*Manuscript
Click here to view linked References

A bi-objective approach for scheduling ground-handling vehicles in airports

Silvia Padrón*[1a], Daniel Guimarans[1,2], Juan José Ramos[1], Salma Fitouri-Trabelsi[3]

[1] *Dpt. Telecommunication and System Engineering, Universitat Autònoma de Barcelona, C/ Emprius, 2, 08202, Sabadell, Barcelona, Spain. e-mail: juanjose.ramos@uab.cat*

[2] *Optimisation Research Group, National ICT Australia (NICTA), 13 Garden Street, Eveleigh NSW 2015, Australia. e-mail: {Daniel.Guimarans@nicta.com.au}*

[3] *Mathématiques Appliquées, Informatique et Automatique pour l'Aérien (MAIAA) & Air Transportation Department, L'Ecole Nationale de l'Aviation Civile (ENAC), 7 Avenue Édouard Belin, 31055, Toulouse Cedex 04, France. e-mail: {Salma.FITOURI-TRABELSI@enac.fr}*

[*] *Corresponding Author: Tel +33 61198437*
[a] *Present address: Industrial Organization, Logistics & Technology Department, Toulouse Business School, 20 Boulevard Lascrosses, 31068 Toulouse, France, e-mail: s.padron@tbs-education.fr, silvia.padron9@gmail.com*

Abstract

In the present paper, we propose a new approach for scheduling ground-handling vehicles, tackling the problem with a global perspective. Preparing an aircraft for its next flight requires a set of interrelated services involving different types of vehicles. Planning decisions concerning each resource affect the scheduling of the other activities and the performance of the other resources. Considering the different operations and vehicles instead of scheduling each resource in isolation allows integrating decisions and contributing to the optimization of the overall ground-handling process. This goal is defined through two objectives: (i) minimizing the waiting time before an operation starts and the total reduction of corresponding time windows and (ii) minimizing the total completion time of the turnarounds. We combine different technologies and techniques to solve the problem efficiently. A new method to address this bi-objective optimization problem is also proposed. The approach has been tested using real data from two Spanish airports, thereby obtaining different solutions that represent a trade-off between both objectives. Experimental results permit inferring interesting criteria on how to optimize each resource, considering the effect on other operations. This outcome leads to more robust global solutions and to savings in resources utilization.

Keywords:
Air transportation, ground handling, multi-objective optimization, constraint programming, Vehicle Routing Problem with Time Windows

1. Introduction

The notable growth of air traffic in recent years has led to increasingly congested airports and significant flights delays. In 2012, approximately 35% of European flights were more than 5 minutes late, with an average of 30 minutes [1]. A more collaborative coordination among all the involved actors, such as airports, airlines, air traffic management, ground handlers, etc., and a better planning of airport resources are crucial to improve the operational efficiency of the air transportation system. Different efforts and important projects are currently being carried out to achieve this goal, such as the Airport-Collaborative Decision Making (A-CDM) and the Single European Sky ATM Research (SESAR) programs [2,3], which is particularly focused on Air Traffic Management.

Regarding turnaround, the TITAN Project [4] proposes to improve the efficiency of airport processes through sharing reliable and timely information among the concerned

actors. Turnaround 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, operations procedures, equipment requirements, and personnel necessary to prepare an aircraft for the next flight. Many aircraft delays can be attributed to overlong turnarounds due to a lack of planning integration of the different activities and an inefficient use of resources [5]. In addition, the ground tasks are very interdependent. Each operation is a potential source of delays that could be easily propagated to other ground operations and other airport processes [6,7].

Divisions of either airports or airlines have historically performed these operations. With the recent process of deregulation of the ground-handling market at European airports, a notable increase in the number of third-party companies has taken place [8]. This new scenario, with several ground handlers providing multiple services, further increases the importance of efficient scheduling of ground activities [9]. Due to the hierarchy of overall airport planning, ground handlers are generally not included in the decision making of other scheduling processes (flight scheduling, stand allocation, etc.). This means they must fit their planning around a set of hard constraints. These constraints include aircraft arrival, departure, turnaround time, and stand allocation, among others [10].

Thus, ground handling appears an interesting and open field for research and technology transfer. In particular, logistics in ground handling [11] and cooperative planning decisions are among the major challenges to improving the quality of ground-handling services. In this context, the development of new tools that can help with the decision making process becomes mandatory. We present a novel and efficient bi-objective approach to tackling the ground-handling scheduling problem. To the best of our knowledge, this is the first time the problem is treated as a whole in the literature. Thus far, other approaches have been developed to optimize operations in isolation [9,12,13], but they do not consider the relationships and entanglements among all the involved activities. In our approach, we do explicitly consider such relationships and entanglements to solve the problem from a global perspective. To do so, we develop a bi-objective optimization methodology and decompose the problem to apply efficient techniques. Thus, we first solve a planning problem that leads to multiple Vehicle Routing Problem with Time Windows (VRPTW) problems. These are solved individually, and decisions made on the routing are propagated to the other VRPTWs through reductions in the available time windows. This process provides a consistent method to solve the complete problem.

Ground-handling procedures are usually divided into two types: terminal and ramp. Terminal activities are performed inside the terminal buildings and concern passenger services. Ramp operations take place at the aircraft parking position between the time it arrives at the stand *(In-Blocks)* and its departure *(Off-Blocks)*. Figure 1 shows an example of the principal activities during a typical turnaround when the aircraft is parked at a contact point (the stand is connected to the terminal via a bridge).

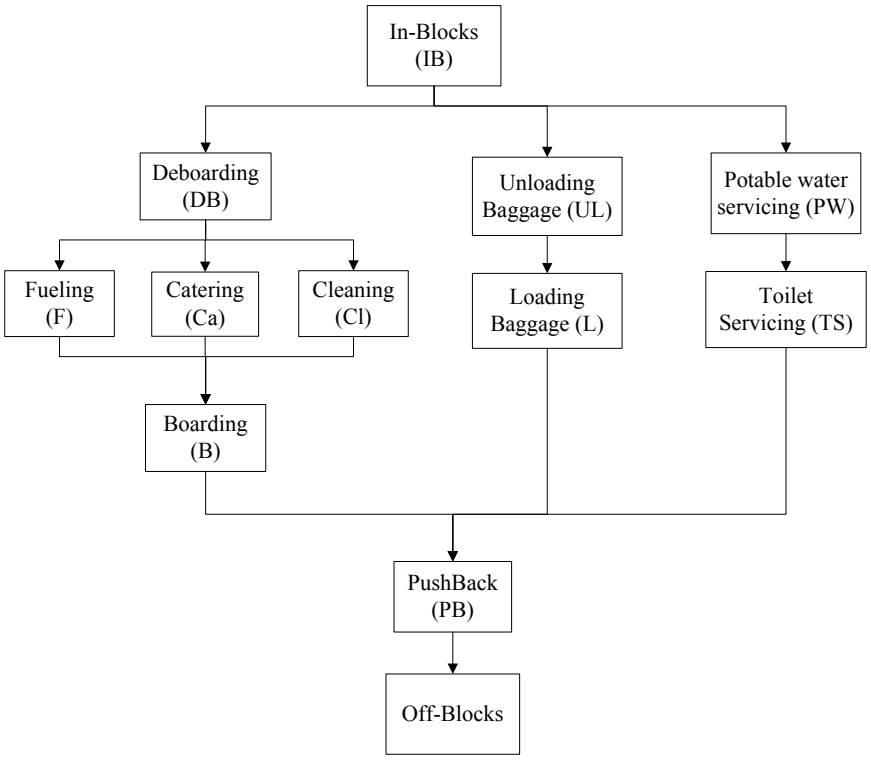


Figure 1. Example of activity flow during a turnaround at a contact point

Because the turnaround is a very complex process, its duration depends on many different variables. These include operational variables related to the aircraft type (size, number of seats), the number of tasks, parking position at a contact or remote stand, and the service time required to carry them out (full servicing or minimum servicing). Some activities are affected by precedence constraints imposed due to security issues, space requirements or airline policy; e.g., *fueling* cannot be performed simultaneously with *deboarding*/*boarding*. In some cases, the precedence constraints can be violated; e.g., fueling and *deboarding* can be performed simultaneously when a fire extinguisher is available. For hygienic reasons, the *toilet* and *potable water* servicing (collect the waste and re-equip with fresh water) cannot be performed at the same time, but either of the two can be performed first. The *catering* and *cleaning* processes usually must be finished before *boarding* starts and, sometimes, they can begin only when *deboarding* ends. The end of the turnaround process is determined by the Off-Block Time (OBT), when all doors are closed, the bridge is removed, the pushback vehicle is present and the aircraft is ready for startup 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 method 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.

Each operation is performed by a specific type of vehicle; therefore, different ground units or vehicles are necessary. According to the task, some vehicles with a given capacity must transport some quantity of resources to the aircraft stand (catering, fueling, or potable water operations) or collect waste from the aircraft (also catering, lavatory services or cleaning tasks). Likewise, some vehicles do not transport any resource (pushback, baggage loader or the fuel dispenser by underground pipelines). To simplify the model according to the goal of this work, we made some assumptions about the turnaround operations. We selected the main activities of a full servicing turnaround on aircraft parking at a contact point. In addition, we have not considered baggage transportation. Baggage transportation has special features in relation to other ground-handling activities; i.e., more than one trip is needed to carry all of the bags, more than one baggage facility can be used, etcetera. It requires a specific model and solution method and is an important field for future research.

At each aircraft, the operations must be performed within the defined turnaround time. Hence, a time window to begin the service is assigned to each activity that considers the duration of each task and the precedence constraints. Ground-handling vehicles must visit the stand where the aircraft is parked in the given time window, perform the operations during a determined service time, and travel to the next stands to perform their next activities.

Scheduling decisions made for one service affect other activities. Tasks belonging to the same aircraft are related according to the precedence restrictions, as well as to their corresponding time windows. The time when an operation begins thus could reduce the time windows of the other activities, depending on these restrictions, and consequently the performance of the vehicles servicing them. Optimizing each resource while considering the effect on other operations permits an integration of planning decisions and contributes to optimizing the overall ground service process. For instance, scheduling the vehicles to complete the tasks as closely as possible to the start of their time window leaves some room to address unexpected events or delays. This is conductive to a more robust global solution, that is, while the operations begin at their corresponding original time windows, a reschedule is avoided.

Solving this problem consists of obtaining a schedule for the ground-handling vehicles that service the aircraft performing turnaround during one working day. The schedule must satisfy temporal, precedence, and capacity constraints. We aim to minimize the operation waiting time, i.e., accomplishing each operation as early as possible in relation to its original time window, minimizing the total reduction of the time windows and considering the vehicles' utilization. This leads to a second objective: to minimize the total completion time of the ground services at each aircraft. That is, we want to balance robustness of scheduling each operation with good performance of the turnaround, using the vehicles efficiently. In addition, we focus on solving the problem at a tactical level. This means that vehicles are scheduled using estimated flight arrival and departures times, predicted duration of operations and a planned gate assignment. In this sense, we are concerned with developing a flexible algorithm that can obtain good solutions with a reasonable computational effort.

Operational planning and resource allocation in ground-handling companies are conditioned by prior mid-term decisions, usually made one month ahead, and by external changes on the day of operation. In the mid-term planning, ground handlers work based on flight schedules, aircraft types to be serviced and, perhaps, expected airport resources. Using this information, the handling equipment is allocated according to the planned workload. This quite often leads to an inefficient use of resources and being in an uncomfortable position to address unexpected events. By employing an optimization process as in the proposed approach, ground handlers can improve the utilization of their equipment while reducing costs. Moreover, we aim with this approach to increase the robustness of the operational plan for the equipment allocated in the mid-term planning. Robust solutions are crucial on the operation day to performing the turnaround within very tight time windows. Short delays caused by late start of a turnaround, perturbations during operations or ground-handler underperformance can be absorbed. Otherwise, departure delays can be produced, which turn into economic penalties.

Different activities and type of vehicles, each of them with their own available fleet, are then considered to study the ground-handling problem from a global perspective. Scheduling these vehicles to perform the services at different aircraft could be modeled as a VRPTW [14]. The ground-handling problem is separated using a decomposition schema inspired by the workcenter-based decomposition for Job Shop Scheduling [15], and one VRPTW is identified for each type of vehicle. The well-known *Insertion Heuristics* method [16] and a hybrid methodology [17] were used to solve each VRPTW sub-problem. In this methodology, Constraint Programming (CP) was combined with Large Neighborhood Search (LNS) and Variable Neighborhood Descent (VND). To address the bi-objective problem, a new method we call *Sequence Iterative Method (SIM)* has been developed. Modifying the order in which the

sub-problems are solved yields a range of solutions representing the best compromise between the two objectives.

The remainder of this article is organized as follows. In Section 2, the previous work related to vehicle scheduling in ground handling, the VRPTW and multi-objective optimization are reviewed. The decomposition approach used and the problem formulation are described in Section 3. Then, the proposed solution method and the method to address the bi-objective problem is presented in Section 4. Next, computational results are presented and discussed in Section 5. Finally, conclusions and future research lines are provided in Section 6.

2. Previous Studies

2.1 Ground-handling vehicle scheduling

Vehicle scheduling in the ground-handling process has received less attention than other airport resources; few works can be found in the literature. Moreover, most of the examples found are focused on one type of resource. To the best of our knowledge, none of these works examine the scheduling of ground-handling vehicles as a whole.

Regarding ramp operations, Du et al. [12] proposed a model to schedule fueling vehicles based on the Vehicle Routing Problem with Tight Time Windows (VRPTTW) with multiple objectives. They considered 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. An improved Ant Colony algorithm is presented to address the multi-objective problem. Clausen [9] focused on connecting baggage transportation and proposed a greedy algorithm based on an Integer Programming model for the Vehicle Routing Problem with Time Windows (VRPTW). Norin et al. [7] proposed an interesting integration of a simulation model of various operations during turnaround and the scheduling of de-icing trucks obtained by a greedy optimization algorithm. Minimizing the delays as well as the traveling time of the trucks are the objectives defined. A more sophisticated solution was proposed by Ho & Leung [13] to tackle airline catering operations including staff workload. They presented a comparison between Tabu Search and Simulated Annealing approaches to solve the problem.

2.2 *VRPTW and the Multi-objective problem*

The Vehicle Routing Problem (VRP) 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 to service a set of customers, subject to different constraints. The VRPTW is an extension of the VRP in which each customer has a time window within which the vehicle must begin its task. The VRPTW has been extensively studied, and several formulations and exact algorithms have been proposed [18]. Solving this combinatorial optimization problem is NP-hard [19]; the use of heuristic algorithms [20], metaheuristics [21] and, recently, hybridization methods [22] has been a very important field of research.

Many real-world optimization problems, including the VRPTW, involve more than one objective to be either minimized or maximized. The field of multi-objective optimization is drawing growing interest among researchers, particularly in VRPTW problems. To minimize the number of vehicles and the traveling distance, Gambardella et al. [23] implemented one of the more used methods: establishing a hierarchy between the objectives. In [12], a similar approach is used to solve a multi-objective model for scheduling fueling vehicles. Tan et al. [24] made a direct interpretation of the multi-objective problem using the concept of Pareto Optimality [25] in a hybrid multi-objective evolutionary algorithm (HMOEA), as did Ombuki et al. [26] with a genetic algorithm. A goal-programming model is proposed by Ghoseiri & Farid [27] in which the aspirations levels to the objectives and theirs deviations are minimized. Liu et al. [28] proposed a multi-objective heuristic in three phases to balance the workload, delivery

time and traveling distance among the vehicles. Hong & Park [29] also used a goal-programming model to minimize total vehicle travel time and total customer waiting time. In a soft time window context, Müller [30] used the *ε-constraint method* to minimize the total cost and the penalties associated with violations of the time windows. Here, one of the objective functions is optimized and the other is converted into a constraint. For further examples and algorithms, an overview of the research in this area is presented in [31].

Usually, there is not a single solution optimizing all objectives simultaneously. Instead, different solutions can 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 x dominates another y if and only if x is better than y 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 obtained set of non-dominated points determines the Pareto optimal solutions of the multi-objective problem. A more formal definition of multi-objective problems and dominance is presented in [25].

Methods to address Multi-Objective Problems (MOP) can be classified according to the role of the decision maker in the decision process [32]. 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 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 classified as described above can be found in the literature, each having strengths and weaknesses. A good summary is presented in [32,33]. The advantage of the *a priori* approach is that it can produce a single compromise solution without requiring further participation of the decision maker. One of the methods more widely used due to its simplicity is the *weighted method*. Specifically, it is used in the mentioned de-icing trucks scheduling [7]. 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 in which expressing the preferences through values or correlate different objectives in the same function could yield inaccurate solutions [32,34].

In our particular case, tackling the vehicle-scheduling problem as a bi-objective problem contributes to the global approach of the ground-handling problem, although it is not easy to define the relationships and the importance among the objectives. The first criterion addresses the VRPTW problem for minimizing the total customer waiting time. However, this leads to more vehicles being involved. Although minimizing the number of vehicles is not an explicit optimization objective, it should be considered to solve the problem. 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 instance, let there be two tasks with different durations, which must be finished before a third one. The task with the shorter service time will end earlier, but the third operation might need to wait for the second task anyway. Hence, it is possible to use the vehicles more efficiently, allowing waiting time on the first activity without affecting the completion time of the overall performance. The second objective is focused on the turnaround process. It is important to perform each operation as early as possible to minimize the completion time of the turnaround. On the other hand, minimizing the completion time leads to a reduction of the time windows of the operations on the same aircraft and thus increases the number of vehicles needed. For that reason, obtaining a set of solutions with a trade-off between the objectives yields the possibility of selecting the one that better suits the problem.

3. Problem Decomposition and Formulation

The decomposition schema used in this work is inspired by the workcenter-based decomposition for the Job Shop scheduling problem [15]. 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. Therefore, a method that integrates the sub-solutions and keeps the consistency of the overall solution is needed.

