# From packing to dispatching: supply chain optimization techniques

Respen, Jean

UNIVERSITÉ DE GENÈVE
GENEVA SCHOOL OF ECONOMICS AND MANAGEMENT

# From packing to dispatching: supply chain optimization techniques

Thèse
présentée à la GSEM - Geneva School of Economics and Management
de l'Université de Genève

par
Jean Respen
Ing. Sys. Comm. Dipl. EPF.

pour l'obtention du grade de
Docteur ès Geneva School of Economics and Management
*mention gestion d'entreprise*

Thèse N° 10
Genève, le 8 juin 2015

# Acknowledgments

First of all, I would like to thank my thesis director, Prof. Nicolas Zufferey for the opportunity he offered me to be his PhD student for 4 years and a half. His supportive guidance throughout my thesis was always present and relevant. His trust, availability, and kindness were such that today I consider him as a friend.

I would also like to thank the members of my thesis committee, Prof. Bernard Gendron, Dr. Daniel Costa, and Prof. Bernard Morard, for their constructive questions and remarks on this thesis.

Thanks to all the members of the Geneva School of Economics and Management, the assistants, and especially my colleague Simon Thevenin. I wish you all good luck for your forthcoming careers.

And of course I want to thank my friends, Sophie and Vincent, Soraya and Florian, Audrey, Pauline, and Polo, Mathilde and Julien, Annick and Alfio. We shared an uncountable number of meals together, of lovely moments, of holidays, and it made my PhD more pleasant.

My family, Lucien, Fanny, Yann, Catherine, Véronique, and Jean-Luc have to be thanked for their support during my thesis. As a supportive family, you were always there for me when I needed it. I spent five amazing years, and I think you earn some credits for it. Finally, I want to express a special thank to my parents Françoise and Marc for supporting my studies and always pushing me forward. I still need some times to realize how lucky I am to have been able to study as much as I did. And I know that it was only possible because you were there. So I surely want to thank you for everything you did for me.

Finally, a big thanks to my soon-to-be wife Tiffany. You were there from the beginning of my PhD “trip”. We had quite our set of events (some were amazing, some were very sad), but we were always tied together. I will never thank you enough for that, and I would like to dedicate this thesis to you.

# Abstract

This thesis aims at proposing relevant optimization techniques, such as metaheuristics, for various logistics problems faced by international companies. Four different projects are studied, each one having its own specificity. They all appear at different levels of the supply chain. The first project focuses on production – it is a scheduling problem with smoothing issues – and was performed in collaboration with *Politecnico di Milano* (Italy). The second project aims at studying a problem proposed by *Renault France*, on packing items in trucks. Almost 600 different loadings were tackled. Then, a project focusing on an online vehicle routing problem – where dynamic travel times and dynamic perturbations (such as traffic jam) are encountered – is jointly proposed with the *CIRRELT* in Montreal (Canada). Finally, a problem faced by a famous Swiss watch brand is exposed and tackled. It consists of an inventory dispatching problem, with various perturbations on the supply chain. For each problem, models and solution techniques are proposed, implemented and compared. Relevant ingredients are highlighted and important parameters are discussed. Each project has its own conclusions, where future works are also discussed. The four projects were the topics of four academic papers submitted to international journals.

# Résumé

Cette thèse a pour vocation de proposer des techniques d'optimisation performantes pour des problèmes logistiques complexes auxquels doivent faire face les entreprises internationales. Quatre projets différents sont étudiés et chacun d'entre eux détient ses propres spécificités, qui apparaissent à différents moments de la chaîne d'approvisionnement. Le premier projet traite de la production – un problème d'ordonnancement avec des contraintes de lissage – et a été réalisé en collaboration avec l'*École Polytechnique de Milan* (Italie). Le second projet a été proposé par *Renault France* et concerne le remplissage de camions. Presque 600 chargements ont été étudiés dans le but d'être optimisés, tout en tenant compte de diverses contraintes métier. Le troisième projet a été réalisé en collaboration avec le *CIRRELT* à Montréal (Canada) et concerne le routage de véhicules. Ici, des temps de trajets dynamiques ainsi que des perturbations dynamiques (telles que des congestions de trafic) sont pris en compte. Le quatrième et dernier projet a été formulé par une entreprise suisse active dans la haute horlogerie de luxe. Plusieurs perturbations apparaissent sur la chaîne d'approvisionnement et le but est de définir la meilleure façon d'envoyer les montres depuis l'usine jusque vers les magasins représentants la marque. Pour chaque problème, des modèles et des techniques de résolution sont proposés afin d'apporter des solutions concrètes dans l'aide au processus de décision. Les paramètres importants sont à chaque fois discutés et mis en valeur dans une optique globale de résolution du problème. Chacun des quatre papiers a été soumis à une revue académique internationale.

