clients and the intermediate travel times. Equation (4.19) denotes the minimization of total travel distance of the entire set of vehicles, from the depot and back to the depot, after visiting all the clients. Equation (4.20) guarantees that only one vertex is selected per level in the network. Equation (4.21) ensures that each agent has at least a client assigned and Equation (4.22) ensures that there are exactly k vehicles.

Equation (4.23) ensures that from level $l+1$ can leave arcs from the nodes that are active at such level, and Equation (4.24) guarantees that at level l can arrive arcs to nodes that are active at that level. Equation (4.10) ensures that if a vertex on level N is active, level $N+1$ must be connected to that vertex. Lastly, Equation (4.26) denotes the nature of the variables.

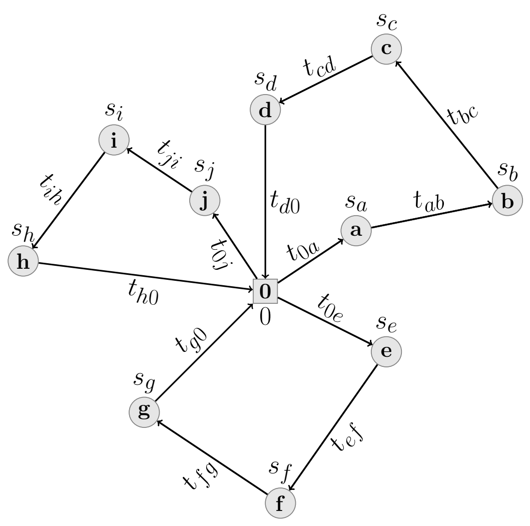
4.3 Complexity of k-MLDP

In this section, we study the computational complexity of the k-MLDP. A short reminder of basic complexity theory is previously given in Section 3.3.

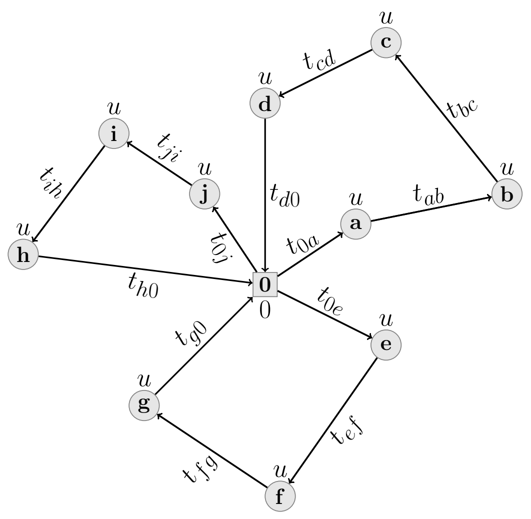
Both of the single-objective optimization problems that are merged into k-MLDP are NP-hard (Afrati et al., 1986; Ausiello et al., 2000; Fakcharoenphol et al., 2003; Papadimitriou, 1994). In this section we demonstrate that also k-MLDP itself is NP-hard.

To demonstrate the complexity of k-MLDP, we are subject to the same assumptions as the model presented in the previous section: we consider a fixed-size fleet of vehicles, each with no capacity restrictions, that visits all of the clients in k disjoint tours. This requires efficiently reducing a known NP-complete problem O_σ to our problem (Garey and Johnson, 1990). We first define the decision problem $D_{\text{k-MLDP}}$ associated to k-MLDP, then show that k-MLDP belongs to NP, and finally establish that an efficient reduction from a known NP-complete problem to our problem exists.

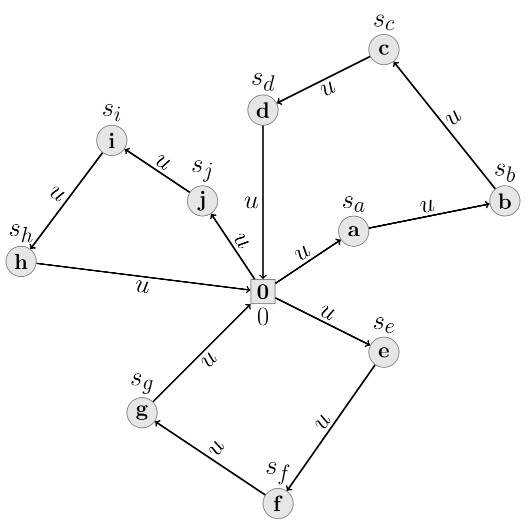
By definition, the k-MLDP consists in finding a set of k disjoint tours which in total visit all clients, leaving from a established depot and returning to it at the end of their shift, while minimizing the total latency of all the clients and the total traveled distance of all k uncapacitated vehicles. Figure 4.3, depicts some particular



(a) s_i, t_{ij}, arbitrary values



(b) constant service times, arbitrary travel times



(c) arbitrary service times, constant travel times

Figure 4.3: Illustration of some particular instances of the k-MLDP. In Figure 4.3a, the general case is shown: in this case, service and travel times all take arbitrary values. In Figure 4.3b, a simpler case is shown: all service times have a constant value u but travel times are arbitrary (a k-TSP instance). Lastly, in Figure 4.3c: all service times are arbitrary and all travel times are equal to a constant u (a k-MLP instance).

cases considered. In a general case, which refers to a more real-life situation for the agents, it considers arbitrary non-zero traveling and service times, (see Figure 4.3a). We also consider some special cases of this problem (see Figures 4.3b, 4.3c).

Nucamendi-Guillén et al. (2016) discuss the latency objective and propose

$$\frac{1}{2}\sum_{k=1}^{n-1}(n-k)\sum_{i=0}^{n}\sum_{j=0}^{n}t_{ij}+(s_i+s_j), \tag{4.27}$$

where t_{ij} refers to travel time among clients i and j, as well as s_i and s_j refers to their corresponding service times.

By using Equation (4.27), we redefine non-zero travel and service times for computing the bounds used to define the decision problem $D_{\text{k-MLDP}}$ associated to k-MLDP.

The corresponding decision problem of k-MLDP is the following: given the costs of Equation (4.2), an upper bound to the total time $\mathcal{T} \leq \Theta$ to Equation (4.4), and an upper bound to the waiting time $\mathcal{W} \leq \Omega$ to Equation (4.3), is there an assignment $\mathbf{X}$ that satisfies all the original constraints as well as the upper bounds set on the objectives?

As stated by Garey and Johnson (1990) to show that k-MLDP belongs to NP it is needed to establish that an efficient reduction from a known NP-complete problem to our problem exists. Theorem 3 proves the reduction of k-MLDP to m-TSP.

Theorem 3. *k-TSP reduces efficiently to k-MLDP.*

Proof. We first establish that the feasibility of a given $\mathbf{X}$ can be verified in polynomial-time: each $\mathbf{X}$ captures a set of k disjoint permutations that start at the depot and can be efficiently recovered from the assignment matrix $\mathbf{X}$ as a set of k disjointed sequences in the form of $v^{(0)}, v^{(1)}, \ldots, v^{(N)}$, where $v^{(0)}$ is always the depot, N is the bound per vehicle defined in Equation (4.1), and the visits from v^1 to $v^{(N)}$ correspond to the clients visited per vehicle.

We need four accumulator variables, one to measure time along the tour, one to count the total travel time per vehicle, k to count the total travel time of the set of vehicles, and the fourth to count the total latency of the fleet. We also need k arrays, one per agent, of N binary variables each, to verify each client assigned to each agent and to guarantee all vehicles are active and have at least a client assigned. We need an array of n binary variables, one for each client, to verify that each one of the clients is properly visited. We proceed in the visiting order, per vehicle, denoting the source by i and the destination by j adding the value of $t_{i,j}$ to both the time accumulator and to both travel-time accumulators, then adding the service time $t_{i,i}$ to the time accumulator, and then the current value of the time accumulator to the latency accumulator, making the client j as visited. Every time a client is visited by one of the k agents, it is marked in the auxiliary array to verify that all clients are visited. This process is carried out for each vehicle, and if at the end of all k permutations, all n binary variables are one, the travel-time accumulator respects its upper bound Θ, and the latency accumulator respects its upper bound Ω, the solution is feasible. As the verification is a polynomial procedure, $D_{\text{k-MLDP}} \in \mathsf{NP}$.

To establish NP-completeness, we reduce the multiple Traveling Salesman Problem (k-TSP) in its decision version to $D_{\text{k-MLDP}}$, as illustrated in Figure 4.4. In the decision version of k-TSP, the input is a constant k which refers to the total number of agents to make the visiting, a cost matrix $\mathbf{C}$ for the travel costs between all n clients (with the diagonal elements being zero) together with an upper bound to the total cost of the whole set of k tours must fulfil, $\mathcal{C}$, and the question is whether a set of k disjointed permutations over the set of clients exists such that it produces a set of tours with the sum of costs of the segments not exceeding $\mathcal{C}$.

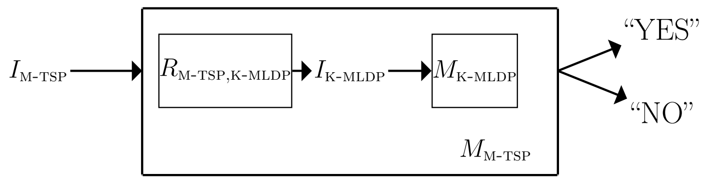


Figure 4.4: Reduction diagram from k-TSP to k-MLDP (both as decision problems). The main box in the figure depicts an algorithm to solve the k-TSP problem that takes as input an instance of k-TSP and determines whether the answer is "yes" or "no" to this instance. To prove the complexity of k-MLDP, a reduction algorithm must transform a k-TSP entry into a k-MLDP one. This k-MLDP input is then solved and receives a "yes" or "no". As a whole, the process takes a k-TSP instance, reduces it into a k-MLDP instance, solves the reduced instance with an algorithm for k-MLDP, and responds correctly "yes" or "no" to the original instance.

In order to transform the input for the decision version of the k-TSP into an input for $D_{\text{k-MLDP}}$, the total number of agents to make the visiting is defined as k_1, we use the elements of $\mathbf{C}$ as $\mathcal{T}$, set $\Theta = \mathcal{C}$, and lastly we only need to compute efficiently an adequate value to describe the total latency bound Ω, which is set for all the k tours an k-MLDP solution must have. Notice that according to Equation (4.1), the worst-case scenario for each agent is given when all the others $k-1$ agents are in charge of visiting a single client each. To compute Ω, we proceed vehicle by vehicle, in terms of the worst-case costs, computing from the cost matrix $\mathbf{C}$ the largest cost per each row (that is, as if the agent at each client chose the most expensive segment to continue the tour when visiting its N assigned clients) in $\mathcal{O}(N^2)$ time, and then sort these worst-case segments from largest to smallest in $\mathcal{O}(n \log N)$ time to construct an upper bound to the worst-case latency per tour. To calculate the complete set of tours we would need $\mathcal{O}(k)(N^2)$ time to compute the expensive segments, and then to sort these worst-case segments from largest to smallest we would need $\mathcal{O}(kN \log N)$. Notice that the service times in k-TSP are the diagonal elements of the cost matrix that are all zero. Therefore the worst-case total latency, per vehicle, is the sum of the cumulative sums of the sorted list of costs. We use the sum of all k worst cases latencies as Ω. Hence, $D_{\text{k-MLDP}}$ responds "yes" to the transformed input if and only if the decision version of k-TSP would respond "yes" for the original input. □

As stated before, this demonstration is subject to the same assumptions considered in the mathematical model presented in Section 4.2: we consider a fixed-sized fleet of vehicles with no capacity restrictions to visit all of the clients in a k disjointed set of tours. This requires efficiently reducing a known NP-complete problem D_σ to our problem (Garey and Johnson, 1990). We first define the decision problem $D_{\text{k-MLDP}}$ associated to k-MLDP, then show that k-MLDP belongs to NP, and finally establish that an efficient reduction from a known NP-complete problem to our problem exists. After proving that k-MLDP reduces efficiently to k-TSP, Theorem 4 establishes the complexity proof of MLDP.

Theorem 4. $D_{k\text{-MLDP}}$ *is* NP-*complete.*

Corollary 4.1. k-MLDP *is* NP-*hard.*

Proof. By Theorem 3 and Corollary 4.1, it is shown that k-MLDP is at least as difficult as k-TSP. Thus, as mentioned in Corollary 4.1, $D_{\text{k-MLDP}}$ is NP-complete and therefore k-MLDP is NP-hard. □

4.4 Solution methods for k-MLDP

k-MLDP is an NP-hard problem and therefore exact methods are only expected to efficiently solve very small instances, making heuristic approaches a necessity.

In this section we discuss the design and implementation of the procedures we implement to obtain the Pareto fronts for a set of instances created specifically for k-MLDP, based on the instances created for MLDP (cf. Section 3.5.1).

Section 4.4.1 discusses obtaining the exact k-MLDP Pareto fronts of these instances, whereas Sections 4.4.2, 4.4.3, and 4.4.4 present heuristic approaches to obtain the approximations of such Pareto fronts.

4.4.1 Exact method

k-MLDP, is a natural generalization of MLDP that deals with the minimization of two objective functions, latency and distance, but instead of minimizing them on a single tour, k-MLDP aims to minimize both objectives using a fixed-sized fleet of vehicles.

We obtain exact solutions for k-MLDP using the linear formulation described in Section 4.2. Similarly as in our MLDP design, we keep the distance objective of Equation (4.4) as the main objective and solve the problem subject to the original constraints along with the latency objective of Equation (4.3); in our case, this produces a single-objective problem which minimizes the distance of the tour constrained to its total latency.

As both of the single-objective problems that k-MLDP may reduce to in this manner are also NP-complete (Afrati et al., 1986; Ausiello et al., 2000; Fakcharoenphol et al., 2003; Papadimitriou, 1994), the asymptotic complexity of computing a Pareto front is exponential in the size of the instance, which in this case is measured in terms of the number of clients. In case the latency objective of Equation (4.3) is the one being fixed and the problem is solved subject to the original constraints along with the distance objective of Equation (4.4), an NP-hard problem is expected as well. Therefore, a metaheuristic approach is justified and desirable. Sections 4.4.2, 4.4.3, and 4.4.4 describe three metaheuristic algorithms to solve this bi-objective problem.

4.4.2 Strategic Memetic Search Algorithm for k-MLDP

We employ Strategic Memetic Search Algorithm (SMSA) (cf. Section 3.4.2) for the k-vehicle version. In this section, we discuss the adjustments made for it.

4.4.2.1 Initial population

A solution of k-MLDP is a permutation of the n vertices that form the client set, considering k times the depot, as k is the number of agents to fulfil the visiting now. An example of a solution of k-MLDP, considering 10 clients and $k = 3$ agents to visit them can be seen in Table 4.1.

Table 4.1: Example of a single k-MLDP solution. This solution has 10 clients and $k = 3$ agents. The depot is denoted as v_0.

π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}
v_0	v_2	v_3	v_4	v_0	v_1	v_5	v_6	v_0	v_8	v_{10}	v_9	v_7

4.4.2.2 Offsprings

The selection of each parent is also made through a binary tournament and after selecting both parents, we apply a combination method based on the OX crossover. Due to the way each k-MLDP solution is coded (see Table 4.1), we apply this modified crossover over each of the k segments individually.

4.4.2.3 Ranking

The ranking scheme requires not additional modifications to work for the multi-vehicle version.

4.4.2.4 Neighborhood exploration

The local search is applied in each of the k segments belonging to each coded solution. This is to say, the process is applied on each one of the k tours in the fleet individually.

4.4.3 Evolutionary algorithm with Intelligent Local Search Algorithm for k-MLDP

In this section we discuss the adjustments made to EiLS (cf. Section 3.4.3) for the multi-vehicle version.