In our problem, each type of vehicle can be viewed as a workcenter, and the vehicles available in the fleet as the machines. Additionally, each task must be performed by just one type of vehicle. Therefore, instead of solving a global VRPTW, it is possible to solve local VRPTWs for each type of vehicle. In addition, scheduling each type of vehicle separately permits the development of specific methods to tackle special features according to the operation they perform. On the other hand, the temporal restrictions on performing each operation due to the defined turnaround time and the precedence constraints must be tackled globally. This ensures that the local solutions can be integrated to obtain a complete solution. The decomposition schema is shown in Figure 2. A procedure we called *Temporal Constraints Level Procedure (TCLP)* was defined to satisfy the temporal restrictions. For each type of vehicle, a VRPTW is modeled using the defined *Routing Level Procedure (RLP)*. *F1* and *F2* are the defined optimization objectives.

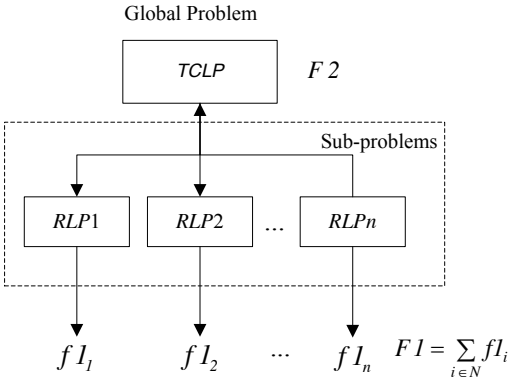


Figure 2. The problem decomposition schema

The main features of this algorithm are modeled and implemented in Constraint Programming (CP). CP is a very attractive paradigm due to its expressiveness for modeling problems with side constraints. It has received much attention in recent decades due to its potential for solving real-world combinatorial optimization problems [35]. These applications often involve a heterogeneous set of side constraints and, typically, they must address frequent update/addition of constraints [36]. The flexibility of CP is thus a powerful characteristic because adding constraints is a modeling issue and does not affect the search process. In CP, problems are expressed by means of three elements: variables, their corresponding domains, and the constraints relating these variables. Solving a problem involves the assignment of values to the variables that satisfy 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* [37]. 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 variable domains that are incompatible with propagated assignments, they are also removed [36]. Through constraint propagation, unfeasible alternatives are eliminated in advance, reducing the exploration of the search space.

The parameters of the ground-handling problem are described as follow. Let $T= \{1,...,t\}$ be the set of tasks to be carried out on the aircraft at its parking position. $N= \{1,..., n\}$ is the set of scheduled aircraft performing a turnaround, and A is the set of aircraft types. Each task $i \in T$ according to aircraft type $a \in A$ has a duration δ_{ia}, a requirement for goods ρ_{ia}, and precedence restriction rules Ψ_{ia}, which represent the set of tasks that must be finished before task i can start on aircraft type a. Each task must be performed by one type of vehicle. The set of types of vehicle is described by VT and each $vt \in VT$ has its own homogeneous fleet $M_{vt}=\{1..m_{vt}\}$ with capacity Q_{vt}.

For each aircraft $j \in N$, the STA_j and STD_j are the scheduled arrival and departure times, respectively. The aircraft type is $a_j \in A$, and $\gamma_j \in \Gamma$ is the stand where the aircraft is parked during the turnaround. Γ is the set of parking positions, and $\pi_{ij} \forall i, j \in \Gamma$, the traveling cost between stands and between them and the vehicle depot $\pi_{0,i}$.

There are $O= T \times N$ operations to be performed by the vehicles from VT. An operation o_{ij} is a task $i \in T$ performed at an aircraft $j \in N$ according to $a_j \in A$, the aircraft type of j.

3.1 Temporal Constraint Level Procedure (TCLP)

In this level, the earliest and latest start times for each operation are obtained. Let the variable τ_{ij} be the start time of each operation o_{ij} with a discretized initial domain τ:: $[STA_j..STD_j]$. The precedence restrictions are described by the following constraint:

$$\tau_{ij} \geq \tau_{i'j} + \delta_{i'a_j} \quad \forall i,i' \in T, i' \in \Psi_{ia_j}, \forall j \in N \tag{1}$$

Equation (1) ensures that temporal relationships among the tasks are fulfilled according to the type of aircraft to which they belong. When this restriction is propagated, the domain of the start time variable of each operation is reduced such that: τ_{ij}:: $[est_{ij}..lst_{ij}]$, where *est* and *lst* are the lower and upper bounds of τ and represent the operation's earliest and latest start time, respectively. During the operation scheduling process, these time windows are modified due to the precedence restrictions. Note that the duration of the last operation, i.e., pushback, is not included. As mentioned, the end of the turnaround is defined as the start time of the pushback.

Because the vehicles are routed separately, an explicit update process of the time windows is required to avoid inconsistency among sub-problems. Suppose two operations on the same aircraft, o_{11} and o_{21} such that o_{11} must be finished before o_{21} could start, that is, $\tau_{21} \geq \tau_{11} + \delta_{1a}$. 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 performs o_{11} is routed first and o_{11} is scheduled such that $\tau_{11} > est_{11}$. The value of est_{21} is now $\tau_{11} + \delta_{1a}$. Then, the time window of o_{21} must 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. Note that 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.

3.2 Routing Level Procedure (RLP)

When the time windows for each operation are calculated, a sub-problem is identified for each $vt \in VT$. We obtain the set of operations to be performed by each vt, $O_{vt}=\{o_1,..,o_n\}$, as well as the

duration d_o and the requirements for goods r_o for each $o \in O_{vt}$ according to the task and the aircraft type where the operation takes place. Because each routing problem is solved separately and to simplify the notation, we identify set O_{vt} as $O=\{1,..,n\}$. The CP model corresponding to each VRPTW sub-problem is based on the formulation presented in [38].

Let $V= O \cup F \cup L$ be the set of visits to be performed where set $O= \{1,..,n\}$ represents the operations, i.e., the aircraft to be serviced. Two special visits describe the depot from which the vehicles start and finish their routes and are modeled by sets $F= \{n+1,..,n+m\}$ and $L= \{n+m+1,...,n+2m\}$, respectively. m is the number of vehicles needed for all the operations to be accomplished by fleet $M= \{1...m\}$. Note that m is a parameter in our model. The visits $f_k= n+k$, $f_k \in F$ and $l_k= n+m+k$, $l_k \in L$, represent the first and last visit of the vehicle $k \in M$, respectively.

The following variables are defined:

- $v_i \; \forall i \in V$: the vehicle which perform each visit i with domain $v :: [1..m]$
- $p_i \; \forall i \in V–F$: the direct predecessor of a visit i with domain $p :: [1..n+m]$
- $s_i \; \forall i \in V–L$: the direct successor of a visit i with domain $s :: [1..n,n+m+1..n+2m]$
- $t_i \; \forall i \in V$: the time when the visit i is performed with domain $t :: [est_i..lst_i]$. Notice the *est* and *lst* values are those obtained from the *TCLP* procedure
- $q_i \; \forall i \in V$: the cumulated capacity after each visit i with domain $q :: [0..Q]$

As mentioned, one of the optimization goals is accomplishing 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 $w_i=t_i-est_i$ be this difference, that is, the client waiting time. The routing problem is then formulated as follows:

$$min \sum_{i \in N} w_i \tag{2}$$

Subject to

$$v_i = v_{p_i} \quad \forall i \in V - F \tag{3}$$

$$v_i = v_{s_i} \quad \forall i \in V - L \tag{4}$$

$$v_{f_k} = v_{l_k} \quad \forall k \in M \tag{5}$$

$$p_i \neq p_j \quad \forall i, j \in V - F \; \wedge i < j \tag{6}$$

$$s_i \neq s_j \quad \forall i, j \in V \; - L \wedge i < j \tag{7}$$

$$s_{p_i} = i \quad \forall i \in V - F \tag{8}$$

$$p_{s_i} = i \quad \forall i \in V - L \tag{9}$$

$$t_i \geq t_{p_i} + \pi_{p_i i} + d_{p_i} \quad \forall i \in V - L \tag{10}$$

$$t_i \leq t_{s_i} - \pi_{i s_i} - d_{s_i} \quad \forall i \in V - F \tag{11}$$

$$q_i = q_{p_i} + r_i \quad \forall i \in V - F \tag{12}$$

$$q_i = q_{s_i} - r_{s_i} \quad \forall i \in V - L \tag{13}$$

Constraints (3) and (4) ensure a visit, its predecessor and successor, are assigned to the same vehicle, as well as the first and last visit of this vehicle with constraint (5). Inequalities (6) and (7) restrict a visit to have one and only one predecessor and successor, whereas constraints (8) and (9) keep the coherence between the successor and predecessor. To ensure the temporal precedence in a route, constraints (10) and (11) specify that the time to visit a client is at least (at most) the time to visit its predecessor (successor) plus (minus) the sum of traveling time and the duration of the service. Constraints (12) and (13) were defined to address the capacity restrictions of the vehicles. The goods picked up or delivered in the route are counted to keep the load along the route.

3.3 Decomposition bi-objective problem

To obtain a more robust scheduling, the operations should be performed as early as possible within their original time windows. Because 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, most likely the waiting time would also be smaller, although it does not contribute to the solution robustness. Additionally, this reduction could lead to an increase in the vehicles needed.

Therefore, our first objective aims at performing operations as soon as possible through two arguments: minimizing the total operation waiting time and the total reduction of the time windows. Let $\Delta_i=(oest_i-est_i)+(olst_i-lst_i)$, 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 times for each operation. w_i is the client waiting time set in Section 3.2. An aggregate function *f1* is defined to describe how early the operations are performed in each routing problem:

$$f1_{vt} = \sum_{i \in N} (\Delta_i + w_i) \quad \forall vt \in VT \tag{14}$$

The first objective *F1* is defined as:

$$min \sum_{vt \in VT} f1_{vt} \tag{15}$$

Let l be the last operation on each aircraft and τ the start time of operations defined in Section 3.1; the second objective *F2* of this problem is:

$$min \sum_{j \in N} \tau_{lj} \tag{16}$$

The objective function *F2* minimizes the completion time in all aircraft.

4. Solution Method

In this section, we describe a new bi-objective algorithm developed for solving the ground-handling problem. This method is based on a workcenter-based decomposition strategy [15].

Most workcenter-based decomposition methods are derived from the Shifting Bottleneck procedure [15] developed by Adams et al. [39] and later improved by Balas et al. [40]. This decomposition heuristic was originally implemented for the classical job shop-scheduling problem and then extended to model other versions such as sequence-dependent times and workcenter problems, also called the 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, whereas the 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 [41]. Applying this procedure in our particular case, where each sub-problem is a VRPTW, can lead to long execution times; the problem becomes impractical to solve. Thus, we followed a similar schema but combined two processes to obtain a complete solution at each iteration. In the first process, which we call *Solving Process*, all sub-problems are solved one after another given a predefined order. Each time a sub-problem is solved, the time windows of the remaining sub-problems are updated to maintain consistency among the sub-solutions. The *Solving Process* is embedded in an iterative schema that we call *Sequence Iterative Method (SIM)*. The goal of this second process is to improve the overall solution when dealing with the defined bi-objective optimization problem. The sequence for solving the sub-

problems is modified at each iteration according to the solution obtained, and the *Solving Process* is repeated with the new sequence.

4.1 Solving Process

In the *Solving Process*, sub-problems are identified and solved according to a given sequence. A schema of this process is shown in Figure 3. First, the *TCLP* implemented using CP is used to find the time windows of each operation. Then, a sub-problem is identified for each type of vehicle and a routing problem is solved by means of the *RLP*.

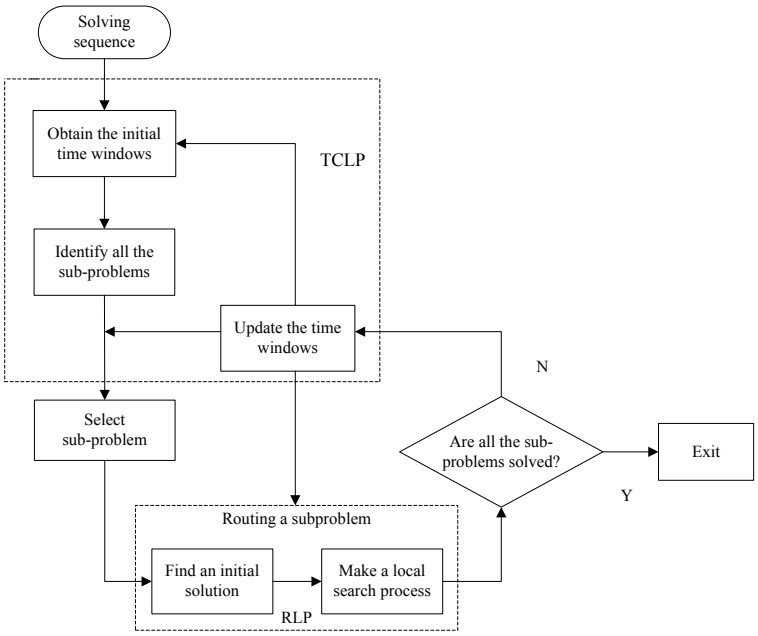


Figure 3. Flow diagram for the Solving Process

The *RLP* procedure was developed in two stages. At the first stage, a well-known route construction heuristic is used to obtain a reasonably good initial solution (see Section 4.2). The number of vehicles obtained in this step is taken as an upper bound of the vehicles needed to perform the operations. Imposing this value as the size of the available fleet, a CP local search process is applied in the second stage to improve the initial solution. With this heuristic, we assume that there are sufficient resources to handle the airport workload. The CP methodology is described in Section 4.3. The aim of this step is to improve the initial solution by minimizing the operation waiting time, *f1*.

After solving a sub-problem, an explicit process to update the time windows is needed to ensure consistency with the rest of the sub-problems, as explained in Section 3.1. Once again, taking advantage of the propagation of CP through the *TCLP*, a simple strategy is applied to maintain consistency. Finally, when all sub-problems are solved, the process is stopped. Algorithm 1 describes the proposed *Solving Process*.

1. Set $SF \leftarrow \phi$
2. Obtain the initial time windows by means of *TCLP*
3. Sub-problems solution by means of the *RLP*
   3.1 Repeat until $|SF|=|VT|$
      a. Choose $s_i \in S$
      b. Obtain an initial solution for s_i using the *I3 Insertion Heuristic*
      c. Apply the CP-based local search process
      d. Set $SF \leftarrow SF \cup \{s_i\}$
      e. Update the time windows of the sub-problems in S\SF by means of the *TCLP*

Algorithm 1. Solving Process

Let S be the set of routing sub-problems to be solved, where $|S|=|VT|$ and SF is the set of sub-problems already solved $SF \subseteq S$. To ensure coherence among sub-solutions, each time a sub-problem is solved, the scheduled decisions are propagated to the rest of the sub-problems not yet solved, keeping the already-scheduled sub-problems fixed. 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. Afterwards, 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. Infeasible intermediate solutions are avoided using this strategy. Note that time window reductions depend on the order in which sub-problems are solved and affect 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 [15], the sequence in which the machines are included in the partial schedule can reduce the re-optimization process without loss in solution quality. Then, the proposed iterative process (see Section 4.4) is developed, aiming at improving the solution by modifying the order in which sub-problems are solved.

4.2 Initial solution

The initial solution of each routing problem is obtained using the *Insertion Heuristic* method [16]. Solomon proposed three variants (I1, I2 and I3), each using a different criterion to select customers to be inserted in a route. Moreover, a set of parameters was defined that permits adjusting the solution method to solve different problems. In particular, we used the third heuristic (I3).

It is known [16, 20] that I1 yields the best results, particularly for traveling distance, which is a critical decision rule in this case. Nevertheless, in our problem, we aim to minimize the operation waiting time without compromising the number of vehicles required. Hence, the route construction should be guided by both geographical and time criteria with similar importance. In I3, the combination of the additional distance and time required to visit a customer is used to decide the best insertion place and the client to be inserted. Note that in this case, I1 and I3 are equivalent using the following parameters: (λ=0) for I1 and (α_3 =0) for I3. However, a third parameter is included in I3 to assign priority to a client having the lowest deadline to begin the service. Among other aspects, this parameter contributes to reducing the vehicle waiting time which, in turn, can lead to reducing the number of vehicles required. Thus, we have decided that I3 is the most appropriate heuristic to obtain an initial solution for our problem.

4.3 Local search process

A hybrid methodology based on [17] was selected to improve the initial solution found with I3. In this methodology, the modeling and the constraint propagation advantages of CP were combined with local search methods. Using the concept of operators based on Large Neighborhood Search (LNS) [42], the local search process is embedded in CP. These operators destroy and repair the solution to re-optimize parts of the problem. Destroy in this case means identifying a set of customers to remove from a solution. Repair refers to finding a better method to reinsert these customers into the partial solution. In addition, the methodology employs Variable Neighborhood Search (VNS) as a metaheuristic to guide the search. VNS was introduced by Hansen and Mladenovic [44] and has been applied to solve different variants of the VRP with interesting results [21,43,45,46]. Specifically, the Variable Neighborhood Descent (VND) [47] method has been adopted. A generic representation of this methodology is depicted in Figure 4.

Employing LNS within VND permits systematic exploration of the search space. Using VND, the algorithm moves from one operator to the next to escape from local minima. Any time an improved solution is found, the process is reset to the first operator; otherwise, the algorithm changes to the next operator.

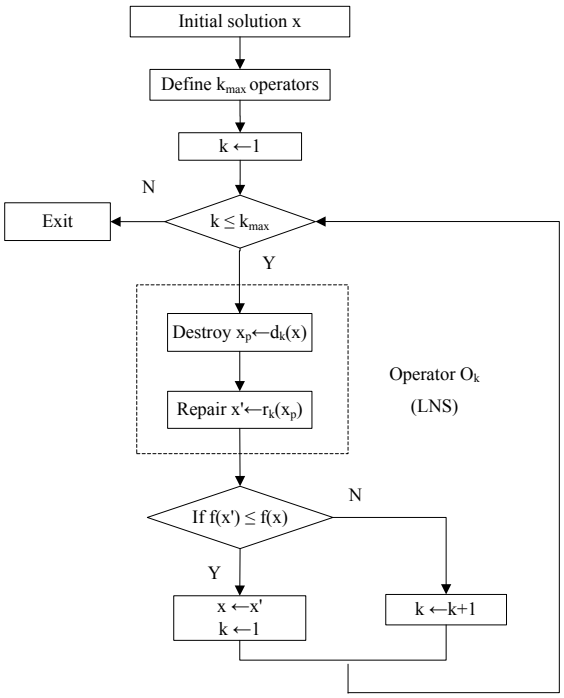


Figure 4 Flow diagram of the VND schema using LNS operators

Two operators have been used for solving our problem: the *Random Pivot OPerator (RPOP)* [17] and the *SMAll RouTing (SMART)* [43]. According to Rousseau et al. [43], the neighborhood structure defined by used operators should be different to succeed with the VND. In *RPOP*, individual customers are removed and re-inserted, whereas *SMART* works with arc exchanges.

