Gustavo Linhares Vieira

Trip Planning Optimization: Minimizing Cost and Travel Time in Itineraries With Multiple Destinations

Belo Horizonte - MG
November 2018

Trip Planning Optimization: Minimizing Cost and Travel Time in Itineraries With Multiple Destinations

Gustavo Linhares Vieira
Federal University of Minas Gerais

Supervisor: Frederico Gadelha Guimarães

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Belo Horizonte - MG
November 2018

Abstract

The internet changed significantly the tourism industry. Nowadays, people have more autonomy than ever to research and plan their trips, customized according to their interests. However, current planning tools still require the traveller to spend a lot of time and effort to find the best tickets and hotels for a complete trip itinerary. In situations where the dates and destination ordering are flexible, the number of possible options is even bigger, making the search even harder.

In order to solve this problem, a method that searches the best combinations of flights and accommodations for a trip itinerary was developed, minimizing costs and travel time. In order to do so, multiobjective optimization methods specially adapted to the problem are used. The system is able to get inputs from the user and handle variations in departure dates, stay lengths in each part of the trip and even destinations ordering, which makes it stand out comparing to other existing tools.

The proposed solution is able to quickly find a set of good itineraries, offering the traveller diverse options with little effort. This system represents a big improvement in the trip planning experience, with better optimization times and solutions quality. This way, this work can offer a significant advantage to tickets and hotels e-commerce platforms and be easily applied to the online tourism market.

Keywords: Multiobjective Optimization, Decision Making, Trip Planning, Combinatorial Optimization, Ant Colony Optimization

Resumo

A internet mudou significativamente a indústria do turismo. Atualmente, as pessoas tem mais autonomia que nunca para pesquisar e planejar suas viagens de forma personalizada, de acordo com seus interesses. No entanto, as ferramentas de planejamento atuais ainda exigem grande tempo e esforço do viajante para descobrir as melhores passagens e hotéis para um roteiro completo de viagem. Em casos com flexibilidade nas datas e ordenação de destinos, o número de possíveis opções é ainda maior, tornando a busca ainda mais trabalhosa.

Para solucionar esse problema, foi desenvolvido um método capaz de buscar as melhores combinações de vôos e acomodações para um roteiro de viagem, minimizando os custos e o tempo de transporte. Para isso, são utilizados métodos de otimização multiobjetivo adaptados para o problema específico. O sistema é capaz de receber inputs do usuário e acomodar variações nas datas de partidas, duração das estadias em cada destino e até na ordem dos destinos, se diferenciando assim das ferramentas existentes.

A solução proposta é capaz de encontrar um conjunto de bons itinerários de forma eficiente, oferecendo ao viajante escolhas rápidas entre diversas opções com esforço mínimo. Esse sistema representa um grande avanço na experiência de planejamento de viagens, com melhorias no tempo necessário e na qualidade das opções encontradas. Com isso, esse trabalho pode oferecer um diferencial significativo para sistemas de vendas de passagens e hotéis e ter aplicabilidade direta no mercado de turismo online.

Palavras Chave: Otimização Multiobjetivo, Tomada de Decisões, Planejamento de Viagens, Otimização Combinatória, Otimização por Colônia de Formigas

Declaration

This thesis is result of my own work, except where explicitly referred to other works, and it was not submitted to any other Institution.

Acknowledgements

This work is only possible due to all the help and support along the way. My deepest gratitude to all whose support is irreplaceable:

First, to my advisor Frederico, for the constant support, patience and constructive guidance over these years. It is extremely reassuring to have well-thought-out counsel whenever necessary. Thank you for the great example as well. A great reference to follow is better than a thousand words.

To Professor Lucas Batista, who first advised me on this work. Much of these results are due to his incredibly deep analysis and suggestions.

To Professor Cristiano Castro, for all the guidance and opportunities. Thank you for the trust.

To Professor Patrícia Pena, who first opened the door to science and research for me. Thanks for all the incentive and for always believing in me.

To all my colleagues at MINDS, for the great company for both conversation and research over the years. It's a pleasure to work with you.

To all the technical body at UFMG who made all this possible, as well as CAPES and FAPEMIG for the financial support along the way.

Any achievement is only possible in the right circumstances. I'm also incredibly grateful to all those whose effort helped me to be here today:

To my mother, Rose, for the ever present encouragement to go beyond my comfort zone, for the unmovable belief in me and for all the sacrifice and dedication to my education. Thank you for doing everything and more so I could be the best possible me.

To my father, Márcio, for the unconditional support, for the resilience and for the great example in many things. I hope we can still help each other grow even more.

To my grandmothers, Neide and Nair, for the all the incredible care. Any hardship gets smaller with a grandmother's hug. And my grandfather, Jair, for the encouragements and help along the way.

To my sisters, Helena, Laura and Laís, for always making me feel so loved. Life is incredibly brighter since you arrived. Sometimes the world looks a bit too daunting, but you girls make me never give up on trying to build a better tomorrow for us all.

To Yann, for making even the hardest days still wonderful, for the patience, and for the best slice of pizza. Life is much better with you walking by my side.

To all my friends, for all the laughs and hugs and company. It is incredible to have all of you in my life.

Finally, a big thank you to all those who believe in science, research and education as tools for building a better world for everyone. I am incredibly lucky for all the opportunities I had, and they were only possible due to all the people who support our schools and universities.

Contents

List of figures

List of tables

Acronyms and Abbreviations

ACO Ant Colony Optimization

GRASP Greedy Randomized Adaptive Search Procedure

HV Hypervolume

MOACO Multi-objective Ant Colony Optimization

MOEA/D Multi-objective Evolutionary Algorithm Based on Decomposition

TSP Travelling Salesman Problem

TSPTW Travelling Salesman Problem with Time Windows

VNS Variable Neighborhood Search

WPM Weighted Product Model

WSM Weighted Sum Model

Chapter 1

Introduction

1.1 Motivation

The tourism market has been on the rise over the past couple of decades. Flights prices dropped, the number of airports increased, and much more connections are available, and reports by the World Travel & Tourism Council [55] indicates that this growth will continue. Multi-billion valuated Airbnb and the rise of *couchsurfing* increased the diversity of accommodations types and prices as well. Finally, information access is much more widespread, with the ubiquity of connected phones and computers. As a result, the number of travellers grew significantly. This market growth is reflected on the increased number of travel-related companies, such as hotel networks and flight companies [4].

Much of this growth happened through online tourism. Traffic in websites for buying flights or booking accommodations keeps increasing [7]. More than just providing a new venue for old habits, the internet essentially changed the way people travel. An online tourism analysis [8] indicates that consumers tend to seek more customization on their trips. Instead of buying a one-for-all package, they now want to tailor the trip to their personal needs. As a result of that, independent trip planning surpassed purchases in travel agencies [4].

However, planning a trip on your own is not a simple task. Even though research indicates that travellers enjoy planning the destinations and activities for their journey [31], buying flights and making reservations still takes some time. The high number of options and prices variations (both over time and through different sellers) make the task even harder.

In light of this situation, many tools that aim to help on trip planning have come up. Many of them try to help users to get and select information needed to plan their trip, such as flights and accommodations available, as well as facilitating the buying process. Among these, some prominent examples are Skyscanner, Expedia, GoEuro, Rome2rio, Decolar, Booking and Hipmunk.

Others have a different approach and aim to help customers decide which destinations they want to travel to and what activities, tours and attractions they want to seek in these destinations, such as TripAdvisor. Some even go a step further and try to build a profile for users and make suggestions based on that, including Utrip. Finally, a few of these tools additionally help the user organize a day plan for their trip and share it with others, like Voyajo and TripHobo.

But while these tools definitely help, they still do not solve completely one of the major pains of planning a trip: hunting for cheaper flights and accommodations. Most existing solutions[1] can only search for flights between one pair of destinations at a time. For bigger itineraries with more destinations, the user needs to search for each branch of the trip separately, which can be time consuming. The problem is even more complex for more flexible trips. What if the departure date or length of the stay in each destination can change? Or even the order of the destinations visited? The number of possible configurations for the trip itinerary becomes prohibitive. None of the current known tools is adequate for this kind of situation.

The problem gets even harder if more factors are taken into consideration besides the cost of the trip. Travel time, for example, is also very important for a traveller. Direct flights are usually faster, but more expensive. The trade-off might be worth for some users, so it is important to give them choice among some of the best options.

In order to find the best options, a traveller would need to consider all the possible itineraries and the combination of available flights and accommodations. There might be a few options that are cheaper and faster than the others, but they might not be found by the user because searching for them would take too much time, given current tools. Even if they tried, the prices might have changed by the time they were done. Thus, people often have to settle for a sub-optimal trip plan, even after spending a long time researching.

[1] Hipmunk is a notable exception

The problem described can be seen as an optimization problem in which the user seeks to minimize cost and travel time for a trip. Optimization is often used to solve similar logistic problems, with various real world applications: traffic organization in big cities [18] [56], delivery of goods [12] [49], route planning for cars [11], agricultural vehicles [1], VANTs [34], satellites [40] and even space probes [50]. While all these problems are quite varied, all of them seek the same thing: given a large number of possible options, how to find the ones that give the best results, considering a set of criteria (such as cost, time, etc)?

Modeling this problem as an optimization problem and applying the appropriate methods to solve it could help users find better flights and accommodations for their trips in a shorter time, with no downsides. This could greatly improve the trip planning experience and become a valuable tool for a great number of users, both in personal and professional settings.

1.2 Objectives

This work aims to build a tool to solve the aforementioned trip planning problem using optimization methods. The resulting system should be able to find good solutions for a trip itinerary in a reasonable time.

The proposed solution should be able to:

- Get a trip itinerary as input from the user. This itinerary consists in a set of destinations to visit and specifications regarding each of them, such as stay duration, desired dates and (optionally) order in the itinerary. It is important that the system can handle flexibility in these dates if defined by the user, as well as other restrictions.
- Query a data source (real or not) for all the available flights and accommodations that are feasible regarding to the user's itinerary.
- Use the data gathered and optimization methods to find the best flights and accommodations combinations for the trip, minimizing cost and travel time.
- Present these options to the user and allow further filtering and selection based on the user's preferences between the criteria, aiding their decision.

The work does not aim to build a finished product ready for real-world application. However, it should assess the viability of such a tool and, if possible, build foundations that could be further used for achieving this goal.

1.3 Dissertation Outline

The remainder of this dissertation is organized as follows: Chapter 2 presents concepts needed for the comprehension of the work and a brief review of the travel planning literature. Chapter 3 describes the mathematical formulation of the problem and the methods used to solve it, including the data query system, the optimization techniques implemented and decision-making support methods used. The experiments performed and results obtained constitute Chapter 4. Finally, Chapter 5 presents some final considerations about the work and its applicability, summarizes the contributions and proposes ideas for future work.

Chapter 2

Theory Overview and Literature Review

This chapter presents an overview of some optimization concepts needed for the comprehension of the work. It also presents a brief review of trip planning problems found in the literature.

2.1 Trip Planning Literature

The planning problem presented in this work consists in finding the best combinations of flights and accommodations that minimize the cost and travel time for a trip. It is subject to several restrictions, such as dates and stay duration for each destination and the whole trip and accommodation and transportation preferences.

The proposed problem is an application of the Traveling Salesman Problem (TSP) with Time Windows Constraints [25]. This problem consists in finding the route with smaller cost that goes through all the cities in a set exactly once and returns to the departing city in the end. Each route between cities has an associated cost, and each city on the set must be visited within a specific time window.

This NP-complete problem [16] can be seen as the combination of the classic TSP problem with scheduling problem, and has many practical applications, such as routing, manufacture tasks scheduling and delivery planning. In this work, each node of the graph is a destination, and the edges between nodes represent a combination of a flight and an

accommodation for that destination. The time windows are optionally defined by the user.

There are many proposed solutions for the TSP with Time Windows with Constraints. In [10], an heuristic based on the solution of smaller subproblems and local search is presented. A Beam-ACO method, a hybrid between ACO and beam search, using stochastic sampling and local search is shown in [38]. A fuzzy approach to the problem is proposed in [43], which gives more flexibility to the modeling and achieves good results. General Variable Neighborhood Search methods are used in [14] to solve the TSPTW problem. In [35], a Discrete Artificial Bee Colony algorithm with local search is applied to the problem.

These solutions, however, aim to solve single objective instances of the TSPTW problem, unlike the one presented in this work. In addition, the problems solved are benchmark problems. There are no practical applications for trip itinerary planning problems similar to the proposed one.

A big part of the trip planning literature focus on the Tourist Trip Design Problem. The essence of the problem is planning a route between Points of Interest (POI) in a city, maximizing the satisfaction score. The problem must consider POI's opening times and the transportation options between them, and can also take in account additional restrictions such as budget or m number of POIs. These problems are usually formulated as *Orienteering Problems* or variations, such as *Team Orienteering Problems* or *Multi Constrained Team Orienteering Problem with Time Windows*.

The main difference between these problems and the proposed Trip Itinerary Planning Problem is that the Tourist Trip Design Problem do not require all the points of interest to be visited, while all destinations must be visited in the Trip Itinerary Planning Problem. In addition, only one possible route cost between POIs is considered, while many variations are present in this work's problem, which increases the complexity. Both are NP-hard and usually solved with heuristics and meta-heuristics.

There are many proposed solutions for the Tourist Trip Design Problem and its variations. In [6] a Fuzzy GRASP approach is presented with interesting results for a single objective problem. A similar problem is solved with a greedy algorithm in [3]. A taboo search method is proposed in [45] and applied to a time constrained problem. An alternative approach based on Simulated Annealing is presented in [46]. [27] introduces an iterated local search meta-heuristic for a multi constrained problem. A more generalist time-dependent extension is presented in [29] using randomized meta-heuristics.

The orienteering problem with hotel selection, an extension of the base problem, is discussed in [17]. A Skewed Variable Neighborhood Search is used to solve it, with problem-specific neighborhood structure is proposed as well. The inclusion of hotels in the optimization problem brings this problem closer to the Trip Itinerary Planning Problem considered in this work. However, it lacks other restrictions such as Time Windows.

There are other optimization research that, although focused on different problems, present interesting approaches that can be extended to the TSPTW. Among these, [52] presents a extensive review of graph search algorithms and proposes extensions for the methods, with focus on real-time solutions.

2.2 Optimization Concepts

An optimization problem can be seen as a search problem in which the aim is finding the best solutions among all the possible ones, given a set of criteria to evaluate the quality of these solutions. This criteria-based evaluation is represented as a mathematical function (known as *objective function*) that take as input a solution and return its quality value. The set of all possible solutions for a problem constitutes the optimization search space.

It is common that there are restrictions to the solutions of an optimization problem. Any solution that violates these restrictions is not valid, even if within the search space. Therefore, restrictions lower the total number of feasible solutions and limit them to a subset of the search space.

As an example, let us consider the allocation of manufacturing tasks to machines in a factory. The search space in this case is all the possible allocations. If the goal is to maximize production, the optimal solution might involve all machines working nonstop. However, there might be other restrictions that need to be considered, such as the need to stop for inspections periodically, or maximum electricity demand during peak hours during the day. Therefore, the optimal solution will be the one that maximizes production without violating any of the restrictions.

The mathematical formulation of an optimization problem can be defined [42] as:

$$\begin{aligned} &\text{minimize } \boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^m \\ &\text{subject to: } \begin{cases} g_i(\boldsymbol{x}) < 0, i = 1, \ldots, p \\ h_i(\boldsymbol{x}) = 0, i = 1, \ldots, q \\ \boldsymbol{x} \in \mathbb{R}^n \end{cases} \end{aligned} \tag{2.1}$$

where $\boldsymbol{f}(\boldsymbol{x})$ represents the set of m objective functions, $\boldsymbol{g}(x)$, $\boldsymbol{h}(x) \in \mathbb{R}$ the equality and inequality restriction functions and $\mathbb{R}^n$ the set of optimization variables.

2.3 Optimization Method Evaluation

When designing optimization algorithms, it is extremely important to consider the target problem characteristics, the desired use case and the context in which it will work. Most optimization problems can be divided into two main categories: one-off problems or repetitive problems [23].

In the first case, it is enough to find a good solution only once. Thus, quality usually takes precedence over execution time, since there is more time to solve the problem. Besides, variance between different executions is not a big concern, as long as one of these runs is able to find a good solution. Some examples of these problems is a university timetable scheduling, or determining the parameters of a motor for maximum efficiency.

Repetitive problems, on the other hand, have a more challenging balance between execution time and the quality of the results. In these problems, finding a "good enough" solution in a short period of time is often better than taking a lot of time to achieve a near-optimal solution. Besides, it is necessary to ensure that the variance of the method is not too high, since the available time might not be enough to run it several times. The route calculation on Google Maps is an example of such problem.

There are three commonly used metrics to evaluate the performance of an optimization meta-heuristic [23]: success rate, quality of the solution (mean or best case) or efficiency (mean number of function evaluations until a stop condition is met). Due to the stochastic nature of these methods, evaluations should always be based on statistical tests performed on several executions.

The success rate indicates how often the method reaches the optimal value. Other success indicators might be used when the optimal value is not known. The quality of the solution evaluates the mean result of the algorithm given an execution limit (such as computation time or number of functions evaluations, for example). Finally, the efficiency measures how much time the method takes on average to find a solution that reaches a predetermined quality criterion.

The most appropriate evaluation method depends on the problem context and the purpose of the method. The success rate is more common in problems where the optimal is known, or the solutions have a binary right-or-wrong nature. Fixed-time quality evaluation is often performed when the problem has a predefined and inflexible maximum execution time. Finally, the efficiency metric is more relevant when a determined quality of the solution is good enough.

Each metric might be evaluated differently according to the problem. Repetitive problems need to consider the mean and variance of the results over several executions, while one-off problems might be more interested in the best values of all the runs. Even the worst values reached by each method might be important to plan for worst-case scenarios in situations where it might be critical, for example.

In performance evaluations, time metrics might not be the best way to evaluate effort. Considering the system on which computational experiments are usually run, the execution time might be affected by external factors such as operational system processes, network bandwidth, etc. This might be unpredictable and introduce randomness to the experiments. These external factors should be minimized for time experiments when possible. An alternative solution to avoid such problems is to use computer effort metrics, since they depend only on the algorithm and, therefore, are more consistent. However, it is still important to also register the time, as it is extremely relevant for any real world application.

Often the number of evaluations of the objective function is used as a computational effort metric. This metric is appropriate for methods in which the other steps of the algorithm are trivially cheap compared with the function evaluation. If that is not the case, the metric might be misleading, e.g. for a method that uses a repair function costlier than the objective function evaluation.

A way to make a more complete and informative analysis is to consider the quality of the solutions over time. It might help differentiate a method that converges quickly but

has a worse convergence value from one that is slower but better after some time, such as shown in Figure 2.1. The choice of algorithm in this case depends on the use case.

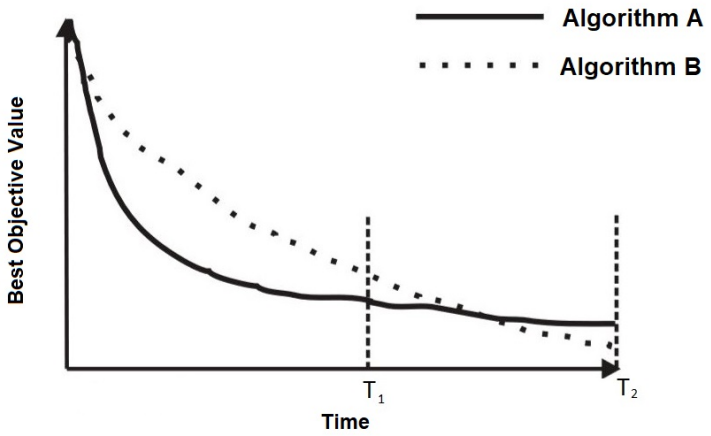


Figure 2.1: Quality over time for two optimization methods (source: [23]).

Finally, it is important to consider other problem characteristics that might affect the results. Different methods might have different behaviours according to the size of the problem, for example. Figure 2.2 depicts a situation like this, in which a method is faster for smaller problems but slower for bigger ones. In situations like this, it might be a good choice to use each method in the most appropriate situation.

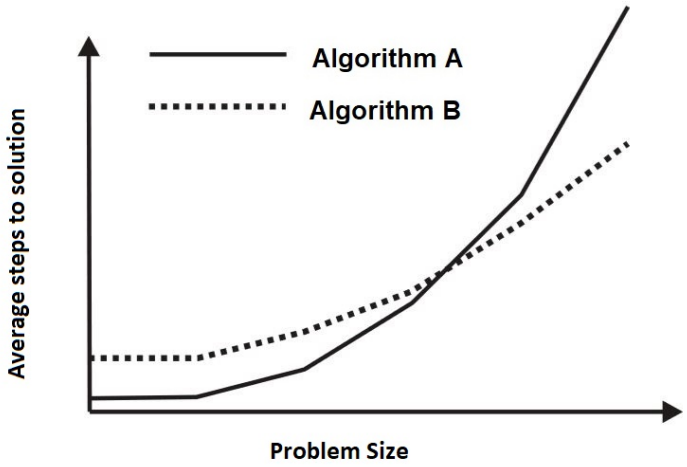


Figure 2.2: Comparison of two algorithms considering the average number of steps until reaching a solution versus the problem size (source: [23]).

2.4 Optimization Methods

Many practical optimization problems are NP or NP-hard. This means that there are no known deterministic algorithm that can solve them in polynomial time. Therefore, the exact solution is often unattainable in reasonable time.

In order to circumvent this obstacle, heuristics and meta-heuristics are used for optimization. While these methods do not guarantee convergence to the global optimum, they tend to reach good solutions in reasonable time, and are used to solve many real world problems [30].

2.4.1 Ant Colony Optimization

Ant Colony Optimization (ACO) is a populational meta-heuristic very commonly used to solve complex combinatorial problems [21]. It is based on the behavior of ants foraging food for the colony. As an ant walks through a path, it leaves behind a trail of pheromones which serves a signal to attract other ants. Trails that lead to close food sources are more often walked through, so the pheromone trail becomes stronger.

The algorithm works in a similar way. A set of artificial agents (the ants) explore the search space building solutions step-by-step. At each point of the way, the agent chooses the next solution component based on probabilities determined by a "pheromone value" and heuristic information. After each solution is built, the pheromone value of its components is updated based on the quality of the solution.

The general structure of an ACO algorithm [30] is:

```
initializePheromoneMatrix()
while(not_termination)
   generateSolutions()
   daemonActions()
   updatePheronomones()
end while
```

Initialization The pheromone matrix is initialized with values τ_0 for each component.

Solution Generation Each agent starts with an empty solution and, step-by-step, adds a new feasible component to its solution. The components are chosen with probability,

usually defined by the following distribution [20]:

$$p(c_i|s_p) = \frac{\tau_i^{\alpha}.[\eta(c_i)]^{\beta}}{\sum_{c_j \in N(s_p)} \tau_j^{\alpha}.[\eta(c_j)]^{\beta}}, \quad \forall c_i \in N(s_p), \tag{2.2}$$

where c_i is the solution component evaluated, $N(s_p)$ is the set of all feasible components for the partial solution s_p, τ_i the pheromone value of component c_i and $\eta(c_i)$ the heuristic evaluation function for component c_i. The parameters α and β balance the influence of heuristic and pheromones values on the probability.

Daemon Actions *Daemon Actions* might refer to central or problem specific actions that might help refining the solutions. A common step is performing local search on the solutions found [21].

Pheromone Update The pheromone value of a solution component influences the probability of it being chosen at the solution generation step. This mechanism helps differentiate the best components that lead to good solutions. To do so, the pheromone value for a component is updated based on the quality of the solutions using it. This update follows Equation 2.3:

$$\tau_i \leftarrow (1-\rho)\tau_i + \sum_{s \in S_{upd} | c_i \in s} g(s), \tag{2.3}$$

where S_{upd} is the set of generated solutions, $\rho \in (0, 1]$ the evaporation rate parameter and $g(s)$ the quality function for the solution.

There are several other variations for ACO, such as the use of elitism [19], solutions rankings that regulate the pheromone deposit rate [9], limits for minimum and maximum pheromone values [44] and more.

2.4.2 Local Search

Local Search (also known as Hill Climbing or Descent) is one of the oldest and simplest heuristics [48]. It starts from a given solution and search for a better one among its

neighbors. This iterative procedure continues until no better neighbor is found, meaning that a local optimum has been found. The general algorithm is:

```
s = generateInitialSolution()
while(True):
   N = generateNeighbors(s)
   s' = selectBetterNeighbor(N, s)
   if s' == null:
       return s
   else:
       s = s'
end while
```

Local Search can be seen as iterative steps descending in a graph representing the search space (Figure 2.3). It will always converge to a local optimum defined by its neighborhood structure.

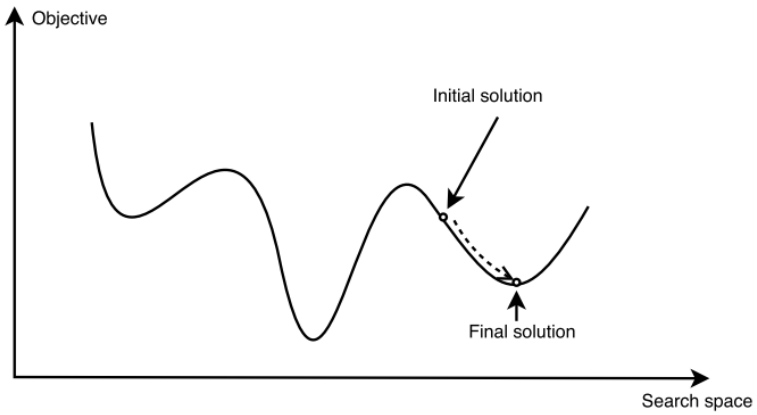


Figure 2.3: Local Search representation in a single objective continuous problem (source: [48]).

The main elements of a Local Search algorithm are the neighborhood structure and the selection strategy. The neighborhood structure refers to the method used to determine what are the neighbors of a given solution. It is particular to each problem and its data representation. In continuous, univariate problem such as the one represented in Figure 2.3, the neighbors for a solution x might be the set $(x + \Delta, x - \Delta)$, where Δ is a previously defined variation. For a Travelling Salesman Problem, a neighborhood of a solution can be defined as any solution obtained by swapping the order of any two cities in the itinerary while leaving the others in the same order.

It is important to note that the same problem can have different neighborhood structures. In the TSP case, for example, another possibility is to get any solution

that swaps the order of any three cities. The structure chosen impacts the size of the neighborhood and the search space connectivity, which have a big impact on the results.

Once the neighbors of a solution are known, the next step is to select which one will be chosen for the next iteration. This is where the selection strategy comes into play. Some common strategies are:

- **Best improvement:** Among all neighbors, the one that improves the most the objective value is selected. This method requires the whole neighborhood to be evaluated, which can be time consuming
- **First improvement:** The first improving neighbor in relation to the current solution is selected. This strategy does not require a complete evaluation of the neighborhood, except in the worst case.
- **Random selection:** Among all neighbors that improve the current solution, one is selected randomly.

In many applications, the first improvement strategy leads to the faster computation times without affecting the quality, although not in all cases.

Although the Local Search method is usually very fast and simple, its main disadvantage is that it always converges to a local optimum. There are many other neighborhood based heuristics that try to overcome this shortcoming. They can be structured in different families according to the strategy used to escape local optima:

- **Change the initial solution:** Iterative Local Search, GRASP
- **Accept non-improving neighbors:** Tabu Search, Simulated Annealing
- **Change the evaluation function or the input data:** Guided Local Search, Smoothing method, Noise methods
- **Change the neighborhood structure:** Variable Neighborhood Search (VNS)

2.5 Multi-objective Optimization

Real-world optimization problems often have more than one goal. Minimizing cost and maximizing productivity in an assembly line, reducing the size and increasing a motor efficiency, minimizing delivery time and fuel consumption for a delivery service, etc. In these types of problems, the objectives might be conflicting between each other. For such

cases, it is best to find a set of diverse solutions with good results for all the objectives, so the trade-off among them can be pictured and a choice made.

One way to solve a multi-objective problem is to transform it into a single-objective one and solve it, with an approach called *A Priori* Preference Articulation. These methods use a weighted combination of each objective function to create a single one that contains information about all the criteria [13] [15]. However, this type of approach requires the user to determine the relative importance of each objective before the set of solutions is known. This uninformed decision might lead to worse results. *A Posteri* Preference Articulation methods do not need this kind of previous decision, which makes them more general.

2.5.1 Dominance and the Pareto Frontier

A dominant solution is one that, compared to another, has better or equal quality values for all the criteria considered, and is better for at least one. Considering $\boldsymbol{a}$ as the array of quality values for a solution A for all m criteria in a minimization problem, and $\succeq$ the symbol that represents dominance, $A \succeq B$ is formalized as:

$$A \succeq B \iff \forall i \in \{1, ..., m\}\ a_i \leq b_i \wedge \exists i \in \{1, ..., m\} \mid a_i < b_i$$

For problems where the objectives are conflicting, there might not exist a single best solution that dominates all others. A non-dominated solution is one that is not dominated by any other. The set of non-dominated solutions evaluated in the objective space is called Pareto Frontier, represented on Figure 2.4. These solutions cannot be improved for a single criteria without getting worse for another. The goal for a multi-objective algorithm is finding all (or a representative set of) the solutions on the Pareto Frontier so that one of them can be chosen as the final solution afterwards.

The Dominance Rank of a solution is defined as the number of other solutions that dominate it. The solutions that belong to the Pareto Frontier have dominance rank equal to zero. This ranking can be used as measure of quality for a solution and even be incorporated to the objective function [32].

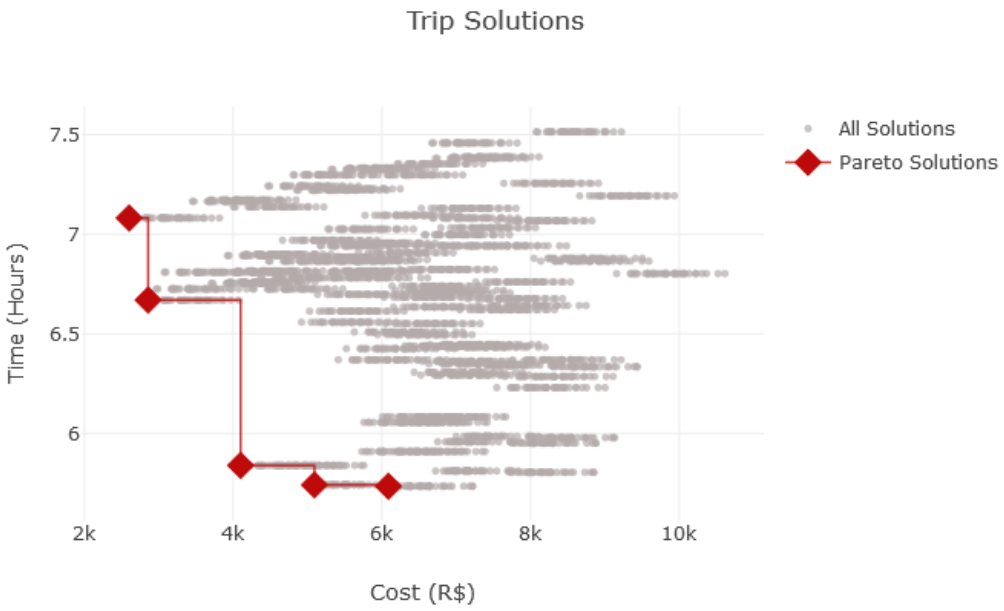


Figure 2.4: Pareto Frontier of a two objective optimization problem

2.5.2 Diversity Preservation for Multi-objective Problems

In many optimization problems, it is not possible to find all the Pareto solutions in reasonable time. In such cases, it is desirable not only to find as many solutions as possible, but also to find diverse solutions that are distributed for all regions of the frontier. This way, the possible solutions are well approximated, even if not all of them are known.

There are several methods to try and make the solutions found more diverse and avoid concentration in a region in the frontier. One of these methods is using *Crowding Distance* metric as another criterion to evaluate the solutions found. This metric represents how distant is the solution from its closest neighbors on the objective space. This value is determined by the area of a rectangle defined by its neighboring solutions. The higher the *crowding distance* value of a solution, the higher its distance is from others, which indicates high diversity.