# Contents

# Positioning of the thesis

Operations research is constantly used to help managers make better decisions and manage risk. It is a science as itself. Using tools as mathematical modeling, statistical analysis and state-of-the-art optimization methods, operations research is used to produce optimal or near-optimal solutions for problems too big to be solved by human souls. This field of mathematics was born during World War II, to help war planners optimize war strategies. It was then widely used in business, finance and governments. Recently, such techniques have been applied to network problems (e.g., cell towers positioning), vehicle routing (e.g., transportation of goods, customers visit), logistics (e.g., supply chain optimization), and finance (e.g., portfolio optimization). The types of problems faced by operations research are diverse but have a common point of interest: they are very hard to solve, often due to the size and the structure of the problem at hand. After a correct modeling of the problem, optimization techniques are used to produce solutions which meet the quality requirements edited by the concerned firm. Important international companies have now understood the potential of such a science and are investing an important amount of money to get new decision tools, available thanks to the recent advances in the computer area. Moreover, large companies are now hiring intensively in operations research, where R&D teams can grow up to several hundreds of people. For example, *Renault France*, *EDF* (the French national electricity company) or *DHL*, have very important R&D teams working in this field, and have partnerships with academicians around the globe to learn recent advances and techniques.

Operations research is very active in logistics and operations management. Figure 1 presents a typical supply chain. The suppliers provide raw material to a

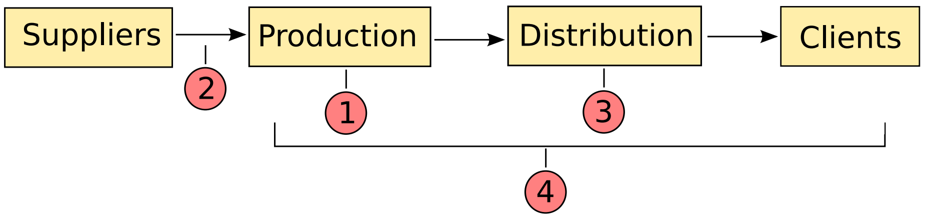

Figure 1: A typical supply chain

factory, where the final products are manufactured. These products are then distributed (e.g., through warehouses or distribution centers) to reach the final clients. Interesting information on supply chains can be found in [22, 23, 118]. Very often, the optimization problems arising along the supply chain are hard to solve together, this is why researchers often tackle the different problems separately. To address such specific problems, researchers must first model the problem, which often consists in formulating the problem with mathematical tools. Depending on the nature of the problem, different methods can then be used to solve the modeled problem. Exact methods, which guarantee to generate optimal solutions, can be used to solve problems when the time limit is consequent or when the size of the problem is reasonable. Recent advances in this field, such as mixed-integer programming, non-linear programming, decomposition techniques, and global optimization, look very promising and could be used in the future to solve bigger problems.

Nowadays realistic problems are often too consequent and exact methods are not competitive due to exponential computation times. As it is frequently the case, (meta)heuristics are designed, but optimality is no more guaranteed. Therefore, techniques such as genetic algorithms, tabu search or ant-colony are considered as standard methods in the metaheuristics field. Such techniques are accurately described in [45]. Metaheuristics operate at a higher level than heuristics, and thus tends to drive a sub-level heuristic for high performances, for instance by diversifying the search. A definition for metaheuristic is proposed in [94], which we quote here: "A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions." Etymologically, *meta* states for *at a higher level* and *heuristic* for *finding*. State-of-the-art reviews on metaheuristics are proposed in [14, 45], where different classification methods are exposed. As stated above, the major drawback of metaheuristics is to provide a solution without guarantee on its optimality. Thus, metaheuristics must be carefully designed and compared with exact methods on small instances where mixed-integer programming tools like CPLEX/Gurobi are still relevant, in order to be trusted on realistic instances.

Depending on the constraints of the problem faced by the industry, different (meta)heuristic methods could be used. We now describe a non-exhaustive list of state-of-the-art algorithms that are relevant to this thesis, and deeper information on each algorithm can be found in the different chapters of this thesis. Greedy heuristics are the most primitive and fastest ones. They start from an empty solution, and at each step, insert the component in the solution which minimizes the augmentation of the objective function. On the opposite, local search methods, such as descent, tabu search [50], or variable neighborhood search [56], start from a feasible solution (typically returned by a greedy algorithm) and at each step, try to improve the best visited solution by performing a tiny modification on the solution (called a move). In the population-based metaheuristics, a pool of solutions is kept in a memory, and new solutions are generated from the pool. Genetic algorithms [54], ant-colony optimization [30], and adaptive memory algorithms [110], are members of the latter methods. Often different methods are combined to reach better performances (such as a tabu search with adaptive memory algorithm). Each set of algorithms have different advantages and drawbacks. Greedy algorithms are fast but very often return solutions of poor quality. Local search methods are slower, but usually find solutions of better quality even if they are often stuck in local optima, which probably forbid to find better solutions. Finally the population-based methods are very powerful but usually require more memory to perform. Therefore, the method used to address a problem must fit the company requirements such as time limit, computation power, techniques, etc. As each problem has its own constraints and specificity, it is hard to find the criteria to accurately measure the quality of a solution, and therefore of a method. In [129], guidelines are provided to produce efficient metaheuristics for various problems, according to various criteria (e.g., quality of the results, speed, robustness, simplicity, ability to take advantage of the problem structure). In addition, for some companies, one objective function can be clearly defined, but as it is commonly the case, multiple objectives must be tackled together. A lexicographic order (i.e., a hierarchy in the objectives is provided) of the objectives is often used to handle multiple objectives, but other techniques such as multicriteria decision-making [82] can also be used. The company requirements are then used to define which techniques suit best for the problem.