In *RPOP*, the destroy method consists of randomly selecting a pivot customer, which is 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 [17]. Two key aspects should be considered when implementing the destroy methods in 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 limitations. 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 remove visits than using geographical rules. In the ground-handling problem, the time windows to accomplish the operations are generally tight, particularly when the activities have precedence restrictions. Therefore, we have defined the closeness of the time windows as the proximity metric of the *RPOP*. Regarding the number of clients, the *RPOP* operator is defined such that the number of visits removed is gradually increased each time the search becomes trapped in a local minimum. One single pivot is again selected when an improvement is found. An upper limit on the number of pivots is defined to avoid exploring too-large neighborhoods.

In the *SMART* operator, sequences of arcs in different routes are removed instead of customers. 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 while making a minimal detour. The same process is applied several times, and a number of precedent and successive customers are removed.

A CP-based *Branch and Bound* (BB) procedure is employed as the repairing method in the *RPOP* operator. Constraint propagation provides efficiency to this pruning of the search tree each time the upper bound is updated when a better solution is found. Additionally, the combination of VND and LNS plays an important role because different neighborhoods can be explored and revisited iteratively with improved upper bounds [17]. Because this process is exact, a limited execution time is established to avoid excessive computational time. In the case of the *SMART*, Limited Discrepancy Search (LDS) is used in the CP-BB procedure for repairing the solution as suggested Rousseau et al. [43]. The *SMART* operator is more likely to produce large neighborhoods than is the *RPOP*. With LDS, large search spaces can be explored more quickly without notably compromising the quality of the solution.

4.4 Sequence Iterative Method (SIM)

As explained in Section 2.2, we define the ground-handling problem as a Multi-objective Optimization Problem (MOP), in particular, as a bi-criteria optimization problem. The first criterion relates to the quality of the local routing decisions and depends on the contraction of the time windows during the process. The second criterion can be observed as a global objective: minimizing the completion time of turnarounds. The iterative process was implemented to re-optimize the global solution due to the decomposition regarding the two defined objectives.

Using *a posteriori* methods, the solution of the MOP is the set of non-dominated solutions, also called the Pareto optimal set. Depending on the problem, obtaining all Pareto optimal solutions is not guaranteed or can take high computational times [32,49]. Heuristic approaches, local search methods, metaheuristics, and evolutionary algorithms find approximations to the Pareto optimal set [50]. A definition of this approximation is also presented in the mentioned work: 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, heuristic methods must be developed with two important principles: (i) find non-dominated points as close as possible to the optimal set and (ii) find solutions sufficiently diverse to provide a good coverage of this set [50].

Many methods such as the w*eighted method*, *ε-constraint*, and *goal programming* among others solve the MOP by *scalarization* [31,32,49], 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. With the goals of more directly addressing the MOP and of finding a set of non-dominated points, these basic scalars, usually *a priori* methods, can be used as *a posteriori* approaches modifying the parameters. The *ε-constraint* procedure is commonly used with this approach. The problem is solved with respect to one objective and, at each iteration, the value of the second objective obtained is used as a constraint to limit the search space [30,32,49].

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 round, the sequence for solving the sub-problems is modified to find a solution in the Pareto set to cover it in the best possible way. Regardless of the type of aircraft, the ground-handling service always finishes by pushing away the aircraft from its parking position (*pushback*). We used this information to create an initial sequence to obtain a lower bound of *F2*. The sub-problems are ordered and solved such 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 2.

Definition *S:* set of sub-problems where $|S|=|VT|$, $s_{lo:}$ last operation, *BL*: sub-problems to solve before s_{lo}
SR: remainder of the sub-problems, *SSol*: set of solutions found,

1 $SSol \leftarrow \phi$
2 $SR \leftarrow$ sort *SR* by the values that were assigned to identify the sub-problems

Step A. Initial solution

Initial sequence to obtain a lower bound of *F2*

3 $BL \leftarrow \phi$
4 $S \leftarrow \{s_{lo}\} \cup SR$
5 $<F1,F2> \leftarrow SolvingProcess(S)$

Sequence to improve *F1* keeping the position of s_{lo}

6 **repeat**
7 $SR' \leftarrow$ sort *SR* by $f1_{vt}$ in a decreasing order
8 $S' \leftarrow \{s_{lo}\} \cup SR'$
9 $<F1',F2'> \leftarrow SolvingProcess(S')$
10 **if** $(F1'<F1)$ **then**
11 $SR \leftarrow SR'$
12 $S \leftarrow S'$
13 $f1_{vt} \leftarrow f1_{vt}' \ \forall vt \in VT$
14 $F1 \leftarrow F1'$
15 $F2 \leftarrow F2'$
16 **end if**
17 **until** *F1* is not improved
18 $SSol \leftarrow <F1,F2>$

Step B. Set of solutions to improve *F1* planning the rest of sub-problems before s_{lo}

19 **repeat**
20 $b \leftarrow max_{vt \in SR} \{f1_{vt}\}$
21 $BL \leftarrow \{b\} \cup BL$
22 $SR \leftarrow SR \backslash \{b\}$
23 $S \leftarrow BL \cup \{s_{lo}\} \cup SR$
24 $<F1,F2> \leftarrow SolvingProcess(S)$
25 $SSol \leftarrow <F1,F2> \cup SSol$
26 **repeat**
27 $SR' \leftarrow$ sort *SR* by $f1_{vt}$ in a decreasing order
28 $BL' \leftarrow$ sort *BL* by $f1_{vt}$ in a decreasing order
29 $S' \leftarrow BL' \cup \{s_{lo}\} \cup SR'$
30 $<F1',F2'> \leftarrow SolvingProcess(S')$
31 **if** $(F1'<F1)$ **then**
32 $SR \leftarrow SR'$
33 $BL \leftarrow BL'$
34 $S \leftarrow S'$
35 $f1_{vt} \leftarrow f1_{vt}' \ \forall vt \in VT$
36 $F1 \leftarrow F1'$
37 $F2 \leftarrow F2'$
38 **else if** $(F2'<F2)$ **then**
39 $SSol \leftarrow <F1',F2'> \cup SSol$
40 **end if**
41 **until** *F1* is not improved
42 **until** $|BL|=|S\backslash\{s_{lo}\}|$
43 **return** *SSol*

Algorithm 2. Sequence Iterative Method (SIM)

Let *S* be the set of sub-problems where each sub-problem corresponds to each type of vehicle involved, $|S|=|VT|$. Each sub-problem or type of vehicle has been identified with an integer (see

Figure 5). The order in *S* describes the sequence in which the sub-problems are solved; s_{lo} is the sub-problem corresponding to the last operation, pushback (PB) in this case; *BL* is the set of sub-problems to solve before s_{lo} such that $BL \subseteq S \backslash \{s_{lo}\}$, and *SR* represents the remainder of the sub-problems such that $SR = S - \{s_{lo}\} - BL$.

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 *SR*. Therefore, a worse value of *F1* is obtained.

At first, the elements in *SR* 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 *SR*. 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 *SR* are ordered by *f1*, that is, the total operation waiting time of the associated routing problem. Then, we repeat the process to obtain a better sequence of *SR*.

In the second step, we aim to improve the value of *F1*, planning the remainder of the sub-problems before the last operation. At each round, the sub-problem with the highest value of *f1* in *SR* is selected for inclusion in *BL* and solved first. Adding sub-problems to *BL*, that is, prioritizing the other operations with respect to s_{lo}, usually leads to a decreasing *F1*. Similar to the above step, the chosen sub-problem is scheduled within its original time windows, which leads to a lower bound of its *f1*. To find a range of solutions that represents the Pareto set, one sub-problem is included in *BL* at each iteration. Thus, an improvement of *F1* is reached when a new value of *F2* is found.

Finally, the process is repeated every time a new sub-problem is chosen. The goal of this step is to explore the search space by modifying the sequence of sub-problems in each subset while keeping the position of s_{lo}.

5. Computational Experiments

The algorithms described in this paper have been implemented in Java and linked to the open-source CP software system ECLiPSe 6.0 [51]. All tests have been performed on a non-dedicated server with an Intel Xeon processor at 2.66GHz and 16GB RAM.

In this implementation, we have specified the parameters of the algorithms as follows. First, we have tuned the I3 heuristic to obtain a compromise between minimizing operation waiting time and the number of vehicles. We defined an interval for each parameter such that α_1=[0.4,0.5], α_2=[0.4,0.5], α_3= [0.01,0.1], and the algorithm was tested on the different combinations. The best results are obtained with similar α_1 and α_2, as well as 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 the case of operations with larger time windows. Combination (α_1=0.49, α_2=0.49, α_3 =0.02) yields the best result for most of the instances tested. Therefore, we have selected these values to specify the parameters. In addition, we have adjusted the initialization criterion to minimize the operation waiting time. The client having the earliest start time to begin the service was selected to initialize the routes.

Second, we have set the parameters of the CP-based VND-LNS methodology. Regarding the *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. Because operations generally have tight time windows, in our problem, removing customers closer to the pivot is more likely to produce feasible movements than client random selection. Thus, we assign 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 entire local search process is applied during a maximum of 250 seconds to ensure an improvement over the initial solution found by the I3 heuristic.

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.

To test the algorithm, we need the information about three crucial aspects: flight schedule, aircraft parking distances, and tasks to be performed. We have used real data from two important airports in Spain: *Palma de Mallorca* (PMI) and Barcelona (BCN). In the case of *PMI*, we considered all aircraft performed a turnaround during a working day. In contrast, we focus on a handling company that provides services at the BCN airport. That is, we have employed the flight information used by the company to plan operations. Note that using data with very different characteristics is quite useful to test the efficiency of the approach. In addition to parking distances, the main difference between the two datasets used is the flight arrival frequency and the type of aircraft planned to be serviced.

The following datasets were used to create the instances:

a) Two real flight schedules: (i) one flight schedule from *PMI* airport corresponding to a summer business day with aircraft performing a turnaround; and (ii) one flight schedule from *BCN* with aircraft scheduled for service during a typical day in June. Both datasets include scheduled arrival and departure times, the type of aircraft, and the parking position.

b) Distances between the parking positions and between them and the depot. *PMI* airport has 180 parking stands, 27 of them remote stands. BCN airport has 263 parking stands. A constant speed was used to calculate vehicle travel times.

c) Tasks information: Using specifications from aircraft manufacturers [52,53,54,55,56,57,58], three types of aircraft with different sizes were modeled for the *PMI* set and nine for *BCN*. For each operation included in the problem and according to the type of aircraft, we considered the duration, precedence restrictions regarding other tasks, and the type of vehicle used.

The vehicle that unloads and loads baggage for any given aircraft is usually the same. Therefore, we have modeled 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). Precedence restrictions between the tasks for each aircraft type defined at *PMI* set are shown in Figure 5. The number assigned to identify each vehicle type is indicated in parentheses. 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 relationships, and their calculations affect the time windows of other operations.

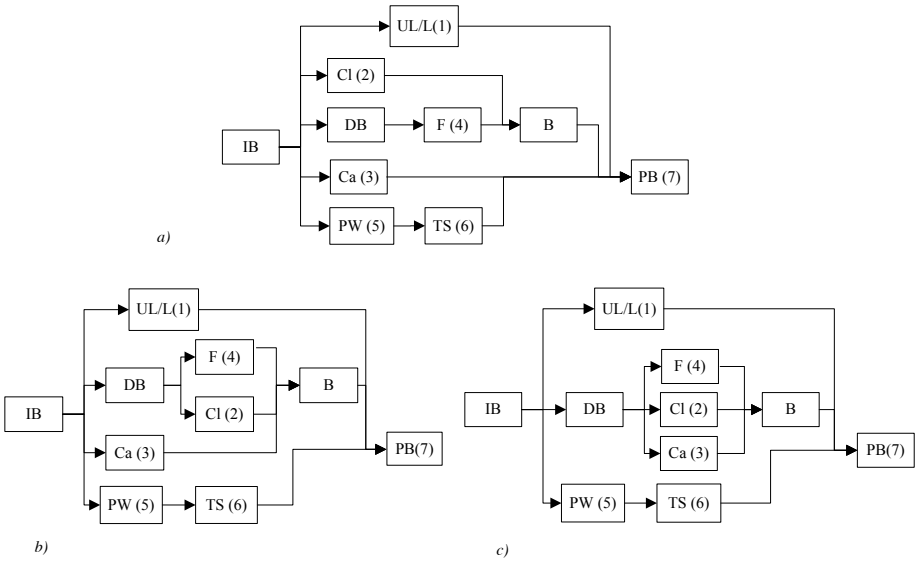


Figure 5. Precedence relationships among tasks according to the modeled aircraft type: a) Aircraft Type I, b) Aircraft Type II, c) Aircraft Type III. The number of the associated vehicle type is indicated in parentheses

According to these precedence relationships and the duration of operations, type I corresponds to aircraft with a turnaround time of between 30 and 40 minutes. Type II and type III are the aircraft with 40-50 minute and 50-60 minute turnarounds, respectively. The different characteristics of a turnaround, particularly the precedence relationships between operations at each aircraft, influence their time windows and, consequently, the solutions obtained. Airlines have some freedom within certain limits to modify the turnaround services specified by the aircraft manufacturers. For that reason, three sets of instances C1, C2 and C3 were generated, modifying the precedence constraints to test the algorithm. The first set was associated to the precedence rules presented in Figure 5. In the second set, the relationships between PW and TS were changed in types II and III, ensuring that TS is always performed before PW. In set C3, Ca is performed independently of the DB/B operations in all aircraft types.

To simplify the scheduling process, the flight schedule has been divided into three eight-hour shifts scheduled separately. In a given data set, the flight arrival frequency is relatively uniform during the day as is the expected workload. We have selected eight hours 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. 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.

In the *PMI* set, 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 shift. The last group, J3, has 83 aircraft between 15:00 and 23:00. J1 has fewer aircraft scheduled compared with J2 and J3. However, the workload is similar because most of the aircraft are planned to arrive between 4:00 and 7:00 in the morning. Each group was scheduled with the precedence rules defined at each set. We assume that workers are available to perform the service when a vehicle is assigned to an aircraft.

Regarding the *BCN* dataset, there is more variety of aircraft sizes covering scheduled flights. We have represented nine types and their corresponding precedence constraints as outlined in Figure 6. Six different precedence relationships have been identified in the aircraft modeled. Notice that aircraft III, IV, V and VI have the same restrictions between the tasks. However, operations have different durations and aircraft have different turnaround times. As with the *PMI* instances, three sets C4, C5 and C6 were generated, modifying the precedence constraints.

The first set was associated to the precedence rules described in Figure 6. In the second set, the relationships between PW and TS were changed in (b) and (c), so TS is always performed before PW. In the C6 set, the Ca is performed independently of the DB/B operations in all the precedence relationships.

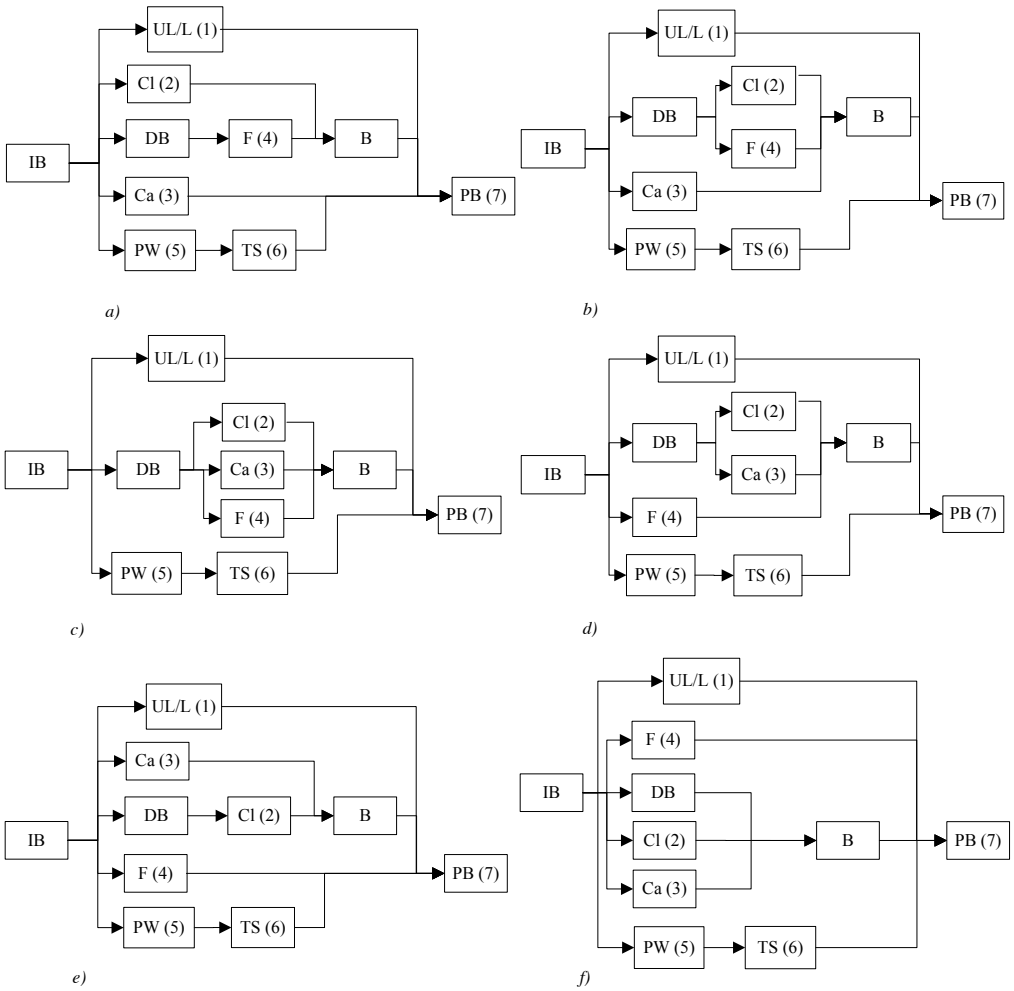


Figure 6. Precedence relationships among tasks according to the modeled aircraft type: a) Aircraft Type I, b) Aircraft Type II, c) Aircraft Type III;IV;VIII;IX, d) Aircraft Type V, e) Aircraft Type VI, f) Aircraft Type VII. The number of the associated vehicle type is indicated in parentheses

In the *BCN* case, flight arrival frequency is lower due to only a subset of the arriving aircraft during a day being handled by the company. In addition, aircraft are more uniformly distributed in the timetable with respect to *PMI*. Because the company only has aircraft planned between 6:00 and 22:00 hours, the flight schedule was divided into 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. Additionally, each group was scheduled with all different precedence constraint sets. Hence, the algorithm was tested over 15 instances: 9 instances from *PMI* (C1J1, C1J2, C1J3, C2J1, C2J2, C2J3, C3J1, C3J2, C3J3), plus 6 from *BCN* airport (C4J4, C4J5, C5J4, C5J5, C6J4, C6J5). Each instance is enumerated with the number of the set and the shift used.