4.4.3.1 Initial population

EiLS solutions are represented the same way as for SMSA in the multi-vehicle version: inserting k-1 non-sequential copies of the depot within the permutation (cf. Table 4.1).

4.4.3.2 Elite set

The ranking scheme is unchanged for the multi-vehicle variant of EiLS. Our *elite set* is constructed similarly as well.

4.4.3.3 Offsprings

The procedure selects two non-overlapping tour segments of size φ and when exchanging them, it verifies that no depot is immediately followed by another one after the exchange both, assuring all agents have at least one client assigned. Table 4.2 shows an example for obtaining an offspring from a 10-client solution with $k = 3$ agents to fulfil the visiting.

Table 4.2: EiLS offspring generation with $n = 10$ and $k = 3$. The exchanged segments are highlighted in grey.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}
p_i	v_0	v_2	v_3	v_4	v_0	v_1	v_5	v_6	v_0	v_8	v_{10}	v_9	v_7
o_i	v_0	v_8	v_{10}	v_9	v_0	v_1	v_5	v_6	v_0	v_2	v_3	v_4	v_7

After interchanging both blocks of vertices, we verify the solution has all k vehicles active, this is to say no depot is followed by another depot in the solution.

4.4.3.4 Mutation

The mutation operator for EiLS is unchanged in the multi-vehicle version.

4.4.3.5 Neighborhood exploration

The neighborhood exploration for the multi-vehicle version of EiLS include all twelve neighborhood of the single-version of EiLS (cf. Section 3.4.3.5), but modified by adding the constraint that no two depot can be consequent in the resulting solution. Four new neigborhoods, made specially for multi-vehicle movements, are added. The exploration process is unchanged of the single-vehicle version of EiLS.

- $\mathcal{N}_{13}$: First, a position $a \in \frac{n}{2}$ is chosen uniformly at random. Then position b such that $b \in \frac{n}{2}$. Contents of a and b positions are exchanged (see Table 4.3). Feasible check-up is made to ensure no depots are consequent.

Table 4.3: An example $\mathcal{N}_{13}$ with $n = 10$ and $k = 3$, where $a = 11$ and $b = 3$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}
Π	v_0	v_7	v_8	v_6	v_0	v_5	v_1	v_2	v_0	v_3	v_4	v_{10}	v_9
$\mathcal{N}_{13}(\Pi)$	v_0	v_7	v_4	v_6	v_0	v_5	v_1	v_2	v_0	v_3	v_8	v_{10}	v_9

- $\mathcal{N}_{14}$: Three positions a, b, and c are chosen uniformly at random and their contents are exchanged by moving the client of a to c, that of c to b, and that of b to a (see Table 4.4). Feasible check-up is made to ensure no depots are consequent.

Table 4.4: An example of $\mathcal{N}_{14}$ with $n = 10$ and $k = 3$, where $a = 1$, $b = 3$, and $c = 9$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}
Π	v_0	v_7	v_8	v_6	v_0	v_5	v_1	v_2	v_0	v_3	v_4	v_{10}	v_9
$\mathcal{N}_{14}(\Pi)$	v_0	v_6	v_8	v_{10}	v_0	v_5	v_1	v_2	v_0	v_3	v_4	v_7	v_9

- $\mathcal{N}_{15}$: Movement intra-tours. A position $a > \frac{n}{2}$ is chosen uniformly at random and its contents are inserted in a random b location in the tour before the tour its current position is located (see Table 4.5). Feasible check-up is made to ensure no depots are consequent.

- $\mathcal{N}_{16}$: Movement intra-tours. A position $a < \frac{n}{3}$ is chosen uniformly at random and its contents are inserted in a random b location in the tour after the tour

Table 4.5: An example of $\mathcal{N}_{15}$ with $n = 10$ and $k = 3$, where $a = 12$, $b = 4$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}
Π	v_0	v_7	v_8	v_6	v_0	v_5	v_1	v_2	v_0	v_3	v_4	v_{10}	v_9
$\mathcal{N}_{15}(\Pi)$	v_0	v_7	v_8	v_{10}	v_6	v_0	v_5	v_1	v_2	v_0	v_3	v_4	v_9

its current position is located (see Table 4.6). Feasible check-up is made to ensure no depots are consequent.

Table 4.6: An example of $\mathcal{N}_{16}$ with $n = 10$ and $k = 3$, where $a = 4$, $b = 12$.

	π_1	π_2	π_3	π_4	π_5	π_6	π_7	π_8	π_9	π_{10}	π_{11}	π_{12}	π_{13}
Π	v_0	v_7	v_8	v_6	v_0	v_5	v_1	v_2	v_0	v_3	v_4	v_{10}	v_9
$\mathcal{N}_{16}(\Pi)$	v_0	v_7	v_8	v_0	v_5	v_1	v_2	v_0	v_3	v_4	v_6	v_{10}	v_9

4.4.3.6 Neighborhood performance

The performance of the neighborhood selection remains unchanged to the single-vehicle version of EiLS.

4.4.4 Automatically adjusting evolutionary algorithm with Intelligent Local Search Algorithm for k-MLDP

In this section, we discuss the adjustments made for the single-vehicle version of our proposed self-adjusting evolutionary algorithm (cf. Section 3.4.4), to generate a multi-vehicle version.

4.4.4.1 Initial population

The representation of the solutions of k-MLDP follows the same structure as SMSA and EiLS (cf. Table 4.1). The generation of solutions follows the same structure as in the multi-vehicle version of EiLS (cf. Section 4.4.3).

4.4.4.2 Elite set

The ranking scheme and the elite set construction remains unchanged as in the single-vehicle version of AEiLS.

4.4.4.3 Offsprings

The crossover operator follows the same guidelines as the crossover of the multi-vehicle version of EiLS (cf. Section 4.4.3.3); but to each solution generated with such crossover, the structure of all counters detailed for the single-vehicle version of AEiLS (cf. Section 3.4.4.3) remains unchanged.

4.4.4.4 Ranking and crowding scheme

The ranking and crowding scheme remains unchanged as in the single-vehicle version of AEiLS (cf. Section 3.4.4.4).

4.4.4.5 Mutation

The mutation operator for the multi-vehicle version of AEiLS remains unchanged.

4.4.4.6 Neighborhood exploration

All sixteen neighborhoods detailed for the multi-vehicle version of EiLS (cf. 4.4.3.5) are included in AEiLS, along the restriction that that no two depot can be consequent in the resulting solution. The exploration process is unchanged of the single-vehicle version of EiLS.

4.4.4.7 Neighborhood performance

The performance of the neighborhood selection remains unchanged to the single-vehicle version of AEiLS (cf. Section 3.4.4.7).

4.4.4.8 Autocalibration of parameters

The structure to measure the obtained success of AEiLS in its single-vehicle version (cf. Section 3.4.4.8), remains unchanged for the multi-vehicle version.

4.5 Experimentation

In this section, we study the performance of our four proposals: Augmecon2, SMSA, EiLS, and AEiLS, all adjusted for the multi-vehicle case.

4.5.1 Trial instances

In the experimentation for k-MLDP, we employ the instances described in Section 3.5.1, varying the number of k vehicles per instance, independent of the size of each instance: $k \in \{\lfloor n/10 \rfloor, \lfloor n/8 \rfloor, \lfloor n/5 \rfloor, \lfloor n/4 \rfloor, \lfloor n/3 \rfloor, \lfloor n/2 \rfloor\}$, and
$n \in \{10, 16, 20, 32, 40, 64, 80, 128, 160, 256\}$ clients, each with 35 instances.

4.5.2 Measurements of the quality of the obtained fronts

To compare the performance of all our proposed metaheuristics against the exact Pareto fronts obtained with the Augmecon2 algorithm, we consider five metrics to measure the quality of the solutions.

- the number of points (#),
- the size of the space covered (SCC),
- k-distance (k-D),
- the distance between the exact Pareto front and the approximate Pareto front (M_1^*),
- the coverage of the fronts (c($\circ$,$\circ$)).

These metrics are shown on Table 3.9 on page 93.

4.5.3 Technical specifications

Our implementation of Augmecon2 was solved using the MILP solver ILOG CPLEX C++ Concert Technology. All our heuristic procedures are implemented in C++ and every instance was solved in a Xenon® Intel® CPU E3-1245 v3 @ 3.40GHz, with 16 GB of RAM with each of these methods.

4.5.4 Parameter calibration

The parameter selection is the following:

- Augmecon2 (Section 4.4.1): step size is fixed as $\delta = 1/2n$, where n denotes the size of the instance.

- SMSA (Section 4.4.2): we vary the size of the population (γ in Algorithm 1), the probability to mutate a solution (β in Algorithm 1), as well as number of iterations (I in Algorithm 2). All this with a fixed $R = 5$.

- EiLS (Section 4.4.3): we fix $\chi_{\max} = \gamma$, $\tau = 7$, $\varepsilon = \lceil \gamma/2 \rceil$ and $\kappa = \lceil \varepsilon/2 \rceil$. We vary the size of the population (γ in Algorithm 3), the number of evaluations per module (ϕ in Algorithm 3) as well as the minimum success in the exploration of each neighborhood (α in Algorithm 3).

- AEiLS (Section 4.4.4): we fixed the size of the elite set to $\varepsilon = \lceil \gamma/2 \rceil$ and the number of couples to generate offsprings to $\kappa = \lceil \varepsilon/2 \rceil$.

The varied parameters were calibrated under a factorial design of experiments and the combination of parameters where the best quality in the approximated fronts was obtained was selected as the fixed combination for further experiments. For each of the 35 replicas, we perform 10 executions, and for each of the 350 resulting fronts we compute the *number of points* (#) in the resulting front, the *size of space covered* (SCC). For both methods, a single parameter combination produced the best value for both metrics (i.e., the highest number of points in the front and the largest hypervolume). Tables 4.7 and 4.8 show the quality obtained with each combination of parameters.

Table 4.7: Factorial design for parameter calibration in SMSA; n is the size of the instance. For the SMSA algorithm we tested the size of the initial population (γ), the probability to mutate a solution (β), as well as number of iterations without success (I).

				SMSA											
k				$\lfloor n/10 \rfloor$		$\lfloor n/8 \rfloor$		$\lfloor n/5 \rfloor$		$\lfloor n/4 \rfloor$		$\lfloor n/3 \rfloor$		$\lfloor n/2 \rfloor$	
Label	γ	β	I	#	SCC	#	SCC	#	SCC	#	SCC	#	SCC	#	SCC
1	$5n$	0.05	$n/5$	5.60	0.452	6.09	0.693	5.67	0.462	5.99	0.607	6.49	0.613	5.62	0.751
2	$5n$	0.05	$n/10$	5.86	0.462	6.12	0.697	5.74	0.642	6.13	0.610	6.52	0.685	6.19	0.688
3	$5n$	0.03	$n/5$	6.28	0.642	6.94	0.726	5.81	0.676	7.04	0.743	6.94	0.707	6.54	0.713
4	$5n$	0.03	$n/10$	6.52	0.734	7.23	0.782	6.42	0.724	7.13	0.825	7.18	0.748	7.05	0.734
5	**$10n$**	**0.05**	**$n/5$**	**7.94**	**0.869**	**7.73**	**0.835**	**7.29**	**0.795**	**7.33**	**0.894**	**7.26**	**0.749**	**7.13**	**0.784**
6	$10n$	0.05	$n/10$	6.95	0.812	5.81	0.776	6.62	0.766	6.51	0.807	6.41	0.694	6.81	0.747
7	$10n$	0.03	$n/5$	5.71	0.793	5.74	0.742	6.14	0.686	6.14	0.750	5.54	0.685	6.43	0.712
8	$10n$	0.03	$n/10$	5.66	0.731	5.67	0.742	5.47	0.635	5.57	0.688	5.17	0.645	6.17	0.673

Table 4.8: Factorial design for parameter calibration in EiLS; n is the size of the instance. For the EiLS algorithm we the size of the population (γ), the number of evaluations per module (ϕ), as well as the minimum success (α).

k				EiLS											
				$\lfloor n/10 \rfloor$		$\lfloor n/8 \rfloor$		$\lfloor n/5 \rfloor$		$\lfloor n/4 \rfloor$		$\lfloor n/3 \rfloor$		$\lfloor n/2 \rfloor$	
Label	γ	ϕ	α	#	SCC	#	SCC	#	SCC	#	SCC	#	SCC	#	SCC
1	n	10	0.3	61.60	0.518	69.29	0.510	85.60	0.494	73.06	0.499	69.87	0.508	61.75	0.700
2	n	10	0.5	66.81	0.531	73.68	0.531	86.96	0.514	78.44	0.500	92.64	0.511	62.81	0.701
3	n	10	0.7	71.41	0.562	83.20	0.536	92.15	0.528	94.48	0.501	120.30	0.522	85.53	0.701
4	n	13	0.3	73.30	0.568	92.15	0.537	97.01	0.530	101.99	0.519	123.59	0.523	101.73	0.707
5	n	13	0.5	76.68	0.569	97.01	0.539	98.25	0.539	120.68	0.519	127.95	0.529	106.05	0.709
6	n	13	0.7	81.58	0.572	98.25	0.542	99.57	0.539	131.38	0.529	128.55	0.532	119.33	0.710
7	n	15	0.3	83.85	0.578	99.57	0.546	102.71	0.540	133.51	0.532	136.01	0.537	129.72	0.713
8	n	15	0.5	90.90	0.583	102.71	0.566	105.63	0.546	134.76	0.539	145.51	0.538	131.32	0.718
9	n	15	0.7	96.32	0.588	105.63	0.570	114.29	0.549	145.73	0.540	154.08	0.538	138.97	0.719
10	n^2	10	0.3	115.27	0.588	111.35	0.589	115.91	0.563	147.99	0.553	159.55	0.543	156.16	0.748
11	n^2	**10**	**0.5**	**159.98**	**0.658**	**150.80**	**0.597**	**161.47**	**0.573**	**161.41**	**0.560**	**160.03**	**0.549**	**158.36**	**0.754**
12	n^2	10	0.7	152.11	0.647	142.07	0.588	155.63	0.570	146.50	0.549	151.75	0.537	146.03	0.699
13	n^2	13	0.3	146.03	0.641	135.28	0.583	152.11	0.567	138.07	0.549	151.46	0.531	144.35	0.697
14	n^2	13	0.5	144.35	0.634	133.69	0.578	146.03	0.562	135.37	0.543	150.39	0.530	140.53	0.689
15	n^2	13	0.7	140.53	0.628	131.71	0.572	140.53	0.558	130.32	0.543	147.29	0.529	133.25	0.688
16	n^2	15	0.3	133.25	0.618	129.22	0.569	133.25	0.551	127.54	0.542	143.91	0.528	128.87	0.688
17	n^2	15	0.5	128.87	0.618	128.49	0.568	125.32	0.502	121.53	0.534	143.06	0.538	125.32	0.683
18	n^2	15	0.7	125.32	0.600	123.19	0.562	121.80	0.500	108.24	0.523	142.07	0.530	121.80	0.682
19	n^3	10	0.3	121.80	0.600	123.18	0.531	119.33	0.492	106.77	0.521	135.28	0.529	119.33	0.661
20	n^3	10	0.5	90.90	0.588	110.03	0.518	114.21	0.490	103.34	0.519	133.69	0.529	114.21	0.658
21	n^3	10	0.7	83.85	0.588	108.87	0.503	109.59	0.483	102.69	0.503	131.71	0.521	113.14	0.658
22	n^3	13	0.3	81.58	0.518	102.75	0.500	90.90	0.480	97.77	0.483	129.22	0.518	109.59	0.652
23	n^3	13	0.5	76.68	0.513	97.98	0.499	83.85	0.477	92.62	0.475	128.49	0.511	90.90	0.641
24	n^3	13	0.7	73.30	0.502	94.12	0.498	81.58	0.477	92.37	0.470	123.19	0.511	83.85	0.639
25	n^3	15	0.3	71.41	0.499	84.96	0.482	76.68	0.473	86.92	0.470	123.18	0.499	81.58	0.638
26	n^3	15	0.5	66.81	0.478	83.69	0.481	73.30	0.463	81.53	0.464	110.03	0.499	76.68	0.631
27	n^3	15	0.7	61.60	0.473	77.47	0.481	71.41	0.460	72.27	0.460	108.87	0.498	73.30	0.628

The calibration of parameters for SMSA is shown in Tables 4.7, and for EiLS in Table 4.8. Both tables highlight the combination of parameters that give, in average, the highest hypervolume of the obtained fronts. With this factorial design, the combination of parameters was selected to start the experimentation analysis.