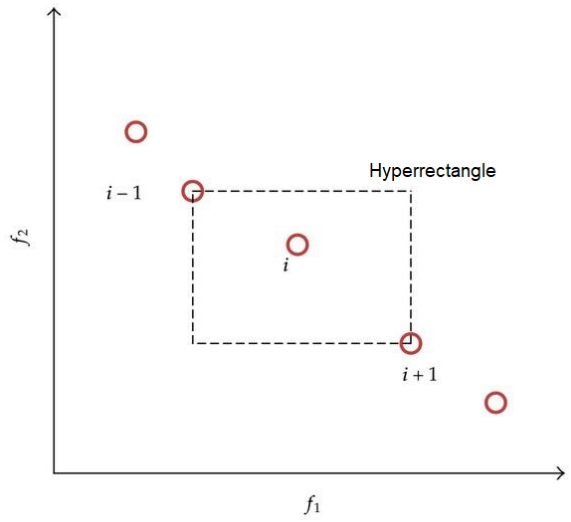


Figure 2.5: Crowding Distance of a solution in a two objective problem

2.5.3 Multi-objective Ant Colony Optimization (MOACO)

[28] and [2] review several multi-objective approaches for the Ant Colony Optimization algorithm, and a taxonomy is presented to classify different methods according to their characteristics. The categories defined in [2] are briefly discussed below.

Pheromone Matrix It might be single or multiple. In the first case, there is a single matrix and a single pheromone value for each solution component. The alternative is having multiple matrices, one for each objective. This second approach usually results in more exploration (as opposed to exploitation) of solutions and is beneficial to find a more diverse set of solutions.

Solution Generation In an ACO algorithm, solutions are built through an iterative process based on heuristic and pheromones values of each component. For multi-objective problems, this process can happen in three different ways: (a) directed, using information about only one objective; (b) fixed, using a combination of information regarding all the criteria, balanced by a priori defined weights; and (c) dynamic, using information of all criteria combined with weights that change during the execution, or even differ from ant to ant, in a way that directs the algorithm to explore different regions of the Pareto frontier.

The dynamic method is useful on problems where the Pareto frontier characteristics are not previously known, which makes it hard to define weights. It favors a balanced distribution of solutions across the whole Pareto front [28]. In such cases, an expanded version of the random proportional equation might be used. A generic form is presented in Equation (2.4), where $N(s_p)$ represents the set of feasible components for the partial solution s_p, $c_i \in N(s_p)$ represents a component of the $N(s_p)$ set, L is the number of pheromone matrices, τ_i^l is the pheromone value for component c_i in the l-th matrix, M is the number of heuristic functions used and η_i^m is the m-th heuristic values for component c_i.

$$p(c_i|s_p) = \frac{\prod_{l=1}^{L}(\tau_i^l)^{\alpha_l} \cdot \prod_{m=1}^{M}(\eta_i^m)^{\beta_m}}{\sum_{c_j \in N(s_p)}(\prod_{l=1}^{L}(\tau_j^l)^{\alpha_l} \cdot \prod_{m=1}^{M}(\eta_j^m)^{\beta_m})}, \quad \forall c_i \in N(s_p), \tag{2.4}$$

Solution Evaluation Solutions might be evaluated in two ways: based on their dominance rank or, in a more traditional fashion, according to their value for each objective. Additional evaluation metrics (such as crowding distance) might be used in a complementary way to any of the two main evaluation methods.

Pheromone Update The update of the pheromone values might happen in an individual way, affecting only one solution matrix for each solution found, or in a global way, updating all the matrices. This decision is usually tied to the solution generation method, that might use a single matrix or a combination of them all. The update follows the Equation:

$$\tau_i^l \leftarrow (1-\rho)\tau_i^l + \gamma \sum_{s \in S_{upd} | c_j^i \in s} g^l(s) \tag{2.5}$$

where l is the pheromone matrix index, S_{upd} is the set of solutions generated in current iteration, $\rho \in (0, 1]$ the evaporation rate parameter, γ the pheromone deposit rate parameter and $g^l(s)$ the quality function that evaluates solution s considering criterion l.

Pareto Archival After each cycle of an ACO algorithm, a *daemon action* might be used to store the best solutions (current Pareto set estimated) found so far. This storage

might occur as an offline storage, which simply saves and updates the best solutions found so far in a *hall of fame* fashion, but does not use these solutions in any way. Alternatively, online storage approaches keep the same set saved, but uses this information during other steps of the cycle (usually in the pheromone update step). Other approaches include keeping a single elite solution or keeping none and simply returning as the final solutions the set obtained after the last cycle.

An important consideration here is that saving the best solutions and keeping the set of Pareto-approximation updated incurs in an additional computational overhead, as the set needs to be reevaluated after each cycle when new solutions are obtained [59], [58]. For problems with higher sets and dimension, this cost might be significant.

2.5.4 Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D)

The Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [41] is a relatively recent multiobjective optimization framework. It proposes decomposing a multiobjective problem into N scalar optimization sub-problems and optimizing them simultaneously.

The objective function of a sub-problem is a weighted aggregation of the original problem objectives. Two common approaches for decomposition are the Weighted and Tchebycheff methods. On the first one, a scalar sub-problem is given by:

$$\text{minimize } g(x|\boldsymbol{\lambda}) = \sum_{i=1}^{m} \lambda_i f_i(x) \tag{2.6}$$

where $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_m)$ is the weight vector for a problem with m objectives, with $\lambda_i \geq 0 \quad \forall i \in (1, ..., m)$ and $\sum_{i=1}^{m} \lambda_i = 1$. If the optimal Pareto Front is convex, generating N weight vectors equally distributed through the objective space works well. In other situations, other approaches might be necessary.

The Tchebycheff approach decomposes problems based on the following equation:

$$\text{minimize } g(x|\boldsymbol{\lambda}, \boldsymbol{z^*}) = \max_{1 \leq i \leq m} (\lambda_i (f_i(x) - z_i^*)) \tag{2.7}$$

where $\boldsymbol{z^*} = (z_1^*, ..., z_m^*)$ is the reference point defined by $z_i^* = \min (f_i(x) | x \in \Omega) \quad \forall i \in (1, ..., m)$.

Each sub-problem optimization uses information of its neighboring sub-problems. The neighbourhood structure is defined based on the closeness of the aggregation weight vectors. The size T of the neighborhood is an input parameter of the method.

The general framework for the MOEA/D method initializes by decomposing the original problem using any decomposition technique, defining the neighbors of each sub-problem and an initial population. Then, the algorithm iterates until a previously defined stop criterion is reached. In each iteration, a new solution is generated for each sub-problem using a genetic operator and two solutions from the neighborhood, randomly selected. After that, for each neighbor sub-problem, replace its solution with the new generated one if it is better regarding that sub-objective. An external Pareto set is then updated if needed.

The MOEA/D framework has been successfully applied to many areas and is specially adequate for many-objective problems. There are several improvements for it, but the general framework usually stays the same. A thorough review on decomposition based methods is presented in [53], presenting many variations and their impact on the framework.

2.5.5 MOEA/D-ACO

The MOEA/D-ACO method [36] is a multiobjective evolutionary algorithm based on the combination of Ant Colony Optimization (ACO) and MOEA/D methods. This approach combines the decomposition into sub-problems, characteristic from MOEA/D, with the ACO framework.

The first step of the method is the decomposition of the main problem into N sub-problems, using either the weighted or Tchebycheff method. Each resulting weight vector is assigned to an ant, which also keeps a corresponding heuristic information matrix.

After that, all ants are divided into K groups by clustering the weight vectors. Each group shares a pheromone matrix. In addition to the groups, each ant has a neighborhood defined by the ants with the T closest weight vectors (including itself). Thus, the neighbors are not necessarily in the same groups.

The algorithm then follows the traditional ACO framework. The pheromone and heuristic matrices are initialized and, at each iteration, every ant builds a solution. In the solution construction step, in addition to using its own heuristic matrix and its group's pheromone matrix, the ant also uses the solution of its neighbors in the process. Solution components present in a neighbor current solution is favored in the probability calculation.

After each ant generates a new solution, they are used to update the corresponding pheromone matrices and an external archive of non dominated solutions is updated. Only non dominated solutions are used to update the pheromone matrix.

Finally, every ant updates its own solution. A new solution is chosen among all solution generated by the ant's neighbors if the new solution has a better objective value for that ant's sub-problem and it is not used by any other neighbor. This process continues until the stop criterion is met.

The main advantage of this method in comparison to the regular ACO is that, due to the decomposition and separate pheromone matrices, different ants focus on different areas of the Pareto Front, which increases solution diversity and favors exploration of the search space. The simplified single-objective sub-problems also improve the computational times at each step.

2.5.6 Hybrid Meta-heuristics and Two-Phase Pareto Local Search

Hybrid meta-heuristics are methods that combine two or more meta-heuristics, or even other optimization approaches. For many real-life or classical optimization problems, the best solutions are obtained with hybrid algorithms [47]. There are many types of combinations [48] and implementation approaches that should take into account the problem and environments characteristics.

These hybridization techniques try to combine the strengths of each algorithm in order to improve the overall performance and quality. Although the No Free Lunch

theorems [54] state that generalist black-box algorithms tend to have the same overall performance over the entire set of optimization problems, a particular approach can still be better for a specific problem.

The Two-Phase Pareto Local Search is a hybrid approach for multiobjective problems, such as bi-objective TSP [39]. The method uses a two-step approach. On the first one, an algorithm is applied to generate an initial set of solutions. In the second step, Pareto Local Search is applied to each solution, refining the results obtained on the first method. The initial step favors exploration of the Pareto Front, while the second exploits and improves the solutions found.

2.5.7 Performance Metrics for Multiobjective Problems

For single objective optimization problems, the performance of a metric is easily evaluated by the quality of the solution found. For multiobjective problems, there is not a single solution as result, but a set of them, so it is not possible to use the same method.

Some common methods for these problems are *Hypervolume* [58] and *Epsilon* indicator [57]. The first one calculates the hypervolume defined by a set of solutions and a fixed reference point in the objective space. The bigger the hypervolume, the better the solution set, and the Pareto optimal set has the best hypervolume possible for a problem. The Epsilon metric is based on the comparison of a solution set with an ideal reference set. It evaluates the change rate needed so that the evaluated solution set is no longer dominated by the reference set, and lower rates have better scores.

These methods, however, require some knowledge of the Pareto frontier. The *Summary Attainment Surface* method proposed in [26] has as its main advantage over other methods its independence of problem specific knowledge. It is based on the distribution of the solutions on the obtained approximated Pareto set and statistical methods.

Still, the hypervolume method is among the most used performance metrics. Besides having good results, it is very simple and intuitive method. Figure 2.6 shows hypervolume for a two objective problem.

As it can be seen, the closer the evaluated set is to the Pareto frontier, the higher the hypervolume will be.

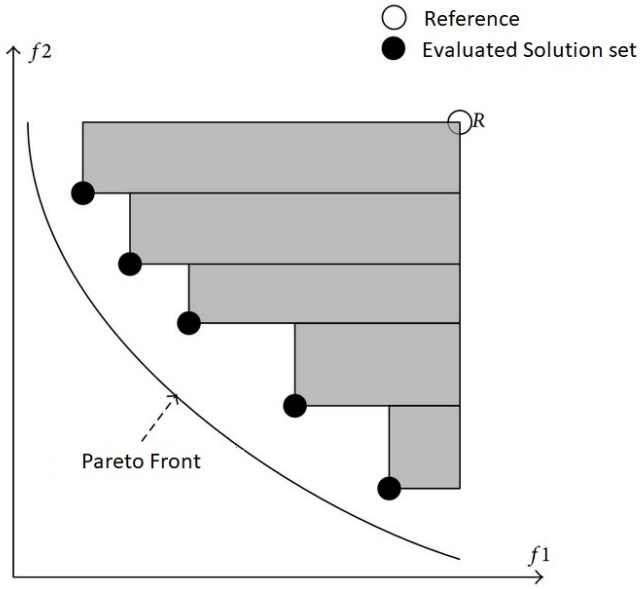


Figure 2.6: Hypervolume evaluated for a set of solutions in a two objective problem (adapted from [37])

2.6 Hyperparameter Selection

Optimization meta-heuristics have many hyperparameters that regulate them. They might be numerical (e.g. pheromone evaporation rate or number of ants in a ACO) or symbolic (*daemon actions* used). These parameters have a significant impact on the algorithm's performance, which makes their choice an important decision.

Performance evaluation methods presented on Section 2.3 can be used to help determining the set of hyperparameters that lead to the best result for the desired use case. It is important to note that many parameters are interdependent. As a consequence, it is not possible to tune one parameter at a time since their interactions need to be evaluated [48].

However, most methods have a significant number of parameters, many of which have many possible values. In order to fully evaluate the parameters effects, there should be performed experiments with all possible combinations. This leads, however, to a unfeasible number of experiments. For a problem with n parameters with k levels, n^k experiments would be necessary.

The fact that many meta-heuristics methods are stochastic makes the situation even more challenging. In order to obtain statistically significant results, many runs of the

same experiments need to be performed, which increases even more the number of executions required.

One alternative to reduce this number is to use fractional factorial designs, in which the number of experiments necessary is greatly reduced. However, even this approach might be unfeasible for methods with many parameters and levels.

Another approach is to treat the search for a good set of hyperparameters for an optimization method as another optimization problem itself. Thus, this problem would be solved with another optimization algorithm. There are several methods in the literature that focus on this very issue [22] and [24]. However, this approach can also be very computationally expensive.

In situations where the complexity of the original optimization problem makes it hard to perform a complete hyperparameter optimization, compromises are necessary. In such cases, smaller statistical studies of the effect of each hyperparameter can be performed to identify the most important ones to focus on, as well as to have a better understanding of their effects on the method.

2.7 Decision Support Methods

Multi-objective optimization methods produce a set of solutions as result, instead of a single solution. These solutions are non-dominated, which means that they all present trade-offs in relation to each other and the objectives. However, for real-world applications, only one solution can be applied. Choosing one among many is challenging for the final users, specially when the number of solutions is high.

Decision support methods aim to help the user sort the final solutions according to their preferences and select one. These methods depend on the set criteria that are important to the decision maker, and can be either deterministic, stochastic or fuzzy [51]. They can also be for a single decision maker or a group.

Methods based on numerical analysis of the alternatives require first of all that the set of relevant criteria $\boldsymbol{C} = (C_1, ..., C_n)$ are defined. After that, it is necessary to attribute to each of the m alternative solutions a numerical value for each criteria: $a_{ij}\ \forall i \in (1, ..., m),\ \ j \in (1, ..., n)$, aggregated in matrix $\boldsymbol{A}$. A weight vector $\boldsymbol{w} = (w_1, ..., w_n)$ for the set of criteria is also defined. It is important that all values are attributed within the same scale.

Finally, these numerical values are processed and a scalar score value is attributed to each alternative. A rank for the solutions is produced based on this score. Each method employs different processing techniques.

One of the most common approaches is the Weighted Sum Model (WSM). It attributes to each alternative a score S_i defined by Equation 2.8, where a_{ij} is the value of the *i-th* alternative in terms of the *j-th* criterion and w_j is the weight of the *j-th* criterion. The ranking is created based on the descending order of scores. This method is one of the simplest ones and is based on the additive utility assumption.

$$Si = \sum_{j=1}^{n} a_{ij} w_j \tag{2.8}$$

A variation of the WSM approach is the Weighted Product Model (WPM). This method evaluates the product of the relative value of each pair of alternatives $i, k \in (1, ..., m)$, for each criteria. The main advantage of this method compared to WSM is that the values considered are dimensionless, since only the ratio is considered. The score attributed to each alternative A_i is given by:

$$Si = \sum_{k=1, k \neq i}^{m} \prod_{j=1}^{n} (a_{ij}/a_{kj})^{w_j} \tag{2.9}$$

Other methods include AHP, ELECTRE, TOPSIS and PROMETHEE (and their variations). They are a bit more complex and some require pairwise comparisons, which require more input from the decision maker and might not be adequate for cases with many alternatives. For a bi-criteria case, WSM and WPM are adequate and very simple for the end user. Besides, they can be easily tweaked by changing the criteria weights.

2.8 Summary

This chapter presents a review of the main concepts and methods used in this work and presents an overview of trip planning literature works. These concepts include optimization concepts, Ant Colony Optimization and Local Search methods, Multi-

objective concepts and methods (MOACO, MOEA/D, MOEA/D-ACO), hyperparameter selection and Decision Support methods.

Next chapter presents the methods used for the solution of the trip planing problem presents, including modeling of the problem, data gathering and generation, optimization methods and decision support methods, applying the concepts presented here.

Chapter 3

Methodology

The trip planning problem presented in Chapter 1 consists in helping a traveller find the accommodations and flights for a trip that minimize the cost and travel time. This chapter presents in detail the methods used to solve this problem and each step of the solution, covering the mathematical modeling, the data gathering, the optimization algorithms and the decision support steps.

3.1 Problem Representation

The first step to solve any problem is to gather information and clearly define it. In order to do so, it is necessary to gather user input. Only after they have clearly defined the trip they want the system can start to optimize it. One of the goals of this work is to let the users have as much flexibility as possible to define their trip. Therefore, the system lets them define:

- The set of destinations to be visited;
- The length of the stay in each of them, with liberty to define minimum and maximum days, required date ranges to be in the destination and the order of it in the trip
- The departing destination (and, optionally, returning);
- The minimum and maximum desired dates to start and finish the trip;
- Transportation restrictions, such as company, class and preferred flight hours during the day;
- Accommodation restrictions, such as type, location and category.

This section proposes a data representation and mathematical model for the problem.

3.1.1 Data Structure

This section defines how the problem data is structured for the optimization. This data is divided between user inputs and solutions components (flights and accommodations information gathered).

User Inputs

Starting Destination: Starting destination of the trip.

Departure Dates: Date interval $(date\_min, date\_max)$ in which the trip must begin.

Arrival Dates: Date interval $(date\_min, date\_max)$ in which the trip must end.

Destinations ($\boldsymbol{D}$): Set of the k destinations to be visited on the trip. Each destination represents a city.

$$\boldsymbol{D} = \{D_1, \ldots, D_k\}$$

Each $D_i \in \boldsymbol{D}$ is a structure with the following information:

- **id**: unique identifier for each destination.
- **stay_length**: interval of integers *(stay_ min, stay_ max)* that represents the desired stay length (in days) for destination D_i.
- **date_interval**: interval of dates $(date_1,\ date_2)$ in which the traveler must be in destination D_i. Optional input.
- **order**: integer in the interval $[1,\ k]$ that the determines the order in which destination D_i must be visited in the trip (e.g. $destination\_order = k$ indicates that the destination must be the last one visited). Optional input.
- **accommodation**: boolean that indicates if an accommodation is necessary for destination D_i in the trip. Defaults to *True*.

The notation $X_{i.property}$ is used to represent an attribute of structure i (e.g. $D_{i.stay\_length}$ indicates the stay length interval for destination D_i).

Round Trip: Boolean that represents if the trip should end back in the initial destination. Defaults to *True*.

Transportation Restrictions (R_T): Structure with optional transportation related restrictions:

- **type**: types of desired flights (i.e. first class, business, economic).
- **hours**: time interval in which flights should not depart (e.g. $(23{:}00, 05{:}00)$ indicates that flights that depart between 23pm and 5am should not be considered valid options for the trip plan).
- **companies**: set of flight companies that should not be included in the search.

Accommodation Restrictions (R_A): Structure with optional accommodation related restrictions:

- **type**: set of acceptable accommodation types for the trip (e.g. hotel, hostel, Airbnb, etc.).
- **location**: set of regions (defined by geographic coordinates and radius from center) in which accommodations must be located.

Solution Components

Solution components for the problem are flights (transportations) and accommodations from which proposed trip itineraries (the problem solutions) are built.

Transportations (T): Set off all transportation options between destinations of set D. Each $T_i \in T$ is an structure with the following properties:

- **id**: unique identifier.
- **type**: transportation category (e.g. first class flight, business flight, etc.).
- **cost**: total cost of the ticket.
- **departure_destination**: id of the destination from which the flight departs from.
- **arrival_destination**: id of the destination where the flight arrives.
- **departure_time**: departure date and time.
- **arrival_time**: arrival date and time.
- **company**: flight company.
- **departure_location**: information about the departure location.
- **arrival_location**: information about the arrival location.

Accommodations ($\boldsymbol{A}$): Set off all accommodation options for destinations of set $\boldsymbol{D}$. Each $A_i \in \boldsymbol{A}$ is an structure with the following properties:

- **id**: unique identifier.
- **name**: name of the accommodation.
- **type**: accommodation type (e.g. hotel, hostel, etc.).
- **cost**: total cost for all days.
- **destination**: id of the destination where the accommodation is located.
- **date**: date interval for the accommodation.
- **checkin_time**: check-in time in the accommodation.
- **checkout_time**: check-out time in the accommodation.
- **location**: information about the accommodation location.

3.1.2 Solution Structure

In the trip planning problem presented, the goal is to find itineraries that minimize the cost and travel time. Any itinerary that is a possible solution for the problem consists of a set of flights that ensure each destination is visited and an accommodation for each of them, respecting user inputs and restrictions. Thus, a solution s can be represented by a list of alternated transportation options and accommodation options. For k destinations, that would be:

$$s = (T_1, A_1, T_2, A_2, \ldots, T_k, A_k, T_{k+1}) \tag{3.1}$$

(considering accommodations for all destinations and a round trip).

3.1.3 Objective Functions

Each proposed itinerary (a solution s) must be evaluated considering the optimization problem criteria: cost and travel time. The functions that take a solution as input and return its value for each criterion are called *objective functions* and defined as:

Cost

Defined simply as the sum of costs of all transportations and accommodations of the solution.

$$F_1(s) = \sum_{i=1}^{k+1} T_{i.cost} + \sum_{j=1}^{k} A_{j.cost} \tag{3.2}$$

Travel Time

The travel time is only affected by the transportations. It considers the sum of the difference between the departure and arrival times for each flight, converted to seconds.

$$F_2(s) = \sum_{i=1}^{k+1} T_{i.arrival\_time} - T_{i.departure\_time} \tag{3.3}$$

There are some not accounted factors that might affect travel time besides the flights duration. A significant one is the distance between the airports and the accommodation in the destinations. Some airports, for example, are located very far from the most touristic areas in cities, and the time to reach the airport might have an impact in the total travel time.

This time could be estimated based on the locations of accommodations and airports, using API's such as Google Maps or similar alternatives. However, searching for estimated times between each pair of accommodations and flights would not be practical, given the number of requests needed and the time it would take.

A possible alternative is estimating the travel time between absolute distance, disregarding traffic and routes between hotels and airports. However, this could distort the results and make them inaccurate and even misleading. In the end, it could bring more harm than good.

Finally, a third possibility is keeping a database of distance between airports and city regions for the most sought after destinations. This could reduce the computational time needed to estimate the accommodation-airport travel times and still provide good approximations. However, implementing this solution would be time-consuming and the returns with little significance. Thus, the travel time between airports and accommodations are not considered in the proposed solution.

3.1.4 Problem Restrictions

A solution $s = (T_1, A_1, T_2, A_2, \ldots, T_k, A_k, T_{k+1})$ is not valid for any combinations of transportations and accommodation options. It is necessary to ensure that they are feasible and respect the restrictions of the defined problem, such as visiting all destinations exactly once, or that there accommodations selected for each day of the trip in the correct destinations and dates.

These feasibility checks are defined in this subsection. Set $\boldsymbol{R}$ contains all these solution restrictions.

Destinations visited: Ensures that each destination is visited exactly once on the itinerary.

$$\bigcup_{i=1}^{k} A_{i.destination} = \bigcup_{i=2}^{k+1} T_{i.arrival\_destination} = \bigcup_{i=1}^{k} T_{i.departure\_destination} = \bigcup_{i=1}^{k} D_{i.id} \tag{3.4}$$

Destinations connections: Ensures that there are arrival and departure transportations for each destination and that they are correctly ordered.

$$T_{i.departure\_destination} = T_{i+1.arrival\_destination} \quad \forall i \in \{1, \ldots, k\} \tag{3.5}$$

Accommodations: Checks if there is an accommodation for each destination and that they are correctly ordered.

$$A_{i.destination} = T_{i.arrival\_destination} \quad \forall i \in \{1, \ldots, k\} \tag{3.6}$$

Stay length: Ensures that the stay length in each destination respects the user defined limits, checking arrival and departure dates for transportations and accommodations dates.

$$\begin{cases} D_{i.id} = T_{i.arrival\_destination} \;\Rightarrow\; (T_{i+1.departure\_time} - T_{i.arrival\_time}).days \in D_{i.stay\_length} \\ D_{i.id} = A_{i.destination} \;\Rightarrow\; A_{i.date} \subseteq D_{i.stay\_length} \end{cases} \tag{3.7}$$

$$\forall i \in \{1, \ldots, k\}$$

Accommodation dates: Checks if the date interval for each destination's accommodation is the same as the interval between arrival and departure dates for this destination.

$$\begin{cases} A_{i.destination} = T_{i.departure\_destination} \;\Rightarrow\; A_{i.date\_end} = T_{i.departure\_date} \\ A_{i.destination} = T_{i.arrival\_destination} \;\Rightarrow\; A_{i.date\_start} = T_{i.arrival\_date} \end{cases} \tag{3.8}$$

Destination date interval: Checks that the stay dates in each destination is within its defined date interval.

$$D_{i.date\_interval} \neq \emptyset \Rightarrow \begin{cases} A_{i.date} \subseteq D_{i.date\_interval} \\ (T_{i.arrival\_time} \cup T_{i+1.departure\_time}) \subseteq D_{i.date\_interval} \end{cases} \tag{3.9}$$

$$\forall i \in \{1, \ldots, k\}$$

Destination order: Checks destinations ordering according to user specifications

$$D_{i.order} \neq \emptyset \Rightarrow D_{i.order} = i \quad \forall i \in \{1, \ldots, k\} \tag{3.10}$$

The accommodation and transportation restrictions sets ($\boldsymbol{R_A}$ and $\boldsymbol{R_T}$) are not treated as optimization restrictions. Instead, they are used beforehand to filter the transportations and accommodations sets $\boldsymbol{T}$ and $\boldsymbol{A}$), right after the data gathering step.

3.1.5 Search Space

The search space for an optimization problem is the set of all existing solutions. Considering the solution structure proposed in (3.1) and the sets of transportation and accommodation options $\boldsymbol{T}$ and $\boldsymbol{A}$, it is possible to represent the search space for the trip planning problem as a graph. In this representation, the graph's edges are solution components (transportation and accommodation options), while the nodes are destinations. If all the edges are added in a way that respects all restrictions in $\boldsymbol{R}$, the graph can be guaranteed to contain only feasible solutions.

Figure 3.1 shows a search space graph for a 2-destination trip containing only feasible solutions. The t-edges represent transportation between two destinations, while a-edges represent accommodations. The graph assumes a tree topology, and any path starting from the origin node (the starting destination) to a leaf node on the graph represents a

solution. If the cost and travel times are added to the edges, the optimization problem becomes a graph search one.

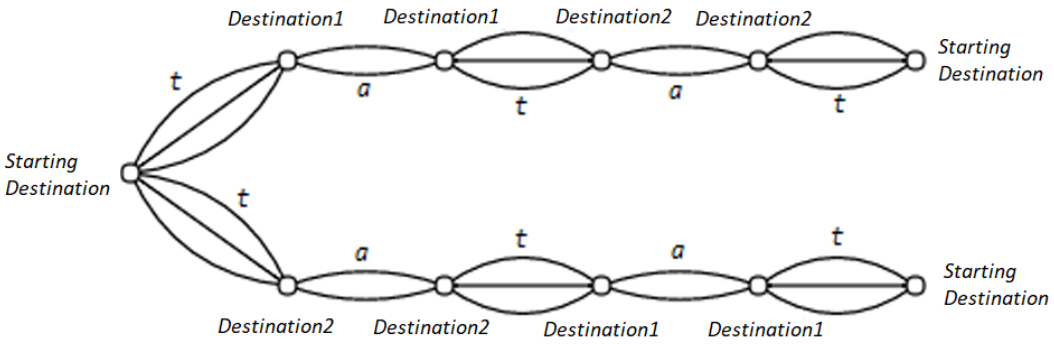


Figure 3.1: Feasible solutions search space graph for a 2 destinations unordered trip

This visualization also helps estimate the size of the feasible search space $\boldsymbol{S}$. Taking k as the number of destinations on the trip, p the average number of feasible transportation options between any two destinations and q the average number of feasible accommodation options in a destination, it is possible to estimate the number of feasible solutions (i.e. the search space size $|\boldsymbol{S}|$) as:

$$|\boldsymbol{S}| = (k)!\ p^{k+1}\ q^{k} \tag{3.11}$$

It is also possible to evaluate the size of the complete search space (which includes unfeasible solutions). To do so, the same equation applies, but using the total number of transportation options ($|\boldsymbol{T}|$) as p and the total number of accommodation options ($|\boldsymbol{A}|$) as q. This would result in a much bigger search space, as $|\boldsymbol{T}|$ and $|\boldsymbol{A}|$ are orders of magnitude higher than p and q. This fact makes optimization methods that can be restrained to the feasible search space and avoid unfeasible solutions preferred over others.

3.2 Data Gathering, Processing and Generation

3.2.1 Data Source

In any optimization problem, the information available to use is a key factor that restricts the quality of the solutions reached. The problem data determines the search space, which in turn determines the limits of all solutions that can be reached. Thus, using wrong or incomplete data can be harmful to the quality of the final solutions. The solution can only be as good as the information available. Thus, data gathering is a key step to solve an optimization problem.

For the trip itinerary optimization problem, the data set should have information about all possible accommodations in the desired destinations and flights between them. This information, however, is not so accessible.

The first obstacle is fragmentation. There is no single source with information about all these flights and hotels. Each flight and accommodation company usually has its own website with a selling or reservations system. Luckily, there are websites and services specialized in aggregating this information. For flights, some of the main ones are Skyscanner, Expedia or Decolar.com. As for accommodations, Expedia, booking.com and Trivago are among the best platforms.

Even so, having access to this data on demand is not that easy. While many sources do have the necessary information, they are not open for anyone. Access is usually restricted to business partners and paying customers. While some websites offer some sort of trial access to developers, the evaluation process is slow and the access is usually restricted.

Web crawlers could be a alternative to get this information, scrapping it directly from flights and hotels websites. However, due to the high number of different data sources, the task is complex and beyond the scope of this work.

As of yet, it was only possible to get a limited access to Skyscanner's API. However, restrictions regarding the number of requests and the information available makes it impossible to use this in real world, practical situations. Some data was gathered through their API to create test cases, but real-time integration was not possible.

In order to be able to test the proposed optimization system, an alternative method was necessary. A Random Information Generator was created, simulating queries to these

APIs. It receives as input a request for flights and accommodations info and, based on the request, created random mock-up data, which is then returned to the optimization problem. From the optimizer point of view, it makes no difference. This system makes it possible to test for different trips and itineraries configurations anytime, without need to gather the information elsewhere.

Although the lack of real data invalidates the proposed system for real-world use, the method development was not affected by it. The search space structure remains the same, and, therefore, the optimization method used on the mock-up data is also applicable to real data without any change. The Random Information Generator can be replaced anytime for a data-gatherer module that is able to access real flight and hotel data and, without any further change, the system would be able to plan real trips.

3.2.2 Data Querying

Once user inputs are gathered, it is necessary to analyze the desired trip information and determine the data necessary to plan and optimize it. In the most general case, the system could search for all the possible accommodations in every destination within the maximum trip date range, and for every flight between every two pair of destinations in the same period. However, this approach presents some problems.

First, the amount of data generated would be huge. This could slow down the optimization methods search for feasible solution components among a set so big. Second, querying for so much information would take a long time and could create an overhead to the process before the optimization even begins. While this problem is smaller with the Random Information Generator, it would be more serious if an API was used. The elevated number of requests required would slow down the performance and could even surpass limits of the API.

In order to avoid these issues, the system should use trip definitions and restrictions to reduce the amount of information gathered. If the feasible date ranges for each destination and for each flight route is determined, it is possible to eliminate unnecessary requests and trim down the number of queries performed. As an example, let us consider a situation where a user defines that a hypothetical *Destination 1* should be the first one visited and that its stay length should be no more than 5 days. In this case, it is not necessary to search for any flight or accommodation information related to *Destination 1*

beyond the fifth day of the trip. This sort of analysis can drastically reduce the amount of data requested and speed up the system.