5.2 Results

The detailed results of the problem C1J1 (precedence constraint rules C1, shift J1) for each iteration of the *SIM* are presented in Table 1. The results are obtained by running the algorithm only one time for each instance. In addition to the value of objectives *F1* and *F2*, the obtained

sequences of tasks and the number of vehicles used for each operation are shown. A solution obtained after iteration is rejected if it has not improved either of the two objectives regarding the previous iteration. In this instance, the sequence was modified 13 times, but only one solution was discarded for that reason. Therefore, this problem has 12 solutions. As mentioned, the value of *F1* is usually improved when the sub-problem with the highest *f1* is included in *BL* to be solved first. In contrast, modifying the sequence of sub-problems keeping the position of PB is less likely to produce an *F1* improvement. In this instance, *F1* is always increased and the process is repeated only one time. The set of solutions is presented in Figure 7. In addition, we show the time spent by the algorithm to solve each sequence.

N. It.	F1	F2	Sequence	# Vehicles							Time(s)
				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	1426.5
2	2165	1613	(6,7,1,2,3,4,5)	18	12	12	9	6	7	4	1263.07
3	1980	1621	(5,6,7,1,2,3,4)	18	11	11	9	4	7	4	1473.71
4	2425	1619	(6,5,7,4,3,2,1)	18	11	12	9	6	7	4	1545.45
5	1850	1655	(4,5,6,7,1,2,3)	18	11	11	9	4	7	4	1257.29
6	2154	1646	(6,5,4,7,3,2,1)	18	11	12	9	6	7	4	1308.99
7	1709	1695	(3,4,5,6,7,1,2)	18	11	11	9	4	7	4	1357.83
8	1998	1681	(6,5,3,4,7,2,1)	18	11	11	9	6	7	4	1471.98
9	1565	1715	(2,3,4,5,6,7,1)	18	10	11	9	4	7	4	1348.54
10	1816	1687	(6,5,3,4,2,7,1)	17	10	11	9	6	7	4	1285.05
11	1510	1736	(1,2,3,4,5,6,7)	16	10	11	9	4	7	5	1348.22
12	1792	1714	(6,5,3,4,2,1,7)	16	10	11	9	6	7	5	1360.61

Table 1. Solutions obtained for the instance C1J1 at each iteration of the SIM

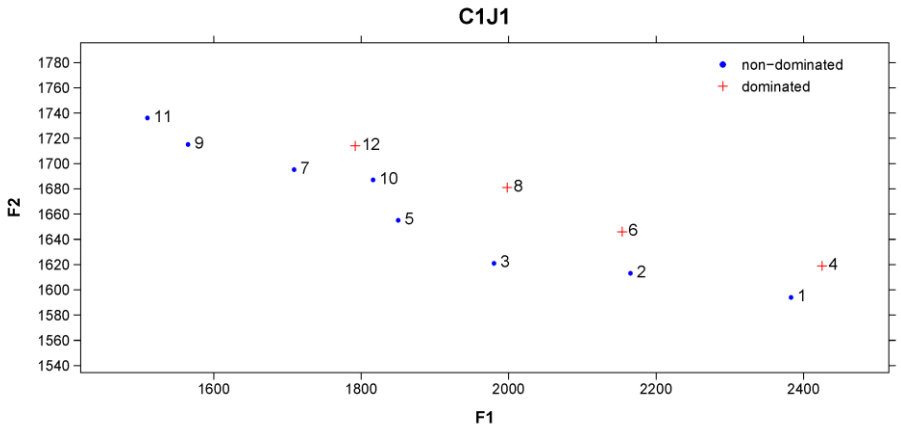


Figure 7. Found solutions for the instance C1J1. Each solution is labeled according to the iteration in which it is obtained. Non-dominated solutions, represented by bullets, define the trade-off curve between the two objectives. The crosses indicate dominated solutions

The first solution in Table 1 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 2, step 1). As shown, the vehicle numbers have been assigned following the operations order presented in Figure 5. 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 sub-problem with the highest value of *f1* is scheduled before PB, in this particular case, TS. The value of *f1* depends on the reduction of the time window and the result of the routing problem. In general, it is possible to identify activities or groups of activities without precedence relationships in the turnaround. Regarding precedence restrictions and task duration, an activity or group of activities could be more restrictive than the other ones. 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 durations must be finished before a third one. The operation with the shorter duration or the larger time window may have some waiting time without affecting the third task. We use an upper bound for the number of vehicles; therefore, scheduling operations with larger time windows usually results in higher waiting times. When the less-constrained operations are included in *BL* before the other ones, the value of *F2* is less affected. 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 relationships with other operations, but it is the longest one. Operations F, Cl and Ca are constrained according to the precedence rules defined for each type of aircraft and are usually longer than TS and PW.

Each time a new vehicle is included in *BL,* the vehicles are ordered again by their *f1* and the process is repeated. Operations PW and TS are interdependent; therefore, 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 the expense of reducing the time window of PW. Notice in Table 1 the increase in the number of vehicles needed to perform PW whenever TS is scheduled first.

Vehicle utilization is an important aspect of how the scheduling decisions of a resource affect the other ones. Notice, for instance, the increase in the vehicles needed to perform 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 consequently an increment of employed vehicles. At the first iteration, the UL/L needs 19 vehicles, whereas it uses 16 when PB is scheduled last (solutions 11 and 12). 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; therefore, solution 11 might be a good choice. On the other hand, the PB requires 4 and 5 vehicles in solutions 9 and 11, respectively. If vehicles needed to perform PB are more limited in number, or they are more expensive to use, solution 9 might be superior.

A summary of the results obtained for all instances from *PMI* data is outlined in Table 2, in which the non-dominated solutions are marked with an asterisk. These solutions are shown in Figure 8. The sequences obtained for the three shifts are very similar for each set because the precedence relationships between operations are the same. Nevertheless, the number of aircraft of each type is different at each instance. In the case of the C2 set, the relationship between vehicles that perform PW and TS was modified in two aircraft types. The service time of PW is shorter than TS in all aircraft, but in type III, the difference is very small. Values of the objective functions are similar in shift J1 because there are few aircraft of type I. On the other hand, the variation is more important in J2 due there being more aircraft of type II. Regarding the C3 set, Ca does not have precedence relationships with the rest of the tasks. This favors the value of *F2* because 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 fewer vehicles are needed to accomplish it.

N. It.	C1J1		C1J2		C1J3		C2J1		C2J2		C2J3		C3J1		C3J2		C3J3	
	F1	F2	F1	F2	F1	F2	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*	2798*	1584*	3688*	2360*	4848*	3029*
2	2165*	1613*	4064*	2346*	4281*	3057*	2369*	1607*	3309*	2357*	4524*	3053*	2748*	1600*	4271*	2345*	4715*	3068*
3	1980*	1621*	3509	2403	4608	3067	2142*	1629*	3002*	2381*	4096*	3083*	2360*	1635*	3504*	2368*	4098*	3110*
4	2425	1619	2619*	2393*	3806*	3112*	2364*	1610*	3438*	2346*	4394	3104	2640*	1613*	3036*	2407*	4640*	3087*
5	1850*	1655*	2464*	2414*	3282*	3130*	1864*	1670*	2623*	2422*	3590*	3109*	2086*	1659*	3832	2403	3741*	3161*
6	2154	1646	3162	2399	4121*	3100*	2096*	1654*	2869*	2385*	3940	3153	2476	1639	2809*	2455*	4521	3133
7	1709*	1695*	2101*	2466*	3169*	3185*	1762*	1694*	2349*	2464*	3360*	3177*	1983*	1670*	3548	2447	3521*	3181*
8	1998	1681	2904	2464	3896	3173	1784*	1680*	2756	2444	3046*	3203*	2275	1662	2543*	2478*	4271	3186
9	1565*	1715*	1833*	2490*	2818*	3207*	1671*	1707*	2157*	2488*	3428	3215	1764*	1714*	2360	2517	3264*	3218*
10	1816*	1687*	2754	2486	3547	3197	1709*	1699*	2586	2486	2790*	3262*	2170	1692	3067	2514	3964	3200
11	1510*	1736*	1756*	2508*	2622*	3264*	1513*	1712*	1924*	2513*	3279	3286	1729*	1734*	2206*	2539*	3108*	3280*
12	1792	1714	2499	2509	3301	3252	1676	1710	-	-	-	-	2054	1711	2863	2522	3690	3265
T.T.(s)	16829.53		18023.17		16375.34		18112.82		18792.33		17971.83		16254.98		18798.34		17.605	

Table 2. Summary of results obtained for PMI instances. Values of objectives F1 and F2 using precedence constraint rules in C with shifts J are given for each column. The non-dominated solutions are marked with an asterisk

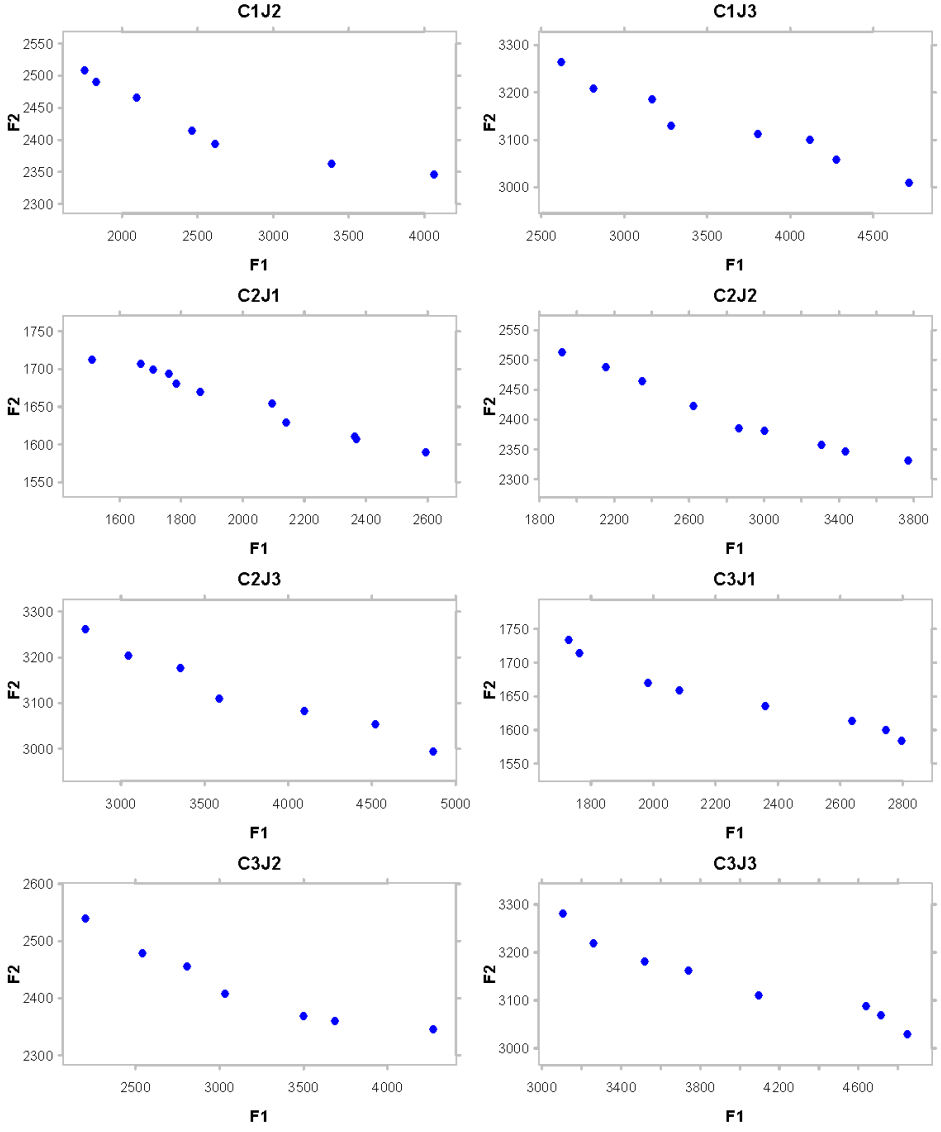


Figure 8. Pareto solutions for PMI instances C1J2, C1J3, C2J1, C2J2, C2J3, C3J1, C3J2, and C3J

In the case of *BCN* instances, the number of resources required to perform operations is one of the main differences regarding the *PMI* set. Many factors can influence these results, for example, task durations or turnaround times in each schedule. Moreover, the flight arrival frequency has a considerable effect on the number of vehicles. For instance, C4J5 and C1J1

have a similar number of scheduled aircraft, 42 and 37, respectively. However, further resources are needed in the former as shown in Tables 1 and 3.

N. It.	F1	F2	Sequence	# Vehicles							Time(s)
				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	1188.92
2	3577	1468	[5,7,1,2,3,4,6]	10	6	6	5	2	3	2	1333.95
3	3825	1449	[6,5,7,1,2,3,4]	10	6	6	5	4	3	2	1311.39
4	2999	1494	[5,6,7,2,4,1,3]	10	6	6	6	2	3	2	1331.87
5	2680	1595	[4,5,6,7,2,3,1]	9	6	6	5	2	3	2	1202.70
6	3252	1558	[6,5,4,7,2,3,1]	9	6	6	5	4	3	2	1339.12
7	2398	1681	[3,4,5,6,7,1,2]	8	6	6	5	2	3	2	1132.17
8	2928	1649	[6,5,3,4,7,1,2]	9	6	6	5	4	3	2	1290.72
9	2263	1715	[2,3,4,5,6,7,1]	8	6	6	5	2	3	2	1311.30
10	2846	1698	[6,3,2,5,4,7,1]	8	6	6	5	4	3	2	1116.89
11	2118	1754	[1,2,3,4,5,6,7]	8	6	6	5	2	3	2	1189.39

Table 3. Solutions obtained for instance C4J5 at each iteration of the SIM

A summary of solutions found for instances C4J4, C4J5, C5J4, C5J5, C6J4, and C6J5 is presented in Table 4. The non-dominated points are highlighted with an asterisk and are shown in Figure 9. Slightly higher values of *F1* are obtained when J5 is performed with set C5. Unlike in J1, a higher number of aircraft with different durations of PW and TS are present in J5. When the precedence relationship between these operations is changed, obtained results can be affected. Solutions found with C6 showed similar behavior for both shifts J4 and J5. Because Ca is less constrained than in the original set C4, the time windows for performing the operation are longer and fewer vehicles are required. In general, this leads to obtaining 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.

N. It.	C4J4		C4J5		C5J4		C5J5		C6J4		C6J5	
	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		17345.39		16902.98	

Table 4. Summary of results obtained for BCN instances. Values of objectives F1 and F2 found using precedence constraint rules in C with shifts J are given for each column. The non-dominated solutions are marked with an asterisk

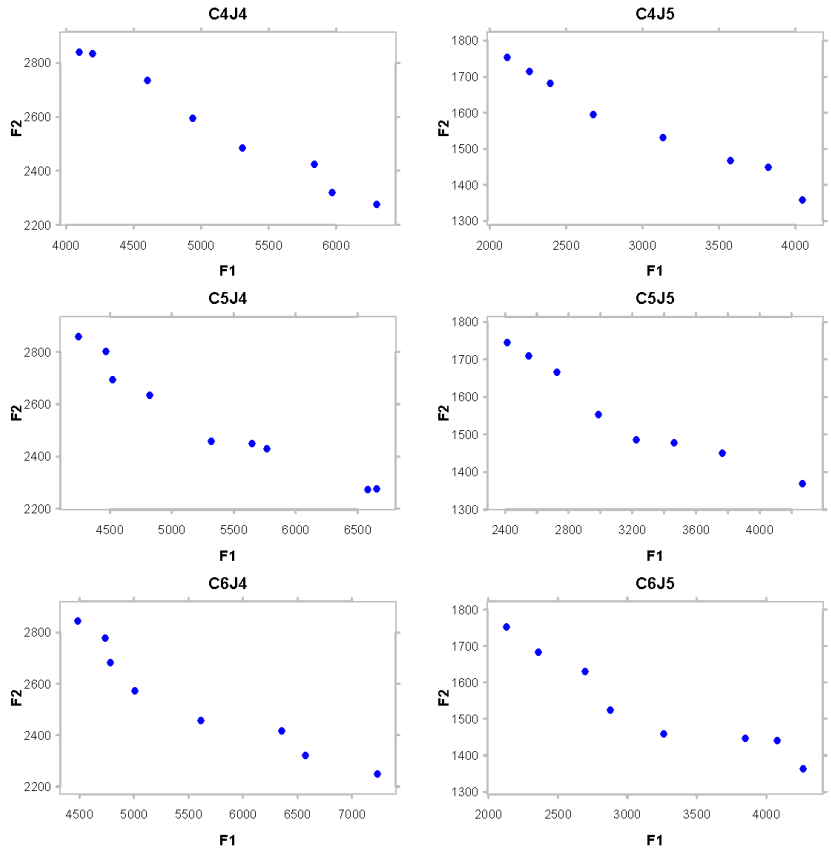


Figure 9. Pareto solutions for BCN instances

5.3 CP-based VND-LNS evaluation

Depending on the problem, CP can be expensive in computational time. However, it provides other advantages such as the flexibility to add new constraints without affecting the search process. According to [21], the CP-based approach proposed by Rousseau et al. [43] may be more effective in real-life applications in spite of its long running time, as are other flexible methods. In our case, each ground-handling operation or vehicle type has special characteristics depending on many factors, e.g., airline policy, airport size, etc. In addition, these particular features usually involve heterogeneous side constraints. Thus, flexibility and expressiveness for modeling constraints are crucial aspects to be considered. Because we aim to use this approach at a tactical level, we have more available time than would be available at an operational level. Therefore computational time becomes less of an issue.