4.5.5 Computational experimentation

Using the calibrated parameter selection, computational experimentation was carried out to compare the methods: the results are presented in two sets: small instances (up to 40 vertices) and large instances (up to 256 vertices), in terms of average values for each size, over sets of 35 instances.

4.5.5.1 Small instances

We consider as small instances those sets that were able to be solved with the exact methodology reported in Section 4.4.1, able to solve sets of up to 40 clients. CPU times Elapsed with all methodologies are reported in Table 4.9.

CPU times are reported in Table 4.9. Note that the number of vehicles affects the performance of EiLS and AEiLS. In smaller instances, EiLS seems faster, but when the instances grow, the effectiveness of AEiLS also grows. In smaller instances with fewer vehicles, EiLS has better performance. In larger instances, regardless of the number of vehicles, AEiLS is more efficient.

Table 4.9: Average CPU times obtained by the four proposed solution methods. The times are averages over set of instances, measured in seconds.

Instance	n	k	Augmecon2	SMSA	EiLS	AEiLS
MLDP_10	10	$\lfloor n/10 \rfloor$	34.561	2.475	0.074	**0.068**
		$\lfloor n/8 \rfloor$	34.561	2.949	**0.068**	0.074
		$\lfloor n/5 \rfloor$	23.964	2.987	**0.073**	0.083
		$\lfloor n/4 \rfloor$	23.964	3.217	**0.081**	0.099
		$\lfloor n/3 \rfloor$	20.464	3.504	**0.173**	0.202
		$\lfloor n/2 \rfloor$	20.328	3.726	**0.305**	0.309
MLDP_16	16	$\lfloor n/10 \rfloor$	233.838	6.443	0.315	**0.137**
		$\lfloor n/8 \rfloor$	204.319	7.637	**0.168**	0.182
		$\lfloor n/5 \rfloor$	170.442	8.322	3.825	**1.358**
		$\lfloor n/4 \rfloor$	148.872	10.544	4.950	**2.529**
		$\lfloor n/3 \rfloor$	121.072	13.635	6.428	**5.638**
		$\lfloor n/2 \rfloor$	109.494	13.725	**3.354**	8.445
MLDP_20	20	$\lfloor n/10 \rfloor$	1729.560	18.261	5.428	**3.173**
		$\lfloor n/8 \rfloor$	1729.560	20.277	5.428	**3.173**
		$\lfloor n/5 \rfloor$	1529.322	23.567	**8.065**	8.643
		$\lfloor n/4 \rfloor$	1289.083	28.377	**10.925**	11.912
		$\lfloor n/3 \rfloor$	907.364	34.134	**9.718**	14.662
		$\lfloor n/2 \rfloor$	827.483	37.163	20.046	**16.655**
MLDP_32	32	$\lfloor n/10 \rfloor$	29193.180	270.816	137.104	**106.283**
		$\lfloor n/8 \rfloor$	27345.543	302.543	138.481	**136.595**
		$\lfloor n/5 \rfloor$	24080.765	371.741	220.678	**137.990**
		$\lfloor n/4 \rfloor$	22194.666	464.359	284.370	**183.074**
		$\lfloor n/3 \rfloor$	18157.022	570.936	299.173	**192.253**
		$\lfloor n/2 \rfloor$	18048.120	605.256	346.299	**196.895**
MLDP_40	40	$\lfloor n/10 \rfloor$	489551.553	595.373	301.325	**204.618**
		$\lfloor n/8 \rfloor$	466971.353	731.482	311.941	**214.353**
		$\lfloor n/5 \rfloor$	296044.833	834.416	542.636	**261.547**
		$\lfloor n/4 \rfloor$	172222.754	1057.544	561.117	**263.814**
		$\lfloor n/3 \rfloor$	96946.754	1192.363	633.107	**372.183**
		$\lfloor n/2 \rfloor$	43147.795	1212.081	637.744	**416.273**

Tables 4.10 and 4.11 show the average values of quality metrics. Similarly,

Table 4.10: Comparison among the metrics obtained per algorithm, averaged over the set of instances.

			Augmecon2				SMSA				EiLS				AEiLS			
Instance	n	k	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*
MLDP_10	10		5.12	0.884	0.0970	0	8.81	0.829	0.1977	0.2967	**45.48**	**0.922**	**0.0003**	0.1777	29.91	0.918	0.0006	**0.0321**
MLDP_16	16		5.54	0.907	0.4601	0	9.27	0.842	0.1361	0.2968	**66.86**	0.923	**0.0003**	0.1602	53.14	**0.934**	0.0003	**0.0290**
MLDP_20	20	$\lfloor n/10 \rfloor$	11.57	0.907	0.8104	0	9.38	0.838	0.1513	0.2983	**80.19**	**0.922**	**0.0005**	0.1450	69.61	0.916	0.0015	**0.0247**
MLDP_32	32		12.66	0.929	1.1851	0	10.93	0.842	0.1555	0.2984	105.10	**0.943**	**0.0007**	0.1700	**113.90**	0.919	0.0083	**0.1444**
MLDP_40	40		13.52	0.948	1.3540	0	11.76	0.837	0.1605	0.2993	169.69	**0.943**	0.0006	**0.1245**	**202.48**	0.906	**0.0003**	0.1313
MLDP_10	10		5.31	0.899	0.2393	0	8.65	0.832	0.1927	0.2992	**33.37**	0.930	0.0006	**0.0233**	31.20	**0.938**	**0.0004**	0.0419
MLDP_16	16		6.54	0.902	0.0701	0	9.07	0.832	0.0938	0.2964	**52.04**	**0.944**	0.0006	0.0633	50.95	0.908	**0.0001**	**0.0249**
MLDP_20	20	$\lfloor n/8 \rfloor$	13.04	0.931	0.2926	0	9.66	0.830	0.1157	0.2987	61.61	**0.942**	**0.0001**	**0.0203**	**94.98**	0.927	0.0009	0.0393
MLDP_32	32		8.35	0.934	1.3725	0	11.22	0.829	0.1763	0.2988	**127.93**	0.912	0.0009	**0.0512**	113.86	**0.942**	**0.0001**	0.0761
MLDP_40	40		11.09	0.969	0.1191	0	11.81	0.832	0.0989	0.2972	228.78	0.893	0.0009	0.0463	**294.43**	**0.922**	**0.0003**	**0.0462**
MLDP_10	10		8.20	0.904	0.5669	0	9.17	0.835	0.1743	0.2978	**32.47**	**0.948**	**0.0015**	0.1675	25.32	0.938	0.0019	**0.1393**
MLDP_16	16		8.58	0.894	0.8349	0	9.69	0.847	0.1236	0.2972	**75.09**	0.923	**0.0018**	**0.1092**	66.07	**0.927**	0.0019	0.1512
MLDP_20	20	$\lfloor n/5 \rfloor$	9.26	0.962	0.2289	0	11.12	0.829	0.1866	0.2963	**97.69**	**0.939**	0.0013	**0.1363**	89.93	0.942	**0.0011**	0.1463
MLDP_32	32		11.28	0.948	0.7019	0	11.30	0.838	0.0838	0.2977	106.15	0.913	**0.0010**	**0.1494**	**237.29**	**0.926**	0.0012	0.1495
MLDP_40	40		13.20	0.932	1.1954	0	11.38	0.845	0.1842	0.2976	349.06	0.941	**0.0010**	**0.1320**	**374.49**	**0.942**	0.0013	0.1405

Table 4.11: Comparison among the metrics obtained per algorithm, averaged over the set of instances.

			Augmecon2				SMSA				EiLS				AEiLS			
Instance	n	k	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*	#	SCC	k-D	M_1^*
MLDP_10	10		4.86	0.977	0.7209	0	8.63	0.831	0.0831	0.2993	38.26	**0.942**	0.0017	0.0515	**39.49**	0.935	**0.0007**	**0.0325**
MLDP_16	16		5.75	0.939	0.3491	0	9.45	0.845	0.1679	0.2992	49.43	**0.927**	0.0016	**0.0312**	**54.59**	0.912	**0.0008**	0.0378
MLDP_20	20	$\lfloor n/4 \rfloor$	7.68	0.916	0.2887	0	9.76	0.844	0.0973	0.2996	72.91	0.916	0.0011	**0.0254**	**80.62**	**0.918**	**0.0007**	0.1846
MLDP_32	32		8.04	0.934	0.3933	0	10.87	0.845	0.1869	0.2929	81.34	0.913	**0.0009**	0.1138	**186.73**	**0.932**	0.0016	**0.0092**
MLDP_40	40		10.28	0.878	0.6946	0	10.91	0.831	0.1992	0.2959	340.91	0.901	**0.0009**	0.1184	**402.66**	**0.949**	0.0018	**0.0689**
MLDP_10	10		5.64	0.948	0.4404	0	8.92	0.846	0.1998	0.2985	38.46	**0.929**	**0.0001**	0.1114	**39.33**	0.927	0.0007	**0.0895**
MLDP_16	16		7.51	0.932	0.7628	0	10.40	0.847	0.0782	0.2991	50.49	0.918	**0.0007**	**0.0426**	**64.20**	**0.938**	0.0008	0.0969
MLDP_20	20	$\lfloor n/3 \rfloor$	7.85	0.957	0.5998	0	8.62	0.842	0.1778	0.2988	87.22	0.892	0.0007	0.1003	**108.72**	**0.941**	**0.0002**	**0.0708**
MLDP_32	32		7.92	0.957	1.2805	0	9.52	0.847	0.2065	0.2967	267.94	0.907	0.0009	0.1130	**275.71**	**0.930**	**0.0006**	**0.0047**
MLDP_40	40		12.74	0.909	0.2133	0	9.75	0.849	0.2118	0.2983	349.39	0.881	0.0009	0.1262	**401.67**	**0.944**	**0.0001**	**0.0288**
MLDP_10	10		4.75	0.966	1.1074	0	8.62	0.844	0.8439	0.3035	39.52	0.930	**0.0001**	0.1701	**37.85**	**0.934**	0.0007	**0.0895**
MLDP_16	16		5.11	0.923	0.1957	0	9.52	0.834	0.8343	0.2296	**51.97**	0.944	**0.0007**	0.1596	47.94	**0.947**	0.0008	**0.0969**
MLDP_20	20	$\lfloor n/2 \rfloor$	5.25	0.919	1.3171	0	9.75	0.849	0.8487	0.2965	68.88	0.925	**0.0007**	**0.1003**	**93.18**	**0.934**	0.0010	0.1509
MLDP_32	32		8.78	0.890	0.6242	0	10.08	0.847	0.8475	0.1429	98.62	0.914	**0.0009**	**0.1130**	**111.50**	**0.922**	0.0009	0.1349
MLDP_40	40		9.38	0.924	1.1778	0	9.78	0.831	0.8312	0.2983	240.91	0.918	0.0009	0.1262	**294.76**	**0.921**	**0.0001**	**0.1031**

both tables are grouped by the number of vehicles. Note that for few clients and few vehicles, EiLS yield better results in terms of the number of points (#), SCC and a good spread (k-D). When the instances and the amount of vehicles are larger, AEiLS found more points in the fronts (#), a larger SCC amount, and a better spread (k-D).

It is clear that in small instances and fewer vehicles, EiLS outperforms the other algorithms. AEiLS has a good SCC and a good performance when instances are larger and involve more vehicles. With larger instances AEiLS has time to find a better calibration and reaches better results.

The coverage of the fronts is presented in Table 4.12. EiLS and AEiLS overperform the SMSA. As Augmecon2 is able to find optimal solutions but it may not find non-supported points in the Pareto front,it produces less points and less variability; hence, we only report coverages for SMSA, EiLS and AEiLS. For small instances, EiLS and AEiLS are able to provide good-quality approximations to the Pareto fronts obtained by an exact methodology in a short computational time.

Table 4.12: Comparison among the metrics obtained per algorithm, averaged over the set of instances.

Instance	n	k	c(SMSA,EiLS)	c(SMSA,AEiLS)	c(EiLS,SMSA)	c(EiLS,AEiLS)	c(AEiLS,SMSA)	c(AEiLS,EiLS)
MLDP_10	10	$\lfloor n/10 \rfloor$	0.197	0.176	0.891	0.037	**0.912**	0.018
MLDP_16	16		0.183	0.111	**0.826**	0.031	0.819	0.013
MLDP_20	20		0.145	0.153	**0.817**	0.036	0.813	0.002
MLDP_32	32		0.062	0.041	0.779	0.023	**0.784**	0.001
MLDP_40	40		<0.001	<0.001	**0.743**	0.004	0.732	0.001
MLDP_10	10	$\lfloor n/8 \rfloor$	0.187	0.139	**0.717**	0.036	0.714	0.001
MLDP_16	16		0.152	0.146	**0.789**	0.033	0.748	0.001
MLDP_20	20		0.174	0.157	0.698	0.028	**0.799**	0.014
MLDP_32	32		0.054	0.068	0.792	0.019	**0.809**	0.009
MLDP_40	40		<0.001	<0.001	0.842	0.003	**0.901**	0.016
MLDP_10	10	$\lfloor n/5 \rfloor$	0.173	0.091	0.699	0.003	**0.765**	0.001
MLDP_16	16		0.176	0.092	0.723	0.013	**0.774**	0.002
MLDP_20	20		0.159	0.124	0.778	0.031	**0.841**	0.006
MLDP_32	32		0.056	0.051	0.817	0.034	**0.834**	0.012
MLDP_40	40		<0.001	<0.001	0.831	0.038	**0.909**	0.015
MLDP_10	10	$\lfloor n/4 \rfloor$	0.147	0.091	0.719	0.004	**0.768**	0.001
MLDP_16	16		0.106	0.124	0.748	0.011	**0.783**	0.005
MLDP_20	20		0.154	0.092	0.752	0.033	**0.834**	0.012
MLDP_32	32		0.052	0.049	0.829	0.032	**0.839**	0.013
MLDP_40	40		<0.001	<0.001	0.835	0.035	**0.901**	0.016
MLDP_10	10	$\lfloor n/3 \rfloor$	0.148	0.096	0.702	0.001	**0.753**	0.001
MLDP_16	16		0.146	0.099	0.737	0.012	**0.758**	0.004
MLDP_20	20		0.101	0.102	0.757	0.018	**0.821**	0.011
MLDP_32	32		0.068	0.061	0.817	0.028	**0.847**	0.013
MLDP_40	40		<0.001	<0.001	0.831	0.031	**0.911**	0.016
MLDP_10	10	$\lfloor n/2 \rfloor$	0.147	0.093	0.694	0.001	**0.798**	0.001
MLDP_16	16		0.145	0.097	0.701	0.009	**0.803**	0.006
MLDP_20	20		0.059	0.046	0.734	0.013	**0.816**	0.011
MLDP_32	32		0.054	0.106	0.793	0.025	**0.834**	0.015
MLDP_40	40		<0.001	<0.001	0.794	0.028	**0.891**	0.015

4.5.5.2 Large instances

In this section we report the results obtained with larger instances with 64, 80, 128, 160, and 256 clients. As our implementation of an exact methodology is not able to solve these instances sizes, the comparison is done between our three metaheuristic approaches: SMSA, EiLS, and AEiLS. All results are reported as averages over the set of instances of each size.