One way to determine the necessary requests is to calculate exactly all the possible destinations permutations and dates, based on the trip restrictions. After that, each destination would have a range of possible dates in which it could be placed in the itinerary, and only information within this range would be queried. However, the computational complexity of this method is in the order of $\mathcal{O}(n!)$ for the number of destinations. For trips with 7 or 8 destinations, for example, the number of permutations is already in the order of thousands. In such cases, this verification process becomes too costly and creates a big overhead. Thus, it solves the costly queries problem but creates a new one.

Another method was proposed to reduce the queries and the data gathered, but with a smaller complexity. This method uses the same principle: based on trip definitions and restrictions, determine the range of possible dates for each destination and restrict the query to those dates. However, it trades exactness for efficiency. Each destination's date range calculated is not guaranteed to be the minimum possible range for the destination, but they can be calculated in reasonable time and still greatly reduce the number of queries.

Finally, there is a final, complementary method used to reduce the number of queries performed. All recent queries are stored in a database in the system, together with their return value. If a repeated query is performed in a short period of time, the information stored on the database is used instead, avoiding repetition and speeding up the return. While this solution is not impactful under the current system structure, it might be very useful for cases with several simultaneous users. In such cases, the number of repeated queries might be high and the proposed method would greatly reduce the number of API requests.

3.2.3 Search Space Reduction

Once the data queries are defined and the data is gathered (or randomly generated), the flight and accommodation information is modelled as the structures defined is section 3.1. Then, entries that violate any user defined accommodation or transportation restrictions are discarded. At this point, all information required to start the optimization process is available.

Before starting the optimization, however, there is a previous data processing step that aims to improve the efficiency of the optimization method by reducing its search space. This is done by identifying transportation and accommodation options that will never contribute to an optimal solution and discarding them.

This process is based on the Dominance concept presented on Section 2.5.1. A solution component C_A is strictly worse than another component C_B if all of their attributes (such as date, destination, etc.) are equal, but it has a worse value for at least one of the optimization criteria (cost or travel time). In these cases, for any solution S_A where C_A is present, there would exist a solution S_B with the exact same components, except using C_B in place of C_A. This solution S_B would **always** dominate S_A. Thus, eliminating component C_A from the search space would have no negative effect on the optimization quality.

The proposed method uses this idea to reduce the search space, comparing every pair of components of the same type and removing those that are strictly worse compared to another (as long as the other attributes are the same). This can reduce the search space significantly and, as a consequence, improve speed and quality in the optimization process.

3.3 Optimization

This section covers the optimization methods used to solve the trip itinerary planning problem. It describes the implementations choices for the chosen method (Ant Colony Optimization), explains the reasons for this choice and covers other improvements and variations.

3.3.1 Multiobjective Ant Colony Optimization

A Multiobjective Ant Colony Optimization framework was chosen as the baseline method for the trip optimization problem. c Considering the taxonomy presented in (2.5.3), the implemented method has the following characteristics:

- Multiple pheromone matrices (one for each optimization criteria). A heuristic function for each objective is used as well.

- Fixed solution generation, using predefined parameters to combine heuristic and pheromone values for each component.
- Solution evaluation based on objective values, with dominance ranking and crowding distance as a secondary criteria.
- Global update of pheromone matrices.
- Offline Pareto archival, with *hall of fame* with the t best solutions (ordered by dominance rank) found.

The pseudo-code below is a high-level representation of the algorithm works. Each step is described further.

```
initializePheromoneMatrix()
while(not_termination)
   generateSolutions()
   evaluateAndArchiveSolutions()
   daemonActions()
   updatePheronomones()
end while
```

Pheromone Matrices Initialization All pheromone matrices are initialized with predefined value τ_0 for every component.

Stop Condition Two criteria are used as stop conditions, and any of them is sufficient to trigger a stop:

1. Maximum time elapsed;
2. Number of iterations without progress. The number of new solutions added to the offline *hall of fame* set is used as progress metric between iterations.

Solution Generation At each iteration, n solutions are obtained, where n is the parameter that determines the number of ants used. As every ant is initialized with the same parameters, this value can be seen as the batch size for the method.

Each solution is built in an iterative way, starting from an empty solution $s_p = \emptyset$. At each step, a component $c_i \in N(s_p)$ is added to s_p, where $N(s_p) \subseteq \boldsymbol{T} \cup \boldsymbol{A}$ is the set of feasible components. The feasible components set includes all components that fulfill every restriction from set $\boldsymbol{R}$, defined in (3.1.4), considering the partial solution s_p.

The added component c_i is chosen randomly, with probabilities for each component defined by Equation (3.12), as presented in (2.5.3). In this equation,

$$p(c_i|s_p) = \frac{\prod_{l=1}^{L}(\tau_i^l)^{\alpha_l} \cdot \prod_{m=1}^{M}(\eta_i^m)^{\beta_m}}{\sum_{c_j \in N(s_p)}(\prod_{l=1}^{L}(\tau_j^l)^{\alpha_l} \cdot \prod_{m=1}^{M}(\eta_j^m)^{\beta_m})}, \quad \forall c_i \in N(s_p), \tag{3.12}$$

Solution Evaluation and Archival After each iteration, all new solutions are evaluated and compared to the set of best solutions found so far (the *hall of fame*). All of them are sorted by dominance ranking, with crowding distance as a secondary ordering criterion. The *hall of fame* is then updated with the t best solutions and the others are discarded.

Pheromone Matrices Update Every value on the pheromone matrices is updated based on the evaporation rate, which decreases the pheromone value for it. Then, for all components used in a solution, their pheromone value in each matrix is updated based on the solution's quality for the respective criterion. A parameter γ was added to the equation presented in 2.5.3 to allow further control over the pheromone deposit rate. The pheromone update follows:

$$\tau_i^l \leftarrow (1-\rho)\tau_i^l + \gamma \sum_{s \in S_{upd} | c_j^i \in s} g^l(s) \tag{3.13}$$

where l is the pheromone matrix index, S_{upd} is the set of solutions generated in current iteration, $\rho \in (0, 1]$ the evaporation rate parameter, γ the pheromone deposit rate parameter and $g^l(s)$ the quality function that evaluates solution s considering criterion l.

The quality values are normalized between zero and one based on a referential *nadir* solution, a worst-case solution calculated for each problem. This keeps every criteria value within the same range and ensures that their effect is the same.

3.3.2 Local-search MOACO

Local search methods are commonly used as one step of a two-phase Pareto Search, as seen in Section 2.5.6, including applications for multiobjective TSP problems [39]. The

ACO heuristic can be easily combined with other optimization methods [5]. This section presents a hybrid ACO method with Local Search.

The proposed method keeps the same framework outlined in Section 3.3.1. The Local-search addition is included as a *daemon action*, which takes place after each iteration of the solution generation step. Each solution generated is subjected to a Pareto local search algorithm. The search results are then evaluated and, if a new solution that dominates the original one is found, the latter is replaced.

A 1-opt neighborhood structure was selected for the Local Search. Given a solution s, the set of neighboring solutions $N(s)$ contains every feasible solution that can be obtained from s changing only one component by another component. That is:

$$s' \in N(s) \iff (|s' \cap s| = |s| - 1) \wedge (s' \in \boldsymbol{S})$$

The Local Search method uses a first improvement strategy. After each neighbor is generated, it is compared to the current solution. If the new solution dominates the previous one, it is selected. This strategy speeds up the execution, although the upper limit is unchanged. The neighbor selection algorithm framework is:

```
s = s0
foreach c in s:
    alternativeComponents = getFeasibleComponents(s, c)
    foreach c' in alternativeComponents:
        s' = getAlternativeSolution(s, c, c')
        if(isFeasible(s') and s.isDominated(s')):
            return s'
```

The neighbor selection method has a $\mathcal{O}(2k(f-1))$ complexity, where k is the number of destinations in the trip and f is the maximum number of feasible components at each step.

This method is necessary to guarantee that only feasible solutions are generated, which keeps the size of the search space under control, as discussed in Section 3.1.5. The non-trivial cost of this step is one of the reasons a smaller 1-opt neighborhood was chosen.

The general Local Search algorithm is:

```
s = s0
while(True):
    s' = getImprovedNeighbor(s)
```

```
if (s'== None):
    return s
else:
    s = s'
```

The hybrid meta-heuristic increases the time required for each iteration, due to the increased computations required for each solution. However, due to the additional exploitation from the Local Search, the convergence of the algorithm improves.

3.3.3 Distributed MOACO

The Ant Colony framework is inherently very adequate to distributed implementations [20]. In each iteration, the solution generation step starts a loop in which each ant agents builds a solution. There are no interaction between the processes of each ant until the pheromone update step, in which all the solutions built are used to update the pheromone matrices. Thus, the solution generation step is a natural candidate for parallelism.

In order to implement a distributed MOACO method, each ant process should run as a independent thread. After each solution is built, they are added to a set. The algorithm stays locked until all solutions are generated. After that, the daemon actions and pheromone update steps are performed without any changes.

This method is also compatible with the hybrid Local-search MOACO meta-heuristic proposed. The only adaptation necessary is to perform the local search for each solution in its own thread, right after the generation step. This way, the **Distributed Local-search MOACO** method is obtained as a combination from the two methods implemented.

This distributed implementation aims to make better use of hardware resources and speed up computational times for the MOACO meta-heuristic. It should have no other impact other than performance related ones, since there are no changes to the general framework.

3.3.4 MOEA/D-ACO

The MOEA/D-ACO meta-heuristic, described in Section 2.5.5, aggregates strengths of both MOEA/D and MOACO methods. The algorithm framework is very similar to traditional MOACO methods. The main differences are on the solution construction

step and on the pheromone matrices organization and update. The general structure is presented in the pseudo-code 3.3.4. Each of the steps is then detailed, and the differences from traditional MOACO highlighted.

```
initialization()
while(not_termination)
    generateSolutions()
    evaluateAndArchiveSolutions()
    daemonActions()
    updatePheronomonesAndSolutions()
end while
```

Initialization Generate a set of weight vectors $\boldsymbol{\lambda_i} = (\lambda_i^1, ..., \lambda_i^m) \quad \forall i \in (1, ..., n)$, uniformly distributed over the objective space. The weights follow the Weighted Sum Approach such that $\sum_{m=1}^{M} \lambda_i^m = 1$. A single optimization scalar sub-problem $g_i(x|\boldsymbol{\lambda_i} = \sum_{m=1}^{M} \lambda_i^m f_m(x))$ is created for each weight vector. Each vector and sub-problem is attributed to an ant i. Each ant also has an heuristic matrix based on its sub-problem i.

Following this step, all n ants are split into K groups. The groups are found by clustering the weight vectors according to their Euclidean Distance in relation to one another. A pheromone matrix is initialized with values τ_0 and attributed to each group.

A neighborhood structure is also defined. For each ant i, its neighbors $N(i)$ are the T ants with closest weight vectors. Neighbor ants are not always on the same group.

Solution Construction Each ant i builds a solution step-by-step. As in the traditional MOACO framework, at each step a component is chosen for the partial solution s_p among the set of feasible ones $F(s_p)$, until it is complete. Each component has as probability value defined by Equation 3.14:

$$p(c_j|s_p) = \frac{[\tau_j^i + \Delta.In(s_i, c_j)]^\alpha . {\eta_j^i}^\beta}{\sum_{c_k \in N(s_p)} [\tau_k^i + \Delta.In(s_i, c_j)]^\alpha . {\eta_k^i}^\beta}, \quad \forall c_j \in F(s_p), \tag{3.14}$$

where τ_j^i is the pheromone value of component c_j in ant's i pheromone matrix, η_j^i is the heuristic value for component c_j for ant i and α and β are pheromone and heuristic influence modifiers. Δ is the current solution influence modifier, which increases the

probability for components that integrate ant's i current solution s_i. Finally, $In(s_i, c_j)$ is an indicator function defined as:

$$In(s_i, c_j) = \begin{cases} 1, & \text{if } c_j \in s_i \\ 0, & \text{otherwise} \end{cases}$$

At each step, there is a r probability that the component with maximum probabilistic value is directly selected. If not, a roulette wheel selection is performed to select the next component based on their probabilities. For all previous MOACO implementation, $r = 0$ and is omitted.

Evaluation and Archival No changes in this step. An external Pareto Set EP is kept and updated with all new solutions generated.

Pheromone Update Each group's pheromone matrix is updated following equation 3.13. Only solutions generated by the group's ants that were added to EP are used to update the pheromone matrices. This is a change from the MOACO implementation defined in Section 3.3.1.

The pheromones are limited between maximum and minimum values τ_{max} and τ_{min}, respectively. These values are defined as:

$$\tau_{max} = \frac{B + 1}{\rho . g_{min}(s_i)}$$

where B is the number of non-dominated solutions generated in the current iteration, ρ is the pheromone evaporation rate and g_{min} is the minimum value for all n sub-problems g_i. $\tau_{min} = \epsilon . \tau_{max}$, where $0 < \epsilon < 1$ is a method's hyperparameter.

These pheromone limitations is also a new addition not present in the MOACO method.

It is interesting to note that the MOEA/D-ACO method proposed is entirely compatible with the additions of both the Local-search MOACO method and the Distributed MOACO. In the first case, the Local-search step is included as a *daemon-action*. In the second, the solution construction step is also distributed to independent threads that later

converge. Thus, three new methods can be obtained just from the combinations of the proposed ones: **Local-search MOEA/D-ACO**, **Distributed MOEA/D-ACO** and even **Distributed Local-search MOEA/D-ACO**.

3.3.5 Method Choice Considerations

As an application of the Traveling Salesman Problem with Time Windows with Constraints, the trip itinerary optimization problem proposed is, by consequence, a combinatorial NP-complete problem. The search space estimation presented in (3.1.5) indicates that the number of possible solutions is big and that unfeasible solutions represent the major part of the complete search space. Thus, it is desirable to use a method that can be restricted to the feasible search space. This way, it could avoid wasting efforts with unfeasible solutions.

Different problem representations (and restrictions) can lead to different search space sizes. These representations are influenced by the optimization method chosen. Genetic algorithms, for example, need some kind of representation in which combinations of solutions can be made to generate new solutions. In some representations, these recombinations might result on unfeasible solutions. The graph representation suggested in (3.1.5) can restrict the problem to the feasible region of the search space, which can be a big advantage for this problem.

Ant Colony Optimization methods are specially adequate to complex combinatorial problems such as the one presented, and shows interesting results for dynamic multi-objective problems [30]. The graph representation also is very fitting to the solution generation process. In [33], an ACO method was successfully used to solve a similar problem.

The main advantage of this method in this for the trip planning problem is that it can easily be restricted to the feasible search spaces and avoid invalid solutions. Due to its iterative solution generation process, it is possible to check, after each step, which solutions components can be added without making the solution unfeasible. In other methods, such as genetic algorithms, it can be harder to guarantee that new solutions created are feasible.

Besides, the pheromone matrices in ACO methods are a good representation of each solution component contribution to each objective and highlights good components.

ACO algorithms also make it easy to use problem related information through heuristic functions.

Decomposition based methods, mainly represented by MOEA/D, have also been applied with great results for many real-world multiobjective problems [53]. These methods are able to explore well different regions of Pareto Fronts, which improves solution diversity. The optimization of simplified scalar sub-problems also leads to efficient solutions.

Local search algorithms are efficient methods that can always lead to good solutions in a region. Their main disadvantage is the inability to explore the complete search space on its own and escape local optima.

The MOEA/D-ACO combines the strength of both approaches really well. This combined approach keeps the structural advantage of the ACO framework and restrains the search to the feasible regions of the search space. Simultaneously, the decomposition approach favors efficiency and diversity.

The use of local search methods favors exploitation and helps improving a population of solutions found. Two-phase approaches help finding a good initial population and exploring more regions of the search space.

Finally, these methods are appropriate to parallel implementations, which can make better use of hardware resources and improve computational times.

3.3.6 Hyperparameter Tuning

As discussed in Section 2.6, meta-heuristics have many hyper parameters, and tuning them can have significant impact in the results. The methods proposed are no exception: the number of hyper parameters is more than 12 for all methods considered, most of which are numeric. These parameters can have great impact in their performance.

A complete hyperparameter selection for so many parameters is absolutely unfeasible. Thus, some of the most impactful were selected to be optimized, while the other parameters were given fixed values based on the literature and empirical experiments. The selected parameters were:

- Number of Ants
- Pheromone Influence (α)

- Heuristic Influence (β)
- Deposit Rate Modifier (γ)
- Evaporation Rate (ρ)

They were tuned using a fractional factorial experiment design, with predetermined levels (between 6 and 2 levels for each). They were all tuned with the base MOACO algorithm and replicated in the other versions. Ideally, they would have to be tuned again for each algorithm variation, but it was not possible due to time constraints.

3.3.7 Robustness evaluation

In many optimization problems, there might be some variation between the planned solution and the executed one, either due to inherent variability in the process or due to unpredictable external factors. In order to prepare for this, it is desirable to estimate how much the quality of a solution changes when it is subjected to perturbations in its variables. The robustness of a solution is a metric that tries to account for this variation. In many problems, it is very important to account for robustness, and not just the ideal (nominal) objective values of the solutions .

In the trip planning itinerary problem, there are two main possible causes of variations between a planned solution and its execution:

1. Price variations between the moment information is gathered and the moment the tickets are bought and reservations are made
2. Flights being sold out or accommodations booked out, which could require changes in the solution planned.

While the first cause is more common, its effects are smaller and probably do not cause too much variation. The second one, on the other hand, is rare, but may have a much greater impact. Thus, it is desirable to measure the possible impact of this problem in the solutions obtained. This information might be useful to help a user decide between solutions after the optimization is finished.

In order to estimate the possible variation of a solution if any of their components becomes unavailable, the following routine is applied to each solution s_i:

- For every component $c_j \in s_i$, $\boldsymbol{C}_j$ is defined as the set of possible components that might replace c_j without making solution s_i unfeasible.

- The best replacement in the set $c'_j \in \boldsymbol{C}_j$ is selected based on the Euclidean Distance between each component in $\boldsymbol{C}_j$ and c_j in the objective space. The closest component is selected.
- Each solution s_{ij} represents the variation of s_i in which component c'_j replaces c_j. Set $\boldsymbol{S'}$ is the set of all s_{ij} $\forall j \in \{1, ..., |s_i|\}$
- The robustness value r_i^m for each objective m in solution s_i is defined as the ratio between the worst objective value for objective m for all $s_{ij} \in \boldsymbol{S'}$ and the objective value of the original solution.
- The aggregated robustness values for solution s_i is calculated by the product of all robustness values for all objectives. That is: $r_i = \prod_{m=1}^{M} r_i^m$.

The robustness values for each solution s_i represents the worst case scenario for each objective in the situation where one of its component becomes unavailable and is replaced by the second-best feasible replacement.

3.4 Decision Support

Once the optimization method ends, a set of non-dominated solutions is generated. In order to help the end user to chose one, a decision support method is used. As discussed in Section 2.7, Weighted Sum Model and Weighted Product Model are two simple decision support approaches, adequate for a two-objective problem.

The Weighted Sum method is chosen due to its intuitiveness to the decision maker. The solutions are presented as a list, ordered by the WS rankings. The initial weights for both objectives are equal, but the user can change them according to their preference.

In addition to the objective values (cost and travel time), other information about the solutions are presented, including detailed itinerary and robustness value. In addition to that, a graph representation of the solutions is also available, in which the axis are the objectives. This helps the user visualize the general distribution of the available itineraries.

3.5 Example

This section presents en example problem and illustrates all steps of the solution.

A traveller wants to visit Rome, Paris and London, departing from Lisbon. She wants to depart on 2018/01/14 and has 15 days available for the trip, so she would like to spend 4 or 5 days at each destination before returning. The destinations can be visited in any order. She doesn't want to travel with the air company Ryanair, and prefer hotels or Airbnb over hostels. Other than that, she has no further restrictions for the trip.

Given this information, the first step is to gather information of the flights and accommodations available for the trip. The available dates for each destination are determined based on the stay lengths and start trip date, as discussed in Section 3.2.2.

The data gathering step results in an initial data set with 845 accommodation options and 712 flights. The next step is to process this data to reduce it's size, as seen in 3.2.3. First, all options that violate the user's restrictions are removed. This results in a set of 512 accommodations and 652 flights. Then, all components strictly worse than others are also removed, leaving a set of 306 accommodations and 378 flights. This set in presented in Tables 3.1 and 3.2. With this data, the optimization process begins.

Table 3.1: Accommodation Options

	DateStart	DateEnd	Dest	Cost	Time
a0	2018-01-01	2018-01-05	Rome	428.820062	0
a1	2018-01-05	2018-01-09	Rome	395.211905	0
a2	2018-01-06	2018-01-10	Rome	405.564270	0
a3	2018-01-07	2018-01-11	Rome	402.640626	0
a4	2018-01-08	2018-01-12	Rome	378.847405	0
a5	2018-01-09	2018-01-13	Rome	385.325028	0
a6	2018-01-10	2018-01-14	Rome	420.050979	0
a7	2018-01-11	2018-01-15	Rome	424.633099	0
a8	2018-01-12	2018-01-16	Rome	446.037460	0
a9	2018-01-01	2018-01-06	Rome	440.864877	0
a10	2018-01-05	2018-01-10	Rome	417.609086	0
a11	2018-01-06	2018-01-11	Rome	422.366528	0
a12	2018-01-07	2018-01-12	Rome	416.822306	0
a13	2018-01-08	2018-01-13	Rome	427.791315	0
a14	2018-01-09	2018-01-14	Rome	442.448159	0
a15	2018-01-10	2018-01-15	Rome	441.435357	0
a16	2018-01-11	2018-01-16	Rome	460.219139	0
a17	2018-01-01	2018-01-05	Rome	485.003120	0
a18	2018-01-05	2018-01-09	Rome	416.150267	0
a19	2018-01-06	2018-01-10	Rome	401.646730	0
a20	2018-01-07	2018-01-11	Rome	369.462666	0
a21	2018-01-08	2018-01-12	Rome	342.863744	0
a22	2018-01-09	2018-01-13	Rome	344.800057	0
a23	2018-01-10	2018-01-14	Rome	357.461813	0
a24	2018-01-11	2018-01-15	Rome	355.225883	0
a25	2018-01-12	2018-01-16	Rome	363.365539	0
a26	2018-01-01	2018-01-06	Rome	524.439101	0
a27	2018-01-05	2018-01-10	Rome	441.082711	0
a28	2018-01-06	2018-01-11	Rome	415.951277	0
a29	2018-01-07	2018-01-12	Rome	381.092698	0

Continued on next page

Table 3.1 – continued from previous page

	DateStart	DateEnd	Dest	Cost	Time
a30	2018-01-08	2018-01-13	Rome	353.796778	0
a31	2018-01-09	2018-01-14	Rome	382.394257	0
a32	2018-01-10	2018-01-15	Rome	369.530431	0
a33	2018-01-11	2018-01-16	Rome	374.995571	0
a34	2018-01-01	2018-01-05	Rome	443.705248	0
a35	2018-01-05	2018-01-09	Rome	422.383313	0
a36	2018-01-06	2018-01-10	Rome	444.796457	0
a37	2018-01-07	2018-01-11	Rome	455.138534	0
a38	2018-01-08	2018-01-12	Rome	440.046463	0
a39	2018-01-09	2018-01-13	Rome	449.034866	0
a40	2018-01-10	2018-01-14	Rome	457.912403	0
a41	2018-01-11	2018-01-15	Rome	440.315301	0
a42	2018-01-12	2018-01-16	Rome	433.327317	0
a43	2018-01-01	2018-01-06	Rome	457.066931	0
a44	2018-01-05	2018-01-10	Rome	458.158140	0
a45	2018-01-06	2018-01-11	Rome	484.924891	0
a46	2018-01-07	2018-01-12	Rome	495.436113	0
a47	2018-01-08	2018-01-13	Rome	489.880489	0
a48	2018-01-09	2018-01-14	Rome	493.687230	0
a49	2018-01-10	2018-01-15	Rome	480.443735	0
a50	2018-01-11	2018-01-16	Rome	473.624896	0
a51	2018-01-01	2018-01-05	Rome	401.620929	0
a52	2018-01-05	2018-01-09	Rome	456.805231	0
a53	2018-01-06	2018-01-10	Rome	430.232114	0
a54	2018-01-07	2018-01-11	Rome	413.684290	0
a55	2018-01-08	2018-01-12	Rome	385.138411	0
a56	2018-01-09	2018-01-13	Rome	354.969892	0
a57	2018-01-10	2018-01-14	Rome	333.997059	0
a58	2018-01-11	2018-01-15	Rome	342.449513	0
a59	2018-01-12	2018-01-16	Rome	340.126385	0
a60	2018-01-01	2018-01-06	Rome	457.058931	0
a61	2018-01-05	2018-01-10	Rome	485.670116	0
a62	2018-01-06	2018-01-11	Rome	449.366553	0
a63	2018-01-07	2018-01-12	Rome	426.774483	0
a64	2018-01-08	2018-01-13	Rome	396.018786	0
a65	2018-01-09	2018-01-14	Rome	362.861944	0
a66	2018-01-10	2018-01-15	Rome	361.583952	0
a67	2018-01-11	2018-01-16	Rome	353.216578	0
a68	2018-01-01	2018-01-05	Rome	395.449280	0
a69	2018-01-05	2018-01-09	Rome	364.650151	0
a70	2018-01-06	2018-01-10	Rome	389.146961	0
a71	2018-01-07	2018-01-11	Rome	412.645860	0
a72	2018-01-08	2018-01-12	Rome	404.090795	0
a73	2018-01-09	2018-01-13	Rome	444.316763	0
a74	2018-01-10	2018-01-14	Rome	407.268119	0
a75	2018-01-11	2018-01-15	Rome	386.150909	0
a76	2018-01-12	2018-01-16	Rome	406.070958	0
a77	2018-01-01	2018-01-06	Rome	428.685376	0
a78	2018-01-05	2018-01-10	Rome	422.383057	0
a79	2018-01-06	2018-01-11	Rome	419.466833	0
a80	2018-01-07	2018-01-12	Rome	428.802551	0
a81	2018-01-08	2018-01-13	Rome	461.198089	0
a82	2018-01-09	2018-01-14	Rome	465.001025	0
a83	2018-01-10	2018-01-15	Rome	416.470782	0
a84	2018-01-11	2018-01-16	Rome	422.227649	0
a85	2018-01-01	2018-01-05	Rome	452.729322	0
a86	2018-01-05	2018-01-09	Rome	390.260236	0

Continued on next page

Table 3.1 – continued from previous page

	DateStart	DateEnd	Dest	Cost	Time
a87	2018-01-06	2018-01-10	Rome	373.845008	0
a88	2018-01-07	2018-01-11	Rome	354.248112	0
a89	2018-01-08	2018-01-12	Rome	350.661388	0
a90	2018-01-09	2018-01-13	Rome	368.500019	0
a91	2018-01-10	2018-01-14	Rome	369.453150	0
a92	2018-01-11	2018-01-15	Rome	392.979646	0
a93	2018-01-12	2018-01-16	Rome	422.414696	0
a94	2018-01-01	2018-01-06	Rome	480.337164	0
a95	2018-01-05	2018-01-10	Rome	401.452850	0
a96	2018-01-06	2018-01-11	Rome	392.513902	0
a97	2018-01-07	2018-01-12	Rome	379.443858	0
a98	2018-01-08	2018-01-13	Rome	381.104154	0
a99	2018-01-09	2018-01-14	Rome	380.645764	0
a100	2018-01-10	2018-01-15	Rome	411.648539	0
a101	2018-01-11	2018-01-16	Rome	447.610441	0
a102	2018-01-01	2018-01-05	Paris	437.469652	0
a103	2018-01-05	2018-01-09	Paris	391.032583	0
a104	2018-01-06	2018-01-10	Paris	397.872154	0
a105	2018-01-07	2018-01-11	Paris	433.150089	0
a106	2018-01-08	2018-01-12	Paris	436.823219	0
a107	2018-01-09	2018-01-13	Paris	432.821849	0
a108	2018-01-10	2018-01-14	Paris	466.650771	0
a109	2018-01-11	2018-01-15	Paris	432.907155	0
a110	2018-01-12	2018-01-16	Paris	423.118272	0
a111	2018-01-01	2018-01-06	Paris	444.311807	0
a112	2018-01-05	2018-01-10	Paris	404.714309	0
a113	2018-01-06	2018-01-11	Paris	446.098041	0
a114	2018-01-07	2018-01-12	Paris	476.898878	0
a115	2018-01-08	2018-01-13	Paris	480.988665	0
a116	2018-01-09	2018-01-14	Paris	480.332497	0
a117	2018-01-10	2018-01-15	Paris	481.133042	0
a118	2018-01-11	2018-01-16	Paris	466.867062	0
a119	2018-01-01	2018-01-05	Paris	454.033102	0
a120	2018-01-05	2018-01-09	Paris	397.449169	0
a121	2018-01-06	2018-01-10	Paris	427.198234	0
a122	2018-01-07	2018-01-11	Paris	441.418651	0
a123	2018-01-08	2018-01-12	Paris	464.258190	0
a124	2018-01-09	2018-01-13	Paris	436.309782	0
a125	2018-01-10	2018-01-14	Paris	402.483383	0
a126	2018-01-11	2018-01-15	Paris	405.233684	0
a127	2018-01-12	2018-01-16	Paris	390.716385	0
a128	2018-01-01	2018-01-06	Paris	475.702846	0
a129	2018-01-05	2018-01-10	Paris	448.867979	0
a130	2018-01-06	2018-01-11	Paris	462.915510	0
a131	2018-01-07	2018-01-12	Paris	492.745899	0
a132	2018-01-08	2018-01-13	Paris	479.104639	0
a133	2018-01-09	2018-01-14	Paris	453.902193	0
a134	2018-01-10	2018-01-15	Paris	440.950960	0
a135	2018-01-11	2018-01-16	Paris	442.043633	0
a136	2018-01-01	2018-01-05	Paris	439.346200	0
a137	2018-01-05	2018-01-09	Paris	423.270375	0
a138	2018-01-06	2018-01-10	Paris	397.137642	0
a139	2018-01-07	2018-01-11	Paris	388.590134	0
a140	2018-01-08	2018-01-12	Paris	392.737032	0
a141	2018-01-09	2018-01-13	Paris	414.833349	0
a142	2018-01-10	2018-01-14	Paris	424.435087	0
a143	2018-01-11	2018-01-15	Paris	433.364003	0

Continued on next page

Table 3.1 – continued from previous page

	DateStart	DateEnd	Dest	Cost	Time
a144	2018-01-12	2018-01-16	Paris	420.478311	0
a145	2018-01-01	2018-01-06	Paris	479.845621	0
a146	2018-01-05	2018-01-10	Paris	437.637063	0
a147	2018-01-06	2018-01-11	Paris	424.352787	0
a148	2018-01-07	2018-01-12	Paris	445.648177	0
a149	2018-01-08	2018-01-13	Paris	425.930505	0
a150	2018-01-09	2018-01-14	Paris	438.801774	0
a151	2018-01-10	2018-01-15	Paris	460.579148	0
a152	2018-01-11	2018-01-16	Paris	477.536353	0
a153	2018-01-01	2018-01-05	Paris	445.372333	0
a154	2018-01-05	2018-01-09	Paris	420.176696	0
a155	2018-01-06	2018-01-10	Paris	437.267443	0
a156	2018-01-07	2018-01-11	Paris	409.873171	0
a157	2018-01-08	2018-01-12	Paris	435.510265	0
a158	2018-01-09	2018-01-13	Paris	390.082346	0
a159	2018-01-10	2018-01-14	Paris	406.492564	0
a160	2018-01-11	2018-01-15	Paris	415.352196	0
a161	2018-01-12	2018-01-16	Paris	386.794713	0
a162	2018-01-01	2018-01-06	Paris	466.438370	0
a163	2018-01-05	2018-01-10	Paris	458.333480	0
a164	2018-01-06	2018-01-11	Paris	446.733674	0
a165	2018-01-07	2018-01-12	Paris	460.644927	0
a166	2018-01-08	2018-01-13	Paris	444.197839	0
a167	2018-01-09	2018-01-14	Paris	444.649348	0
a168	2018-01-10	2018-01-15	Paris	424.818427	0
a169	2018-01-11	2018-01-16	Paris	437.566468	0
a170	2018-01-01	2018-01-05	Paris	442.709208	0
a171	2018-01-05	2018-01-09	Paris	429.337007	0
a172	2018-01-06	2018-01-10	Paris	403.387275	0
a173	2018-01-07	2018-01-11	Paris	409.851324	0
a174	2018-01-08	2018-01-12	Paris	387.579568	0
a175	2018-01-09	2018-01-13	Paris	360.577065	0
a176	2018-01-10	2018-01-14	Paris	391.063524	0
a177	2018-01-11	2018-01-15	Paris	403.496890	0
a178	2018-01-12	2018-01-16	Paris	423.792003	0
a179	2018-01-01	2018-01-06	Paris	480.994947	0
a180	2018-01-05	2018-01-10	Paris	441.673014	0
a181	2018-01-06	2018-01-11	Paris	445.842933	0
a182	2018-01-07	2018-01-12	Paris	416.300758	0
a183	2018-01-08	2018-01-13	Paris	403.915533	0
a184	2018-01-09	2018-01-14	Paris	403.399532	0
a185	2018-01-10	2018-01-15	Paris	445.952549	0
a186	2018-01-11	2018-01-16	Paris	430.241437	0
a187	2018-01-01	2018-01-05	Paris	420.209189	0
a188	2018-01-05	2018-01-09	Paris	462.433090	0
a189	2018-01-06	2018-01-10	Paris	484.446958	0
a190	2018-01-07	2018-01-11	Paris	473.963099	0
a191	2018-01-08	2018-01-12	Paris	446.031159	0
a192	2018-01-09	2018-01-13	Paris	394.555218	0
a193	2018-01-10	2018-01-14	Paris	351.698946	0
a194	2018-01-11	2018-01-15	Paris	360.122619	0
a195	2018-01-12	2018-01-16	Paris	352.239209	0
a196	2018-01-01	2018-01-06	Paris	448.557763	0
a197	2018-01-05	2018-01-10	Paris	512.795532	0
a198	2018-01-06	2018-01-11	Paris	514.096390	0
a199	2018-01-07	2018-01-12	Paris	498.699543	0
a200	2018-01-08	2018-01-13	Paris	452.838060	0

