A multi-objective optimization approach to the ground handling scheduling problem

Silvia Padrón Astorga

PhD Thesis

Advisors
Dr. Juan José Ramos González
Dr. Daniel Guimarans Serrano

Dpt. Telecomunicació i d'Enginyeria de Sistemes
Escola d'Enginyeria
Programa de Doctorat en Telecomunicació i d'Enginyeria de Sistemes
Universitat Autònoma de Barcelona
May 2014

Dr. Juan José Ramos González, Associate Professor at the Universitat Autònoma de Barcelona, and
Dr. Daniel Guimarans Serrano, Researcher at the National ICT Australia (NICTA)

CERTIFY:

That the thesis entitled *A multi-objective optimization approach to the ground handling scheduling problem* and submitted by **Silvia Padrón Astorga** in partial fulfillment of the requirements for the degree of Doctor, embodies original work done by her under their supervision.

Dr. Juan José Ramos González
Thesis Director

Dr. Daniel Guimarans Serrano
Thesis Director

Dpt. Telecomunicació i d Enginyeria de Sistemes
Escola d Enginyeria
Universitat Autònoma de Barcelona
May 2014

Acknowledgements

El fin de esta aventura ha llegado. Tantas cosas han pasado en estos anos, tantas nuevas experiencias y tantas personas que, cada una a su manera, me han ayudado a salir adelante.

Quiero agradecer especialmente a mis directores, por su guía, su paciencia, por compartir conmigo su conocimiento y por todas las horas dedicadas. Gracias Juanjo y Dani por haberme apoyado y brindado tranquilidad en los momentos más difíciles.

Muchas gracias a mis padres, a mi hermano y a mi familia en Cuba. Sin su amor y apoyo incondicional no hubiera sido capaz de llegar hasta aquí. También a Xavi, gracias por tu comprensión, por tu ayuda cuando más lo he necesitado y por haber creído siempre en mí.

Doy gracias a mis amigos, a Laura, a Bety y Chuchi que fueron mi familia en Barcelona por mucho tiempo y al resto del piquetico aunque estemos tan lejos. A Mayte, a Anabel, a Nuria, gracias por estar siempre ahí.

Agradezco tambien a mis companeros del departamento, han sido muchos anos compartiendo juntos cada día. Gracias especialmente a Jenaro, Mónica, Catya, Yue y Tang.

Je tiens à remercier aussi Felix Mora-Camino et Catherine Mancel de leur accueil chaleureux à l ENAC. Merci également à Salma, Marcela et Isabelle pour leur agréable compagnie et pour avoir tenu le role de professeur de français.

Muchas gracias a todos, de verdad.

Contents

Chapter 1

Introduction

Air transportation is an essential factor in the economic and social progress of the modern world. It provides the principal means for long distances and international passenger traveling [1], which is crucial for global business and tourism development. According to [2], 40% of international tourists travel by air and 25% of sales in companies are dependent on air transport. Nowadays, facing the notable growing of traffic is one of the major challenges of the aviation industry. The considerable rise in the number of passengers carried in last years has led to increasingly congested airports, negative impact on the environment and important flight delays [1]. In 2012, around 35% of European flights were more than 5 minutes late with an average of 30 minutes [3]. Increasing airport capacities or building new infrastructures is an expensive and difficult solution. Hence, a better planning and an efficient use of the resources is decisive for a sustainable growth of this important industry.

Moreover, air transport is a very complex system, where various interconnected processes should be performed in compliance with numerous regulations and security restrictions. Different actors are involved in the different processes and each of them aim at optimizing their own resources and maximize their benefits. However, the goals and decisions made are conflicting in most of the cases. An integration of the planning decisions of each partner is needed to optimize the overall utilization of the limited infrastructure. In this sense, the Collaborative Decision Making (CDM) [4] approach was introduced with the aim to achieve a more collaborative coordination among airports, airlines, air traffic management, ground handlers, etcetera. CDM is the base of different efforts and important projects which are currently carried out such as the Airport-Collaborative Decision Making (A-CDM) [4] and the Single European

Sky ATM Research (SESAR) program [5], particularly focused on the Air Traffic Management. The TITAN Project [6] proposes to improve the efficiency of the turnaround through sharing reliable and timely information between the concerned actors.

Turnaround is a critical airport process. It is defined as the period of time the aircraft is on the ramp between an inbound and outbound flight, and different ground handling operations are performed. Ground handling comprises the activities, operative procedures, equipment requirements, and personnel necessary to prepare an aircraft for the next flight. Since ground tasks are very interdependent, each operation is a potential source of delays which could be easily propagated to other ground operations and other airport processes [7, 8]. Moreover, planning decisions about each vehicle required to perform an operation affect the scheduling of other activities and the performance of other resources. According to [6], the lack of integration of the different activities and an inefficient use of the resources during turnaround are important causes of flight delays.

Divisions of either airports or airlines have historically performed these operations. With the aim to improve competitiveness and the quality of the services provided, a process of deregulation of ground handling market was promoted at European airports [9]. In this sense, the number of self-handling services was limited and the subsidiary initiatives were encouraged. Since the Council Directive 96/67/EG was established in 1996, a notable increase in the number of third party companies has taken place [10]. In practice, and caused by the hierarchy of the overall airport planning, ground handlers are generally not included in the decision making of airlines, airports or other stakeholders. This means that handling scheduling procedures are conditioned to these prior decisions [11]. For instance, flight schedules defined by airlines determine the available time to complete ground activities and aircraft parking positions are set by the airport stand allocation.

This scenario with several ground handlers providing multiple services further increases the importance of efficient scheduling of ground activities [12]. Particularly, logistics in ground handling [13] and cooperative planning decisions are among the major challenges to improve the quality of the ground handling services. This context suggests the development of new tools that can help on the decision making process.

This thesis aims to contribute to the operational efficiency of ground handling facing the problem from a global perspective. To the best of our knowledge, this is the first time the problem is treated as a whole in the literature.

So far, other approaches have been developed to optimize the operations in isolation [12, 14, 15], but they do not consider the relations and entanglements between all the involved activities. We do explicitly consider these relations in our approach, and different operations and types of vehicles, each of them with their own available fleet, are modeled. We introduce a multi-objective approach for scheduling the ground handling vehicles that perform this set of interconnected activities. Solving this problem consists of obtaining a schedule for the vehicles that service aircraft at an airport during one working day. The schedule has to satisfy temporal, precedence, and capacity constraints. Besides obtaining optimized schedules for each activity, it is important to take into account the impact on other tasks. This way, we can integrate the scheduling decisions made for each service and contribute to optimize the overall ground handling process.

Two objectives are defined in order to integrate the scheduling of each resource and to improve the global solution: (i) minimizing the waiting time before an operation starts and the total reduction of the available time windows, using the vehicles efficiently; and (ii) minimizing the total completion time of the turnarounds. A method we call *Sequence Iterative Method* (SIM) has been developed to address the multi-objective problem. With the aim to achieve a more flexible and accurate decision process, the method was implemented to produce a range of solutions according to the Pareto concept of optimality [16]. This set of solutions with trade-off between the objectives gives decision makers the possibility of selecting the one that better suits the problem.

In order to apply efficient techniques to solve the problem, a decomposition scheme inspired by the workcenter-based decomposition for the Job Shop Scheduling [17] has been chosen. This schema provides a consistent method to solve the complete problem and simplifies the model and the solution process. First, a planning problem is solved and a time window for each operation is calculated according to the temporal and precedence restrictions. One Vehicle Routing Problem with Time Windows (VRPTW) is identified for each type of vehicle involved, which leads to multiple VRPTWs. These are solved individually and decisions made on each routing problem are propagated to the other VRPTWs through reductions on the available time windows.

The main features of this approach are modeled and implemented in *Constraint Programming* (CP). CP is a promising paradigm for solving real-life combinatorial optimization problems. It provides flexibility for modeling problems with side constraints and an efficient mechanism to discard infeasible so-

lutions by the constraint propagation mechanism. CP is used to obtain the operations time windows and to keep the consistency of the global solution. Furthermore, CP is used in combination with *Variable Neighborhood Descent* (VND) and *Large Neighborhood Search* (LNS) as a part of a hybrid methodology [18] to solve each of the routing problems. The well-known *Insertion Heuristics* method [19] has been applied to produce a quick first solution which is later improved using the hybrid methodology.

Finally, a version of the approach is proposed with the goal to allow a more exhaustive exploration to find Pareto solutions. In this case, solutions are obtained using only the Insertion Heuristic method and just the most promising are improved by the hybrid methodology. Two strategies are suggested to determine which solutions are the most promising and these rules can be also employed to guide the decision maker towards the selection of the best schedule to implement.

1.1 Objectives

The objectives of this thesis are:

- The development of an efficient approach to tackle the ground handling scheduling problem from a holistic perspective.
- The modeling of ground handling services during turnaround as a multi-objective optimization problem with the aim to integrate the planning decisions about each resource and enhance the performance of the global process.
- The development of an efficient method to address the defined multi-objective optimization problem. This method should provide reliable solutions that can satisfy the preferences of the decision maker.
- The employment of hybrid strategies to solve the general vehicle scheduling problem on ground handling operations. The study and application of a methodology based on *Constraint Programming*, *Variable Neighborhood Descent*, and *Large Neighborhood Search* to solve the approach is proposed.
- The assessment of the developed approach by its application over real airport data.

1.2 Structure of this Thesis

A brief description of the research context and the motivation of this thesis is introduced in this chapter. Moreover, the objectives of this work are defined. In Chapter 2, a background of the ground handling process in airports is presented. A state of the art of previous work related to vehicle scheduling in ground handling is also included. The principal technologies and techniques used to implement our approach are described in Chapter 3. The proposed approach is explained in detail in Chapter 4, as well as the variation developed to increase the exploration of the algorithm. Chapter 5 outlines the instances used to assess this methodology and the computational results obtained. Finally, conclusions and contributions of this thesis are provided in Chapter 6, as well as possible future research lines.

Chapter 2

Ground handling process

Airline operations are typically based on a flight network structure where an aircraft perform a set of consecutive flight legs over a period of time. An inbound flight arrives at an airport and passengers reach their destination or a connecting point to continue their journey. The aircraft is immediately prepared to an outbound flight during a turnaround or overnight stay. Between two flight legs, different activities are executed. These activities are known as ground handling operations.

Ground handling procedures are usually divided into two types: terminal and ramp. Terminal activities are performed inside the terminal buildings and concern passenger services. These include: check-in at counters for departing passengers, assistance at boarding gates, baggage handling and claim, etc. Ramp operations comprise most of the ground handling works and take place at aircraft parking position between the time of arrival at the stand (*In-Blocks*) and the departure (*Off-Blocks*). Figure 2.1 shows an example of the principal activities during a typical turnaround, when the aircraft is parked at a contact point, i.e. the aircraft is connected to the terminal via bridge.

- *Deboarding and Boarding*: These are key activities during turnaround. They can be carried out in different ways depending on the type of parking position assigned to the aircraft: contact or remote stands. When aircraft are parked at a contact point, loading bridges directly connected to the gate are used. Sometimes, when the stand is close enough to the terminal, passengers are allowed to walk to the aircraft [20]. Passenger buses and stairs are needed in case of remote stands.

- *Catering*: This is the main on board service and provides the required

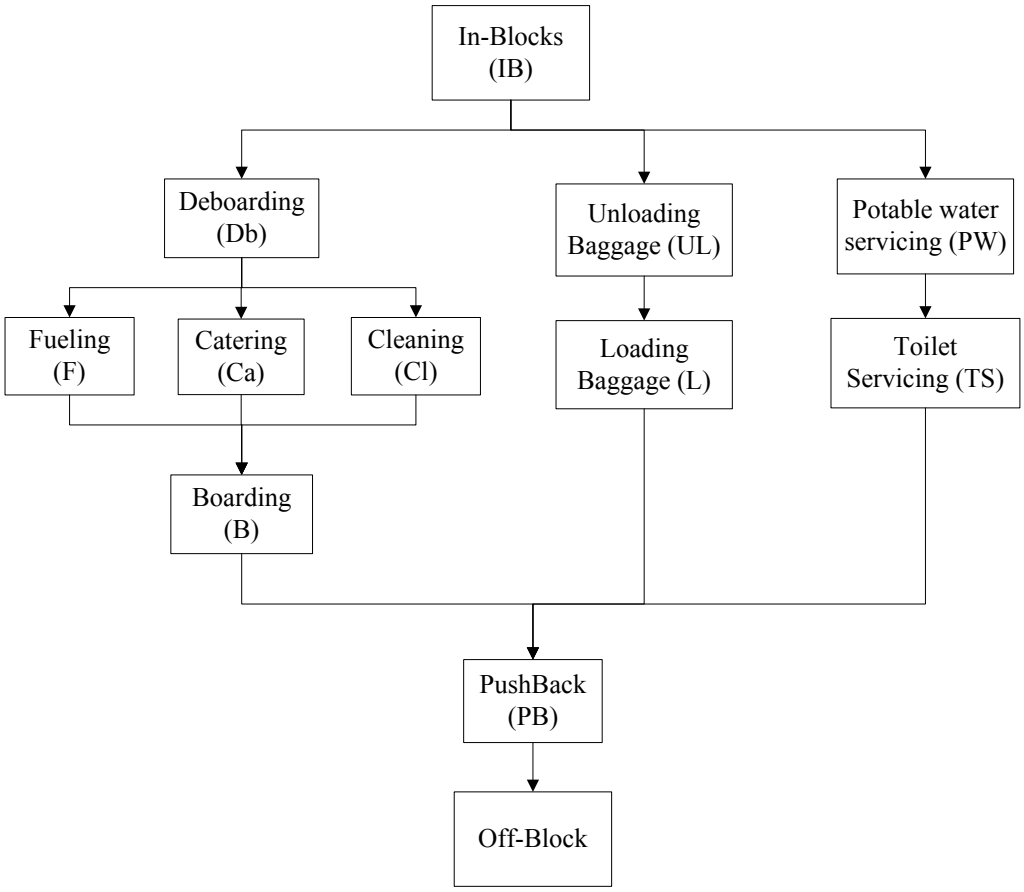

Figure 2.1: Example of activity flow during a turnaround at a contact point

meal for passengers according to the flight duration and the airline policy. Food is packed into trolleys at the catering airport facilities and stored at a handling station to be later carried to aircraft. Catering transportation is performed using high-loader trucks that permit trolleys to be rolled on and off the aircraft, for new food and waste [21] , respectively.

- *Cleaning*: Depending on the turnaround time available, cleaning cabin operation consists in taking out the garbage, or can include a larger number of tasks: empty seat-pockets, vacuum cleaning, galley and toilet disinfecting, exchange pillows and blankets, etc [7].

- *Fueling*: This is also an important and usually critical activity in short turnarounds. It can be performed using two systems. For small aircraft

or airports with low traffic, fuel can be transported by a tanker. On the other hand, the hydrant system is the preferred method for larger aircraft and airports. In this case, the fuel is provided by underground pipelines and a dispenser vehicle is used to connect a valve located at the apron with the aircraft. This method is safer than the first one. The dispenser vehicle can also service different types of aircraft [20].

- *Potable water and lavatory service*: Supply fresh water and collect the toilet waste are subject to many sanitary rules to avoid any contamination. Vehicles must not be parked in the same area and the personnel carrying out these services must be different.

- *Unloading and loading baggage*: Baggage handling is one of the most complex airline processes and requires a large number of resources and personnel. Baggage can be stored at aircraft in bulk or using containers depending on the type of aircraft. According to the type of storage, a conveyor belt or a container vehicle is used to unload and load the baggage in the aircraft.

- *Pushback*: It is the most common and safest way for an aircraft to leave the parking position [20]. A tug is connected to the nose gear and pushes back the aircraft out the stand, where the engines can be started to proceed under its own power.

Other activities are conducted to inspect the aircraft or to provide air conditioning and electric power. In general, theses operations do not have precedence relations and can be performed in any moment during turnaround. Other procedures are only required in special situations such as deicing services in winter.

Ground handling operations are provided by Ground Support Equipment (GSE), which can be mobile or fixed. Each operation is served by a specific type of GSE, therefore different ground units or vehicles are necessary. According to the task, some vehicles with a given capacity have to transport some quantity of resources to the aircraft stand (catering, fueling, and potable water operations), or collecting waste from the aircraft (also catering, lavatory services, and cleaning tasks). Likewise, there are some vehicles that do not transport any resource (pushback, baggage loader, and the fuel dispenser by underground pipelines).

Handling activities and GSE performance are regulated by the Airport Service Committee (ASC). The ASC is responsible for the management and

specification development of all the International Air Transport Association (IATA) airport services. GSE functional specifications and maintenance procedures, handling operation requirements, staff training, etc, are defined by this organization according to the IATA Standard Ground Handling Agreement [22].

Following these regulations, all the needed vehicles are located around the aircraft on the parking position to perform the associated operations. Some activities are carried out simultaneously, while other tasks are related by precedence constraints imposed due to security issues, space requirements or airline policies. For instance, fueling cannot be performed at the same time as passenger deboarding or boarding due to a security standard. In some cases, airlines can avoid this restriction with prior airport authority permission, e.g when a fire extinguisher is available. Because of hygienic reasons, the toilet and potable water servicing (collect the waste and re-equip with fresh water) cannot be conducted at the same time, but any of them can be performed first. Catering process usually has to be finished before boarding starts and, sometimes, they can only begin when deboarding is finished because it obstructs passenger ways. At small aircraft galleys are placed at the back and catering can be conducted simultaneously with boarding and deboarding without problems. A similar situation happens regarding cabin cleaning, which would preferably be performed between disembarking and embarking of passengers.

Activities duration, as well as turnaround time, depends on many different variables. These include operational variables related to the aircraft type (size, number of seats) and the service time required to carry them out (full servicing or minimum servicing). Handling vehicles have to visit the stand where the aircraft is parked, perform the assigned operation during a determined service time, and travel to next stands to perform next activities. Moreover, operations have to be executed in a predefined, and usually limited, turnaround time and precedence restrictions between tasks have to be met.

2.1 Planning in ground handling

Decision-making process in airlines can be classified into three levels according to the moment they are implemented: strategic, planning and operations [23]. Strategic level involves long term decisions and often takes place more than six months before the day of operation. Investments in new aircraft fleets or personal according to the forecast market demand are examples of strate-

gic decisions. Planning or tactical decisions occur between six months and one day before the execution and is concerned with the efficient use of the available resources defined on the strategic step. An airline planning is a complex process subject to many aircraft and maintenance regulations, personnel constraints, and large numbers of resources have to be synchronized. Flight scheduling is an example of decisions made at this level, which is an essential step determining the airline operations for next season. Usually, a new timetable is generated taking previous schedules as a base, where key aspects, such as origin-destination pairs, flight arrival and departure times, operation frequency, etc, are defined or updated. Others decisions made at this stage are the assignment of an aircraft type to each flight according to the number of passengers expected, among other considerations. Later, a specific aircraft, as well as pilots and cabin crew, are also assigned. Finally, the operation decisions take place the day of operation and they are verified or even changed every hour. This includes carrying out possible recovery actions to response to unexpected events or flight delays.

Decisions regarding ground handling operations are also made at each level. Buying new equipment or bidding for new contracts are examples of strategic decisions. At tactical level, resources and staff required to handle the expected demand for next season is defined. Shifts for workers and vehicles are designed according to the resource availability and labor regulations. Concerning vehicle scheduling procedures, they are generally taking place closer to the day of operation. Tasks to be carried out are assigned to shifts and a detailed execution plan for each day is generated [12]. Finally, during the day of operation, processes are replanned to adapt the schedule to external perturbations, such as arrival and departure flight changes or even internal disruptions.

With the liberalization process of the European airports started in 1996 [9], most of ground services are performed by independent companies. Due to the hierarchy of the overall airport planning, ground handlers are generally not included on the decision making of other scheduling processes (flight scheduling, stand allocation, etc). This means ground handlers have to fit their planning around a set of hard constraints. These include: aircraft arrival, departure, turnaround time, and parking position, among others [24]. Moreover, ground handling operations planning is usually limited to execute daily plans and react to external changes, rather than a real optimization process [12].

Airlines have a limited control over other actors in the system. For that reason, new strategies for improving flight punctuality are addressed to a better design of the flight schedule and the operational efficiency of ground handling

services [11]. The turnaround is a complex process and there are several tasks involved that have to be coordinated. Since some activities have to be finished before others can started, any perturbation can produce delay propagation to other operations and cause departure delays.

In order to ensure ground activities are finished within the scheduled turnaround, a buffer time is incorporated to absorb arrival delays and unexpected events during handling operations. Scheduled turnaround service time comprises the mean time to accomplish all the tasks satisfying the constraints imposed between them, and a specific buffer time. The right design of the flight schedule and the size of the buffer time have been a relevant issue of research [8, 11, 25] due to it is a trade-off situation not always easy to solve. Long turnaround time gives airlines a margin to cope with uncertainty and minimize delays. On the other hand, this reduces the utilization of the aircraft and the productivity. Since airlines revenue is associated to maximize the number of flights, reducing the time aircraft are on the ground is a desired goal. This principle is followed by the so called Low Cost Airlines, which usually work with very short turnaround times.

The buffer time gives also a leeway of action to ground handlers to plan their resources. In this sense, a time window can be assigned to each service. Figure 2.2 shows the available time to perform each ground handling activity during a turnaround subject to some precedence constraints.

2.2 Delays in ground handling

When the planned time of arrival or departure is not met, it is said a flight delay occurred. Ground processes at airport are the main cause of late departures, and in turn, they are the principal reason of arrivals delays [25]. Delays on flight schedule are frequently taking place and have an important impact on the airlines operational cost and passengers satisfaction. There are other causes of delays, such as not enough airport and airside capacities to face the demand of traffic, adverse weather conditions, technical problems of equipment, etc.

These unexpected events could be classified into two groups: internal or boundary [25]. Internal problems are associated directly to inefficiencies in the ground handling performance, e.g. when a service vehicle arrives late at the parking position. When disruptions are originated externally to ground handlers they are called boundaries.

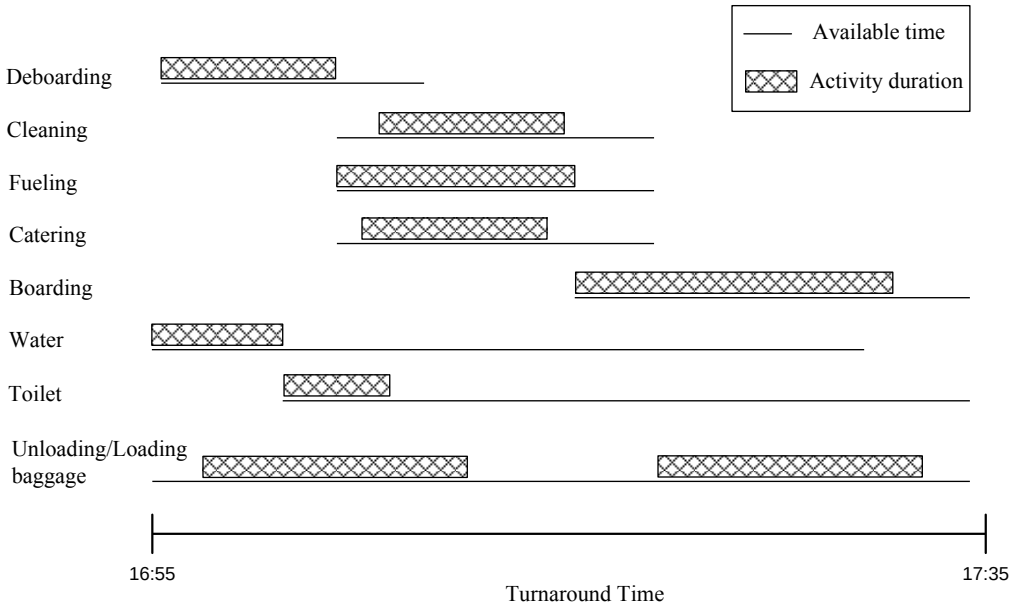

Figure 2.2: Time available to perform each ground handling activity

2.2.1 Internal events

As mentioned, airlines define the scheduled arrival and departure times for each flight during the planning process, as well as the turnaround time, including the buffer to deal with uncertainty. Turnaround operations are regulated by aircraft manufactures and the mean time to perform ground services is calculated using these standards. However, airlines can modify these specifications within certain limit to cope with their operational needs, which are usually confidential.

In this sense, airlines revenue interests can be a source of conflicts with a right definition of the turnaround time and the performance of ground handlers. Or viceversa, the cost impact associated to a departure delay lies principally on airlines so they need to ensure a correct ground handling accomplishment. Often, service level agreements are specified between these two partners in order to guarantee the efficiency of the turnaround process [11].

Due to bad scheduling or when the turnaround time established by airlines is insufficient, ground handlers often incur in safety violations to arrive on time, such as reckless driving [25]. In order to go faster or take a short cut, drivers usually move across prohibited zones, e.g. under aircraft wings, or sometimes

do not yield other vehicles when it is necessary. This increases the probability of collision and damage, even to aircraft, and can be an important cause of delays.

Other disturbances come from the inefficient use of ground handling equipment that can lead to an excessive usage of resources without the adequate maintenance period. Also, when a disruption occurs, a usual solution is allocating more vehicles to accelerate ground activities and reduce the risk of delay propagation [11]. When a vehicle breaks down, a spare vehicle is needed and it has to arrive from ground handling facilities, which are usually located near the terminal. This situation can cause major delays, particularly at remote stands.

2.2.2 Boundary events

External problems often cause major disruptions to the turnaround process. Many problems are associated with lack of airport or airside resources due to a poor planning or a wrong traffic prediction. Taxiway congestion, missing the assigned take-off slot time, last minute parking position changes, etc, lead important delays and significant modifications of the handling schedule.

One of the most common cause of incidents is adverse weather conditions. Regarding the performance of ground activities, the state of the pavement and low visibility make vehicles go slow and can produce operation delays and increase the risk of collision. Furthermore, wet pavement can produce incidents during aircraft taxiing and delay the beginning of the turnaround. Soil particles, such as sand or stones, can cause engines flaw, or aircraft can slide and hamper ground vehicles routes or even other aircraft [25].

Passengers are also another source of delays in ground handling. Long distance to arrive to the gate or long security procedures cause usually missing passengers, particularly when they are in transfer. Likewise, emergency situations due to ill passengers or specific requirements with people that need special attention can lead to unexpected issues.

In general, many of these events could be avoided or faced without resulting in a departure delay if they are considered when defining the turnaround time and vehicles are properly scheduled. When the turnaround is too short or resource planning is poor, it is difficult possible to absorb disruptions, reduce the risk of late departures and knock-on delays.

2.3 Scheduling vehicles. State of the art

Scheduling ground handling means have received less attention than other resources, both in industrial solutions and scientific literature. Expensive resources like aircraft or crews [26], or key processes, like the gate assignment problem [27], have been more relevant to airlines and airports and also an important subject in research. The growing competence between handling companies and the pressure to comply with airlines service level agreements have increased the necessity of new strategies and tools for assisting ground handling process. The adoption of novel technology solutions contributes to more accurate and timely performance and to an efficient use of the resources.

2.3.1 Industrial applications

Air transportation has benefited much from the development of information and telecommunication technologies. Air Traffic Control (ATC), airline and airport processes are supported by different kind of Information Technology (IT) applications that permit the expected achievements in this complex environment. A global overview of information systems used to assist different areas at a major airport is presented in [25].

There are few industrial solutions specifically designed to manage ramp operations and vehicles. Most of the current systems address terminal activities such as passenger and baggage handling. There are applications to handle operations at the apron providing specific functionalities, but in practice they are not widely used at present. In some cases, even at important airports, basic procedures like load sheet [28, 29] filling and delivery are carried out manually and using hard copies. Voice-based communication for dispatching and manual tracking vehicles are other common practices conducing to errors and reducing productivity [30].

According to marketing brochures, novel applications that propose advanced IT solutions have been developed for monitoring ground handling operations during turnaround both at airside and terminal area. Most of them are information support systems that supervise activity progress, as well as personnel and resource coordination. Main functionalities are updating and transmitting real time information, such as flight plan, task and vehicle engine status, using mobile communications. Fleet tracking is another common feature where Global Positioning System (GPS) technology is commonly used to know actual vehicle positions.

GroundStar system [31] is one of the most relevant examples developed by the Airport Systems Division of the software company INFORM [32]. The automatic generation, assignment and reassignment of tasks taking into account flight data, parking position or aircraft changes is the essence of this application. Different modules are included to deal with diverse functionalities: *GS GroundFleet* permits vehicles to be localized and tracked using a GPS system; operation status and vehicle positions are transferred by *GS RealTime* module, which provides a framework to manage information updating process. A wide amount of information and statistics is also provided and full daily reports about vehicles utilization can be obtained.

GroundStar application has also contributed to the Total Airport Management Suite (TAMS) [33]. This is an international effort based on the Eurocontrol A-CDM initiative aiming at integrating airport process management into one control operation center. TAMS project joins global IT leaders like Siemens Mobile with partners from the air transport sector, such as the German Aerospace Center and the Airport of Stuttgart. *GS HubControl* module controls and guides the progress of each aircraft turnaround in real time and provides early detection of bottlenecks and delays [34].

Other IT companies like Zebra Technologies or Huawei propose ground handling applications. Zebra Enterprise Solution [35] division has developed the *Fleet Management System*, which other functionalities in addition to vehicle track utilization are included. Management of the emptying and refilling fluid process (fuel, fresh or waste water), security issues monitoring (speed limits and vehicle collisions) and equipment maintenance control are examples of features provided by this application. Radio-frequency identification (RDIF) technology is the base of Zebra hardware to locate vehicles.

Huawei *Wireless Ground Handler Solution* [30] uses a 4G wireless network to transmit information including picture and video. Vehicle track and management solution is based on Geographical Information Systems (GIS), where vehicle location is sent every few seconds or minutes to the dispatching center. The system permits modifying the schedule according to the current situation and find the nearest available vehicle to service a specific task.

Finally, major airlines (e.g. Lufthansa [36]), or airports (e.g. Frankfurt [37]) have developed their own solutions to manage handling activities.

Regarding vehicles scheduling, this kind of tools are limited to automate resource allocation taking into account their availability. Classical heuristics like first-come first-served are the most common algorithms for dispatching vehicles [7, 14, 38] and important parameters, such as traveling distance or

service time duration, are rarely considered when resources are assigned to tasks [39]. *GroundStar* application comprises a planning module focused on the resource demand calculation and the staff shifts definition. That is, it is more about determining the number of resources or crews needed to cope with a certain workload than scheduling resources itself. Actually, obtaining optimized vehicle routes considering resource capacity limitations as well as time windows to begin services are future action lines of INFORM [40].

2.3.2 Scientific literature

Ground handling has neither been a rich research field and few examples of vehicle scheduling can be found in the literature. Moreover, most of studies are focused on one type of resource. To the best of our knowledge, none of these works address the scheduling of ground handling vehicles as a whole.

Fueling trucks

Du et al.[14] proposed a model to schedule the fueling tank truck vehicles based on Vehicle Routing Problem with Time Windows (VRPTW) with multiple objectives. They considered the minimization of the number of vehicles, the start time of the service and the total servicing time of the trucks, following this order of importance. Given a number of flights to serve ordered by their earliest service start time and a number of trucks with different capacities, a service schedule of the trucks is obtained. They solved the problem using Ant Colony System [41], which is inspired on the real ant behavior looking for food. Artificial ants, or vehicles in this case, update the pheromone using a defined updating rules and trace the search avoiding to get trapped into a local minimum.