Table 4.13 shows the comparison in CPU time consumed in obtaining these fronts. EiLS and AEiLS have a better performance than SMSA, only consuming about half the time SMSA takes to approximate the Pareto fronts. AEiLS has a better CPU time than EiLS, above all, for the larger sizes. The more vehicles are available, the less time AEiLS takes: when more vehicles are available, the latency objective is less restricted and less computational heavy . An ideal solution for the MLP is the one when there is a vehicle for each client, the complete opposite for the ideal solution of TSP in which doing so would give a really expensive solution (Blum et al., 1994).

To compare the effectiveness these algorithms have in larger instances, Tables 4.14 and 4.15 present the average value per set of instances of the quality-metrics #, SCC, k-D, and M_1^*. With these tables it is shown that AEiLS outperforms EiLS and SMSA. It is able to find more points in the fronts (#) and a larger SCC amount, and a better spread in the fronts (M_1^*). It is clear that in larger instances, AEiLS has enough time to calibrate itself and give good-quality fronts.

Table 4.13: Average CPU times obtained by our four proposed solution methods in averages over each set of instances, measured in seconds.

Instance	n	k	SMSA	EiLS	AEiLS
MLDP_64	64	$\lfloor n/10 \rfloor$	587.227	289.374	**276.384**
		$\lfloor n/8 \rfloor$	479.372	282.264	**270.975**
		$\lfloor n/5 \rfloor$	477.275	277.845	**260.975**
		$\lfloor n/4 \rfloor$	465.283	273.975	**247.842**
		$\lfloor n/3 \rfloor$	432.372	268.863	**235.859**
		$\lfloor n/2 \rfloor$	401.438	263.372	**215.372**
MLDP_80	80	$\lfloor n/10 \rfloor$	1001.028	367.277	**300.474**
		$\lfloor n/8 \rfloor$	994.473	359.483	**293.489**
		$\lfloor n/5 \rfloor$	979.372	351.374	**286.483**
		$\lfloor n/4 \rfloor$	981.273	348.374	**274.389**
		$\lfloor n/3 \rfloor$	976.372	343.234	**262.375**
		$\lfloor n/2 \rfloor$	967.373	338.364	**259.857**
MLDP_128	128	$\lfloor n/10 \rfloor$	1419.411	577.277	**499.274**
		$\lfloor n/8 \rfloor$	1413.372	569.483	**485.273**
		$\lfloor n/5 \rfloor$	1383.573	561.374	**479.273**
		$\lfloor n/4 \rfloor$	1288.746	558.374	**464.372**
		$\lfloor n/3 \rfloor$	1198.373	543.234	**465.281**
		$\lfloor n/2 \rfloor$	1134.274	538.364	**453.476**
MLDP_160	160	$\lfloor n/10 \rfloor$	1924.739	913.473	**875.273**
		$\lfloor n/8 \rfloor$	1910.372	907.273	**871.765**
		$\lfloor n/5 \rfloor$	1903.247	900.372	**865.865**
		$\lfloor n/4 \rfloor$	1889.372	899.372	**861.499**
		$\lfloor n/3 \rfloor$	1873.328	892.372	**856.754**
		$\lfloor n/2 \rfloor$	1869.327	879.428	**845.362**
MLDP_256	256	$\lfloor n/10 \rfloor$	2323.624	1199.362	**932.472**
		$\lfloor n/8 \rfloor$	2314.382	1092.373	**927.942**
		$\lfloor n/5 \rfloor$	2267.972	1078.853	**915.483**
		$\lfloor n/4 \rfloor$	2249.273	1066.964	**903.479**
		$\lfloor n/3 \rfloor$	2243.846	1064.119	**899.327**
		$\lfloor n/2 \rfloor$	2199.579	1063.374	**893.437**

Table 4.14: Comparison among the metrics obtained per algorithm, in averages over each set of instances

			SMSA			EiLS			AEiLS		
Instance	n	k	#	SCC	k-D	#	SCC	k-D	#	SCC	k-D
MLDP_64	64	$\lfloor n/10 \rfloor$	18.44	0.858	0.1157	695.49	0.921	0.0017	**889.92**	**0.928**	**0.0007**
MLDP_80	80		31.26	0.842	0.1763	899.86	0.913	0.0016	**1053.13**	**0.914**	**0.0008**
MLDP_128	128		46.92	0.822	0.0989	1180.19	0.902	0.0011	**1269.61**	**0.901**	**0.0007**
MLDP_160	160		54.24	0.820	0.1745	1565.10	0.893	0.0009	**1673.90**	**0.901**	**0.0001**
MLDP_256	256		82.37	0.815	0.1236	1839.69	0.890	0.0009	**1922.48**	**0.896**	**0.0018**
MLDP_64	64	$\lfloor n/8 \rfloor$	23.90	0.851	0.1763	673.37	0.921	0.0015	**899.20**	**0.948**	**0.0007**
MLDP_80	80		33.57	0.832	0.0989	902.04	0.915	0.0018	**1050.95**	**0.939**	**0.0008**
MLDP_128	128		46.99	0.836	0.1742	1261.61	0.902	0.0013	**1294.98**	**0.926**	**0.0002**
MLDP_160	160		58.26	0.820	0.1236	1567.93	0.900	0.0010	**1693.86**	**0.923**	**0.0006**
MLDP_256	256		89.36	0.802	0.0989	1898.77	0.893	0.0010	**1994.43**	**0.912**	**0.0001**
MLDP_64	64	$\lfloor n/5 \rfloor$	26.31	0.865	0.0831	682.47	0.918	0.0006	**902.44**	**0.918**	**0.0007**
MLDP_80	80		32.19	0.850	0.1679	879.09	0.913	0.0006	**1066.07**	**0.913**	**0.0008**
MLDP_128	128		43.47	0.835	0.0973	1297.69	0.909	0.0001	**1389.93**	**0.902**	**0.0010**
MLDP_160	160		57.82	0.820	0.1870	1656.15	0.900	0.0009	**1701.29**	**0.902**	**0.0009**
MLDP_256	256		92.37	0.828	0.1992	1901.06	0.891	0.0009	**1974.49**	**0.892**	**0.0001**

Table 4.15: Comparison among the metrics obtained per algorithm, in averages over each set of instances.

Instance	n	k	#	SCC	k-D	#	SCC	k-D	#	SCC	k-D
MLDP_64	64		26.99	0.839	0.1869	698.26	0.922	**0.0001**	**859.42**	**0.935**	0.0019
MLDP_80	80		31.48	0.835	0.1992	919.43	0.917	**0.0007**	**1054.59**	**0.920**	0.0019
MLDP_128	128	$\lfloor n/4 \rfloor$	49.72	0.832	0.1998	1272.91	0.906	**0.0007**	**1301.62**	**0.914**	0.0011
MLDP_160	160		58.94	0.825	0.0782	1681.33	0.894	**0.0009**	**1686.73**	**0.903**	0.0012
MLDP_256	256		92.41	0.821	0.1778	1905.91	0.891	**0.0009**	**1995.66**	**0.898**	0.0013
MLDP_64	64		28.79	0.853	0.1998	699.46	**0.929**	**0.0001**	**891.36**	0.922	0.0006
MLDP_80	80		34.36	0.846	0.0782	920.49	0.911	0.0007	**1064.20**	**0.913**	**0.0003**
MLDP_128	128	$\lfloor n/3 \rfloor$	48.62	0.842	0.1778	1297.22	0.893	**0.0007**	**1308.72**	**0.910**	0.0015
MLDP_160	160		60.38	0.835	0.2065	1593.94	0.890	**0.0009**	**1775.70**	**0.900**	0.0083
MLDP_256	256		92.36	0.829	0.2118	1940.39	0.881	0.0009	**1999.67**	**0.890**	**0.0003**
MLDP_64	64		28.59	0.856	0.2439	709.52	**0.930**	**0.0003**	**901.85**	0.914	0.0004
MLDP_80	80		36.83	0.845	0.1343	931.97	**0.912**	0.0003	**1067.94**	0.907	**0.0001**
MLDP_128	128	$\lfloor n/2 \rfloor$	49.75	0.839	0.2487	1268.88	**0.905**	**0.0005**	**1393.18**	0.894	0.0009
MLDP_160	160		56.08	0.835	0.1475	1688.65	0.894	0.0007	**1691.50**	**0.895**	**0.0001**
MLDP_256	256		93.78	0.821	0.0112	1924.92	0.892	0.0006	**2004.76**	**0.892**	**0.0003**

Tables 4.16 and 4.17 show the coverage of the fronts and allow us to highlight the superiority, in larger instances, of both: EiLS and AEiLS algorithms over the SMSA. Our experimentation in larger instances have a tendency to point out that AEiLS, despite the number of vehicles, tends to outperform other methods.

Table 4.16: Comparison among the metrics obtained per algorithm, as averages over each set of instances

Instance	n	k	c(SMSA,EiLS)	c(SMSA,AEiLS)	c(EiLS,SMSA)	c(EiLS,AEiLS)	c(AEiLS,SMSA)	c(AEiLS,EiLS)
MLDP_64	64	$\lfloor n/10 \rfloor$	0.297	0.216	**0.743**	0.021	0.732	0.067
MLDP_80	80		0.245	0.203	0.779	0.021	**0.784**	0.061
MLDP_128	128		0.183	0.201	**0.817**	0.022	0.813	0.056
MLDP_160	160		0.162	0.141	**0.826**	0.026	0.819	0.063
MLDP_256	256		0.121	0.118	0.891	0.028	**0.912**	0.064
MLDP_64	64	$\lfloor n/8 \rfloor$	0.202	0.199	0.698	0.021	**0.714**	0.053
MLDP_80	80		0.194	0.168	0.717	0.021	**0.748**	0.069
MLDP_128	128		0.172	0.146	0.789	0.029	**0.799**	0.068
MLDP_160	160		0.171	0.138	0.792	0.024	**0.809**	0.073
MLDP_256	256		0.167	0.131	0.842	0.026	**0.901**	0.069
MLDP_64	64	$\lfloor n/5 \rfloor$	0.201	0.197	0.699	0.011	**0.765**	0.067
MLDP_80	80		0.193	0.185	0.723	0.012	**0.774**	0.073
MLDP_128	128		0.189	0.184	0.778	0.016	**0.841**	0.071
MLDP_160	160		0.176	0.172	0.817	0.012	**0.837**	0.064
MLDP_256	256		0.173	0.161	0.831	0.015	**0.909**	0.068

Table 4.17: Comparison among the metrics obtained per algorithm, as averages over each set of instances

Instance	n	k	c(SMSA,EiLS)	c(SMSA,AEiLS)	c(EiLS,SMSA)	c(EiLS,AEiLS)	c(AEiLS,SMSA)	c(AEiLS,EiLS)
MLDP_64	64		0.245	0.199	0.719	0.011	**0.768**	0.054
MLDP_80	80		0.222	0.149	0.728	0.015	**0.783**	0.051
MLDP_128	128	$\lfloor n/4 \rfloor$	0.206	0.124	0.755	0.012	**0.834**	0.063
MLDP_160	160		0.154	0.092	0.809	0.013	**0.839**	0.062
MLDP_256	256		0.147	0.091	0.825	0.016	**0.901**	0.055
MLDP_64	64		0.198	0.181	0.702	0.011	**0.853**	0.041
MLDP_80	80		0.182	0.161	0.737	0.024	**0.858**	0.052
MLDP_128	128	$\lfloor n/3 \rfloor$	0.171	0.142	0.757	0.021	**0.821**	0.058
MLDP_160	160		0.166	0.139	0.817	0.023	**0.847**	0.058
MLDP_256	256		0.158	0.126	0.831	0.016	**0.899**	0.061
MLDP_64	64		0.199	0.189	0.694	0.011	**0.898**	0.051
MLDP_80	80		0.189	0.146	0.701	0.016	**0.903**	0.059
MLDP_128	128	$\lfloor n/2 \rfloor$	0.178	0.106	0.734	0.021	**0.896**	0.053
MLDP_160	160		0.165	0.097	0.793	0.015	**0.894**	0.065
MLDP_256	256		0.151	0.093	0.794	0.015	**0.891**	0.068

4.5.6 Conclusions

Smart neighborhood exploration to approximate Pareto fronts of k-MLDP leads to better results than a classical multi-objective technique.

Each solution of k-MLDP is formed by k disjoint tours. For small instances EiLS excels in the number of points obtained, SCC, and a good spread of the front found points. In small instances and fewer vehicles, EiLS has better performance of any of the algorithms.

On the other hand, AEiLS has a good performance when the search space is larger. With larger instances, AEiLS has time to find a better calibration and give better results in the long run.

4.6 Conclusions and future research

The k-Minimum Latency-Distance Problem (k-MLDP) minimizes the travel distance and the total latency of a set of k disjoint tours, each with an infinite capacity. Each one of the k agents must be active and have at least a client assigned. We aim to improve two important aspects of the problem: the economic decisions of the company and the degree the company's clients service, by the minimization of the total waiting time of all the clients.

k-MLDP is a NP hard problem. We proposed a non-linear formulation for this problem, which, by the addition of several linearization variables, it gives a linear formulation that can be solved to obtain the exact Pareto fronts.

With adjustments for multiple vehicles, we experiment on the proposed solution methods Strategic Memetic Search Algorithm (SMSA), Evolutionary algorithm with Intelligent Local Search (EiLS), and Automatically adjusting Evolutionary algorithm with Intelligent Local Search (AEiLS). We found that EiLS and AEiLS, guided by an intelligent selection of a more appropriate neighborhood in the local search procedure, are superior than SMSA, in terms of density of the front, hypervolume percentage, computational time required, and other quality metrics. The number of vehicles directly affects the performance: in smaller instances with fewer vehicles, a manual calibration of parameters suffices to find good solutions, while having larger instances and more vehicles gives opportunity to self-tunning algorithms to calibrate and obtain good solutions.

Again, capacitated and stochastic versions are of interest for future work on k-MLDP.

Chapter 5

Conclusions

This dissertation studies a combination of objectives to pursue not only a company benefit but also a benefit to the clients waiting for a service. In classic routing, an economic approach is widely studied: minimize travel distance to perform the visits. We study the consideration of the satisfaction of the client in the decision-making process.

We part from several assumptions into account: an agent must depart from and return to a known depot, all travel times are known, as well as all service times.

We work with two scenarios: first, assuming there is a single uncapacitated vehicle that visits all clients in a single tour, and second, visiting the clients with a fleet of k uncapacitated vehicles, each visiting at least one client. We call the single-vehicle approach The Minimum Latency-Distance Problem (MLDP), and the fleet approach The k-Minimum Latency-Distance Problem (k-MLDP).

For each one of these problems we proposed a linear and a non-linear formulation, both of them based on two multilevel networks. We solve our linear formulations with our implementation of Augmecon2 to obtain exact fronts of small instances. We studied the complexity of each one of these problems, and proved that both are NP-hard. Hence, we proposed three metaheuristic procedures to compute fronts in larger instances. Strategic Memetic Search Algorithm (SMSA), based on NSGA-II, Evolutionary algorithm with Intelligent Local Search (EiLS), that employs an intelligent neighborhood selection, and Automatically Adjusted Evolutionary Algorithm with Intelligent Local Search (AEiLS) with a novel ranking scheme.