To evaluate the efficiency of the adopted CP-based VND-LNS we have compared the methodology with some state-of-art algorithms for solving VRPTW problems. First, we have used the well-known benchmark set developed by Solomon [16]. Second, we have implemented the simulated annealing (SA) method presented in [59] for scheduling the handling vehicles in our bi-objective approach.

Solomon instances are divided into six classes: R1, R2, C1, C2, RC1 and RC2. Customers in C1 and C2 are grouped in clusters whereas they are randomly distributed in R1 and R2. Classes RC1 and RC2 have a mix of random and clustered distribution. Benchmarks are given in terms of distance and number of vehicles. In our problem, we have employed the methodology to minimize operation waiting time. The I3 heuristic was used to obtain an initial solution that improves this objective without compromising the number of vehicles required. Therefore, some adjustments of the algorithm should be done to use Solomon sets for comparison. First, we modified the optimization objective of the algorithm to traveling distance. In addition, we

used the I1 variant to make a fairer comparison. As mentioned, I1 yields the best distance results. Therefore, it is the more appropriate variant to minimize traveling distance. In this case, we used the parameter settings used by Solomon to obtain the best solutions. Regarding the parameters of the VND-LNS local search process, we used the same values in the case of the *SMART* operator. Values of *RPOP* are also the same, except that the maximum number of pivots is set to 7. Due to benchmark instances having wider time windows than do operations at the airport, more computational time is needed to improve the solutions. A CP-based algorithm is expected to behave better as the problem gets tighter. When domains are larger, the algorithm tries more values for each variable, growing the search tree size. We have increased the limit of the branch-and-bound method to 50 seconds, and the process is applied for 500 seconds. The results are also obtained by running the algorithm only one time for each instance.

Average results obtained with our approach (OS) for each class are outlined in Table 5. The number of vehicles and the total traveling distance (TD) are shown. We have compared our results with the best known solutions (BKS) [60] and with the SA method [59, 61]. In addition, we have included a hybrid approach [62] and a recent LNS algorithm [63]. The relative error (Gap) between OS and the other approaches has been calculated.

Prob	BKS		OS		Gap(%)	SA		Gap(%)	Berger et al.[60]		Gap(%)	Hong [61]		Gap(%)
	# Veh.	TD	# Veh.	TD	OS-BKS	# Veh.	TD	OS-SA	# Veh.	TD	OS-[61]	# Veh.	TD	OS-[62]
C1	10.00	828.38	10.00	847.58	2.32	10.00	828.38	2.32	10.00	828.50	2.30	10.00	833.10	1.74
C2	3.00	589.86	3.00	606.25	2.78	3.00	589.86	2.78	3.00	590.06	2.74	3.00	590.31	2.70
RC1	11.50	1384.16	12.88	1455.61	5.16	11.50	1384.38	5.15	11.88	1414.86	2.88	12.13	1369.57	6.28
RC2	3.25	1119.24	3.875	1295.75	15.77	3.25	1144.95	13.17	3.25	1258.15	2.99	3.75	1131.18	14.55
R1	11.92	1210.34	12.66	1300.50	7.45	12.25	1203.37	8.07	12.17	1251.40	3.92	12.25	1218.28	6.75
R2	2.73	951.03	3.63	1117.54	17.51	2.91	962.51	16.11	2.73	1056.90	5.74	3.27	964.11	15.91

Table 5. Average results for the Solomon benchmarks obtained with our local search approach (OS). Results from the Simulated Annealing (SA) method [59,61], Berger et al. [62], and Hong [63] are included for comparison

As observed, the results of classes C1 and C2 are better than those obtained for classes RC and R. Our solutions in class C are closer to the best known solutions having a gap between 2 and 3 percent. In the ground-handling problem, visits are generally clustered due to the distribution of stands around the terminal. Moreover, distances are shorter with respect to benchmark problems. In relation to problems R and RC, they are far from the best results. The reason may be found on the fact that the algorithm is stopped after 500 seconds, not providing enough time to reach a minimum.

The next step was to evaluate the CP-based methodology for solving the VRPTWs in the ground-handling problem. The SA method addresses the bi-objective nature of classical VRPTW, i.e. minimizing number of routes and distance travelled, in this order. It has found new best solutions for two Solomon problem instances [60]. For solving our problem, the cost function of SA was modified to obtain a compromise between minimizing number of vehicles (weighted by parameter a) and waiting time (weight b). We have adjusted a and b such that similar number of vehicles with respect to our methodology are obtained. After running several experiments we have set a to 30, 55 and 100 depending on to the instance and b =1. Moreover, the number of annealing steps and iterations has been tuned to improve the performance of SA. Good results were found with n^2 to $4n^2$ steps and between 20 and 90 iterations. Regarding remaining parameters, the same setting proposed by the author were employed.

We have tested SA on instances C1J1, C2J2, C3J3, C4J4, and C4J5 and we show in Figure 10 the non-dominated solutions obtained with both methods. Visually, Pareto frontiers are similar.

Solutions found with the same sequences are in general non-dominated, i.e. if one objective is improved with SA the other is worst with respect to CP-based VND-LNS. The most notable difference is the space covered by the Pareto set in instances C1J2 and C1J3. Also, there is a small variation in the number of non-dominated solutions found.

To make a more precise comparison we have selected the coverage metric used in [64]. Different performance metrics have been proposed to compare results in a multi-objective context. In particular, the coverage metric measures the number of solutions of one algorithm that dominates the solutions of another algorithm. The coverage, average fleet size and execution time associated to each instance are presented in Table 6.

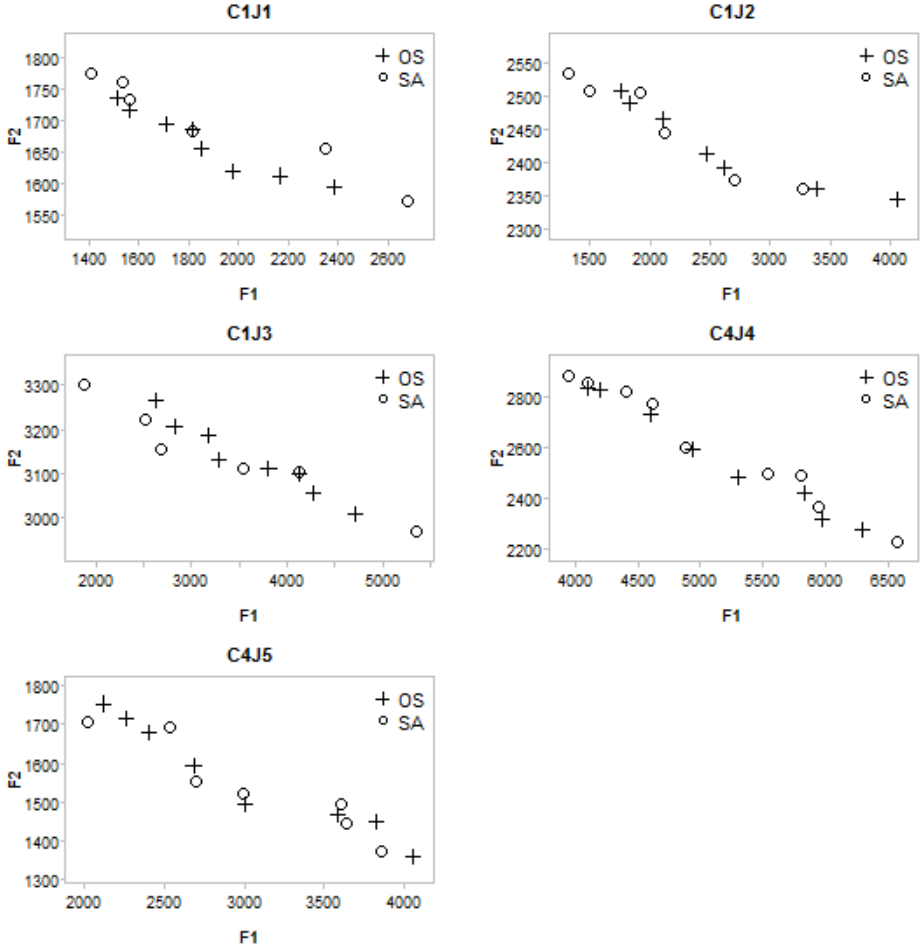


Figure 10. Pareto solutions obtained with our local search approach (OS) and the Simulated Annealing (SA) method. Crosses and bullets represent OS and SA solutions, respectively

The coverage C(A,B) value is an interval [0,1] in which C(A,B) = 1 means that all solutions of B are covered by A. In contrast, none of the solutions in B are dominated by A when C(A,B)=0. Both C(A,B) and C(B,A) need to be calculated since C(A,B) =1- C(B,A) is not necessarily true. As observed, either C(OS,SA) and C(SA,OS) values are less than 0.5 for all instances. That is, most solutions are non-dominated between them and algorithms are comparable. In the case of instances C1J2 and C4J5 there is a slight difference between both methods. SA improves the Pareto set in C1J3 while our approach is better in C4J4.

Problem	OS			SA		
	#Vehicles	Time (s)	C(OS,SA)	#Vehicles	Time (s)	C(SA,OS)
C1J1	9.41	16829.53	0.33	9.36	12580.97	0.13
C1J2	7.61	18023.17	0.16	7.42	15966.86	0.14
C1J3	6.54	16375.34	0.17	6.69	17460.36	0.38
C4J4	4.28	18438.25	0.33	4.22	18220.93	0.00
C4J5	4.9	17201.87	0.29	4.99	12836.44	0.37

Table 6. Coverage, average fleet size and execution time obtained by using OS and SA for instances C1J1, C1J2, C1J3, C4J4 and C4J5

Regarding the computational effort, SA provides a higher performance for instances C1J1, C1J2, and C4J5. However, annealing steps and iterations needed to be adjusted for each instance to obtain good results in a good time.

We consider that the CP methodology is an efficient method for scheduling ground-handling vehicles. It is a fact that SA yields better results for the Solomon instances. However, both methods are comparable when solving VRPTWs associated to the ground-handling problem. SA has a time advantage but it needs a harder parameter adjustment, which can be a major drawback for real problems. The CP-based VND-LNS requires more time but solution quality is comparable. In addition, our approach is more flexible to extend, a powerful feature to address the particularities of ground-handling activities.

5.4 Evaluation of SIM

In workcenter-based decomposition methods, which are the reference cases for *SIM*, sub-problems are solved in several sequences to improve the global solution. In our problem, the *SIM* method modifies these sequences to find Pareto solutions for the bi-objective optimization problem. Considering that solving each VRPTW is a complex problem, *SIM* was defined to find a minimum set of solutions that can provide a proper representation of the Pareto set. Making an exhaustive exploration and obtaining all possible solutions allows us to assess the performance of the heuristic. However, computing all possible combinations, or even a high number as in workcenter-based methods, is not viable in terms of computational effort.

The local search process is clearly the most time-consuming part of the algorithm, whereas a quick, reasonably good initial solution is found by I3. Thus, we have run *SIM* to solve each routing problem only with I3. Next, we compared *SIM* solutions with results found through an exhaustive exploration of sequences, also with I3. Thus, we can verify how close *SIM* solutions are to the Pareto frontier. Note that sequences produced by I3 are not necessarily the same as using the local search process. However, this provides an effective mechanism to test the performance of *SIM*.

SIM takes less than 1 second to generate a set of solutions. The maximum number of sequences in an exhaustive exploration is a permutation of all vehicle types. In this problem, we have 7 vehicle types; therefore, an acceptable amount of CPU time is required to produce all solutions. The algorithm spends approximately 8 minutes on each instance; therefore, this is a viable option in this case. We have selected instances C1J1, C2J2, C3J3, C4J4, and C4J5 to test. The results are depicted in Figure 11.

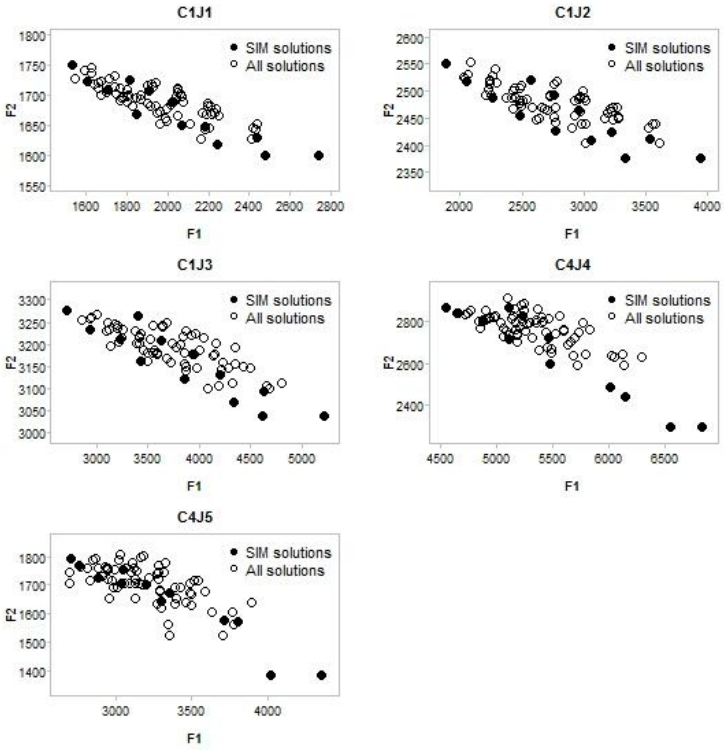


Figure 11. Solutions obtained by SIM and using an exhaustive exploration for instances C1J1, C1J2, C1J3, C4J4 and C4J5. Black and white bullets represent SIM and exhaustive exploration solutions, respectively

Notice that in less than 1 second, *SIM* is able to find Pareto solutions and provide a good coverage of the frontier for most instances. Regarding instance C4J5, the coverage reached is more discrete. This can be caused by the fact that there is a lower density of points on the Pareto set, and non-dominated solutions are far from the rest. Considering 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.

6. Conclusions and Future Research

In the present paper, we propose a first approach for scheduling ground-handling vehicles at an airport. Different operations and types of vehicles have been considered to tackle this problem from a holistic perspective. We have modeled ground-handling services as a bi-objective optimization problem, aiming to integrate the scheduling decisions about each resource and to contribute to the optimization of the overall process. This goal is defined through two objectives: (i) minimizing the operations waiting time and the total reduction of the time windows, and (ii) minimizing the total completion time of the turnarounds.

The problem has been decomposed to allow the model and the solution method to be simplified without losing the global approach of the proposal. First, time windows to complete operations are obtained according to precedence constraints and turnaround time. One VRPTW for each operation is identified and solved separately, and decisions made are propagated to the other VRPTWs by reducing their time windows. This decomposition schema ensures that local routing solutions can be integrated to obtain a consistent global solution. Time window calculations as well as the reduction process have been performed very efficiently through Constraint Programming's propagation mechanism.

A quick first solution for each routing problem is obtained using the Solomon *Insertion Heuristic I3* method. A CP-based VND-LNS methodology to solve the VRPTW is applied as a

local search process. This approach has been demonstrated to be effective at improving these initial solutions. A new method we called *Sequence Iterative Method* was developed to improve the global solution when dealing with the bi-objective problem.

The approach was tested using real-life data from the Palma de Mallorca and Barcelona airports and specifications from aircraft manufacturers. The results show that different solutions representing a trade-off between objectives were found, thus modifying the order in which vehicles are scheduled. Moreover, the number of vehicles needed to perform operations can change according to this order. This might be an important criterion to select between two solutions with similar values of the objective functions. Schedules in which longer operations use fewer vehicles could favor the robustness of the solution because they leave spare vehicles that could be used in case of unexpected events or delays. Prioritizing activities with expensive or fewer available vehicles might be another aspect contributing to the overall process optimization. In addition, allowing some waiting time on the less-constrained operations leads to saving on resources utilization without affecting the completion time of the turnaround.

Different aspects remain for further development of the presented work. We have considered the most important operations during a turnaround, but we did not consider baggage transportation or passenger transfer when aircraft are parked at a remote stand. Although we obtained interesting conclusions on how the scheduling decisions of a resource affect scheduling of other resources, the inclusion of these additional operations will further improve and enrich this study. Furthermore, we have assumed a homogeneous fleet of each type of vehicle. Usually, vehicles are compatible with a specific type of aircraft and cannot serve other types. Hence, considering a heterogeneous fleet and including this constraint in the model is another topic for future research. To this end, the flexibility of the adopted CP-based local search process allows introducing new constraints with minimum modifications in the methodology. Including manpower planning and rostering would also be an interesting topic for future development of our approach.