In this thesis, four different problems are presented and tackled. The first project (bullet 1 in Figure 1), jointly realized with *Politecnico di Milano* (Italy), focuses on a job scheduling problem with smoothing issues. This problem arises in factories where different machines are used to perform different tasks, such as car plants, where cars are scheduled to be produced on customers orders. All the tasks cannot be performed on all the machines, and managers want to smooth the use of the available resources.

In the second project (bullet 2 in Figure 1), performed with *Renault France*, items must be packed in trucks such that they fulfill factory constraints. It is an extension of the well-known strip-packing problem, where items must be packed in a bin of fixed size. Here, a set of items must be packed in a truck, with truck sizes constraints. A total of almost 600 instances were tackled. These trucks are used to bring car pieces to the plants which produce the cars of *Renault*, and therefore packing is extremely important to ensure correct delivery times and the minimization of the number of used trucks.

The third project (bullet 3 in Figure 1) was realized with the *CIRRELT* center in Montreal (Canada). The *CIRRELT* is the Interuniversity research center on Enterprize Networks, Logistics and Transportation. The problem tackled there was an extension of the well-known vehicle routing problem. Here, global positioning devices (GPS) are used to constantly monitor the positions of the vehicles in an online fashion. Dynamic travel times and perturbations are encountered along the routes, as well as time-windows (time frames to visit customers). Optimization techniques are proposed to deal with such uncertainties.

Finally, the last project (bullet 4 in Figure 1) is an inventory dispatching problem faced by a Swiss watch company (which cannot be named due to a non-disclosure agreement). Three different perturbations are encountered along the 3-level supply chain (one at the production level, one at the wholesalers level, and one on the demand). An accurate simulator has been designed and solutions' solvers were defined to help decision-makers select the best decision at the right moment. Exhaustive sensitivity analysis is proposed to highlight the parameters that need to be addressed more carefully.

The projects are presented as four different chapters. Each chapter is closed by relevant conclusions and future works, and was submitted to an international journal for publication. The authors of each paper are mentioned at the beginning of the corresponding chapter.

Chapter 1

# Metaheuristics for a job scheduling problem with smoothing costs relevant for the car industry

Jean Respen - *University of Geneva, Switzerland*
Nicolas Zufferey - *University of Geneva, Switzerland*
Edoardo Amaldi - *Polytechnic University of Milan, Italy*

We study a new multi-objective job scheduling problem on non-identical machines with applications in the car industry, inspired by the problem proposed by the car manufacturer *Renault* in the ROADEF 2005 Challenge. Makespan, smoothing costs and setup costs are minimized following a lexicographic order, where smoothing costs are used to balance resource utilization. We first describe a mixed integer linear programming (MILP) formulation and a network interpretation as a variant of the well-known vehicle routing problem (VRP). We then propose and compare several solution methods, ranging from greedy procedures to a tabu search and an adaptive memory algorithm. For small

instances (with up to 40 jobs) whose MILP formulation can be solved to optimality, tabu search provides remarkably good solutions. The adaptive memory algorithm, using tabu search as an intensification procedure, turns out to yield the best results for large instances, with up to 500 jobs and 8 machines.

## 1.1 Introduction

Since production systems involve various important aspects such as cost, time and available (human and material) resources, multi-objective planning problems are of growing practical relevance. As mentioned in [78], multi-objective scheduling problems can be tackled using five different approaches: the *lexicographic, utility, goal programming, simultaneous* or *interactive* approaches. No perfect approach comes out and each one has its own advantages and drawbacks. Some approaches require more parameters whereas others are unable to generate efficient solutions under certain conditions. See the survey [32] on multi-objective combinatorial optimization, including theoretical results as well as exact and heuristic methods, and [64] on multi-objective metaheuristics. A good reference book on this topic can be found in [31].

Multi-objective scheduling problems often involve minimizing the makespan while considering setup costs and times. Various approaches have been proposed to tackle makespan minimization in the literature (e.g., [97]). For a survey on scheduling techniques accounting for setup issues, the reader is referred to [1]. Single machine scheduling methods to minimize a regular objective function under setup constraints are proposed in [6]. In [37], heuristic methods and a branch and bound procedure are described for parallel machines subject to setup constraints. For the same problem, a computationally intensive nonlinear mixed integer model and an approximation method for real-world sized instances are proposed in [2]. In [88], a hybridization of a particle swarm and local search algorithms is proposed to solve the flexible multi-objective job-shop problem. Whereas in a job-shop problem a set of jobs must be scheduled on a set of different machines and each job has a specific routing on the machines, a flexible job-shop problem is an extension where each job can be processed by a set of different machines along different routes.