The procedure built an initial solution using the Earliest Start Time First strategy, i.e. a flight with the earliest start time is fueled first. Then, the solution is improved regarding the objectives of the problem, which are encoding in the pheromone. A heuristic function is often used in the ant colony system to select next node to visit. In this case, a combination of different criteria is considered: earliest service time, shorter travel and waiting time and earlier latest service time. In order to reduce the exploration time and improve the performance of this algorithm, the authors include an auxiliary ant that generates a solution using the Earliest Due Date First heuristic [42], i.e. the flight with the earlier latest start time is served first. When the best solution

obtained in an iteration is worse than the auxiliary solution, the auxiliary one is taken as the best.

Catering process

Ho and Leung [15] proposed a sophisticated solution to tackle airline catering operations including staff workload. The research was focused on a major catering company that provides services to a considerable number of airlines at many airports, including Hong Kong International Airport. Generally, vehicles performing catering operation at aircraft require two workers: a driver and a loader. Food trolley transportation and loading procedures depend on the airline policy and the type of plane. Workers need specific skills to perform the operation and often they are not qualified to service all the possible aircraft or airline configurations.

The authors tackle the problem as a Site-Dependent Vehicle Routing Problem with Time Windows and Multiple Trips (SDVRPTWMT). Vehicles have to be scheduled to service aircraft at a given time window, during worker shifts and fulfilling skills requirements. The solution method was developed in two stages. A paring problem is solved in the first step and worker teams are obtained through a greedy heuristic approach. Next, aircraft operations are assigned to teams and scheduled in the second step. A Tabu Search metaheuristic is presented to solve this step, which is also divided into two stages to improve execution times.

First, an initial solution is built. Operations are selected in a random way and are allocated to the team in which time window, shift time, and skill constraints are satisfied and less tasks are assigned. Due to this method does not guarantee all operations are assigned, remaining tasks are attributed to some team anyway, provided that time windows are not violated. That is, the global problem is treated as a Manpower Scheduling problem [43] and the objective function maximizes the number of assigned jobs favoring balanced workloads. The solution is improved in the second stage considering this optimization goal.

This is a very complete approach for dealing with a particular ground handling operation. Different formulations and solution variants have been proposed at each step to deal with diverse situations or drawbacks produced by the decomposition. Moreover, the algorithm was developed using another approach (Simulated Annealing) in order to compare results obtained.

Connecting baggage transportation

Clausen [12] focused on baggage transportation for passengers in transit. Connecting baggage is not treated as regular baggage and the process can be different depending on the airport. In general, bags are carried from the inbound flight to a dispatching center, where the next step is decided according to the available time. There are two common possibilities: (i) bags are transferred to a baggage station if the regular bags delivery to the outbound flight has not started yet or (ii) baggage is directly transported to the flight. The problem is to obtain vehicle routes to transport bags either to the station or immediately to the flight according to the time windows associated to each option.

The author presents two views of the same problem, also called offline (static) and online (dynamic) optimization problem, as a case of study in a major European Airport. The offline version of the problem is tackled as a VRP with multiple time windows and multi-trips and an Integer Programming model is proposed. Minimizing the number of undelivered bags is the principal optimization objective, where the total travel time can be included as a second objective.

On the other hand, the online aspect of the second version means that, sometimes, all data required for scheduling is not known in advance. The author proposes dynamic optimization to address the operational planning in ground handling due to different reasons. Information updating and sharing processes at airports are not precise and reliable enough. Then, sometimes it is preferable scheduling resources only with the available data to avoid wrong assumptions. Also, airports are a complex environment subject to many perturbations and it is not always easy to accomplish a schedule made ahead of the day of operation and based on expected events. Furthermore, connecting baggage transportation has usually to be performed within a very tight period of time, which makes more difficult to cope with uncertainty.

There are two common classes of strategies to solve dynamic routing problems according to the way the dynamism is tackled: *local* and *local ahead* approaches [44]. In the former, routes are constructed only with the available information at the moment and a static problem is solved at each decision epoch. The planning horizon is divided into a set of sequences of time or decision epochs, where a new decision should be made. The latter includes probabilistic or predicted information into the static problem at each decision epoch.

Local approach is used to solve this problem. The lack of accurate infor-

mation which is the principal motivation for dynamic solutions, means often a considerable simplification of the optimization problem. In this case, the following modifications regarding the offline problem are considered: (i) only bags already arrived at dispatch facilities are considered, (ii) only available vehicles are scheduled and, (iii) vehicles are scheduled individually and for just one trip. The problem is transformed into a Traveling Salesman Problem (TSP) with profits, where the objective function in this case is an aggregation of three criteria. *Route length* and *departure* favors short routes and large differences between the expected bag delivery time and the flight departure, respectively. Alternating between handling station and flight positions is penalized by *Location* parameter.

A greedy algorithm is proposed to solve the problem, which is developed in two phases aiming at simulating a real dispatching environment. First, the route is obtained using the defined cost function and considering all the restrictions as well as the expected traveling times. Then, the route is tested using random traveling times in the second step. If the vehicle arrives late to the handling station, it is redirected to the flight as an extra visit before other deliveries. In contrast, if the vehicle did not arrive on time to the flight, the bag is considered as undelivered. Bag arrival times to the dispatching center besides traveling times are defined as stochastic variables. A log-normal distribution is selected to generate values for bag arrivals, while a normal distribution was chosen to model traveling times.

Passenger buses

Diepen et al. [38] present an alternative approach to cope with uncertainty and insufficient retrieval information: generating a robust schedule. They present a column generation algorithm for planning passenger buses at Amsterdam Schiphol Airport such that small disruptions can be absorbed without requiring huge reschedules. They considered robustness from an effective perspective: maximize the idle times between two pairs of trips assigned to the same bus.

Usually, passenger and regular baggage transportation present particular features in relation to other ground handling activities. They can be considered special cases of the Pickup and Delivery with Time Window problem (PDTW). In this sense, pickup passengers or bags from (or to) different flights is not allowed in the same trip. At each route, a bus performs a certain number of trips where each trip consists in three parts. First, passengers are picked up at the gate. Then, they are transported to the remote aircraft stand. Finally,

they get off the bus to board the aircraft for departure flights. In case of arrival flights, origin and destination are exchanged. Moreover, luggage and passengers transportation often need more than one trip to carry out one operation due to vehicle limited capacity or safety standards. Even when such trips can be done with different vehicles, they should not arrive at the gate at the same time for inbound flights.

An Integer Programming model is proposed based on a previous work for solving the gate assignment problem. A Column Generation algorithm is used to simplify the solution. Just a subset of bus plans (number of trips planned for each bus) is selected in such a way each trip is included in one of the selected bus plans.

A schedule for next day of operation using the expected arrival and departure flight times is obtained with this proposal. In order to evaluate the algorithm at an operational level, both the proposal and the method used at the airport were simulated. Flight times were updated as often as in the real process. The current system is based on first-come-first-serve heuristic and has a three hours scheduling horizon. Computer experiments show that the schedule obtained with the proposed procedure is less affected by arrival and departure changes than the current planning.

During operational planning, the schedule has to be recalculated each time a disturbance occurs. The authors suggested two alternatives for adjusting the algorithm to produce fast solutions. First, reducing the time horizon from one day to three hours ahead and lose the advantage of having a global overview of the planning. On the other hand, avoiding to calculate the whole schedule each time a change produces a conflict. In this case, it is possible to update trips associated to the disrupted flight and reschedule or create new routes to cover other trips that have become unfeasible.

Generic approach

A multi-vehicle version of the online scheduling is proposed as a part of the Car Management on Aprons (CARMA)Project [45]. CARMA initiative aims to spread the Advanced Surface Movement Guidance and Control System (A-SMGCS) [46] concept to vehicles at the apron in order to support the activities during turnaround. Most A-SMGCS implementations have been focused on offering guidance and surveillance to direct aircraft during runway or taxiway operations. Moreover, integrating this technology with fleet management permits a more accurate prediction of aircraft on-block times and a more efficient

use of resources.

In this case, the solution addresses the handling vehicle scheduling problem in a generic fashion without considering specific characteristics of one type of vehicle or operation. Authors of this work present several simple heuristics for planning vehicles that get bad solutions. Considering they are able to detect and identify vehicles in the apron, an improved algorithm solving the static scheduling over a short period of time is presented. The planning is executed every 10 minutes over a 60 minutes scheduling horizon, i.e. only aircraft requests with in the next 60 minutes period will be scheduled.

This approach permits a better dispatching avoiding poor suppositions. The drawback of this proposal is that arriving late to the aircraft position is allowed, as well as, operations do not have time windows to be serviced. This suggests the algorithm is may be best suited for solving a disruption problem than a planning problem.

2.4 A global approach

In the state of the art presented in Section 2.3, we find two main actions addressed to optimize ground handling operations during turnaround. First action focuses on improving vehicle monitoring and coordination process through a reliable and timely information transmission. Planning resources by using more suitable vehicle scheduling procedures is the goal of the second effort. This thesis deal with the second objective.

In general, ground handling activities are planned as a black box event taking place during the turnaround [11, 25]. The impact each individual operation has on others, airport processes and over the entire network, has been hardly outlined. Moreover, operation time stamps such as beginning, duration and ending time are rarely registered and this makes more difficult their study. According to the TITAN project [25] the turnaround is an essential component in the flight network and it has to be addressed in an explicit way to deal with the reactionary delays. Even in the A-CDM initiative [4], the turnaround has been treated as a single continuous process and the milestones defined have focused in the airside operations of the airport, the runway and the surface movement of the aircraft during taxiing. The TITAN project has expanded the scope of the A-CDM defining a set of milestones in order to control the progress of the different activities during turnaround. Some other examples can be found in the literature showing the importance of recognizing

this relationship as well as the need of collecting time stamps. [11, 39].

Closer to the objective of this thesis is the work proposed by Norin et al. [7]. They introduced the Airport Logistics concept that study the resource management at airports with a global vision of different logistic operations, resources and actors involved. Focusing on the turnaround process, authors proposed to get this whole overview through a simulation model of various operations during turnaround at Stockholm Arlanda airport. A detailed representation of turnaround is modeled and a set of checks are created to validate the simulation results with the schedule provided by airport experts.

An interesting integration between the simulation model and the schedule of deicing trucks obtained by a greedy optimization algorithm is outlined. They aimed to evaluate the turnaround performance in two situations: when one service is planned using an optimization process and when simple scheduling rules are adopted. Total delay and operation waiting time are defined as indicators to assess performance.

A formulation based on VRPTW with some further constraints to model the particular characteristics of deicing process is described. During winter, deicing operations are carried out to remove snow or frost from the aircraft. Two types of fluids are used: a warm liquid to eliminate the ice and an anti-icing to avoid new frost or ice covering the aircraft again before takeoff. Minimizing delays, as well as the traveling time of the trucks, are the optimization goals, which are summed into the objective function using weight parameters. Greedy Randomized Adaptive Search Procedure (GRASP) [47] is employed to solve the optimization problem. Two schedules were selected to be included in the simulation model from the set of solutions obtained with a balance between two objectives. The solution GRASP 1 with the least traveling time and solution GRASP 2, which produces the schedule with less total delay.

Three scenarios were created to test the inclusion of the optimized schedules in the model. In the first scenario, deicing trucks are planned using a basic rule based on the first-come-first-served procedure, where the flight with the earliest departure time is served first. The other two represent different optimization strategies. A first context is designed from the deicing operator perspective and aims to improve the performance of deicing trucks. So, the schedule GRASP 1 which have the best results in terms of traveling time is the preferred. In contrast, the goal of the third scenario is to provide the best solution for the overall turnaround in terms of delays. Therefore, the schedule GRASP 2 is selected in order to enhance the global process through the reduction of deicing delays.

Experiments show that the schedule optimized for the third scenario gives better results according to the total airport performance. This is a simple but effective mean of confirming the need of solutions that consider the influence the scheduling decisions about each resource have on the efficiency of the turnaround. Furthermore, the waiting time for the de-icing activity is reduced when schedules obtained with the optimization process are used. Results shows the benefit of employing a veritable optimization process for scheduling resources, both at a tactical level to increase the resource utilization and at operational level for better dealing with perturbations [7].

In this thesis we propose a global approach for scheduling ground handling vehicles at a tactical level. By tactical level we understand vehicle schedules are calculated using expected flight arrival and departure times, foreseen duration of operations, and using a planned gate assignment. Performing replanning actions to face short-term changes on the flight schedule or disruptions caused by internal or external incidents are out of the scope of this work. Instead, we aim to provide robust solutions that reduce inefficiencies in the resource usage. Moreover, we focus on solving this problem as a whole, taking into account relations between different tasks and their influence on the turnaround.

Tasks belonging to the same aircraft are related by precedence restrictions, as well as the time available to perform the each operation. The time when an operation begins could reduce the time windows of other activities depending on these restrictions, and therefore affect the performance of vehicles servicing them. For example, boarding has an earliest start time assigned to begin, but it has to wait until cleaning, fueling and catering have finished. If fueling does not start at its possible earliest time, boarding cannot begin at the earliest point of its time window. This conduces to a reduction of the original boarding time window and a new earliest start time has to be defined. This situation is represented in Figure 2.3.

The punctuality of each operation affects other operations and is a key aspect to ensure on-time flight departure. Scheduling vehicles to perform tasks as closer as possible to the start of their time window leaves some room to deal with unexpected events. This way, we can obtain a more robust global solution. While activities began at their corresponding original time windows, the turnaround can be completed on time and a reschedule is avoided. However, visiting the activities as early as possible leads to a higher number of vehicles required. Even though minimizing the number of vehicles is not an explicit optimization objective for us, it should be considered to solve the problem.

We aim to minimize the operation waiting time, i.e. serving each operation

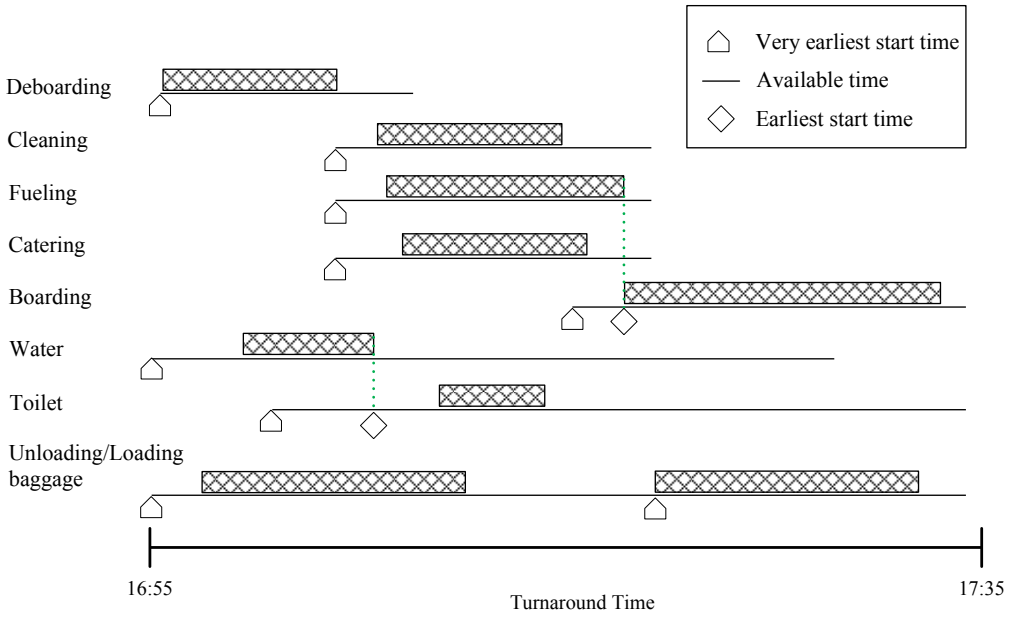

Figure 2.3: Scheduling decision about a resource cause time window reduction on other activities. The very and actual earliest start times are represented by a green line and dashed green line respectively.

as early as possible regarding its original time window, minimizing the total reduction of time windows and considering vehicle utilization. This leads to a second objective: to minimize the total completion time of the ground services at each aircraft. It is important to serve each operation as early as possible in such a way that the completion time of the turnaround is minimized. Having waiting time on the preceding operation does not always imply that successive operations could not be started at the earliest point of their time windows. For example, cleaning and fueling have different durations and they have to be finished before boarding. The task with the shortest service time (in this case, cleaning) will end before, but boarding operation have to wait for fueling task anyway. Hence, it is possible to use vehicles more efficiently allowing waiting time on certain activities (like cleaning) without affecting the completion time of the overall process.

Chapter 3

Technologies

In this chapter, we review the main technologies used to implement the approach proposed. The first section 3.1 describes the Constraint Programming (CP) paradigm and its principal features and mechanisms. Section 3.2 provides a background of the Vehicle Routing Problem with Time Windows (VRPTW) problem. The methods used for solving the VRPTW in this work are also explained, such as the Insertion Heuristic, Variable Neighborhood Search (VNS), and Large Neighborhood Search (LNS). Finally, the multi-objective optimization problem is discussed in Section 3.3.

3.1 Constraint Programming

CP is a very attractive paradigm due to the expressiveness and flexibility for modeling problems with side constraints. It has received much attention during last decades due to its potential for solving particularly real-world combinatorial optimization problems [48]. These applications often involve a heterogeneous set of side constraints and typically they have to deal with frequent update/addition of constraints [49]. The flexibility of CP is then a powerful characteristic, since adding constraints is a modeling issue and does not affect the search process.

In CP, the problems are described by means of three elements: variables, their corresponding domains, and the constraints relating these variables. *Constraints* represent logical relations among *variables*, each taking a value from a set of accepted values called *domain*, which can be a range with lower and upper bounds or a discrete list of numbers. This representation of the problem

in terms of constraints results in short and simple models that can be quickly implemented and modified to cope with updating requirements. Moreover, this permits testing different models until the best and fastest program for a particular problem has been found [18].

In general, CP concepts are embedded in a programming language, particularly in logic programming languages, such as Prolog. In this case, it is called Constraint Logic Programming (CLP). The algorithms developed in this thesis have been implemented using the CLP platform ECLiPSe [50]. Other typical languages are also employed to solve CP problem such as C/C++, e.g. COMET [51] or ILOG [52], and Java, e.g. Cream [53].

3.1.1 Constraint Satisfaction Problem

Since CP is the study of computational systems based on constraints, its idea is to solve problems by stating constraints (requirements) about the problem area and, consequently, finding a solution satisfying all the constraints. This class of problems is usually termed Constraint Satisfaction Problems (CSP) and the core mechanism used in solving them is constraint propagation [54]. It involves deleting from variable domains values that cannot satisfy the problem constraints. When a value is assigned to a variable, it is propagated through the associated constraints to the rest of the variables involved in these constraints. If there are values in other variables domains that are incompatible with propagated assignments, they are also removed [49].

A CSP problem P can be formulated as a triple P= (X,D,C) [55] where:

- X is a finite set of *variables* $x_1, x_2, ..., x_n$;
- D is the corresponding set *domains* $D_1, D_2, ..., D_n$ such that $x_1 \in D_i$
- C is a finite (possibly empty) set of *constraints* $C_1, C_2, ..., C_t$

Each constraint c is a relation defined on a set of variables such that $X(c) = (x_{i_1}, ..., x_{i_{|X(c)|}})$ where $|X(c)|$ is called the *arity* of the constraint, i.e. the number of variables involved. A constraint can be *intentionally* defined using a formula which the variables of the constraint have to satisfy or *extensionally*, where the set of satisfying values are explicitly established. Constraints with arity one are *unary* and with arity two are called *binary*. When the arity is greater than two it is said non-binary, and global constraints are defined by a formula of arbitrary arity [54].

Solving a CSP problem involves the assignment of values to the variables (*labeling*) that satisfy all the constraints. A solution is an n-tuple $A = a_1, a_2, ..., a_n$ such that $a_i \in D_i$ and C_j is satisfied. A deeper definition and an example of how constraint propagation mechanism works are presented in [18].

Depending on the problem we may need to obtain: (i) just one solution, if exists, with no preference to which one; (ii) the set of all solutions; or (iii) an optimal, or at least a good solution, given some objective function defined in terms of some or all of the variables. In the last case the CSP becomes a *Constraint Optimization Problem* (COP). If the set of solutions is empty it is said that the CSP is unsatisfiable.

Inference and search are in general combined in the solution process of a CSP. Trough constraint propagation, unfeasible alternatives are eliminated in advance reducing the exploration of the search space. If all the variables are instantiated after the propagation algorithm is triggered, a solution of the problem is found. Otherwise, a search process is launched through the possible assignments of values to variables, generating the whole search tree.

3.1.2 Constraint propagation

Most of the efforts to improve the performance of backtracking have been focused on constraint propagation and making possible early inconsistency detection that avoid fail instantiations [54]. Different consistency techniques are proposed, the most common are:

- *Node-Consistency*: it removes values from variables domains that are inconsistent with constraints involving one variable, i.e. unary constraints. It is the simplest consistency technique.

- *Arc-Consistency*: it removes values from variables domains which are inconsistent with constraints involving two variables, i.e. binary constraints.

- *Path-Consistency*: it requires for every pair of values of two variables x and y satisfying the respective binary constraint that there exists a value for each variable along some path between x and y such that all binary constraints in the path are satisfied.

- *K-Consistency* and *Strong K-Consistency*: a constraint graph is k-consistent if for every system of values for $k-1$ variables satisfying all the constraints among these variables, there is a value for an arbitrary k^{th} variable such that the constraints among these variables are satisfied. A constraint graph is strongly k-consistent if it is j-consistent for all $j \leq k$. All previously mentioned techniques can be generated by k-consistency and strong k-consistency.

In general, consistency methods are incomplete and a search process is necessary to obtain a solution [56]. In any case, they offer good mechanism to remove inconsistent values from variables domains during search. However, keeping certain level of consistency on the constraints can lead to long running times. Thus, special attention should be paid to designing search algorithms in order to achieve the equilibrium between reduction of the search space and efficiency.

3.1.3 Search methods in CP

Search space can be explored either systematically, e.g. using backtracking, or using variants of local search. Backtracking search is a basic procedure for solving CSPs which performs a depth-first traversal of the search tree, where each node describes a choice of a value for a variable, and each branch represents a candidate partial solution. The process consists in extending incrementally a partial instantiation, that contains feasible values for some of the variables, to a complete solution by repeatedly choosing a value for another variable consistent with the values in the current partial solution. Variables are assigned sequentially and when all the variables associated to a constraint are labeled, the method verifies if the constraint is satisfied. If a partial solution violates any of the constraints, backtracking undoes the last assignment made and tries with the most recently instantiated variable that still has alternatives available. Hence, it does not has to wait until a complete solution has been generated to check consistency. Backtracking is a high-consuming time method mainly caused by three aspects [18]: (i) the so-called thrashing where repeated exploration of failure subtrees is performed because they differ in assignments to variables no relevant to the failure [55]; (ii) redundant work due to conflicting values are not remembered which makes the exploration fail the same way in different branches of the tree; and (iii) late detection of conflicts.

Some alternatives have been proposed in order to deal with these problems.

Backjumping aims to overcome trashing performing the backtrack to the last recent conflicting variable instead of the last instantiation. *Backchecking* or *Backmarking* address the redundant work reducing the number of compatibility checks. The looser avoid also redundant discoveries of inconsistencies.

The way of combining the reasoning provided by constraint propagation with the search process affects the detection of inconsistencies and two principal schemes can be identified. Backtracking search, as well as the variants mentioned above such as backjumping, backchecking, and backmarking, belong to the *look back* group. *Look back* schema performs compatibility checks among already instantiated variables conducing to a late detection of the conflict. On the other hand, *look ahead* approaches aim to prevent future violation of constraints. *Forward Checking* is the most common example and enforces arc consistency between pairs of uninstantiated and instantiated variables. When a variable is labeled, any inconsistent value in the domain of the still unlabeled variables which conflicts with this assignment is (temporally) removed from the domain. Other methods such as *Partial Look Ahead* or *Maintaining Arc Consistency* remove more incompatibilities but are more expensive in computational effort.

Other aspects having a crucial impact on the efficiency of backtracking are the order the variables are instantiated and the order values from their domain are selected [57]. Variable ordering is a very important decision since it can transform significantly the shape of the search tree. Value ordering selects which subtrees are first explored and this choice can influence how soon a very good solution, or even an optimal one, may be found. Backtrack search may be considerably improved with a right combination of both strategies, while a wrong choice could conduce the algorithm to explore infeasible parts of the search tree.

Variable and value ordering can be either fixed, where sequences are defined *a priori*, or dynamic, where ordering decisions are made according to the information available at that point of search. Dynamic approaches have been widely studied and different strategies have been proposed with interesting results. *First-fail* is one of the most typical heuristic where variables with the smallest domain size are preferred. With a similar goal, the *most-constrained* heuristic is used, but if several variables have the same domain size, the one with the largest number of attached constraints is selected. Other strategies are based on domain values of the variables, such as *smallest* heuristic, where the variable with the smallest value is favored. Regarding the value ordering, different approaches have also been proposed, such as selecting the values in in-

creasing order, starting from the middle of the domain or employing a random selection. Most of these variable and value ordering heuristics are implemented in most CP solvers.

3.1.4 Constraint Optimization Problem

As mentioned in Section 3.1.1, solving a CSP implies to find an optimal, or quasi optimal solution that minimizes or maximizes a cost function. In order to deal with COP, backtracking search is generally adapted and called *Branch-and-Bound* (BB). During the search, BB keeps the current best cost of the objective function (bound) and, each time a solution with a smaller cost is found, its value is updated. There are many variations of the BB method. One consideration is what to do after a solution is found with a new best cost. The simplest approach is to restart the algorithm with the bound variable initialized to this new best cost. A less naive strategy is to continue the search for better solutions without restarting. In this case, the cost function upper bound is constrained to the bound variable value. Each time a solution with a new best cost is found, this cost is dynamically imposed through this constraint. The constraint propagation triggered by this constraint leads to a pruning of the search tree by identifying the nodes under which no solution with a smaller cost can be present.

Depending on the problem, using only BB to find an optimal or good solution may need excessive execution time. Hybrid algorithms combining backtracking search and local search appear to be a good alternative to solve complex problems like VRPTW.

3.2 VRPTW

The Vehicle Routing Problem (VRP) [58] is one of the most popular combinatorial optimization problems. It is aimed at determining an optimal set of routes for an available fleet of vehicles in order to service a set of customers, subject to different constraints. These constraints depend on the specific problem to solve and several VRP variants have been studied to cope with more common situations.

The Vehicle Routing Problem with Time Windows (VRPTW) [59] is a widely researched VRP version and states that each customer has a time window within the vehicle has to begin the service. These time windows can be

hard, like in our particular problem, or soft. In the former, the service cannot begin at all before the earliest start time, as well as after the latest start time. Nevertheless, vehicle waiting time is generally allowed, i.e. a vehicle can arrive early at the customer site and wait until the time window is open. It should be notice that vehicle waiting time and operation waiting time is not the same concept. The later refers to the difference between the lower bounds of the time window and the moment the operation begin. In the soft case, time window violations are permitted at the expense of having penalty cost.

The VRPTW is an extension of the Capacited Vehicle Routing Problem (CVRP). CVRP is the most basic VRP version, where a fixed fleet of vehicles with uniform capacity should deliver goods to a set of customers. Each vehicle starts and finishes its route in a central depot.

The symmetric CVRP can be considered as a complete undirected graph $G = (I, E)$, connecting the vertex set $I = \{1, 2, ..., n\}$ through a set of undirected edges $E = \{(i, j) \mid i, j \in I\}$. The edge $e_{ij} \in E$ has associated a travel cost c_{ij}, supposed to be the lowest cost route connecting node i to node j. Each vertex $i \in I \setminus \{1\}$ has a nonnegative demand qi, while node 1 corresponds to a depot without associated demand. A fixed fleet of m identical vehicles, each of capacity Q, is available at the depot to accomplish the required tasks. Solving the CVRP consists of determining a set of m routes such that: (i) each customer is visited exactly once by a single vehicle, (ii) each route starts and ends at the depot, and (iii) the total demand of the customers assigned to a route does not exceed the vehicle capacity. In case of the fleet size is not fixed, minimizing the total number of used vehicles becomes an additional objective.

The VRPTW extends the CVRP by associating a travel time t_{ij} to each edge $e_{ij} \in E$. Each vertex $i \in I \setminus \{1\}$ has a time demand t_i required to perform the service, which has to start within a defined time window $[a_i, b_i]$. The primary objective of the VRPTW is to find the minimum number of routes, i.e. use the minimum number of vehicles. A secondary objective is imposed to minimize the total cost of routes, that can be expressed in terms of the total traveled distance or the total scheduled time. The depot may also have a time window associated.

The VRPTW has been extensively studied and several formulations and exact algorithms have been proposed [60]. Solving this combinatorial optimization problem is NP-hard [61] and the use of heuristic algorithms [62], metaheuristics [63] and recently hybridization methods [64] have been intensive subjects of research.

In relation to the heuristic algorithms, they can be classified in constructive

and improvement methods. A route construction heuristic select nodes (or arcs) sequentially (or in parallel) until a feasible solution has been created. Nodes are chosen based on some cost minimization criteria, often subject to the restriction that the selected nodes do not create a violation of vehicle capacity or time window constraints. Most popular constructive algorithms for VRPTW are those proposed by Solomon in 1987 [19], which in general have higher solution quality/time ratio [62]. One of the most successful methods is the Insertion Heuristic and we have described it in more detail in Section 3.2.1. Solomon provides three variants of Insertion Heuristic: I1, I2, I3. We have adopted the I3 version for solving the routing problem in our approach.

Improvement methods are concerned to the concept of local search, which iteratively improves the solution by exploring the neighborhood. Edge-exchange based algorithms are the most typical improvement methods employed to solve the VRP [62]. The disadvantage of local search methods is they are likely to get trapped into a local minimum. They perform a myopic exploration and only accept solutions that improve the objective function [62]. Thus, the outcome has a strong dependence on initial solutions and the mechanism applied for neighborhood generation.

Metaheuristics are used to overcome this situation providing a guide for exploring the search space. A successful metaheuristic should provide a balance between the diversification, i.e identify parts of the search space with high quality solutions, and intensification, i.e. intensify the search in promising regions [65]. Developing and applying this high level strategy for solving combinatorial optimization problems has been an important field of research and a considerable variety of examples can be found in the literature.

Metaheuristics can be classified according to several criteria, and the more basic distinction is the number of solutions generated at each time [66]. In this sense, methods can be divided into Single-Solution Based and Population-Based groups. Methods from the first group work on a single solution at each iteration and, in general, are based on a local search process. Most popular methods for solving routing problems, such as *Tabu Search* [67], *Simulated annealing* [68], *Guided Local Search* [69], and *Greedy Randomized Adaptive Search Procedure* (GRASP) [70], are part of this group. Another interesting example is *Variable Neighborhood Search* (VNS) [71], where a set of neighborhoods is employed in order to improve the diversification and intensification of the search. We have selected VNS for solving the VRPTW and it is explained in more detail in Section 3.2.2.

In contrast, *Ant Colony Optimization* [72], *Genetic algorithms* and *Evolu-*

tionary Algorithms [73] are examples of population-based metaheuristics. In this group, the exploration is based on the evolution of a set of solutions in the search space. A recent overview of the more representative methods of both groups is presented in [65] and exhaustive reviews of the application of metaheuristics for solving VRPTW and others VRP variants can be found in [18, 63, 74].

Finally hybridization of exact and heuristic methods has raised an increased interest in the research community in last years. Promising results have been obtained combining the strengths of both approaches, particularly for real life problems. A hybrid methodology which exploits the advantages of Constraint Programming (CP) in combination with Large Neighborhood Search (LNS) and VNS [18] has been used in our problem. CP and LNS are described in details in Section 3.1, 3.2.3 respectively.