ACKNOWLEDGEMENTS:
NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

[1] EUROCONTROL. CODA Digest - Annual 2012. Delays to Air Transport in Europe; 2012.
[2] EUROCONTROL. Airport CDM Operational Concept Document; 2006.
[3] SESAR. Milestone Deliverable D1: Air Transport Framework: The Current Situation; 2008.
[4] TITAN. Operational Concept – Issue 1. Turnaround Integration in Trajectory and Network Project; 2010.
[5] TITAN. Analysis of current situation. Turnaround Integration in Trajectory and Network Project; 2010.
[6] Fricke H, Schultz M. Delay impacts onto turnaround performance. Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009); 2009.
[7] Norin A, Granberg TA, Värbrand P, Yuan D. Integrating optimization and simulation to gain more efficient airport logistics. Eighth USA/Europe Air Traffic Management Research and Development Seminar (ATM2009); 2009.
[8] ARC. Study on the Impact of Directive 96 / 67 / EC on Ground Handling Services 1996-2007 Final Report. Airport Research Center; 2009.
[9] Clausen T. Airport ground staff scheduling. PhD Thesis. Technical University of Denmark; 2011.
[10] Leeuwen P van. CAED D2: Modelling the Turnaround Process, CARE INO III: The Co-ordinated Airport through Extreme Decoupling. Eurocontrol; 2007.

[11] Schmidberger S, Bals L, Hartmann E, Jahns C. Ground handling services at European hub airports: Development of a performance measurement system for benchmarking. International Journal of Production Economics 2009; 117(1):104–116.
[12] Du Y, Zhang Q, Chen Q. ACO-IH: An improved ant colony optimization algorithm for Airport Ground Service Scheduling. IEEE International Conference on Industrial Technology; 2008, p 1-6.
[13] Ho SC, Leung JMY. Solving a manpower scheduling problem for airline catering using metaheuristics. European Journal of Operational Research 2010; 202(3): 903–921.
[14] Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. The VRP with time windows. In: Toth P, Vigo D, editors. The Vehicle Routing Problem, USA: SIAM; 2002, pp. 157–186.
[15] Sourirajan K, Uzsoy R. Hybrid decomposition heuristics for solving large-scale scheduling problems in semiconductor wafer fabrication. Journal of Scheduling 2007; 10(1): 41–65.
[16] Solomon, MM. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations research 1987; 35(2): 254–265.
[17] Guimarans D. Hybrid algorithms for solving routing problems. PhD Thesis. Autonomous University of Barcelona; 2012.
[18] Kallehauge B. Formulations and exact algorithms for the vehicle routing problem with time windows. Computers & Operations Research 2008; 35(7): 2307–2330.
[19] Lenstra JK, Kan AHGR. Complexity of vehicle routing problem with time windows 1981; Networks 11: 221–227.
[20] Bräysy O, Gendreau M. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transportation science 2005; 39(1): 104–118.
[21] Bräysy O, Gendreau M. Vehicle routing problem with time windows, part II: Metaheuristics. Transportation science 2005; 39(1): 119–139.
[22] Jourdan L, Basseur M, Talbi EG. Hybridizing exact methods and metaheuristics: A taxonomy. European Journal of Operational Research 2009; 199(3): 620–629.
[23] Gambardella LM, Taillard E, Agazzi G. MACS-VRPTW. A multiple ant colony system for vehicle routing problems with time windows. In: Corne D, Dorigo M, Glover F, editors. New Ideas in Optimization, London: McGraw-Hill; 1999, pp. 63–76.
[24] Tan KC, Chew YH, Lee LH. A Hybrid Multiobjective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows. Computational Optimization and Applications 2005; 34(1): 115–151.
[25] Collette Y, Siarry P. Multiobjective Optimization: Principles and Case Studies, Berlin: Springer; 2003.
[26] Ombuki B, Ross BJ, Hanshar F. Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows. Applied Intelligence 2006; 24(1): 17–30.
[27] Ghoseiri K, Ghannadpour SF. Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm. Applied Soft Computing 2010; 10: 1096–1107.
[28] Liu C, Chang TC, Huang LF. Multi-objective heuristics for the vehicle routing problem. International Journal of Operations Research 2006; 3(3): 173–181.
[29] Hong SC, Park YB. A heuristic for bi-objective vehicle routing with time window constraints. International Journal of Production Economics 1999; 62(3), 249– 258.
[30] Müller J. Approximative solutions to the bicriterion Vehicle Routing Problem with Time Windows. European Journal of Operational Research 2010; 202(1): 223–231.
[31] Jozefowiez N, Semet F, Talbi EG. Multi-objective vehicle routing problems. European Journal of Operational Research 2008; 189(2): 293–309.
[32] Miettinen K. Introduction to Multi-Objective Optimization: Noninteractive approaches. In: Branke J, Deb K, Miettinen Kaisa, Słowiński R, editors. Multiobjective Optimization Interactive and Evolutionary Approaches, Lecture Notes in Computer Science, vol. 5252, Berlin: Springer; 2008, pp. 1–25.
[33] Miettinen K, Ruiz F, Wierzbicki AP. Introduction to Multi-Objective Optimization: Interactive approaches. In: Branke J, Deb K, Miettinen K, Słowiński R, editors. Multiobjective Optimization Interactive and Evolutionary Approaches, Lecture Notes in Computer Science, vol. 5252, Berlin: Springer; 2008, pp. 27–57.

[34] Fonseca CM, Fleming PJ. An Overview of Evolutionary Algorithms in Multiobjective Optimization. Evolutionary Computation 1995; 3(1): 1–16.
[35] Rossi F, Beek P van, Walsh T. Handbook of Constraint Programming, Elsevier Inc; 2006.
[36] Focacci F, Laburthe F, Lodi A. Local search and constraint programming. In: Glover F, Kochenberger G.A, editors. Handbook of Metaheuristics, Boston: Kluwer Academic; 2003, pp. 369–403.
[37] Bessiere C. Constraint Propagation. In: Rossi F, van Beek P, Walsh T, editors. Handbook of Constraint Programming, Elsevier Science Inc; 2006, pp. 29–69.
[38] Kilby P, Shaw P. Vehicle routing. In: Rossi F, van Beek P, Walsh T, editors. Handbook of Constraint Programming, Elsevier Science Inc; 2006, pp. 801–836.
[39] Adams J, Balas E, Zawack D. The Shifting Bottleneck Procedure for Job Shop Scheduling. Management Science 1988; 3: 391–401.
[40] Balas E, Lenstra JK, Vazacopoulos A. The one machine problem with delayed precedence constraints and its use in job shop scheduling. Management Science 1995; 41: 94–109.
[41] Lafayette W. A Computational Study of Shifting Bottleneck Procedures for Shop Scheduling Problems. Journal of Heuristics 1997; 3(2): 111–137.
[42] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems. 4th International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science 1520, Springer-Verlag; 1998, pp 417–431.
[43] Rousseau LM, Gendreau M, Pesant G. Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of Heuristics 2002; 8(1): 43–58.
[44] Mladenovic N, Hansen P. Variable neighborhood search. Computers & Operations Research 1997; 24(11): 1097–1100.
[45] Bräysy O. A Reactive Variable Neighborhood Search for the Vehicle Routing Problem with Time Windows. INFORMS Journal on Computing 2003; 15(4), pp. 347–368.
[46] Guimarans D, Herrero R, Ramos JJ, Padrón S. Solving vehicle routing problems using constraint programming and lagrangian relaxation in a metaheuristics framework. International Journal of Information Systems and Supply Chain Management 2011; 4(2), 61–81.
[47] Hansen P, Mladenovic N. A tutorial on variable neighborhood search. Tech. Rep. No. G-2003-46, Montreal: Les Cahiers du GERAD, HEC Montreal and GERAD; 2003.
[48] Harvey W D and Ginsberg M L. Limited discrepancy search. International Joint Conference on Artificial intelligence IJCAI;1995.
[49] Gavanelli M. An algorithm for multi-criteria optimization in CSPs. International Conference on Integration of Artificial Intelligence and Operations Research techniques in Constraint Programming (CPAIOR'02); 2002, pp. 2–6.
[50] Jozefowiez N, Glover F, Laguna M. Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits. Journal of Mathematical Modelling and Algorithms 2008; 7(2), 177–195.
[51] Apt K, Wallace MG. Constraint Logic Programming using ECLiPSe, Cambridge: Cambridge University Press; 2007.
[52] Airbus320. A320 Airplane Characteristics for Airport Planning. Airbus, S.A.S; 2011.
[53] Boeing. B737-Airplane Characteristics for Airport Planning. The Boeing Company; 2009.
[54] Airbus319. A319 Airplane Characteristics for Airport Planning. Airbus, S.A.S; 2011.
[55] Airbus321. A320 Airplane Characteristics for Airport Planning. Airbus, S.A.S; 2011.
[56] Airbus330. A330 Airplane Characteristics for Airport Planning. Airbus, S.A.S; 2011.
[57] Boeing767. B767-Airplane Characteristics for Airport Planning. The Boeing Company; 2009
[58] Boeing777. B777 -Airplane Characteristics for Airport Planning. The Boeing Company; 2009
[59] Woch M, Łebkowski P. Sequential Simulated Annealing for the Vehicle Routing Problem with Time Windows. Decision Making in Manufacturing and Services 2009; 1-2(3), 87-100.
[60] Solomon Benchmarks. http://www.sintef.no/Projectweb/TOP/VRPTW/Solomon-benchmark/100-customers/. Last update: 20/08/2013.

[61] Woch M. Rozwiazanie problemu dostaw z oknami czasowymi za pomoca symulowanego wyzarzania (Solving Vehicle Routing Problem with Time Windows Using Simulated Annealing). Studia Informatica 2004; 25(2), 67–81.
[62] Berger J, Barkaoui M, Bräysy O. A route-directed hybrid genetic approach for the vehicle routing problem with time windows. Information Systems of Operation Research 2003; 41 179–194.
[63] Hong L. An improved LNS algorithm for real-time vehicle routing problem with time windows. Computers & Operations Research 2012; 39(2) 151-163.
[64] Garcia-Najera A, Bullinaria, J.A. An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. Computers & Operations Research 2011; 38(1) 287–300.

Figure_1

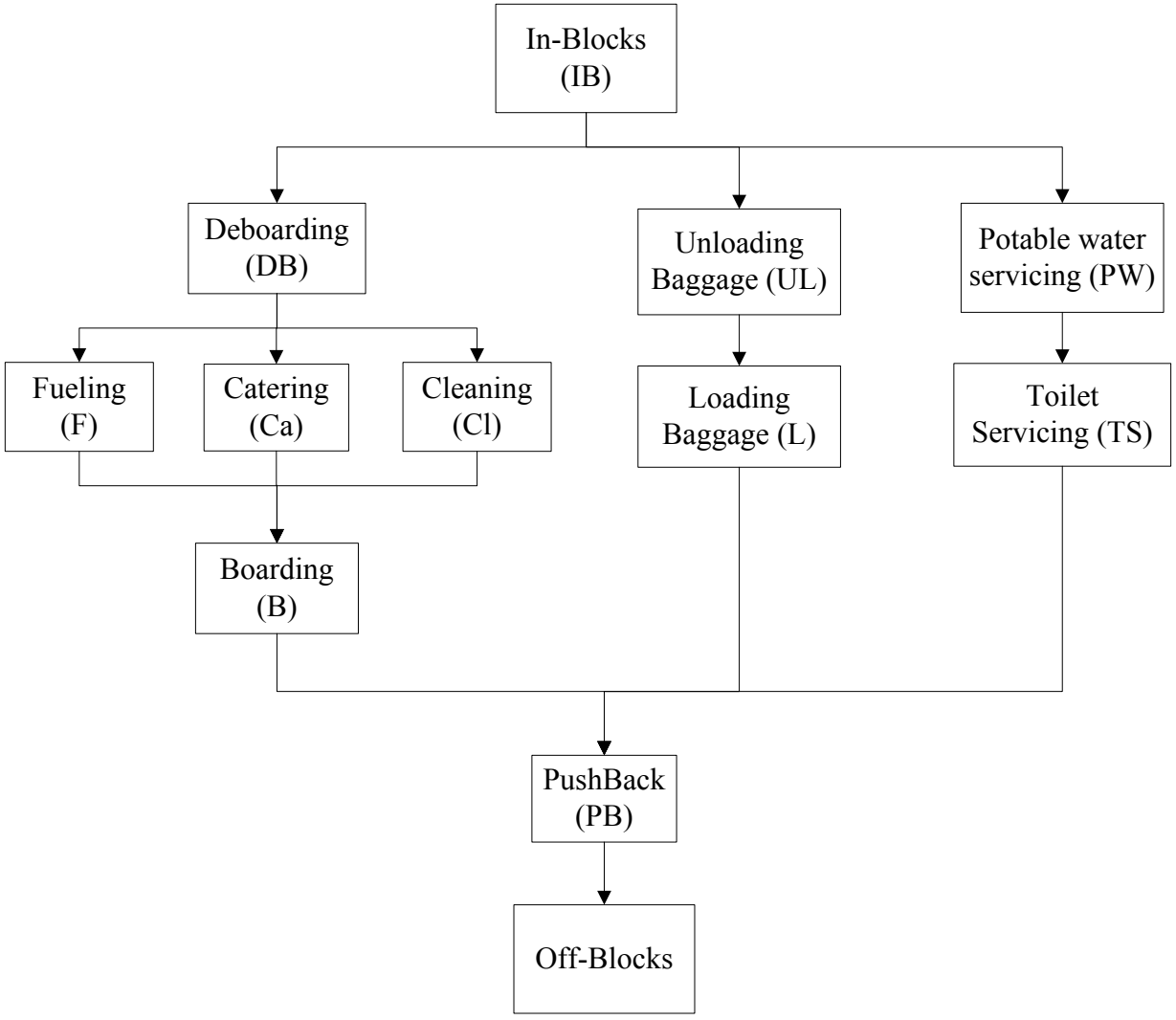

In-Blocks
(IB)
Deboarding
(DB)
Unloading
Baggage (UL)
Potable water
servicing (PW)
Fueling
(F)
Catering
(Ca)
Cleaning
(Cl)
Loading
Baggage (L)
Toilet
Servicing (TS)
Boarding
(B)
PushBack
(PB)
Off-Blocks

Figure_2

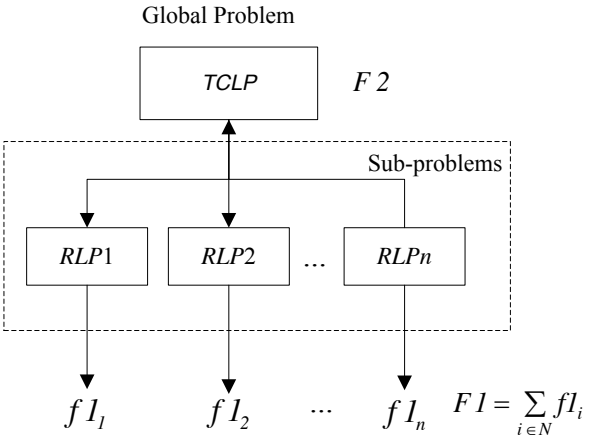

Global Problem
TCLP
F 2
Sub-problems
RLP1
RLP2
...
RLPn
$f1_1$
$f1_2$
...
$f1_n$
$F1 = \sum_{i \in N} f1_i$

Figure_3

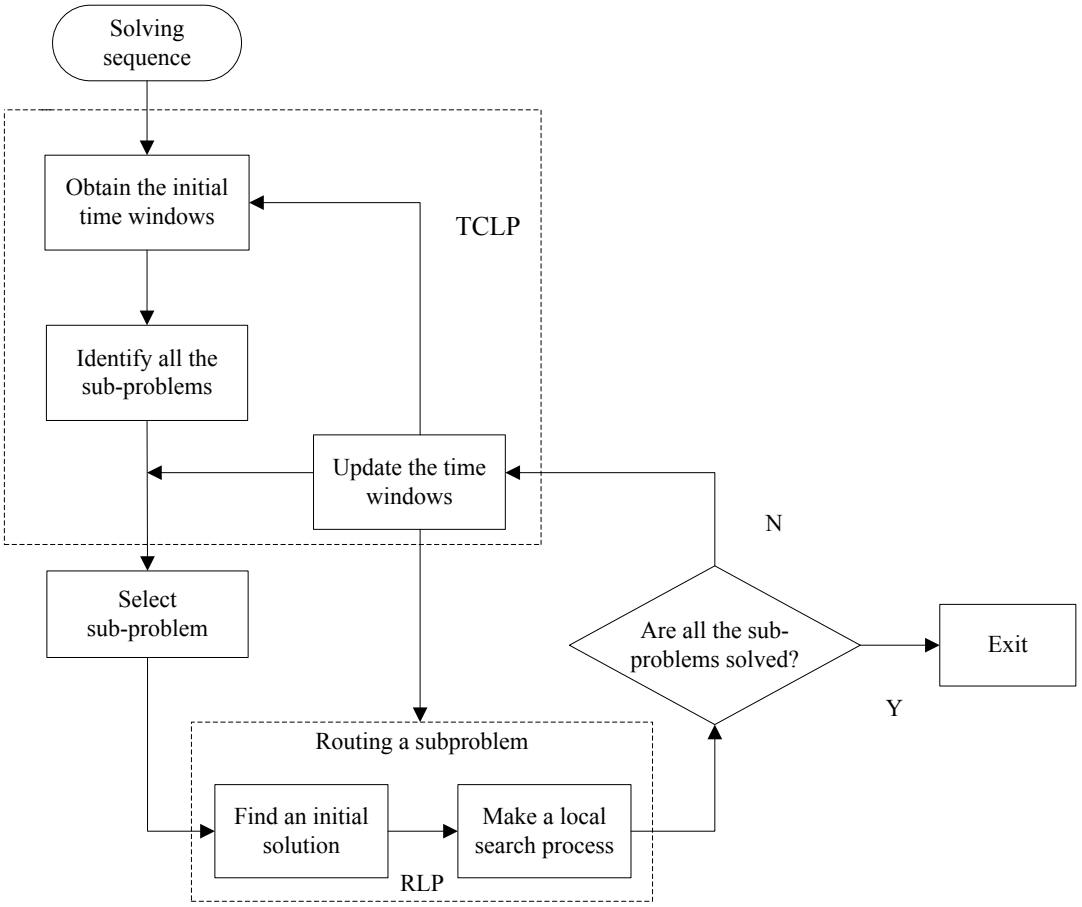

Solving sequence
Obtain the initial time windows
TCLP
Identify all the sub-problems
Update the time windows
Select sub-problem
Are all the sub-problems solved?
N
Y
Exit
Routing a subproblem
Find an initial solution
Make a local search process
RLP