The three metaheuristics were able to solve larger instances of up to 256 vertices each. In smaller instances EiLS is able to quickly provide fronts close to the exact ones, while for larger instances, AEiLS gives the best results as it is able to self-calibrate.

5.1 Future research

For future research some improvements can be considered. One is no longer assuming linearity in the relationship between travel time to distance traveled, which is the case in high-traffic cities and transport by airplanes.

Another is constraining the capacity of the vehicles employed in visiting the clients. Also considering heterogeneous fleets.

The inclusion of stochastic service and travel times is also of interest, as well as the possibility that new clients can be added in real time. The consideration of penalization for not delivering the goods within an established time-window, flexible or not, or the inclusion of work shifts for the drivers of the vehicles is also relevant.

The inclusion of additional objectives, such as the minimization of the size of the fleet or balancing the work among the agents are of interest as future work. Allowing each vehicle to return several times to the depot to refill is also another possible scenario.

Bibliography

F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the travelling repairman problem. *RAIRO – Theoretical Informatics and Applications*, 20(1):79–87, 1986. doi: 10.1051/ita/1986200100791.

J. Aghaei, N. Amjady, and H. A. Shayanfar. Multi-objective electricity market clearing considering dynamic security by lexicographic optimization and augmented epsilon constraint method. *Applied Soft Computing*, 11(4):3846–3858, 2011. doi: 10.1016/j.asoc.2011.02.022.

S. R. Agnihothri. A mean value analysis of the travelling repairman problem. *IIE Transactions*, 20(2):223–230, 1988. doi: 10.1080/07408178808966173.

Z. H. Ahmed. Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator. *International Journal of Biometrics & Bioinformatics*, 3(6):96, 2010. doi: 10.1.1.371.7771.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice Hall, London, England, 1993. ISBN 9780343248604.

F. Angel-Bello, A. M. Álvarez-Socarrás, and I. García. Two improved formulations for the minimum latency problem. *Applied Mathematical Modelling*, 37(4):2257–2266, 2013. doi: 10.1016/j.apm.2012.05.026.

F. Angel-Bello, I. Martínez-Salazar, and A. M. Álvarez-Socarrás. Minimizing waiting times in a route design problem with multiple use of a single vehicle. *Electronic Notes in Discrete Mathematics*, 41:269–276, 2013. doi: 10.1016/j.endm.2013.05.102.

F. Angel-Bello, Y. Cardona-Valdés, and A. Álvarez-Socarrás. Mixed integer formulations for the multiple minimum latency problem. *Operational Research*, 23(1): 1–30, 2017. doi: 10.1007/s12351-017-0299-4.

D. Angus. Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, pages 333–340. IEEE, 2007. doi: 10.1109/MCDM.2007.369110.

D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, Princeton, New Jersey, USA, 2007. ISBN 9780691129938.

A. Archer and D. P. Williamson. Faster approximation algorithms for the minimum latency problem. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 88–96. SIAM, 2003. ISBN 0898715385.

A. Archer, A. Levin, and D. P. Williamson. A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing*, 37(5):1472–1498, 2008. doi: 10.1137/07068151X.

N. Arellano-Arriaga and S. E. Schaeffer. MLDP instance set. `https://doi.org/10.6084/m9.figshare.5584816.v1`, 2017. [Online].

N. A. Arellano-Arriaga, A. M. Álvarez-Socarrás, and I. A. Martínez-Salazar. A sustainable bi-objective approach for the minimum latency problem. In E. Alba,

F. Chicano, and G. Luque, editors, *Smart Cities*, volume 10268 of *Lecture Notes in Computer Science*, pages 11–19, Cham, Switzerland, 2017. Springer. doi: 10.1007/978-3-319-59513-9_2.

N. A. Arellano-Arriaga, J. Molina, S. E. Schaeffer, A. M. Álvarez-Socarrás, and I. A. Martínez-Salazar. Complexity of MLDP. Technical Report arXiv:1809.02656 [cs.CC], arXiv.org, Sept. 2018.

N. A. Arellano-Arriaga, J. Molina, S. E. Schaeffer, A. M. Álvarez-Socarrás, and I. A. Martínez-Salazar. A bi-objective study of the minimum latency problem. *Journal of Heuristics*, 2019. Accepted for publication.

S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998. doi: 10.1145/290179.290180.

S. Arora and G. Karakostas. Approximation schemes for minimum latency problems. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 688–693. ACM, 1999. doi: 10.1137/S0097539701399654.

A. Asadpour, M. X. Goemans, A. Mądry, S. O. Gharan, and A. Saberi. An $\wr(logn/loglogn)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 379–389, PA, USA, 2010. SIAM. ISBN 9780898716986.

T. W. Athan and P. Y. Papalambros. A note on weighted criteria methods for compromise solutions in multi-objective optimization. *Engineering Optimization*, 27(2):155–176, 1996. doi: 10.1080/03052159608941404.

G. Ausiello, S. Leonardi, and A. Marchetti-Spaccamela. On salesmen, repairmen, spiders, and other traveling agents. In *Italian Conference on Algorithms and Com-*

plexity, pages 1–16, New York, USA, 2000. Springer. doi: 10.1007/3-540-46521-9_1.

M. G. Avci and M. Avci. An iterated local search algorithm for multiple traveling repairman problem with profits. `sciencesconf.org/141380`, 2017. Unpublished manuscript, online.

W. Bair, J. R. Cavanaugh, and J. A. Movshon. Time course and time-distance relationships for surround suppression in macaque V1 neurons. *Journal of Neuroscience*, 23(20):7690–7701, 2003. doi: 10.1523/JNEUROSCI.23-20-07690.2003.

B. H. Bang and N. D. Nghia. Improved genetic algorithm for minimum latency problem. In *Proceedings of the 2010 Symposium on Information and Communication Technology*, pages 9–15, New York, USA, 2010. ACM. doi: 10.1145/1852611.1852614.

S. Bechikh, L. B. Said, and K. Ghedira. Estimating nadir point in multi-objective optimization using mobile reference points. In *Evolutionary computation (CEC), 2010 IEEE congress on*, pages 1–9, New Jersey, United States, 2010. IEEE. doi: 10.1109/CEC.2010.5586203.

T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006. doi: 10.1016/j.omega.2004.10.004.

T. Bektaş and S. Elmastaş. Solving school bus routing problems through integer programming. *Journal of the Operational Research Society*, 58(12):1599–1604, 2007. doi: 10.1057/palgrave.jors.2602305.

M. Bellmore and G. L. Nemhauser. The traveling salesman problem: A survey. *Operations Research*, 16(3):538–558, 1968. doi: 10.1287/opre.16.3.538.

J. J. Bentley. Fast algorithms for geometric traveling salesman problems. *ORSA Journal on computing*, 4(4):387–411, 1992. doi: 10.1287/ijoc.4.4.387.

J.-F. Bérubé, M. Gendreau, and J.-Y. Potvin. 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, 2009. doi: 10.1016/j.ejor.2007.12.014.

L. Bianchi, L. M. Gambardella, and M. Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. In *Parallel Problem Solving from Nature*, pages 883–892, Berlin, Germany, 2002. Springer. doi: 10.1007/3-540-45712-7_85.

N. Bjelić and D. Popović. Two-phase algorithm for solving heterogeneous travelling repairmen problem with time windows. *International Journal for Traffic and Transport Engineering*, 5(1):22–29, 2015. doi: 10.7708/ijtte.2015.5(1).081.

N. Bjelić, M. Vidović, and D. Popović. Variable neighborhood search algorithm for heterogeneous traveling repairmen problem with time windows. *Expert Systems with Applications*, 40(15):5997–6006, 2013. doi: 10.1016/j.eswa.2013.05.036.

A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In F. T. Leighton and M. Goodrich, editors, *Proceedings of the twenty-sixth annual ACM symposium on Theory of Computing*, pages 163–171, New York, USA, 1994. ACM. ISBN 0-89791-663-8. doi: 10.1145/195058.195125.

S. Bock. Solving the traveling repairman problem on a line with general processing times and deadlines. *European Journal of Operational Research*, 244(3):690–703, 2015. doi: 10.1016/j.ejor.2015.02.009.

S. J. Borah, S. K. Dhurandher, I. Woungang, V. Kumar, and L. Barolli. A multi-objectives based technique for optimized routing in opportunistic networks. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–12, 2017. doi: 10.1007/s12652-017-0462-z.

R. Bowerman, B. Hall, and P. Calamai. A multi-objective optimization approach to urban school bus routing: Formulation and solution method. *Transportation Research Part A: Policy and Practice*, 29(2):107–123, 1995. doi: 10.1007/s10732-011-9184-0.

A. Bozorgi-Amiri, M. Jabalameli, and S. M. Al-e Hashem. A multi-objective robust stochastic programming model for disaster relief logistics under uncertainty. *OR Spectrum*, 35(4):905–933, 2013. doi: 10.1007/s00291-011-0268-x.

T. Bulhoes, R. Sadykov, and E. Uchoa. A branch-and-price algorithm for the minimum latency problem. *Computers and Operations Research*, 93(1):66–78, 2018. doi: 10.1016/j.cor.2018.01.016.

E. Burke, P. Cowling, and R. Keuthen. Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem. *Applications of evolutionary computing*, pages 203–212, 2001. doi: 10.1016/0360-8352(96)00154-4].17.

D. Camara and A. A. Loureiro. A GPS/ant-like routing algorithm for ad hoc networks. In *Wireless Communications and Networking Confernce, 2000*, volume 3, pages 1232–1236, New Jersey, United States, 2000. IEEE. doi: 10.1109/WCNC.2000.904807.

F. Carrabs, J.-F. Cordeau, and G. Laporte. Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing*, 19(4):618–632, 2007. doi: 10.1287/ijoc.1060.0202.

A. E. Carter and C. T. Ragsdale. A new approach to solving the multiple traveling salesperson problem using genetic algorithms. *European journal of operational research*, 175(1):246–257, 2006. doi: 10.1016/j.ejor.2005.04.027.

A. M. Caunhye, X. Nie, and S. Pokharel. Optimization models in emergency logistics: A literature review. *Socio-Economic Planning Sciences*, 46(1):4–13, 2012. doi: 10.1016/j.seps.2011.04.004.

K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In *Proceedings of the fourty-fourth annual IEEE symposium on Foundations of Computer Science*, pages 36–45, Cambridge, MA, USA, 2003. IEEE. doi: 10.1109/SFCS.2003.1238179.

L. Chen and K. Aihara. Chaotic simulated annealing by a neural network model with transient chaos. *Neural networks*, 8(6):915–930, 1995. doi: 10.1016/0893-6080(95) 00033-V.

S.-M. Chen and C.-Y. Chien. Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, 38(12):14439–14450, 2011. doi: 10.1016/j.eswa.2011.04.163.

I.-C. Choi, S.-I. Kim, and H.-S. Kim. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers & Operations Research*, 30(5):773–786, 2003. doi: 10.1016/S0305-0548(02)00050-3.

G. L. Clark, P. F. Kaminski, and D. R. Rink. Consumer complaints: Advice on how companies should respond based on an empirical study. *Journal of Consumer Marketing*, 9(3):5–14, 1992. doi: 10.1108/07363769210035189.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964. doi: 10.1287/opre.12.4.568.

M. Clerc. Discrete particle swarm optimization, illustrated by the traveling salesman problem. In *New optimization techniques in engineering*, pages 219–239. Springer, Berlin, 2004. doi: 10.1007/978-3-540-39930-8_8.

C. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999. doi: 10.1007/BF03325101.

W. Cook. *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, Princeton, New Jersey, 2012. ISBN 9781400839599.

J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo. Vehicle routing. In C. Barnhart and G. Laporte, editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367–428. Elsevier, Amsterdam, 2007. doi: 10.1016/S0927-0507(06)14006-2.

G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812, 1958. doi: 10.1287/opre.6.6.791.

J. Crowcroft, M. Segal, and L. Levin. Improved structures for data collection in static and mobile wireless sensor networks. *Journal of Heuristics*, 21(2):233–256, 2015. doi: 10.1007/s10732-014-9250-5.

K. M. Curtin, G. Voicu, M. T. Rice, and A. Stefanidis. A comparative analysis of traveling salesman solutions from geographic information systems. *Transactions in GIS*, 18(2):286–301, 2014. doi: 10.1111/tgis.12045.

R. F. Da Silva and S. Urrutia. A general vns heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4):203–211, 2010. doi: 10.1016/j.disopt.2010.04.002.

G. Dantzig. *Linear Programming and Extensions*. Princeton landmarks in mathematics and physics. Princeton University Press, New Jersey, USA, 1963. ISBN 9780691059136.

G. Dantzig, R. Fulkerson, and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4): 393–410, 1954. doi: 10.1287/opre.2.4.393.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959. doi: 10.1287/mnsc.6.1.80.

G. Darcis, S. Bouchat, A. Kula, B. Van Driessche, N. Delacourt, C. Vanhulle, V. Avettand-Fenoel, S. De Wit, O. Rohr, C. Rouzioux, et al. Reactivation capacity by latency-reversing agents ex vivo correlates with the size of the HIV-1 reservoir. *AIDS*, 31(2):181–189, 2017. doi: 10.1097/QAD.0000000000001290.

P. Dare and H. Saleh. Gps network design: logistics solution using optimal and near-optimal methods. *Journal of Geodesy*, 74(6):467–478, 2000. doi: 10.1007/s001900000104.

A. Das, S. Gollapudi, A. Kim, D. Panigrahi, and C. Swamy. Minimizing latency in online ride and delivery services. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 379–388. International World Wide Web Conferences Steering Committee, 2018. doi: 10.1287/ijoc.1030.0052.

L. Davis. Applying adaptive algorithms to epistatic domains. In *Proceedings of the ninth international joint conference on Artificial Intelligence*, volume 1, pages

162–164, San Francisco, CA, USA, 1985. Morgan Kaufmann Publishers Inc. ISBN 0-934613-02-8.

K. Deb and K. Miettinen. A review of nadir point estimation procedures using evolutionary approaches: A tale of dimensionality reduction. *Indian Institute of Technology*, 2(008):004, 2008. doi: 10.1.1.716.8133.

K. Deb and K. Miettinen. Nadir point estimation using evolutionary approaches: better accuracy and computational speed through focused search. In *Multiple criteria decision making for sustainable energy and transportation systems*, pages 339–354. Springer, 2010. doi: 10.1007/978-3-642-04045-0_29.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2000. doi: 10.1109/4235.996017.

K. Deb, S. Chaudhuri, and K. Miettinen. Towards estimating nadir objective vector using evolutionary approaches. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 643–650. ACM, 2006. doi: 10.1145/1143997.1144113.

K. Deb, K. Sindhya, and J. Hakanen. Multi-objective optimization. In R. N. Sengupta, A. Gupta, and J. Dutta, editors, *Decision Sciences: Theory and Practice*, chapter 3, pages 145–184. CRC Press, Boca Raton, FL, USA, 2016. doi: 10.1007/0-387-28356-0_10.

G. W. Depuy, M. W. Savelsbergh, J. C. Ammons, and L. F. McGinnis. An integer programming heuristic for component allocation in printed circuit card assembly systems. *Journal of Heuristics*, 7(4):351–369, 2001. doi: 10.1023/A:1011388227723.

T. Dewilde, D. Cattrysse, S. Coene, F. C. Spieksma, and P. Vansteenwegen. Heuristics for the traveling repairman problem with profits. *Computers & Operations Research*, 40(7):1700–1707, 2013. doi: 10.1016/j.cor.2013.01.003.

F. Dikbyk. Multi-repairmen problem for disaster recovery of optical networks. *Sakarya University Journal of Science*, 21(1):47–53, 2017. doi: 10.16984/saufenbilder.283845.