3.2.1 Insertion Heuristic method

As mentioned in Section 3.2, different methods for solving the VRPTW were proposed by Solomon[19]. In general, the goal was adapting the main VRP heuristics to cover the time restrictions of VPRTW problems. In this sense, besides the distance, the time dimension was included in the heuristic process. *Insertion Heuristic* is one of these heuristics and we have adopted it in our research work. In spite of being proposed more than twenty years this method is still frequently used by researchers and very recent examples can be found in the literature [75, 76, 77]. In general, *Insertion Heuristic* is employed to obtain fast and acceptable good initial solutions, which are later improved by more sophisticated methods.

Basically, this procedure constructs the routes by inserting one unrouted customers at each iteration in a sequential way. At first, a route is generated and initialized with the best visit according to different criteria. Then, one of the remaining unrouted clients is selected to be added in the current route in the best feasible position. Both criteria to decide the selection of the customer and position are defined by two cost functions c_1 and c_2. When no more customers with feasible insertion positions can be found, a new route is created. If all the clients are already routed, the algorithm is stopped. In detail, the general *Insertion Heuristic* consists of the following steps:

1. Initialization

Solomon proposes three alternatives to select the first client on the route: (i) the farthest unrouted customer in relation to the depot, (ii) the client which has the lowest deadline to begin the service, and (iii) the customer which the combination between distance and time to get to it from the depot is the minimum.

2. Checking feasibility

   Feasible conditions have to be examined before determining the best insertion place. For adding a client in the current route, both time and capacity constraints have to be respected. In this case, capacity restrictions are quite easy to check and do not depend on the position that the client is inserted in. In fact, it is sufficient to verify that the new demand plus the total cumulative demand on the route does not exceed the vehicle capacity.

   On the other hand, temporal restrictions require particular attention. A position is feasible if the time windows of the new customer, as well as those of all the customers already included in the route are satisfied. The start time of the visits after this position may be modified.

3. Find the best feasible position

   The best feasible position between two consecutive customers i and j on the emergency route is calculated for each unrouted customer u by $c_1(i,u,j)$ such that:

$$c_1(i(u),u,j(u)) = \min\left[c_1(i_{p-1},u,i_p)\right] \quad p = 1 \ldots m \tag{3.1}$$

   where m is the number of customers already included on the route

4. Select the customer

   In the next step, c_2 is used to choose the customer u^* (with a feasible position on the emerging route) to be inserted in the position found with c_1.

$$c_2(i(u^*),u^*,j(u^*)) = optimum\left[c_2(i(u),u,j(u))\right] \tag{3.2}$$

5. Build a new route

   If no unrouted customers have feasible insertion positions in the current route, a new route is initialized.

Solomon proposed three variants of the *Insertion Heuristic* method called I1, I2 and I3. The goal of including not only the distance but also the time dimension in the construction of the route is presented in the three heuristics. Nevertheless, the definition of the cost criteria depends on each variant and the approach followed in each case. Solutions found with each heuristic were analyzed in terms of number of vehicles, total scheduling time, total distance and vehicle waiting time. The total schedule time is the sum of the travel time, the total service time and and the vehicle waiting time.

I1 variant

In the I1 version, the best position is the one minimizing the weighted combination of the additional distance and time required to add the customer in the current route. Cost c_1 of inserting the unrouted customer u between the two consecutive customers i and j is calculated with function 3.3. The two elements of c_1, $c_{11}(i,u,j)$ and $c_{12}(i,u,j)$, are described in equations 3.4 and 3.5. The cost c_2 used to select the customer is formulated in equation 3.6. I1 chooses the customer with the largest difference between the distance cost of visiting this client in a new route and the cost of insert it in the current route.

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) \tag{3.3}$$

$\alpha_1 + \alpha_2 = 1;\ \alpha_1, \alpha_2, \alpha_3 \geq 0$

$$c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij};\ \mu \geq 0 \tag{3.4}$$

where d represent the direct distance between two customers

$$c_{12}(i,u,j) = b_{j_u} - b_j \tag{3.5}$$

b_j is the time for begin the service at customer j and b_{j_u} is the new start time at j, given that u is in the route.

$$c_2(i,u,j) = \lambda d_{0u} - c_1(i,u,j);\ \lambda \geq 0 \tag{3.6}$$

That is, in the first variant I1, the distance remains as a critical decision rule and the time dimension is still only considered in a partial way. For that reason, I1 can be viewed as the most classic of the three insertion heuristics and gives the best distance results. According to Solomon, even if the difference with the other two methods is not exceptional, particularly regarding the

number of vehicles, I1 version was able to find the best known solution for an appreciable number of instances when it was proposed. In next variants, the goal of including the distance as well as the time dimension is taken into account with the same importance in both decisions.

I2 variant

Function $c_1(i,u,j)$ introduced in equation 3.3 is used again to determine the best insertion place in I2. Regarding $c_2(i,u,j)$, I2 prefers customers which produce the minimum cost regarding the total distance and the total scheduled time of the route. Function c_2 is presented in equation 3.7.

$$c_2(i,u,j) = \beta_1 R_d(u) + \beta_2 R_t(u) \tag{3.7}$$

where $\beta_1 + \beta_2 = 1;\ \beta_1 \geq 0\, \beta_2 > 0$. $R_d(u)$ and $R_t(u)$ are the total route distance and time of the current partial route, if u is inserted.

I3 variant

The third version I3 gives priority to the most pressing customers. A third parameter is incorporated in the c_1 function 3.3 in order to prioritize the customer having the lowest deadline to begin the service. The formulation of the function c_1 is presented in relation 3.8 and the new component $c_{13}(i,u,j)$ is introduced in equation 3.9. Function c_2 is defined in equality 3.10.

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j) + \alpha_3 c_{13}(i,u,j) \tag{3.8}$$

$\alpha_1 + \alpha_2 + \alpha_3 = 1;\ \alpha_1, \alpha_2\, \alpha_3 \geq 0$

$$c_{13}(i,u,j) = l_u - b_j \tag{3.9}$$

where l_u is the latest start time to begin the service at j

$$c_2(i,u,j) = c_1(i,u,j) \tag{3.10}$$

The principal difference regarding the other two variants is that the same criterion is used to select both the best place for insertion and the best client. Therefore, the individual cost of the client to be added is in this case taken as a measure rather than the total route cost proposed in I2.

In our problem, we aim to minimize the operation waiting time without sacrificing the number of vehicles required. Hence, the insertion of unrouted customers should be guided by both geographical and time criteria with the same importance. Cost c_1 defined in 3.3 appears as a suitable function to decide the best feasible position and the best customer to insert. In this sense, either I1 and I3 heuristic can be adopted and become equivalent using the following weight combinations: $(\lambda = 0, \alpha_1, \alpha_2)$ for I1 and $(\alpha_1, \alpha_2, \alpha_3 = 0)$ for I3. However, the third parameter included in I3 contributes to reduce the vehicle waiting time which in turn can lead to reduce the number of vehicles required [16]. Thus, we have decide that I3 is the most appropriate variant for solving our problem.

3.2.2 Variable Neighborhood Search

Compared with other popular metaheuristics such as Simulated Annealing (1983) [78] or Tabu Search (1986) [79], *Variable Neighborhood Search* (VNS) is a recent metaheuristic which was introduced in 1997 by [71]. It has received a fast growing attention and promising results have been reported when applying it to solve different VRP problems [63, 80, 81, 82, 83]. VNS adopts the principle of changing the neighborhoods systematically either in the descent phase to find a local minimum as well as in the perturbation phase in order to escape from the corresponding valley. In this sense, the search process using a set of neighborhoods is based on two main observations:

1. A local minimum with respect to one neighborhood structure is not necessarily a local minimum for another neighborhood structure.

2. A global minimum is a local minimum with respect to all possible neighborhood structures.

Moreover, experiments show that, for many problems, local minima with respect to one or several neighborhoods are relatively close to each other [84]. This implies that a local optimum often provides some information about the global optimum and it is used in VNS to guide the search. Regarding other metaheuristics, VNS and its variants are simple and require few, and sometimes no parameters. A complete description for the VNS and the different extensions are presented in [18, 85, 84, 86].

Algorithm 1 Variable Neighborhood Descent (VND)

Initialization: - Find an initial solution x.
- Define a set of neighborhoods $N_k(x), k = 1, .., k_{max}$
Repeat the following sequence until no improvement is obtained:

1. Set $k \leftarrow 1$.
2. Repeat the following steps until $k = k_{max}$:
   (a) Exploration of neighborhood: find the best neighbor x' of x ($x' \in N_k(x)$).
   (b) Move or not: if the solution x' thus obtained is better than x, set $x \leftarrow x'$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$.

Variable Neighborhood Descent (VND)

The *Variable Neighborhood Descent* (VND) is the simplest VNS variant. The change of neighborhoods is performed in a deterministic way. It is often used as a local search method for more complex frameworks, such as VNS itself. Departing from an initial solution and a neighborhood structure, if an improvement is reached the process is restarted with the best value; otherwise, a change of neighborhood is enforced. The VND steps are presented in Algorithm 1.

Let N_k, $(k = 1, ..., k_{max})$ be a finite set of pre-selected neighborhood structures, and $N_k(x)$ the set of feasible solutions in the k^{th} neighborhood of x. An optimal solution x_{opt} (or global minimum) is a feasible solution where a minimum of the problem is reached. Let X be the set of feasible solutions, $x' \in X$ is a local minimum with respect to N_k if there is no solution $x \in N_k(x') \subseteq X$ such that $f(x) < f(x')$. We assume that an initial solution x is given.

When the *best accepted* strategy is used to explore the neighborhood, the accepted move lead to the best neighbor. In this case, the chances to reach a global optimum are larger when using VND than with a single neighborhood structure. However, performing a complete search in huge neighborhoods can be very expensive in computational time. A fast alternative is a *first accept* strategy where the first improvement of the solution is approved for making a move.

VND is a very simple and effective algorithm. Nevertheless, some issues

have to be considered in order to get an efficient exploration of the search space [18]. First, the complexity of the different moves used to determine the neighborhoods has an important impact on algorithm s performance. In general, a move defined over a simple elementary change may generate a very large neighborhood. If exploring such a neighborhood involves checking too many elementary changes, the resulting heuristic may be very slow and often take more time than an exact algorithm on small or medium size examples [18].

Another relevant aspect when implementing a VND method is the sequence in which the different neighborhoods are applied. This decision can affect the running time and the quality of solutions obtained. A frequent implementation consists of ranking neighborhoods by order of complexity of their application.

Finally, it is crucial that the moves considered lead to a thorough exploration of the region containing x. For some problems, elementary moves are not sufficient to escape from a narrow valley, and heuristics using them only can give very poor results. This is also related to the desired quality of the VND final solution. In most situations, a better solution will be pursued when VND is used alone, while poorer solutions may be accepted when it is embedded in a larger framework, such as VNS itself.

3.2.3 Large Neighborhood Search

Large Neighborhood Search (LNS) is a search algorithm proposed by Shaw [87] and consists in gradually improving an initial solution by alternately destroying and repairing it. A complete description for the LNS is provided in [18, 88]. LNS is considered a *Very Large Scale Neighborhood search* (VLSN) heuristic and as its name suggest, this kind of local search methods try to escape from local minima generating large neighborhoods [89]. VLSN algorithms are based on the perception that searching a large neighborhood results in finding better local optima and hence better final solutions. However, searching a large neighborhood is time consuming, therefore different filtering techniques are used to reduce the search [88]. For that reason, the neighborhood is typically restricted to a subset of the solutions, which can be explored efficiently.

LNS has proved to be competitive with other local search techniques, especially when combined with CP [18]. In this sense, the CP framework benefits from the the efficiency of the search process provided by LNS while LNS benefits from the propagation achieved with CP [90]. Actually, LNS was originally introduced to solve the VRP problem in combination with CP [91]. In this

work, the search operates by choosing in a randomized fashion a set of customer visits. The selected customers are removed from the schedule, and then reinserted at optimal cost. To create opportunities for interchanging customer visits between routes, the removed visits are chosen so that they are related. A BB method combined with CP is then used to reschedule removed visits. Due to high computational requirements, this approach can be applied only to problems with relatively low number of customers per route. A similar work is presented in [87], which uses constraint-based limited discrepancy search in the reinsertion of customers within the BB procedure. The number of visits to be removed is increased during the search each time a number of consecutive attempted moves has not resulted in an improvement of the cost. Limited discrepancy search is applied to explore the search tree with an increasing number of discrepancies, a discrepancy being a branch against the best reinsertion places, e.g. inserting a customer at its second cheapest position.

In LNS, neighborhoods are implicitly defined by *destroy* and *repair* methods (often heuristics), which are used to destroy part of the current solution and to restore the subsequent partial solution. The destroy method usually includes a component of stochasticity such that different parts of the solution are destroyed in every invocation of the method. The neighborhood $N(x)$ of a solution x is then defined as the set of solutions that can be generated by first applying the destroy method and then the repair method. Since the destroy method can destruct a large part of the solution, the neighborhood contains a large amount of solutions.

The steps of the LNS method are outlined in Algorithm 2. Three variables are included in the algorithm: x^b is the best solution observed so far during the search, x is the current solution, and x' is a temporary solution that can be discarded or promoted to the status of current solution. The function $d(\cdot)$ is the destroy method while $r(\cdot)$ is the repair method. More specifically, $d(x)$ returns a copy of x that is partially destroyed. Applying $r(\cdot)$ to a partly destroyed solution repairs it, i.e. it returns a feasible solution built from the destroyed one. Both destroy and repair methods can be implemented in different ways obeying different criteria. In step 2 the new solution is evaluated, and then the heuristic determines whether this solution should become the new current solution or should be rejected. The *accept* function can be developed in different ways. The simplest choice is to only accept improving solutions, but some works propose an acceptance criteria borrowed from SA [92], that is, accepting solutions that may be worse than the incumbent aiming to diversify the search. As it can be noticed in Algorithm 2, LNS does not explore the

Algorithm 2 Large Neighborhood Search

Initialization: Find an initial solution x and set $x^b \leftarrow x$ as the best solution found so far; choose a stopping condition.

Repeat the following sequence until the stopping condition is met:

1. $x' \leftarrow r(d(x))$
2. If $accept(x', x)$ then $x \leftarrow x'$
3. If $c(x') \leq c(x^b)$ then $x^b \leftarrow x'$

entire neighborhood of a solution, but merely samples this neighborhood [18].

The destroy method is a crucial component of LNS. The most important decision when implementing the destroy method is the degree of destruction: if only a small portion of the solution is destroyed then the goal of leaving local minimums through exploring large neighborhoods is lost. If a very large part of the solution is destroyed, then the LNS heuristic almost degrades into repeated re-optimization or a multi-start process. This can be time consuming or yield poor quality solutions depending on how the partial solution is rebuilt. Shaw [93] proposed to gradually increase the degree of destruction, while Ropke and Pisinger [92] choose the degree of destruction randomly at each iteration from a specific range according to the instance size. The destroy method must also be chosen such that the whole search space can be reached, or at least the interesting part of the search space where the global optimum is expected to be found. Therefore, it cannot focus on always destroying a particular component of the solution but must make it possible to destroy every part of the solution.

The selection of the repair method permits much more freedom when implementing a LNS heuristic. A first decision is whether the repair method should be optimal in the sense that the best possible full solution is built from the partial solution, or should be a heuristic assuming that one is satisfied with a good solution constructed from the partial solution. An optimal repair operation will be slower than a heuristic one, but may potentially lead to high quality solutions in a few iterations. However, from a diversification point of view, an optimal repair operation may not be attractive: only improving or identical-cost solutions will be produced and it can be difficult to leave valleys in the search space unless a large part of the solution is destroyed at each

iteration.

Intuitively, searching a very large neighborhood should lead to higher quality solutions than searching a small neighborhood. Nevertheless, in practice, small neighborhoods can provide similar or superior solution quality if embedded in a metaheuristic framework because they typically can be searched more quickly. Large neighborhoods generally lead to local solutions of better quality, but the search is more time-consuming. Hence, a natural idea is to gradually extend the size of the neighborhood, each time the search gets trapped in a local minimum.

3.3 Multi-objective optimization

Many real-world optimization problems, including the VRPTW, involve more than one objective to be either minimized or maximized. According to [16] and considering a minimization of each function, a Multi-objective Optimization Problem (MOP) can be formulated as:

$$\min f_i(x) = (f_1(x), f_2(x), ..., f_n(x)) \; s.t \; x \in X \tag{3.11}$$

where f is the vector of n objective functions to be optimized, $x = (x_1, x_2, .., x_m)$ the decision variable vector and X the feasible set of decision vectors.

In general, there is not a single solution optimizing all the objectives simultaneously. Instead, different solutions could be found with a trade-off among the different objectives. The concept of *domination* or Pareto Optimality is used to determine this set of optimal solutions. It is said that a solution $f_i(x^*)$ is *non-dominated* iff there is not another solution $f_i(^*)$ such that:

$$\forall i \in \{1, ..., n\} \, | f_i(x) <= f_i(x^*) \wedge \exists i \in \{1, ..., n\} \, | f_i(x) < f_i(x^*) \tag{3.12}$$

As equation 3.12 states, a solution $f(x)$ dominates another $f(x^*)$ if and only if $f(x)$ is better than $f(x^*)$ in at least one objective and not worse in the other ones. That is, a solution is Pareto optimal if there is no other one that improves at least one objective function without sacrificing the others. Therefore, the set of non-dominated points obtained determine the Pareto optimal solutions of the MOP.

3.3.1 Methods for solving multi-objective optimization problem

In general, solving a MOP involves helping a decision maker in the presence of multiple and often conflicting objectives and providing a solution or a set of solutions that satisfy the requirements. In this sense, the methods to deal with MOP can be classified according to the role of the decision maker in the decision process [94]. The more common classes are: *a priori*, *a posteriori*, and *interactive* methods. The *a priori* approach uses specific information about the relevance of the objectives and the user preferences before the solution process. As a result, one solution is found according to these preferences. In the *a posteriori* schema, a set of Pareto optimal solutions is generated and the preference information of each objective is used to select the most satisfactory one. Finally, in the *interactive* methods the preference information is updated during the solution process. Different examples of methods according to the above classification can be found in the literature, each having strengths and drawbacks. A good summary is presented in [94, 95, 96].

The advantage of the *a priori* approach is to produce a single compromise solution without requiring a further participation of the decision maker. One of the methods more widely used due to its simplicity is the *weighted* method [94]. It is precisely used in the mentioned deicing trucks scheduling [7] discussed in Section 2.3. It involves aggregating all the objectives into one composite function with different weights that are used to indicate the relative importance among the criteria. However, there are problems where expressing the preferences through values or correlate different objectives in a unique objective function may give inaccurate solutions [94, 97].

Other common algorithm is the $\epsilon-constraint$ method, where one of the objective functions is optimized and the others are transformed into constraints. In this case, the function to be optimized should be selected, as well as the upper bounds associated to each objective-constraint according to the user requirements. Comparing with the *weighted* method, defining bounds seems more evident to the decision maker than assuming weights for representing the relation between the objectives. On the other hand, promising solutions which are not within the defined boundary but very close are discarded.

In the *hierarchical* method, also known as *lexicographic* method, the objectives are ranked in decreasing order of priority. Then, the problem is solved optimizing only the most relevant objective function subject to the original restrictions. The value obtained is added as constraint in order to ensure the

best value for the most important function, and the next relevant objective is optimized. If not many objective functions are present in the problem and the preference between them is clear enough for the decision maker this is a quite simple and practical method to solve a MOP.

Proposed in 1955 by Charnes et al., *Goal programming* [98, 99] is one of the first methods specially developed for solving MOP. It is an extension of linear programming for dealing with multiple objectives [96]. In this case, the decision maker should specify a target value or a goal for each objective which represents the level of achievement required. Deviation variables are included in the model in order to minimize the absolute deviations between the targets and the objectives. When an order of preference between the objectives is indicated by the decision maker the method is called *lexicographic goal programming*. In this case, the resulting deviation of the target with the highest priority has more importance than the rest of goal variations. In contrast, if the direct relation of the objectives has more relevance to the decision maker, the deviations are associated with coefficients. These values are summed into a single function in a similar way to the *weight* method and the procedure is known as *weighted goal programming*. Goal programming is a popular method and has been successfully applied to solve real application problems [94]. Representing the MOP in terms of goals makes this methodology simple and easy to understand. However, the aspiration levels should be overoptimistic enough to produce Pareto optimal solutions [94].

The *a posteriori* approach uses the concept of Pareto dominance and provides a set of solutions to the decision maker. This schema makes the decision process more flexible and no preference requirements should be specified beforehand. In contrast, depending on the problem, it can be very expensive in computational time and having a large number of alternatives may create difficulties for making a choice. With the goal of finding a set of non-dominated points, basic and usually *a priori* methods can be used as *a posteriori* approach modifying the parameters. However, in the *weighted* method, the performance of the algorithm is highly affected in case of non-convex problems where obtaining the Pareto optimal points is not guaranteed [100]. In contrast, the convexity does not represent an obstacle for the ϵ-*constraint* procedure and it is commonly used in *a posteriori* way. The problem is solved with respect to one objective and at each iteration the value of the second objective is used as a constraint to limit the search space [76, 94, 100].

The use of Evolutionary Algorithms (EA) for solving combinatorial problems including MOP has received an increasing attention in last years. As

mentioned in Section 3.2, EA are population-based methods and work with a set of solutions at each iteration. This feature seems quite suitable for dealing with MOP due to several Pareto optimal solutions can be generated in a single round of the algorithm [101, 96]. Evolutionary computation is inspired by the Darwin theory of evolution: a population of individuals (set of solutions) are generated; the fittest (best solutions) are modified using recombination and mutation operators in order to produce solutions that increase the fitness (improve the objective function) [102].

Interactive approaches can be seen as an attempt to overcome the drawbacks of *a priori* and *a posteriori* methods. The participation of the decision maker in the optimization process avoids an early, usually inaccurate, specification of preferences. Moreover, this produces only relevant Pareto solutions, improving the performance of the algorithm. The interactive methods can be grouped into three classes depending on the way the preference information is indicated [95]: (i) trade-off information, (ii) reference points, and (iii) classification-based methods. In trade-off approaches such as *Z-W* [103], the decision maker provides a measure of how much is possible to sacrifice one objective with the aim to achieve an improvement in another objective. In the second group, desired values for the objective functions are procured at each iteration according to the accumulated experience along the search process. Finally, in classification-based method, the decision maker gives different kinds or classes of modifications in order to transform a current Pareto solution into the most preferred one, e.g which function needs to be improved, which one could be deteriorated or even wished improvement or deterioration [104, 105].

3.3.2 Multi-objective optimization problem and VRPTW

The field of MOP has seen a growing interest among researchers particularly in VRPTW problems. Different examples can be found in the literature applying either the most basic methods or sophisticated mechanisms. An overview of the research in this area is presented in [106].

Gambardella et al. [41] implemented the hierarchy method to minimize the number of vehicles and the traveling distance in this order of importance. In [14] (see Section 2.3), a similar approach is used to solve a multi-objective model for scheduling fueling vehicles. In both cases, the Ant Colony Optimization metaheuristic is adopted to perform the search.

A goal programming model is proposed by Ghoseiri and Farid [75] where also the vehicles required and the traveling distance are minimized, but in this

case with the same level of priority. A genetic algorithm is employed to solve the problem generating a range of Pareto solutions. Hong and Park [107] also used a goal programming model to minimize the total travel time and the total customer waiting time. A two-phase heuristic based on the cluster-first route-second procedure is introduced in order to improve the performance of the algorithm. Altering the aspiration levels of the goal, a set of solutions is obtained.

Liu et al. [108] proposed a multi-objective heuristic in three phases to minimize the traveling distance among the vehicles and to balance the workload and the delivery time. In fact, only the first objective is optimized and the second and third functions are converted into constraints defining the maximum value allowed for each of them.

In a soft time window context, Muller [76] used the $\epsilon - constraint$ method to minimize the total cost and the penalties associated with the violations of the time windows. Here, the total cost consists of the number of required vehicles and the total distance. It is optimized and the cumulative penalties is transformed into a constraint. At first, any violation is permitted, thus the solution with the worst cost is generally obtained. At each iteration, the penalty limit allowed is increased and the problem is solved with the new conditions producing a set of solutions.

Tan et al. [109] presented a hybrid multi-objective evolutionary algorithm (HMOEA), which combines evolutionary search with a local search process for dealing simultaneously with travel distance and fleet size. Hybridization between EA and other metaheuristics is included in [110] and [111]. In the former, the number of vehicles is minimized in a first step using evolution strategy and then the distance is minimized with Tabu Search. Simulated Annealing and evolution search operators are used in the second work in order to minimize the distance and the imbalance of the routes.

Chapter 4

Multi-objective global approach

In this chapter we describe our global approach to deal with the ground handling problem. Different activities and vehicles are considered in order to combine scheduling decisions about each resource in an overall planning.

As discussed in chapter 2, we address the scheduling of ground handling vehicles at a tactical level. With a tactical scope we have further available time for executing more sophisticated scheduling algorithms and find better solutions than at operational level. However, the scheduling process usually needs to be performed very close to the day of operation aiming to use more reliable expected information about number of passengers, gates, etc. Hence, we are concerned to obtain good solutions with a reasonable computational effort.

A decomposition schema has been adopted to model the problem and it is described in Section 4.1. In a first level, the time windows to begin the operations at each aircraft are calculated considering the duration of each task, the precedence constraints and the available time to complete the turnaround. Then, the vehicles that perform each task are scheduled independently, where a VRPTW problem is solved for each type of vehicle involved. Two procedures have been modeled to tackle each level and both formulations are presented in Section 4.2.

Regarding the solution method, this is depicted in Section 4.3. All the VRPTWs, one for each type of vehicle, are solved one after another following a specific sequence. Each routing problem is solved in two steps: a quick initial solution is obtained using the *Insertion Heuristics* method (see Section 3.2.1) and a CP-based VND-LNS methodology is applied after as a local search process see Sections 3.1, 3.2.2, 3.2.3). In addition, an updating time window

process is implemented in order to propagate the routing decisions made for each type of vehicle and avoid infeasible global solutions. As mentioned, we have tackled the ground handling problem as a multi-objective problem. A heuristic is proposed to improve the global solution in this decomposition context and to deal with the MOP. Modifying the sequence the sub-problems are solved it is possible to find approximations to the non-dominated solutions of this problem.

Finally, a variant of the proposed approach is introduced in Section 4.4 aiming to provide a more exhaustive method to find the Pareto solutions. The scheduling process has been carried out in two stages: solutions are obtained using only the I3 heuristic and just the most promising are improved with the local search process. In order to decide which solutions are the best, two strategies are proposed. In the first rule, solutions with the best global results according to the values of the objectives are selected. Solutions having the best resource planning for critical activities are preferred in the second rule.

4.1 Problem decomposition

As mentioned, we aim at scheduling ground handling services as a whole taking into account different activities and vehicles. At each aircraft, several operations related by precedence constraints are executed during turnaround and they have to be completed within a predefined time (see Chapter 2). At the same time, each operation is performed independently employing specific resources that are not used by other activities, e.g. one specific type of vehicle, independent vehicle station, etc. So, it is possible to decompose the problem and schedule each operation separately but taking into account the precedence relations. This way, the global perspective of the approach is kept and the whole process can be optimized. Also, we want to tackle the problem as a whole designing and implementing an approach for solving it in a reasonable time.

Representing and solving this multiple VRPTW by a unique model that cover all the operations could be very expensive in effort and computational time. Then, decomposing the problem permits to simplify the model and the solution method. With this goal in mind, temporal restrictions to serve each operation due to the defined turnaround time and precedence constraints have to be tackled in a global way. That is, time windows to begin the operations are calculated considering all the operations included and ensuring that routing

solutions for each type of vehicle can be integrated to obtain a feasible complete solution. A routing problem for each type of vehicle is solved separately using the time windows obtained. Available vehicles from the associated fleet are assigned to service the corresponding operations and the corresponding routes are generated.

This decomposition schema is based on the workcenter-based decomposition for the Job Shop Scheduling [17]. Due to each operation has to be performed by just one type of vehicle, this decomposition method can be applied to our problem in a natural way. A workcenter is a group of machines performing similar operations. In this approach, the overall problem is broken down into workcenter based sub-problems and they are solved independently. The operations of a sub-problem are related to other sub-problem operations by the precedence restrictions. Each time a sub-problem is scheduled, new constraints for the other operations are generated. Thus, a method that integrates the sub-solutions and keeps the consistency of the overall solution is needed. In our problem, each type of vehicle could be viewed as a workcenter, and the vehicles available in the fleet as machines. Therefore, instead of solving a global VRPTW, it is possible to solve a local VRPTW for each of them. In addition, scheduling each type of vehicle separately permits the development of specific methods to tackle special features according to the operation they perform.

The decomposition schema adopted is outlined in Figure 4.1. A procedure we called *Temporal Constraints Level Procedure (TCLP)* was defined to satisfy the temporal restrictions. For each type of vehicle, one VRPTW sub-problem is identified and modeled using the *Routing Level Procedure (RLP)* also defined. The $F1$ and $F2$ are the optimization objectives. Formulation of both procedures as well as the objective functions are presented in section 4.2.

4.2 Problem formulation

The main features of the proposed approach are modeled and implemented in CP. As mentioned in Section 3.1, problems in CP are expressed by means of three elements: variables, their corresponding domains, and the constraints relating these variables. Therefore, the problem formulation is defined according to these elements.

The parameters of the ground handling problem are described as follow:

- $N = \{1, ..., n\}$: is the set of scheduled aircraft performing a turnaround.

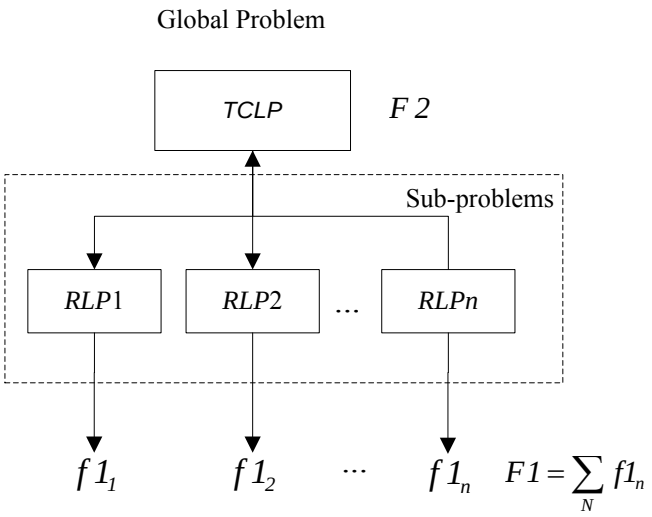

Figure 4.1: Problem decomposition schema

For each aircraft $n \in N$ the STA_n and STD_n are the scheduled time of arrival and departure, respectively.

- $A = \{1, ..., a\}$ is the set of aircraft types where $a_n \in A$ represents the type of the aircraft n

- $\Gamma = \{1, ..., \gamma\}$ is the set of parking positions, and $\gamma_n \in \Gamma$ is the stand where the aircraft n is parked during the turnaround. π_{ij} $\forall i, j \in \Gamma$ is the traveling cost between stands and $\pi_{0\,i}$ is the traveling cost between stand i and the vehicle depot.

- $T = \{1, ..., nt\}$ are the tasks to be performed on the aircraft at its parking position. According to the aircraft type $a \in A$, each task $t \in T$ has a duration δ_{ta}, a demand ρ_{ta}, and precedence restriction rules Ψ_{ta}, which represent the set of tasks that need to be finished before task t starts. Each task has to be served by one type of vehicle.