Figure_4

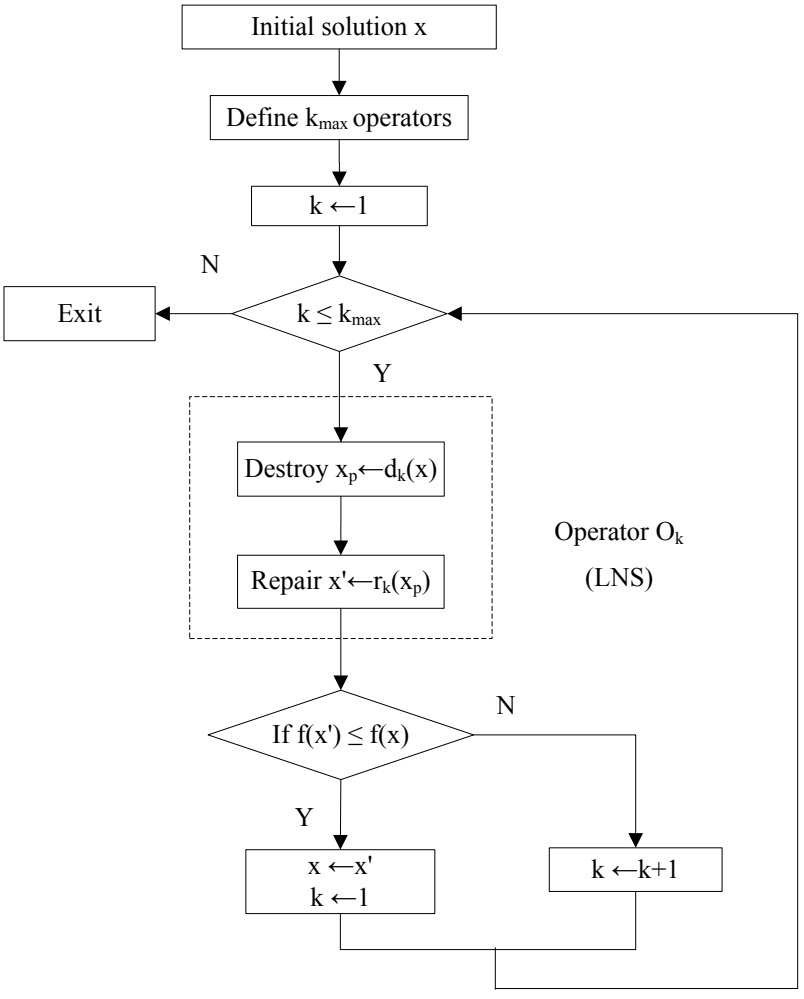
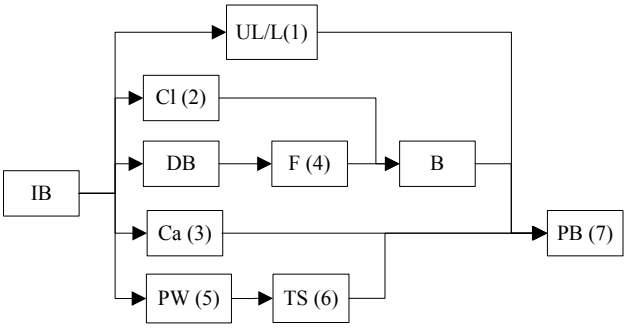
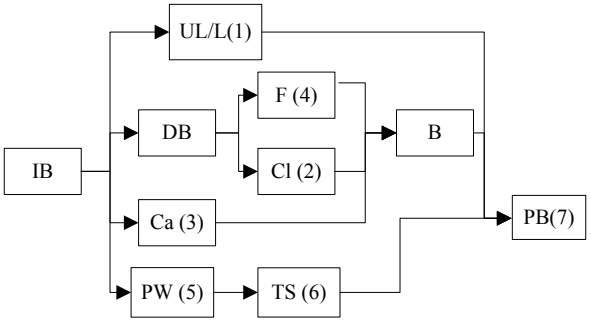

Initial solution x
Define k_{max} operators
k ←1
N
Exit
$k \le k_{max}$
Y
Destroy $x_p \leftarrow d_k(x)$
Repair $x' \leftarrow r_k(x_p)$
Operator O_k
(LNS)
N
If $f(x') \le f(x)$
Y
x ←x'
k ←1
k ←k+1

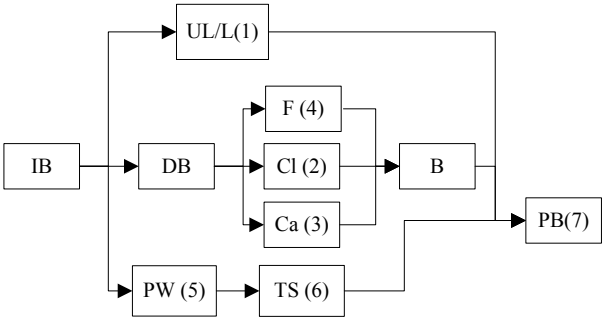
Figure_5


UL/L(1)
Cl (2)
DB
F (4)
B
IB
Ca (3)
PB (7)
PW (5)
TS (6)

a)


UL/L(1)
F (4)
DB
B
IB
Cl (2)
Ca (3)
PB(7)
PW (5)
TS (6)

b)


UL/L(1)
F (4)
IB
DB
Cl (2)
B
Ca (3)
PB(7)
PW (5)
TS (6)

c)

Figure_6

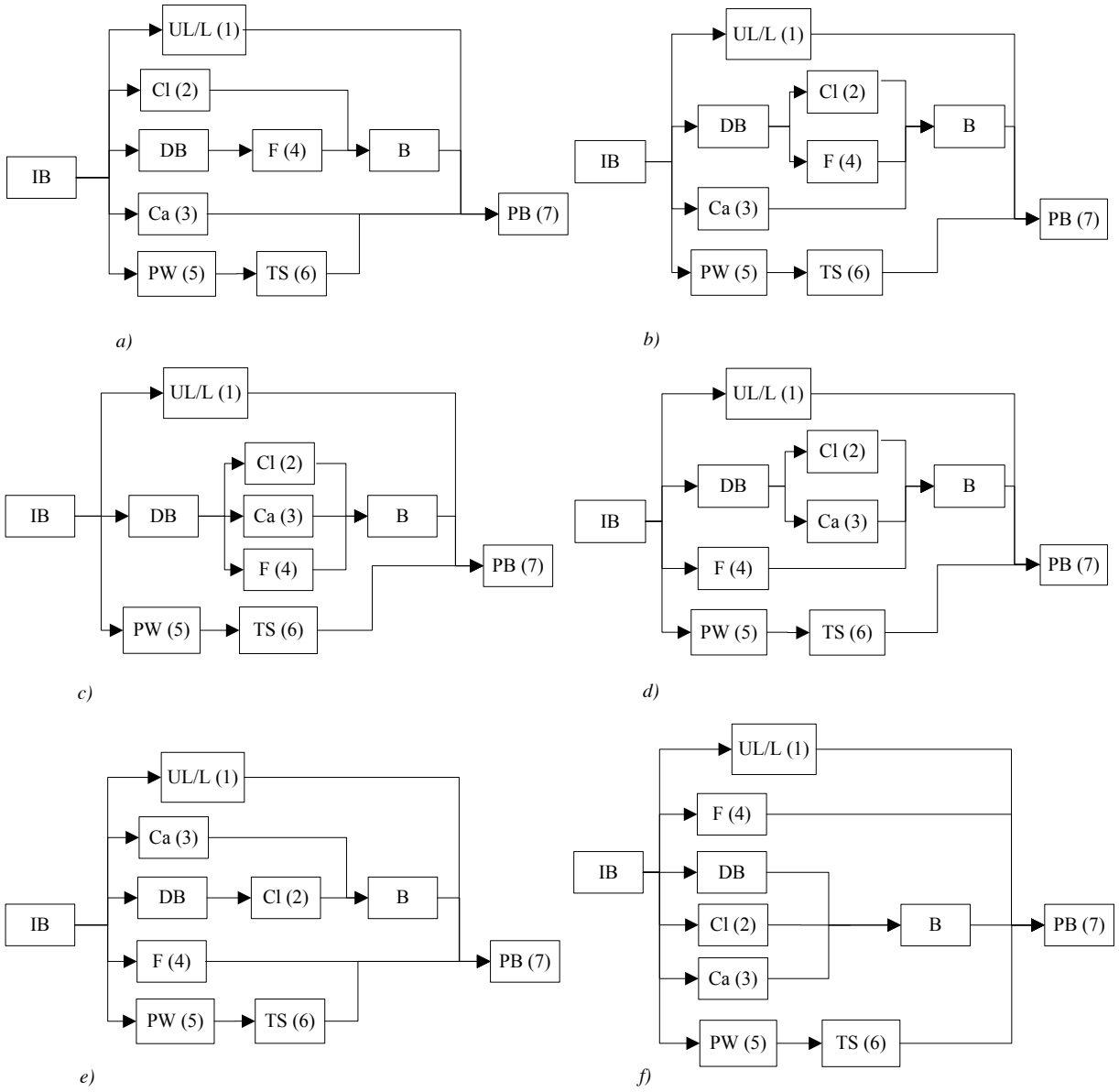


Figure_7

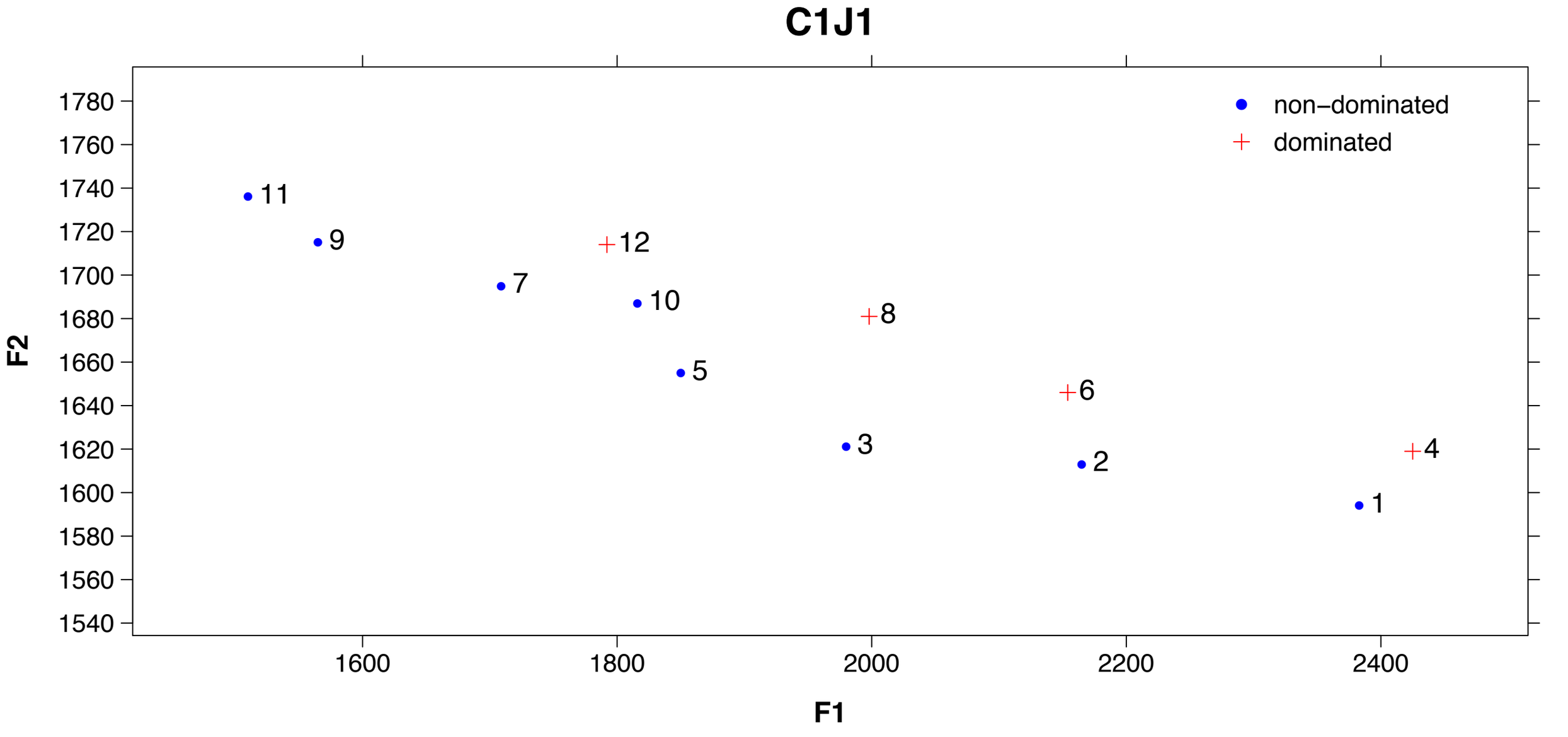

C1J1
non−dominated
dominated
F1
F2
1600
1800
2000
2200
2400
1540
1560
1580
1600
1620
1640
1660
1680
1700
1720
1740
1760
1780
11
9
12
7
10
8
5
6
3
4
2
1

Figure_8

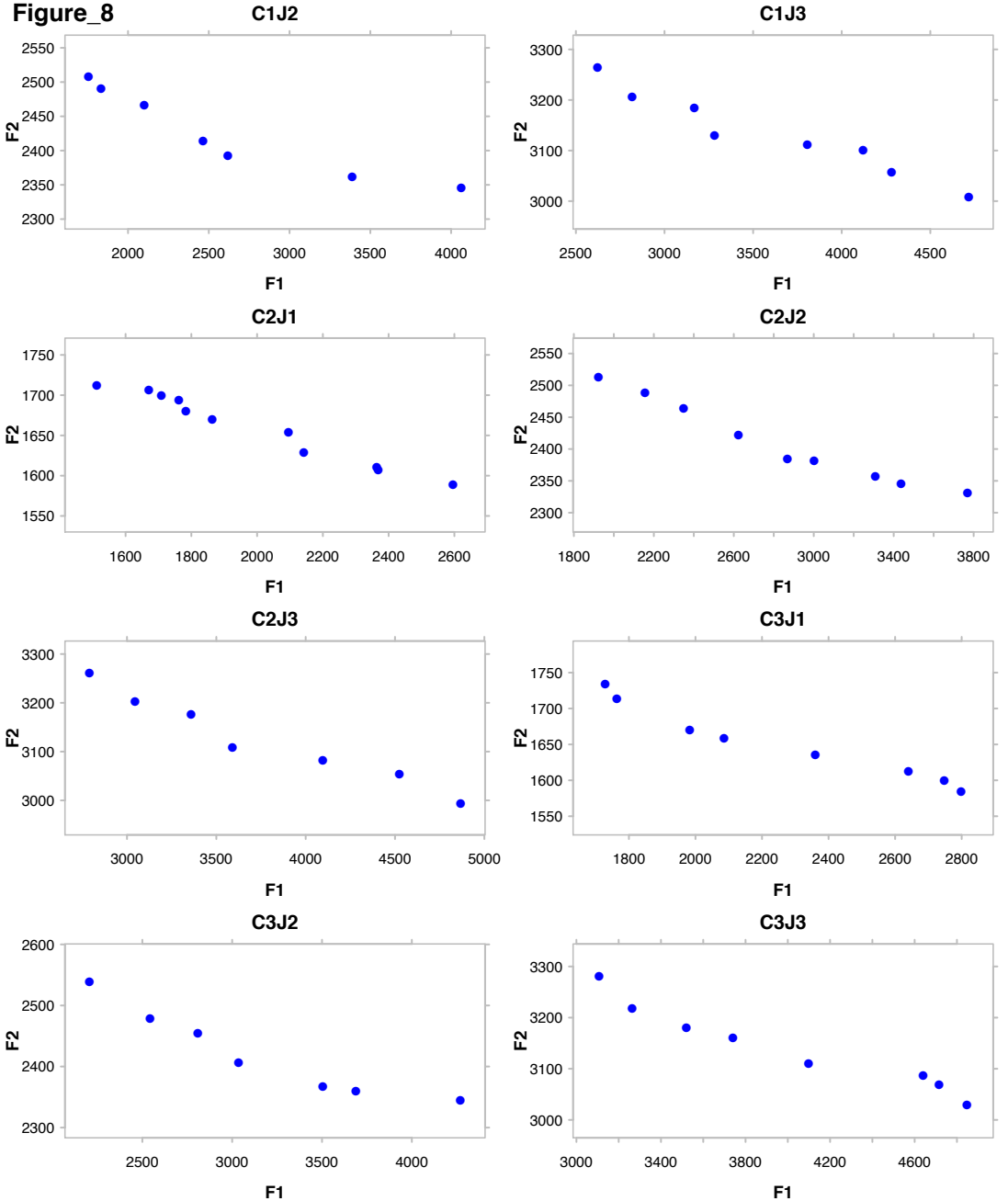
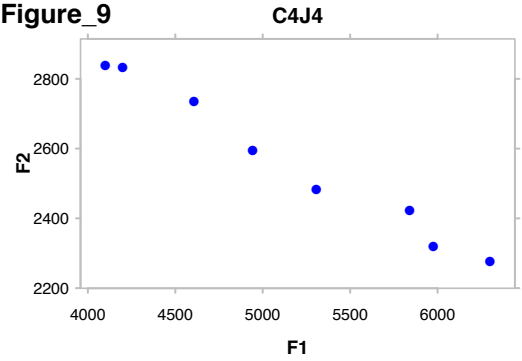
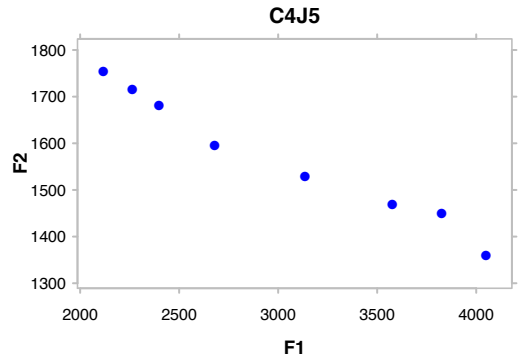
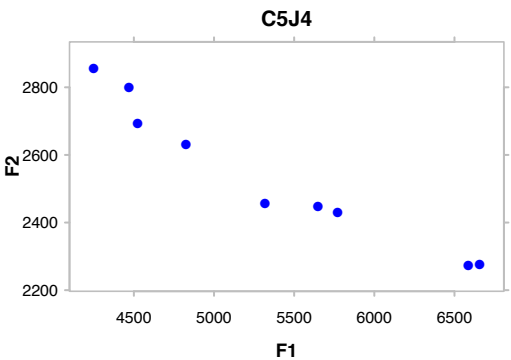
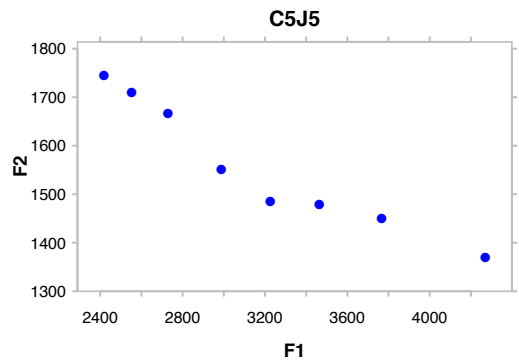