Nowadays, new constraints, known as *smoothing constraints*, are attracting a growing attention in the area of job scheduling (see the survey on smoothing constraints known as “balancing in assembly line” in [10]). Smoothing constraints (or costs) allow to schedule the jobs with a well-balanced consumption of the production resources. These constraints are now widely used, in particular for car sequencing problems (see for example [8]), where cars must be scheduled before production in an order respecting various constraints (colors, optional equipment, due dates, etc.), while avoiding overloading some important resources. As an example, if the yellow cars with air-conditioning are scheduled first, the unlucky customer who ordered a grey car without air-conditioning may wait for a long time. For the car plant, balancing between optional equipment and colors allows to respect customers deadlines and to prevent overloading some resources (machines or employees), which has an impact on cost reduction. As mentioned in [126], there is a complex tradeoff at the core of many practical scheduling problems, which involves balancing the benefits of long production runs of a similar product against the costs of completing work before it is needed (and potentially causing other work to be tardy).

In this paper, we address a multi-objective production scheduling problem with smoothing costs inspired by the problem proposed by the car manufacturer *Renault* in the ROADEF 2005 Challenge (`http://challenge.roadef.org/2005/en/`). In the *Renault* problem, car families are defined so that two cars of the same family contain the same optional equipment. Each car option $i$ is associated with a $p_i/q_i$ ratio constraint requiring that at most $p_i$ vehicles with option $i$ can be scheduled in any subsequence of $q_i$ vehicles, otherwise a penalty occurs. This penalty was artificially created by *Renault* in order to penalize the possible congestion (which corresponds to an unbalanced use of some production resources) that might occur in the assembly line. Since another goal consists in minimizing the number of color changes in the production sequence, the overall objective is to minimize a weighted function involving the numbers of ratio constraint violations and color changes. The resulting problem is $NP$-hard and no exact algorithm can be competitive because the instances involve hundreds of cars. A survey of the above challenge can be found in [120]. The winning team proposed a very fast local search which combines a standard local search with a tuned transformation step [33]. The team ranked second developed a variable neighborhood search based on an iterated local search procedure, along with intensification and diversification strategies [108]. Efficient tabu search methods were also devised (e.g., [26, 130]).

The multi-objective scheduling problem we propose and investigate here is a variant of the *Renault* problem. Unlike in the *Renault* problem, we consider several machines (resources) which are not identical, eligibility constraints (a job cannot necessarily be performed on all the machines), setup constraints, and three different objectives functions. Adopting a realistic priority among the objectives, we aim at minimizing in a lexicographic order the overall makespan, smoothing costs, and setup costs. Mathematically such a lexicographic order allows to convert a multi-objective problem into a problem with a single objective. Preliminary experiments showed the relevance of the proposed lexicographical approach, as many solutions with the same first objective values are very likely to occur. In the studied problem, we use a $p_i/q_i$ ratio constraint with $p_i = 2$ and $q_i = 3$. In other words, a penalty occurs if more than two jobs of the same family are sequentially produced on the same machine. As confirmed in [90], the problem considered in this paper is of obvious interest from a modern car production point of view. Indeed, car manufacturers often face problems where smoothing different resources is mandatory. As stated above, color changes is of important matter. In addition, resources such as the workforce must often be smoothed, for example by alternatively performing tasks that requires more workers with less constrained workforce tasks. The problem we describe here is interestingly complete as it intends to include different type of real-world constraints, which were never jointly considered before.

Building on our preliminary work [106], this paper adds: an extended literature review, a practical motivation, an accurate test of a mixed integer linear programming (MILP) formulation, new and more efficient metaheuristics (namely, a greedy randomized adaptive search procedure (GRASP), a refined tabu search with intensification and diversification procedures, and an adaptive memory algorithm), and computational results for a much larger set of instances.

The remainder of the paper is organized as follows. In Section 1.2, we describe the problem under consideration, pointing out the differences with respect to the *Renault* problem proposed in the ROADEF 2005 Challenge and we give a MILP formulation. In Section 1.3 we provide a network interpretation. Since the MILP formulation is very challenging even for small-size instances, in Section 1.4, we describe three advanced metaheuristic methods, namely a GRASP, a tabu search and an adaptive memory algorithm. Computational results are reported and discussed in Section 1.5. First, we evaluate the quality of the solutions provided by our tabu search method for small instances (with up to 40 jobs) with respect to the optimal solutions found with the MILP formulation. Then we compare the performance of all the heuristics on larger instances (with more than 100 jobs) that cannot be tackled with the MILP formulation. Finally, Section 1.6 contains some concluding remarks.

## 1.2 Problem description and MILP formulation

In the considered problem (P), a set of $n$ independent jobs have to be scheduled on a set of $m$ unrelated (non identical) parallel machines. Each job $j$ belongs to one of the $g$ available families and has a processing time $p_j^i$ depending on the machine $i$. For each pair of job $j$ and machine $i$, the eligibility of $j$ on $i$ is specified by the binary parameter $u_j^i$. The goal is to minimize not only the makespan ($f_1$) but also the smoothing costs ($f_2$) and the setup costs ($f_3$), which take real values and have different units. A realistic lexicographic approach is adopted with the following priority order: $f_1 > f_2 > f_3$. Note by the way that a lexicographic approach was also used in the ROADEF 2005 Challenge, and is relevant from a practical standpoint. As mentioned in Equation (1.1), the lexicographic optimization is achieved via multiplicative coefficients $\alpha > \beta > \gamma$, where $\alpha$, $\beta$ and $\gamma$ are chosen such that no deterioration on $f_i$ can be compensated with improvements on $f_{i+1}$. Hence, the units of the different $f_i$'s are of minor importance. A small value of $C_{max}$ usually indicates a high occupancy rate of the machines, and as a consequence, the production system will be available sooner for future commands.