- $VT = \{1, ..., nvt\}$ describe the different type of vehicles which serve the tasks, where each $vt \in VT$ has its own homogeneous fleet $M_{vt} = \{1..m_{vt}\}$ with a capacity Q_{vt}.

Then, there are $O = T \times N$ operations to be performed by vehicles from

VT. An operation o_{tn} is a task t performed at an aircraft n according to a_n, the type of aircraft of n.

Temporal Constraint Level Procedure (TCLP)

In this level, precedence and temporal restrictions at each aircraft are modeled in CP. With this procedure the earliest and latest start time for each operation are obtained. Let the variable τ_{tn} be the start time of each operation o_{tn} with a discretized initial domain $\tau :: [STA_n..STD_n]$. Precedence restrictions are described by the following constraint:

$$\tau_{tn} \geq \tau_{t'n} + \delta_{t'a_n} \; \forall t \in T \;\; \forall t' \in \Psi_{ta_n} \, \forall n \in N \tag{4.1}$$

Equation 4.1 ensures that precedence relations among the tasks are fulfilled according to the type of aircraft they belong to. Specifying the domain of τ_{tn}, we guarantee that operations are performed within the available turnaround time. This can be done either with the domain definition or with an extra constraint, (4.2):

$$STA_n \leq \tau_{tn} \leq STD_n \; \forall t \in T \; \forall n \in N \tag{4.2}$$

When restriction 4.1 is propagated, the domain of the start time variable of each operation is reduced such that $\tau_{tn} :: [est_{tn}..lst_{tn}]$ where est and lst are the lower and upper bounds of τ and represent the earliest and latest start time of the operation, respectively. An example is presented in Figure 4.2. Suppose an aircraft where two operations should be performed, o_{11} and o_{21} such that o_{11} has to be finished before o_{21} could start, that is $t_{21} \geq t_{11} + \delta_{1a}$. At first the domain of both operations is $[STA..STD]$. After applying 4.1, feasible time windows are obtained.

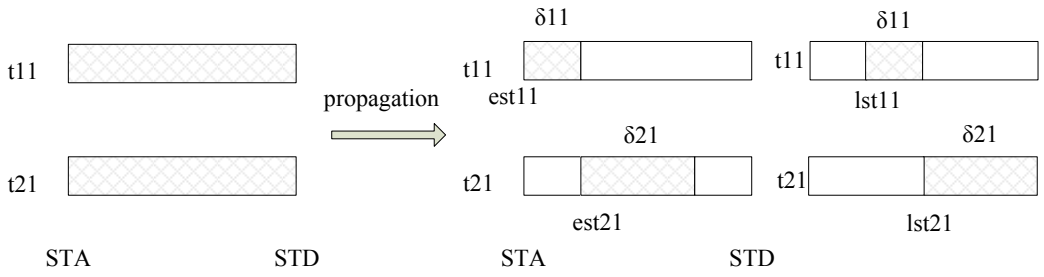

Figure 4.2: Propagation of the domain after applying precedence constraints

TCLP is also used to update the time windows with the aim to propagate the routing decisions made for each type of vehicle. Along the operations scheduling process, these time windows are modified due to the precedence restrictions. As a result of routing the vehicles separately, an explicit update process of the time windows is required to avoid inconsistency among the sub-problems. Suppose the same operations o_{11} and o_{21} on the same aircraft. The difference between est_{11} and est_{21} is the duration of o_{11}, as well as between lst_{11} and lst_{21}. Suppose also that the type of vehicle which serves o_{11} is routed first and o_{11} is scheduled such that $\tau_{11} \geq est_{11}$. The value of est_{21} is now $\tau_{11} + \delta_{1a}$. Then, the time window of o_{21} has to be reduced. Otherwise, when the type of vehicle associated with o_{21} is routed with the original time window, an infeasible solution might be obtained. Notice both the earliest and the latest start time could be reduced. For example, if o_{21} is solved first, the lst_{11} will be $\tau_{21} - \delta_{1a}$. The strategy followed to update the time windows and ensure the consistency of the solutions is further explained in Section 4.3.1.

Routing Level Procedure (RLP)

The CP model corresponding to each VRPTW sub-problem is based on Kilby and Shaw formulation [112]. This model represents a basic VRPTW that can be enriched to cope with the particularities of each ground handling vehicle type. Taking advantage of CP flexibility new constraints could be introduced with minimum modifications in the model and the solution process.

When the time windows for each operation are calculated, a sub-problem is identified for each $vt \in VT$. We get the set of operations to serve by each vt, $O_{vt} = \{o_1, ..., o_n\}$, as well as the duration d_o and the requirements of goods r_o for each $o \in O_{vt}$ according to the task and the aircraft type where the operation takes place. Thus, this set of operations represents the set of visits to be performed by vehicles. There is one visit per operation plus two special visits per vehicle. This special visits describe the depot from which the vehicles start and finish their routes. Due to each routing problem is solved separately and in order to simplify the notation, we identify the set O_{vt} as $O = \{1, .., n\}$. Let $V = O \cup F \cup L$ be the set of visits which is described by the following subsets:

- $O = \{1, .., n\}$ represents the operations, i.e the aircraft to be serviced.
- $F = \{n + 1, .., n + m\}$ is the set of first visits, i.e the depot in outbound trips.

- $L = \{n+m+1, .., n+2m\}$ is the set of last visits, i.e the depot in inbound trips.

Furthermore, m is the number of vehicles needed to serve all the operations from the fleet $M = \{1..m\}$ with capacity Q. It should be noticed that m is a parameter at this level. Its value is determined as a solution of the first stage procedure for solving the routing problem (details are provided in section Section 4.3).

The indexes f_k and l_k express the first and last visit of a vehicle $k \in M$ respectively:

- $f_k = n + k$, $f_k \in F$ is the first visit of vehicle k
- $l_k = n + m + k$, $l_k \in L$ is the last visit of vehicle k

In order to model the routes, the following variables are defined:

- p_i, $\forall i \in V - F$ is the direct predecessor of a visit i with domain $p :: [1..n+m]$

Besides predecessors, a variable that represent the customer s successor in a route is defined. In principle, only one of both variables is required to model routes. Nevertheless, this redundant formulation allows making additional inferences and prune the search tree in a notable way.

- s_i, $\forall i \in V - L$ represents the direct successor of a visit i with domain $s :: [1..n, n+m+1..n+2m]$

By convention, the first visit of a vehicle has as predecessor the vehicle s last visit such that $p_{f_k} = l_k$, $\forall k \in M$. Similarly, each last visit of a vehicle has as successor the vehicle first visit $s_{l_k} = f_k$, $\forall k \in M$.

In order to restrict a visit to have one and only one predecessor and successor the *difference constraints* (4.3) and (4.4) have been introduced. That is, these inequalities force predecessor and successor sets to contain no repetitions.

$$p_i \neq p_j \quad \forall i, j \in V - F \wedge i < j \tag{4.3}$$

$$s_i \neq s_j \quad \forall i, j \in V - L \wedge i < j \tag{4.4}$$

In addition, the *coherence constraints* (4.5) and (4.6) are established to keep the consistency between the successor and predecessor variables.

$$s_{p_i} = i \quad \forall i \in V - F \tag{4.5}$$

$$p_{s_i} = i \quad \forall i \in V - L \tag{4.6}$$

Restrictions (4.5) and (4.6) link the successor and predecessor concepts, such that i is the successor of its predecessor and i is the predecessor of its successor, respectively.

- $v_i \; \forall i \in V$ is the vehicle which perform each visit i with domain $v :: [1..m]$

The variable v_i is established to cope with multiple vehicles. Also, the following *path constraints* are included in order to guarantee that all visits are performed by the same vehicle along the route: constraints (4.7) and (4.8) ensure a visit, its predecessor and successor, are assigned to the same vehicle.

$$v_i = v_{p_i} \quad \forall i \in V - F \tag{4.7}$$

$$v_i = v_{s_i} \quad \forall i \in V - L \tag{4.8}$$

Likewise, the first and last visit of this vehicle are associated with equation (4.9).

$$v_{f_k} = v_{l_k} = k \quad \forall k \in M \tag{4.9}$$

Another two sets of variables are defined to represent cumulated capacities and the time for each visit:

- t_i, $\forall i \in V$ is the time when the visit i is performed with domain $t :: [est_i..lst_i]$.

- q_i, $\forall i \in V$ is the accumulated capacity after each visit i, with domain $q :: [0..Q]$

Notice the *est* and *lst* values are those obtained from the *TCLP* procedure. The domain of the time variable determines the time window of the customer and is equivalent to *time windows constraints* (4.10).

$$est_i \leq t_i \leq lst_i \quad \forall i \in V \tag{4.10}$$

Equation (4.10) as well as the time domain, specifies that customer i must be visited between times est and lst. In general, the concept of *scheduling horizon* is defined by the time windows of the depots.

To ensure the temporal precedence in a route, *time constraints* (4.11) and (4.12) are enforced. The first equation states that the time to visit a client i is at least the time its predecessor is visited plus the sum of traveling time ($\pi_{p_i i}$) from p_i to i and the duration of the service (d_{p_i}) in the predecessor. In a similar way, the time to visit a client is at most the time its successor is visited minus the sum of traveling time ($\pi_{i s_i}$) from i to s_i and the duration of the service (d_{s_i}) at s_i. Notice (4.11) and (4.12) are inequalities because waiting time is normally allowed.

$$t_i \geq t_{p_i} + \pi_{p_i i} + d_{p_i} \quad \forall i \in V - F \tag{4.11}$$

$$t_i \leq t_{s_i} - \pi_{i s_i} - d_{s_i} \quad \forall i \in V - L \tag{4.12}$$

The *capacity constraints* (4.13) and (4.14) are defined to maintain the load in the vehicles at each point in their route. The amount of goods picked up ($r_i > 0$) or delivered ($r_i < 0$) in the route are counted to keep the load along the route.

$$q_i = q_{p_i} + r_i \quad \forall i \in V - F \tag{4.13}$$

$$q_i = q_{s_i} - r_{s_i} \quad \forall i \in V - L \tag{4.14}$$

The domain of the variable q_i limits the accumulated capacity demand according to the capacity of the vehicle Q. This can also be restricted using an specific constraint such that:

$$0 \leq q_i \leq Q \quad \forall i \in V \tag{4.15}$$

In addition, we state the objective function of the routing problem. As mentioned, one of the optimization goals is serving the operations as early as possible. Therefore, the objective function aims at minimizing the total difference between the earliest possible time a vehicle could perform each visit and the corresponding earliest service time. Let (4.16) be this difference, that is, the client waiting time, the objective function for the routing problem is formulated as (4.17).

$$w_i = t_i - est_i \tag{4.16}$$

$$\min \sum_{i=1}^{n} w_i \tag{4.17}$$

Multi-objective functions

Finally, the objective functions of the global problem are formulated. The first criterion aims at improving the robustness of the solution minimizing the operation waiting time. As a result of the decomposition, the operation waiting time is calculated by the *RLP* with the updated time window (if it is reduced). If the size of the time window becomes smaller, probably the waiting time would be also smaller, although it does not contribute to the solution robustness. So, in order to obtain a more robust scheduling, operations should be performed as early as possible within their original time windows. Therefore, our first objective aims at performing operations as soon as possible through two arguments: minimizing the total operation waiting time (4.16) and the reduction of the time windows.

Let Δ_i be the reduction of the time window defined in (4.18), where $oest_i$ and $olst_i$ mean the original values of the time windows obtained from the *TCLP*, i.e., the very earliest and latest start time for each operation.

The w_i is the client waiting time set in equation (4.16). An aggregate function $f1$ is defined in order to describe how early operations are performed at each routing problem:

$$\Delta_i = (oest_i - est_i) + (olst_i - lst_i) \tag{4.18}$$

$$f1_{vt} = \sum_{i=1}^{n} (\Delta_i + w_i) \;\; \forall i \in O_{vt} \tag{4.19}$$

Thus, the first objective $F1$ is defined as the total $f1$ value of each type of vehicle:

$$F_1 = \min \sum_{vt \in VT} f1_{vt} \tag{4.20}$$

The goal of the second objective is to minimize the completion time of the aircraft turnaround. Let τ_{nt} be the start time of the last operation on each aircraft, i.e. the *pushback*. The second objective $F2$ of this problem is therefore defined as:

$$F_2 = \min \sum_N \tau_{nt} \tag{4.21}$$

4.3 Solution method

In this section, we describe the algorithms developed for solving the ground handling problem modeled using the decomposition schema outlined in Section 4.1 and formulated in Section 4.2.

Most workcenter-based decomposition methods are solved using the Shifting Bottleneck procedure [17] developed by Adams et al. [113]. This decomposition heuristic was originally implemented for the classical job shop scheduling problem and then extended to model other versions like the sequence-dependent times [114] and the workcenter problems [17], also called parallel machine problem. At each round, a critical unscheduled sub-problem according to the optimization criterion is identified and solved as a one-machine (or workcenter) scheduling problem. Using this result, each sub-problem solved in the previous iterations is re-optimized by solving again a one-machine problem, while the rest of machines already scheduled remain fixed. This re-optimizing cycle is repeated a number of times modifying the order in which the machines are solved.

The Shifting Bottleneck is computationally intensive and involves solving many single machine scheduling problems [115]. Applying this procedure in our particular case, where each sub-problem is a VRPTW, can lead to long execution times and becomes impractical to solve the problem. Thus, we followed a similar schema but combining two processes in order to obtain a complete solution at each iteration.

Two important aspects have to be considered to solve this problem as a result of the decomposition applied: (i) to avoid an infeasible global solution and (ii) to improve the global solution obtained. In the first developed process, which we call *Solving Process* (SP), all sub-problems are solved one after another given a predefined sequence. Each time a sub-problem is solved, the time windows of the remaining sub-problems are updated in order to keep the consistency among the sub-solutions. The *SP* is embedded in an iterative schema which we call *Sequence Iterative Method* (SIM) and is further described in Section 4.3.4. The goal of this second process is to improve the overall solution while dealing with the multi-objective optimization problem defined. The sequence for solving the sub-problems is modified at each iteration according

to the solution obtained and the *SP* is repeated with the new sequence. The relation between *SP* and *SIM* method is summarized in Figure 4.3.

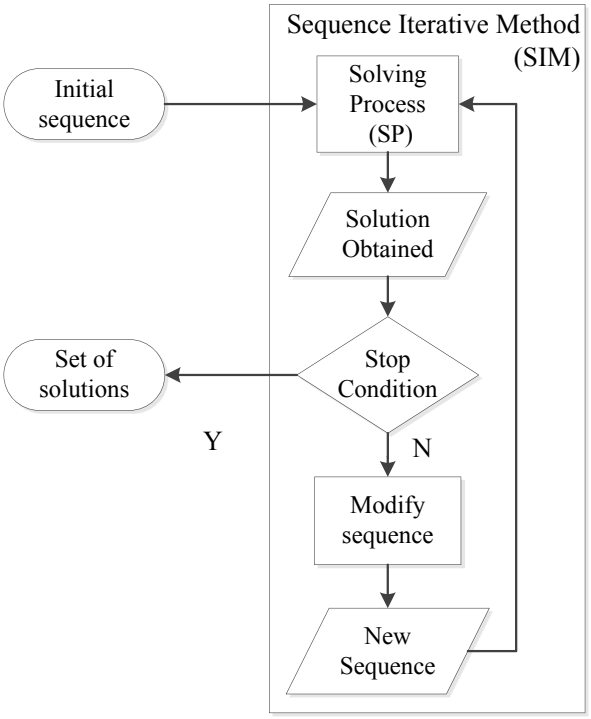

Figure 4.3: Solution method schema

Solving Process

In the *SP*, the problem is solved according to an specific sequence. This sequence defines the order in which the type of vehicles are scheduled, that is, the order in which the routing problems associated to each type of vehicle are solved. A schema of this process is shown in Figure 4.4.

First, the *TCLP* is used to find the initial time window to begin each operation. Time windows to begin the operations are calculated considering the duration of each task, the precedence constraints and the available time to complete the turnaround at each aircraft as it is described in Section 4.1. Time windows are obtained through the domain reduction provided by the propagation mechanism of CP. Then, the sub-problems associated to each type of vehicle are identified. This means operations performed by the same

type of vehicle are grouped in order to solve each VRPTW separately. Then, sub-problems are solved by means of the *RLP* following the given sequence.

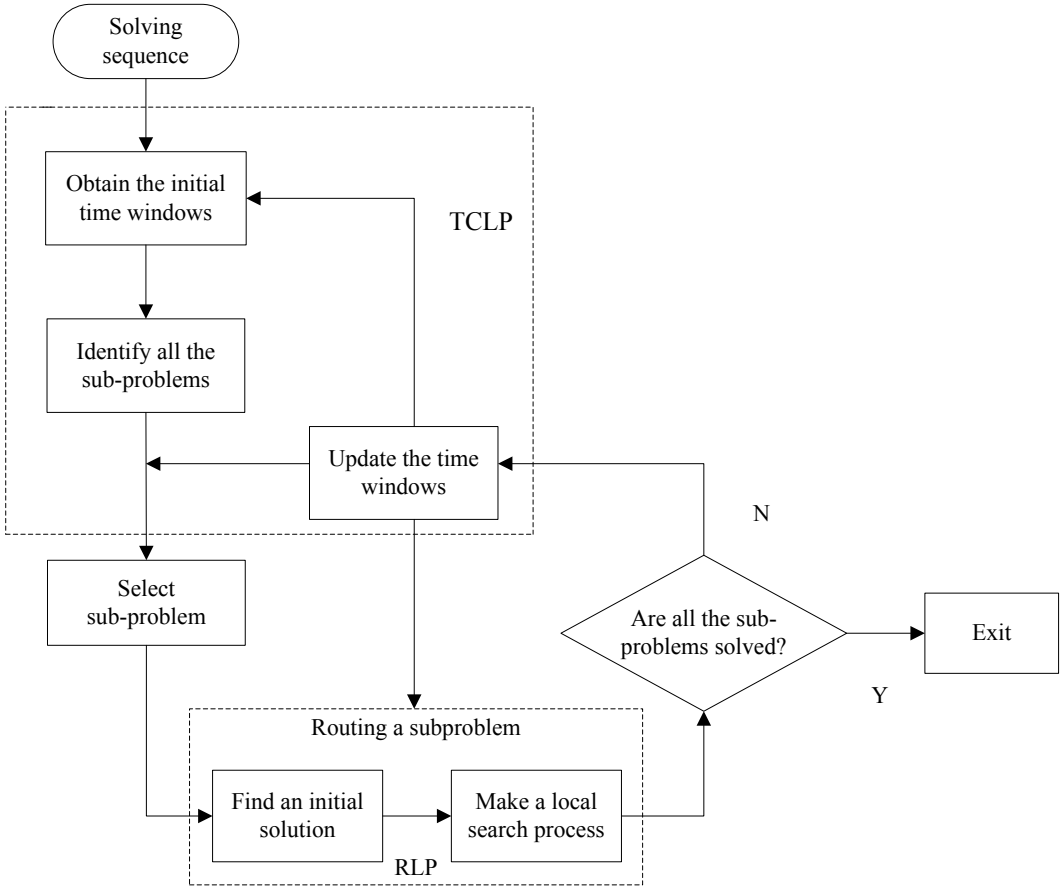

Figure 4.4: Flow diagram for the *SP*

The *RLP* procedure is developed in two steps, as will be further described in Section 4.4. At the first step, a well-known route construction heuristic is used to obtain a reasonable good initial solution: the Insertion Heuristic I3 method (see Section 3.2.1). In order to use the I3 to solve our specific problem, the parameters of the heuristic have been tuned and are introduced in Section 4.3.2. The number of vehicles obtained in this step is taken as an upper bound of the vehicles needed to serve the operations. Imposing this value as the size of the available fleet, a CP local search process [18] is applied in the second step. This methodology is described in Section 4.3.3. The aim of this step is to improve the initial solution by minimizing the operation waiting time

according to the formulation of the *RLP* procedure given in Section 4.2.

After solving a sub-problem, an explicit process to update the time windows is needed to ensure consistency with the other sub-problems, as explained in Section 4.2. Once again, taking advantage of CP propagation through the *TCLP*, a simple strategy described in 4.3.1 is applied to keep this consistency. Finally, when all the sub-problems are solved, the process is stopped.

4.3.1 Update time windows

This step aims to ensure a feasible complete solution for the global problem. Due to precedence constraints between operations, the decomposition process can lead to obtain an infeasible global solution. An explicit time windows updating process is defined in order to integrate the sub-solutions. It is outlined in Algorithm 3.

Let S be the set of routing sub-problems to be solved, where $|S| = |VT|$. F is the set of sub-problems already solved such that $F \subseteq S$ and $S \setminus F$ are the sub-problems not solved yet. τ_s is the set of start times corresponding to the visits in sub-problem s, $s \in S$. τ_F and $\tau_{S \setminus F}$ are the set of start times of sub-problems in F and $S \setminus F$, respectively.

To ensure the coherence among the sub-solutions, each time a sub-problem is solved, the scheduled decisions are propagated to the rest of sub-problems not solved yet, keeping already scheduled sub-problems fixed. At first, no sub-problem has been solved. The very earliest and latest start time of each operation is obtained by means of the *TCLP*. When a sub-problem is scheduled, the start times of the operations belonging to this sub-problem are calculated. Afterward, the *TCLP* is recalled with the start times of the sub-problems already solved. These values are propagated to the operations of the unscheduled sub-problems and their time windows are updated. This process is repeated for each sub-problem, always keeping the preceding scheduling decisions. Therefore, infeasible intermediate solutions are avoided using this strategy.

Time window reduction depends on the order sub-problems are solved and affects the quality of the global solution. In fact, in the decomposition procedures based on Shifting Bottleneck, determining the next machine to be scheduled is one of the more important steps. According to [17] the sequence in which machines are included in the partial schedule can reduce the re-optimization process without loss in solution quality. Then, the iterative process *SIM* is developed aiming at improving the solution by modifying the order the sub-problems are solved.

Algorithm 3 Solving Process (SP)

Initialization:
- Set $\tau :: [STA_n..STD_n]$
- Obtain the initial time windows by means of $TCLP \Rightarrow \tau_{tn} :: [oest_{tn}..olst_{tn}]$
- Set $F \leftarrow 0$

Repeat until $|F| = |VT|$

1. Choose $s \in S$. τ_s is a subset of τ_{tn} which define the start times of operations t included in sub-problem s.
2. Sub-problem solution by means of the *RLP*
   (a) Obtain an initial solution for s using the *I3 Insertion Heuristic*
   (b) Apply the CP-based local search process (Algorithm 4)
3. Time windows update process for the sub-problems in $S \setminus F$ by means of the *TCLP*
   (a) $\tau_s \leftarrow$ start times obtained solving s with *RLP*
   (b) Set $F \leftarrow F \cup \{s\}$
   (c) $TCLP(\tau_F) \Rightarrow \tau_{S \setminus F} :: [est_{tn}..lst_{tn}]$

4.3.2 Initial solution

As discussed in Section 3.2.1, one of the more interesting features of the *Insertion Heuristic* method is that both time dimension and distance are included in the route building process. Moreover, the parameters defined for each criterion give a considerable degree of freedom to adapt the solution method for solving different problems. This parameters can be adjusted to obtain solutions that improve classical objectives like distance as well as more time-oriented goals. In this case, the Solomon study [19] was focused on the total scheduling time and the vehicle waiting time in addition to the distance and the number of vehicles, being the last one the main objective.

Minimizing the operation waiting time is the objective of the routing problem in our case. So, the parameters have been adjusted for tuning the heuristic to this optimization objective. It should be remarked that we are also interested in obtaining good solutions regarding the number of vehicles. Thus, we

have assigned the weights to the parameters in order to accommodate the algorithm to improve the operation waiting time without sacrificing the number of routes needed to serve the clients.

In the *I3* method the same cost function is used to build the routes i.e to select the best place and the best client to insert in this place. This cost function c_1 formulated in Section 3.8 involves three criteria: (i) the distance, (ii) the schedule time to begin the visit, and (iii) the urgency of serving the customer (see equations (3.4), (3.5), and (3.9), respectively in Section 3.2.1). Each criterion has an associated weight which defines the relevance of the goal in the cost function: α_1, α_2, and α_3, respectively, where $\alpha_1 + \alpha_2 + \alpha_3 = 1$, and α_1, α_2, $\alpha_3 \geq 0$.

The second rule seems the goal we would prioritize to benefit the operation waiting time. However, making α_2 too high can increase significantly the number of routes required. In order to get a compromise between minimizing operation waiting time and the number of vehicles, parameters α_1 and α_2 are set to have similar weights. Regarding α_3, this parameter contributes to reduce the vehicle waiting time [19]. Due to our main objective can be critically affected by this feature, we gave α_3 the lowest importance in the cost function.

With these ideas in mind, we defined an interval for each parameter, such that: $\alpha_1 = [0.4, 0.5]$, $\alpha_2 = [0.4, 0.5]$, and $\alpha_3 = [0.01, 0.1]$ and tested the algorithm over the different combinations. The best results are obtained with a low α_3. In general, the customer waiting time begins to rise when α_3 takes values greater than 0.05 and its influence is bigger in case of operations with larger time windows. Combination $(0.49, 0.49, 0.02)$ gives the best result for most of the instances tested (see Section 5.2). Therefore, we have selected these values to specify the parameters. Notice that this procedure gives the initial solution of the routing problem and a local search process is then executed in order to improve it.

Furthermore, we have adjusted the initialization criterion to minimize the operation waiting time. As described in Section 3.2.1, initializing the route is the first step of the *Insertion Heuristic* method. According to Solomon [19], the rule to select the first client has an appreciable impact on the quality of the solution. Three main criterion were proposed: (i) the farthest unrouted customer, (ii) the unrounted customer with the earliest deadline, and (iii) the customer which the minimum weight combination between distance and time. In our problem, the client having the earliest start time to begin the service was inserted first at each new route generated.

4.3.3 Local search process

A hybrid methodology that combines modeling and constraint propagation advantages of CP with local search methods is applied to improve the initial solution. The drawback of local search strategies is they are likely to get trapped in a local minimum when no better solution can be found in a neighborhood. To overcome this obstacle, the methodology employs VNS as a metaheuristic to guide the search, specifically the VND version explained in Section 3.2.2.

During local search, a sequence of moves is performed to transform a current solution into another solution in its neighborhood. This process may violate a basic operating principle in CP, the so-called *chronological backtracking*. If a worse solution or even an infeasible one is obtained during search, some decisions made can be always undone. Under chronological backtracking, decisions must be undone in the reverse order they were made. So, in order to undo the last decision made during search, all operations performed since that time would have to be undone too, which would be unacceptable from a local search point of view. In order to avoid this undesirable situation the changes made at the local search level are embedded in CP using the concept of operators. An operator determines a neighborhood and a set of solutions can be found when the operator is applied to one solution [80].

Many of these operators are based on Large Neighborhood Search (LNS), described in Section 3.2.3, which destroys and repairs the solution in order to re-optimize parts of the problem. Destroy in this case means identifying a set of customers to be removed from a solution. Repair refers to finding a better way to reinsert these customers into the partial solution. One iteration of removal and re-insertion can be considered as a move to find a new solution in the neighborhood. If the re-insertion process is successful and results in a lower cost than the best solution found so far, this new solution is kept as the current one.

Employing LNS with a VND, the methodology allows the exploration of the search space in a systematic way as depicted in Algorithm 4 proposed by [18]. Using VND, the algorithm moves from one operator to the next in order to escape from local minima. Anytime an improving solution is found, the process is reset to the first operator; otherwise, the algorithm changes to the next operator. Operators are used to destroy the solution and a CP-based *branch-and-bround* (BB) procedure is employed as the repairing method. Constraint propagation provides efficiency to this process pruning the search

Algorithm 4 VND-LNS

Initialization:
- Select the set of operators O_k, for $k = 1, ..., k_{max}$, that will be used;
- Find an initial solution x (using *I3 Insertion Heuristics*)
- Choose a stopping condition (limited execution time) - Set $k \leftarrow 1$.

Repeat the following steps until $k = k_{max}$:

1. Apply the O_k operator to obtain the solution x' (LNS)
2. If the solution x' is better than x ($f(x') < f(x)$), set $x \leftarrow x'$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$

tree each time the upper bound is updated when a better solution is found. Due to this process is an exact repairing procedure, a limited execution time is established to avoid excessive computational time. Also here, the combination of VND and LNS plays an important role because different neighborhoods can be explored and revisited iteratively with improved upper bounds [18].

RPOP and SMART Operators

Rousseau et al. [80] propose a set of constraint-based operators for searching in large neighborhoods with the aim to solve the VRPTW. They use the VND strategy to take advantage of the different solutions generated by the different operators. According to this work, the neighborhood structure defined by used operators should be different in order to succeed with the VND. Following this idea, we have implemented: the *Random Pivot OPerator (RPOP)* proposed by [18] in which individual customers are removed and re-inserted and the *SMAll RouTing (SMART)* [80] concerning arc exchanges. The VND schema using both operators is outlined in Figure 4.5.

Two key aspects should be considered when implementing operators: (i) how to select customers, and (ii) how many, i.e. the size of the neighborhood. Concerning the first point, a typical strategy is removing clients that are related according to some criteria, i.e geographic proximity. As for the second consideration, small neighborhoods are usually preferred to large ones due to computational time reasons. Also, small neighborhoods can provide similar or superior solution quality if they are embedded in a metaheuristic framework

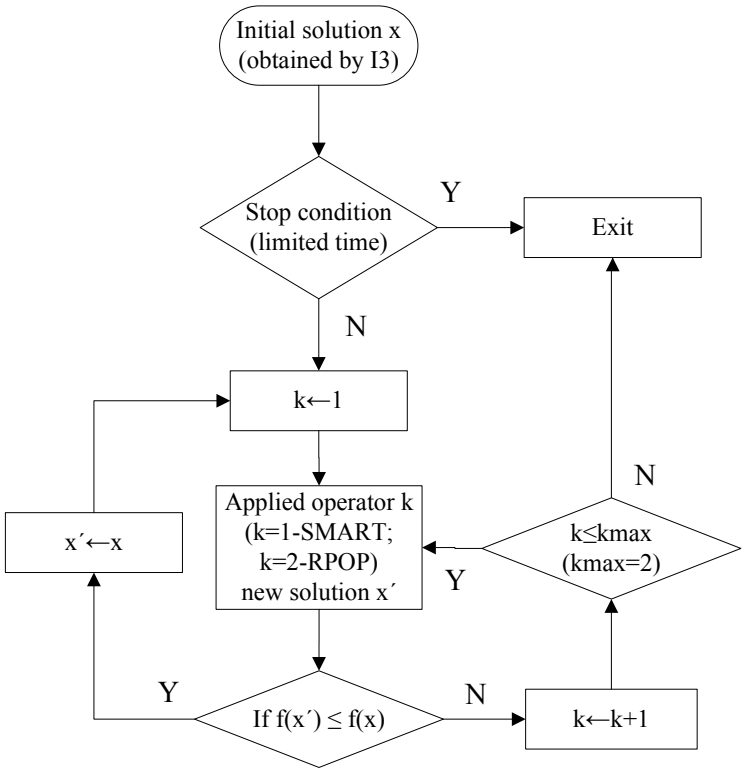

Figure 4.5: Flow diagram of the VND schema using operators RPOP and SMART

[18].

In RPOP, which is described in Algorithm 5, a pivot customer is randomly selected and removed from the solution. Then, a set of the nearest customers according to their geographic proximity is also removed forming a hole around the pivot [18]. In our particular problem, different operations (clients) can have the same parking position due to the length of the schedule time horizon. Hence, establishing temporal criteria seems more suitable to associate visits than using geographical rules. In the ground handling problem, the time windows to serve the operations are generally tight, particularly when the activities have precedence restrictions. So, we have defined the closeness of the time windows as the proximity metric of the RPOP.