Continued on next page

Table 3.1 – continued from previous page

	DateStart	DateEnd	Dest	Cost	Time
a201	2018-01-09	2018-01-14	Paris	402.061388	0
a202	2018-01-10	2018-01-15	Paris	389.772051	0
a203	2018-01-11	2018-01-16	Paris	376.975653	0
a204	2018-01-01	2018-01-05	London	390.802191	0
a205	2018-01-05	2018-01-09	London	442.026904	0
a206	2018-01-06	2018-01-10	London	432.467877	0
a207	2018-01-07	2018-01-11	London	472.084183	0
a208	2018-01-08	2018-01-12	London	461.900628	0
a209	2018-01-09	2018-01-13	London	483.538231	0
a210	2018-01-10	2018-01-14	London	492.782578	0
a211	2018-01-11	2018-01-15	London	474.716332	0
a212	2018-01-12	2018-01-16	London	465.657199	0
a213	2018-01-01	2018-01-06	London	446.382055	0
a214	2018-01-05	2018-01-10	London	488.047741	0
a215	2018-01-06	2018-01-11	London	488.812577	0
a216	2018-01-07	2018-01-12	London	513.095200	0
a217	2018-01-08	2018-01-13	London	519.062304	0
a218	2018-01-09	2018-01-14	London	538.803415	0
a219	2018-01-10	2018-01-15	London	531.061032	0
a220	2018-01-11	2018-01-16	London	506.668216	0
a221	2018-01-01	2018-01-05	London	383.297623	0
a222	2018-01-05	2018-01-09	London	382.809059	0
a223	2018-01-06	2018-01-10	London	383.744130	0
a224	2018-01-07	2018-01-11	London	376.784690	0
a225	2018-01-08	2018-01-12	London	411.840834	0
a226	2018-01-09	2018-01-13	London	448.241440	0
a227	2018-01-10	2018-01-14	London	450.895193	0
a228	2018-01-11	2018-01-15	London	436.249859	0
a229	2018-01-12	2018-01-16	London	428.600321	0
a230	2018-01-01	2018-01-06	London	427.508503	0
a231	2018-01-05	2018-01-10	London	427.955010	0
a232	2018-01-06	2018-01-11	London	404.918857	0
a233	2018-01-07	2018-01-12	London	423.846659	0
a234	2018-01-08	2018-01-13	London	463.699627	0
a235	2018-01-09	2018-01-14	London	496.041144	0
a236	2018-01-10	2018-01-15	London	457.424586	0
a237	2018-01-11	2018-01-16	London	475.662290	0
a238	2018-01-01	2018-01-05	London	417.468761	0
a239	2018-01-05	2018-01-09	London	374.091797	0
a240	2018-01-06	2018-01-10	London	408.211025	0
a241	2018-01-07	2018-01-11	London	401.209406	0
a242	2018-01-08	2018-01-12	London	388.835702	0
a243	2018-01-09	2018-01-13	London	398.986288	0
a244	2018-01-10	2018-01-14	London	384.637046	0
a245	2018-01-11	2018-01-15	London	390.224258	0
a246	2018-01-12	2018-01-16	London	394.784905	0
a247	2018-01-01	2018-01-06	London	436.637333	0
a248	2018-01-05	2018-01-10	London	427.379597	0
a249	2018-01-06	2018-01-11	London	426.691053	0
a250	2018-01-07	2018-01-12	London	421.837124	0
a251	2018-01-08	2018-01-13	London	412.426444	0
a252	2018-01-09	2018-01-14	London	437.924847	0
a253	2018-01-10	2018-01-15	London	408.704286	0
a254	2018-01-11	2018-01-16	London	415.412622	0
a255	2018-01-01	2018-01-05	London	437.655092	0
a256	2018-01-05	2018-01-09	London	374.442206	0
a257	2018-01-06	2018-01-10	London	367.621830	0

Continued on next page

Table 3.1 – continued from previous page

	DateStart	DateEnd	Dest	Cost	Time
a258	2018-01-07	2018-01-11	London	348.546266	0
a259	2018-01-08	2018-01-12	London	372.403334	0
a260	2018-01-09	2018-01-13	London	397.267096	0
a261	2018-01-10	2018-01-14	London	438.794691	0
a262	2018-01-11	2018-01-15	London	441.307087	0
a263	2018-01-12	2018-01-16	London	417.703751	0
a264	2018-01-01	2018-01-06	London	454.690239	0
a265	2018-01-05	2018-01-10	London	384.656977	0
a266	2018-01-06	2018-01-11	London	378.722508	0
a267	2018-01-07	2018-01-12	London	385.172443	0
a268	2018-01-08	2018-01-13	London	428.728804	0
a269	2018-01-09	2018-01-14	London	449.009462	0
a270	2018-01-10	2018-01-15	London	452.407764	0
a271	2018-01-11	2018-01-16	London	454.329928	0
a272	2018-01-01	2018-01-05	London	404.723571	0
a273	2018-01-05	2018-01-09	London	404.440102	0
a274	2018-01-06	2018-01-10	London	406.224976	0
a275	2018-01-07	2018-01-11	London	392.509662	0
a276	2018-01-08	2018-01-12	London	358.226274	0
a277	2018-01-09	2018-01-13	London	351.719819	0
a278	2018-01-10	2018-01-14	London	338.110935	0
a279	2018-01-11	2018-01-15	London	388.803156	0
a280	2018-01-12	2018-01-16	London	428.613544	0
a281	2018-01-01	2018-01-06	London	437.768135	0
a282	2018-01-05	2018-01-10	London	439.269540	0
a283	2018-01-06	2018-01-11	London	412.922019	0
a284	2018-01-07	2018-01-12	London	401.702230	0
a285	2018-01-08	2018-01-13	London	376.227044	0
a286	2018-01-09	2018-01-14	London	372.940374	0
a287	2018-01-10	2018-01-15	London	395.500198	0
a288	2018-01-11	2018-01-16	London	437.806112	0
a289	2018-01-01	2018-01-05	London	450.380604	0
a290	2018-01-05	2018-01-09	London	408.674267	0
a291	2018-01-06	2018-01-10	London	438.017023	0
a292	2018-01-07	2018-01-11	London	456.014234	0
a293	2018-01-08	2018-01-12	London	460.717359	0
a294	2018-01-09	2018-01-13	London	461.184001	0
a295	2018-01-10	2018-01-14	London	475.070936	0
a296	2018-01-11	2018-01-15	London	451.381007	0
a297	2018-01-12	2018-01-16	London	435.849898	0
a298	2018-01-01	2018-01-06	London	457.562468	0
a299	2018-01-05	2018-01-10	London	445.198886	0
a300	2018-01-06	2018-01-11	London	492.660059	0
a301	2018-01-07	2018-01-12	London	487.348415	0
a302	2018-01-08	2018-01-13	London	516.399524	0
a303	2018-01-09	2018-01-14	London	511.595555	0
a304	2018-01-10	2018-01-15	London	506.024043	0
a305	2018-01-11	2018-01-16	London	467.184078	0

Table 3.2: Transports Options

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t0	2014-01-01 04:22:00	2014-01-01 09:56:00	Lisbon	Rome	359.75	334
t1	2014-01-01 08:00:00	2014-01-01 13:08:00	Lisbon	Rome	334.84	308

Continued on next page

Table 3.2 – continued from previous page

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t2	2014-01-01 11:12:00	2014-01-01 16:35:00	Lisbon	Rome	546.61	323
t3	2014-01-01 13:38:00	2014-01-01 19:31:00	Lisbon	Rome	321.61	353
t4	2014-01-01 17:07:00	2014-01-01 22:49:00	Lisbon	Rome	560.78	342
t5	2014-01-01 20:14:00	2014-01-02 01:57:00	Lisbon	Rome	364.30	343
t6	2014-01-13 05:26:00	2014-01-13 09:27:00	Rome	Lisbon	462.46	241
t7	2014-01-13 07:58:00	2014-01-13 11:22:00	Rome	Lisbon	483.28	204
t8	2014-01-13 10:03:00	2014-01-13 13:14:00	Rome	Lisbon	457.26	191
t9	2014-01-13 13:38:00	2014-01-13 17:13:00	Rome	Lisbon	586.63	215
t10	2014-01-13 16:45:00	2014-01-13 21:11:00	Rome	Lisbon	454.56	266
t11	2014-01-13 19:48:00	2014-01-13 23:04:00	Rome	Lisbon	578.52	196
t12	2014-01-14 05:26:00	2014-01-14 09:47:00	Rome	Lisbon	478.48	261
t13	2014-01-14 07:58:00	2014-01-14 11:46:00	Rome	Lisbon	345.22	228
t14	2014-01-14 10:03:00	2014-01-14 13:44:00	Rome	Lisbon	525.95	221
t15	2014-01-14 13:38:00	2014-01-14 17:57:00	Rome	Lisbon	383.09	259
t16	2014-01-14 16:45:00	2014-01-14 20:26:00	Rome	Lisbon	584.94	221
t17	2014-01-14 19:48:00	2014-01-14 23:53:00	Rome	Lisbon	532.67	245
t18	2014-01-15 05:26:00	2014-01-15 09:01:00	Rome	Lisbon	374.20	215
t19	2014-01-15 07:58:00	2014-01-15 12:19:00	Rome	Lisbon	340.02	261
t20	2014-01-15 10:03:00	2014-01-15 14:05:00	Rome	Lisbon	321.06	242
t21	2014-01-15 13:38:00	2014-01-15 17:40:00	Rome	Lisbon	599.05	242
t22	2014-01-15 16:45:00	2014-01-15 20:14:00	Rome	Lisbon	360.05	209
t23	2014-01-15 19:48:00	2014-01-15 23:56:00	Rome	Lisbon	563.49	248
t24	2014-01-16 05:26:00	2014-01-16 09:23:00	Rome	Lisbon	609.48	237
t25	2014-01-16 07:58:00	2014-01-16 11:19:00	Rome	Lisbon	543.33	201
t26	2014-01-16 10:03:00	2014-01-16 13:17:00	Rome	Lisbon	472.64	194
t27	2014-01-16 13:38:00	2014-01-16 17:49:00	Rome	Lisbon	465.92	251
t28	2014-01-16 16:45:00	2014-01-16 21:11:00	Rome	Lisbon	596.11	266
t29	2014-01-16 19:48:00	2014-01-16 23:29:00	Rome	Lisbon	471.88	221
t30	2014-01-05 04:28:00	2014-01-05 10:51:00	Rome	Paris	504.20	383
t31	2014-01-05 08:12:00	2014-01-05 14:28:00	Rome	Paris	588.24	376
t32	2014-01-05 10:06:00	2014-01-05 15:50:00	Rome	Paris	462.06	344
t33	2014-01-05 14:01:00	2014-01-05 19:39:00	Rome	Paris	617.90	338
t34	2014-01-05 16:34:00	2014-01-05 22:48:00	Rome	Paris	499.40	374
t35	2014-01-05 19:57:00	2014-01-06 02:03:00	Rome	Paris	576.59	366
t36	2014-01-06 04:28:00	2014-01-06 10:08:00	Rome	Paris	604.34	340
t37	2014-01-06 08:12:00	2014-01-06 14:39:00	Rome	Paris	574.94	387
t38	2014-01-06 10:06:00	2014-01-06 15:38:00	Rome	Paris	368.23	332
t39	2014-01-06 14:01:00	2014-01-06 19:36:00	Rome	Paris	516.14	335
t40	2014-01-06 16:34:00	2014-01-06 22:40:00	Rome	Paris	443.53	366
t41	2014-01-06 19:57:00	2014-01-07 01:37:00	Rome	Paris	337.76	340
t42	2014-01-07 04:28:00	2014-01-07 10:52:00	Rome	Paris	451.60	384
t43	2014-01-07 08:12:00	2014-01-07 13:35:00	Rome	Paris	399.51	323
t44	2014-01-07 10:06:00	2014-01-07 15:40:00	Rome	Paris	585.50	334
t45	2014-01-07 14:01:00	2014-01-07 20:03:00	Rome	Paris	328.50	362
t46	2014-01-07 16:34:00	2014-01-07 22:54:00	Rome	Paris	620.14	380
t47	2014-01-07 19:57:00	2014-01-08 01:07:00	Rome	Paris	429.97	310
t48	2014-01-08 04:28:00	2014-01-08 09:49:00	Rome	Paris	430.39	321
t49	2014-01-08 08:12:00	2014-01-08 13:32:00	Rome	Paris	323.15	320
t50	2014-01-08 10:06:00	2014-01-08 15:50:00	Rome	Paris	376.36	344
t51	2014-01-08 14:01:00	2014-01-08 20:21:00	Rome	Paris	444.42	380
t52	2014-01-08 16:34:00	2014-01-08 22:06:00	Rome	Paris	610.78	332
t53	2014-01-08 19:57:00	2014-01-09 01:31:00	Rome	Paris	349.39	334
t54	2014-01-09 04:28:00	2014-01-09 10:50:00	Rome	Paris	615.83	382
t55	2014-01-09 08:12:00	2014-01-09 13:22:00	Rome	Paris	591.94	310
t56	2014-01-09 10:06:00	2014-01-09 16:02:00	Rome	Paris	461.09	356
t57	2014-01-09 14:01:00	2014-01-09 19:12:00	Rome	Paris	420.93	311
t58	2014-01-09 16:34:00	2014-01-09 22:07:00	Rome	Paris	391.33	333

Continued on next page

Table 3.2 – continued from previous page

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t59	2014-01-09 19:57:00	2014-01-10 01:26:00	Rome	Paris	511.60	329
t60	2014-01-10 04:28:00	2014-01-10 10:29:00	Rome	Paris	328.43	361
t61	2014-01-10 08:12:00	2014-01-10 14:23:00	Rome	Paris	322.92	371
t62	2014-01-10 10:06:00	2014-01-10 15:51:00	Rome	Paris	453.10	345
t63	2014-01-10 14:01:00	2014-01-10 19:27:00	Rome	Paris	339.45	326
t64	2014-01-10 16:34:00	2014-01-10 22:53:00	Rome	Paris	397.38	379
t65	2014-01-10 19:57:00	2014-01-11 01:28:00	Rome	Paris	387.68	331
t66	2014-01-11 04:28:00	2014-01-11 10:34:00	Rome	Paris	397.77	366
t67	2014-01-11 08:12:00	2014-01-11 14:01:00	Rome	Paris	359.30	349
t68	2014-01-11 10:06:00	2014-01-11 15:24:00	Rome	Paris	321.80	318
t69	2014-01-11 14:01:00	2014-01-11 19:50:00	Rome	Paris	354.39	349
t70	2014-01-11 16:34:00	2014-01-11 22:20:00	Rome	Paris	512.86	346
t71	2014-01-11 19:57:00	2014-01-12 01:28:00	Rome	Paris	624.95	331
t72	2014-01-12 04:28:00	2014-01-12 09:30:00	Rome	Paris	630.02	302
t73	2014-01-12 08:12:00	2014-01-12 13:25:00	Rome	Paris	446.88	313
t74	2014-01-12 10:06:00	2014-01-12 15:17:00	Rome	Paris	369.35	311
t75	2014-01-12 14:01:00	2014-01-12 20:04:00	Rome	Paris	519.25	363
t76	2014-01-12 16:34:00	2014-01-12 22:37:00	Rome	Paris	472.48	363
t77	2014-01-12 19:57:00	2014-01-13 01:52:00	Rome	Paris	629.72	355
t78	2014-01-05 04:02:00	2014-01-05 10:53:00	Rome	London	529.69	411
t79	2014-01-05 07:30:00	2014-01-05 14:32:00	Rome	London	459.00	422
t80	2014-01-05 10:56:00	2014-01-05 17:56:00	Rome	London	537.31	420
t81	2014-01-05 13:25:00	2014-01-05 20:50:00	Rome	London	437.20	445
t82	2014-01-05 16:18:00	2014-01-05 22:18:00	Rome	London	349.52	360
t83	2014-01-05 19:10:00	2014-01-06 02:08:00	Rome	London	620.54	418
t84	2014-01-06 04:02:00	2014-01-06 10:56:00	Rome	London	340.83	414
t85	2014-01-06 07:30:00	2014-01-06 14:22:00	Rome	London	324.09	412
t86	2014-01-06 10:56:00	2014-01-06 18:22:00	Rome	London	626.62	446
t87	2014-01-06 13:25:00	2014-01-06 19:26:00	Rome	London	364.78	361
t88	2014-01-06 16:18:00	2014-01-06 23:20:00	Rome	London	399.51	422
t89	2014-01-06 19:10:00	2014-01-07 02:23:00	Rome	London	385.91	433
t90	2014-01-07 04:02:00	2014-01-07 10:47:00	Rome	London	500.71	405
t91	2014-01-07 07:30:00	2014-01-07 14:00:00	Rome	London	625.42	390
t92	2014-01-07 10:56:00	2014-01-07 18:07:00	Rome	London	404.02	431
t93	2014-01-07 13:25:00	2014-01-07 19:33:00	Rome	London	623.63	368
t94	2014-01-07 16:18:00	2014-01-07 22:57:00	Rome	London	326.04	399
t95	2014-01-07 19:10:00	2014-01-08 01:56:00	Rome	London	597.79	406
t96	2014-01-08 04:02:00	2014-01-08 11:04:00	Rome	London	541.26	422
t97	2014-01-08 07:30:00	2014-01-08 13:38:00	Rome	London	405.52	368
t98	2014-01-08 10:56:00	2014-01-08 17:16:00	Rome	London	599.94	380
t99	2014-01-08 13:25:00	2014-01-08 20:01:00	Rome	London	326.36	396
t100	2014-01-08 16:18:00	2014-01-08 23:14:00	Rome	London	515.12	416
t101	2014-01-08 19:10:00	2014-01-09 02:29:00	Rome	London	421.83	439
t102	2014-01-09 04:02:00	2014-01-09 10:57:00	Rome	London	622.70	415
t103	2014-01-09 07:30:00	2014-01-09 13:42:00	Rome	London	371.12	372
t104	2014-01-09 10:56:00	2014-01-09 18:24:00	Rome	London	602.02	448
t105	2014-01-09 13:25:00	2014-01-09 20:43:00	Rome	London	421.95	438
t106	2014-01-09 16:18:00	2014-01-09 23:03:00	Rome	London	598.16	405
t107	2014-01-09 19:10:00	2014-01-10 02:33:00	Rome	London	479.46	443
t108	2014-01-10 04:02:00	2014-01-10 10:50:00	Rome	London	478.88	408
t109	2014-01-10 07:30:00	2014-01-10 14:53:00	Rome	London	439.88	443
t110	2014-01-10 10:56:00	2014-01-10 18:10:00	Rome	London	485.01	434
t111	2014-01-10 13:25:00	2014-01-10 20:52:00	Rome	London	615.64	447
t112	2014-01-10 16:18:00	2014-01-10 23:40:00	Rome	London	599.31	442
t113	2014-01-10 19:10:00	2014-01-11 01:13:00	Rome	London	531.33	363
t114	2014-01-11 04:02:00	2014-01-11 10:17:00	Rome	London	519.33	375
t115	2014-01-11 07:30:00	2014-01-11 14:05:00	Rome	London	490.77	395

Continued on next page

Table 3.2 – continued from previous page

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t116	2014-01-11 10:56:00	2014-01-11 18:21:00	Rome	London	403.92	445
t117	2014-01-11 13:25:00	2014-01-11 19:52:00	Rome	London	364.72	387
t118	2014-01-11 16:18:00	2014-01-11 22:32:00	Rome	London	587.79	374
t119	2014-01-11 19:10:00	2014-01-12 02:29:00	Rome	London	518.08	439
t120	2014-01-12 04:02:00	2014-01-12 10:42:00	Rome	London	412.41	400
t121	2014-01-12 07:30:00	2014-01-12 14:51:00	Rome	London	463.00	441
t122	2014-01-12 10:56:00	2014-01-12 17:10:00	Rome	London	325.73	374
t123	2014-01-12 13:25:00	2014-01-12 20:24:00	Rome	London	493.83	419
t124	2014-01-12 16:18:00	2014-01-12 22:57:00	Rome	London	431.79	399
t125	2014-01-12 19:10:00	2014-01-13 01:16:00	Rome	London	476.39	366
t126	2014-01-01 04:39:00	2014-01-01 10:32:00	Lisbon	Paris	537.43	353
t127	2014-01-01 07:16:00	2014-01-01 13:14:00	Lisbon	Paris	395.95	358
t128	2014-01-01 11:04:00	2014-01-01 16:04:00	Lisbon	Paris	330.49	300
t129	2014-01-01 13:47:00	2014-01-01 20:14:00	Lisbon	Paris	336.89	387
t130	2014-01-01 16:28:00	2014-01-01 22:27:00	Lisbon	Paris	337.25	359
t131	2014-01-01 19:00:00	2014-01-02 00:26:00	Lisbon	Paris	603.99	326
t132	2014-01-13 05:02:00	2014-01-13 11:48:00	Paris	Lisbon	381.13	406
t133	2014-01-13 07:30:00	2014-01-13 14:39:00	Paris	Lisbon	609.65	429
t134	2014-01-13 10:13:00	2014-01-13 17:39:00	Paris	Lisbon	415.44	446
t135	2014-01-13 13:05:00	2014-01-13 19:15:00	Paris	Lisbon	445.04	370
t136	2014-01-13 16:21:00	2014-01-13 22:30:00	Paris	Lisbon	517.45	369
t137	2014-01-13 19:38:00	2014-01-14 02:31:00	Paris	Lisbon	598.61	413
t138	2014-01-14 05:02:00	2014-01-14 12:09:00	Paris	Lisbon	454.30	427
t139	2014-01-14 07:30:00	2014-01-14 14:46:00	Paris	Lisbon	610.45	436
t140	2014-01-14 10:13:00	2014-01-14 17:37:00	Paris	Lisbon	504.54	444
t141	2014-01-14 13:05:00	2014-01-14 20:33:00	Paris	Lisbon	454.04	448
t142	2014-01-14 16:21:00	2014-01-14 22:56:00	Paris	Lisbon	504.76	395
t143	2014-01-14 19:38:00	2014-01-15 02:12:00	Paris	Lisbon	429.69	394
t144	2014-01-15 05:02:00	2014-01-15 11:15:00	Paris	Lisbon	525.00	373
t145	2014-01-15 07:30:00	2014-01-15 14:31:00	Paris	Lisbon	523.52	421
t146	2014-01-15 10:13:00	2014-01-15 17:12:00	Paris	Lisbon	576.89	419
t147	2014-01-15 13:05:00	2014-01-15 20:22:00	Paris	Lisbon	444.90	437
t148	2014-01-15 16:21:00	2014-01-15 22:29:00	Paris	Lisbon	329.84	368
t149	2014-01-15 19:38:00	2014-01-16 02:22:00	Paris	Lisbon	628.22	404
t150	2014-01-16 05:02:00	2014-01-16 11:54:00	Paris	Lisbon	459.20	412
t151	2014-01-16 07:30:00	2014-01-16 13:51:00	Paris	Lisbon	613.41	381
t152	2014-01-16 10:13:00	2014-01-16 16:28:00	Paris	Lisbon	529.74	375
t153	2014-01-16 13:05:00	2014-01-16 20:22:00	Paris	Lisbon	402.03	437
t154	2014-01-16 16:21:00	2014-01-16 22:26:00	Paris	Lisbon	495.72	365
t155	2014-01-16 19:38:00	2014-01-17 02:20:00	Paris	Lisbon	619.49	402
t156	2014-01-05 04:02:00	2014-01-05 11:24:00	Paris	Rome	354.50	442
t157	2014-01-05 07:12:00	2014-01-05 14:31:00	Paris	Rome	462.00	439
t158	2014-01-05 10:46:00	2014-01-05 18:06:00	Paris	Rome	626.75	440
t159	2014-01-05 13:20:00	2014-01-05 19:49:00	Paris	Rome	451.50	389
t160	2014-01-05 16:33:00	2014-01-05 23:37:00	Paris	Rome	588.05	424
t161	2014-01-05 19:04:00	2014-01-06 01:42:00	Paris	Rome	354.97	398
t162	2014-01-06 04:02:00	2014-01-06 10:29:00	Paris	Rome	403.47	387
t163	2014-01-06 07:12:00	2014-01-06 13:18:00	Paris	Rome	445.22	366
t164	2014-01-06 10:46:00	2014-01-06 17:23:00	Paris	Rome	443.97	397
t165	2014-01-06 13:20:00	2014-01-06 19:54:00	Paris	Rome	529.53	394
t166	2014-01-06 16:33:00	2014-01-06 22:47:00	Paris	Rome	426.61	374
t167	2014-01-06 19:04:00	2014-01-07 01:35:00	Paris	Rome	542.88	391
t168	2014-01-07 04:02:00	2014-01-07 10:10:00	Paris	Rome	519.38	368
t169	2014-01-07 07:12:00	2014-01-07 14:30:00	Paris	Rome	443.76	438
t170	2014-01-07 10:46:00	2014-01-07 17:46:00	Paris	Rome	454.03	420
t171	2014-01-07 13:20:00	2014-01-07 19:37:00	Paris	Rome	511.62	377
t172	2014-01-07 16:33:00	2014-01-07 23:55:00	Paris	Rome	340.07	442

Continued on next page

Table 3.2 – continued from previous page

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t173	2014-01-07 19:04:00	2014-01-08 01:56:00	Paris	Rome	577.11	412
t174	2014-01-08 04:02:00	2014-01-08 10:43:00	Paris	Rome	523.87	401
t175	2014-01-08 07:12:00	2014-01-08 14:09:00	Paris	Rome	546.84	417
t176	2014-01-08 10:46:00	2014-01-08 17:08:00	Paris	Rome	487.17	382
t177	2014-01-08 13:20:00	2014-01-08 20:09:00	Paris	Rome	352.81	409
t178	2014-01-08 16:33:00	2014-01-08 22:44:00	Paris	Rome	445.61	371
t179	2014-01-08 19:04:00	2014-01-09 01:22:00	Paris	Rome	445.72	378
t180	2014-01-09 04:02:00	2014-01-09 10:48:00	Paris	Rome	419.15	406
t181	2014-01-09 07:12:00	2014-01-09 13:20:00	Paris	Rome	327.44	368
t182	2014-01-09 10:46:00	2014-01-09 18:02:00	Paris	Rome	550.28	436
t183	2014-01-09 13:20:00	2014-01-09 19:55:00	Paris	Rome	352.59	395
t184	2014-01-09 16:33:00	2014-01-09 22:53:00	Paris	Rome	509.03	380
t185	2014-01-09 19:04:00	2014-01-10 01:56:00	Paris	Rome	539.56	412
t186	2014-01-10 04:02:00	2014-01-10 11:09:00	Paris	Rome	518.00	427
t187	2014-01-10 07:12:00	2014-01-10 14:06:00	Paris	Rome	620.19	414
t188	2014-01-10 10:46:00	2014-01-10 17:38:00	Paris	Rome	350.55	412
t189	2014-01-10 13:20:00	2014-01-10 19:35:00	Paris	Rome	591.21	375
t190	2014-01-10 16:33:00	2014-01-10 23:34:00	Paris	Rome	327.20	421
t191	2014-01-10 19:04:00	2014-01-11 02:31:00	Paris	Rome	486.53	447
t192	2014-01-11 04:02:00	2014-01-11 10:56:00	Paris	Rome	445.36	414
t193	2014-01-11 07:12:00	2014-01-11 13:41:00	Paris	Rome	483.15	389
t194	2014-01-11 10:46:00	2014-01-11 17:58:00	Paris	Rome	433.03	432
t195	2014-01-11 13:20:00	2014-01-11 20:35:00	Paris	Rome	378.04	435
t196	2014-01-11 16:33:00	2014-01-11 23:10:00	Paris	Rome	324.03	397
t197	2014-01-11 19:04:00	2014-01-12 01:41:00	Paris	Rome	481.25	397
t198	2014-01-12 04:02:00	2014-01-12 11:26:00	Paris	Rome	583.53	444
t199	2014-01-12 07:12:00	2014-01-12 14:16:00	Paris	Rome	435.59	424
t200	2014-01-12 10:46:00	2014-01-12 16:48:00	Paris	Rome	388.22	362
t201	2014-01-12 13:20:00	2014-01-12 19:58:00	Paris	Rome	343.38	398
t202	2014-01-12 16:33:00	2014-01-12 22:41:00	Paris	Rome	344.88	368
t203	2014-01-12 19:04:00	2014-01-13 02:32:00	Paris	Rome	387.76	448
t204	2014-01-05 05:20:00	2014-01-05 09:30:00	Paris	London	499.23	250
t205	2014-01-05 08:09:00	2014-01-05 12:28:00	Paris	London	520.80	259
t206	2014-01-05 10:32:00	2014-01-05 14:48:00	Paris	London	432.83	256
t207	2014-01-05 13:55:00	2014-01-05 18:31:00	Paris	London	528.37	276
t208	2014-01-05 16:25:00	2014-01-05 21:32:00	Paris	London	498.17	307
t209	2014-01-05 20:18:00	2014-01-06 01:05:00	Paris	London	562.07	287
t210	2014-01-06 05:20:00	2014-01-06 10:03:00	Paris	London	601.59	283
t211	2014-01-06 08:09:00	2014-01-06 12:09:00	Paris	London	445.38	240
t212	2014-01-06 10:32:00	2014-01-06 15:10:00	Paris	London	600.09	278
t213	2014-01-06 13:55:00	2014-01-06 18:00:00	Paris	London	338.51	245
t214	2014-01-06 16:25:00	2014-01-06 20:43:00	Paris	London	533.97	258
t215	2014-01-06 20:18:00	2014-01-07 01:41:00	Paris	London	390.81	323
t216	2014-01-07 05:20:00	2014-01-07 09:39:00	Paris	London	487.78	259
t217	2014-01-07 08:09:00	2014-01-07 13:26:00	Paris	London	561.55	317
t218	2014-01-07 10:32:00	2014-01-07 15:44:00	Paris	London	553.22	312
t219	2014-01-07 13:55:00	2014-01-07 18:09:00	Paris	London	545.30	254
t220	2014-01-07 16:25:00	2014-01-07 21:19:00	Paris	London	470.14	294
t221	2014-01-07 20:18:00	2014-01-08 00:28:00	Paris	London	379.13	250
t222	2014-01-08 05:20:00	2014-01-08 10:25:00	Paris	London	382.77	305
t223	2014-01-08 08:09:00	2014-01-08 13:06:00	Paris	London	486.71	297
t224	2014-01-08 10:32:00	2014-01-08 15:45:00	Paris	London	455.94	313
t225	2014-01-08 13:55:00	2014-01-08 18:38:00	Paris	London	625.18	283
t226	2014-01-08 16:25:00	2014-01-08 21:47:00	Paris	London	412.78	322
t227	2014-01-08 20:18:00	2014-01-09 00:22:00	Paris	London	342.89	244
t228	2014-01-09 05:20:00	2014-01-09 09:46:00	Paris	London	547.01	266
t229	2014-01-09 08:09:00	2014-01-09 13:13:00	Paris	London	625.32	304