M. Dorigo and L. Gambardella. Ant-q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of ML-95, Twelfth International Conference on Machine Learning*, pages 252–260, California, USA, 2016. doi: 10.1.1.52.7194.

M. Dorigo and L. M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997. doi: 10.1109/4235.585892.

M. Dorigo, M. Birattari, and T. Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006. doi: 10.1109/CEC.1999.782657.

H. Duan and X. Yu. Hybrid ant colony optimization using memetic algorithm for traveling salesman problem. In *Approximate Dynamic Programming and Reinforcement Learning. IEEE International Symposium on*, pages 92–95, New York, USA, 2007. IEEE. doi: 10.1109/ADPRL.2007.368174.

S. B. Ebrahimi. A stochastic multi-objective location-allocation-routing problem for tire supply chain considering sustainability aspects and quantity discounts. *Journal of Cleaner Production*, 198:704–720, 2018. doi: 10.1016/j.jclepro.2018.07.059.

M. Ehrgott. *Multicriteria Optimization.* Springer, New Jersey, USA, 2005. ISBN 9783540276593.

S. Elaoud, J. Teghem, and T. Loukil. Multiple crossover genetic algorithm for the multiobjective traveling salesman problem. *Electronic Notes in Discrete Mathematics*, 36:939–946, 2010. doi: 10.1016/j.endm.2010.05.119.

J. Fakcharoenphol, C. Harrelson, and S. Rao. The k-traveling repairman problem. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 655–664, New York, USA, 2003. SIAM. doi: 10.1145/1290672.1290677.

D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005. doi: 10.1287/trsc.1030.0079.

J. M. Ferrer, M. T. Ortuño, and G. Tirado. A GRASP metaheuristic for humanitarian aid distribution. *Journal of Heuristics*, 22(1):55–87, 2016. doi: 10.1007/s10732-015-9302-5.

C. N. Fiechter. A parallel tabu search algorithm for large traveling salesman problems. *Discrete Applied Mathematics*, 51(3):243–267, 1994. doi: 10.1016/0166-218X(92)00033-I.

W. Fischer, K. Wictorin, A. Björklund, L. Williams, S. Varon, F. Gage, et al. Amelioration of cholinergic neuron atrophy and spatial memory impairment in aged rats by nerve growth factor. *Nature*, 329(6134):65–68, 1987. doi: 10.1038/329065a0.

M. M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956. doi: 10.1287/opre.4.1.61.

D. B. Fogel. A parallel processing approach to a multiple traveling salesman problem using evolutionary programming. In *Proceedings of the fourth annual symposium on parallel processing*, pages 318–326, Fullerton, CA, 1990. doi: 10.12785/amis/090218.

C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary computation*, 3(1):1–16, 1995. doi: 10.1162/evco.1995.3.1.1.

C. M. Fonseca, P. J. Fleming, et al. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. doi: 10.1.1.48.9077.

H. Frederick and G. J. Lieberman. *Introduction to operations research*. McGraw-Hill, New York, 1982. ISBN 978-0073523453.

G. N. Frederickson and B. Wittman. Approximation algorithms for the traveling repairman and speeding deliveryman problems. *Algorithmica*, 62(3-4):1198–1221, 2012. doi: 10.1137/050645464.

B. Freisleben and P. Merz. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In *IEEE International Conference on Evolutionary Computation*, pages 616–621, New York, USA, 1996. IEEE. doi: 10.1109/ICEC.1996.542671.

C. Frye. The neurosteroid 3α,5α-THP has antiseizure and possible neuroprotective effects in an animal model of epilepsy. *Brain Research*, 696(1):113–120, 1995. doi: 10.1016/0006-8993(95)00793-P.

M. Gansterer and R. F. Hartl. Collaborative vehicle routing: a survey. *European Journal of Operational Research*, 268(1):1–12, 2018. doi: 10.1016/j.ejor.2017.10.023.

A. García, P. Jodrá, and J. Tejel. A note on the traveling repairman problem. *Networks*, 40(1):27–31, 2002. doi: 10.1002/net.10031.

M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, USA, 1990. ISBN 0716710455. doi: 10.1137/1024022.

N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *37th Annual Symposium on Foundations of Computer Science*, pages 302–309, New York, USA, 1996. IEEE. doi: 10.1109/SFCS.1996.548489.

B. Gavish. A note on the formulation of the m-salesman traveling salesman problem. *Management Science*, 22(6):704–705, 1976. doi: 10.1287/mnsc.22.6.704.

B. Gavish and K. Srikanth. An optimal solution method for large-scale multiple traveling salesmen problems. *Operations Research*, 34(5):698–717, 1986. doi: 10.1287/opre.34.5.698.

M. Gendreau and J.-Y. Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010. ISBN 9783319910857.

M. Gendreau, A. Hertz, and G. Laporte. New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research*, 40(6):1086–1094, 1992. doi: 10.1287/opre.40.6.1086.

M. Gendreau, G. Laporte, and F. Semet. A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, 106(23):539–545, 1998. doi: 10.1016/S0377-2217(97)00289-0.

X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. *Applied Soft Computing*, 11(4):3680–3689, 2011. doi: 10.1016/j.asoc.2011.01.039.

S. Ghafurian and N. Javadian. An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems. *Applied Soft Computing*, 11(1): 1256–1262, 2011. doi: 10.1016/j.asoc.2010.03.002.

K. Gilbert and R. B. Hofstra. A new multiperiod multiple traveling salesman problem with heuristic and application to a scheduling problem. *Decision Sciences*, 23(1): 250–259, 1992. doi: 10.1111/j.1540-5915.1992.tb00387.

F. W. Glover and G. A. Kochenberger. *Handbook of metaheuristics*, volume 57. Springer Science & Business Media, 2006. ISBN 9780306480560.

M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82(1):111–124, 1998. doi: 10.1137/1.9781611973075.36.

M. Goldfarb, A. S. Rosenberg, Q. Li, and T. H. Keegan. Impact of latency time on survival for adolescents and young adults with a second primary malignancy. *Cancer*, 124(6):1260–1268, 2018. doi: 10.1002/cncr.31170.

F. González and J. C. Rivera. A multi-start iterative local search for the k-traveling repairman problem. `http://www.eafit.edu.co/programas-academicos/pregrados/ingenieria-matematica/practicas-investigativas/Documents/multi-start-iterative-local-search-k-traveling-repairman.pdf`, 2015. Unpublished manuscript, online.

C. Goodwill. The carpal tunnel syndrome: long-term follow-up showing relation of latency measurements to response to treatment. *Rheumatology*, 8(1):12–21, 1965. doi: 10.1093/rheumatology/8.1.12.

L. Gouveia and S. Voß. A classification of formulations for the time-dependent traveling salesman problem. *European Journal of Operation Research*, 83(1):69–82, 1995. doi: 10.1016/0377-2217(93)E0238-S.

J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht. Genetic algorithms for the traveling salesman problem. In *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, pages 160–168. Lawrence Erlbaum, New Jersey, 1985. doi: 10.1007/BF02125403.

M. Grötschel, M. Jünger, and G. Reinelt. Optimal control of plotting and drilling machines: a case study. *Mathematical Methods of Operations Research*, 35(1): 61–84, 1991. doi: 0.1007/BF01415960.

V. Grout, J. McGinn, and J. Davies. Real-time optimisation of access control lists for efficient internet packet filtering. *Journal of Heuristics*, 13(5):435–454, 2007. doi: 10.1007/s10732-007-9019-1.

P. D. Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech House, London, UK, 2013. ISBN 9781580532556.

J. Gu and X. Huang. Efficient local search with search space smoothing: A case study of the traveling salesman problem (tsp). *IEEE Transactions on Systems, Man, and Cybernetics*, 24(5):728–735, 1994. doi: 10.1109/21.293486.

K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the second ACM SIGCOMM Workshop on Internet Measurement*, pages 5–18, New York, USA, 2002. ACM. ISBN 158113603X. doi: 10.1145/637201.637203.

G. Gutin and A. Punnen. *The traveling salesman problem and its variations*. Combinatorial Optimization. Springer, New York, USA, 2006. ISBN 9780306482137.

B. Ha Bang. A grasp+vnd algorithm for the multiple traveling repairman problem with distance constraints. *Journal of Computer Science and Cybernetics*, 33:272–288, 2018. doi: 10.15625/1813-9663/33/3/10511.

J. M. Hammersley. Monte carlo methods for solving multivariable problems. *Annals of the New York Academy of Sciences*, 86(1):844–874, 1960. doi: 10.1111/j.1749-6632.1960.tb42846.x.

D. He, G. Mujica, J. Portilla, and T. Riesgo. Modelling and planning reliable wireless sensor networks based on multi-objective optimization genetic algorithm with changeable length. *Journal of Heuristics*, 21(2):257–300, 2015. doi: 10.1007/s10732-014-9261-2.

Y. He and M. Xiang. An empirical analysis of approximation algorithms for the euclidean traveling salesman problem. Technical Report arXiv:1705.09058 [cs.CC], arXiv.org, 2017.

V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009. doi: 10.1016/j.ejor.2007.08.048.

D. J. Henry. A more compact formulation of the symmetric multiple traveling salesman problem with fixed charges. *Networks*, 11(1):73–75, 1981. doi: 10.1002/net.3230110109.

H. Hernández-Pérez, I. Rodríguez-Martín, and J. J. Salazar-González. A hybrid GRASP/VND heuristic for the one-commodity pickup-and-delivery traveling salesman problem. *Computers & Operations Research*, 36(5):1639–1645, 2009. doi: 10.13140/RG.2.1.4192.5286.

F. Hillier and G. Lieberman. *Introduction to operations research*. Holden-Day, Michigan, USA, 1986. ISBN 9780816238712.

K. L. Hoffman, M. Padberg, and G. Rinaldi. Traveling salesman problem. In S. I. Gass and M. C. Fu, editors, *Encyclopedia of Operations Research and Management Science*, pages 1573–1578. Springer, Boston, MA, 2013. ISBN 978-3-540-85363-3.

S. Hong and M. W. Padberg. A note on the symmetric multiple traveling salesman problem with fixed charges. *Operations Research*, 25(5):871–874, 1977. doi: 10.1287/opre.25.5.871.

Z.-h. Hu and M. Zhao. Simulation on traveling salesman problem (TSP) based on artificial bees colony algorithm. *Transactions of Beijing Institute of Technology*, 11:010, 2009. doi: 10.2991/icmmcce-15.2015.94.

A. Jaszkiewicz and P. Zielniewicz. Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. *European Journal of Operational Research*, 193(3):885–890, 2009. doi: 10.1016/j.ejor.2007.10.054.

M. John and T. Picton. Human auditory steady-state responses to amplitude-modulated tones: phase and latency measurements. *Hearing research*, 141(1-2):57–79, 2000. doi: 10.1016/S0378-5955(99)00209-9.

D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974. doi: 10.1016/S0022-0000(74)80044-9.

D. S. Johnson. Local optimization and the traveling salesman problem. In *International colloquium on automata, languages, and programming*, pages 446–461. Springer, 1990. doi: 10.1007/BFb0032050.

D. S. Johnson and L. A. McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1:215–310, 1997. doi: 10.1.1.70.7639.

R. Jonker and T. Volgenant. An improved transformation of the symmetric multiple traveling salesman problem. *Operations Research*, 36(1):163–167, 1988. doi: 10.1287/opre.36.1.163.

L. Joo Ghee, Q. Junaid, C. Chun Tung, and M. Archan. Minimum latency broadcasting in multiradio, multichannel, multirate wireless meshes. *IEEE Transactions on Mobile Computing*, 8(11):1510–1523, 2009. doi: 10.1109/TMC.2009.68.

N. Jozefowiez, F. Glover, and M. Laguna. Multi-objective meta-heuristics for the traveling salesman problem with profits. *Journal of Mathematical Modelling and Algorithms*, 7(2):177–195, 2008. doi: 10.1007/s10852-008-9080-2.

H. Jung, K. Lee, and W. Chun. Integration of GIS, GPS, and optimization technologies for the effective control of parcel delivery service. *Computers & Industrial Engineering*, 51(1):154–162, 2006. doi: 10.1016/j.cie.2006.07.007.

P. Junjie and W. Dingwei. An ant colony optimization algorithm for multiple travelling salesman problem. In *First International Conference on Innovative Computing, Information and Control.*, volume 1, pages 210–213, New Jersey, USA, 2006. IEEE. doi: 10.1109/ICICIC.2006.40.

Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai. Hermes: Latency optimal task assignment for resource-constrained mobile computing. *IEEE Transactions on Mobile Computing*, 16(11):3056–3069, 2017. doi: 10.1109/TMC.2017.2679712.

I. Kara and T. Bektas. Integer linear programming formulations of multiple salesman problems and its variations. *European Journal of Operational Research*, 174(3): 1449–1458, 2006. doi: 10.1016/j.ejor.2005.03.008.

I. Kara, B. Y. Kara, and M. K. Yetis. Energy minimizing vehicle routing problem. In *International Conference on Combinatorial Optimization and Applications*, volume 4616 of *Lecture Notes in Computer Science*, pages 62–71, Berlin, 2007. Springer. doi: 10.1287/mnsc.40.10.1276.

D. Karaboga and B. Gorkemli. A combinatorial artificial bee colony algorithm for traveling salesman problem. In *International symposium on innovations in intel-*

ligent systems and applications, pages 50–53, New York, USA, 2011. IEEE. doi: 10.1109/INISTA.2011.5946125.

R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane. *Mathematics of operations research*, 2(3):209–224, 1977. doi: 10.1287/moor.2.3.209.

M. Karpinski, M. Lampis, and R. Schmied. New inapproximability bounds for TSP. *Journal of Computer and System Sciences*, 81(8):1665–1677, 2015. doi: 10.1016/j.jcss.2015.06.003.

Y. Kergosien, C. Lenté, and J.-C. Billaut. Home health care problem: An extended multiple traveling salesman problem. In *Proceedings of the 4th Multidisciplinary International Scheduling Conference: Theory and Applications*, pages 85–92, Dublin, Ireland, 2009. doi: 10.1.1.431.7935.

E. Khodabandeh, L. Bai, S. S. Heragu, G. W. Evans, T. Elrod, and M. Shirkness. Modelling and solution of a large-scale vehicle routing problem at GE appliances & lighting. *International Journal of Production Research*, 55(4):1100–1116, 2017. doi: 10.1080/00207543.2016.1220685.

P. Kilby, P. Prosser, and P. Shaw. Guided local search for the vehicle routing problem with time windows. In *Meta-heuristics*, pages 473–486. Springer, New York, USA, 1999. doi: 10.1007/978-1-4615-5775-3_32.

D. Kim, B. H. Abay, R. Uma, W. Wu, W. Wang, and A. O. Tokuta. Minimizing data collection latency in wireless sensor network with multiple mobile elements. In *Proceedings IEEE International Conference on Computer Communications*, pages 504–512, New York, USA, 2012. IEEE. doi: 10.1109/INFCOM.2012.6195791.

I. Y. Kim and O. L. de Weck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and multidisciplinary optimization*, 29(2):149–158, 2005. doi: 10.1007/s00158-004-0465-1.

S. Kim and J.-H. Na. Improving latency using codes in mission-critical communication. In *Proceedings IEEE 19th international conference on advanced communication technology*, pages 730–734, Piscataway, New Jersey, USA, 2017. IEEE. doi: 10.23919/ICACT.2017.7890189.

A. Király and J. Abonyi. *Optimization of Multiple Traveling Salesmen Problem by a Novel Representation Based Genetic Algorithm*, pages 241–269. Springer, Berlin, Heidelberg, 2011. doi: 10.1007/978-3-642-21705-0_9.