Regarding the number of clients, RPOP operator is defined such that the number of visits removed is gradually increased each time the search gets trapped in a local minimum. One single pivot is again selected when an im-

Algorithm 5 RPOP LNS-based operator

Initialization:
- Select a maximum number of pivots k_{max} to be used;
- Select the number of customers l around each pivot to be removed;
- Let x be the initial solution;
- Set $k \leftarrow 1$.

Repeat the following steps until $k = k_{max}$:

1. Choose randomly k pivots from the solution $x \Rightarrow x_k$
2. Choose the l closest neighboring visits around each pivot $k \Rightarrow x_l = \bigcup_k x_{l_k}$
3. Assign the partial labeling $x' = x \setminus \{x_k \cup x_l\}$
4. Repair the partial solution x' using branch-and-bound during t_{max} time to get x''
5. If the solution x'' is better than x ($f(x'') < f(x)$), set $x \leftarrow x''$ and $k \leftarrow 1$; otherwise, set $k \leftarrow k + 1$

provement is found. That is, a VND exploration strategy is adopted to determine the size of the neighborhood. An upper limit on the number of pivots is defined to avoid exploring neighborhoods that are too large. We selected the same strategy in our case.

In the SMART operator, outlined in Algorithm 6, sequences of arcs in different routes are removed instead of customers. The incomplete solution generated by the SMART operator can be viewed as small VRPTWs. In this case, a set of consecutive arcs is selected from each route and first and last customers of the removed sequences are considered the new depots. Also here, the sequences of customers to be removed from each route need to be close in order to ensure feasible movements. First, a random primary pivot is identified and a certain number of clients after and before the pivot are disconnected making a hole in its route. Then, a set of secondary pivots is selected, that is, the one customer in each other route that can be visited in that hole making a minimal detour.

In relation to the neighborhood dimension, this operator is more likely to produce large neighborhoods than the RPOP. Rousseau et al. [80] suggest to

use Limited Discrepancy Search (LDS) [116] for repairing the solution. LDS is a form of incomplete search where only the most promising values on the domain are tried during the exploration. Considering that a value ordering heuristic has been applied (see Section 3.1), the best value correspond to the left-hand branch of the search tree, the second preferred to the next branch and so on. LDS explores the tree according to a number of defined discrepancies which means the number of values tried besides the first. With LDS, large search spaces can be explored more quickly without compromising notably the quality of the solution.

The *most-constrained* heuristic described in Section 3.1.3 has been adopted to repair the solution in both operators.

Algorithm 6 SMART LNS-based operator

Initialization:

- Select the number of predecessors p and successors s to be removed
- Let x be the initial solution;

1. Choose randomly one *primary pivot* P from the solution $x \Rightarrow x_P$
2. Choose p and s customers before and after P
3. Assign the partial labeling $xp = x \setminus \{x_P \cup x_p \cup x_s\}$
4. Obtain the set of *secondary pivot* SP to be removed $x \Rightarrow x_{SP}$
5. Choose p and s customers before and after each $sp \in SP$
6. Assign the partial labeling $x' = xp \setminus \{x_{SP} \cup x_{SP_p} \cup x_{SP_s}\}$
7. Repair the partial solution x' using branch-and-bound and LDS with d_{max} discrepancies during t_{max} time to get x''
8. If the solution x'' is better than x ($f(x'') < f(x)$), set $x \leftarrow x''$

4.3.4 Sequence Iterative Method (SIM)

The ground handling problem has been defined as a particular Multi-Objective Problem (MOP), in particularly as a bi-criteria optimization problem. The

first criterion relates to the quality of the local routing decisions, and also depends on the reduction of the time windows along the process. The second criterion can be seen as a global objective; minimizing the completion time of the turnarounds. The iterative process was implemented to improve the global solution as a result of the decomposition regarding the two objectives defined. In our particular case, tackling the vehicle scheduling problem as a MOP contributes to the global approach of the ground handling problem, although it is not easy to define the relation and the importance between the objectives. For that reason, we aim to obtain a set of solutions with a trade-off between the objectives in order to avoid inaccurate specifications of preferences and give the possibility to select the solution that better suits the problem.

Using *a posteriori* methods, the solution of the MOP is the set of the non-dominated solutions, also called the Pareto optimal set. Depending on the problem, obtaining all the Pareto optimal solutions is not guaranteed or can take high computational times [94, 100]. Due to an heuristic approach is used to solve the problem in our case, approximations to the Pareto optimal set will be obtained. A definition of an approximated Pareto optimal solution is presented in [117]: a solution y obtained by an algorithm A is Pareto optimal relative to A, if A does not find another solution z, such that z dominates y. In general, the heuristic methods have to be developed with two important principles: (i) find non-dominated points as close as possible to the optimal set and (ii) find solutions diverse enough to provide a good coverage of this set [117].

Many methods such as the *weighted* method, the ϵ-*constraint*, and *goal programming* among others (see Section 3.3.1) solve the MOP by *scalarization* [106, 94, 100], i. e. transforming the problem into a single objective or a set of single objective problems. This strategy employs efficient and already tested single-objective algorithms existing in the literature and applies them to solve the MOP. Following this *scalarization* schema, we developed *SIM* to find the potential non-dominated solutions for our problem. The problem is solved with respect to the first objective and the value of the second objective is calculated from the obtained solution. At each iteration, the sequence for solving the sub-problems is modified in order to find a solution in the Pareto set to cover it in the best possible way.

The end of the turnaround process is determined by the Off-Block Time (OBT), when all the doors are closed, bridge removed, pushback vehicle present and the aircraft is ready for start up and push back [6]. Although this operation might not be necessary for aircraft parked at a remote position, pushing

away the aircraft (*pushback*) is the most typical way used for leaving the parking position. For that reason, we have defined *pushback* as the last task of the ground handling service in our problem. We used this information to create an initial sequence to obtain a lower bound of $F2$. That is, the turnaround is planned to be performed in the minimum time possible at the expense of reducing the operation time windows on the same aircraft. Moreover, when time windows are too tight, the number of vehicles needed to serve the tasks increases significantly. At each iteracion of the algorithm, we selected operations to be scheduled before *pushback* in order to improve the robustness but affecting the completion time as less as possible. Thus, the sub-problems are ordered and solved in such a way that a promising search space will be explored to improve $F1$ while a new value for the second objective is obtained. This method is described in Algorithm 7.

Let S be the set of sub-problems, where each sub-problem corresponds to each vehicle type, $|S| = |VT|$. The order in S describes the sequence the sub-problems are solved, s_{lo} is the sub-problem corresponding to the last operation, B the set of sub-problems to solve before s_{lo} such that $B \subseteq S \setminus s_{lo}$ and R the rest of sub-problems, such that $R = S - s_{lo} - B$.

In the first step of the algorithm, an initial sequence in S is created such that the s_{lo} is the first sub-problem to solve. When a sub-problem is solved first, the operations are scheduled within their original time windows. If this sub-problem is the *pushback*, a lower bound of $F2$ is obtained. On the other hand, this reduces the original time windows of the other tasks on the same aircraft, i.e. the time windows of the elements in R. So, a worse value of $F1$ is obtained.

At first, the elements in R are ordered by the values that were assigned to identify the sub-problems. In principle, when solving the last operation first, the best value of $F2$ is obtained regardless the order of the elements in R. However, solutions found should be as close as possible to the Pareto optimal set, i.e. a solution with a lower bound of $F2$ with the minimum value of $F1$. Therefore, after obtaining a solution with the initial sequence, the sub-problems in R are ordered by $f1$, that is, the total operation waiting time of the associated routing problem. Then, we repeat the process in order to obtain a better sequence of R.

In the second step, we aim to improve the value of $F1$ planning the rest of sub-problems before the last operation. At each iteration, the sub-problem with the highest value of $f1$ in R is chosen to be included in B and solved first. Adding sub-problems to B, that is, prioritizing the other operations with

Algorithm 7 Sequence Iterative Method (SIM)

1. Sequence to obtain a lower bound of $F2$

   (a) Create an initial sequence in S
   - $B \leftarrow \phi$
   - $R \leftarrow$ sort R by the values of the operation associated to each sub-problem
   - $S \leftarrow \{s_{lo}\} \cup R$
   - $F1 \leftarrow SolvingProcess(S)$

   (b) Repeat until no improvement is found
   - $R' \leftarrow$ sort the elements in R by their $f1$ in a decreasing order
   - $S' \leftarrow \{s_{lo}\} \cup R'$
   - $F1' \leftarrow SolvingProcess(S')$
   - if $(F1' < F1)$ then
     i. $S \leftarrow S'$
     ii. $F1 \leftarrow F1'$

2. Sequence to improve $F1$

   (a) Repeat until $|B| = |S \setminus \{s_{lo}\}|$
   - $b \leftarrow b \in R | b = \max_R \{f1\}$
   - $B \leftarrow \{b\} \cup B$
   - $R \leftarrow R \setminus b$
   - $S \leftarrow B \cup \{s_{lo}\} \cup R$
   - $F1 \leftarrow SolvingProcess(S)$

   (b) Repeat until no improvement is found
   - $R' \leftarrow$ sort the elements in R by their $f1$ in a decreasing order
   - $B' \leftarrow$ sort the elements in B by their $f1$ in a decreasing order
   - $S' \leftarrow B' \cup \{s_{lo}\} \cup R'$
   - $F1' \leftarrow SolvingProcess(S')$
   - if $(F1' < F1)$ then
     i. $S \leftarrow S'$
     ii. $F1 \leftarrow F1'$

respect to s_{lo}, usually leads to a $F1$ decreasing. Similar to the above step, the sub-problem selected is scheduled within its original time windows which lead to a lower bound of its $f1$. In order to find a range of solutions that represent the Pareto set, we have included one sub-problem at each iteration. Thus, an improvement of $F1$ is reached while a new value of $F2$ is found.

Finally, the process is repeated every time a new sub-problem is chosen. The goal of this part is to explore the search space modifying the sequence of the sub-problems in each subset keeping the position of s_{lo}. Up to now, we have focused on the relation between the last operation and the rest of the activities to improve the $F1$ value. At this last level, we address the relationship between the operations itself, particularly, activities having precedence constraints between them. The order in which these sub-problems are solved can also influence the value of the objectives. So, we perform a similar procedure to the step 1 in Algorithm 7: sub-problems in R and B are ordered by $f1$ and the procedure is launched following this new sequence.

4.3.5 Solution method discussion

In this section we have described the solution method developed for solving the proposed multi-objective approach. Algorithms implemented permit scheduling ground handling vehicles with a global perspective and to obtain better global solutions according to the optimization objectives defined: F_1 and F_2. Furthermore, the *SIM* procedure provides an easy and fast way to find a set of Pareto solutions that allows the decision maker to select the best solution according to its preferences.

The *SIM* method consists in modifying iteratively the sequence in which the operations are scheduled according to the solution found at each iteration. To obtain a solution for the global problem, each involved resource is scheduled following this sequence, that is, one VRPTW for each resource is solved. Solving this set of VRPTWs is an expensive procedure in computational time. Moreover, the process should be repeated several times in order to improve the global solution and obtain a range of Pareto solutions. For that reason, the *SIM* method was defined with the aim to generate the minimum number of solutions required to provide an adequate representation of the Pareto set. With this goal, the sequence for solving the sub-problems is modified at each iteration such that an improvement of $F1$ is reached, but affecting as little as possible the value of F_2. This way, each new solution could represent the best trade-off between the objectives and a good coverage of the Pareto set could

be obtained.

Nevertheless, the number of sequences explored with the *SIM* may be not sufficient depending on the decision maker requirements. Applying a more exhaustive method might be necessary to guarantee solutions as close as possible to the optimal Pareto points or to provide a wider Pareto set. In addition, the computational time required to produce this minimum set of solutions can be a disadvantage if the planning is not performed with enough time from the moment operations will take place. Although the goal of this approach is to tackle the problem at a tactical level, carrying out the scheduling process as close as possible to the time the operations occur ensures the schedule is generated on the basis of more accurate information.

4.4 Solution method variation

In this section we propose a variation of the solution method presented in Section 4.3 with the goal to increase the number of explored sequences. These sequences have a strong influence on the quality of the solution. However, computing all the possible combinations, or even a high number, can be too expensive in computational effort. Solving one VRPTW is already a complex problem and one VRPTW for each type of vehicle has to be solved for each sequence.

As explained in Section 4.3 we have developed the *RLP* procedure in two steps. A quick, reasonable good initial solution is found by a constructive heuristic and a local search process is applied in order to improve the solution. The local search process is clearly the most time-consuming part of the algorithm. Thus, we direct our efforts to reduce the number of times the routing problem is solved using both steps. The solution method proposed (see Figure 4.3) is used in a similar way except that local search process is not executed every time the VRPTW for each sub-problem is solved during *SolvingProcess*. Only the most promising solutions obtained will be improved by the hybrid methodology. A schema of this variation is outlined in Figure 4.6.

With this purpose we aim to determine different criteria to select a subset of promising sequences of tasks. A set of sequences is explored using only the I3 heuristic and just the most convenient found solutions according to the defined criteria will be improved.

Depending on the degree of exhaustiveness desired, different procedures could be used to explore sequences using only I3, including *SIM*. In the first

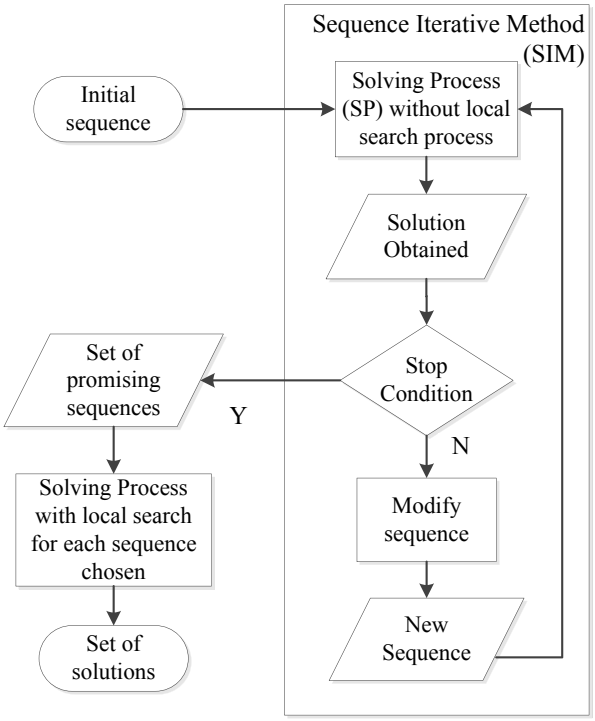

Figure 4.6: Schema of the solution method variation

approach, the *SIM* modified the sequence at each iteration in order to generate the potential Pareto solutions of the problem. With the new version, we aim at employing the *SIM* to find the sequences which could generate the best solutions. In this sense, we run the *SIM* without applying the local search process in any case. Next, we select the best sequences from the solutions found. Finally the *SolvingProcess* is executed with each promising sequence chosen, including the local search process in this occasion.

Selecting promising sequences

A range of solutions is obtained as a result of exploring a set of sequences with I3. The next step is identifying the best solutions and select corresponding sequences as promising.

In general, identifying the best solution(s) in a multi-objective context is not a trivial task. Many variables can influence the decision and even lead to contradictory results. Hence, the aim of this section is to provide simple rules

to guide the decision-making process in order to select the most representative solutions. According to some specific pattern or based on the experience, we can select solutions that better suit the problem in practice.

First, we select the non-dominated solutions in the same way as in the original schema. Then, two criteria are suggested to select a subset of the Pareto solutions. The goal of the first strategy is to find the best relation between the objectives which can produce the best global solutions, i.e. the points giving the best results for all the vehicles. In the second rule we focus on prioritizing certain operations that in practice could become part of the critical path due to different reasons.

In the first rule, an intuitive idea is to identify points that represent different ranges from the complete set of Pareto solutions. In this sense, we split the Pareto frontier into three areas according to the objective values. In the first area we have solutions where the completion time of the turnaround is prioritized. Both objectives are balanced in the second area and minimizing the robustness of the solution is favored in the third range. An example is presented in Figure 4.7. Notice that the number of areas the Pareto frontier is divided can be different and can be specified according to different needs.

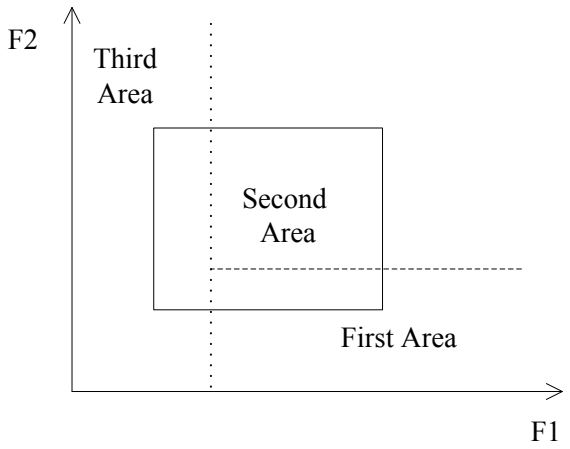

Figure 4.7: Pareto frontier division into three areas

Then, we select a specific number of points from each area such that the relation between the objectives is the best. Let L be the set of solutions in the Pareto frontier and $K = \{1, 2, 3\}$ the areas established. The best relation between two points i, j in the area l_k is defined as:

$$r(i,j) = (|F1_i - F1_j|)/(|F2_i - F2_j|)|F2_i < F2_j \ \forall i,j \in l_k \ \forall k \in K \quad (4.22)$$

$$r(i^*,j^*) = \max r(i,j) \quad (4.23)$$

Relation (4.22) aims to measure how the improvement of $F1$ affects the $F2$ function. If the value of $r(i,j)$ is greater than 1, the first objective is improved to a greater extent than the second one is worsened. In this case, the solution j is considered better than the solution i. In contrast, if $F1$ is less improved at expenses of a higher increase of $F2$, solution i is the preferred. Two solutions are equivalent with respect to the relation between objectives when r is close to 1.

The second rule proposed is applied sequentially after this criterion. That is, it is used to decide which solutions are the best from the set of best points obtaining.

For the second criterion, we consider the operations individually and how they are scheduled in the different solutions. For instance, the number of vehicles required to perform certain operation might be a measure to determine the preferred solution. Most efficient resource planning of activities prone to be involved in delays may contribute to improve flight punctuality. With this idea in mind, we have defined two patterns to guide the selection process and choose schedules where the minimum number of vehicles is required in two kinds of operations:

- Complex activities with expensive vehicles

  The availability of resources for performing the operations is a key aspect to guarantee flight on-time, particularly when these resources are expensive or complex to use. Fueling is one of the most complex tasks during the turnaround and strict precautionary measures have to be fulfilled in order to guarantee safety. A right configuration and a proper maintenance of all the necessary equipment, including fueling vehicles are crucial in this operation. So, saving in vehicles utilization particularly in this task permit reducing cost and favors the efficiency of the process.

- Operations with longer duration

These operations have little margin to deal with unexpected events or incidents with vehicles. Employing fewer resources in longer activities contributes to the robustness of the solution since they leave spare vehicles that could be used in case of complications.

These patterns are also applied sequentially, i.e. if two solutions require the same number of vehicles for performing fueling, the second rule is used to decide.

With the criteria proposed we can select a small subset of promising solutions and reduce the computational effort of the approach. It should be stated that these rules can be employed to select the points to be explored with the local search process, but also to determine which solution is the best to be implemented in a particular case.

4.4.1 Discussion

This version of the approach is an interesting alternative to overcome time limitations without compromising quality as results show in Section 5.4 and permits to increase the number of sequences that can be explored. This schema can be implemented since the heuristic and the local search process have similar minimization criteria. That is, although the *I3 Insertion Heuristic* method adopted to find the initial solution does not minimize explicitly the operation waiting time, this criterion is considered when clients are selected to be inserted in the routes. Moreover, earliest start time criterion has been used to initialize any route, contributing to reduce the waiting time.

The main objection to the schema is that sequences obtained only with I3 method are not necessarily the same as the original approach. This means that some sequences which can generate Pareto solutions may not be found using just the I3 method. Furthermore, good sequences may be rejected because they appeared as dominated in the first step, and therefore they are not explored. However, experiments presented in Chapter 5 show that the Pareto fronts obtained with both versions are very similar.

Chapter 5

Results

Experiments performed to assess the proposed multi-objective approach are presented in this chapter. We have used real data from two important airports in Spain and developed instances are described in Section 5.1. Results obtained with the original solution method, which we called first approach, are analyzed in Section 5.3. Solutions found with the solving method variation, which we called second approach, are outlined and compared with the first approach in Section 5.4. Finally, simple rules are suggested to guide the most promising solution selection process and enhance the second approach performance.

5.1 Instances generation

To the best of our knowledge, no benchmark instances exist for the ground handling problem. Therefore, a set of scenarios was developed to validate the proposed approach. In order to test the algorithm we need information about three crucial aspects: flight schedule, aircraft parking distances and tasks to be performed. For specifying the flight schedule and parking distances we have used real data from two important airports in Spain: *Palma de Mallorca* (PMI) and *Barcelona* (BCN).

The main difference between the two flight schedules is the number of aircraft planned to serve. In the case of PMI, we considered all aircraft performing a turnaround during a working day. On the other hand, we focused on a handling company that provide services at BCN airport. That is, we have used the flight information used by the company to plan operations. It should be stated that using data with very different characteristics is quite useful to test

the efficiency of the approach. Besides parking distances, the main difference between the two used datasets is the flight arrival frequency and the types of aircraft planned to be serviced.

Regarding activities information, we have employed the standards provided by aircraft manufactures [118, 119, 120, 121, 122, 123, 124]. It is a fact that most airlines adapt specifications to their own business model. However, these modifications usually remain confidential as discussed in Section 2.2.1. So, applying aircraft standards is a common method to study the turnaround [11].

As described in the problem formulation in Section 4.2, operation properties such as duration and precedence restrictions depend on the aircraft type. Three types of aircraft with different sizes have been modeled for PMI instances. Scheduled aircraft have been associated to the most appropriate type according to the turnaround time. Regarding BCN dataset, there is more variety of aircraft sizes covering scheduled flights. Thus, we have modeled all aircraft types presented in the flight schedule (9).

Palma de Mallorca instances

PMI is one of the busiest airports in Europe during high season, with around 22 million passengers in 2012 [125]. Due to we have considered all the aircraft, the instances generated using these data are larger than the instances from the BCN airport.

In summary, the following datasets were used in order to create the instances:

1. One day flight schedule: a real flight schedule corresponding to a summer business day was selected for evaluating the approach. This includes the list of aircraft performing a turnaround during the day, the scheduled arrival and departure times, the type of aircraft, and the parking position.

2. Distances between the parking positions and between them and the depot. PMI airport has 180 parking stands, 27 of them remote stands. A constant speed was used to calculate vehicle s traveling time.

3. Task information: Using specifications [119, 122], aircraft types I,II, and III were modeled. According to precedence relations and operations duration, type I corresponds to aircraft with a turnaround time between 30 and 40 minutes. Type II and type III are the aircraft with a 40-50

minutes and 50-60 minutes turnaround, respectively. For each operation included in the problem and according to each type of aircraft, we have defined the duration, the precedence restrictions regarding other tasks, and the type of vehicle used.

The precedence restrictions between tasks for each defined aircraft type are outlined in Figure 5.1. We have represented the main operations during a typical turnaround, when the aircraft is parked at a contact point (e.j. Figure 2.1). The vehicle that unloads and loads baggage for any given aircraft is usually the same. Therefore, we represented both operations as one task, the UL/L. The other operations are: deboarding (Db), boarding (B), catering (Ca), cleaning (Cl), fueling (F), potable water (PW), and toilet services (TS). In addition, the number in parentheses denote the type of vehicle servicing each operation.

We do not consider aircraft parked at remote stands. When aircraft are parked at a contact point, the Db and B operations do not have vehicles associated because they are performed by means of bridges connected to the gate. Nevertheless, these activities are very important during the turnaround service. They appear in precedence relations and their calculation affect the time windows of other operations.

The different characteristics of a turnaround, particularly precedence restrictions between operations at each aircraft, influence their time windows and, consequently, the obtained solutions. Due to airlines can modify the standard turnaround services, we have tested the algorithm using different precedence relations. Three sets of instances, C1, C2 and C3, were generated changing the precedence constraints presented in Figure 5.1. The first set was associated to the original precedence rules. In the second set, relations between PW and TS were changed in aircraft types I and II, so TS is always performed before PW. In the set C3, Ca is performed independently of the Db/B operations in all aircraft types.

We have divided the daily flight schedule in order to plan each period separately. This division was done according to the airport workload. At the given data set, the flight arrival frequency is relatively uniform during the day and the expected workload is also uniform. We have selected eight-hour periods for dividing the schedule because this is the maximum duration of shifts. Shift duration can vary between 2 and 8 hours depending on several aspects, such as staff policies, workload etc. In general, the employees who drive the vehicles should come back to the depot when they finish their shift.

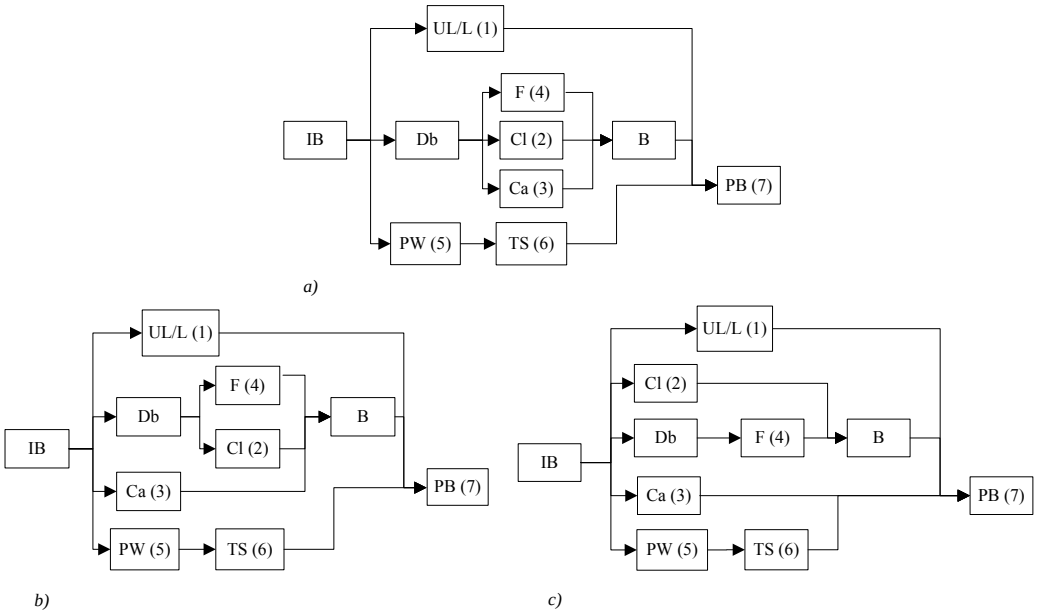

Figure 5.1: Precedence relations between tasks according to the modeled aircraft type: a) Type I, b) Type II, and c) Type III. The number of the associated vehicle type is indicated in parentheses

Nevertheless, during these eight hours, a shift change can be performed or vehicles can be planned to come back to the depot for other reasons. A first group J1 was created between 23:00 and 7:00 hours with 42 aircraft. The time interval between 7:00 and 15:00 hours with 64 aircraft corresponds to the J2 group. The last shift, J3, has 83 aircraft between 15:00 and 23:00. J1 has fewer aircraft scheduled than J2 and J3. However, the workload is similar because most of aircraft are planned to arrive between 4 and 7 in the morning. Each group was scheduled with the precedence rules defined at each set. The algorithm was tested over a total of 9 instances from PMI, where each instance is enumerated with the number of the set and the shift used: C1J1, C1J2, C1J3, C2J1, C2J2, C2J3, C3J1, C3J2, C3J3.

Barcelona instances

Barcelona-El Prat (BCN) is the second largest airport in Spain by passenger traffic and the 34th in the world according to the Airports Council International [125]. With more than 35 million passengers in 2012 and a growing

tendency, Barcelona airport is also the most important of Catalonia and the Mediterranean Sea. The handling company that provided the data, which for confidential reasons we cannot disclose its name, gives service to different airlines. Similar datasets to the PMI case were employed to generate the instances.

1. One day flight schedule: The flight schedule includes aircraft planned to be serviced during a typical working day in June, the scheduled arrival and departure times, the type of aircraft, and the parking position.

2. Distances between parking positions and between them and the depot. Barcelona airport has 263 parking stands. Using the geographical coordinates of each stand we have calculated the distance while a constant speed was used to determine vehicles traveling time.

3. Tasks information: Duration, precedence restrictions regarding other tasks, and the type of vehicle used have been specified for each operation according to the type of aircraft they belong. We have used the information provided by the aircraft manufacturers [118, 119, 120, 121, 122, 123, 124] to model each type of aircraft planned to be serviced, 9 in total.

We have identified six different precedence relations in the represented aircraft and they are outlined in Figure 5.2. Most common restrictions are presented in a), where three independent groups of activities can be distinguished. First, the deboarding and boarding group, where catering, fueling and cleaning are performed between them; a second branch for water and toilet services, and finally, baggage handling. We have discussed in Chapter 2 that precedence constraints between tasks rely on several issues, such as safety or space requirements. These restrictions imply longer turnaround times and better coordination between handle operators. So, they can be adjusted within certain limits according to the airline policy. Aircraft specifications reflect possible variations, particularly regarding first group also known as the critical path.