C1J2
F2
2550
2500
2450
2400
2350
2300
2000
2500
3000
3500
4000
F1
C1J3
F2
3300
3200
3100
3000
2500
3000
3500
4000
4500
F1
C2J1
F2
1750
1700
1650
1600
1550
1600
1800
2000
2200
2400
2600
F1
C2J2
F2
2550
2500
2450
2400
2350
2300
1800
2200
2600
3000
3400
3800
F1
C2J3
F2
3300
3200
3100
3000
3000
3500
4000
4500
5000
F1
C3J1
F2
1750
1700
1650
1600
1550
1800
2000
2200
2400
2600
2800
F1
C3J2
F2
2600
2500
2400
2300
2500
3000
3500
4000
F1
C3J3
F2
3300
3200
3100
3000
3000
3400
3800
4200
4600
F1

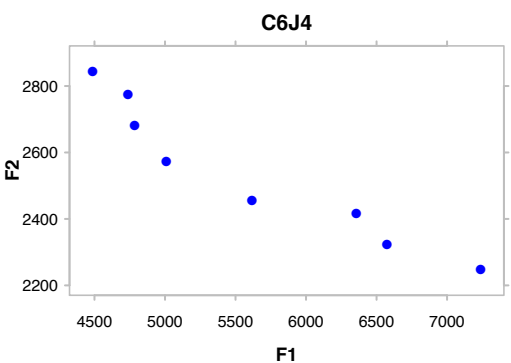
Figure_9


C4J4
F2
2800
2600
2400
2200
4000
4500
5000
5500
6000
F1


C4J5
F2
1800
1700
1600
1500
1400
1300
2000
2500
3000
3500
4000
F1


C5J4
F2
2800
2600
2400
2200
4500
5000
5500
6000
6500
F1


C5J5
F2
1800
1700
1600
1500
1400
1300
2400
2800
3200
3600
4000
F1


C6J4
F2
2800
2600
2400
2200
4500
5000
5500
6000
6500
7000
F1

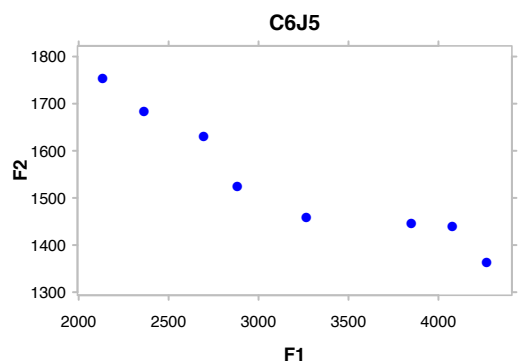

C6J5
F2
1800
1700
1600
1500
1400
1300
2000
2500
3000
3500
4000
F1

Figure_10
Click here to download high resolution image

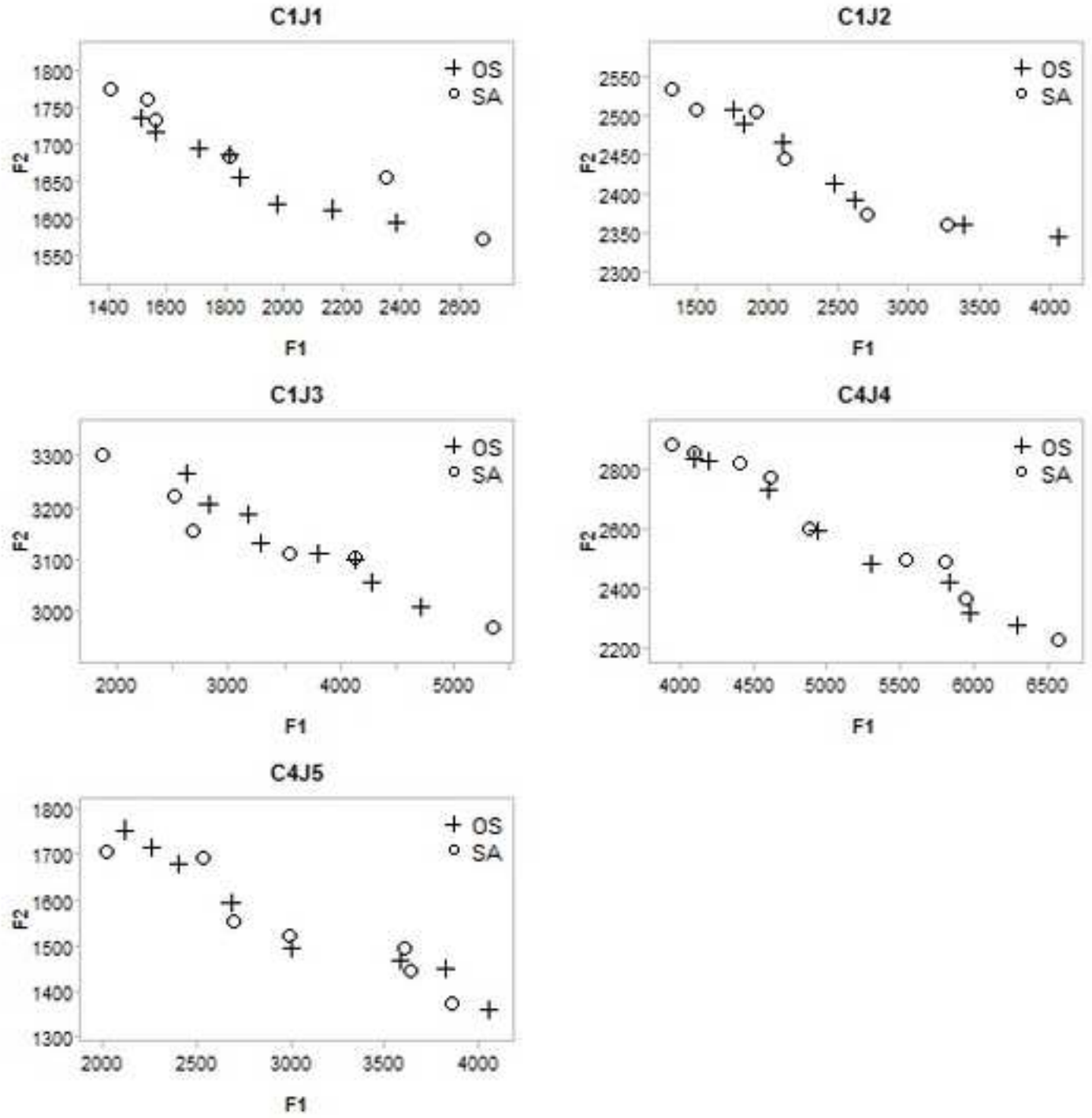


Figure_11
Click here to download high resolution image

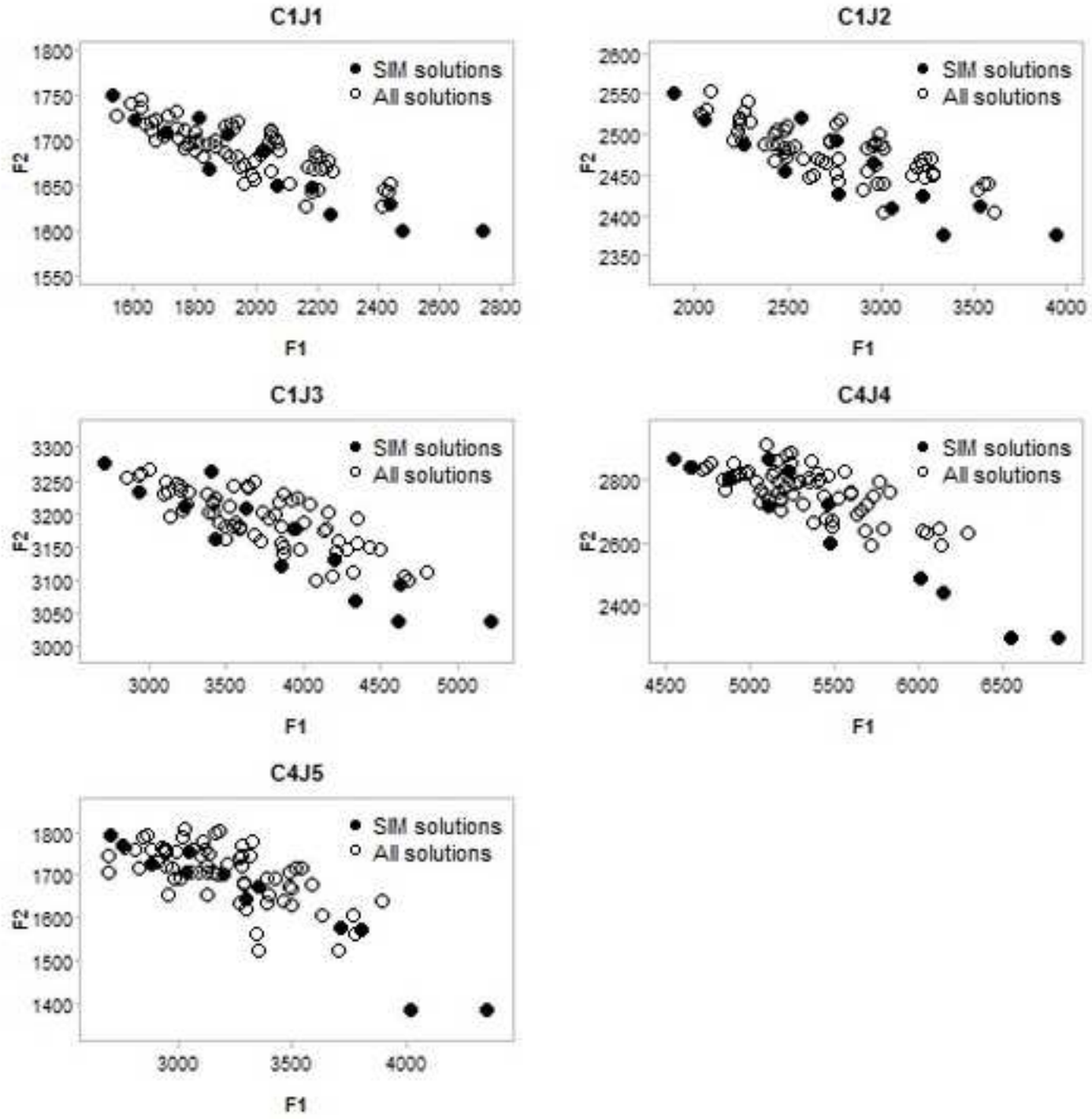


Table_1

N. It.	F1	F2	Sequence	# Vehicles							Time(s)
				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	1426.5
2	2165	1613	(6,7,1,2,3,4,5)	18	12	12	9	6	7	4	1263.07
3	1980	1621	(5,6,7,1,2,3,4)	18	11	11	9	4	7	4	1473.71
4	2425	1619	(6,5,7,4,3,2,1)	18	11	12	9	6	7	4	1545.45
5	1850	1655	(4,5,6,7,1,2,3)	18	11	11	9	4	7	4	1257.29
6	2154	1646	(6,5,4,7,3,2,1)	18	11	12	9	6	7	4	1308.99
7	1709	1695	(3,4,5,6,7,1,2)	18	11	11	9	4	7	4	1357.83
8	1998	1681	(6,5,3,4,7,2,1)	18	11	11	9	6	7	4	1471.98
9	1565	1715	(2,3,4,5,6,7,1)	18	10	11	9	4	7	4	1348.54
10	1816	1687	(6,5,3,4,2,7,1)	17	10	11	9	6	7	4	1285.05
11	1510	1736	(1,2,3,4,5,6,7)	16	10	11	9	4	7	5	1348.22
12	1792	1714	(6,5,3,4,2,1,7)	16	10	11	9	6	7	5	1360.61

Table_2

N. It.	C1J1		C1J2		C1J3		C2J1		C2J2		C2J3		C3J1		C3J2		C3J3	
	F1	F2	F1	F2	F1	F2	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*	2798*	1584*	3688*	2360*	4848*	3029*
2	2165*	1613*	4064*	2346*	4281*	3057*	2369*	1607*	3309*	2357*	4524*	3053*	2748*	1600*	4271*	2345*	4715*	3068*
3	1980*	1621*	3509	2403	4608	3067	2142*	1629*	3002*	2381*	4096*	3083*	2360*	1635*	3504*	2368*	4098*	3110*
4	2425	1619	2619*	2393*	3806*	3112*	2364*	1610*	3438*	2346*	4394	3104	2640*	1613*	3036*	2407*	4640*	3087*
5	1850*	1655*	2464*	2414*	3282*	3130*	1864*	1670*	2623*	2422*	3590*	3109*	2086*	1659*	3832	2403	3741*	3161*
6	2154	1646	3162	2399	4121*	3100*	2096*	1654*	2869*	2385*	3940	3153	2476	1639	2809*	2455*	4521	3133
7	1709*	1695*	2101*	2466*	3169*	3185*	1762*	1694*	2349*	2464*	3360*	3177*	1983*	1670*	3548	2447	3521*	3181*
8	1998	1681	2904	2464	3896	3173	1784*	1680*	2756	2444	3046*	3203*	2275	1662	2543*	2478*	4271	3186
9	1565*	1715*	1833*	2490*	2818*	3207*	1671*	1707*	2157*	2488*	3428	3215	1764*	1714*	2360	2517	3264*	3218*
10	1816*	1687*	2754	2486	3547	3197	1709*	1699*	2586	2486	2790*	3262*	2170	1692	3067	2514	3964	3200
11	1510*	1736*	1756*	2508*	2622*	3264*	1513*	1712*	1924*	2513*	3279	3286	1729*	1734*	2206*	2539*	3108*	3280*
12	1792	1714	2499	2509	3301	3252	1676	1710	-	-	-	-	2054	1711	2863	2522	3690	3265
T.T.(s)	16829.53		18023.17		16375.34		18112.82		18792.33		17971.83		16254.98		18798.34		17.605	

Table_3

N. It.	F1	F2	Sequence	# Vehicles							Time(s)
				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	1188.92
2	3577	1468	[5,7,1,2,3,4,6]	10	6	6	5	2	3	2	1333.95
3	3825	1449	[6,5,7,1,2,3,4]	10	6	6	5	4	3	2	1311.39
4	2999	1494	[5,6,7,2,4,1,3]	10	6	6	6	2	3	2	1331.87
5	2680	1595	[4,5,6,7,2,3,1]	9	6	6	5	2	3	2	1202.70
6	3252	1558	[6,5,4,7,2,3,1]	9	6	6	5	4	3	2	1339.12
7	2398	1681	[3,4,5,6,7,1,2]	8	6	6	5	2	3	2	1132.17
8	2928	1649	[6,5,3,4,7,1,2]	9	6	6	5	4	3	2	1290.72
9	2263	1715	[2,3,4,5,6,7,1]	8	6	6	5	2	3	2	1311.30
10	2846	1698	[6,3,2,5,4,7,1]	8	6	6	5	4	3	2	1116.89
11	2118	1754	[1,2,3,4,5,6,7]	8	6	6	5	2	3	2	1189.39

Table_4

N. It.	C4J4		C4J5		C5J4		C5J5		C6J4		C6J5	
	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		17345.39		16902.98	

Table_5

Prob	BKS		OS		Gap(%)	SA		Gap(%)	Berger et al.[60]		Gap(%)	Hong [61]		Gap(%)
	# Veh.	TD	# Veh.	TD	OS-BKS	# Veh.	TD	OS-SA	# Veh.	TD	OS-[61]	# Veh.	TD	OS-[62]
C1	10.00	828.38	10.00	847.58	2.32	10.00	828.38	2.32	10.00	828.50	2.30	10.00	833.10	1.74
C2	3.00	589.86	3.00	606.25	2.78	3.00	589.86	2.78	3.00	590.06	2.74	3.00	590.31	2.70
RC1	11.50	1384.16	12.88	1455.61	5.16	11.50	1384.38	5.15	11.88	1414.86	2.88	12.13	1369.57	6.28
RC2	3.25	1119.24	3.88	1295.75	15.77	3.25	1144.95	13.17	3.25	1258.15	2.99	3.75	1131.18	14.55
R1	11.92	1210.34	12.66	1300.50	7.45	12.25	1203.37	8.07	12.2	1251.40	3.92	12.25	1218.28	6.75
R2	2.73	951.03	3.63	1117.54	17.51	2.91	962.51	16.11	2.73	1056.90	5.74	3.27	964.11	15.91

Table_6

Problem	OS			SA		
	#Vehicles	Time (s)	C(OS,SA)	#Vehicles	Time (s)	C(SA,OS)
C1J1	9.41	16829.53	0.33	9.36	12580.97	0.13
C1J2	7.61	18023.17	0.16	7.42	15966.86	0.14
C1J3	6.54	16375.34	0.17	6.69	17460.36	0.38
C4J4	4.28	18438.25	0.33	4.22	18220.93	0.00
C4J5	4.9	17201.87	0.29	4.99	12836.44	0.37