As previously mentioned, a smoothing cost occurs whenever more than two jobs of the same family are sequentially produced on the same machine. For every triple of jobs $\{j, j', j''\}$, the parameter $f_{jj'j''} = 1$ if $j, j', j''$ are of the same family, and 0 otherwise. Then $k_f$ denotes the smoothing cost associated with the family $f$ to which job $j$ belongs, which is incurred only if $f_{jj'j''} = 1$ and if jobs $j, j', j''$ are consecutively scheduled on the same machine.

The setups are job and machine dependent: $c_{jj'}^i$ (resp. $s_{jj'}^i$) is the setup cost (resp. time) encountered to prepare machine $i$ to perform job $j'$ after job $j$. There are two types of setups: major ones (if the involved jobs belong to two different families) and minor ones (otherwise). Minor setups can be considered as a small encountered time/cost associated with a technician who needs to slightly modify the configuration of the machine to perform the next job. On the opposite, a major setup occurs when external technicians (working at a higher hourly rate and bill for transportation) or a significant machine transformation (e.g., its capacity, its reprogramming) are required.

The multi-objective scheduling problem (P) is clearly $NP$-hard since it admits as a special case the classical $NP$-hard multiprocessor problem of minimizing the makespan (every job is eligible on every machine and no smoothing and setup costs are considered).

### 1.2.1 MILP formulation

Consider the following sets of binary decision variables:

- for every machine $i$ and pair of jobs $j, j'$, $x^i_{jj'} = 1$ if job $j$ is followed by job $j'$ on machine $i$, and 0 otherwise;
- for every machine $i$ and triple of jobs $j, j', j''$, $y^i_{jj'j''} = 1$ if jobs $j, j', j''$ are consecutively scheduled in this order on machine $i$, 0 otherwise;
- for every machine $i$ and job $j$, $z^i_j = 1$ if job $j$ is scheduled on machine $i$, and 0 otherwise.

In addition, for every job $j$ and machine $i$, let $r^i_j$ be a real variable representing the order (or rank) of job $j$ if it is scheduled on machine $i$ ($r^i_j = 0$ if job $j$ is not scheduled on machine $i$).

Using the above variables and parameters, the problem can be formulated as the following mixed integer linear program:

minimize

$$\begin{aligned} f &= \alpha \cdot f_1 + \beta \cdot f_2 + \gamma \cdot f_3 \\ &= \alpha \cdot C_{max} + \beta \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{j'=1}^{n} \sum_{j''=1}^{n} k_j \cdot f_{jj'j''} \cdot y^i_{jj'j''} + \gamma \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{j'=1}^{n} c^i_{jj'} \cdot x^i_{jj'} \end{aligned} \tag{1.1}$$

subject to

$$\sum_{j=1}^{n} p^i_j \cdot z^i_j + \sum_{j=1}^{n} \sum_{j'=1}^{n} s^i_{jj'} \cdot x^i_{jj'} \leq C_{max} \quad \forall i \tag{1.2}$$

$$z^i_j \leq u^i_j \quad \forall i, j \tag{1.3}$$

$$z^i_j + z^i_{j'} \geq 2 \cdot x^i_{jj'} \quad \forall i, j, j' \tag{1.4}$$

$$2 \cdot y^i_{jj'j''} \leq (x^i_{jj'} + x^i_{j'j''}) \cdot f_{jj'j''} \quad \forall i, j, j', j'' \tag{1.5}$$

$$(x^i_{jj'} + x^i_{j'j''}) \cdot f_{jj'j''} - 1 \leq y^i_{jj'j''} \quad \forall i, j, j', j'' \tag{1.6}$$

$$\sum_{j=1}^{n} z^i_j - 1 = \sum_{j=1}^{n} \sum_{j'=1}^{n} x^i_{jj'} \quad \forall i \tag{1.7}$$

$$\sum_{j'=1}^{n} x^i_{jj'} \leq 1 \quad \forall i, j \tag{1.8}$$

$$\sum_{j=1}^{n} x^i_{jj'} \leq 1 \quad \forall i, j' \tag{1.9}$$

$$z^i_j \leq r^i_j \leq n \cdot z^i_j \quad \forall i, j \tag{1.10}$$

$$r^i_{j'} \geq (r^i_j + 1) - n \cdot (1 - x^i_{jj'}) \quad \forall i, j, j' \tag{1.11}$$