J. Knox. Tabu search performance on the symmetric traveling salesman problem. *Computers & Operations Research*, 21(8):867–876, 1994. doi: 10.1016/0305-0548(94)90016-7.

N. Kohara, J. Kimura, R. Kaji, Y. Goto, J. Ishii, M. Takiguchi, and M. Nakai. F-wave latency serves as the most reproducible measure in nerve conduction studies of diabetic polyneuropathy: multicentre analysis in healthy subjects and patients with diabetic polyneuropathy. *Diabetologia*, 43(7):915–921, 2000. doi: 10.1007/s001250051469.

P. Korhonen, S. Salo, and R. E. Steuer. A heuristic for estimating nadir criterion values in multiple objective linear programming. *Operations Research*, 45(5):751–757, 1997. doi: 10.1287/opre.45.5.751.

G. Kovács and K. Spens. Humanitarian logistics in disaster relief operations. *International Journal of Physical Distribution and Logistics Management*, 37(2):99–114, 2007. doi: 10.1108/09600030710734820.

G. Kovács and K. Spens. Identifying challenges in humanitarian logistics. *Interna-*

tional Journal of Physical Distribution and Logistics Management, 39(6):506–528, 2009. doi: 10.1108/09600030910985848.

N. Krasnogor and J. Smith. A memetic algorithm with self-adaptive local search: TSP as a case study. In *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*, pages 987–994, Massachusetts, United States, 2000. Morgan Kaufmann Publishers Inc. doi: 10.1016/j.swevo.2011.11.003.

J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proceedings of the American Mathematical society*, volume 7, pages 48–50, New York, USA, 1956. JSTOR. doi: 10.2307/2033241.

G. Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operations Research*, 59(2):231–247, 1992. doi: 10.1016/0377-2217(92)90138-Y.

G. Laporte. A short history of the traveling salesman problem. `neumann.hec.ca/chairedistributique/common/laporte-short.pdf`, 2006. [Online].

G. Laporte, C. Sriskandarajah, and A. Asef-Vaziri. *Some Applications of the Generalized Traveling Salesman Problem.* Centre de recherche sur les transports, (Montréal, Québec), 1995. doi: 10.1057/jors.1996.190.

P. Larranaga, C. M. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999. doi: 10.1023/A:1006529012972.

M. Laumanns, L. Thiele, and E. Zitzler. An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. *European Journal of Operational Research*, 169(3):932–942, 2006. doi: 10.1016/j.ejor.2004.08.029.

J. Lee and M. Choi. Optimization by multicanonical annealing and the traveling salesman problem. *Physical Review E*, 50(2):R651, 1994. doi: 10.1103/PhysRevE.50.

A. Leiras, I. de Brito Jr, E. Queiroz Peres, T. Rejane Bertazzo, and H. Tsugunobu Yoshida Yoshizaki. Literature review of humanitarian logistics research: trends and challenges. *Journal of Humanitarian Logistics and Supply Chain Management*, 4(1):95–130, 2014. doi: 10.1108/JHLSCM-04-2012-0008.

J. K. Lenstra and A. H. G. R. Kan. Some simple applications of the travelling salesman problem. *Operational Research Quarterly*, 26(4):717–733, 1975. doi: 10.1057/jors.1975.151.

L. Li and Z. Fu. The school bus routing problem: a case study. *Journal of the Operational Research Society*, 53(5):552–558, 2002. doi: 10.1057/palgrave/jors/2601341.

P. Lichtsteiner, C. Posch, and T. Delbruck. A 128×128 120db 15μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. doi: 10.1109/JSSC.2007.914337.

S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973. doi: 10.1287/opre.21.2.498.

M. Littner, C. Kushida, M. Wise, D. Davila, T. Morgenthaler, T. Lee-Chiong, M. Hirshkowitz, L. Daniel, D. Bailey, R. Berry, S. Kapen, and M. Kramer. Practice parameters for clinical use of the multiple sleep latency test and the maintenance of wakefulness test. *Journal of Clinical Sleep Medicine*, 28(1):113–121, 2005. doi: 10.1378/chest.08-0822.

H. S. Lope and L. S. Coelho. Particle swarm optimization with fast local search for the blind traveling salesman problem. In *Fifth International Conference on Hybrid Intelligent Systems*, pages 245–250, New Jersey, USA, 2005. IEEE. doi: 10.1109/ICHIS.2005.86.

G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency MAC for tree-based data gathering in sensor networks. *Wireless Communications and Mobile Computing*, 7(7):863–875, 2007. doi: 10.1002/wcm.503.

A. Lucena. Time-dependent traveling salesman problem–the deliveryman case. *Networks*, 20(6):753–763, 1990. doi: 10.1002/net.3230200605.

Z. Luo, H. Qin, and A. Lim. Branch-and-price-and-cut for the multiple traveling repairman problem with distance constraints. *European Journal of Operational Research*, 234(1):49–60, 2014. doi: 10.1016/j.ejor.2013.09.014.

T. Lust and J. Teghem. The multiobjective traveling salesman problem: a survey and a new approach. In C. Coello Coello, C. Dhaenens, and L. Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, volume 272 of *Studies in Computational Intelligence*, pages 119–141. Springer, Berlin, Heidelberg, 2010. doi: 10.1007/978-3-642-11218-8_6.

C. Ma, J. Zhang, Y. Zhao, M. Habib, S. Savas, and B. Mukherjee. Traveling repairman problem for optical network recovery to restore virtual networks after a disaster. *IEEE Journal of Optical Communications and Networking*, 7(11):B81–B92, 2015. doi: 10.1364/JOCN.7.000B81.

H. V. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, and A. Venkataramani. A structural approach to latency prediction. In *Proceedings of the sixth*

ACM SIGCOMM conference on Internet Measurement, pages 99–104, New York, USA, 2006. ACM. ISBN 1595935614. doi: 10.1145/1177080.1177092.

S. Madsen, J. Harte, C. Elberling, and T. Dau. Accuracy of averaged auditory brainstem response amplitude and latency estimates. *International Journal of Audiology*, 57(5):345–353, 2018. doi: 10.1080/14992027.2017.1381770.

A. Malarky. Real-time vehicle position determination using communications with variable latency, Feb. 26 2013. US Patent 8,384,560.

M. Malek, M. Guruswamy, M. Pandya, and H. Owens. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research*, 21(1):59–84, 1989. doi: 10.1007/BF02022093.

R. Mangharam, R. Rajkumar, M. Hamilton, P. Mudaliget, and F. Bait. Bounded-latency alerts in vehicular networks. In *2007 Mobile Networking for Vehicular Environments*, pages 55–60, New York, USA, 2007. IEEE. doi: 10.1109/MOVE.2007.4300804.

Y. Marinakis and M. Marinaki. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. *Computers & Operations Research*, 37(3):432–442, 2010. doi: 10.1016/j.cor.2009.03.004.

Y. Marinakis, A. Migdalas, and P. M. Pardalos. Expanding neighborhood grasp for the traveling salesman problem. *Computational Optimization and Applications*, 32(3):231–257, 2005. doi: 10.1007/s10589-005-4798-5.

Y. Marinakis, M. Marinaki, and G. Dounias. Honey bees mating optimization algorithm for the euclidean traveling salesman problem. *Information Sciences*, 181 (20):4684–4698, 2011. doi: 10.1016/j.ins.2010.06.032.

R. T. Marler and J. S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, 41(6): 853–862, 2010. doi: 10.1007/s00158-009-0460-7.

C. S. Martin and M. R. Salavatipour. Approximation algorithms for capacitated k-travelling repairmen problems. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 64. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2016. doi: 10.4230/LIPIcs.

O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the Traveling Salesman Problem. *Operations Research Letters*, 11(4):219–224, 1992. doi: 10. 1016/0167-6377(92)90028-2.

G. Mavrotas. Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009. doi: 10.1016/j.amc.2009.03.037.

G. Mavrotas and K. Florios. An improved version of the augmented ε-constraint method AUGMECON2 for finding the exact pareto set in multi-objective integer programming problems. *Applied Mathematics and Computation*, 219(18):9652–9669, 2013. doi: 0.1016/j.amc.2013.03.002.

I. Méndez-Díaz, P. Zabala, and A. Lucena. A new formulation for the traveling deliveryman problem. *Discrete Applied Mathematics*, 156(17):3223–3237, 2008. doi: 10.1016/j.dam.2008.05.009.

P. Merz and B. Freisleben. Genetic local search for the tsp: New results. In *IEEE International Conference on Evolutionary Computation*, pages 159–164, New York, USA, 1997. IEEE. doi: 10.1109/ICEC.1997.592288.

B. Metev and V. Vassilev. A method for nadir point estimation in molp problems. *Cybernetics and Information Technologies*, 3(2):15–24, 2003. doi: 10.1007/s00500-015-1940-x.

C. E. Miller, A. W. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM*, 7(4):326–329, 1960. ISSN 0004-5411. doi: 10.1145/321043.321046.

H. Min. The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, 23(5):377–386, 1989. doi: 10.1016/0191-2607(89)90085-X.

A. Misevičius. Using iterated tabu search for the traveling salesman problem. *Information technology and control*, 32(3):44–57, 2004. doi: 10.1.1.119.8965.

A. Mjirda, R. Todosijević, S. Hanafi, P. Hansen, and N. Mladenović. Sequential variable neighborhood descent variants: an empirical study on the traveling salesman problem. *International Transactions in Operational Research*, 24(3):615–633, 2017. doi: 10.1111/itor.12282.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997. doi: 10.1016/S0305-0548(97)00031-2.

A. Modares, S. Somhom, and T. Enkawa. A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. *International Transactions in Operational Research*, 6(6):591–606, 1999. doi: 10.1111/j.1475-3995.1999.tb00175.x.

J. Molina, M. Laguna, R. Marti, and R. Caballero. SSPMO: A scatter tabu search procedure for non-linear multiobjective optimization. *INFORMS Journal on Computing*, 19(1):91–100, 2007. doi: 10.1287/ijoc.1050.0149.

J. Molina, A. López-Sánchez, A. Hernández-Díaz, and I. Martínez-Salazar. A multi-start algorithm with intelligent neighborhood selection for solving multi-objective humanitarian vehicle routing problems. *Journal of Heuristics*, 24(2):111–133, 2018. doi: 10.1007/s10732-017-9360-y.

M. Moshref-Javadi and S. Lee. The customer-centric, multi-commodity vehicle routing problem with split delivery. *Expert Systems with Applications*, 56:335–348, 2016. doi: 10.1016/j.eswa.2016.03.030.

C. C. Murray and A. G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86–109, 2015. doi: 10.1016/j.trc.2015.03.005.

V. Nagarajan and R. Ravi. The directed minimum latency problem. In A. Goel, K. Jansen, J. Rolim, and R. Rubinfeld, editors, *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 5171 of *Lecture Notes in Computer Science*, pages 193–206, Berlin, Heidelberg, 2008. Springer. doi: 10.1007/978-3-540-85363-3_16.

M. Najafi, K. Eshghi, and W. Dullaert. A multi-objective robust optimization model for logistics planning in the earthquake response phase. *Transportation Research*, 49(5):217–249, 2013. doi: 10.1016/j.tre.2012.09.001.

R. Nallusamy, K. Duraiswamy, R. Dhanalaksmi, and P. Parthiban. Optimization of non-linear multiple traveling salesman problem using k-means clustering, shrink wrap algorithm and meta-heuristics. *International Journal of Nonlinear Science*, 9:1749–3889, 2010. doi: IJNS.2009.12.30/306.

R. M. Newton and W. H. Thomas. Design of school bus routes by computer. *Socio-Economic Planning Sciences*, 3(1):75–85, 1969. doi: 10.1016/0038-0121(69) 90051-2.

S. Nucamendi-Guillén, I. A. Martínez-Salazar, F. Angel-Bello, and J. M. Moreno-Vega. A mixed integer formulation and an efficient metaheuristic procedure for the k-travelling repairmen problem. *Journal of The Operational Research Society*, 67(8):1121–1134, 2016. doi: 10.1057/jors.2015.113.

I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the second international conference on Genetic Algorithms and their application*, pages 224–230, Hillsdale, New Jersey, USA, 1987. L. Erlbaum Associates Inc. ISBN 0805801588.

R. L. Oliver. An investigation of the interrelationship between consumer (dis)satisfaction and complaint reports. *Advances in Consumer Research*, 14(1): 218–222, 1987. doi: 10.1108/03090569110140489.

G. Onder, I. Kara, and T. Derya. New integer programming formulation for multiple traveling repairmen problem. *Transportation Research Procedia*, 22:355–361, 2017. doi: 10.1016/j.trpro.2017.03.042.

G. C. Onwubolu. *Optimizing CNC Drilling Machine Operations: Traveling Salesman Problem-Differential Evolution Approach*, pages 537–565. Studies in Fuzziness and Soft Computing. Springer, Berlin, Heidelberg, 2004. doi: 10.1007/978-3-540-39930-8_22.

A. Orman and H. Williams. A survey of different integer programming formulations of the travelling salesman problem. In E. J. Kontoghiorghes and C. Gatu, editors, *Optimisation, Econometric and Financial Analysis*, chapter 5, pages 91–104. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-36626-3.

M. Ortuño, P. Cristóbal, J. Ferrer, F. Martín-Campo, S. Muñoz, G. Tirado, and B. Vitoriano. Decision aid models and systems for humanitarian logistics. a survey. In B. Vitoriano, J. Montero, and D. Ruan, editors, *Decision aid models for disaster* *management and emergencies*, Atlantis Computational Intelligence Systems, pages 17–44. Atlantis Press, Paris, France, 2013. doi: 10.2991/978-94-91216-74-9_2.

R. E. Overstreet, D. Hall, J. B. Hanna, and R. Kelly Rainer Jr. Research in humanitarian logistics. *Journal of Humanitarian Logistics and Supply Chain Management*, 1(2):114–131, 2011. doi: 10.1108/20426741111158421.

J. Paniagua-Soto, J. Ruiz-García, M. Iznaola-Muñoz, and L. Ruiz-Serrano. Multiple sleep latency test in patients with suspected narcolepsy. Review of 45 cases. *Journal of Clinical Sleep Medicine*, 14(1):272–273, 2013. doi: 10.1016/j.sleep.2013.11.665.

C. M. Papadimitriou. *Computational complexity.* Addison-Wesley, Boston, Massachusetts, USA, 1994. ISBN 0201530821.

L. Paquete and T. Stützle. A two-phase local search for the biobjective traveling salesman problem. In C. Fonseca, P. Fleming, E. Zitzler, L. Thiele, and K. Deb, editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 69–89. Berlin, Heidelberg, 2003. doi: 10.1007/s10732-009-9103-9.

L. Paquete and T. Stützle. Design and analysis of stochastic local search for the multiobjective traveling salesman problem. *Computers & Operations Research*, 36 (9):2619–2631, 2009. doi: 10.1016/j.cor.2008.11.013.

L. Paquete, M. Chiarandini, and T. Stützle. Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In X. Gandibleux, M. Sevaux, K. Sorensen, and V. Tkindt, editors, *Metaheuristics for Multiobjective Optimisation*, volume 35 of *Lecture Notes in Economics and Mathematical Systems*, pages 177–199. Springer, Berlin, Heidelberg, 2004. doi: 10.1007/978-3-642-17144-4_7.

J. Park and B.-I. Kim. The school bus routing problem: A review. *European Journal of Operational Research*, 202(2):311–319, 2010. doi: 10.1016/j.ejor.2009.05.017.

D. A. Patterson. Latency lags bandwith. *Communications of the ACM*, 47(10): 71–75, 2004. doi: 10.1145/1022594.1022603.

W. Peng, Q. Zhang, and H. Li. Comparison between MOEA/D and NSGA-II on the multi-objective travelling salesman problem. In C. Goh, Y. Ong, and K. Tan, editors, *Multi-objective memetic algorithms*, volume 171 of *Studies in Computational Intelligence*, pages 309–324. Springer, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-88051-6_14.