According to these precedence relations and operations duration, we can obtain the minimum turnaround time associated to each aircraft type. It is shown in Table 5.1. Notice this is the minimum time required. In general, the time scheduled to perform the turnaround is higher because airlines add a buffer to deal with uncertainty, as we described in Section 2.1. As part of the TITAN project [25], an study has been done to estimate the average of a buffer

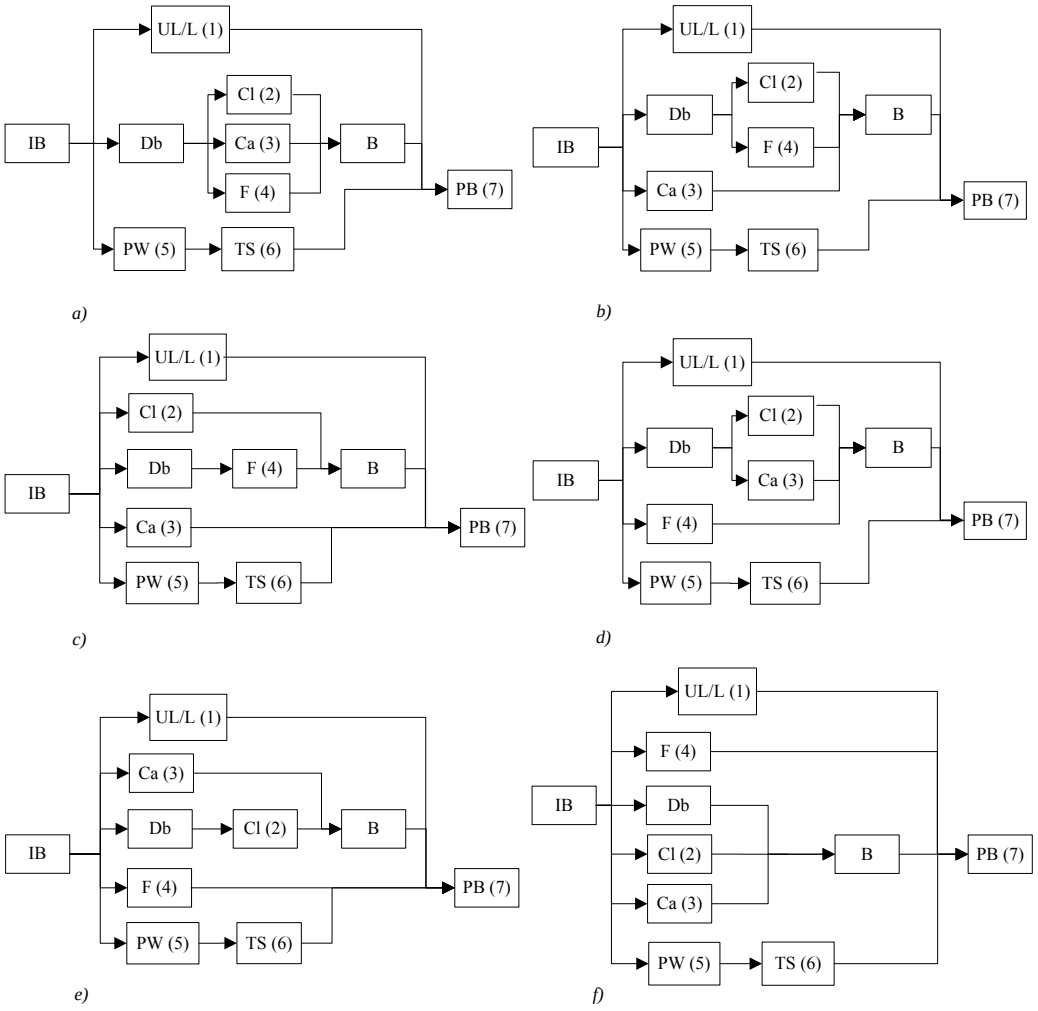

Figure 5.2: Precedence relations between tasks according to the modeled aircraft type: a) Type I; b) Type II; c) Type III;IV;VIII;IX; d) Type V; e) Type VI; and f) Type VII. The number of the associated vehicle type is indicated in parentheses

time. The analysis shows that most aircraft have between 5 and 22 minutes of buffer, being 12 minutes the mean size. With the goal to make instances more realistic we have verified that the difference between the minimum turnaround time and the scheduled time are in this interval for most aircraft. This way, we ensured that aircraft standards we have used are similar to those employed by airlines. It should be remarked that aircraft types I,II and III modeled for

Table 5.1: Minimum turnaround time, in minutes, for each modeled aircraft type

AT	T.Time	Prec. Relat.
I	27	a
II	35	b
III	43	c
IV	51	c
V	56	d
VI	40	e
VII	53	f
VIII	29	c
IX	36	c

PMI instances are the same as aircraft I, II and III in the BCN case.

In the same way as the PMI instances, three sets, C4, C5 and C6, were generated modifying precedence constraints. The first set was associated to the precedence rules described in Figure 5.2. In the second set, relations between PW and TS were changed in a) and b), so TS is always performed before PW. In the set C6, the Ca is performed independently of the Db/B operations in all the precedence relations.

In the BCN case, flight arrival frequency is lower due to only a subset of the arriving aircraft during a day is handled by the company. In addition, aircraft are more uniformly distributed in the timetable with respect to PMI. Due to the company only has aircraft planned between 6:00 and 22:00 hours, the flight schedule was divided in two eight-hour shifts. A first group J4 was created between 6:00 and 14:00 hours with 56 aircraft. The time interval between 14:00 and 22:00 hours with 37 aircraft corresponds to the J5 group. Also here, each group was scheduled with all different precedence sets. Hence, 6 instances were generated from the BCN set. Instance is enumerated with the number of the set and the shift used: C4J4, C4J5, C5J4, C5J5, C6J4, C6J5.

5.2 Parameters setting

In this section we describe the set of parameters used in the different proposed algorithms. First, we present the results of the process used to tune the Insertion heuristic method. Then, the values used in the CP-based VND-LNS methodology are introduced.

Main parameters in the I3 heuristic are α_1, α_2, and α_3. As discussed in Section 4.3.2, an interval was defined for each parameter, such that: $\alpha_1 =$

$[0.4, 0.5]$, $\alpha_2 = [0.4, 0.5]$, and $\alpha_3 = [0.01, 0.1]$. We have tested the algorithm over different combinations and we have selected the weights that give the best results for most of the instances. Instances C1J1, C2J2, C3J3, C4J4 and C4J5 have been employed to carry out the experiments.

Initially, we have run the first approach with the I3 heuristic only over all instances using the weight combinations defined by the intervals. For each instance we have selected the two parameter settings giving the best results. Next, we have analyzed the obtained solutions using this best combination set over all the instances. The five weight combinations from the best set giving the best solutions for all instances were finally chosen. We have calculated the average of the solutions found for each instance and they are outlined in Figure 5.3.

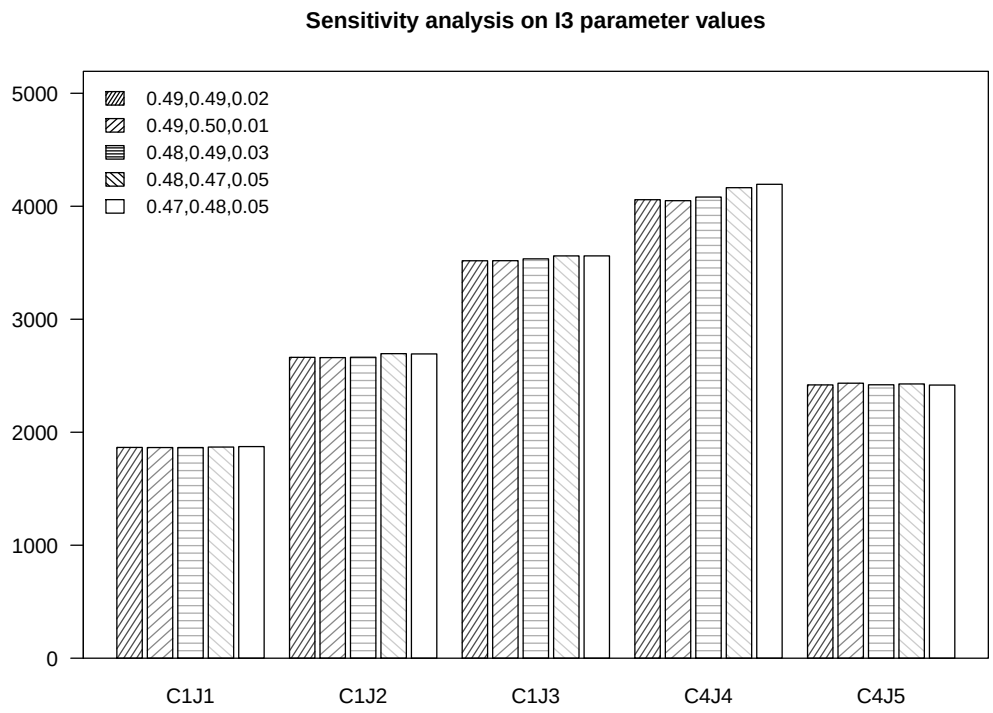

Figure 5.3: Results of sensitivity analysis performed for tuning the I3 heuristic

In general, the best results are obtained with lower values of α_3. In all the instances, but C4J5, better solutions are found using α_3 between 0.01

and 0.03 although the difference is less significant in C1J1. Combinations $(0.49, 0.50, 0.01)$ and $(0.49, 0.49, 0.02)$ appear as the best ones for most cases. However, in the case of C4J5, the difference with the best solution is more notable in $(0.49, 0.50, 0.01)$, so we have decided to select $(0.49, 0.49, 0.02)$ as the weights in our problem.

With respect to the parameters of the CP-based VND-LNS methodology they have been specified in the following way. Concerning *SMART* operator, we assigned 2 and 3 to the number of customers to be disconnected before and after the pivots. In addition, the number of discrepancies has been limited to 2. As for the *RPOP*, the maximum number of pivots to be chosen is set to 5 and the number of close customers to be removed around the pivot is set to 7. Due to operations generally have tight time windows in our problem, removing customers closer to the pivot is more likely to produce feasible movements than random selection. Thus, we give a higher value to the second parameter. Furthermore, the branch-and-bound method used to repair the solution is limited to a maximum execution time of 30 seconds in both operators. The whole local search process is applied during a maximum of 250 seconds in order to ensure the improvement of the initial solution found by the I3 heuristic.

5.3 First approach

In this section, we present the results obtained with the original solution method over the generated instances. Both the first and the second approach described in this thesis have been implemented in Java and linked to the open-source CP software system ECLiPSe 6.0 [50].

Palma de Mallorca instances

The detailed results of the instance C1J1 (precedence constraints rules C1, shift J1) for each iteration of the *SIM* are presented in Table 5.2. Results are obtained running the algorithm only one time. Besides the value of objectives $F1$ and $F2$, the obtained sequences and the number of vehicles used for each operation are shown. It should be stated that if a solution obtained after an iteration has not improved any of the two objectives regarding the previous iteration, it is rejected. In this instance the sequence was modified 13 times but one solution was discarded for that reason. So, this problem has 12 solutions. As mentioned, the value of $F1$ is usually improved when the sub-problem

Table 5.2: Solutions obtained for the instance C1J1 at each iteration of the *SIM*

N. S.	F1	F2	Sequence	#Vehicles						
				UL/L(1)	Cl(2)	Ca(3)	F(4)	PW(5)	TS(6)	PB(7)
1	2383	1594	(7,1,2,3,4,5,6)	19	12	12	10	4	8	4
2	2165	1613	(6,7,1,2,3,4,5)	18	12	12	9	6	7	4
3	1980	1621	(5,6,7,1,2,3,4)	18	11	11	9	4	7	4
4	2425	1619	(6,5,7,4,3,2,1)	18	11	12	9	6	7	4
5	1850	1655	(4,5,6,7,1,2,3)	18	11	11	9	4	7	4
6	2154	1646	(6,5,4,7,3,2,1)	18	11	12	9	6	7	4
7	1709	1695	(3,4,5,6,7,1,2)	18	11	11	9	4	7	4
8	1998	1681	(6,5,3,4,7,2,1)	18	11	11	9	6	7	4
9	1565	1715	(2,3,4,5,6,7,1)	18	10	11	9	4	7	4
10	1816	1687	(6,5,3,4,2,7,1)	17	10	11	9	6	7	4
11	1510	1736	(1,2,3,4,5,6,7)	16	10	11	9	4	7	5
12	1792	1714	(6,5,3,4,2,1,7)	16	10	11	9	6	7	5

with the highest *f1* is included in B to be solved first (see Section 4.3.4). In contrast, modifying the sequence of sub-problems keeping the position of PB is less likely to produce an $F1$ improvement. The set of solutions is presented in Figure 5.4.

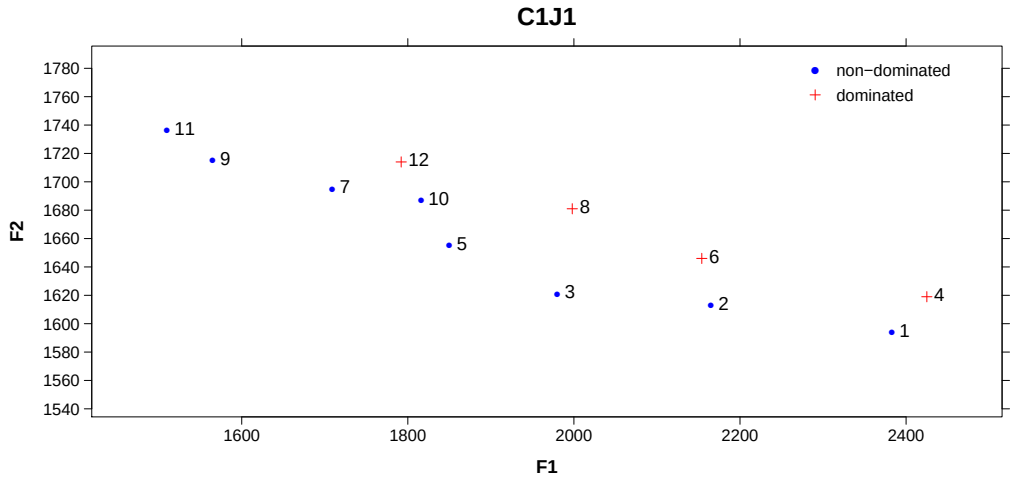

Figure 5.4: Found solutions for the instance C1J1. Each solution is labeled according to the iteration in which it is obtained. Non-dominated solutions, represented by bullets, define the trade-off curve between the two objectives. The crosses indicate dominated solutions

The first solution in Table 5.2 corresponds to the first step of the algorithm

and shows the sequence to obtain the lower bound of $F2$. Regardless of the problem, the process is always started with the same initial sequence of vehicles $(7, 1, 2, 3, 4, 5, 6)$ (see Algorithm 7, step 1). As can be seen, vehicles numbers have been assigned following the order of the operations presented in Figure 5.1. Each time a new sequence is obtained, the *Solving Process* is invoked and the routing problem associated to each type of vehicle is solved.

In the next iteration, the operation with the highest value of $f1$ is scheduled before PB, in this particular case TS. The value of $f1$ depends on the result of the routing problem and the reduction of the time windows. In general, it is possible to identify activities or groups of activities without precedence relations in the turnaround. Regarding precedence restrictions and tasks duration, an activity or group of activities could be more restrictive than the others. The less constrained operations have larger time windows, i.e. the service may be scheduled to begin with higher tolerance. This situation is similar to the case in which two operations with different duration have to be finished before a third one. The operation with the shortest duration or the largest time window may have some waiting time without affecting the third task. When the less constrained operations are included in B before the other ones, the value of $F2$ is less affected. Due to we use an upper bound for the number of vehicles, scheduling operations with larger time windows usually results in high waiting times. So, in general, the less constrained operations has high $f1$. Usually, TS and PW are operations with short durations and we could say they belong to the less restrictive group in C1. The UL/L does not have precedence relations with other operations but it is the longest one. Operations F, Cl and Ca are constrained according to the precedence rules defined at each aircraft type, and are usually longer than TS and PW. Each time a new sub-problem is included in B, vehicles are ordered again by their $f1$ and the process is repeated. Activities PW and TS are interdependent and so, the one which is solved later will have a higher value of $f1$. In C1, PW is always performed before TS. When TS is scheduled before PW, a better value of $F2$ is reached at expense of reducing the time window of PW. Notice in Table 5.2 the increase in the number of vehicles whenever TS is scheduled first.

An important aspect of how scheduling decisions of a resource affect the other ones is the vehicle utilization. Notice, for instance, the increase in the vehicles needed to serve the operations whenever PB is solved first, particularly in the most constrained operations. Obtaining lower values of $F2$ implies a time window reduction on the operations at the same aircraft, and conse-

quently an increment of used vehicles. At the first iteration, the UL/L needs 19 vehicles while it uses 16 when PB is scheduled last (solution 11 and 16). As discussed in Section 4.4, this might be an interesting criterion to select a solution or prioritize an operation according to the particular situation of a vehicle type. For instance, the schedule associated with solution 9 is very similar to that of solution 11 regarding F1 and F2. However, the UL/L needs 18 vehicles in the former and 16 in the latter. The UL/L is usually the longest operation. If a delay or an unexpected event occurs, it is more likely to need a spare vehicle, and therefore the solution 11 might be a good choice. On the other hand, the PB requires 4 and 5 vehicles, respectively. If the vehicles to perform PB are more limited in number, or they are more expensive to use, solution 9 might be better.

A summary of the results obtained for all instances is outlined in Table 5.3. The non-dominated solutions are marked with an asterisk and are presented in Figure 5.5. In addition, the total time spent by the algorithm to solve each instance is outlined. The obtained sequences for the three shifts are very similar for each set, because precedence relations between operations are the same. Nevertheless, the number of aircraft of each type is different at each instance. In the case of C2, the relation between operations PW and TS was modified in two aircraft types. The service time of PW is shorter than TS in all the aircraft, but in type I the difference is very small. The values of the objective functions are very similar in the shift J1, because most of the aircraft are type I. On the other hand, the variation is more important in J2 due to there are more aircraft of type II. Regarding the C3 set, Ca does not have precedence relations with the rest of tasks. This favors the value of $F2$ due to the group of Db and B is less restrictive. This also increases the time window of Ca, because the operation is less constrained and therefore less vehicles are needed to serve it.

Table 5.3: Summary of results for PMI instances. Obtained values of objectives $F1$ and $F2$ using precedence constraint rules in C with shifts J are given at each column. The non-dominated solutions are marked with an asterisk.

No.	**C1J1**		**C1J2**		**C1J3**		**C2J1**		**C2J2**		**C2J3**	
Sol.	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**
1	2383*	1594*	3389*	2362*	4717*	3009*	2596*	1589*	3770*	2331*	4867*	2994*
2	2165*	1613*	4064*	2346*	4281*	3057*	2369*	1607*	3309*	2357*	4524*	3053*
3	1980*	1621*	3509	2403	4608	3067	2142*	1629*	3002*	2381*	4096*	3083*
4	2425	1619	2619*	2393*	3806*	3112*	2364*	1610*	3438*	2346*	4394	3104
5	1850*	1655*	2464*	2414*	3282*	3130*	1864*	1670*	2623*	2422*	3590*	3109*
6	2154	1646	3162	2399	4121*	3100*	2096*	1654*	2869*	2385*	3940	3153
7	1709*	1695*	2101*	2466*	3169*	3185*	1762*	1694*	2349*	2464*	3360*	3177*
8	1998	1681	2904	2464	3896	3173	1784*	1680*	2756	2444	3046*	3203*
9	1565*	1715*	1833*	2490*	2818*	3207*	1671*	1707*	2157*	2488*	3428	3215
10	1816*	1687*	2754	2486	3547	3197	1709*	1699*	2586	2486	2790*	3262*
11	1510*	1736*	1756*	2508*	2622*	3264*	1513*	1712*	1924*	2513*	3279	3286
12	1792	1714	2499	2509	3301	3252	1676	1710	-	-	-	-
T.T.(s)	16829.53		18023.17		16375.34		18112.82		18792.33		17971.83	

No.	**C3J1**		**C3J2**		**C3J3**	
Sol.	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**
1	2798*	1584*	3688*	2360*	4848*	3029*
2	2748*	1600*	4271*	2345*	4715*	3068*
3	2360*	1635*	3504*	2368*	4098*	3110*
4	2640*	1613*	3036*	2407*	4640*	3087*
5	2086*	1659*	3832	2403	3741*	3161*
6	2476	1639	2809*	2455*	4521	3133
7	1983*	1670*	3548	2447	3521*	3181*
8	2275	1662	2543*	2478*	4271	3186
9	1764*	1714*	2360	2517	3264*	3218*
10	2170	1692	3067	2514	3964	3200
11	1729*	1734*	2206*	2539*	3108*	3280*
12	2054	1711	2863	2522	3690	3265
T.T.(s)	16254.98		18798.34		17.605	

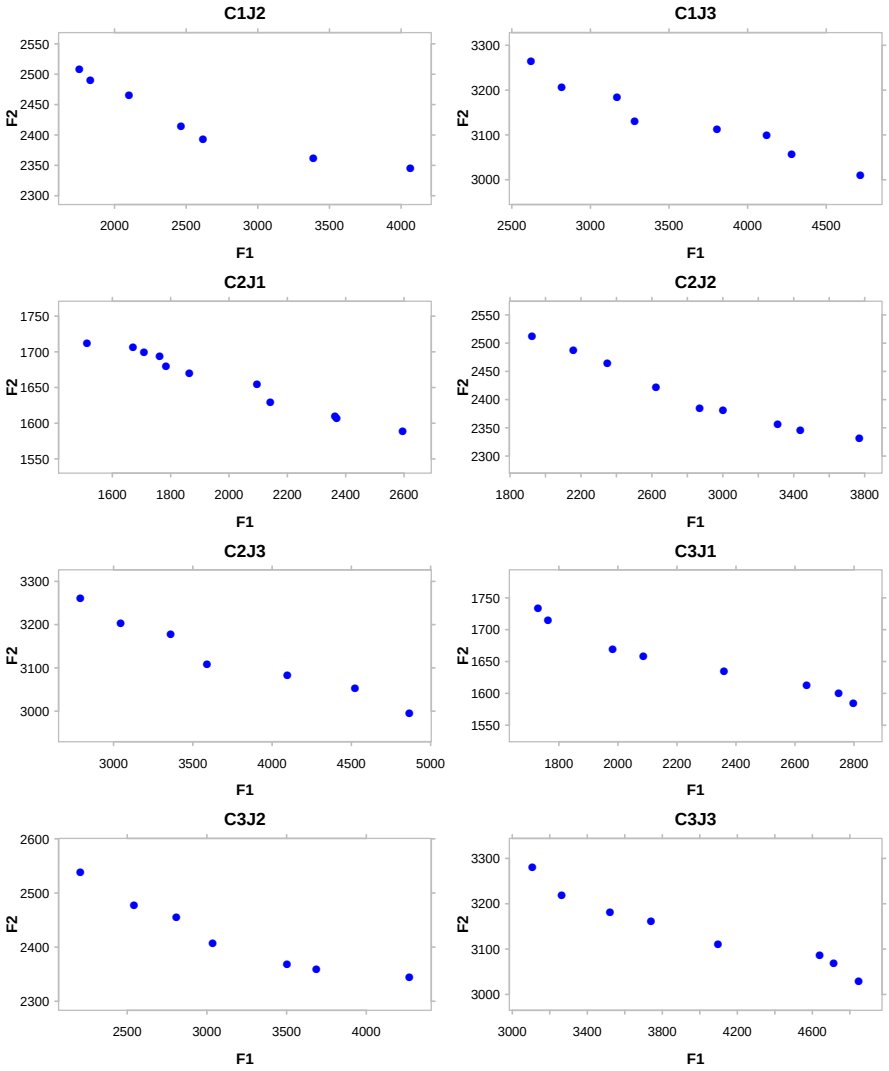

Figure 5.5: Pareto solutions for PMI instances C1J2, C1J3, C2J1, C2J2, C2J3, C3J1, C3J2, and C3J3

Barcelona instances

In the case of BCN instances, the number of resources required to perform operations is one of the main differences regarding PMI instances. Many factors can influence these results, such as task durations or turnaround times at each schedule. Moreover, the flight arrival frequency has a considerable impact in

Table 5.4: Solutions obtained for the instance C4J5 at each iteration of the *SIM*

No. Sol.	F1	F2	Sequence	#Vehicles						
				UL/L(1)	Cl(2)	Ca(3)	F(4)	PW(5)	TS(6)	PB(7)
1	4049	1359	(7,1,2,3,4,5,6)	11	7	7	6	2	4	2
2	3577	1468	(5,7,1,2,3,4,6)	10	6	6	5	2	3	2
3	3825	1449	(6,5,7,1,2,3,4)	10	6	6	5	4	3	2
4	2999	1494	(5,6,7,2,4,1,3)	10	6	6	6	2	3	2
5	2680	1595	(4,5,6,7,2,3,1)	9	6	6	5	2	3	2
6	3252	1558	(6,5,4,7,2,3,1)	9	6	6	5	4	3	2
7	2398	1681	(3,4,5,6,7,1,2)	8	6	6	5	2	3	2
8	2928	1649	(6,5,3,4,7,1,2)	9	6	6	5	4	3	2
9	2263	1715	(2,3,4,5,6,7,1)	8	6	6	5	2	3	2
10	2846	1698	(6,3,2,5,4,7,1)	8	6	6	5	4	3	2
11	2118	1754	(1,2,3,4,5,6,7)	8	6	6	5	2	3	2

the number of vehicles. For instance, C1J1 and C4J5 have a similar number of scheduled aircraft 42 and 37, respectively. However, C1J1 requires more resources as shown in Table 5.4. The set of solutions obtained is presented in Figure 5.6.

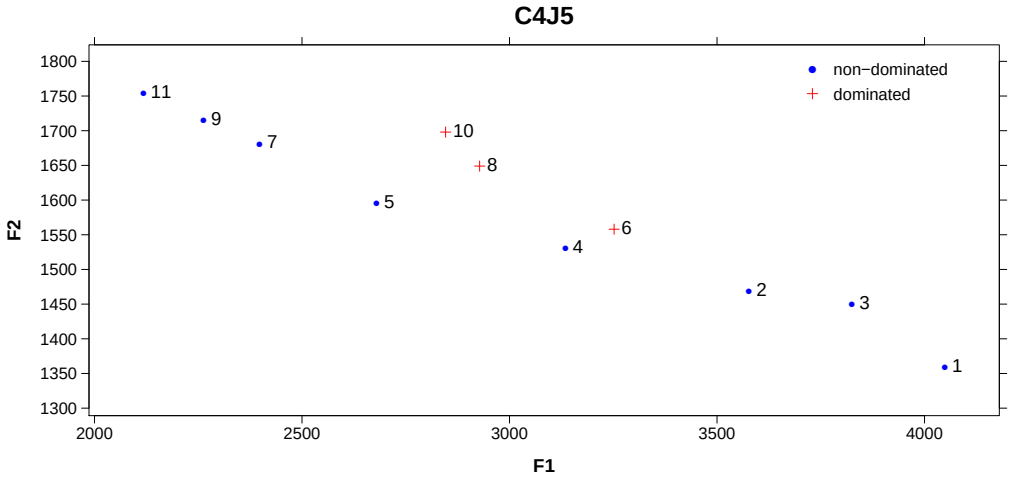

Figure 5.6: Found solutions for instance C4J5. Each solution is labeled according to the iteration in which it is obtained. Non-dominated solutions, represented by bullets, define the trade-off curve between the two objectives. The crosses indicate dominated solutions

A summary of solutions found for BCN instances is depicted in Table 5.5. The non-dominated solutions are marked with an asterisk and are presented

in Figure 5.7. Slightly higher values of *F1* are obtained when J5 is coupled with the preference set C5. Unlike J1, a higher number of aircraft with different durations of PW and TS are present in J5. When precedence relations between these operations are changed, obtained results can be affected. Solutions found for C6 showed similar behavior for both shifts J4 and J5. Being Ca less constrained than in the set C4, the available time windows to perform the operation are longer and a lower number of vehicles is required. In general, this conduces to obtain worse values of *F1* because waiting times are increased. Nevertheless, having waiting time in the less constrained operation permits vehicles to be used more efficiently without affecting the overall performance of the turnaround.

Table 5.5: Summary of results obtained for BCN instances. Values of objectives $F1$ and $F2$ found using precedence constraints rules in C with shifts J are given at each column. The non-dominated solutions are marked with an asterisk.

No.	**C4J4**		**C4J5**		**C5J4**		**C5J5**		**C6J4**		**C6J5**	
Sol.	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**	**F1**	**F2**
1	6300*	2276*	4049*	1359*	6658*	2276*	4271*	1369*	7239*	2248*	4269*	1363*
2	5975*	2319*	3577*	1468*	6587*	2274*	3767*	1450*	6574*	2322*	3849*	1446*
3	5840*	2424*	3825*	1449*	5772*	2429*	3226*	1486*	6356*	2416*	4078*	1440*
4	5307*	2484*	2999*	1494*	5648*	2448*	3464*	1478*	5618*	2457*	3266*	1459*
5	4942*	2595*	2680*	1595*	5319*	2458*	3204	1632	5009*	2572*	2882*	1524*
6	5684	2586	3252*	1558*	4826*	2632*	2988*	1552*	5863	2577	3556	1534
7	4608*	2736*	2398*	1681*	5306	2627	2730*	1666*	4784*	2682*	2696*	1630*
8	4200*	2833*	2928	1649	4524*	2693*	2611	1722	5639	2630	3196	1573
9	5012	2753	2263*	1715*	4469*	2801*	2553*	1709*	4739*	2776*	2363*	1683*
10	4100*	2838*	2846	1698	4249*	2857*	2560	1731	5261	2717	2998	1632
11	4875	2833	2118*	1754*	4768	2855	2418*	1745*	4487*	2843*	2133*	1753*
12	-	-	-	-	-	-	-	-	5322	2830	2949	1736
T.T. (s)	18438.25		17201.87		18982.47		18564.34		173453.94		16902.98	

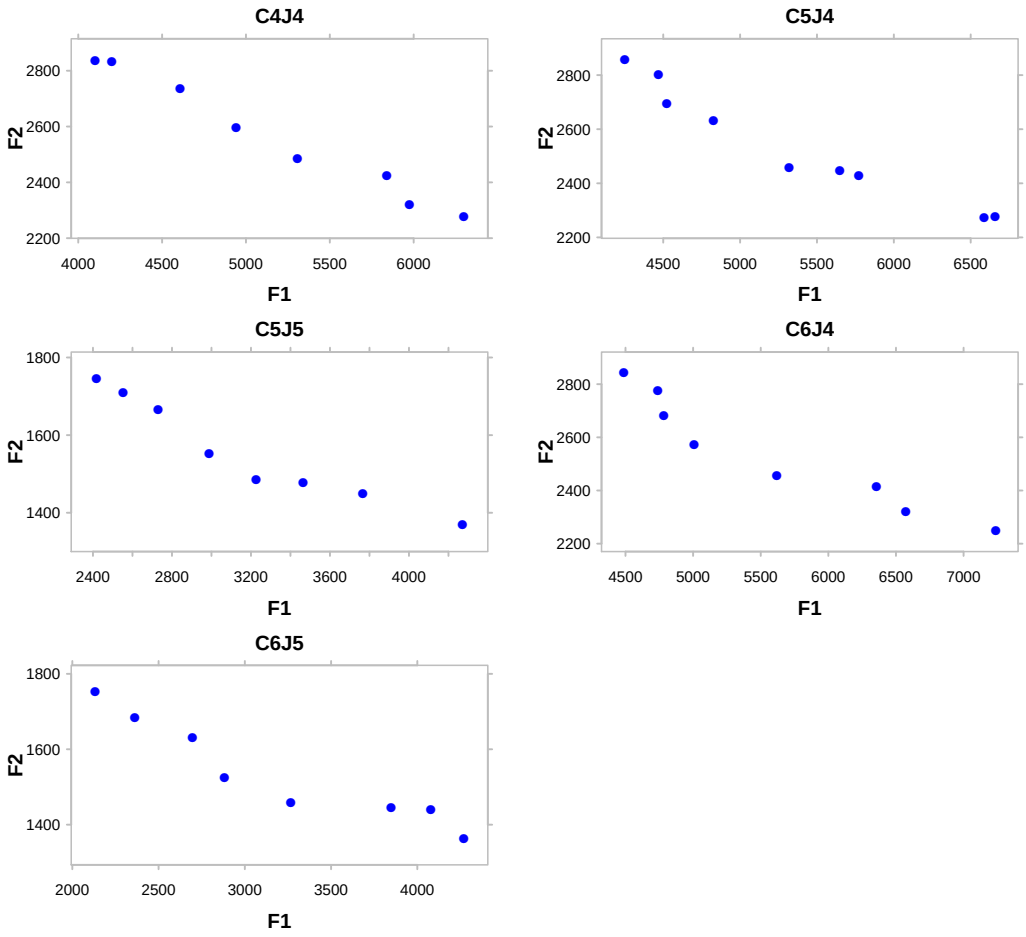

Figure 5.7: Pareto solutions for instances C4J4, C5J4, C5J5, C6J4, and C6J5

5.4 Second approach

Results presented in Section 5.3 shown that the proposed approach permits scheduling ground handling vehicles as a whole in an acceptable time. In this sense, a set of Pareto solutions were obtained modifying the order in which vehicles are scheduled. However, the number of explored sequences might not be sufficient to ensure the best Pareto set. Moreover, the computational effort required could be an obstacle depending on the available time for the scheduling process. As described in Section 4.4, we have conceived a variation of the

approach allowing more sequences to be explored, as well as the improvement of algorithm s performance. BCN instances C4J5 and C4J4 have been used to test this variation.