Continued on next page

Table 3.2 – continued from previous page

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t230	2014-01-09 10:32:00	2014-01-09 15:09:00	Paris	London	564.41	277
t231	2014-01-09 13:55:00	2014-01-09 18:10:00	Paris	London	395.84	255
t232	2014-01-09 16:25:00	2014-01-09 20:34:00	Paris	London	329.52	249
t233	2014-01-09 20:18:00	2014-01-10 01:31:00	Paris	London	446.21	313
t234	2014-01-10 05:20:00	2014-01-10 10:02:00	Paris	London	594.55	282
t235	2014-01-10 08:09:00	2014-01-10 13:28:00	Paris	London	454.11	319
t236	2014-01-10 10:32:00	2014-01-10 15:37:00	Paris	London	407.51	305
t237	2014-01-10 13:55:00	2014-01-10 18:31:00	Paris	London	612.68	276
t238	2014-01-10 16:25:00	2014-01-10 20:58:00	Paris	London	561.31	273
t239	2014-01-10 20:18:00	2014-01-11 01:04:00	Paris	London	510.50	286
t240	2014-01-11 05:20:00	2014-01-11 10:40:00	Paris	London	318.85	320
t241	2014-01-11 08:09:00	2014-01-11 13:15:00	Paris	London	611.48	306
t242	2014-01-11 10:32:00	2014-01-11 14:32:00	Paris	London	530.30	240
t243	2014-01-11 13:55:00	2014-01-11 18:57:00	Paris	London	388.73	302
t244	2014-01-11 16:25:00	2014-01-11 21:47:00	Paris	London	325.80	322
t245	2014-01-11 20:18:00	2014-01-12 01:21:00	Paris	London	320.36	303
t246	2014-01-12 05:20:00	2014-01-12 09:35:00	Paris	London	610.19	255
t247	2014-01-12 08:09:00	2014-01-12 12:52:00	Paris	London	536.39	283
t248	2014-01-12 10:32:00	2014-01-12 14:44:00	Paris	London	540.17	252
t249	2014-01-12 13:55:00	2014-01-12 18:27:00	Paris	London	505.80	272
t250	2014-01-12 16:25:00	2014-01-12 21:49:00	Paris	London	487.26	324
t251	2014-01-12 20:18:00	2014-01-13 01:41:00	Paris	London	461.54	323
t252	2014-01-01 04:57:00	2014-01-01 09:15:00	Lisbon	London	407.11	258
t253	2014-01-01 08:08:00	2014-01-01 13:19:00	Lisbon	London	425.01	311
t254	2014-01-01 10:32:00	2014-01-01 14:55:00	Lisbon	London	507.11	263
t255	2014-01-01 14:03:00	2014-01-01 18:29:00	Lisbon	London	621.47	266
t256	2014-01-01 16:52:00	2014-01-01 22:20:00	Lisbon	London	364.57	328
t257	2014-01-01 20:12:00	2014-01-02 00:16:00	Lisbon	London	398.95	244
t258	2014-01-13 05:22:00	2014-01-13 10:27:00	London	Lisbon	377.79	305
t259	2014-01-13 07:10:00	2014-01-13 13:38:00	London	Lisbon	482.11	388
t260	2014-01-13 10:30:00	2014-01-13 15:38:00	London	Lisbon	436.46	308
t261	2014-01-13 13:15:00	2014-01-13 19:32:00	London	Lisbon	394.28	377
t262	2014-01-13 16:10:00	2014-01-13 22:00:00	London	Lisbon	533.68	350
t263	2014-01-13 20:20:00	2014-01-14 01:53:00	London	Lisbon	528.85	333
t264	2014-01-14 05:22:00	2014-01-14 10:52:00	London	Lisbon	621.64	330
t265	2014-01-14 07:10:00	2014-01-14 13:18:00	London	Lisbon	594.81	368
t266	2014-01-14 10:30:00	2014-01-14 15:58:00	London	Lisbon	632.25	328
t267	2014-01-14 13:15:00	2014-01-14 19:14:00	London	Lisbon	432.34	359
t268	2014-01-14 16:10:00	2014-01-14 21:56:00	London	Lisbon	332.26	346
t269	2014-01-14 20:20:00	2014-01-15 02:03:00	London	Lisbon	420.49	343
t270	2014-01-15 05:22:00	2014-01-15 11:43:00	London	Lisbon	474.56	381
t271	2014-01-15 07:10:00	2014-01-15 12:59:00	London	Lisbon	372.26	349
t272	2014-01-15 10:30:00	2014-01-15 16:44:00	London	Lisbon	344.81	374
t273	2014-01-15 13:15:00	2014-01-15 19:20:00	London	Lisbon	460.64	365
t274	2014-01-15 16:10:00	2014-01-15 21:13:00	London	Lisbon	611.85	303
t275	2014-01-15 20:20:00	2014-01-16 02:29:00	London	Lisbon	558.44	369
t276	2014-01-16 05:22:00	2014-01-16 10:41:00	London	Lisbon	416.14	319
t277	2014-01-16 07:10:00	2014-01-16 13:31:00	London	Lisbon	597.97	381
t278	2014-01-16 10:30:00	2014-01-16 15:33:00	London	Lisbon	537.69	303
t279	2014-01-16 13:15:00	2014-01-16 18:44:00	London	Lisbon	463.98	329
t280	2014-01-16 16:10:00	2014-01-16 21:18:00	London	Lisbon	401.38	308
t281	2014-01-16 20:20:00	2014-01-17 02:02:00	London	Lisbon	499.33	342
t282	2014-01-05 04:53:00	2014-01-05 08:30:00	London	Rome	576.36	217
t283	2014-01-05 08:03:00	2014-01-05 11:05:00	London	Rome	411.95	182
t284	2014-01-05 10:32:00	2014-01-05 13:37:00	London	Rome	365.56	185
t285	2014-01-05 13:56:00	2014-01-05 18:12:00	London	Rome	422.06	256
t286	2014-01-05 16:39:00	2014-01-05 20:41:00	London	Rome	574.42	242

Continued on next page

Table 3.2 – continued from previous page

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t287	2014-01-05 19:37:00	2014-01-05 22:57:00	London	Rome	362.23	200
t288	2014-01-06 04:53:00	2014-01-06 08:03:00	London	Rome	389.64	190
t289	2014-01-06 08:03:00	2014-01-06 11:06:00	London	Rome	339.70	183
t290	2014-01-06 10:32:00	2014-01-06 14:15:00	London	Rome	540.34	223
t291	2014-01-06 13:56:00	2014-01-06 17:48:00	London	Rome	442.53	232
t292	2014-01-06 16:39:00	2014-01-06 20:59:00	London	Rome	415.94	260
t293	2014-01-06 19:37:00	2014-01-06 23:59:00	London	Rome	544.41	262
t294	2014-01-07 04:53:00	2014-01-07 08:25:00	London	Rome	423.86	212
t295	2014-01-07 08:03:00	2014-01-07 12:26:00	London	Rome	547.29	263
t296	2014-01-07 10:32:00	2014-01-07 14:21:00	London	Rome	574.84	229
t297	2014-01-07 13:56:00	2014-01-07 17:47:00	London	Rome	386.58	231
t298	2014-01-07 16:39:00	2014-01-07 19:55:00	London	Rome	624.81	196
t299	2014-01-07 19:37:00	2014-01-07 22:47:00	London	Rome	369.16	190
t300	2014-01-08 04:53:00	2014-01-08 08:14:00	London	Rome	409.64	201
t301	2014-01-08 08:03:00	2014-01-08 11:14:00	London	Rome	374.65	191
t302	2014-01-08 10:32:00	2014-01-08 13:52:00	London	Rome	426.86	200
t303	2014-01-08 13:56:00	2014-01-08 17:44:00	London	Rome	469.25	228
t304	2014-01-08 16:39:00	2014-01-08 20:57:00	London	Rome	482.52	258
t305	2014-01-08 19:37:00	2014-01-08 23:48:00	London	Rome	586.94	251
t306	2014-01-09 04:53:00	2014-01-09 07:55:00	London	Rome	598.23	182
t307	2014-01-09 08:03:00	2014-01-09 12:05:00	London	Rome	387.35	242
t308	2014-01-09 10:32:00	2014-01-09 13:36:00	London	Rome	514.25	184
t309	2014-01-09 13:56:00	2014-01-09 18:25:00	London	Rome	353.13	269
t310	2014-01-09 16:39:00	2014-01-09 20:49:00	London	Rome	462.61	250
t311	2014-01-09 19:37:00	2014-01-09 22:57:00	London	Rome	419.56	200
t312	2014-01-10 04:53:00	2014-01-10 08:37:00	London	Rome	417.72	224
t313	2014-01-10 08:03:00	2014-01-10 12:09:00	London	Rome	470.05	246
t314	2014-01-10 10:32:00	2014-01-10 14:47:00	London	Rome	547.94	255
t315	2014-01-10 13:56:00	2014-01-10 17:16:00	London	Rome	339.80	200
t316	2014-01-10 16:39:00	2014-01-10 19:51:00	London	Rome	594.99	192
t317	2014-01-10 19:37:00	2014-01-11 00:00:00	London	Rome	549.51	263
t318	2014-01-11 04:53:00	2014-01-11 08:53:00	London	Rome	373.61	240
t319	2014-01-11 08:03:00	2014-01-11 12:21:00	London	Rome	613.89	258
t320	2014-01-11 10:32:00	2014-01-11 13:52:00	London	Rome	477.52	200
t321	2014-01-11 13:56:00	2014-01-11 17:07:00	London	Rome	633.00	191
t322	2014-01-11 16:39:00	2014-01-11 20:10:00	London	Rome	380.15	211
t323	2014-01-11 19:37:00	2014-01-11 22:59:00	London	Rome	486.53	202
t324	2014-01-12 04:53:00	2014-01-12 07:59:00	London	Rome	409.45	186
t325	2014-01-12 08:03:00	2014-01-12 12:01:00	London	Rome	413.84	238
t326	2014-01-12 10:32:00	2014-01-12 13:54:00	London	Rome	504.22	202
t327	2014-01-12 13:56:00	2014-01-12 17:36:00	London	Rome	608.40	220
t328	2014-01-12 16:39:00	2014-01-12 20:04:00	London	Rome	571.71	205
t329	2014-01-12 19:37:00	2014-01-12 23:59:00	London	Rome	546.09	262
t330	2014-01-05 04:03:00	2014-01-05 09:10:00	London	Paris	459.85	307
t331	2014-01-05 08:03:00	2014-01-05 13:22:00	London	Paris	595.21	319
t332	2014-01-05 11:22:00	2014-01-05 15:31:00	London	Paris	331.87	249
t333	2014-01-05 13:47:00	2014-01-05 18:55:00	London	Paris	486.81	308
t334	2014-01-05 16:31:00	2014-01-05 20:32:00	London	Paris	460.64	241
t335	2014-01-05 19:10:00	2014-01-06 00:37:00	London	Paris	420.75	327
t336	2014-01-06 04:03:00	2014-01-06 08:58:00	London	Paris	412.58	295
t337	2014-01-06 08:03:00	2014-01-06 12:52:00	London	Paris	402.06	289
t338	2014-01-06 11:22:00	2014-01-06 15:48:00	London	Paris	380.87	266
t339	2014-01-06 13:47:00	2014-01-06 19:10:00	London	Paris	473.21	323
t340	2014-01-06 16:31:00	2014-01-06 20:54:00	London	Paris	426.74	263
t341	2014-01-06 19:10:00	2014-01-07 00:24:00	London	Paris	422.16	314
t342	2014-01-07 04:03:00	2014-01-07 09:31:00	London	Paris	338.86	328
t343	2014-01-07 08:03:00	2014-01-07 13:13:00	London	Paris	322.37	310

Continued on next page

Table 3.2 – continued from previous page

	DateStart	DateEnd	Dest Start	Dest End	Cost	Time (min)
t344	2014-01-07 11:22:00	2014-01-07 16:08:00	London	Paris	605.23	286
t345	2014-01-07 13:47:00	2014-01-07 17:52:00	London	Paris	448.36	245
t346	2014-01-07 16:31:00	2014-01-07 21:13:00	London	Paris	531.31	282
t347	2014-01-07 19:10:00	2014-01-07 23:49:00	London	Paris	460.54	279
t348	2014-01-08 04:03:00	2014-01-08 08:21:00	London	Paris	381.22	258
t349	2014-01-08 08:03:00	2014-01-08 12:41:00	London	Paris	610.84	278
t350	2014-01-08 11:22:00	2014-01-08 15:54:00	London	Paris	356.65	272
t351	2014-01-08 13:47:00	2014-01-08 18:01:00	London	Paris	469.67	254
t352	2014-01-08 16:31:00	2014-01-08 21:10:00	London	Paris	346.91	279
t353	2014-01-08 19:10:00	2014-01-08 23:33:00	London	Paris	390.24	263
t354	2014-01-09 04:03:00	2014-01-09 08:49:00	London	Paris	533.95	286
t355	2014-01-09 08:03:00	2014-01-09 12:06:00	London	Paris	483.85	243
t356	2014-01-09 11:22:00	2014-01-09 16:43:00	London	Paris	491.30	321
t357	2014-01-09 13:47:00	2014-01-09 19:00:00	London	Paris	580.21	313
t358	2014-01-09 16:31:00	2014-01-09 21:20:00	London	Paris	337.15	289
t359	2014-01-09 19:10:00	2014-01-10 00:26:00	London	Paris	603.08	316
t360	2014-01-10 04:03:00	2014-01-10 09:29:00	London	Paris	523.18	326
t361	2014-01-10 08:03:00	2014-01-10 12:12:00	London	Paris	519.93	249
t362	2014-01-10 11:22:00	2014-01-10 16:18:00	London	Paris	327.52	296
t363	2014-01-10 13:47:00	2014-01-10 19:16:00	London	Paris	604.73	329
t364	2014-01-10 16:31:00	2014-01-10 21:59:00	London	Paris	326.82	328
t365	2014-01-10 19:10:00	2014-01-11 00:04:00	London	Paris	376.43	294
t366	2014-01-11 04:03:00	2014-01-11 08:14:00	London	Paris	600.27	251
t367	2014-01-11 08:03:00	2014-01-11 12:55:00	London	Paris	357.93	292
t368	2014-01-11 11:22:00	2014-01-11 15:22:00	London	Paris	443.39	240
t369	2014-01-11 13:47:00	2014-01-11 18:04:00	London	Paris	557.46	257
t370	2014-01-11 16:31:00	2014-01-11 21:57:00	London	Paris	495.37	326
t371	2014-01-11 19:10:00	2014-01-11 23:39:00	London	Paris	548.15	269
t372	2014-01-12 04:03:00	2014-01-12 09:00:00	London	Paris	378.76	297
t373	2014-01-12 08:03:00	2014-01-12 12:28:00	London	Paris	447.82	265
t374	2014-01-12 11:22:00	2014-01-12 16:47:00	London	Paris	475.03	325
t375	2014-01-12 13:47:00	2014-01-12 18:52:00	London	Paris	394.76	305
t376	2014-01-12 16:31:00	2014-01-12 21:00:00	London	Paris	632.40	269
t377	2014-01-12 19:10:00	2014-01-13 00:21:00	London	Paris	529.44	311

The MOEA/D-ACO Algorithm is used in this step. As described in Section 2.5.5, it initializes a set of ants with weight vectors uniformly distributed over the objective space. They are then separated in groups, each with a pheromone matrix. After that, a loop starts: each ant generate iteratively a solution, the solutions are evaluated and the estimated Pareto Set is built and the pheromone matrices are updated. The loop ends after 60 seconds, the established stop condition.

The resulting solution set has a 0.9937 relative hypervolume and 0.71 Pareto Success Rate. The solutions are presented in Figure 3.2 and Table 3.3.

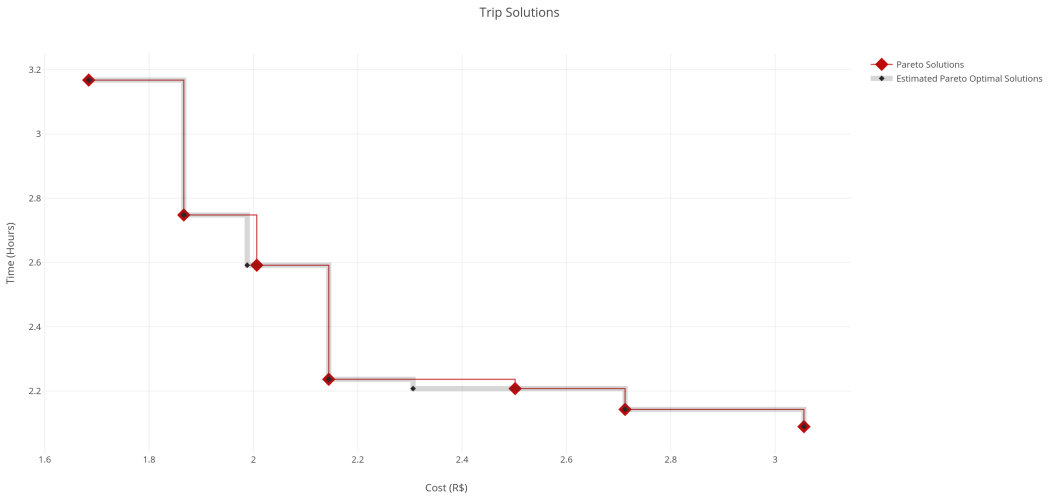


Figure 3.2: Trip Solutions (normalized values)

Components	Cost	Time	Destinations
t128-a187-t156-a69-t103-a277-t258	2570.47	1419.0	Lisbon->Paris->Rome->London->Lisbon
t128-a187-t204-a239-t306-a22-t7	3050.33	936.0	Lisbon->Paris->London->Rome->Lisbon
t128-a187-t204-a239-t306-a65-t13	2930.33	960.0	Lisbon->Paris->London->Rome->Lisbon
t128-a187-t204-a265-t312-a57-t13	2731.52	1002.0	Lisbon->Paris->London->Rome->Lisbon
t128-a187-t204-a299-t312-a32-t18	2856.58	989.0	Lisbon->Paris->London->Rome->Lisbon
t252-a221-t330-a120-t181-a65-t13	2683.23	1161.0	Lisbon->London->Paris->Rome->Lisbon
t256-a221-t330-a103-t181-a65-t13	2634.27	1231.0	Lisbon->London->Paris->Rome->Lisbon

Table 3.3: Optimization Results

Given the solutions, the decision support system aides the user to chose the best one. The traveler assigns 0.7 importance to cost and 0.3 to travel time, leading to the ranking presented in Table 3.4.

3.6 Summary

This chapter presents every step of the proposed solution, covering the problem representation and data structure; data gathering, generation and treatment; optimization methods and variations proposed, as well as a discussion about the chosen method and

Components	Cost	Time	Destinations	Ranking
t128-a187-t204-a265-t312-a57-t13	2731.52	1002.0	Lisbon->Paris->London->Rome->Lisbon	0
t252-a221-t330-a120-t181-a65-t13	2683.23	1161.0	Lisbon->London->Paris->Rome->Lisbon	1
t128-a187-t204-a299-t312-a32-t18	2856.58	989.0	Lisbon->Paris->London->Rome->Lisbon	2
t256-a221-t330-a103-t181-a65-t13	2634.27	1231.0	Lisbon->London->Paris->Rome->Lisbon	3
t128-a187-t204-a239-t306-a65-t13	2930.33	960.0	Lisbon->Paris->London->Rome->Lisbon	4
t128-a187-t156-a69-t103-a277-t258	2570.47	1419.0	Lisbon->Paris->Rome->London->Lisbon	5
t128-a187-t204-a239-t306-a22-t7	3050.33	936.0	Lisbon->Paris->London->Rome->Lisbon	6

Table 3.4: Solution Ranking

robustness evaluation strategies; and the user interface for input and presenting the optimization results.

On the next chapter, the experiments performed to test the system are described and the results presented.

Chapter 4

Results

The methods presented in Chapter 3 were subject to several experiments in order to evaluate their performance and study their characteristics. This chapter presents the design, setup and results of these experiments for several instances of the trip itinerary planning problem. The proposed methods were evaluated independently and in comparison to each other, and the contribution of each element evaluated.

4.1 Experimental Definitions and Setup

4.1.1 Technical Information

All experiments were performed on a computer with the following specifications:

- Operating System: Windows 10 (version 1803)
- CPU: Intel i7 8700K 3.7 GHZ
- RAM: 8 GB DDR4

All computational implementation was developed using Python 3.6 and R. The code and experimental results are stored at .

4.1.2 Problem Instances

There are several trip characteristics that define the problem complexity and affect the optimization algorithm performance. They are:

1. Trip size, i.e. the number of destinations to be visited in the trip itinerary
2. Destination order restrictions
3. Allowed variation on the trip's start and end dates
4. Allowed variation in the stay length defined for each destination
5. Accommodation and transports restrictions
6. Average number of transports options per day between two destinations
7. Average number of accommodation options per day on a destination

In an experimental perspective, an instance of the trip itinerary planning problem is defined by the combination of these characteristics. Additional information of the itinerary, such as destination names and departing dates, do not affect the optimization results and are therefore disregarded.

The number of available solutions for the trip itinerary, both feasible and unfeasible, is related to the problem complexity. Problems with more solutions have a bigger search space and, thus, are harder to solve. The higher the trip size, the number of transports and accommodation options and the flexibility (allowed variation) on stay lengths and start and end dates, the higher the number of solutions and problem complexity. On the other hand, restrictions related to accommodation, transportation and destinations ordering reduce the number of valid solutions and make the problem easier.

The user is responsible for deciding items 1 to 5 when creating the trip itinerary. The search space calculation presented in Section 3.1.5 helps estimate the impact of some of these factors. The most significant one is the size of the trip, followed by the number of accommodation and transportation options. Although disregarded in the search space size expression, the ordering and stay variations also have a significant impact on the complexity. The average number of transports and accommodations options depends on external factors, but are affected by users restrictions. More restrictions reduces the number of available options.

The problem instances used for the experiments were generated using the Random Component Generator presented in Section 3.2. The trip characteristics listed can be viewed as parameters for the generation of an instance. The use of an automated generator allows total control over the parameters. From an experimental perspective, this helps the analysis by making it possible to change one parameter without affecting the others, which helps evaluating its effect.

The number of possible problem instances defined by the combination of all possible definitions is infinite. The instances considered were restricted to a small subset to keep

the experiments feasible and close to real world use cases. Unless otherwise stated, the results presented were obtained with instances with trip sizes varying between 2 and 7 destinations and the following values for the other parameters:

- Average number of transports options: 6
- Average number of accommodation options: 6
- Destination ordering: unordered
- Allowed variation for trip start and end dates: 0 days
- Allowed variation for stay length in each destinations: 1 day

The average number of transports and accommodations values defined are estimations of frequent values for trips itineraries, based on real world data collected. These values correspond to the average number of options left after some are excluded due to the user restrictions and the search space reduction method presented in Section 3.2.3.

The destination ordering was kept unordered as it allows for the maximum flexibility, which is the main advantage of the proposed solution over regular methods for trip planning. A maximum variation of 1 day is set for each destination. This value is considered a reasonable compromise between small variations in the desired stay days and a higher number of solutions, which may potentially lead to better itineraries.

For all instances generated, an estimated optimal Pareto Set was calculated. In instances with up to 4 destinations, the exact set was obtained through exhaustive search. For bigger problems, this is not viable. In such cases, the estimated set was generated aggregating the solutions of several (at least 750) optimization executions and random searches, with high time limits and varied parameters combinations.

While there is no guarantee that the estimated set is a good approximation of the real Pareto set, there are indications that they are good enough approximations:

- In the optimization experiments, all the solutions obtained were always worse or equal to the estimated set.
- For instances in which the exact solution is know, the estimated set was identical to the exact one.

For all distributed computing experiments performed, 10 out of the 12 logical cores present in the CPU were allocated to the processes. The remaining 2 were kept free so that they would handle any background process and minimize external factors.

4.1.3 Results Evaluation

Each solution is evaluated based on the travel time and cost, as presented in Section 3.1.3. The objective evaluations are given by Equations 4.1 and 4.2:

Cost:

$$F_1(s) = \sum_{i=1}^{k+1} T_{i.cost} + \sum_{j=1}^{k} A_{j.cost} \tag{4.1}$$

Travel Time:

$$F_2(s) = \sum_{i=1}^{k+1} T_{i.arrival\_time} - T_{i.departure\_time} \tag{4.2}$$

The hypervolume metric is used to evaluate the results of the solution sets obtained by the optimization methods proposed. The hypervolume is calculated based on the nadir point defined for each instance, as described in Section 2.6. In order to compare the optimization results with the optimal solution, the relative hypervolume HV_R is used. It is given by:

$$HV_R = \frac{HV_{opt}}{HV^*}$$

where HV_{opt} is the hypervolume of the optimization resulting solution set and HV^* is the hypervolume of the estimated optimal Pareto Set. The closer the relative hypervolume is to 1, the better is the result of the optimization.

The Pareto Success Rate (PSR) indicates the percentage of the Pareto Optimal Solution found by the optimization:

$$PSR = \frac{ParetoSolutionsFound}{TotalParetoSolutions}$$

Due to the stochastic nature of the optimization methods used, the results can be different for each execution. In order to have a better representation of the behaviour of each method, the average hypervolume of several executions is considered.

The stop criterion for the optimization algorithms was execution time. Although subject to external factors (such as processes in the background of the computer), these occasional disturbances are minimized in the averaging processes of the several executions of the methods. The execution time also provides a fair comparison criteria between methods with very different average iteration times (e.g. regular MOACO and Local-search MOACO). The maximum execution time was set at 80 seconds, as empirical results demonstrated that all methods converge within this time for all experiments. Convergence is defined as the moment when an iteration of the methods do not lead to improvements in the relative hypervolume greater than 0.005.

For all methods, the Relative Hypervolume over time plot is presented. This representation shows both the speed of convergence and the final convergence value for the methods, and highlights trade-offs between speed and final quality. It also makes it possible to compare the start-up times for different methods. In these plots, the shadowed region represents the 95% confidence interval of all the executions, as seen in Figure 4.1.

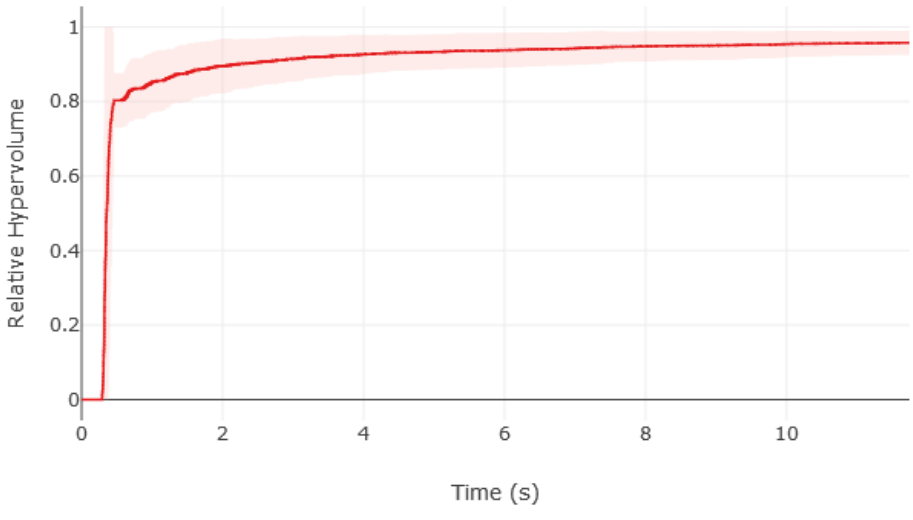


Figure 4.1: Average relative hypervolume over time for a problem instance with 5 destinations.

Plots of optimization solutions on the objective space are also used for qualitative evaluation, as in Figure 4.2. In such cases, the optimization execution represented is selected randomly among all executions.

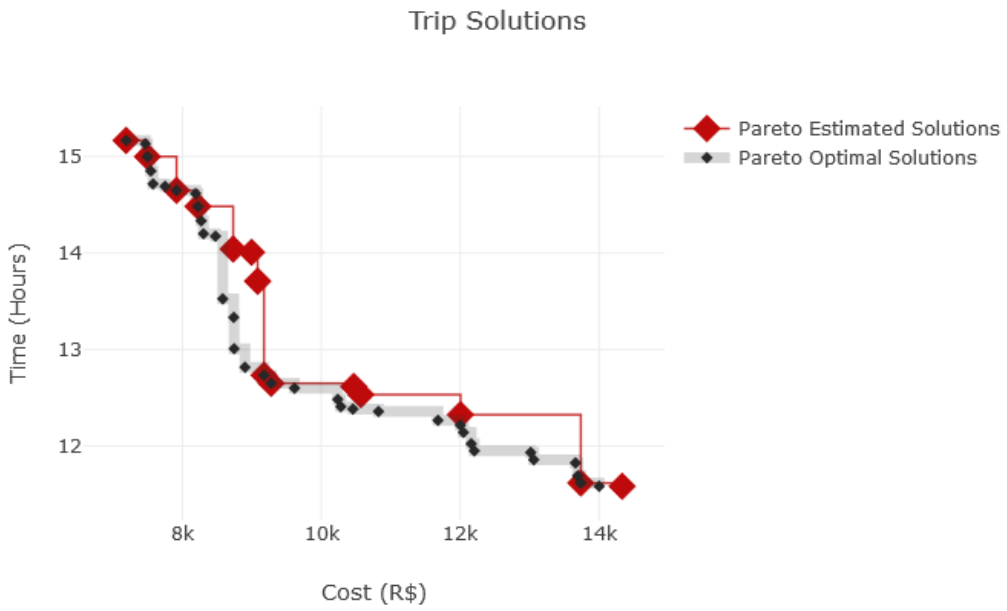


Figure 4.2: Solution set obtained after the optimization of a 5 destination problem instance.

The summed up results for all experiments is presented as a table. It contains the average and standard deviation of relative hypervolume at the 60 seconds mark, the average and standard deviation of the iteration count and the average start-up time. The start-up time is defined as the time necessary for the method to start and the first set of solutions to be obtained.

All relative hypervolume comparisons performed take into account the average relative hypervolume at the 60 seconds mark.

4.2 Experiments Results

This section presents the results for each method proposed, along with a brief commentary. The results are evaluated based on the metrics outlined in Section 4.1.3. Figure 4.3 presents a summarized representation of the results of 4 different methods.