$$x^i_{jj'} + x^i_{j'j} \leq 1 \quad \forall i, j, j' \tag{1.12}$$

$$\sum_{i=1}^{m} z^i_j = 1 \quad \forall j \tag{1.13}$$

$$0 \leq y^i_{jj'j''} \leq 1 \quad \forall i, j, j', j'' \tag{1.14}$$

$$x^i_{jj'}, z^i_j \in \{0, 1\} \quad \forall i, j, j'. \tag{1.15}$$

Constraints (1.2) ensure a correct makespan computation. Constraints (1.3) ensure that eligibility is satisfied on each machine. Constraints (1.4) allow two jobs to be scheduled consecutively only if they are scheduled on the same machine, whereas Constraints (1.5) and (1.6) ensure that three jobs can be scheduled consecutively only if all the three jobs are scheduled consecutively two at a time. Constraints (1.7) guarantee that the correct number of jobs will be consecutively scheduled, while Constraints (1.8), (1.9) and (1.12) ensure the correct number of consecutive jobs. Constraints (1.10) and (1.11) are the sub-tour elimination constraints given in [115], which involve, for each pair of job $j$ and machine $i$, a real variable $r_j^i$ indicating the order of job $j$ on machine $i$. Constraints (1.13) guarantee that each job is scheduled exactly once. Finally, fractional values are possible in Constraints (1.14) because of the objective function (1.1) and of the Constraints (1.5) and (1.6). Clearly, relaxing the integrality of the $y_{jj'j''}^i$ variables makes the MILP formulation easier, and ensures that $y_{jj'j''}^i \in \{0, 1\}$. Note that all constraints are needed to have a valid MILP formulation of problem (P).

### 1.2.2 Illustration

To illustrate the problem, we consider a small instance with two machines ($m_1$ and $m_2$). Table 1.1 gives processing times $p_j^i$ per job per machine, family identifier $f_j$ per job $j$, eligibility constraints $u_j^i$ per job and machine, and finally smoothing costs $k_f$ per family. Table 1.2 gives setup costs and times (it is assumed that setup times and costs are equal). For a realistic implementation of the lexicographic order, we take $\alpha = 1,000,000$, $\beta = 1,000$, and $\gamma = 1$. A feasible solution (fulfilling eligibility constraints) is proposed in the Gantt diagram of Figure 1.1. Let us now derive its objective function value. $C_{max}$ is equal to 551 as machine 1 involves the longest total processing times plus minor setup times $(119+5+105+5+152+6+159)$. Setup costs are equal to $5+5+6+5 = 21$ and are represented as black stripes on Figure 1.1. Machine 1 involves smoothing costs because it schedules two times three consecutive jobs (jobs 3-1-4 then 1-4-6) of the same family, and the smoothing costs amount to $45+45 = 90$. Thus the objective function value is $1 \cdot 21 + 1,000 \cdot 90 + 1,000,000 \cdot 551 = 551,090,021$. Figure 1.2 shows the optimal solution found by the above MILP formulation. The objective function value is $452,045,048$.

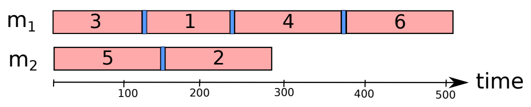

Figure 1.1: Gantt diagram for a feasible solution

Table 1.1: Processing times and families

| $j$ | $p_j^1$ | $p_j^2$ | $f_j$ | $u_j^1$ | $u_j^2$ | $k_f$ |
|---|---|---|---|---|---|---|
| 1 | 105 | 108 | 1 | 1 | 1 | 45 |
| 2 | 105 | 154 | 2 | 0 | 1 | 52 |
| 3 | 119 | 155 | 1 | 1 | 1 | 45 |
| 4 | 152 | 110 | 1 | 1 | 1 | 45 |
| 5 | 150 | 154 | 2 | 0 | 1 | 52 |
| 6 | 159 | 125 | 1 | 1 | 1 | 45 |

Table 1.2: Setup costs and times per machine

| $j$ | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m_1$ | $m_2$ | $m_1$ | $m_2$ | $m_1$ | $m_2$ | $m_1$ | $m_2$ | $m_1$ | $m_2$ | $m_1$ | $m_2$ |
| 1 | - | - | 35 | 32 | 6 | 9 | 5 | 7 | 32 | 31 | 5 | 10 |
| 2 | 37 | 49 | - | - | 32 | 38 | 45 | 50 | 6 | 7 | 40 | 42 |
| 3 | 5 | 7 | 45 | 35 | - | - | 6 | 7 | 35 | 48 | 8 | 5 |
| 4 | 6 | 8 | 47 | 30 | 7 | 9 | - | - | 33 | 30 | 6 | 8 |
| 5 | 36 | 47 | 9 | 5 | 45 | 36 | 35 | 39 | - | - | 42 | 41 |
| 6 | 5 | 8 | 45 | 40 | 5 | 7 | 8 | 5 | 38 | 32 | - | - |

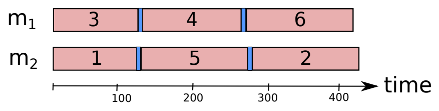

Figure 1.2: Gantt diagram for the optimal solution

## 1.3 Network interpretation

Assuming that setup costs are proportional to setup times (which is realistic from a practical standpoint), our production problem can be easily interpreted as a variant of the well-known vehicle routing problem (VRP) in a network.