J. C. Picard and M. Queyranne. The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. *Operations Research*, 26(1):86–110, 1978. doi: 10.1287/opre.26.1.86.

J. Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63(3):337–370, 1996. doi: 10.1007/BF02125403.

E. J. Powley, D. Whitehouse, and P. I. Cowling. Monte Carlo tree search with macro-actions and heuristic route planning for the physical travelling salesman problem. In *IEEE Conference on Computational Intelligence and Games*, pages 234–241, New York, USA, 2012. IEEE. doi: 10.1007/978-3-642-29178-4_26.

C. Prins. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computer and Operations Research*, 31(12):1985–2002, 2004. ISSN 03050548. doi: 10.1016/s0305-0548(03)00158-8.

A. Przybył. Hard real-time communication solution for mechatronic systems. *Robotics and Computer-Integrated Manufacturing*, 49:309–316, 2018. doi: 10.1016/j.rcim.2017.08.001.

S. Raff. Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63–211, 1983. doi: 10.1016/0305-0548(83)90030-8.

R. Randelman and G. S. Grest. n-city traveling salesman problem: Optimization by simulated annealings. *Journal of Statistical Physics*, 45(5):885–890, 1986. doi: 10.1007/BF01020579.

M. R. Rao. Technical note. a note on the multiple traveling salesmen problem. *Operations Research*, 28(3):628–632, 1980. doi: 10.1287/opre.28.3.628.

J. Rey Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence*, volume 1, pages 82–87, New York, USA, 1994. doi: 10.1109/ICEC.1994.350037.

C. Romero. *Teoría de la decisión multicriterio: conceptos, técnicas y aplicaciones.* Alianza Universidad textos. Alianza Editorial, 1993. ISBN 9788420681443.

D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. Approximate algorithms for the traveling salesperson problem. In *Proceedings of the fifteenth annual symposium on Switching and Automata Theory*, pages 33–42, Schenectady, New York, USA, 1974. General Electric Corporate Research and Development. doi: 10.1109/SWAT.1974.4.

D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977. doi: 10.1137/0206041.

M. Saito, N. Sano, K. Ueda, Y. Shibata, K. Kuriyama, T. Komiyama, K. Marino, S. Aoki, and H. Onishi. Evaluation of the latency and the beam characteristics of a respiratory gating system using an elekta linear accelerator and a respiratory indicator device, abches. *Medical Physics*, 45(1):74–80, 2018. doi: 10.1002/mp.12664.

M. Salami, C. Itami, T. Tsumoto, and F. Kimura. Change of conduction velocity by regional myelination yields constant latency irrespective of distance between thalamus and cortex. *Proceedings of the National Academy of Sciences*, 100(10):6174–6179, 2003. doi: 10.1073/pnas.0937380100.

H. A. Saleh and R. Chelouah. The design of the global navigation satellite system surveying networks using genetic algorithms. *Engineering Applications of Artificial Intelligence*, 17(1):111–122, 2004. doi: 10.1016/j.engappai.2003.11.001.

A. Salehipour, K. Sörensen, P. Goos, and O. Bräysy. Efficient GRASP+ VND and GRASP + VNS metaheuristics for the traveling repairman problem. *4OR: A Quarterly Journal of Operations Research*, 9(2):189–209, 2011. doi: 10.1007/s10288-011-0153-0.

F. Samanlioglu, W. G. Ferrell Jr, and M. E. Kurz. A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem. *Computers & Industrial Engineering*, 55(2):439–449, 2008. doi: 10.1016/j.cie.2008.01.005.

D. Sarddar, J. Banerjee, T. Jana, S. K. Saha, U. Biswas, and M. Naskar. Minimization of handoff latency by angular displacement method using GPS based map. *International Journal of Computer Applications*, 7(3):29–37, 2010. doi: 10.5120/3039-4122.

D. Sarddar, S. Chatterjee, R. Jana, S. S. Babu, H. N. Khan, U. Biswas, and M. Naskar. Minimization of handoff latency by distance measurement method. *International Journal of Computer Applications*, 8(2):283–289, 2011. doi: 10.1007/s11227-008-0175-3.

J. Sarubbi, H. Luna, and G. Miranda. Minimum latency problem as a shortest path problem with side constraints. In *Book of Extended Abstracts of the XIV Latin Ibero-American Congress on Operations Research*, Cartagena, Colombia, 2008.

P. Schittekat, M. Sevaux, and K. Sorensen. A mathematical formulation for a school bus routing problem. In *International Conference on Service Systems and Service Management*, volume 2, pages 1552–1557, New York, USA, 2006. IEEE. doi: 10.1109/ICSSSM.2006.320767.

P. Schittekat, J. Kinable, K. Sörensen, M. Sevaux, F. Spieksma, and J. Springael. A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research*, 229(2):518–528, 2013. doi: 10.1016/j.ejor.2013.02.025.

C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing sensor networks in the energy-latency-density design space. *IEEE Transactions on Mobile Computing*, 1(1):70–80, 2002. ISSN 1536-1233. doi: 10.1109/tmc.2002.1011060.

R. Sengupta. Other decision-making methods. In R. Sengupta, A. Gupta, and J. Dutta, editors, *Decision Sciences: Theory and Practice*, chapter 5, pages 220–257. CRC Press, Boca Raton, FL, USA, 2016. ISBN 146656430X.

V. A. Shim, K. C. Tan, and C. Y. Cheong. A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(5):682–691, 2012a. doi: 10.1109/TSMCC.2012.2188285.

V. A. Shim, K. C. Tan, and K. K. Tan. A hybrid estimation of distribution algorithm for solving the multi-objective multiple traveling salesman problem. In *IEEE Congress on Evolutionary Computation*, pages 1–8, New York, USA, 2012b. IEEE. doi: 10.1109/CEC.2012.6256438.

A. Shokouhi rostami, F. Mohanna, H. Keshavarz, and A. A. Rahmani Hosseinabadi. Solving multiple traveling salesman problem using the gravitational emulation local search algorithm. *Applied Mathematics & Information Sciences*, 9(2):679–709, 2015. doi: 10.12785/amis/090218.

S. J. Shyu, B. M. Lin, and P.-Y. Yin. Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time. *Computers & Industrial Engineering*, 47(2-3):181–193, 2004. doi: 10.1016/j.cie.2004.06.006.

M. M. Silva, A. Subramanian, T. Vidal, and L. S. Ochi. A simple and effective metaheuristic for the minimum latency problem. *European Journal of Operational Research*, 221(3):513–520, 2012. doi: 10.1016/j.ejor.2012.03.044.

E. A. Silver, R. Victor, V. Vidal, and D. de Werra. A tutorial on heuristic methods. *European Journal of Operational Research*, 5(3):153–162, 1980. doi: 10.1016/0377-2217(80)90084-3.

A. Sipahioglu, A. Yazici, O. Parlaktuna, and U. Gurel. Real-time tour construction for a mobile robot in a dynamic environment. *Robotics and Autonomous Systems*, 56(4):289–295, 2008. doi: 10.1016/j.robot.2007.09.011.

R. Sitters. Polynomial time approximation schemes for the traveling repairman and other minimum latency problems. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete Algorithms*, pages 604–616, New Jersey, USA, 2014. SIAM. doi: 10.1016/j.cor.2013.01.003.

T. Smith, Z. Shen, and J. Friedman. Evaluation of coefficients for the weighted sum of gray gases model. *Journal of heat transfer*, 104(4):602–608, 1982. doi: 10.1115/1.3245174.

D. Sofge, A. Schultz, and K. De Jong. Evolutionary computational approaches to solving the multiple traveling salesman problem using a neighborhood attractor schema. *Applications of Evolutionary Computing*, pages 153–162, 2002. doi: 10.1007/3-540-46004-7_16.

N. K. Squires, K. C. Squires, and S. A. Hillyard. Two varieties of long-latency positive waves evoked by unpredictable auditory stimuli in man. *Electroencephalography and Clinical Neurophysiology*, 38(4):387–401, 1975. doi: 10.1016/0013-4694(75)90263-1.

M. O. Stephenson. The theory and practice of international humanitarian relief coordination. In J. A. Koops and R. Biermann, editors, *Palgrave Handbook of Inter-Organizational Relations in World Politics*, pages 485–502. Palgrave Macmillan UK, London, 2017. ISBN 9781137360397.

S. Sternberg, S. Monsell, R. L. Knoll, and C. E. Wright. The latency and duration of rapid movement sequences: Comparisons of speech and typewriting. In G. E. Stelmach, editor, *Information Processing in Motor Control and Learning*, chapter 6, pages 117–152. Elsevier Academic Press, New Yok, USA, 1978. ISBN 9780126659603.

C. Sutcliffe and J. Board. Optimal solution of a vehicle-routeing problem: Transporting mentally handicapped adults to an adult training centre. *Journal of The Operational Research Society*, 41(1):61–67, 1990. doi: 10.1038/sj/jors/0410107.

J. A. Svestka and V. E. Huckfeldt. Computational experience with an m-salesman traveling salesman algorithm. *Management Science*, 19(7):790–799, 1973. doi: 10.1287/mnsc.19.7.790.

H. A. Taha. *Operations Research: An Introduction.* Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA, 1992. ISBN 0131889230.

L. Tang, J. Liu, A. Rong, and Z. Yang. A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex. *European Journal of Operational Research*, 124(2):267–282, 2000. doi: 10.1016/S0377-2217(99)00380-X.

M. F. Tasgetiren, P. N. Suganthan, and Q.-Q. Pan. A discrete particle swarm optimization algorithm for the generalized traveling salesman problem. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 158–167, New York, USA, 2007. ACM. doi: 10.1145/1276958.1276980.

R. Tavakkoli-Moghaddam, S. Shishegar, A. Siadat, and M. Mohammadi. Design of a reliable bi-objective relief routing network in the earthquake response phase. *Procedia Computer Science*, 102:74–81, 2016. doi: 10.1016/j.procs.2016.09.372.

D. Teodorovic, P. Lucic, G. Markovic, and M. Dell'Orco. Bee colony optimization: principles and applications. In *Seminar on Neural Network Applications in Electrical Engineering*, pages 151–156, New York, USA, 2006. IEEE. doi: 10.1109/NEUREL.2006.341200.

R. Thamilselvan and P. Balasubramanie. A genetic algorithm with a tabu search GTA for traveling salesman problem. *International Journal of Recent Trends in Engineering*, 1(1):607–610, 2009. doi: 10.1.1.379.217.

A. Thomas and M. Mizushima. Logistics training: necessity or luxury. *Forced Migration Review*, 22(22):60–61, 2005. ISSN 14609819.

R. Tomasini, L. Van Wassenhove, and L. Van Wassenhove. *Humanitarian Logistics*. Palgrave Macmillan UK, London, UK, 2009. ISBN 9781349302123.

P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on Computing*, 15(4):333–346, 2003. doi: 10.1287/ijoc.15.4.333.24890.

P. Toth and D. Vigo. *Vehicle routing: problems, methods, and applications.* SIAM, Philadelphia, PA, USA, 2014. ISBN 9781611973587.

J. N. Tsitsiklis. Special cases of traveling salesman and repairman problems with time windows. *Networks*, 22(3):263–282, 1992. doi: 10.1002/net.3230220305.

N. L. Ulder, E. H. Aarts, H.-J. Bandelt, P. J. Van Laarhoven, and E. Pesch. Genetic local search algorithms for the traveling salesman problem. In S. HP and M. R, editors, *Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 109–116, Berlin, Heidelberger, 1990. Springer. doi: 10.1007/BFb0029740.

M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and B. W. Thomas. Route-based Markov decision processes for dynamic vehicle routing problems. `web.winforms.phil.tu-bs.de/paper/ulmer/Ulmer_model.pdf`, 2017.

M. Van Ee and R. Sitters. The a priori traveling repairman problem. *Algorithmica*, 80(10):2818–2833, 2018. doi: 10.1007/s00453-017-0351-z.

D. Vanderbilt and S. G. Louie. A Monte Carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56(2): 259–271, 1984. doi: 10.1016/0021-9991(84)90095-0.

L. Vargas, N. Jozefowiez, and S. U. Ngueveu. A dynamic programming operator for tour location problems applied to the covering tour problem. *Journal of Heuristics*, 23(1):53–80, 2017. doi: 10.1007/s10732-017-9324-2.

B. Vitoriano, M. T. Ortuño, G. Tirado, and J. Montero. A multi-criteria optimization model for humanitarian aid distribution. *Journal of Global Optimization*, 51(2): 189–208, 2011. doi: 10.1007/s10898-010-9603-z.

C. Voudouris and E. Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469–499, 1999. doi: 10.1016/S0377-2217(98)00099-X.

K.-P. Wang, L. Huang, C.-G. Zhou, and W. Pang. Particle swarm optimization for traveling salesman problem. In *International Conference on Machine Learning and Cybernetics*, volume 3, pages 1583–1585, New York, USA, 2003. IEEE. doi: 10.1109/ICMLC.2003.1259748.

L. Wong, M. Y. H. Low, and C. S. Chong. A bee colony optimization algorithm for traveling salesman problem. In *Second Asia International Conference on Modeling & Simulation*, pages 818–823, New York, USA, 2008. IEEE. doi: 10.1109/AMS.2008.27.

L.-P. Wong, M. Y. H. Low, and C. S. Chong. Bee colony optimization with local search for traveling salesman problem. *International Journal on Artificial Intelligence Tools*, 19(03):305–334, 2010. doi: 10.1109/INDIN.2008.4618252.

W. H. Wong and F. Liang. Dynamic weighting in Monte Carlo and optimization. *Proceedings of the National Academy of Sciences*, 94(26):14220–14224, 1997. doi: 10.1073/pnas.94.26.14220.

S. H. Zanakis, J. R. Evans, and A. A. Vazacopoulos. Heuristic methods and applications: a categorized survey. *European Journal of Operational Research*, 43(1): 88–110, 1989. doi: 10.1016/0377-2217(89)90412-8.

T. Zhang, W. Gruver, and M. H. Smith. Team scheduling by genetic search. In *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials*, volume 2, pages 839–844, New York, USA, 1999. IEEE. doi: 10.1109/IPMM.1999.791495.

F. Zhao, J. Dong, S. Li, and X. Yang. An improved genetic algorithm for the multiple traveling salesman problem. In *Control and Decision Conference*, pages 1935–1939. IEEE, 2008. doi: 10.1109/CAR.2010.5456787.

E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland, 1999. https://sop.tik.ee.ethz.ch/publicationListFiles/zitz1999a.pdf.

E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms: a comparative case study. In A. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel, editors, *Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, pages 292–301, Berlin, Heidelberg, 1998. Springer. doi: 10.1007/BFb0056872.

E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999. doi: 10.1109/4235.797969.

E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. Technical Report TIK-Report 103, ETH Zurich, Zurich, Switzerland, 2001. https://pdfs.semanticscholar.org/6672/8d01f9ebd0446ab346a855a44d2b138fd82d.pdf.

Autobiography

Nancy Aracely Arellano Arriaga

Candidate for the double degree of Doctor in Engineering
with a specialization in Systems Engineering
and Doctor in Economics and Business.

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Universidad de Málaga

Facultad de Ciencias Económicas y Empresariales

Thesis:

Several approaches for the Traveling
Salesman Problem

Born the 26th of September, 1989. Holds a Bachelor degree in Applied Mathematics, a M.Sc. in Systems Engineering and is currently a candidate for a double doctoral degree at UANL and UMA under the guidance of Iris Martínez-Salazar and Julián Molina.