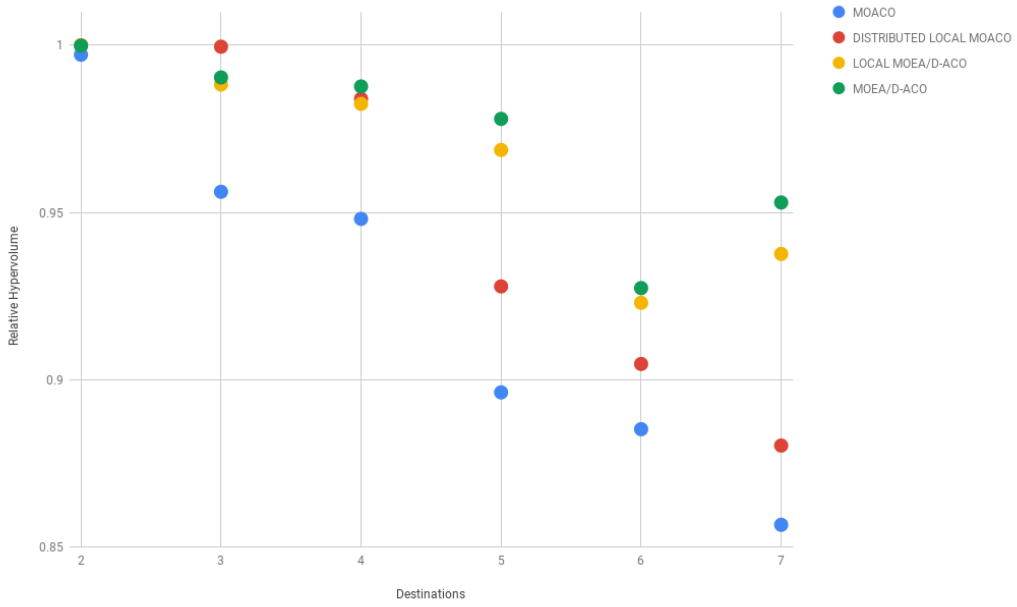


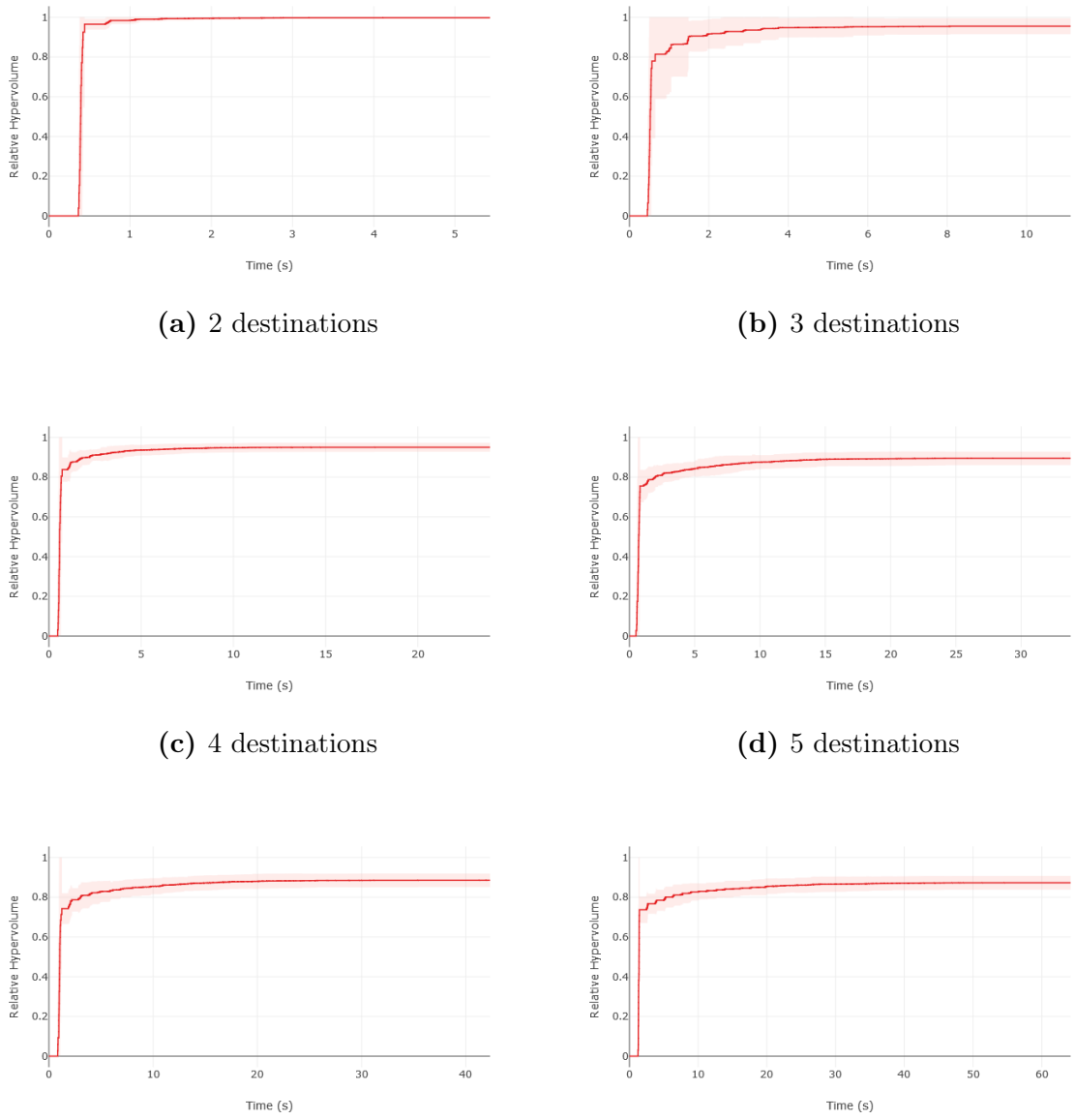
Figure 4.3: Summarized results for MOACO, Distributed Local MOACO, Local MOEA/D-ACO and MOEA/D-ACO.

4.2.1 Multiobjective Ant Colony Optimization (MOACO)

Figure 4.4 presents the relative hypervolume over time for executions of the MOACO algorithm described in Section 2.5.3. These results are summed up in Table 4.1. The following parameters values were used for the experiments:

- $\alpha_l = 1 \quad \forall l \in [1, L]$
- $\beta_m = 2 \quad \forall m \in [1, M]$
- $\gamma = 0.1$
- $\rho = 0.05$
- $\tau_0 = 1$
- $n = 30$

The results indicate that, for smaller problems, the algorithm quickly converges to solutions very close to estimated optimal Pareto set. As the problem size grows, the quality of the results worsens and the convergence times increases. This increased difficulty is expected, since the growth of the number of destinations very quickly leads to a drastic increase in the complexity due to the combinatorial explosion in the size



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

(e) 6 destinations

(f) 7 destinations

Figure 4.4: Relative Hypervolume over time results for MOACO method on instances with 2 to 7 destinations

of search space, as discussed in Section 3.1.5. The start-up time is very small for all problem sizes in this case.

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time	PSR
2	0.9972	0.0031	226.3500	2.2644	0.3895	0.984
3	0.9562	0.0221	179.0500	10.0473	0.5178	0.849
4	0.9481	0.0111	137.0923	2.0360	0.5611	0.644
5	0.8962	0.0166	109.6632	14.2324	0.7424	0.408
6	0.8852	0.0150	126.6000	1.5333	1.0363	0.317
7	0.8566	0.0175	96.0800	1.9271	1.3681	0.192

Table 4.1: Summarized results of MOACO experiments

4.2.2 Local Search MOACO

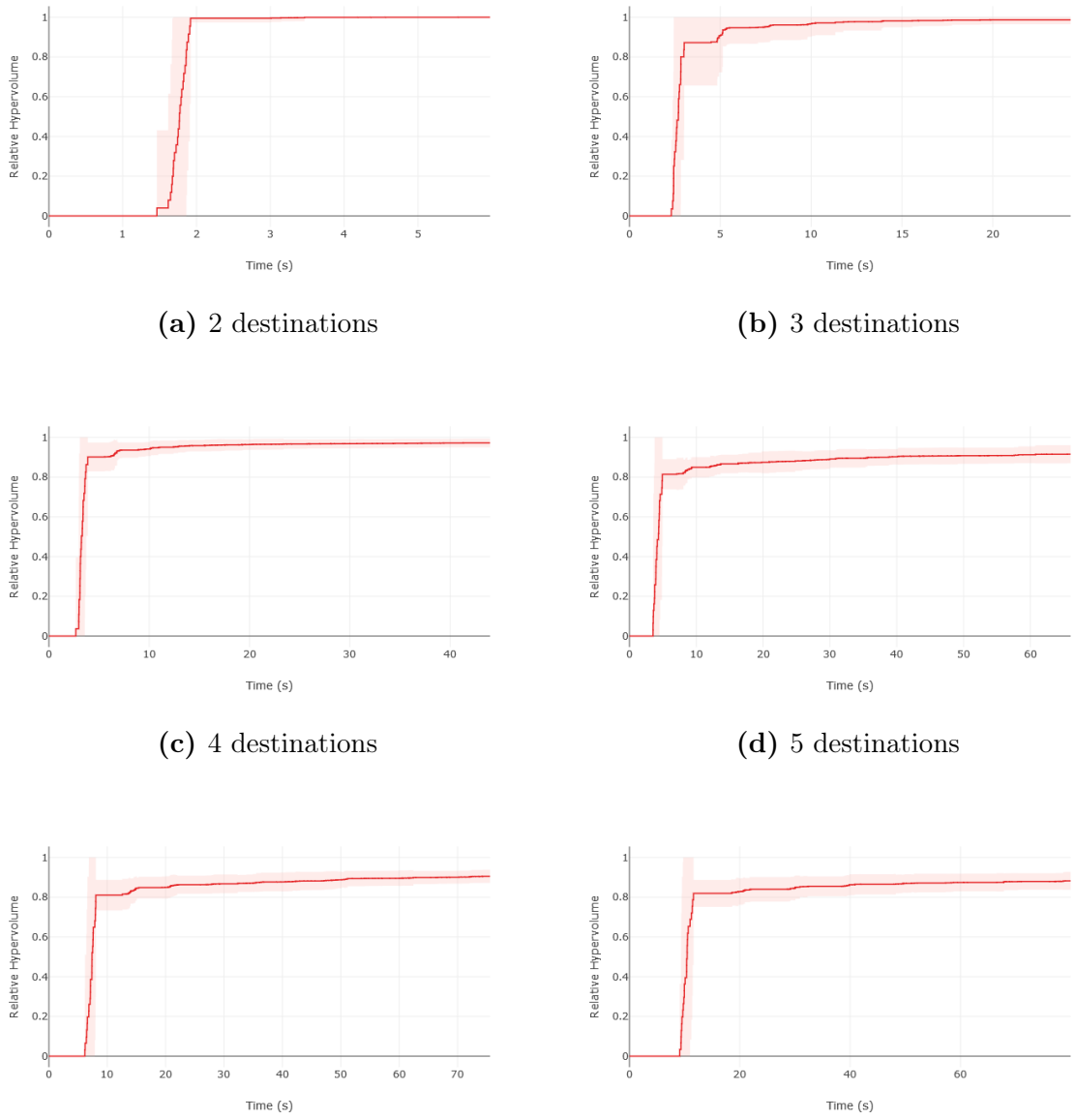
Figure 4.5 presents the relative hypervolume over time for executions of the Local Search MOACO method described in Section 3.3.2. These results are summed up in Table 4.2. The MOACO parameters are the same presented in 4.2.1. The Local search method selected is first-improvement strict-dominance search.

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time	PSR
2	1.0000	0.0000	78.6500	1.0137	1.7730	1
3	0.9851	0.0149	48.7667	0.7386	2.6390	0.803
4	0.9727	0.0102	30.3125	0.8887	3.3438	0.685
5	0.9125	0.0216	16.4632	1.4712	4.3776	0.475
6	0.8890	0.0198	14.2750	0.4994	7.3812	0.298
7	0.8582	0.0222	9.7400	0.4386	10.2604	0.208

Table 4.2: Summarized results of Local Search MOACO experiments

Overall, Local Search MOACO leads to better hypervolumes after convergence. However, the Local Search method has a much higher start-up time compared to the traditional MOACO method. This difference is more significant for bigger problems. This is consequence of the added overhead of the Local Search method.

The number of iterations is also much smaller, due to the extra computational cost of the Local-search, discussed on Section 3.3.2.



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

(e) 6 destinations

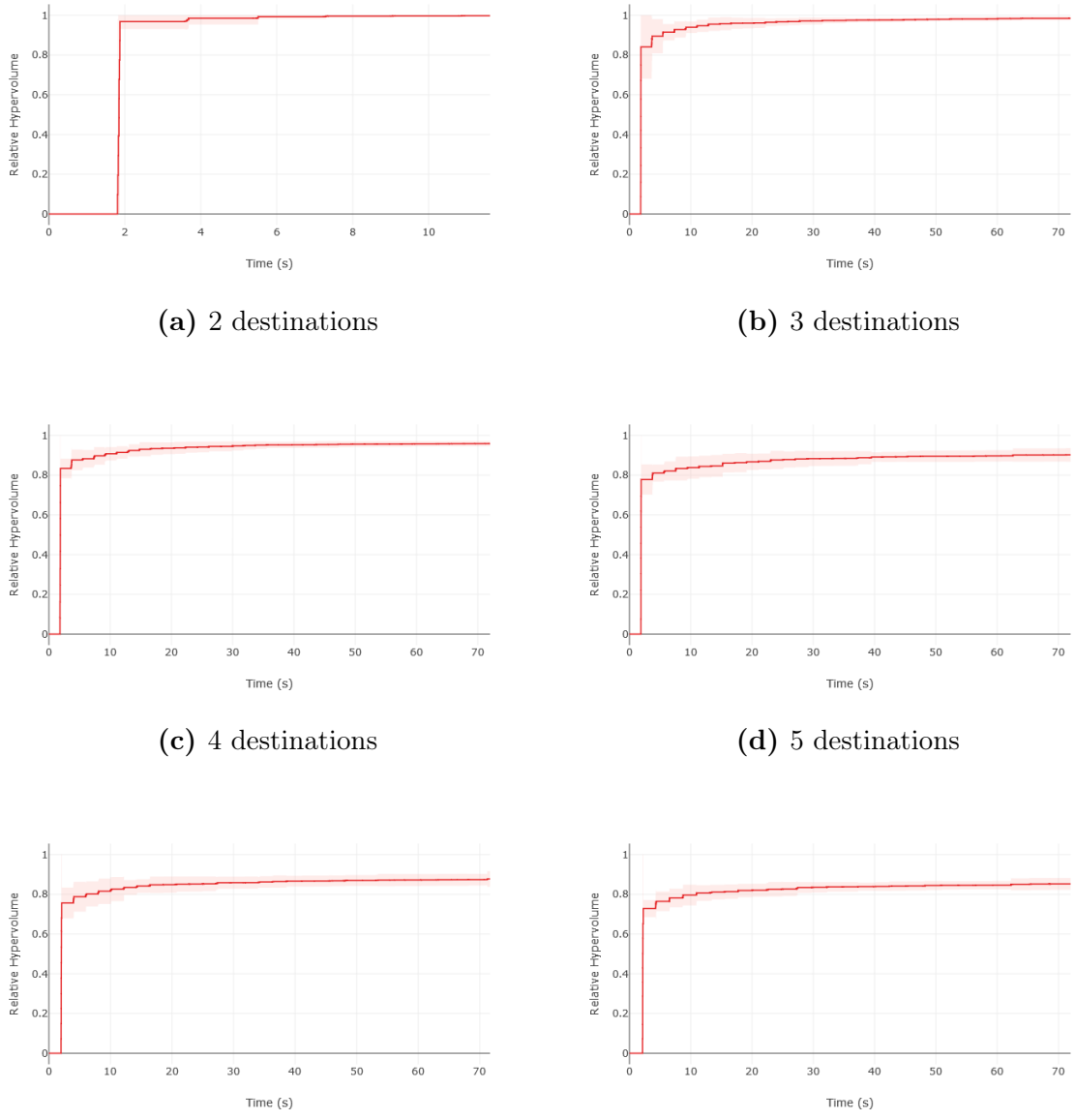
(f) 7 destinations

Figure 4.5: Relative Hypervolume over time results for Local Search MOACO method on instances with 2 to 7 destinations

4.2.3 Distributed MOACO

Figure 4.6 presents the relative hypervolume over time for executions of the Distributed MOACO method described in Section 3.3.3. These results are summed up in Table 4.3.

The MOACO parameters are the same presented in 4.2.1. The number of cores used for the tasks is 6.



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

(e) 6 destinations

(f) 7 destinations

Figure 4.6: Relative Hypervolume over time results for Distributed MOACO method on instances with 2 to 7 destinations

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time
2	0.9999	0.0002	37.4000	1.5232	2.0439
3	0.9844	0.0058	37.1333	0.5416	1.9513
4	0.9581	0.0083	36.5667	0.4955	1.8710
5	0.8920	0.0224	34.9000	0.3958	1.9164
6	0.8698	0.0191	38.4667	0.5207	2.0418
7	0.8463	0.0241	33.4250	1.8559	2.2869

Table 4.3: Summarized results of Distributed MOACO experiments

The results indicates that the distributed implementation of the MOACO algorithm is not a significant improvement compared to regular MOACO. Statistical tests show that there are no significant differences between the hypervolume after convergence.

There is, however, a significant difference in the number of iterations. This difference is higher for smaller problems, and get smaller as the number of destinations increases. For all experiments with up to 7 destinations, the non-distributed MOACO method has significantly more iterations. The hypervolume over time plots shows that the regular MOACO method converges faster than the distributed approach.

The reason for this is that the distributed approach has some computational overhead setting up the distribution structure and managing inter-process communications. In addition to that, the distributed approach has a bottleneck after each iteration in which all the solutions generated are evaluated and the pheromone matrices are updated. However, the solution construction processes being distributed are very quick. The results indicate that this method spends more time on the distribution overhead than on the actual distributed process. As a consequence, the speed-up brought by the parallel computing is overshadowed by the distribution slowdown, and the method in general performs less iterations.

As the number of destinations on the trip increases, the solution construction process becomes more complex. This increases the time spent on it, while the distribution overhead cost stay the same. This is the reason why the relative difference in the number of iterations of regular MOACO and Distributed MOACO becomes smaller as the problem grows. However, for up to 7 destinations, there are no improvements with Distributed

MOACO compared to regular MOACO is still better, as it converges faster without decreases in quality.

Distributed Local Search MOACO

The next experiment shows the results of the same distribution approach applied to the Local Search MOACO. The results are presented in Figure 4.7 and Table 4.4.

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time
2	1.0000	0.0000	34.2800	1.3121	2.1783
3	0.9996	0.0015	31.9778	0.3938	2.3604
4	0.9840	0.0028	27.7556	1.0361	2.5055
5	0.9279	0.0180	24.8833	0.3210	2.5029
6	0.9047	0.0147	23.0500	0.2179	3.1425
7	0.8803	0.0184	18.7625	0.4256	3.8479

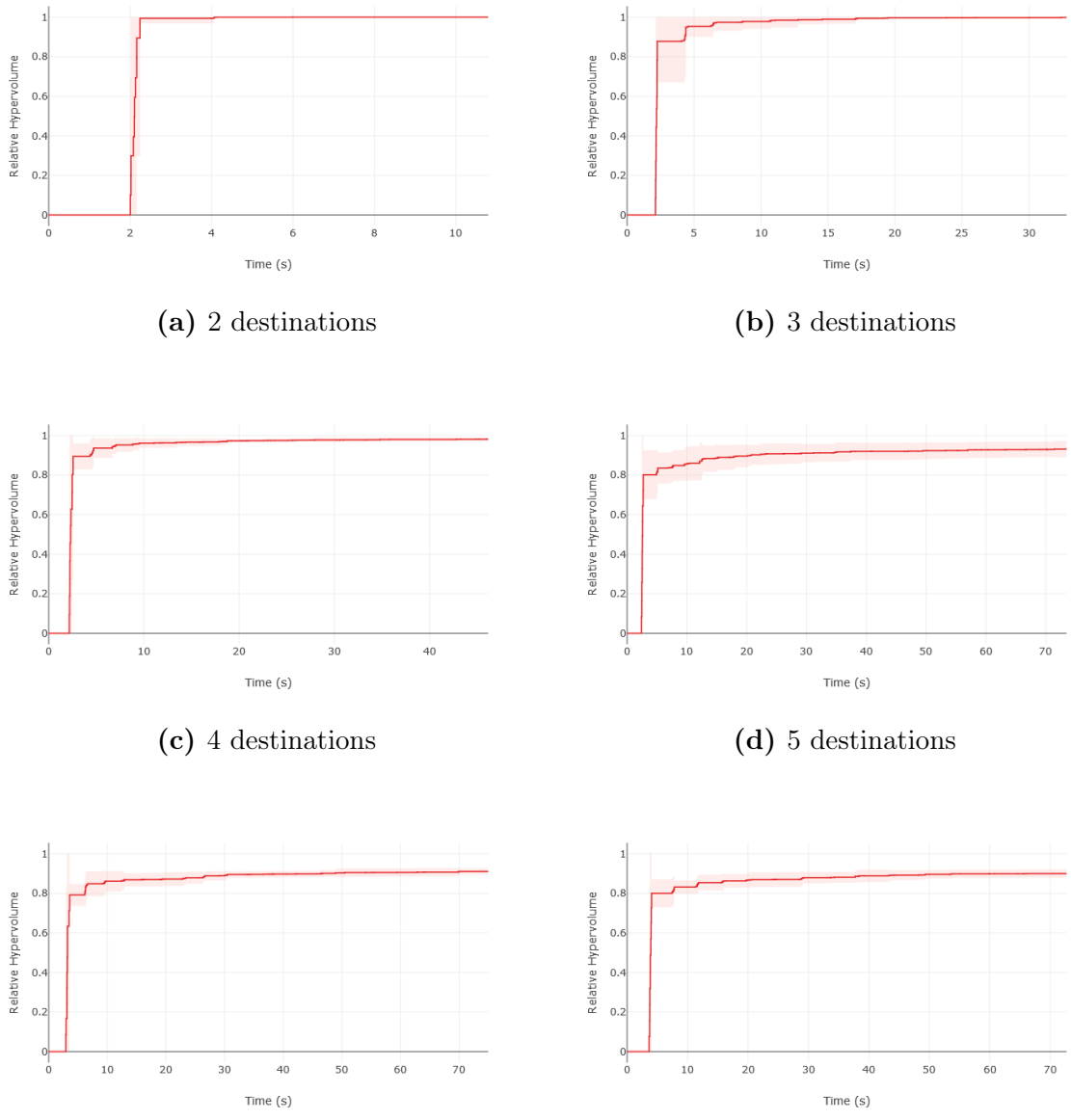
Table 4.4: Summarized results of Distributed Local Search MOACO experiments

Considering the convergence hypervolume, the Distributed Local Search MOACO method shows improvements for all trip sizes when compared to regular MOACO or Distributed MOACO. As for the non-distributed Local-search MOACO, the difference between the results is not statistically significant.

In terms of start-up time, however, the approach is only slightly worse than the regular MOACO method and not significantly different than Distributed MOACO. It is much better than the Local-search MOACO, though, which leads to better convergence times.

4.2.4 MOEA/D-ACO

Figure 4.8 presents the relative hypervolume over time for problem instances solved by the MOEA/D-ACO method described in Section 2.5.5. These results are summed up in Table 4.5. The ant colony parameters used are the same presented in 4.2.1. Additional parameters related to the MOEA/D method are:



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

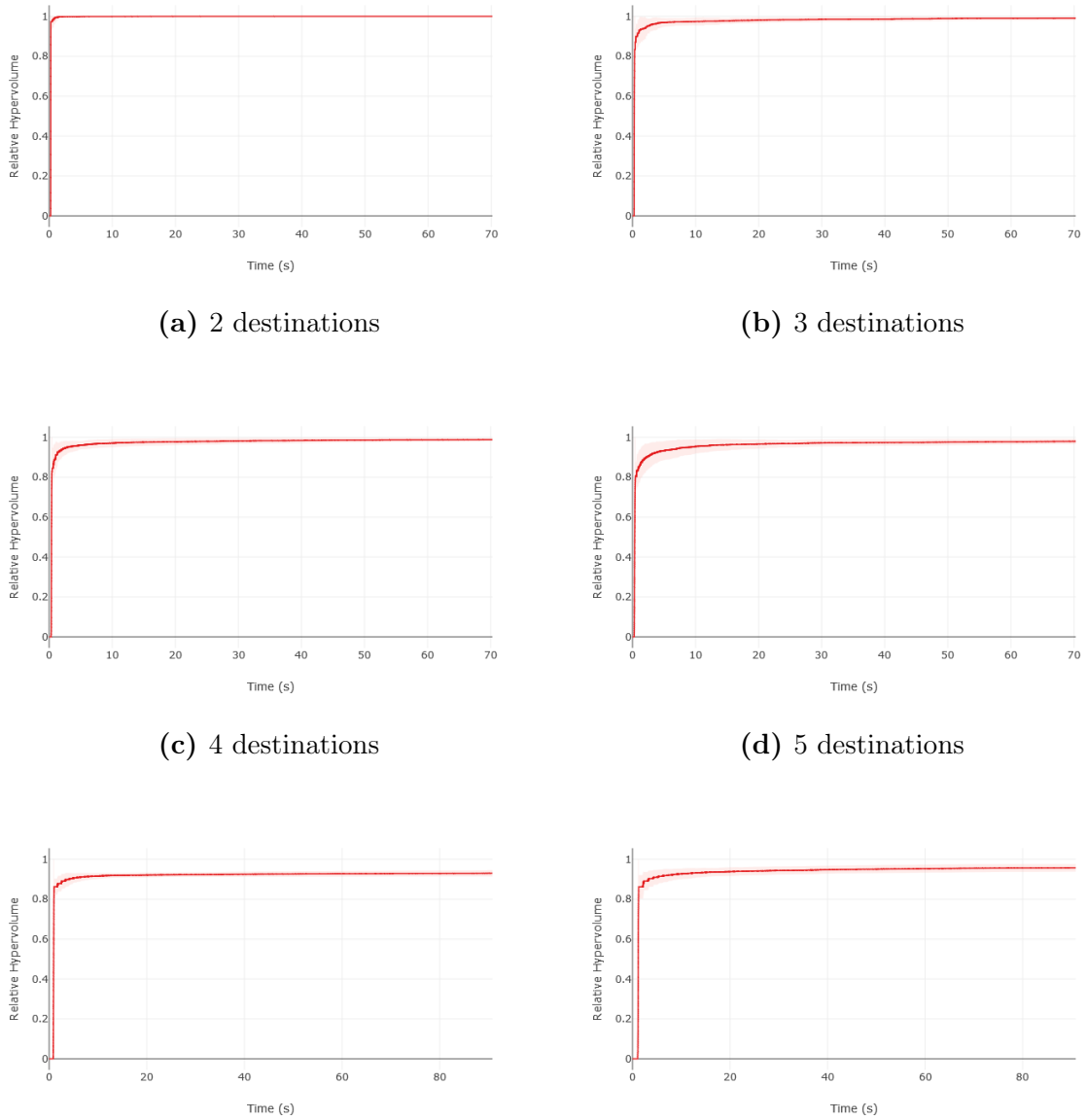
(e) 6 destinations

(f) 7 destinations

Figure 4.7: Relative Hypervolume over time results for Distributed Local Search MOACO method on instances with 2 to 7 destinations

- $K = 5$
- $T = 4$
- $\Delta = 0.05 \times \tau_{max}$
- $\epsilon = \frac{1}{\sqrt{|C|}}$

where K is the number of groups, T is the neighborhood size, Δ is the influence of the current solution, τ_{max} is the maximum pheromone value, ϵ is a modifier for the minimum pheromone value and $|C|$ is the size of the components set.



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

(e) 6 destinations

(f) 7 destinations

Figure 4.8: Relative Hypervolume over time results for MOEA/D-ACO method on instances with 2 to 7 destinations

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time	PSR
2	0.9999	0.0000	343.3600	2.7405	0.2659	0.978
3	0.9904	0.0053	270.5200	1.8137	0.3053	0.811
4	0.9877	0.0049	243.9000	4.3324	0.3789	0.692
5	0.9780	0.0070	260.0500	4.1440	0.3503	0.503
6	0.9274	0.0069	124.6923	1.4876	0.9071	0.435
7	0.9530	0.0100	97.7556	1.0145	1.1778	0.295

Table 4.5: Summarized results of MOEA/D-ACO experiments

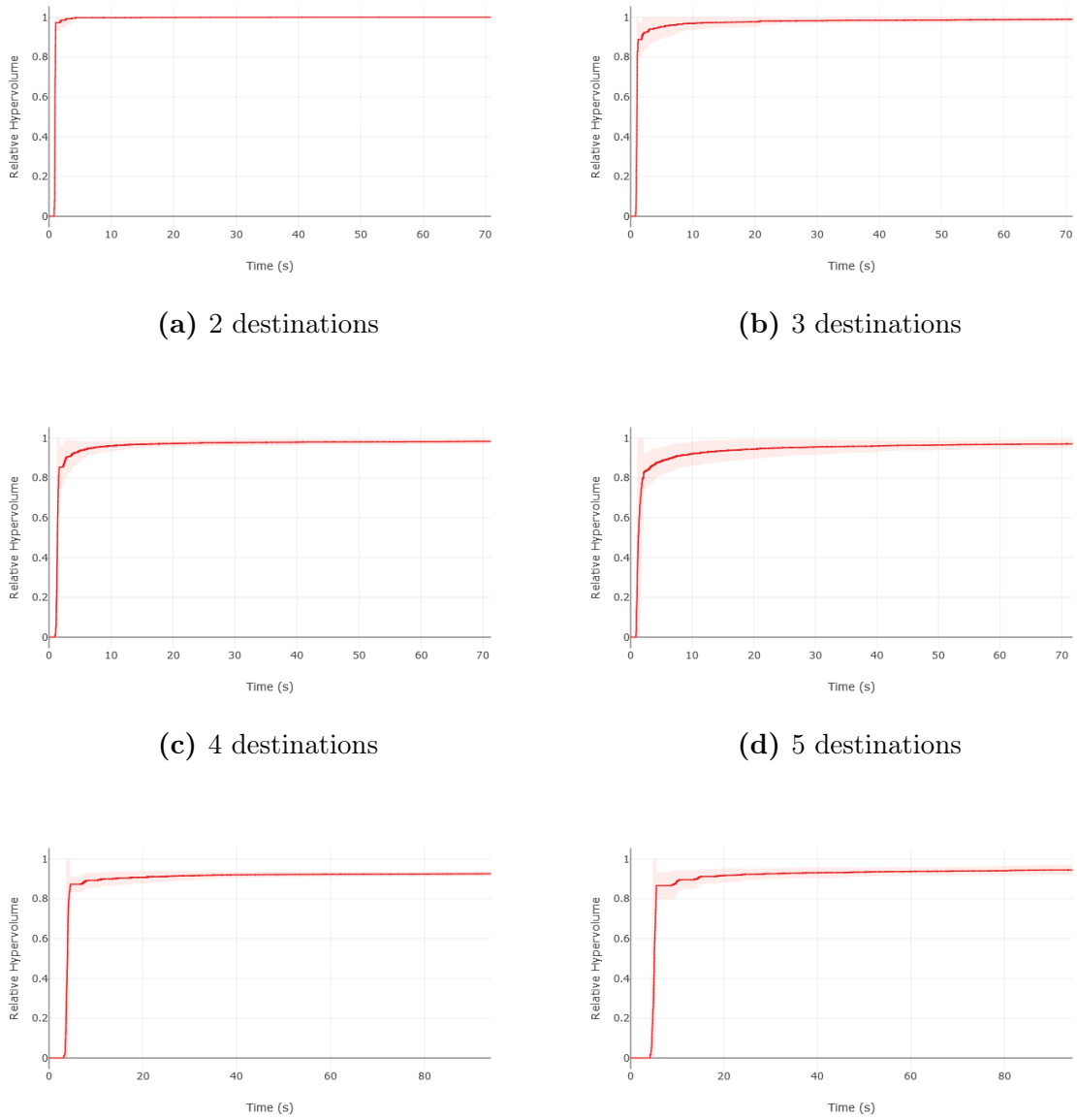
The MOEA/D-ACO method shows improvements in the convergence hypervolume compared to any of the MOACO methods variations. The statistical tests confirm that the method is better than the previous ones. It also presents a high number of iterations, a low start-up time and converges quickly.

Overall, the results show that the MOEA/D-ACO method is a strict improvement over the previous ones for the problem instances considered.

4.2.5 Local-search MOEA/D-ACO

The results for the Local-search variant of the MOEA/D-ACO method are presented in this Section. Figure 4.9 shows the relative hypervolume over time for the test problem instance and the summed up results are brought in Table 4.6. The parameters used are the same presented in 4.2.4. The Local search method selected is first-improvement strict-dominance search.

The results indicate no statistically significant difference between the convergence hypervolume of the presented method and the base MOEA/D-ACO approach. There is, however, an increase in the start-up times. Overall, the Local-search integration does not lead to improvements in the MOEA/D-ACO results for the problem instances considered.



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

(e) 6 destinations

(f) 7 destinations

Figure 4.9: Relative Hypervolume over time results for Local-search MOEA/D-ACO method on instances with 2 to 7 destinations

4.2.6 Distributed MOEA/D-ACO

Figure 4.10 presents the relative hypervolume over time for executions of the Distributed MOEA/D-ACO method, an extension of the MOEA/D-ACO described in Section 2.5.5.