Let the $n$ jobs be represented by $n$ vertices labeled $1, \ldots, n$ of a complete directed graph $G = (V, E)$. A color is associated with each job family. Since each vertex belongs to a unique family, it has a single color and the vertices can be grouped by color. Each vertex is connected to all the other vertices by an arc. Arcs are used instead of edges as setup times are not necessarily symmetric. If two vertices are of the same family, then the arc linking them is colored with the family color and the corresponding setup time is minor. Otherwise, the arc is uncolored and the setup time is major. Two vectors are associated with each arc $(j, j')$: a processing time vector $P_{j'} = (p^1_{j'}, p^2_{j'}, \ldots, p^m_{j'})$ and a setup time vector $S_{jj'} = (s^1_{jj'}, s^2_{jj'}, \ldots, s^m_{jj'})$. In addition, a virtual uncolored vertex labeled 0 is defined. With each arc $(0, j)$ are associated a processing time vector $P_j = (p^1_j, p^2_j, \ldots, p^m_j)$, and a setup time vector $S_{0j}$ where $s^i_{0j} = 0$ for each machine $i$. With each arc $(j, 0)$ are associated a processing time vector $P_0$ where $p^i_0 = 0$ for each machine $i$, and a setup time vector $S_{j0}$ where $s^i_{j0} = 0$ for each machine $i$.

In the VRP, $n$ clients have to be visited exactly once by a set of $m$ vehicles so as to optimize an objective function (see [47, 68] for recent surveys on the topic). Numerous extensions of this problem have been proposed, such as the vehicle routing problem with time windows (VRPTW) or the vehicle routing problem with pick-up and delivery (VRPPD) (see [55] for a good reference book on the existing VRP extensions). The VRP is considered as a reference $NP$-hard combinatorial problem and is still attracting considerable attention. A particularly interesting extension is the multimodal VRP problem, where different means of transport are used. It is especially relevant to overseas transportation of goods. A typical example would be a trip starting with a truck, continuing with a boat or a plane, followed by a train, and the final trip to the delivery point uses a second truck. In this example, it is obvious that changing the mean of transport would cost a lot to the company selling the good, but would still be cheaper than using a single mean of transport (imagine a truck traveling from Asia to West of Europe) or would be mandatory if a single mean of transport is impossible. Thus a smoothing cost would force the algorithm to find intermodal solutions. Setup costs and times would be involved when changing the mode of transport, and the processing times would be the trip time needed by each vehicle.

The correspondence between (P) and the VRP is the following. The clients correspond to the jobs, the vehicles to the machines, and the sequence of vertices visited by vehicle $i$ to the sequence of jobs executed by machine $i$. Given the graph $G$, the objective is to find $m$ circuits (starting from and ending at vertex 0, representing a depot), such that every vertex is visited exactly once. An illustration with $n = 8$ and $m = 2$ is given in Figure 1.3, where edges are used instead of arcs for the sake of simplicity (the graph is not drawn complete for the same reason). Vertices are grouped by color classes, and dashed edges connecting vertices of different color classes are uncolored. On the contrary, colored plain edges connect vertices of the same color class. A double weighted edge $[1, 2]$ is indicated as an example: job 1 has a processing time of 110 on machine 1 and 135 on machine 2; a minor setup of 6 occurs when finishing job 1 and preparing machine 1 for job 2, and the corresponding setup for the second machine is 8. Figure 1.4 shows a feasible solution for the instance of Figure 1.3. Note that the circuit visiting the first color class encounters a smoothing cost because it consecutively visits three vertices with the same color.

Network formulations (NETFORMs) (see [52, 53]) have shown significant efficiency to model complex problems, with huge cost savings and greatly speed completion. The NETFORM concept is used to model both linear and nonlinear problems with a graph, and is able to model entirely complex issues. Whereas the standard approach is to develop a linear formulation to model the problem, in contrast, a NETFORM can be used to model and then solve the problem directly from its graphical representation (e.g., [57, 112]). Regarding problem (P), a NETFORM has been developed, but the resulting graphical formulation is not representative enough to be showed here with two dimensions. Indeed, a NETFORM would require the $n$ jobs and the $m$ machines to be represented with vertices. From each job vertex $j$ to each machine vertex $i$, arcs of type $(j_{a,b}, i)$ should be drawn, where $j_{a,b}$ represents a sequence of three consecutive jobs $a - b - j$ multiplied by $k_j$ ($f_{abj} \cdot k_j$ with the notation introduced in Section 1.2.1). In addition, setups (which are job and machine dependent) should weight these arcs with $s_{bj}$ and $c_{bj}$ for each job $b$. Even if this formulation is too complex to be drawn, it has allowed us to improve the quality of our algorithms, and is currently still a good approach to validate methods to tackle (P) and to ensure their accuracies.

Since (P) contains specific features such as smoothing costs, eligibility constraints, and setup costs and times, existing VRP algorithms cannot be directly applied. Therefore, we decided to address (P) as a scheduling problem with dedicated specialized algorithms, and leave the vehicle routing approach for future research. However, since metaheuristics such as tabu search and adaptive memory algorithms proved to be very powerful for the VRP and some of its extensions (e.g., [25, 43, 110]), we design tabu search and adaptive memory algorithms for (P).