First, we aim to compare the non-dominated solutions obtained with both approaches. With this goal, the *SIM* is employed to generate the sequences. Each routing problem is solved using only the I3 heuristic as explained in Section 4.6, i.e, the local search (LS) process is not performed. Obtained solutions for instance C4J5 are shown in Figure 5.8.

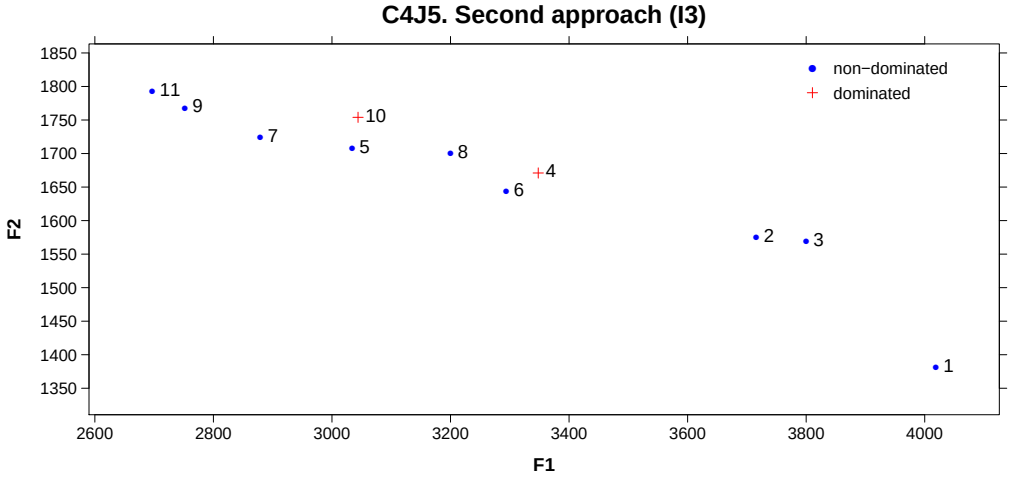

Figure 5.8: Obtained solutions with the second approach using only the I3 heuristic for instance C4J5. Each solution is labeled according to the iteration in which it is obtained. Bullets and crosses represent non-dominated and dominated solutions, respectively.

Then, the Pareto solutions are improved using the CP-based VNS-LNS, i.e 1,2,3,5,6,7,8,9, and 11. Solutions obtained by applying the local search process (I3+LS), as well as using only the I3 heuristic, are depicted in Figure 5.9.

As expected, better values are found when LS is applied. However, notice that the $F2$ improvement regarding I3 has been reached at the expense of increasing the $F1$ objective in solutions 1 and 3. Detailed results are presented in Table 5.6. For each improved solution, the associated sequence and objective values are shown. Likewise, the reduction of the original time windows (Δ) and the operation waiting time (w) are also included.

In the first iteration, sub-problem PB(7) is scheduled first with the aim to find a lower bound of $F2$. In this case, $F2$ value depends directly on

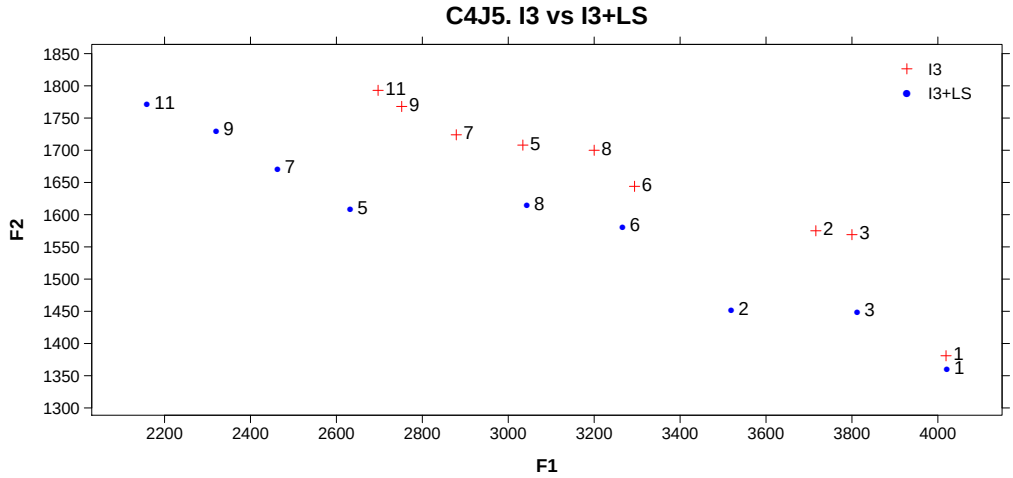

Figure 5.9: Non-dominated solutions found with the I3 heuristic and later explored with the local search process for instance C4J5. Crosses and bullets represent I3 and I3+LS solutions, respectively.

the routing problem and the quality of the found solution. The lower the value of the waiting time, the lower the value of $F2$. On the other hand, this reduces the original time windows of other operations, so a worse value of $F1$ is expected if $F2$ is lower. Nevertheless, this relation does not need to be in inverse proportion. The value of $F1$ depends not only on $F2$ and the waiting time of each vehicle type, but also on the reduction of the time windows produced by activities having precedence relations.

Differences (in percentage) for the most important parameters are summarized in Table 5.7. We see that, in general, a greater reduction of completion time provokes an increase in the time window reduction. This reduction shoots up the difference in the waiting time. As explained in Section 4.2, when time windows are reduced, it is more likely to have also smaller values of waiting time.

For comparison purposes, Pareto solutions found using the first and second approach are presented in Figure 5.10. As discussed in section 4.4, different sequences can be obtained using each approach, for instance, solutions 5,7,8,9, and 11. The problem is that we cannot ensure that sequences found with the I3 method produce Pareto solutions with respect to solutions generated with the original approach. In this case, solutions 9 and 11 obtained applying the first approach dominate solutions found with the second approach in the

Table 5.6: Results obtained for instance C4J5 at each iteration of the *SIM* using I3. Non-dominated solutions are later explored using LS. Δ and w represent the reduction of the available time window and the waiting time, respectively.

N. S.	**Sequence**	**I3**					**I3+LS**				
		F1	**F2**	Δ	**W**	**CPU(s)**	**F1**	**F2**	Δ	**W**	**CPU(s)**
1	(7,1,2,3,4,5,6)	4019	1381	3621	398	0.046	4021	1360	3702	347	1178.57
2	(5,7,1,2,3,4,6)	3716	1575	2597	1119	0.075	3519	1451	2855	722	1311.39
3	(6,5,7,1,2,3,4)	3800	1569	2510	1290	0.059	3812	1449	3085	740	1173.71
4	(5,6,7,4,1,2,3)	3348	1671	1926	1422	0.075	-	-	-	-	
5	(4,5,6,7,1,2,3)	3034	1708	1567	1467	0.059	2632	1608	1511	1121	1339.00
6	(6,5,4,7,2,3,1)	3294	1644	1988	1306	0.059	3266	1581	2358	894	1123.91
7	(2,4,5,6,7,1,3)	2879	1724	1308	1571	0.059	2463	1671	1159	1304	1180.00
8	(6,5,2,4,7,3,1)	3200	1700	1614	1586	0.06	3043	1614	1903	1140	1227.86
9	(3,2,4,5,6,7,1)	2752	1768	1094	1658	0.065	2320	1730	889	1431	1253.00
10	(6,5,3,2,4,7,1)	3044	1754	1357	1687	0.059	-	-	-	-	
11	(1,3,2,4,5,6,7)	2697	1793	924	1773	0.343	2159	1771	629	1530	1095.12

Table 5.7: Differences for the most important parameters using I3 and I3+LS

No.Sol.	**F1** (%)	**F2** (%)	Δ (%)	**W** (%)
1	0.05	-1.52	1.57	-13.82
2	-5.30	-7.87	10.28	-41.47
3	0.32	-7.65	23.11	-44.03
5	-13.25	-5.85	-3.57	-23.59
6	-0.85	-3.83	14.99	-24.96
7	-14.45	-3.07	-11.39	-17.00
8	-4.91	-5.06	17.91	-28.12
9	-15.70	-2.15	-18.74	-13.69
11	-19.95	-1.23	-31.93	-13.71

similar iterations 9 and 11. On the other hand, although sequences 5 and 7 are different, the four solutions are non-dominated. A similar situation is presented in the case of solution 8, where both solutions are non-dominated between them. However, they are dominated by the other solutions, either in the first or the second approach.

It should be remarked that CP-based VNS-LNS is a non-deterministic procedure, and different solutions can be found with the same sequence. The same sequences have been generated in iterations 1,2,3,4,6. In this sense, another inconvenience is that sequences that have produced non-dominated solutions with the original approach, could be refused when using only I3. For instance, solution 4 was refused in the first step (see Table 5.6) because it was dominated and therefore it was not chosen to be explored. This sequence could generated a non-dominated solution if the local search process had been applied (see Figure 5.6).

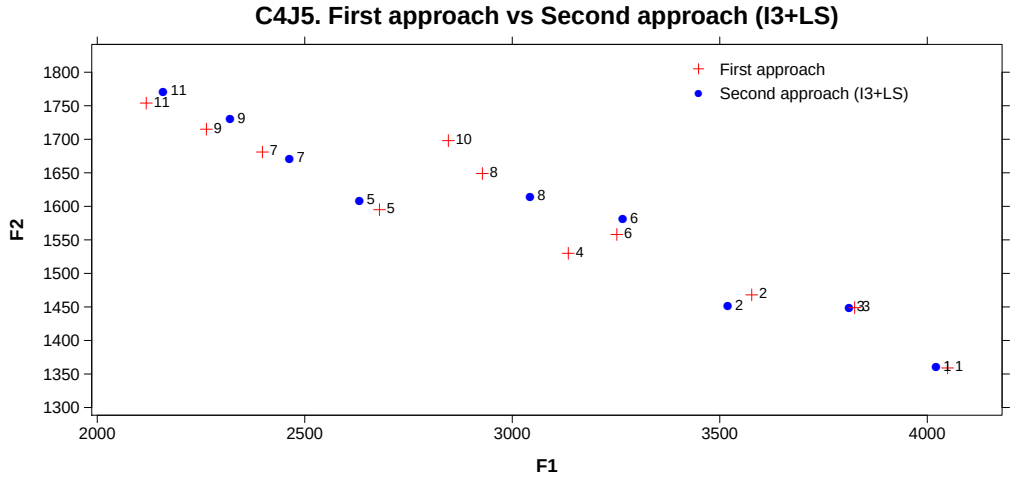

Figure 5.10: Solutions obtained using the first and second approach for instance C4J5. Crosses and bullets represent first and second approach solutions, respectively.

Finally, results obtained for instance C4J4 are presented in Figures 5.11 and 5.12. In this case, almost the same sequences (1,2,4,5,8,10) have been obtained using both the first and second approach.

Analyzing the results obtained by means of both methods, we consider that the second approach may be a proper option depending on the situation. It is a fact that found sequences are not identical. However, solutions are similar and the produced Pareto frontiers are close. The first approach is more appropriate at a tactical level, where the quality of solutions is a crucial aspect. When the available time for performing the scheduling process is lower, the second approach might be more useful. With the latter we can explore the most interesting solutions and, this way, reduce the number of times the routing problem is solved with the local search process. In this case we have chosen all non-dominated solutions found in the first step in order to compare the results with the original approach. Nevertheless, the goal of the second approach is to improve only the most promising solutions. Results in next subsection show the subset of selected Pareto solutions according to the criteria introduced in Section 4.4.

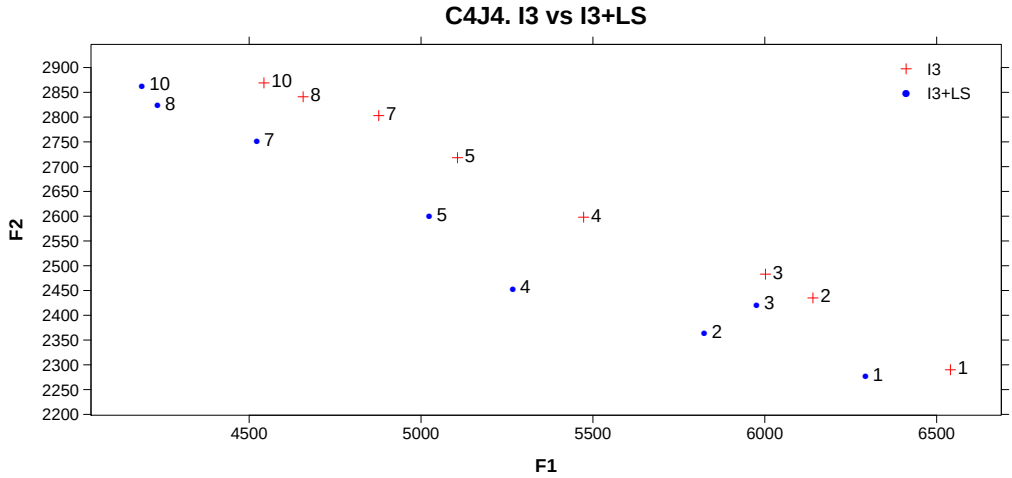

Figure 5.11: Non-dominated solutions found with I3 heuristic and later explored with the local search process for the instance C4J4. Crosses and bullets represent I3 and I3+LS solutions, respectively.

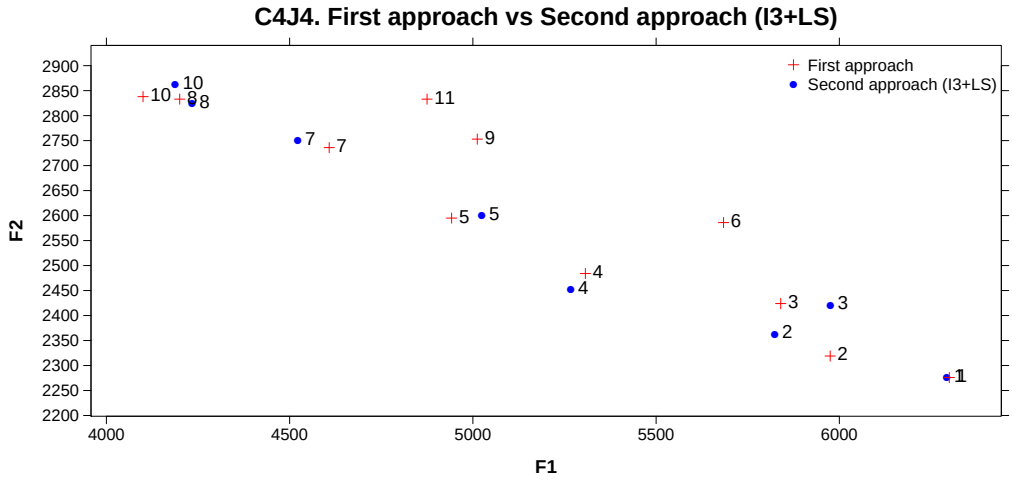

Figure 5.12: Solutions obtained using the first and second approach for the instance C4J4. Crosses and bullets represent the first and second approach solutions, respectively.

Selecting the most promising solutions

Considering the results obtained for the problem C4J5 with the I3 heuristic, an example of how the proposed selection rules are applied is described as follows.

Using the first defined criterion (see Section 4.4), the Pareto frontier is divided into three areas according to the objective values. For each area we select the solution where the greater improvement of $F1$ is reached with the lower growth of $F2$, according to equation (4.23). The relation $r(i, j)$ established in equation (4.22) is calculated for each pair of solutions at each area. Results are presented in Table 5.8. Values are sorted in descendent order, so, the first pair represents the best relation. Due to all values are greater than 1, solutions 2, 5 and 9 are the best according to the relation between the objectives. This criterion can be seen as a first step to sort the obtained Pareto solutions permitting the selection of promising sequences in a fast and simple way. Depending on the available time for scheduling the vehicles or the decision maker preferences, the number of solutions to be selected can vary. If the goal is to select one solution at each area, we have finished the selection process in this case, and these three solutions will be improved using the local search process.

Table 5.8: Relation between each pair of solutions obtained with I3 according to the objective values at each area of the Pareto frontier. Values are sorted in descendent order.

Area 1		**Area 2**		**Area 3**	
i-j	**r(i,j)**	**i-j**	**r(i,j)**	**i-j**	**r(i,j)**
3-2	14.00	8-5	20.75	7-9	2.89
1-2	1.56	6-5	4.06	7-11	1.67
1-3	1.16	6-8	1.68	9-11	1.56

Nevertheless, there are other situations where we need to consider the second rule proposed in Section 4.4 and analyze the operations individually in order to select the best solutions. A first situation is when the value of $r(i, j)$ is close to 1 and both solutions i and j can be considered equivalents. Suppose, we aim to select the first two best points of each area. In the third area, the r value of the second best pair is $r(7, 11) = 1.67$. Depending on the problem, we may determine that solutions 7 and 11 are equivalents due to the r value is around 1. In this case, the second rule is used to decide which solution to select.

Another example where we may need to focus on how the operations are scheduled is when two pairs of solutions have similar values of r. For instance, in the first area is clearer that solution 2 is better than solution 3. However, solutions 1 and 3 are more similar with respect to the relation between objectives ($r(1, 2) = 1.56$ and $r(1, 3) = 1.16$) and in this case may be more useful to decide which is the best using the second rule. Hence, we followed the two

criteria included in Section 4.4 in order to make a decision in these two situations. In any case, both criteria can be combined from the beginning of the selection process.

Table 5.9: Number of vehicles required for performing operations at each Pareto solution found using I3 and explored with LS for the instance C4J5.

		#Vehicles. I3						
N.S.	**Sequence**	**UL/L(1)**	**Cl(2)**	**Ca(3)**	**F(4)**	**PW(5)**	**TS(6)**	**PB(7)**
1	(7,1,2,3,4,5,6)	11	8	7	6	2	4	2
2	(5,7,1,2,3,4,6)	9	6	7	4	2	4	2
3	(6,5,7,1,2,3,4]	8	6	6	5	3	3	2
5	(4,5,6,7,1,2,3)	8	6	7	4	2	3	3
6	(6,5,4,7,3,2,1)	9	6	7	5	3	3	2
7	(2,4,5,6,7,1,3)	8	6	7	5	2	3	3
8	(6,5,2,4,7,3,1)	8	6	6	5	3	3	2
9	(3,2,4,5,6,7,1)	8	6	6	5	2	3	3
11	(1,3,2,4,5,6,7)	8	6	6	5	3	3	3

The schedules where the most complex operations, as well as the activity with longer duration, require less resources are the preferred in this order. The number of vehicles needed for each operation when the problem is solve using I3 is presented in Table 5.9. As shown, 6 vehicles are required to carry out operation F in solution 1 against 5 vehicles used in solution 3. Fueling is the most complex task according to the pattern defined. Hence, we choose solution 3. Regarding solutions 7 and 11, the same number of vehicles are used to perform fueling. So, we applied the second criteria. As mentioned, a more efficient use of resources in longer operations could contribute to the robustness of the solution. In this example, the two solutions need the same number of vehicles to perform UL/L, Cl, and F. Only Ca requires one vehicle more when the problem is solved using sequence 7. Therefore, we opted for solution 11.

Solutions 2, 3, 5, 6, 9, and 11 were chosen using both criteria to be improved by the local search process. In order to evaluate this decision we analyzed the case when all the non-dominated solutions were explored with I3+LS (see Table 5.6). That is, we aim to verify if the selected promising sequences are also the best solutions after applying the local search.

Three areas with the same points found with I3 are identified using local search and the relation between the objectives are shown in Table 5.10. Solutions 2 and 5 are also the best according to the first objective. In the case of the third area, sequence 11 are the best solution according to this criterion and not 9 as using I3.

Table 5.10: Relation between each pair of solutions obtained with I3+LS according to the objective values at each area of the Pareto frontier. Values are sorted in descendent order

Area 1		**Area 2**		**Area 3**	
i-j	**r(i,j)**	**i-j**	**r(i,j)**	**i-j**	**r(i,j)**
3-2	146.50	8-5	68.50	9-11	3.93
1-2	5.52	6-5	23.48	7-11	3.04
1-3	2.35	6-8	6.76	7-9	2.42

Regarding the number of vehicles, it should be remarked that the value obtained with I3 is imposed as an upper bound in the local search process. That is, regarding the routing problem, the number of vehicles needed to serve an operation is at last the same independently of the method used. However, the available time window of an operation can be different at each case. The time window reduction caused by precedence relations depends on the results obtained along the scheduling process. In other words, the variation of the objective functions as a result of solving the problem using I3 or I3+LS can result in a different number of required vehicles. The number of vehicles required for performing each operation when applying I3+LS is presented in Table 5.11.

Table 5.11: Number of vehicles required for performing operations at each Pareto solution found using I3+LS for the instance C4J5.

		#Vehicles. I3+LS						
N.S.	**Sequence**	**UL/L(1)**	**Cl(2)**	**Ca(3)**	**F(4)**	**PW(5)**	**TS(6)**	**PB(7)**
1	(7,1,2,3,4,5,6)	11	7	7	6	2	4	2
2	(5,7,1,2,3,4,6)	10	6	6	4	2	3	2
3	(6,5,7,1,2,3,4]	10	6	6	5	4	3	2
5	(4,5,6,7,1,2,3)	8	6	6	4	2	3	3
6	(6,5,4,7,3,2,1)	9	6	7	5	4	3	2
7	(2,4,5,6,7,1,3)	8	6	6	5	2	3	2
8	(6,5,2,4,7,3,1)	8	6	6	5	4	3	2
9	(3,2,4,5,6,7,1)	8	6	6	5	2	3	2
11	(1,3,2,4,5,6,7)	8	6	6	5	4	3	2

In this particular example, we have considered the number of vehicles to select between solutions 1 and 3, as well as 7 and 11. In the former, both methods give the same vehicle values for operation F. In the second case, we have refused sequence 7 because Ca needed one more vehicle. However, when the problem is solved with I3+LS, the same number of vehicles is obtained either for solution 7 and 11. So, both solutions are equivalent regarding this strategy.

In summary, only a small subset of sequences is improved with the local search process using the proposed criteria. With this schema, the computational effort can be reduced without compromising too much the quality of the solutions. The promising sequences, selected from the non-dominated points found with *SIM* using only I3 are, in most of the cases also the best sequences when the local search is applied. However, selecting the most promising solutions may be a difficult task. Different situations should be *a priori* defined, e.g. when two solutions can be considered equivalent, what to do when two pairs of solutions have similar values of r, etc. In some cases, solutions are very similar and the specified criteria are not enough to decide which solutions are the best.

5.4.1 A more exhaustive method to find the Pareto solutions

In this subsection, we aim to use the second approach to provide a more exhaustive exploration of sequences which could produce better Pareto solutions. The maximum number of sequences is a permutation of all considered vehicle types. In this problem, we have 7 vehicle types. Exploring all permutations requires an acceptable CPU time taking into account that I3 spends less than 0.1 seconds to solve one sequence. *SIM* takes less than 1 second to generate a set of solutions and the algorithm spends about 8 minutes to produce all sequences. Figure 5.13b shows all solutions for instance C4J5 as well as sequences found by *SIM* using I3. Solutions obtained for instance C4J4 are depicted in Figure 5.13a. It should be remarked that sequences produced by I3 do not need to be the same than using the original approach. However, this gives an idea of the performance of *SIM*.

Notice that solutions are more homogeneously distributed in the case of C4J4. This situation can explain the similarity of sequences obtained with the general and the second approach, shown in Figure 5.13a. *SIM* solutions are very close to the best Pareto solutions and they provide a good coverage of the frontier. Regarding C4J5, the coverage is more discrete, particularly in the second area, i.e. where both objectives are balanced. This can be caused by the fact that there is less density of points in the Pareto set and non-dominated solutions are far from the rests. The behavior of *SIM* appears better in the area where the function *F1* has the best values, i.e. the first area.

After producing the sequences, the next step is to apply the described criteria to select the most promising sequences from the Pareto solutions. We

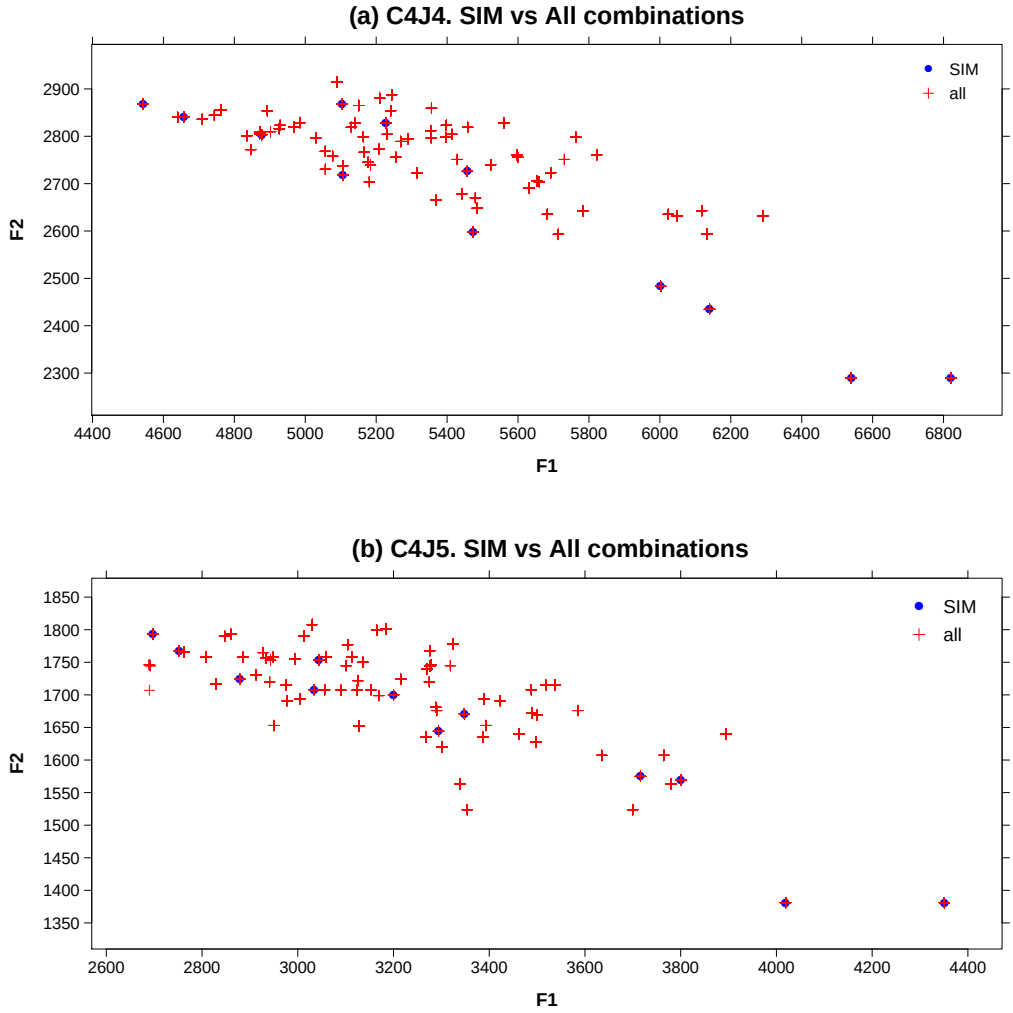

Figure 5.13: Solutions obtained with the exhaustive method and with *SIM* using I3 for instances (a) C4J4 and (b) C4J5. Bullets and crosses represent *SIM* and all solutions, respectively.

have also chosen C4J5 to illustrate this step. Non-dominated solutions are outlined in Table 5.12.

The relation between objectives has been determined according to the first criterion (See Section 4.22) and is shown in Table 5.13. We have also divided the Pareto frontier in three areas. However, there is only one solution (1) in the

Table 5.12: Non-dominated solutions for instance C4J5 obtained using the exhaustive method

N. S.	F1	F2	Sequence	#Vehicles						
				UL/L(1)	Cl(2)	Ca(3)	F(4)	PW(5)	TS(6)	PB(7)
1	3955	1373	(7,1,2,3,4,5,6)	11	8	7	6	2	2	2
2	3354	1524	(4,7,5,6,3,1,2)	10	7	7	5	2	4	2
3	3339	1563	(3,7,1,5,4,2,6)	10	6	6	5	2	4	2
4	3301	1620	(4,5,7,6,2,1,3)	9	6	7	6	2	4	2
5	3268	1635	(2,4,7,5,1,3,6)	8	6	6	5	2	4	2
6	3128	1652	(4,5,2,7,1,3,6)	8	6	6	5	2	4	2
7	2950	1653	(3,4,7,1,2,5,6)	9	6	6	5	2	4	2
8	2690	1707	(2,4,3,7,1,5,6)	8	6	6	5	2	4	2
9	2562	1758	(1,4,2,3,7,5,6)	8	6	6	5	2	4	2

first area. Then, we have directly selected sequence 1. Regarding the other two areas, we aim to select the first two best solutions. In this case, there are several pairs of solutions where the value of $r(i, j)$ is lower than 1. As mentioned in Section 4.4, in these cases the $F1$ objective is less improved at expenses of a higher increase of $F2$ and solution i is the preferred. Thus, we should also analyze these relations besides the maximum value of $r(i, j)$. Solution 6 is the best point and it is not part of any relation lower than 1. Solution 5 seems the second best. However, solution 2 is better than 5 according to the value of r in the pair $2 - 5$. For that reason, we select solutions 6 and 2 in the second area. In the case of the third area, we opted for sequences 8 and 9.

Table 5.13: Relation between each pair of non-dominate solutions obtained according to the objective values at each area of the Pareto frontier. Values are sorted in descendent order

Area 2		Area 3	
i-j	r(i,j)	i-j	r(i,j)
5-6	8.24	7-8	4.81
4-6	5.41	7-9	3.7
3-6	2.37	8-9	2.51
4-5	2.20		
2-6	1.77		
3-5	0.99		
2-5	0.77		
3-4	0.67		
2-4	0.55		
2-3	0.38		

In Section 5.4, the *SIM* was used to generated the sequences, where the most promising solutions have been selected to be improved. Figure 5.14 shows the solutions found with each schema after applying the local search process.

Notice that Pareto frontiers are similar in spite of the difference between *SIM* and the exhaustive method outlined in Figure 5.13b.

Taking into account that only a small subset of solutions is explored in a minimum time, we consider that *SIM* provides a good representation of the Pareto set. Nevertheless, better Pareto solutions can be obtained applying a more exhaustive exploration. The second approach permits exploring a greater number of sequences without increasing the computational time. In this sense, the *SIM* method can be adapted to be used with the second approach in order to intensify the exploration and improve the coverage of the Pareto frontier.

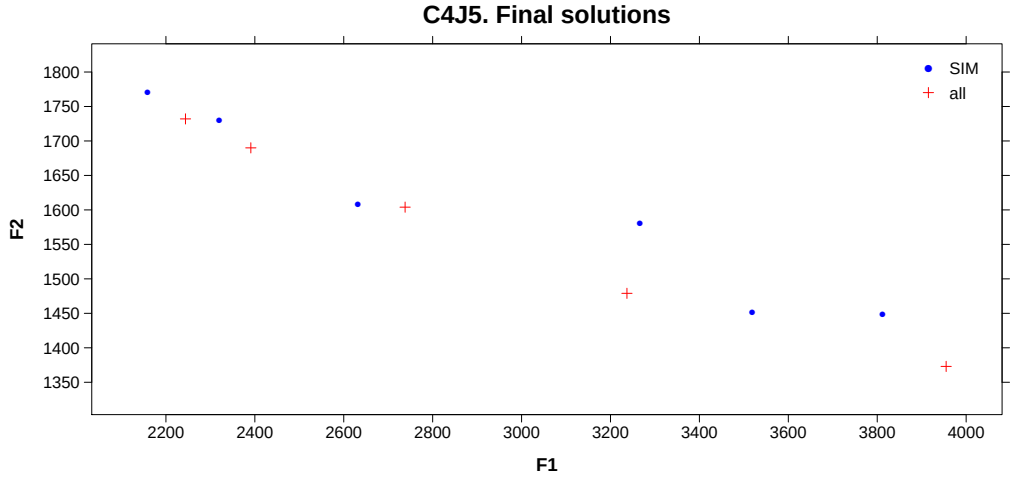

Figure 5.14: Final solutions found using *SIM* and the exhaustive exploration for the instance C4J5. Bullets and crosses represent *SIM* and exhaustive method solutions, respectively.