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time
2	1.0000	0.0000	82.6400	0.7940	0.9714
3	0.9883	0.0078	71.7778	1.8725	0.9921
4	0.9825	0.0069	61.5538	1.3013	1.3295
5	0.9687	0.0114	66.6667	7.7230	1.2426
6	0.9230	0.0065	26.7222	0.6151	3.8642
7	0.9376	0.0103	20.2200	0.4600	4.9903

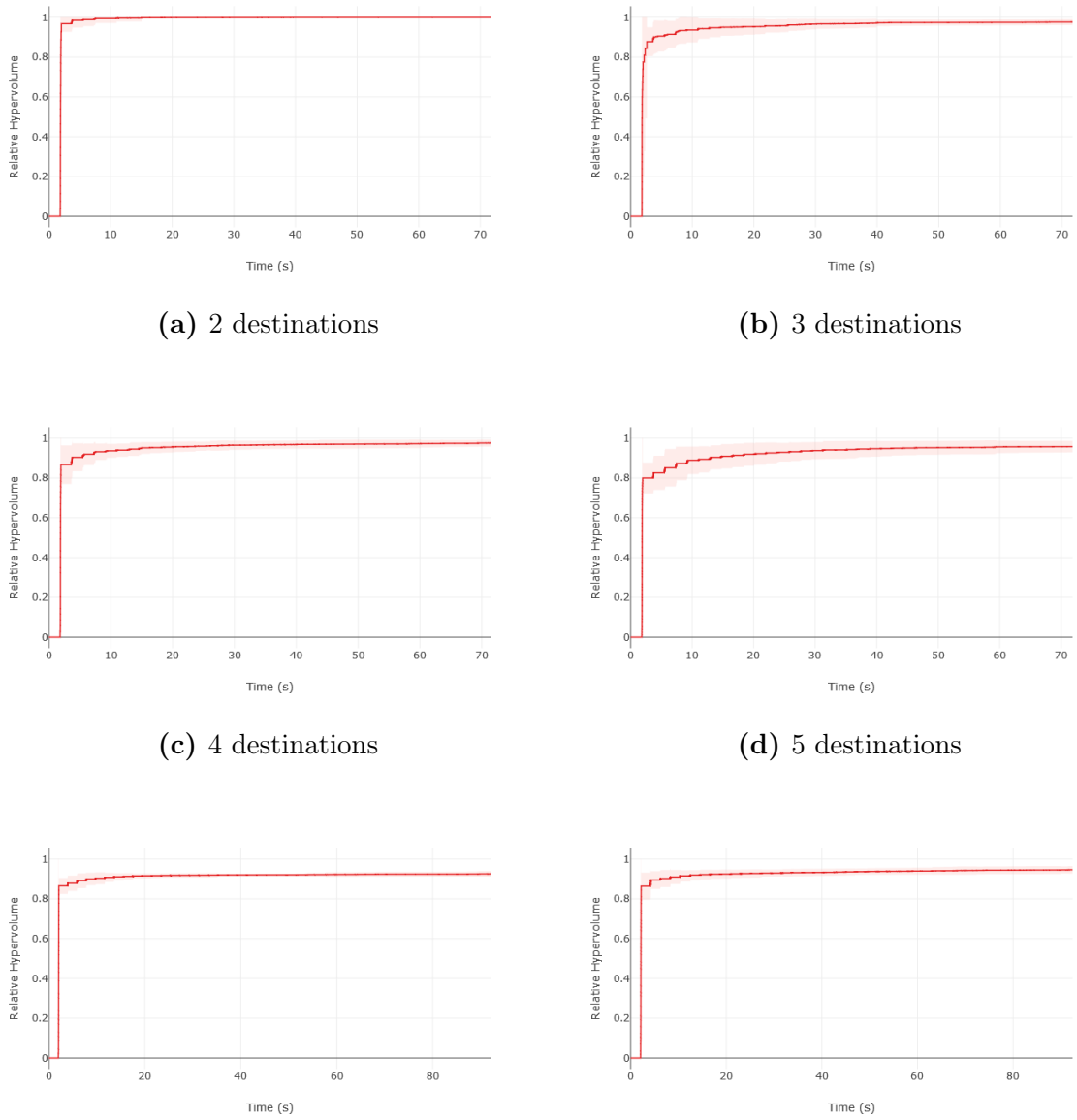
Table 4.6: Summarized results of Local-search MOEA/D-ACO experiments

These results are summed up in Table 4.7. The parameters are the same presented in 4.2.4. The number of cores used for the tasks is 6.

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time
2	0.9992	0.0012	38.520	0.4996	1.8868
3	0.9745	0.0076	38.880	0.3250	1.9355
4	0.9717	0.0108	38.600	0.4899	1.8676
5	0.9561	0.0155	38.050	0.2179	1.8776
6	0.9222	0.0058	44.875	0.3307	1.9951
7	0.9394	0.0088	40.940	0.2375	2.1385

Table 4.7: Summarized results of Distributed MOEA/D-ACO experiments

Once again, no statistically significant difference of the convergence hypervolume is noted between the Distributed MOEA/D-ACO method and the base approach. There is a small increase in the start-up times, but it does not affect the convergence times. The distributed approach brings no advantage to the MOEA/D-ACO method in the problem instances considered.



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

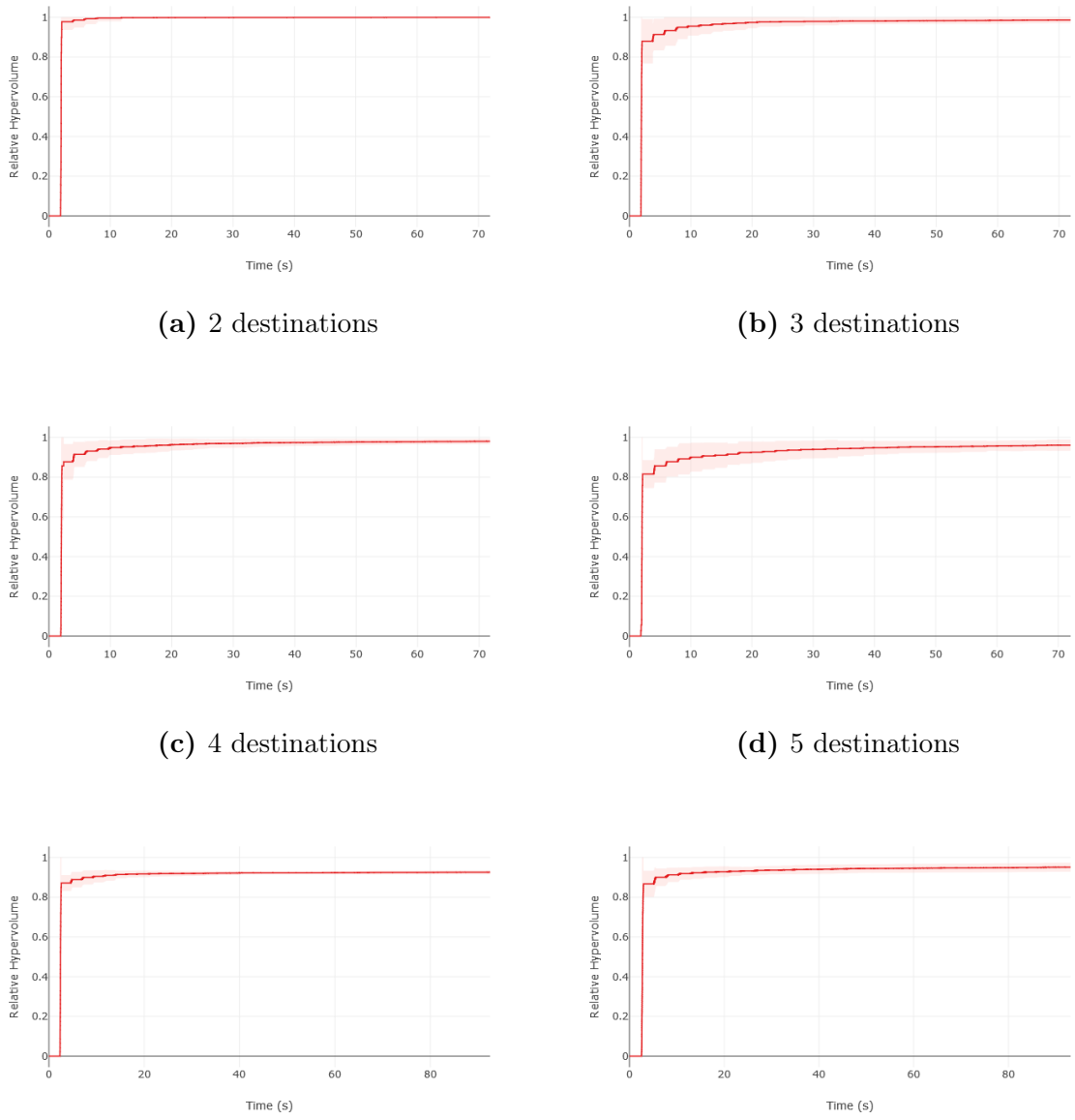
(e) 6 destinations

(f) 7 destinations

Figure 4.10: Relative Hypervolume over time results for Distributed MOEA/D-ACO method on instances with 2 to 7 destinations

4.2.7 Distributed Local-search MOEA/D-ACO

The Distributed Local-search MOEA/D-ACO experiment result summary is presented on Table 4.8. Figure 4.11 shows the relative hypervolume over time for each test problem instance. The parameters selected are the same as the ones presented on Section 4.2.6.



(a) 2 destinations

(b) 3 destinations

(c) 4 destinations

(d) 5 destinations

(e) 6 destinations

(f) 7 destinations

Figure 4.11: Relative Hypervolume over time results for Distributed Local Search MOEA/D-ACO method on instances with 2 to 7 destinations

The proposed method follows the trend of other MOEA/D-ACO extensions and brings no improvement compared to the base approach. There are no statistically significant differences on the final relative hypervolume nor on the convergence times.

Destinations	HV	HV Std	Iterations	Iterations Std	Start-up Time
2	0.9992	0.0012	38.520	0.4996	1.8868
3	0.9745	0.0076	38.880	0.3250	1.9355
4	0.9717	0.0108	38.600	0.4899	1.8676
5	0.9561	0.0155	38.050	0.2179	1.8776
6	0.9222	0.0058	44.875	0.3307	1.9951
7	0.9394	0.0088	40.940	0.2375	2.1385

Table 4.8: Summarized results of Distributed Local-search MOEA/D-ACO experiments

4.3 Results Analysis

This section presents an in-depth analysis of the results, including a study on different elements of the implemented solutions and a comparison of all the proposed methods.

4.3.1 Methods Comparison

Pairwise comparisons were performed for all methods presented. The statistical experiment shows how the methods compare to each other in terms of relative hypervolume after convergence. The experiments performed consider a confidence level of 95% and an effect size of 0.01. The results for all pairs is presented on Figure 4.12.

Based on these results, the following observations can be made considering the test problem instances:

- The MOEA/D-ACO method and all its variations are strictly better than MOACO methods in terms of relative hypervolume. The only exception is the Local-search MOACO method, which is statistically equivalent to the Distributed MOEA/D-ACO.
- All MOEA/D-ACO based methods are statistically equivalent to each other in terms of relative hypervolume. Thus, Local-search and Distributed variations were not able to improve the methods results.
- All Local-search based MOACO methods are better than the other in terms of relative hypervolume.

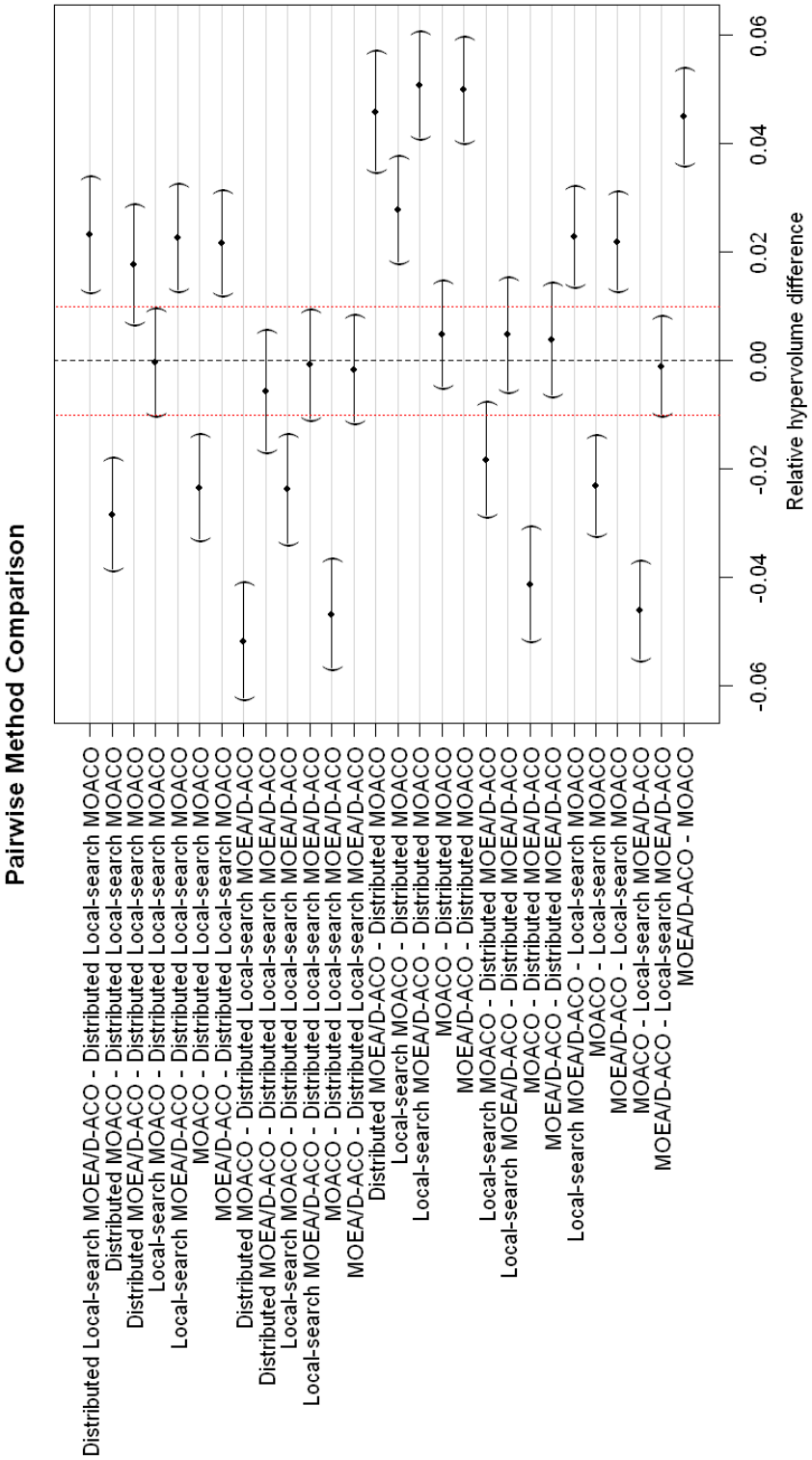


Figure 4.12: Pairwise comparison of relative hypervolume of all methods presented.

- None of the Distributed variations of the methods bring improvements in the relative hypervolume in comparison to the non-distributed ones.

Thus, in terms of relative hypervolume, the MOEA/D-ACO methods provide the best results, followed by the Local-search and Distributed Local-search MOACO. The base MOACO and Distributed MOACO are the worst of the presented methods.

The start-up time is another feature that helps differentiate the methods. As a rule, the base methods (MOACO and MOEA/D-ACO) have the smallest times, followed by the Distributed and Distributed Local-search versions. The Local-search variations have the biggest start-up times, specially for bigger problem instances.

These results suggest that, for problems with up to 7 destinations, the best method is the base MOEA/D-ACO, given the high quality of solutions obtained and the lower start-up times. This result indicates that the decomposition approach applied to ACO methods can lead to better and more diverse solution sets in the Trip Itinerary Planning Problem.

4.3.2 Local-search Effect

As discussed on Session 4.3.1, the Local-search can improve the base MOACO methods, although it does not help MOEA/D-ACO. This result is better illustrated in Figure 4.13.

It is also noted that the Local-search methods have a much smaller number of iterations and higher start-up times, as seen in Figure 4.14.

This is the result of the added computational cost of performing a local search after each solution is generated. As discussed on Section 3.3.2, this search is non-trivial, as it needs to check all restrictions in order to guarantee that only feasible solutions are generated. Since every iteration takes more CPU time, the consequence is a smaller number of iterations.

The solution construction process saves all queries results in memory, in order to improve processing times. As a consequence, the computational cost of solution construction is bigger in the first iteration and gets smaller in subsequent ones. This also contributes to higher start-up times on Local-search methods.

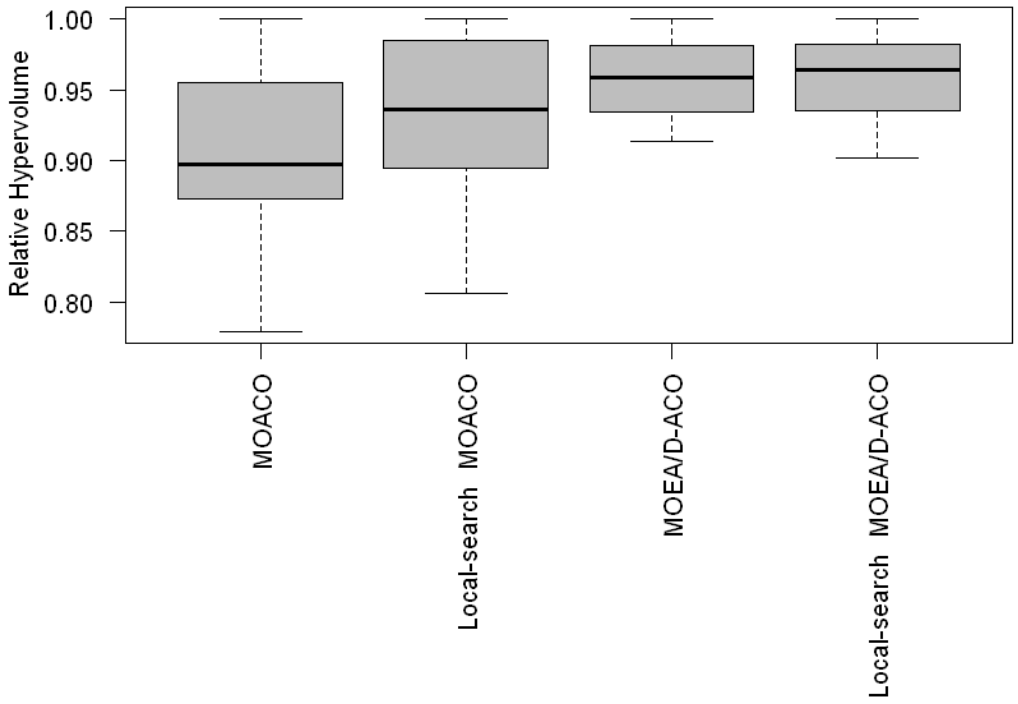


Figure 4.13: Local-search effect on relative hypervolume for MOACO and MOEA/D-ACO methods

Even with higher start-up times and lower iterations, the MOACO method hypervolume improves with Local-search. This indicates that it accelerates the convergence of the algorithm and leads to better solutions in fewer iterations.

This phenomenon does not repeat on MOEA/D-ACO methods. The decomposition based method already generates good solutions without Local-search. In this case, the advantages of the Local-search are likely lost due to the lower number of iterations, leading to an equivalent result.

4.3.3 Distributed Implementation Effect

For all methods, the results show that distributed approaches do not lead to better hypervolumes. This is further illustrated by Figure 4.15, which shows the relative hypervolume box-plot for distributed and base approaches of MOACO and MOEA/D-ACO methods.

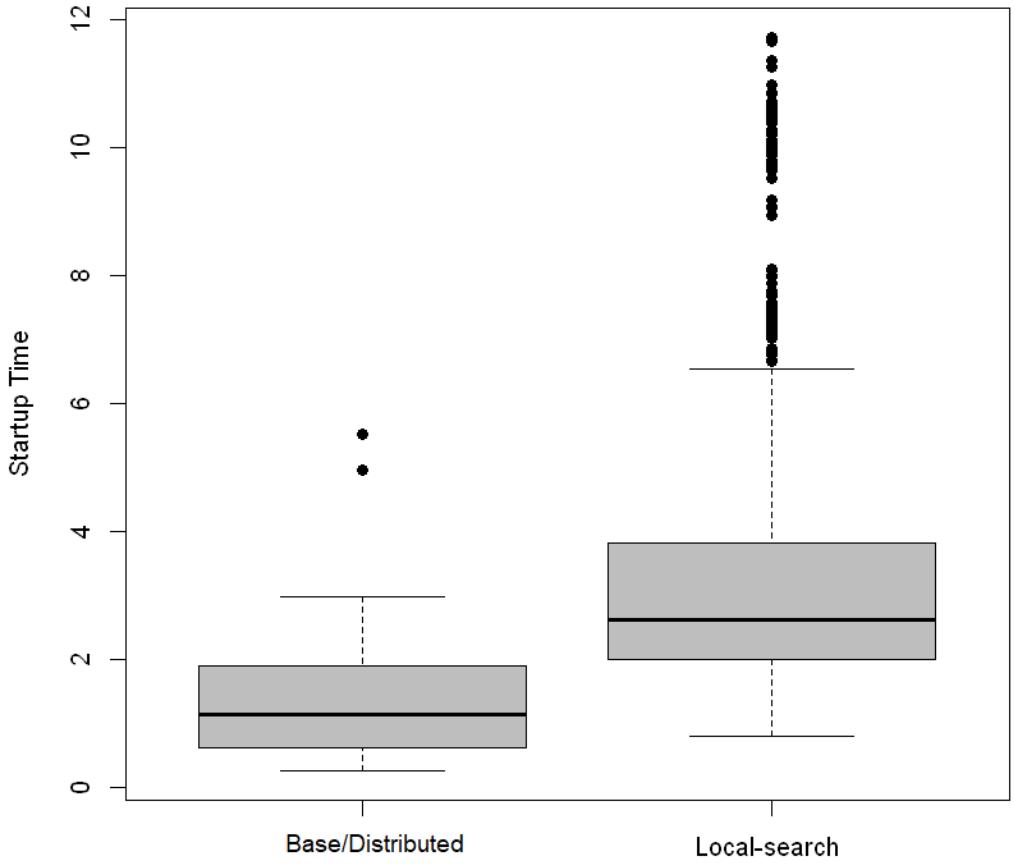


Figure 4.14: Local-search effect on start-up times.

This result contradicts the initial expectations. The distributed approaches do not change the algorithmic approach. Instead, they simply split the workload in concurrent processes that run on different CPU cores, making better use of the computational resources available. Thus, this approach should improve the performance based on the number of cores.

It is important to notice that there is a small overhead to this approach, due to the need to coordinate this distribution and wait for all processes to finish at junction points. In this case, these junction points occur after every iteration. However, this overhead is not enough to explain why the distributed approach did not improve the results.

In order to do so, it is necessary to consider the environment and language-specific characteristics. The Python language uses a Global Interpreter Lock (GIL) that prevents multiple threads from executing code at once. As a consequence, it is not possible to implement parallel multi-threaded applications in Python.

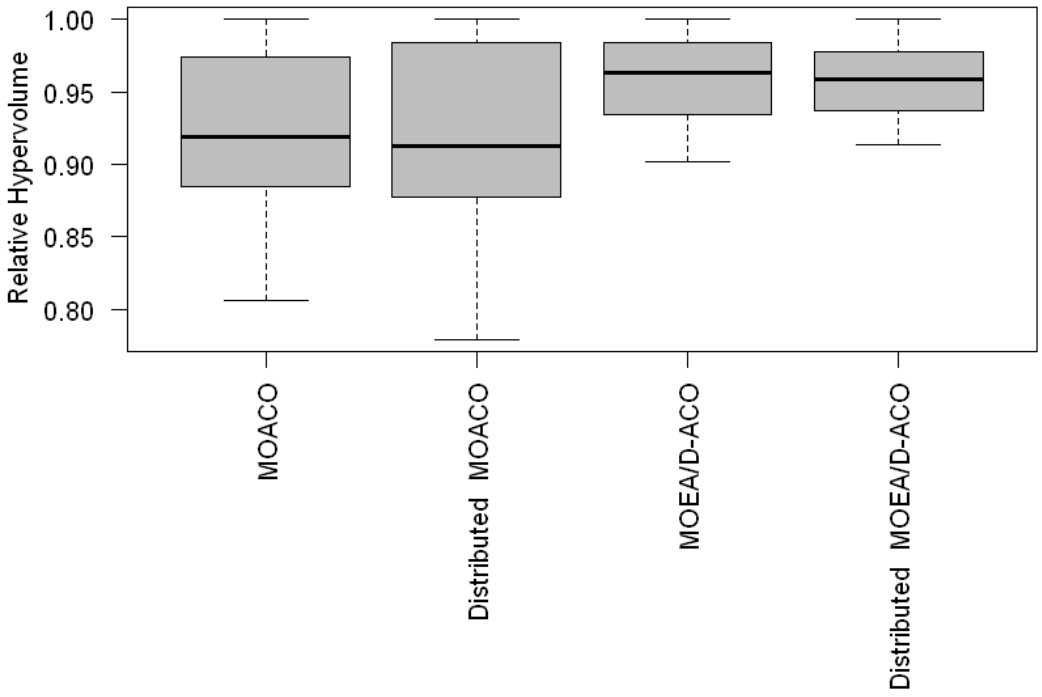


Figure 4.15: Distributed approach effect on hypervolumes of MOACO and MOEA/D-ACO methods.

Thus, all parallelization in Python must be achieved via Multiprocessing. This way, each process has its own interpreter. There is a big downside to this approach, however. Since multiple processes do not have shared memory, all the necessary data must be copied between the processes every time.

In the Trip Itinerary Optimization context, this has been proved problematic. The solution construction process is a relatively fast operation, but it requires a lot of data: all the trip specifications, components information and optimization methods. Every time the operation is distributed between processes, all this data must be copied. This increases both the necessary memory and processing power required. This operation happens several times per iteration, based on the number of ants. Thus, the data copying cost offsets the computational gains from the distributed approach.

This is an unfortunate consequence of the developed approach and the Python language limitations. In future works, this problem could be eliminated by using another language for the optimization process implementation.

It is interesting to note that, when the distributed process is more computationally intensive, the cost of copying data is mitigated. This can be seen comparing distributed and regular Local-search approaches. The distributed ones have smaller start-up times, as seen in Figure 4.16. This happens due to the increased cost of the Local-search process in the solution construction.

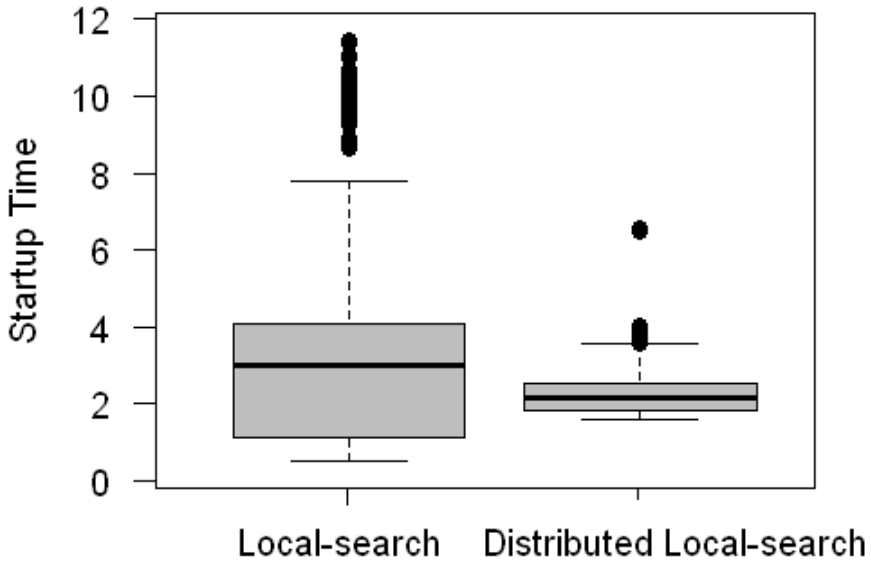


Figure 4.16: Distributed approach effect on start-up times on Local-search methods.

4.3.4 Number of Groups on MOEA/D-ACO

The number of groups is an important parameter of the MOEA/D-ACO method. The number of pheromone matrices is determined by it, since a single pheromone matrix is used for each group. These shared pheromone matrices is a way an ant exchanges information with other ants and contributes to their search.

However, the method also introduces many other new features to the base MOACO method, such as: objective decomposition; best solutions contribution to the construction process; pheromone limits; neighborhood information exchange and best component selection probability.

In order to determine how important the number of groups is, an experiment on the parameter effect was performed. To do so, all test problem instances were solved using

the base MOEA/D-ACO method with different group sizes, and the relative hypervolume evaluated. The summed-up results are presented in Figure 4.17.

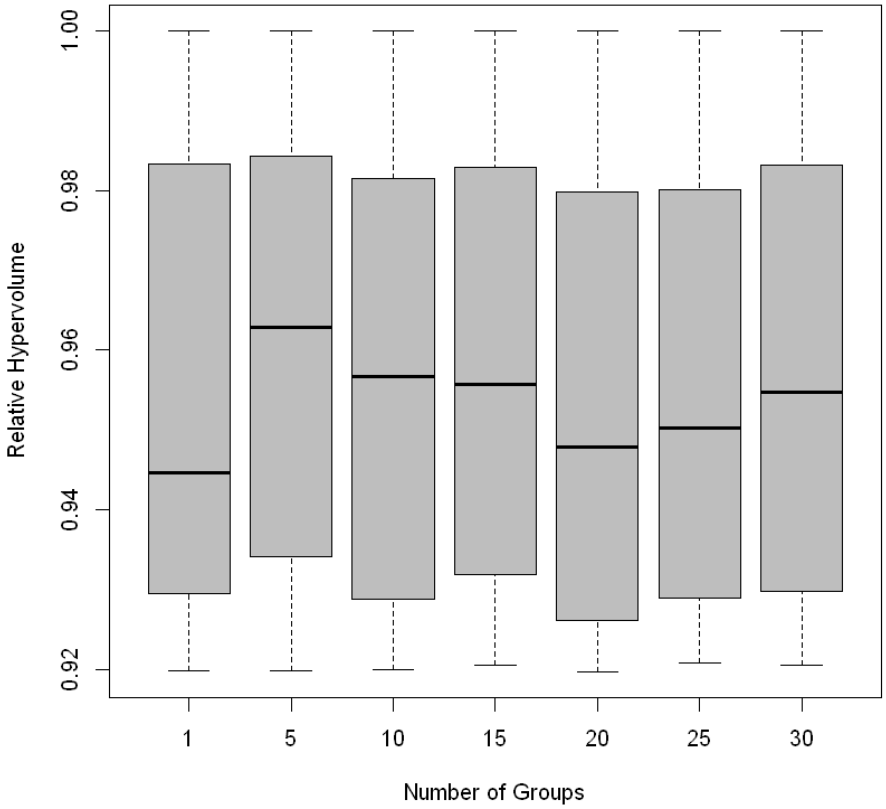


Figure 4.17: Relative hypervolume distribution for MOEA/D-ACO with different number of groups.

The result indicates that there is no significant difference in the hypervolume distribution due to number of groups in the MOEA/D-ACO method for the Trip Itinerary Planning Problem. Statistical tests performed with a significance level of 0.05 confirm this conclusion.

Based on the results, it is possible to conclude that the optimization advantages of the MOEA/D-ACO method on the problem are not consequence of the separation of ants in groups. It is possible that this factor is more important for problems with higher dimensionality, though.

4.4 Trip Size Analysis

The number of destinations on an itinerary is the most influential factor on the search space size, as discussed in Section 3.1.5. Therefore, the size of the trip has a big impact on the instance complexity. The growth in the number of destinations eventually leads to a combinatorial explosion. It is, therefore, the main limitation for the Trip Itinerary Planning Problem.

The complete search space was evaluated for the test instances with up to 4 destinations. Figure 4.18 shows the incredible growth with the increase on the number of destinations. Instances with more than 4 destinations were not included because the exhaustive search space evaluation is unfeasible for bigger problems.

This shows that the problems get harder to solve as they grow, until they cannot be solved in feasible time. This can be observed in the test problems. As the size of the problem grows, the convergence time increases and the quality of the solutions found decreases. Figures 4.19 shows the decrease in the quality with problem growth.