Chapter 6

Conclusions and Future Research

In the present thesis we have introduced a new approach for scheduling ground handling services. Different operations and types of vehicles, as well as the interconnections between activities during a typical turnaround, have been modeled in order to study this problem as a whole. Furthermore, ground handling is tackled as a multi-objective optimization problem, where two objectives were determined: (i) minimizing the waiting time of the operations and the total reduction of the available time windows; and (ii) minimizing the completion time of the aircraft turnaround. This way, scheduling decisions made for each resource are combined in order to favor the robustness of the solution and to contribute to the optimization of the overall process.

A new method, called *Sequence Iterative Method*, has been developed to enhance the global solution while dealing with the multi-objective problem. *SIM* is an *a posteriori* heuristic that modifies iteratively the sequence for solving the VRPTW sub-problems in such a way that approximations to the non-dominated solutions of the problem are obtained. This set of solutions representing the best compromise between objectives facilitates the selection of the most satisfactory solution under diverse circumstances. Furthermore, this avoids specifying preferences beforehand that can lead to inaccurate solutions. Considering that solving each VRPTW is a complex problem, the *SIM* was defined to find a minimum set of Pareto solutions that can provide a proper representation of the Pareto frontier.

As described in Section 4.1, the problem has been decomposed into two levels that interact along the solving process. The first level ensures that

precedence restrictions between activities are respected and the turnaround is finished on time. For that reason, time windows are calculated considering all involved operations. In the second level, each type of vehicle servicing a specific operation is scheduled independently, which leads to solving a set of VRPTW sub-problems. They are solved sequentially and decisions made at each sub-problem are propagated to the other VRPTWs by reducing their time windows. This decomposition schema, which comes from the Job Shop Scheduling problem, is an appropriate representation of the ground handling problem. Each operation is performed independently using specific resources and, at the same time, they are interdependent due to the precedence restrictions. Furthermore, this schema permits the reduction of the computational effort to embrace the complete problem. Solving a routing problem for each type of vehicle is more efficient than a unique model including all vehicles, although it does not guarantee reaching a global optimum. Local routing solutions can be integrated to obtain a consistent complete solution so the global approach of the proposal is kept. Moreover, planning each type of vehicle separately allows the development of specific methods to address the particularities of each operation.

Both levels are modeled using the *Constraint Programming* paradigm. Time window calculations, as well as their reduction process, have been accomplished in a very efficient way through the propagation mechanism of CP. In order to deal with the routing problems, we have employed a VRPTW model derived from Kilby and Shaw formulation [112], where minimizing the operation waiting time is the objective function in our case. The flexibility of CP facilitates the extension of this formulation to cope with the special features of each handling activity. In addition, scheduling vehicles using CP is benefited from the expressiveness provided by this paradigm to model different operational constraints.

The application of pure CP methods to solve the VRPTW is expensive in computational time. For that reason hybrid strategies combining the advantages of CP with the efficiency of the search process provided by heuristics are adopted to schedule the vehicles. Each routing problem is solved in two stages in order to improve the performance of the approach. First, a quick initial solution is obtained using the *Insertion Heuristics* method [19]. A CP-based methodology combined with *Variable Neighborhood Search* and *Large Neighborhood Search* [18] is applied after as a local search process.

As explained in Section 3.2.1, the I3 variant of the *Insertion Heuristic* is selected due to it is the most suitable for dealing with our optimization

objective. In this method, the route construction is guided by geographical and time criteria with similar importance. In addition, the included parameter giving priority to the most pressing customers can lead to reduce the number of required vehicles. The parameters of the heuristic have been adjusted in Section 4.3.2 in order to tune the algorithm to get a compromise between minimizing operation waiting time and the number of routes. In the CP-based VND-LNS methodology described in Section 4.3.3, local search process is enclosed in CP through two operators based on LNS. A variation of the RPOP operator proposed in [18], and the SMART [80] are included in our case. Finally, VND is used in this methodology to guide the search and escape from local minima. As result show, this procedure has demonstrated to be effective to improve these initial solutions.

Due to the sequence for scheduling the different types of vehicles has an important influence on the solution quality, we also propose a version of the approach that explores exhaustively all sequences. With this purpose, the scheduling process is performed in two steps. First, solutions are calculated solving the routing problems with the constructive heuristic I3. Then, a set of promising solutions is selected to be improved with the local search process, being the most time expensive procedure of the approach. This variation permits a greater number of sequences to be explored leading to obtain better Pareto solutions or a wider Pareto set.

Two criteria are defined in order to decide which solutions to improve by the hybrid methodology. With the first criterion, the Pareto frontier is divided into different areas according to the values of the objective functions. Solutions with the best relation between objectives are selected at each area, i.e. solutions where the greater improvement of one goal can be reached with the least impact on the other goal. Thus, we select the solutions that best represent different ranges from the total set of Pareto points. The second criterion is used to decide which solutions are the best from the set of points obtained after applying the first rule. It chooses solutions where the minimum number of vehicles is required to perform complex operations, such as fueling or activities with longer duration. Saving in fueling vehicles, which are resources that have to be thoroughly maintained, reduces company costs and contribute to avoid delays caused by unavailability. Moreover, schedules where longer operations use less vehicles could favor the robustness of the solution since they leave spare vehicles that could be used in case of unexpected events or delays.

The first approach was assessed using real-life data from Palma the Mal-

lorca and Barcelona airports and specifications from aircraft manufacturers. Results included in Section 5.3 show our approach is able to schedule ground handling vehicles as a whole obtaining better global solutions according to the defined optimization objectives. Different solutions representing a trade-off between objectives were found modifying the order in which operations are scheduled. Moreover, the number of vehicles needed to serve the activities can change according to this order. In these cases, the second criterion defined in the second approach can be very useful to select between two solutions with similar values of the objective functions. In addition, allowing some waiting time on the less constrained operations leads to savings on resources utilization without affecting the completion time of the turnaround.

Finally, the second approach has been tested using instances from Barcelona airport data. First, instances were solved with *SIM* using only the I3 and all non-dominated solutions were improved using the local search process. Final solutions were compared with the results obtained with the first approach. As experiments show, an acceptable number of the same sequences were produced with both schema, and obtained Pareto frontiers are very similar. Then, the second approach was employed to provide a more exhaustive exploration of sequences. The two proposed criteria were applied in both cases to select the most promising solutions from the non-dominated points. Although solutions generated with all combination were better, the coverage of the Pareto frontier reached by *SIM* is comparable to the exhaustive set.

With these approaches we propose two alternatives to solve the problem according to the particular situation. The first approach provides an efficient and easy method to obtain a set of good solutions with an acceptable computational effort. Any parameter or criterion for selecting promising solutions needs to be defined beforehand, which simplifies the decision process. In contrast, the time required can be a drawback depending on the available time for scheduling the vehicles. In this case, the second approach is more useful. Additionally, if the quality of Pareto solutions or the Pareto set coverage is a crucial issue, the second method can be more appropriate. However, special attention should be paid to criteria definition in order to select the most interesting solutions regarding decision maker preferences. Moreover, it should be considered the potential of the constructive heuristic to produce sequences that can generate non-dominated solutions. Taking into account these considerations both approaches are able to schedule handling vehicles as a whole in real scenarios.

6.1 Contributions of this work

In the previous section we have discussed the contributions of this thesis. They are summarized as follows:

- A first approach has been proposed for solving the ground handling scheduling problem with a global perspective.
- The scheduling decisions made for each operation during turnaround have been integrated in order to improve the global process. This integration contributes to the scheduling robustness and to reduce the completion time of aircraft turnarounds using vehicles efficiently.
- The ground handling scheduling problem has been modeled as a multi-objective optimization problem. Two objective were defined: (i) minimizing the total operation waiting time and the reduction of time windows, and (ii) minimizing the total completion time of the turnaround.
- A new method has been developed to solve the multi-objective optimization problem. Modifying the sequence in which operations are scheduled, a range of potential Pareto solutions is generated. Providing a set of solutions avoids *a priori* specifications of preferences and allows decision makers to select the solution that better suits the problem.
- The ground handling scheduling problem has been decomposed using a workcenter-based decomposition schema. This schema permits modeling and solving the problem globally, in a natural and efficient way.
- A CP method has been implemented to ensure temporal and precedence restrictions between activities during turnaround. Constraint propagation provides a fast and effective resource to deal with this kind of process.
- The scheduling of vehicles performing each operation has been developed in two steps in order to reduce the computational effort. The I3 heuristic has been implemented to obtain a quick first solution for the routing problem, which is later improved using a CP-based VND-LNS methodology.
- A variation of the approach has been implemented in order to increase the number of explored sequences to obtain the Pareto solutions. Solutions are first calculated with the I3 heuristic and only the most promising

solutions are improved with the local search process. In addition, this schema can be used to reduce the time to produce a set of solutions without compromising the quality of the decision process.

- A set of instances has been developed using real-data from two important Spanish airports in order to test the proposed approaches. With the first approach, a set of good solutions with a trade-off between objectives is produced in an acceptable time. Depending on the problem and the desired number of solutions, the second approach appears as an interesting alternative for reducing the execution time and improving the quality of the Pareto solutions set.

Publications

The work presented in this thesis has been partially included in the following journal article:

- S. Padron, D. Guimarans, J.J. Ramos, S. Fitouri-Trabelsi. A bi-objective approach for scheduling ground handling vehicles in airports. *Journal of Computers and Operations Research* (Submitted)

6.2 Future Research

The work presented in this thesis is a first approach to cope with the ground handling scheduling problem as a whole, so many open lines remain for further research. In this section, we introduce a review of some of these aspects:

- Other operations are to be modeled, such as baggage transportation and the passenger transfer when aircraft are parked at a remote stand. Passenger and baggage transportation have special features in relation to other ground handling activities. Vehicles perform these operations in two parts: passengers or bags are picked up at a specific location, such as gates or baggage facilities (or at an aircraft stand), and they are delivered to an aircraft stand (or to any location). Vehicles performing these operations can be considered as special cases of the Pickup and Delivery with Time Window problem (PDTW). In this sense, picking up passengers or bags from (or to) different aircraft is not allowed in the same trip. Furthermore, they can be also modeled as an instance

of the Split Delivery Vehicle Routing Problem (SDVRP). Usually, due to vehicle limited capacity or safety standards, more than one trip is needed to carry out these activities. Even when such trips can be done with different vehicles, they should not arrive to the stand at the same time. The CP model corresponding to each VRPTW sub-problem may be easily extended to other VRPTW, particularly this special case of PDTW.

- Heterogeneous fleets are to be considered. Usually, vehicles are compatible with a specific type of aircraft and cannot serve other types. In this sense, the flexibility of the adopted CP-based local search process allows introducing new constraints in the model with minimum modifications in the methodology.

- The *SIM* method is to be enhanced to explore a larger number of sequences when it is used in the second approach. More intensive strategies are to be studied for modifying sequences in order to improve the obtained Pareto solutions. In the current method, the sub-problem with the highest value of $f1$ is selected to be included in B and the process is launched with the new sequence. This step can be repeated with the second or third sub-problem with the highest $f1$. The sub-problem which gives the greater improvement of $F1$ is to be finally selected to be included in B.

- Other criteria are to be defined to select the most promising solutions in the modified version of the approach. Operations with more precedence constrains are very likely to produce delays during the turnaround. Scheduling these activities with the minimum operation waiting time can contribute to the overall process optimization.

Bibliography

[1] EUROCONTROL, Challenges of air transport 2030, surveys of expert views, 2008.

[2] A. Air Transport Action Group, The economic and social benefits of air transport, 2005.

[3] EUROCONTROL, CODA Digest - Annual 2012. Delays to Air Transport in Europe, 2012.

[4] EUROCONTROL, Airport CDM Operational Concept Document, 2006.

[5] SESAR, Milestone Deliverable D1: Air Transport Framework The Current Situation, 2008.

[6] TITAN, Turnaround Integration in Trajectory And Network. Operational Concept - Issue 1, 2010.

[7] A. Norin, T. A. Granberg, P. Varbrand, and D. Yuan, Integrating optimization and simulation to gain more efficient airport logistics, *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009*, 2009.

[8] H. Fricke and M. Schultz, Delay impacts onto turnaround performance, in *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)*, pp. 1 10, 2009.

[9] Council directive 96/67/ec, *Official Journal of the European Communities*, vol. 36, no. 272, 1996.

[10] ARC, Study on the Impact of Directive 96 / 67 / EC on Ground Handling Services 1996-2007 Final Report, 2009.

[11] C.-L. Wu, *Airline Operations and Delay Management: Insights from Airline Economics, Networks, and Strategic Schedule Planning*. Ashgate, 2010.

[12] T. Clausen, *Airport ground staff scheduling*. PhD thesis, Technical University of Denmark, 2011.

[13] S. Schmidberger, L. Bals, E. Hartmann, and C. Jahns, Ground handling services at European hub airports: Development of a performance measurement system for benchmarking, *International Journal of Production Economics*, vol. 117, no. 1, pp. 104 116, 2009.

[14] Y. Du, Q. Zhang, and Q. Chen, ACO-IH: An improved ant colony optimization algorithm for Airport Ground Service Scheduling, *2008 IEEE International Conference on Industrial Technology*, pp. 1 6, 2008.

[15] S. C. Ho and J. M. Leung, Solving a manpower scheduling problem for airline catering using metaheuristics, *European Journal of Operational Research*, vol. 202, pp. 903 921, May 2010.

[16] Y. Collette and P. Siarry, *Multiobjective Optimization: Principles and Case Studies*. Springer, 2003.

[17] K. Sourirajan and R. Uzsoy, Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication, *Journal of Scheduling*, vol. 10, no. 1, pp. 41 65, 2007.

[18] D. Guimarans, *Hybrid algorithms for solving routing problems*. PhD thesis, Autonomus University of Barcelona, 2012.

[19] M. M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, *Operations research*, vol. 35, no. 2, pp. 254 265, 1987.

[20] A. C. R. Program, *Airport Passenger Terminal Planning and Design: Guidebook*, vol. 1. Airport Cooperative Research Program, 2010.

[21] J. Portrait, *Flight-Catering*, pp. 39 55. London: Behrs Verlag, 2007.

[22] IATA Airport Handling Manual 28th Edition, 2008.

[23] A. F. Abdelghany and K. Abdelghany, *Modeling Applications in the Airline Industry*. Ashgate, 2009.

[24] P. V. Leeuwen, CAED D2 : Modelling the Turnaround Process CARE INO III : The Co-ordinated Airport through Extreme Decoupling, 2007.

[25] TITAN, Turnaround Integration in Trajectory And Network. Analysis of current situation, 2010.

[26] C. Barnhart, P. Belobaba, and A. R. Odoni, Applications of Operations Research in the Air Transport Industry, *Transportation Science*, vol. 37, pp. 368 391, Nov. 2003.

[27] U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch, Flight gate scheduling: State-of-the-art and recent developments, *Omega*, vol. 35, no. 3, pp. 326 334, 2007.

[28] H. Farhan, H. Owaied, and M. Shanableh, Modeling Airplane Load Sheet using Wireless Communication, *Information Technology Journal*, vol. 10, no. 1, pp. 213 218, 2011.

[29] FAA, *Aircraft weight and balance handbook*. Oklahoma City: U.S. Department of tranSportation, Federal Aviation Administration, 2007.

[30] Huawei, Huawei Airport Wireless Ground Handling Solution. http://enterprise.huawei.com/en/solutions/trade/transportation/aviation/hw-198598.htm.

[31] INFORM, Groundstar. http://www.inform-software.com/products/groundstar/.

[32] INFORM, Ground Handling Providers. http://www.inform-software.com/ references/aviation/ground-handling-providers/.

[33] TAMS, Flagship Project TAMS-Total Airport Mangement Suite, 2012.

[34] INFORM, Total Airport Management Suite Project Successfully Concluded, *airport-technology*, 2012.

[35] Z. E. Solutions, Equipment Fleet Management for Aviation, 2010.

[36] LufthansaSystems, Airlines solutions & Services. Ground Handling. http://www.inform-software.com/references/aviation/ground-handling-providers/.

[37] FRAPORT, Products & Services. http://www.fraport-groundservices.com/content/groundservices/en/products-services.html.

[38] G. Diepen, B. Pieters, J. van den Akker, and J. Hoogeveen, Robust planning of airport platform buses, *Computers & Operations Research*, vol. 40, pp. 747 757, Mar. 2013.

[39] S. Loth, M. Venzke, D. Moller, A. Husfeldt, V. Turau, and C. Meier, Car management on aprons (carma). developing integrated solutions for vehicle management at mid-size airports, in *Workshop of Aircraft System Technologies (AST)*, pp. 1 10, 2007.

[40] INFORM, Hub Monitoring and Steering. A software solution, in *AG-IFORS*, 2009.

[41] L. Gambardella, E. Taillard, and G. Agazzi, *MACS-VRPTW. A multiple ant colony system for vehicle routing problems with time windows.* London: McGraw-Hill, 1999.

[42] K. R. Baker, Heuristic procedures for scheduling job families with setups and due dates, *Naval Research Logistics (NRL)*, vol. 46, no. 8, pp. 978 991, 1999.

[43] H. C. Lau, Combinatorial approaches for hard problems in manpower scheduling, *Journal of Operations Research Society of Japan*, vol. 39, 1996.

[44] Z.-L. Chen and H. Xu, Dynamic Column Generation for Dynamic Vehicle Routing with Time Windows, *Transportation Science*, vol. 40, pp. 74 88, Feb. 2006.

[45] K. Kuhn and S. Loth, Airport service vehicle scheduling, in *Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009)*, 2009.

[46] ICAO, *Advanced Surface Movement Guidance and Control Systems (A-SMGCS) manual.* First Edition, 2004.

[47] T. Feo and M. Resende, Greedy randomized adaptive search procedures, *Journal of Global Optimization*, vol. 6, no. 2, pp. 109 133, 1995.

[48] F. Rossi, P. van Beek, and T. Walsh, *Handbook of Constraint Programming*. Elsevier Science Inc., 2006.

[49] F. Focacci, F. Laburthe, and A. Lodi, *Local search and constraint programming*. Boston, MA: Kluwer Academic, 2003.

[50] K. Apt and M. G. Wallace, Constraint Logic Programming using ECLiPSe, 2007.

[51] P. van Hentenryck and L. Michel, *Constraint-based local search*. The MIT Press, 2009.

[52] *ILOG Solver 6.3 Reference Manual*. France: ILOG, 2006.

[53] N. Tamura, Cream version 1.2 programmers guide. Available online: http://bach.istc.kobe-u.ac.jp/cream/ (last accessed 5th Jan. 2012), 2004.

[54] C. Bessiere, Constraint Propagation, in *Handbook of Constraint Programming* (F. Rossi, P. van Beek, and T. Walsh, eds.), pp. 29 69, Elsevier Science Inc., 2006.

[55] E. Freuder and A. Mackworth, Constraint satisfaction: An emerging paradigm, in *Handbook of Constraint Programming* (F. Rossi, P. van Beek, and T. Walsh, eds.), ch. 2, pp. 13 28, Elsevier Science Inc., 2006.

[56] A. M. Cheadle, W. Harvey, A. J. Sadler, J. Schimpf, K. Shen, and M. G. Wallace, Eclipse: A tutorial introduction, 2003.

[57] P. van Beek, Backtracking search algorithms, in *Handbook of Constraint Programming* (F. Rossi, P. van Beek, and T. Walsh, eds.), ch. 4, pp. 85 134, Elsevier Science Inc., 2006.

[58] J. Cordeau, G. Laporte, M. Savelsbergh, and D. Vigo, *Vehicle Routing Problem*, pp. 157 186. Philadelphia, PA: Elsevier, 2007.

[59] J. Cordeau, G. Desaulniers, J. Desrosiers, M. Solomon, and F. Soumis, *The VRP with time windows*, pp. 157 186. Monographs on Discrete Mathematics and Applications, Philadelphia, PA: SIAM, 2002.

[60] B. Kallehauge, Formulations and exact algorithms for the vehicle routing problem with time windows, *Computers & Operations Research*, vol. 35, no. 7, pp. 2307 2330, 2008.

[61] J. Lenstra and A. Rinnooy, Complexity of vehicle routing problem with time windows, *Networks*, vol. 11, pp. 221 227, 1981.

[62] O. Braysy and M. Gendreau, Vehicle routing problem with time windows, Part I: Route construction and local search algorithms, *Transportation science*, vol. 39, no. 1, pp. 104 118, 2005.

[63] O. Braysy and M. Gendreau, Vehicle routing problem with time windows, part II: Metaheuristics, *Transportation science*, vol. 39, no. 1, pp. 119 139, 2005.

[64] L. Jourdan, M. Basseur, and E. G. Talbi, Hybridizing exact methods and metaheuristics: A taxonomy, *European Journal of Operational Research*, vol. 199, no. 3, pp. 620 629, 2009.

[65] I. Boussaid, J. Lepagnot, and P. Siarry, A survey on optimization metaheuristics, *Information Sciences*, vol. 237, no. 0, pp. 82 117, 2013.

[66] C. Blum and A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Comput. Surv.*, vol. 35, pp. 268 308, Sept. 2003.

[67] M. Gendreau, *An introduction to Tabu Search*, pp. 37 54. Boston, MA: Kluwer Academic, 2003.

[68] A. Nikolaev and S. Jacobson, *Simulated Annealing*, pp. 1 40. Boston, MA: Kluwer Academic, 2003.

[69] C. Voudouris and E. Tsang, *Guided local search*, pp. 185 218. Boston, MA: Kluwer Academic, 2003.

[70] M. Resende and C. Ribeiro, *Greedy Randomized Adaptive Search Procedure*, pp. 185 218. Boston, MA: Kluwer Academic, 2003.

[71] N. Mladenovic and P. Hansen, Variable neighborhood search, *Computers & Operations Research*, vol. 24, no. 1, pp. 1097 1100, 1997.

[72] M. Dorigo and T. Stutzle, *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances*, pp. 185 218. Boston, MA: Kluwer Academic, 2003.

[73] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, 2003.

[74] O. Braysy, W. Dullaert, and M. Gendreau, Evolutionary algorithms for the vehicle routing problem with time windows, *Journal of Heuristics*, vol. 10, pp. 587 611, Dec. 2004.

[75] K. Ghoseiri and S. F. Ghannadpour, Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm, *Applied Soft Computing*, vol. 10, pp. 1096 1107, Sept. 2010.

[76] J. Muller, Approximative solutions to the bicriterion Vehicle Routing Problem with Time Windows, *European Journal of Operational Research*, vol. 202, no. 1, pp. 223 231, 2010.

[77] C.-B. Cheng and K.-P. Wang, Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm, *Expert Syst. Appl.*, vol. 36, pp. 7758 7763, May 2009.

[78] S. Kirkpatrick, C. Gelatt, and M. Vecchi, Optimization by simulated annealing, *Science*, no. 220, pp. 671 680, 1983.

[79] F. Glover, Future paths for integer programming and links to artificial intelligence, *Computers & Operations Research*, vol. 13, no. 5, pp. 533 549, 1986.

[80] L. M. Rousseau, M. Gendreau, and G. Pesant, Using constraint-based operators to solve the vehicle routing problem with time windows, *Journal of Heuristics*, pp. 43 58, 2002.

[81] O. Braysy, A reactive variable neighborhood search for the vehicle-routing problem with time windows, *INFORMS Journal on Computing*, pp. 1 37, 2003.

[82] G. Hasle and O. Kloster, *Geometric Modelling, Numerical Simulation, and Optimization*, ch. Industrial vehicle routing, pp. 397 435. Berlin Heidelberg: Springer-Verlag, 2007.

[83] D. Guimarans, R. Herrero, J. Ramos, and S. Padrón, Solving vehicle routing problems using constraint programming and lagrangian relaxation in a metaheuristics framework, *International Journal of Information Systems and Supply Chain Management*, vol. 4, no. 2, pp. 61 81, 2011.

[84] P. Hansen and N. Mladenovic, A tutorial on variable neighbourhood search, Technical report G-2003-46, Les Cahiers du GERAD, HEC Montreal and GERAD, Montreal, Quebec, Canada, 2003.

[85] P. Hansen and N. Mladenovic, Variable neighborhood search: principles and applications (invited review), *European Journal of Operational Research*, vol. 130, pp. 449 467, 2001.

[86] P. Hansen, N. Mladenovic, and J. Moreno Pérez, Variable neighbourhood search: methods and applications, *Annals of Operations Research*, vol. 175, pp. 367 407, 2010.

[87] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, *Principles and Practice of Constraint Programming*, pp. 417 431, 1998.

[88] D. Pisinger and S. Ropke, *Handbook of Metaheuristics*, ch. Large Neighborhood Search, pp. 399 420. Boston, MA: Kluwer Academic, 2003.

[89] R. Ahuja, O. Ergun, J. Orlin, and A. Punnen, A survey of very large-scale neighborhood search techniques, *Discrete Applied Mathematics*, vol. 123, pp. 75 102, 2002.

[90] L. Perron, P. Shaw, and V. Furnon, Propagation guided large neighborhood search, in *10th International Conference on Principles and Practice of Constraint Programming (CP-04)* (M. Wallace, ed.), vol. 3258 of *Lecture Notes in Computer Science*, (Berlin Heidelberg), pp. 468 481, Springer-Verlag, 2004.

[91] P. Shaw, A new local search algorithm providing high quality solutions to vehicle routing problems. Working paper, University of Strathclyde, Glasgow, Scotland, 1997.

[92] S. Ropke and D. Pisinger, An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, *Transportation Science*, vol. 40, no. 4, pp. 455 472, 2006.

[93] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in *4th International Conference on Principles and Practice of Constraint Programming (CP-98)*, vol. 1520 of *Lecture Notes in Computer Science*, (Berlin Heidelberg), pp. 417 431, Springer-Verlag, 1998.

[94] K. Miettinen, *Introduction to Multi-Objective Optimization: Noninteractive approaches*, vol. 5252 of *Lecture Notes in Computer Science*, ch. 1, pp. 1 25. Berlin Heidelberg: Springer-Verlag, 2008.

[95] K. Miettinen, F. Ruiz, and A. P. Wierzbicki, *Introduction to Multi-Objective Optimization: Interactive approaches*, ch. 2, pp. 27 57. Berlin Heidelberg: Springer-Verlag, 2008.

[96] C. A. Coello, G. B. Lamont, and D. A. Van Veldhuisen, *Evolutionary Algorithms for Solving Multi-Objective Problems, Second Edition*. Springer, 2007.

[97] C. M. Fonseca and P. J. Fleming, An Overview of Evolutionary Algorithms in Multiobjective Optimization, *Evolutionary Computation*, vol. 3, no. 1, pp. 1 16, 1995.

[98] A. Charnes, W. Cooper, and R. Ferguson, Optimal estimation of executive compensation by linear programming, *Management Science*, vol. 2, no. 1, pp. 138 151, 1955.

[99] A. Charnes and W. Cooper, Goal programming and multiple objective optimization, *European Journal of Operational Research*, vol. 1, no. 1, pp. 39 54, 1977.

[100] M. Gavanelli, An algorithm for multi-criteria optimization in CSPs, *CPAIOR'*, pp. 2 6, 2002.

[101] K. Deb, *Introduction to Evolutionary Multiobjective Optimization*, ch. 2, pp. 27 57. Berlin Heidelberg: Springer-Verlag, 2008.

[102] A. Garcia-Najera and J. A. Bullinaria, An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows, *Computers & Operations Research*, vol. 38, no. 1, pp. 287 300, 2011.

[103] S. Zionts and J. Wallenius, An interactive programming method for solving the multiple criteria problem, *Management Science*, no. 22, pp. 652 663, 1976.

[104] K. Miettinen and M. M. Makela, Interactive multiobjective optimization system www-nimbus on the internet, *Computers & Operations Research*, vol. 27, no. 7-8, pp. 709 723, 2000.

[105] K. Miettinen and M. M. Makela, Synchronous approach in interactive multiobjective optimization, *European Journal of Operational Research*, vol. 170, no. 3, pp. 909 922, 2006.

[106] N. Jozefowiez, F. Semet, and E.-G. Talbi, Multi-objective vehicle routing problems, *European Journal of Operational Research*, vol. 189, no. 2, pp. 293 309, 2008.

[107] S. C. Hong and Y. B. Park, A heuristic for bi-objective vehicle routing with time window constraints, *International Journal of Production Economics*, vol. 62, 1999.

[108] C. M. Liu, T. C. Chang, and L. F. Huang, Multi-objective heuristics for the vehicle routing problem, *International Journal of Operations*, vol. 3, no. 3, pp. 173 181, 2006.

[109] K. C. Tan, Y. H. Chew, and L. H. Lee, A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows, *Computational Optimization and Applications*, vol. 34, no. 1, pp. 115 151, 2005.

[110] J. Homberger and H. Gehring, A two-phase hybrid metaheuristic for the vehicle routing problem with time windows, *European Journal of Operational Research*, vol. 162, no. 1, pp. 220 238, 2005.

[111] R. Banos, J. Ortega, C. Gil, A. Fernández, and F. de Toro, A simulated annealing-based parallel multi-objective approach to vehicle routing problems with time windows, *Expert Systems with Applications*, vol. 40, no. 5, pp. 1696 1707, 2013.

[112] P. Kilby and P. Shaw, *Vehicle routing*, ch. 23, pp. 801 836. New York: Elsevier Science Inc., Oct. 2006.

[113] J. Adams, E. Balas, and D. Zawack, The shifting bottleneck procedure for job shop scheduling, *Management Science*, vol. 3, pp. 391 401, 1988.

[114] E. Balas, N. Simonetti, and A. Vazacopoulos, Job shop scheduling with setup times, deadlines and precedence constraints, *Journal of Scheduling*, vol. 11, no. 4, pp. 253 262, 2008.

[115] W. Lafayette, A Computational Study of Shifting Bottleneck Procedures for Shop Scheduling Problems, *Journal of Heuristics*, vol. 137, no. 2, pp. 111 137, 1997.

[116] W. D. Harvey and M. L. Ginsberg, Limited discrepancy search, in *International Joint Conference on Artificial intelligence IJCAI*, pp. 607 613, 1995.

[117] N. Jozefowiez, F. Glover, and M. Laguna, Multi-objective Metaheuristics for the Traveling Salesman Problem with Profits, *Journal of Mathematical Modelling and Algorithms*, vol. 7, no. 2, pp. 177 195, 2008.

[118] S. A. S. Airbus, A319 Airplane Characteristics for Airport Planning AC, 2011.

[119] S. A. S. Airbus, A320 Airplane Characteristics for Airport Planning AC, 2012.

[120] S. A. S. Airbus, A321 Airplane Characteristics for Airport Planning AC, 2011.

[121] S. A. S. Airbus, A330 Airplane Characteristics for Airport Planning AC, 2011.

[122] Boeing, B373 Airplane Characteristics for Airport Planning, 2009.

[123] Boeing, B767 Airplane Characteristics for Airport Planning, 2009.

[124] Boeing, B777 Airplane Characteristics for Airport Planning, 2009.

[125] ARP, 2012 airport traffic report, 2013.