It is interesting to note that there is an outlier for this problem. For all MOEA/D-ACO methods, the relative hypervolume for the 6 destination problem instance is worse than the 7 destination counterpart, as shown in Figure 4.10. The most likely explanation for this is that the method and parameters chosen are very well adjusted to the test instance with 7 destinations (or badly adjusted for the 6 destination instance). A larger number of tests on different instances would probably change this, but were not performed due to time limitations.

The combinatorial explosion limits the maximum size that can be solved by the proposed methods in reasonable time. All the tests were performed with up to 7 destinations due to the high cost of calculating the approximated Pareto Front. For 8 or more destinations, it would take a very long time to obtain a good estimated front. Thus, it is not possible to determine how well the methods work with problems with 8 or more destinations.

However, the results with up to 7 destinations indicate that they work well for the problem. Even with worse relative hypervolumes in comparison to the optimal front, it is possible that the methods would provide better solutions than current tools for such situations.

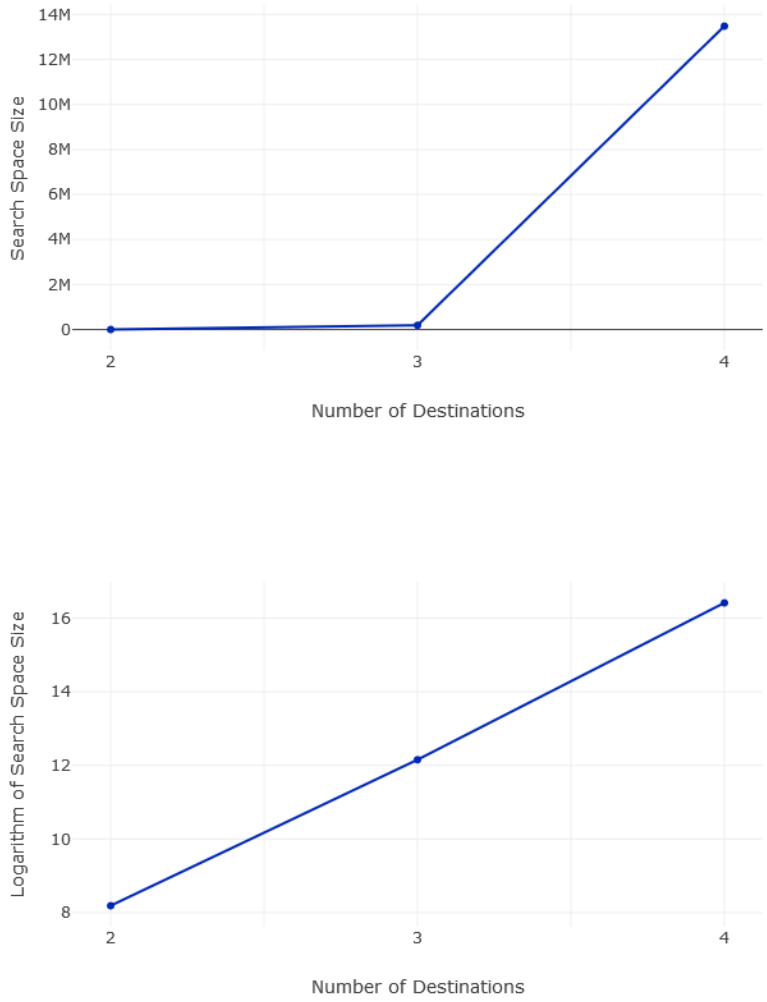


Figure 4.18: Search space size for problems with 2, 3 and 4 destinations.

Furthermore, itineraries with more than 7 destinations are quite rare. Thus, the proposed method works very well for the majority of the real world itinerary planning problems.

4.5 Restrictions Effects and Discussion

In the Trip Itinerary Planning Problem, the size of an instance is primarily determined by the number of destinations on the itinerary, as discussed in Section 4.4. Other restrictions,

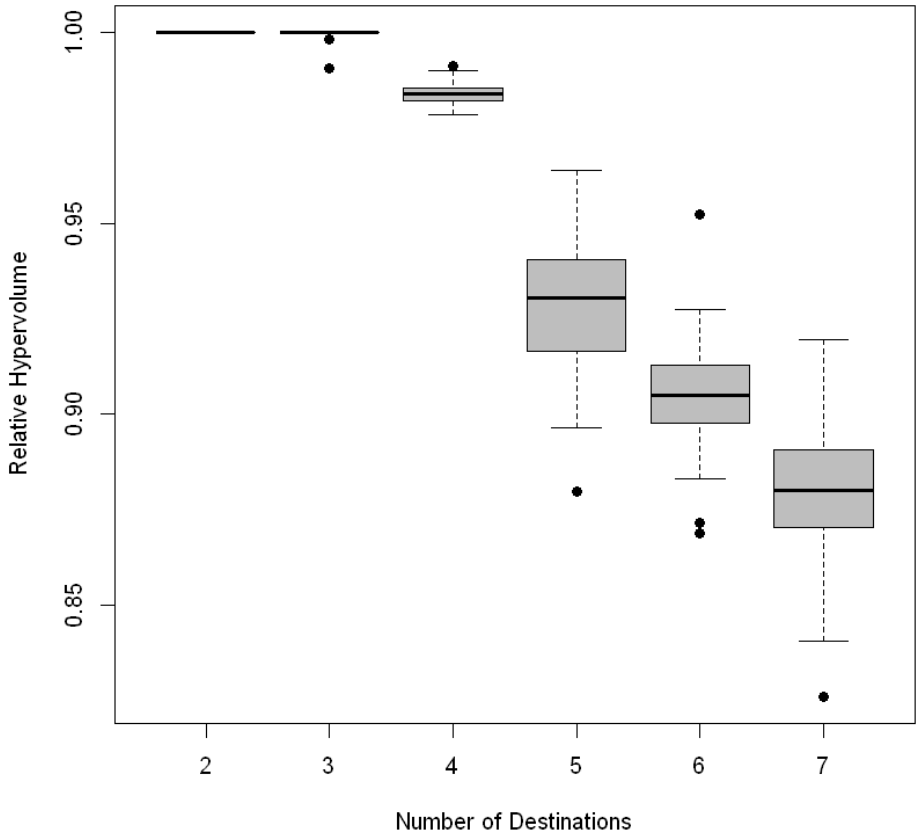


Figure 4.19: Relative hypervolume distribution for Distributed Local-search MOACO with different number of destinations.

however, also have a big influence on the instance complexity. This section provides a brief analysis on these restrictions effects.

The first one to consider is flexibility on dates. This is present both on variations on the trip's start and end dates and on the stay length defined for each destination. The higher the range of accepted dates for departure or arrival, the more flexible the trip is. The same is true for more variations on stay lengths for each destination. This leads to an increase in the complexity, due to the higher number of feasible solutions.

The order of the destinations is also an important factor to the problem complexity. Itineraries with unordered destinations have more solutions options and a bigger search space. Defining the order of some or all destinations as a restriction can greatly reduce the problem complexity.

A final important factor is the number of transportation or accommodation options for each branch of the trip. More options increase the problem complexity. Even though this

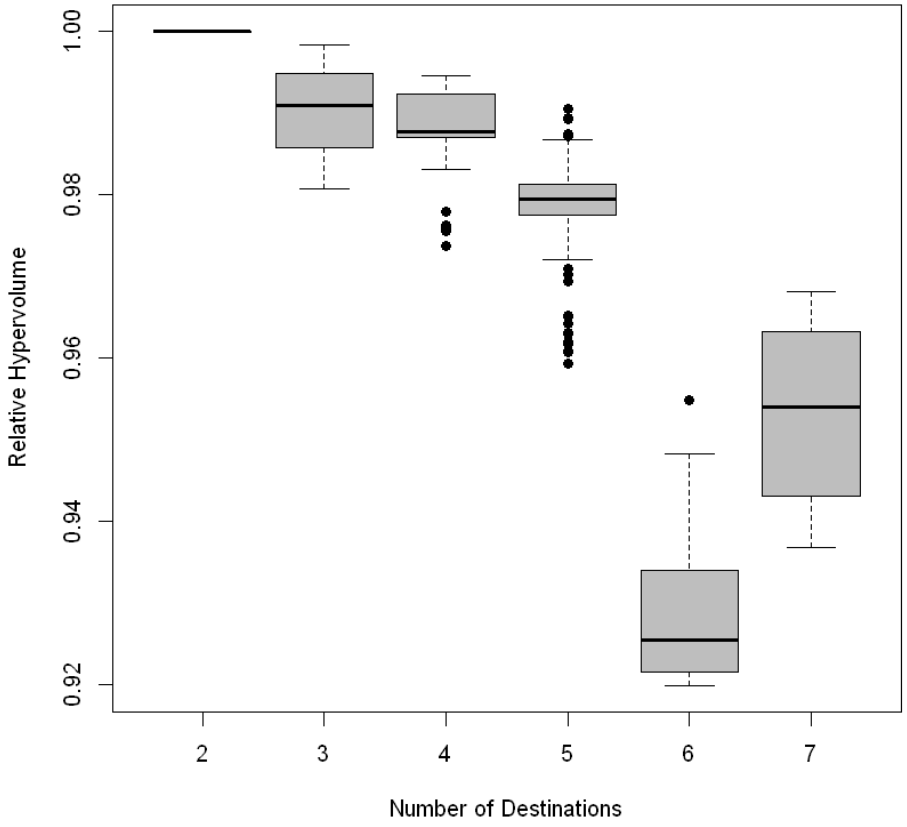


Figure 4.20: Relative hypervolume distribution for MOEA/D-ACO with different number of destinations.

is an uncontrollable factor, this number is reduced when transportation or accommodation restrictions are defined. Thus, it is important to also consider the average number of options as well.

Figure 4.21 illustrates how different restrictions affect the search space size. In order to do so, the complete search space size was calculated for problems with different restrictions in stay lengths, destination order and average number of components at each step. The test problem instances with unordered destinations, stay length of 1 day for all destinations and an average number of 6 components at each step is also included as reference.

The results indicate that the restrictions greatly reduce the problem size. The higher the number of destinations, the greater the effect. This results indicate that the introduction of restrictions is a possible way to solve bigger problems instances in reasonable time.

4.6 Summary

The results indicate that the proposed methods work very well for itinerary planning problems with up to 7 destinations. The best methods converge in up to one minute even for the bigger problems, and obtain solution sets very close to the optimal estimated Pareto Front.

The tested MOEA/D-ACO method is verified to be a strict improvement over the traditional MOACO method, though the division of ants in groups does not play a prominent role in this improvement in the considered problem. The Local-search improvement helps the traditional MOACO method, but is not so effective in the MOEA/D-ACO approach. The distributed implementation does not improve the quality in either approach, although this is likely due to programming language limitations and implementation.

The trip size is confirmed as the most limiting characteristic of the problem, eventually leading to a combinatorial explosion as it grows. Other problem restrictions, such as destinations ordering, stay length on destinations and accommodation and transportations restrictions can be used a way to reduce the problem complexity growth.

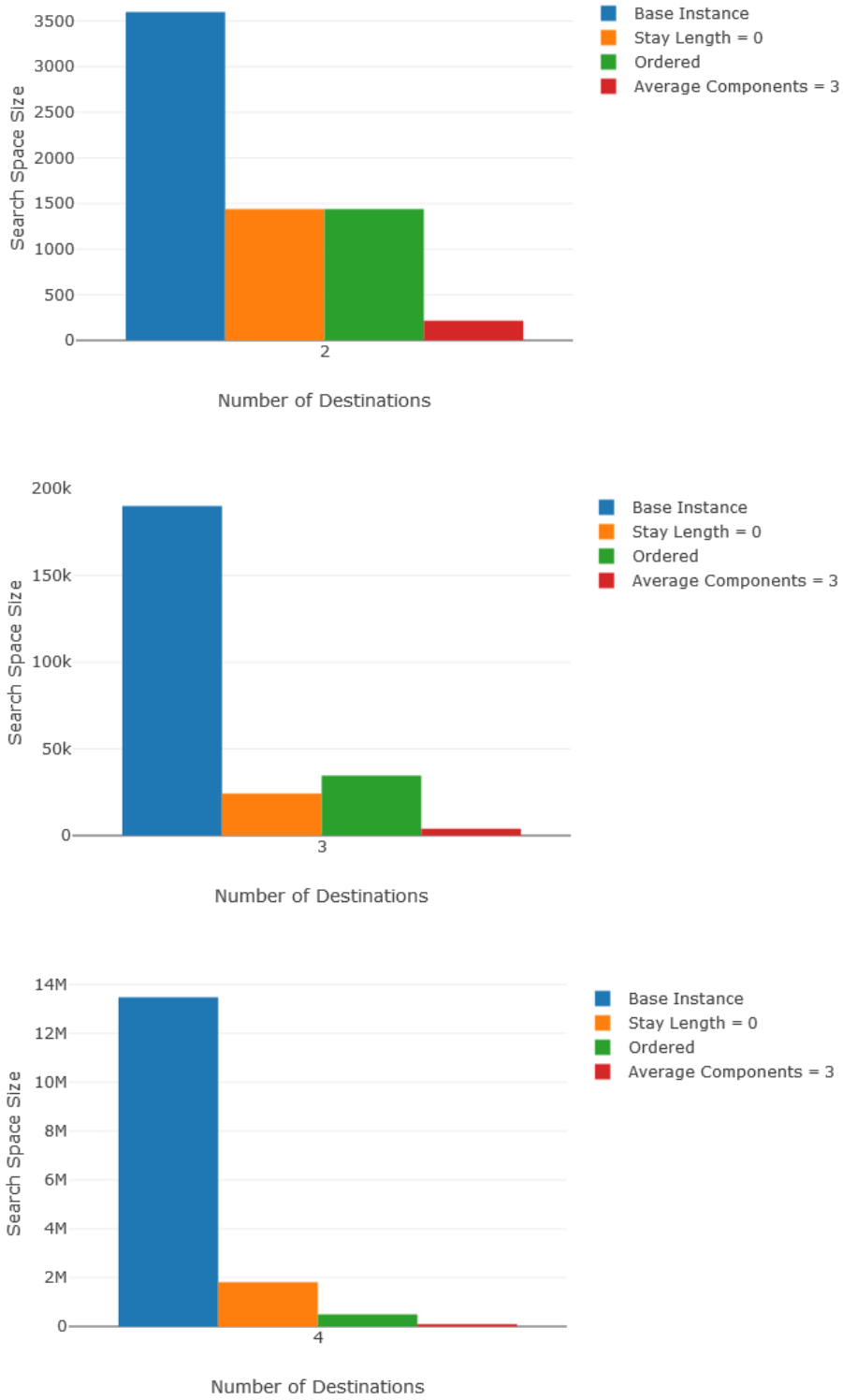


Figure 4.21: Restrictions effects on search space size for problems with 2, 3 and 4 destinations.

Chapter 5

Conclusion

This Chapter presents a final analysis of the Trip Itinerary Planning Problem and the proposed optimization methods. It covers the work contributions and applicability in real world situations, current limitations and future work proposals.

5.1 Contributions and Applicability

5.1.1 Trip Itinerary Planning System

This work presented and discussed the Trip Itinerary Planning Problem as an application and extension of the Traveling Salesman Problem with Time Windows with constraints and proposes multiobjective optimization methods to solve it, minimizing cost and travel time. In addition, a data gathering and pre-processing framework is presented, as well as a decision support method. These elements together compose a system that can help users solve trip planning problems and find the best flights and accommodations for their trips, according to their preferences.

The main characteristics of the system as a trip planning tool are:

- Flexibility on trip itinerary definitions: users can define a range of acceptable departure and arrival dates, the desired stay length range for each destination, the desired destination order (if any) and accommodations and flights restrictions.
- Data gathering system that automatically determines all the information needed based on the itinerary definitions.

- Data pre-processing system that eliminates sub-optimal flights and hotels from the options gathered, reducing the search space.
- Optimization method that can determine near-optimal solutions for trips with up to 7 destinations in up to one minute.
- Decision support system that ranks solutions based on the user's preferences and can be easily tuned.

Currently there are no other known trip planning tools that allows so much flexibility as the proposed solution. This increased flexibility allows users to define exactly what they want for the trip and perform a single search. It also leads to a bigger search space, in which better solutions might be found. Finally, once the itinerary definitions are set, the system is able to quickly optimize the problem and provide the user with a large number of high quality solutions.

This way, the proposed solution might represent a big improvement to the trip planning experience. Since it evaluates many more possible trip solutions than a user could do manually, it can find solutions with better quality. The speed and ease of use also helps making the task of searching and buying flights tickets and booking hotels much easier. Finally, the user has full control over every step of the process, from the definition of every detail in the itinerary to the choice of the final selected solution, which preserves the user's autonomy.

In short, the system offers better results with less time and effort from the user, providing a much better trip planning experience. This can be a great competitive advantage for any flight and/or hotel e-commerce, and bring progress and improvements to the online tourism market.

5.1.2 Ant Colony Optimization Application

In addition to the practical contributions, this work also explores the application of a classic literature optimization method to the real world Trip Planning Problem, as well as extensions of this method.

The Ant Colony Optimization method is one of the main techniques for combinatorial optimization problems, as well as its extension for multiobjective problems, the Multi-objective Ant Colony Optimization (MOACO). The use of Local Search and Distributed extensions are also established in the literature. The MOEA/D-ACO method is a newer method with promising results for multiobjective problems.

The results of the methods applications to the Trip Itinerary Planning Problem reinforces their applicability to complex combinatorial optimization problems. The use of Local-search extensions for the MOACO method led to improvements, which shows the benefit of these hybrid approaches. The MOEA/D-ACO was strictly better than the base MOACO method for the problem, which suggests that the changes proposed in the method are beneficial and can improve the optimization results. It is noted, however, that not all elements introduced in this new approach are necessary for this improvement. This is the case for the ant grouping in the proposed problem.

Since the implementation of distributed approaches was hindered by language limitations, it was not possible to evaluate how well they would contribute to the presented methods. It is noted, however, that these approaches work better with fewer and more computationally costly processes than with many cheap ones.

5.1.3 Limitations

The Trip Itinerary Planning Problem suffers from combinatorial explosion as the number of destinations in the itinerary grows, which leads to problems so large that they cannot be treated. This work only presents experiments with problems with up to 7 destinations, as estimating the optimal Pareto Solutions for bigger problems was too costly given the experimental design and the time and hardware constraints.

Given these limitations, it is not possible to know how well the proposed methods work for bigger problems. Given the complexity of these problems, however, it is reasonable to expect that the methods should work better than manual planning.

In addition, it was verified that restrictions can be used to greatly reduce the problem complexity. This way, bigger problems can be solved in reasonable time with the use of restrictions, such as destinations order. This should allow itineraries of up to 9 or 10 destinations to be solved in reasonable time with reasonable quality, which covers most of an average traveler needs. The use of restrictions, however, only delays the combinatorial explosion, and is not a solution for every problem.

The system is mostly suited to average traveler use cases for personal trips, focusing on obtained good solutions for small and medium trips in a short time. A quick solution is very important for the typical user on the web. In such cases, a large amount of users might use the system, which require that each trip planning does not put a big strain on a server.

Despite this, the trip planning solution might also be useful for other scenarios, such as business. A company that requires its employees to travel a lot might need solutions that work for bigger itineraries. In such cases, however, the need for such short optimization times is smaller. Thus, bigger itineraries could also be accommodated with bigger optimization times and better processing power. In such cases, a better distributed optimization method could work very well.

Finally, it is important to note that the language used for the project development, Python, is not among the best in terms of performance. Even with recent developments and code optimization, it falls behind other options, such as C, C# or even Julia. The implemented method works well as a proof of concept and solves most of the use cases proposed, being a good solution for the Trip Itinerary Planning Problem. Nevertheless, it could benefit of a re-implementation using a performance-focused language.

5.2 Future Work Proposals

The proposed method solves the Trip Itinerary Planning Problem and achieves the established objectives. While it demonstrates that the problem can be solved, it is still not a complete product. Besides, the results achieved suggest that improvements can still be made to the method.

This section covers possible improvements and additions that can either help shape the proposed solution into a complete product for the end user or continue the development of the optimization method and the technical investigations performed, improving the results found.

5.2.1 Data Gathering

The one thing preventing the developed Trip Itinerary Planning System to be used in real world situations is access to the necessary data. As discussed before, real-time access to this data is hard and expensive. Even so, it is a key point for a finished product.

There are two main ways to get the necessary data: either buy it from a provider, or arrange a partnership with another company that already has access to this information. Both are valid options, and the choice depends primarily on the available resources and opportunities.

Either way, it is important that this data is up to date with hotels and airlines and is available to be consulted without access limitations. Access to complete and correct information is key for any optimization system.

5.2.2 User Experience and Interface

Any system intended for final users and widespread access need to focus on user experience. The user interface plays a paramount part in this.

In order to interact with the system and plan their trips in a satisfactory way, users need an interface that is easy to use and understand, and yet conveys all necessary information and allows full control. Among other elements, this interface should have:

- An itinerary definition page in which users can quickly define the important parts of their trip, such as destinations and dates. This page should also provide advanced options for tweaking details such as dates flexibility, destinations orders and hotels and flights restrictions.
- A solution selection page in which the user views all optimization results for their trip and selects the desired solution. The solutions should be ranked according to decision support methods and the user should be able to quickly tweak their preferences. All proposed itineraries should be accompanied by links that direct the user to the service provider, so they can buy the tickets and make hotel reservations.
- A trip visualization page showing the selected trip itinerary. This page should highlight all important information and provide ways to check the details as well. Interactive and visual representations, such as maps, would greatly benefit the user experience. The user should also be able to return to previous pages and change the trip definitions and chosen solution, if necessary.

Other non-essential features that could benefit the user experience are social features, such as itinerary sharing, team planning and comments; profile information and travel history; and links with useful information about the chosen destinations and recommendations, among others.

This interface should be developed as a website, hosted and made available to the public. The website format is the best in order to provide access to a wider audience. In the future, it can also be adapted to phone apps.

5.2.3 Optimization Improvements

Complexity Reduction Methods

The results presented show that restrictions can be used to greatly reduce the complexity of a problem instance. Therefore, restrictions can be used to simplify very large problems and help their solution.

While the restrictions are usually defined by users, they can also be automatically introduced when a very large problem instance is detected. A pre-optimization data analysis can help choosing good restrictions that are more likely to remove bad solutions and keep the better ones.

A possible way to do this is using Monte-Carlo simulations to sample some solutions and, based on these results, determine destinations orders that usually lead to bad solutions. These orders would then be excluded via restrictions before the optimization process.

It is likely that some destination orders very often lead to bad solutions. This is the case of destination orders that increase the total distance covered on all flights, since the distance directly impacts on travel times and costs.

Distributed Implementation

The results indicate that the distributed implementations of the proposed methods did not lead to better results. However, as discussed, this result is very likely consequence of language limitations.

A proper distributed implementation allows the method to make full use of the hardware available and greatly increase the algorithms convergence time. In addition, distributed implementations can be used to reduce the time difference between itineraries of different sizes and reduce variation in user wait time. This can be done assigning more CPU cores to complex problem instances, and keeping smaller instances bound to a single core.

The proposed distributed implementation can be done using a language more appropriate to parallelization.

Algorithm Improvements

The results indicate that some elements of the MOEA/D-ACO (such as the number of ant groups) algorithm are not impactful in the final result. These elements might increase the computational cost of the method without any meaningful contribution.

Statistical experiments can be performed assessing the impact of every element of the heuristic. Should more elements with null impact be identified, they can be removed from the method. This would lead to a striped down version of the method that keeps all important parts and would probably have a better performance.

In addition to this, hyperparameter optimization methods could be used to improve the selected heuristic parameter, which could lead to better results.

References

[1] O. Ali and D. VanOudheusden. Logistics planning for agricultural vehicles. In *2009 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 311–314. IEEE, dec 2009.

[2] Daniel Angus and Clinton Woodward. Multiple objective ant colony optimisation. *Swarm Intelligence*, 3(1):69–85, 2009.

[3] Jinling Bao, Xiaochun Yang, Bin Wang, and Jiaying Wang. An Efficient Trip Planning Algorithm under Constraints. *2013 10th Web Information System and Application Conference*, 1:429–434, 2013.

[4] Carmen Berne, Margarita Garcia-Gonzalez, and Jose Mugica. How ICT shifts the power balance of tourism distribution channels. *Tourism Management*, 33(1):205–214, 2012.

[5] Christian Blum. Hybrid metaheuristics in combinatorial optimization: A tutorial. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7505 LNCS(6):1–10, 2012.

[6] Julio Brito, Airam Expósito-Márquez, and José A Moreno. A fuzzy GRASP algorithm for solving a Tourist Trip Design Problem. In *IEEE International Conference on Fuzzy Systems*, 2017.

[7] Dimitrios Buhalis. eAirlines: strategic and tactical use of ICTs in the airline industry. *Information & Management*, 41(7):805–825, 2004.

[8] Dimitrios Buhalis and Rob Law. Progress in information technology and tourism management: 20 years on and 10 years after the Internet - The state of eTourism research. *Tourism Management*, 29(4):609–623, 2008.

[9] Bernd Bullnheimer, Richard F Hart, and Christine Straub. A New Rank-Based Version of the Ant System: A Computational Study. *Central European Journal of*

Operations Research and Economics, 7(1):25 – 38, 1999.

[10] Roberto Wolfler Calvo. A New Heuristic for the Traveling Salesman Problem with Time Windows. *Transportation Science*, 34(1):113–124, 2000.

[11] B. Chakraborty, T. Maeda, and G. Chakraborty. Multiobjective route selection for car navigation system using genetic algorithm. In *Proceedings of the 2005 IEEE Midnight-Summer Workshop on Soft Computing in Industrial Applications, 2005. SMCia/05.*, pages 190–195. IEEE, 2005.

[12] Chonlatis Charoenwong and Somchai Pathomsiri. Vehicle routing for improving financial performance: A case study of a freight transportation service provider in Thailand. In *2015 4th International Conference on Advanced Logistics and Transport (ICALT)*, pages 263–268. IEEE, may 2015.

[13] Carlos a Coello Coello, Gary B. Lamont, and David a. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer, 2007.

[14] Rodrigo Ferreira Da Silva and Sebastin Urrutia. A General VNS heuristic for the traveling salesman problem with time windows. *Discrete Optimization*, 7(4):203–211, 2010.

[15] Kalyanmoy Deb. Multi-objective optimization using evolutionary algorithms: an introduction. *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 1–24, 2011.

[16] Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and François Soumis. Network Routing. *Handbooks in Operations Research and Management Science*, 8:35–139, 1995.

[17] A. Divsalar, P. Vansteenwegen, and D. Cattrysse. A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145(1):150–160, 2013.

[18] Soufiene Djahel and John Murphy. A comparative study of vehicles' routing algorithms for route planning in smart cities. In *2012 First International Workshop on Vehicular Traffic Management for Smart Cities (VTM)*, pages 1–6. IEEE, nov 2012.

[19] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278, 2005.

[20] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1):29–41, 1996.

[21] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. Bradford Company, 2004.

[22] A E Eiben and S K Smit. Evolutionary Algorithm Parameters and Methods to Tune Them. In *Autonomous search*, pages 15–36. Springer, 2011.

[23] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2008.

[24] A.E. Eiben and S.K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[25] Filippo Focacci, Michela Milano, and Andrea Lodi. Solving TSP with time windows with constraints. *Proceedings of the 1999 international conference on Logic programming*, pages 515–529, 1999.

[26] Carlos M Fonseca and Peter J Fleming. On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In *International Conference on Parallel Problem Solving from Nature*, pages 584—-593. Springer, 1996.

[27] A. Garcia, P. Vansteenwegen, W. Souffriau, O. Arbelaitz, and M.T. Linaza. Solving Multi Constrained Team orienteering Problems to Generate Tourist Routes, 2009.

[28] C. García-Martínez, O. Cordón, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, 180(1):116–148, 2007.

[29] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, Grammati Pantziou, and Nikolaos Vathis. Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers & Operations Research*, 62:36–50, 2015.

[30] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*, volume 2. Springer, 2010.

[31] D. Gilbert and J. Abdullah. A study of the impact of the expectation of a holiday on an individual's sense of well-being. *Journal of Vacation Marketing*, 8(4):352–361, 2002.

[32] David E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[33] Mhand Hifi and Lei Wu. A hybrid metaheuristic for the Vehicle Routing Problem with Time Windows. In *2014 International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 188–194. IEEE, nov 2014.

[34] Aleksandar Jevtic, Diego Andina, Aldo Jaimes, Jose Gomez, and Mo Jamshidi. Unmanned Aerial Vehicle route optimization using ant system algorithm. In *2010 5th International Conference on System of Systems Engineering*, pages 1–6. IEEE, jun 2010.

[35] Korhan Karabulut and M. Fatih Tasgetiren. A discrete artificial bee colony algorithm for the traveling salesman problem with time windows. *2012 IEEE Congress on Evolutionary Computation*, pages 1–7, 2012.

[36] L. Ke, Q. Zhang, and R. Battiti. MOEA/D-ACO: A Multiobjective Evolutionary Algorithm Using Decomposition and AntColony. *IEEE Transactions on Cybernetics*, 43(6):1845–1859, 2013.

[37] Kian Sheng Lim, Salinda Buyamin, Anita Ahmad, Mohd Ibrahim Shapiai, Faradila Naim, Marizan Mubin, and Dong Hwa Kim. Improving Vector Evaluated Particle Swarm Optimisation Using Multiple Nondominated Leaders. *The Scientific World Journal*, 2014, 2014.

[38] Manuel López-Ibáñez and Christian Blum. Beam-ACO for the travelling salesman problem with time windows. *Computers and Operations Research*, 37(9):1570–1583, 2010.

[39] Thibaut Lust and Jacques Teghem. Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510, jun 2010.

[40] Nuri Ozalp, Ozgur Koray Sahingoz, and Ugur Ayan. Autonomous multi unmanned aerial vehicles path planning on 3 dimensional terrain. In *2014 22nd Signal Processing and Communications Applications Conference (SIU)*, pages 228–231. IEEE, apr 2014.

[41] Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[42] Singiresu S Rao. *Engineering Optimization: Theory and Practice*. John Wiley & Sons, 2009.

[43] Hassan Sarhadi and Keivan Ghoseiri. An ant colony system approach for fuzzy traveling salesman problem with time windows. *The International Journal of Advanced Manufacturing Technology*, 50(9):1203–1215, Oct 2010.

[44] Thomas Stützle and Holger H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[45] Kadri Sylejmani and Agni Dika. Solving touristic trip planning problem by using taboo search approach. *International Journal of Computer Science Issues (...*, 8(5):139–149, 2011.

[46] Kadri Sylejmani, Atdhe Muhaxhiri, Agni Dika, and Lule Ahmedi. Solving tourist trip planning problem via a simulated annealing algorithm. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, pages 1124–1129. IEEE, 2014.

[47] El Ghazali Talbi. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics*, pages 541–564, 2002.

[48] El Ghazali Talbi. *Metaheuristics from design to implementation*, volume 74. John Wiley & Sons, Inc., 20029.

[49] S. Teng, E. Chan, C. Yang, M. Yu, and S. H. Tan. An efficient solution framework for a large scale delivery problem. In *2014 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 647–651. IEEE, dec 2014.

[50] P. Tompkins, A. Stentz, and D. Wettergreen. Global path planning for mars rover exploration. In *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720)*, volume 2, pages 801–815. IEEE, 2004.

[51] Evangelos Triantaphyllou. *Multi-Criteria Decision Making Methods: A Comparative Study*. Springer, 2000.

[52] Eder Lúcio Trindade. *Um estudo sobre algoritmos de busca em grafos em tempo real*. PhD thesis, PUC, 2009.

[53] Anupam Trivedi, Dipti Srinivasan, Krishnendu Sanyal, and Abhiroop Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, 21(3):440–462, 2017.

[54] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[55] WTTC. World 2018 Annual Research: Key facts. Technical report, World Travel and Tourism Council, 2018.

[56] Li Yanfeng, Gao Ziyou, and Li Jun. Vehicle routing problem in dynamic urban traffic network. In *ICSSSM11*, pages 1–6. IEEE, jun 2011.

[57] E. Zitzler, L. Thiele, M. Laumanns, C. Da Fonseca, and V. Da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2):117–132, 2003.

[58] Eckart Zitzler, Dimo Brockhoff, and Lothar Thiele. The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. *Evolutionary Multi-Criterion Optimization*, 4403:862–876, 2007.

[59] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *TIK-report*, 103:95–100, 2001.