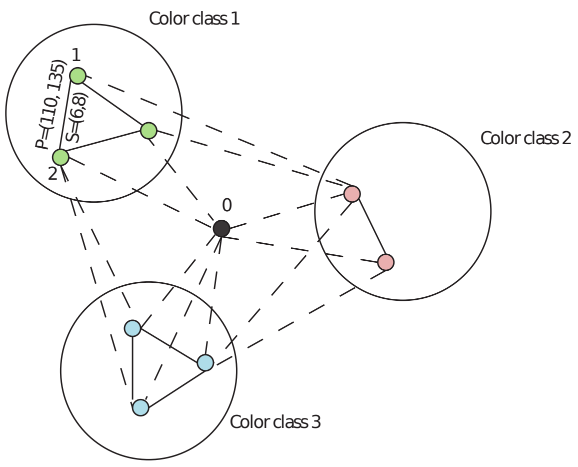

Figure 1.3: Network representation of the problem

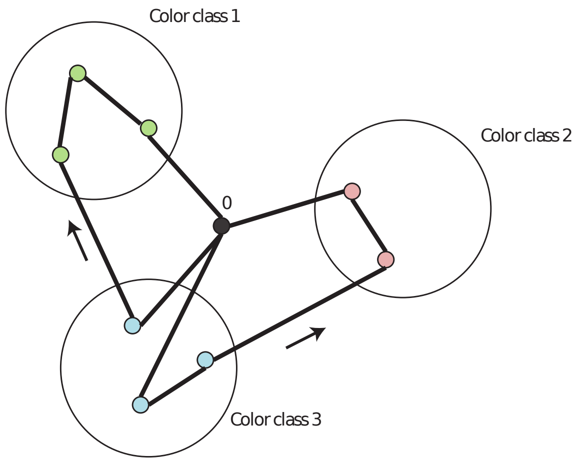

Figure 1.4: A possible feasible solution for Figure 1.3

## 1.4 Heuristic algorithms

In this section, we describe different solution methods for problem (P), ranging from a greedy randomized adaptive search procedure (GRASP) to more efficient metaheuristics such as tabu search and population based algorithms. All these methods will be compared according to a given time limit of $t$ seconds. If a method stops before $t$, it is restarted as long as $t$ is not reached, and the provided solution is the best one generated within $t$ seconds. In the following algorithms, the lexicographic order is used as follows when comparing the value of two solutions $s_1$ and $s_2$: $f_i(s_1)$ and $f_i(s_2)$ are only computed if $f_{i-1}(s_1) = f_{i-1}(s_2)$. This allows to save a significant amount of computing time. The reader is referred to [45] for information on local search techniques and metaheuristics, and to [129] for guidelines on an efficient design of metaheuristics according to various criteria.

### 1.4.1 Greedy and descent procedures

First we briefly summarize some simple procedures (three greedy ones and two descent ones) that we use as components of the proposed metaheuristics and for comparison purposes.

In the greedy procedures, to build a feasible solution step by step from scratch, we *select* at each step an unscheduled job and *insert* it at the best position (i.e., leading to the least increase in the objective function value), while respecting eligibility and setup constraints. Different job selection criteria are considered: (1) *randomly* among all the unscheduled jobs; (2) the least *flexible* job first (the flexibility of a job is defined as the number of machines it can be performed on); (3) *exhaustive* search (each insertion is tested for each non performed job). We propose three greedy procedures for (P): $GrR$ with random selection, $GrF$ based on the flexibility strategy, and $GrE$ based on the exhaustive strategy. Ties, which often occur, are randomly broken.

For the greedy algorithm, each restart occurs when a complete solution is built, whereas for a descent method, each restart occurs when a local optimum is found.

In $Des$, a simple descent local search is performed to improve an initial solution provided by $GrE$. In contrast, $LDes$ is a two-phase algorithm. In the first phase, the three greedy procedures are alternately used to generate initial solutions, on which a descend is then applied, if the running time is not above $\frac{t}{2}$ seconds. At the end of such a learning phase, a weight is assigned to each of the three greedy procedures, which is proportional to the average quality of the resulting solutions (i.e., the ones obtained at the end of the descent process). In the second phase, the next greedy procedure to be applied is randomly selected based on those weights. Thus, the greedy procedure leading in average to the best results in the first phase has more chance to be selected in the second phase. For both $Des$ and $LDes$, the best move out of a portion of $r\%$ (parameter) of all possible moves is performed at each iteration. $r = 100\%$ has been tuned for $Des$ and $LDes$. Note that several values for $r$ were tested in the interval $[0\%, 100\%]$.

### 1.4.2 GRASP

As mentioned in [34] and detailed in [104], a two-phase greedy randomized adaptive search procedure (GRASP) tends to produce better results compared to simple greedy procedures.

In the first phase, a set of solutions are built from scratch with a randomized greedy procedure. At each step of the solution construction process, a *restricted candidate list* (RCL) is filled with the $b$ best possible insertions, a single candidate is picked at random from RCL and inserted at the best position. Thus different runs are likely to yield different solutions.

In the second phase, a local search is typically used to rapidly improve the quality of the best solution found. The type of *move* (modification of the current solution) we consider consists in reinserting a job somewhere else in the solution. More precisely, a job can be moved to a different position on the same machine, or it can be moved from a machine to another (as long as the eligibility constraint is satisfied). Preliminary experiments showed that $r = 100\%$ is an appropriate choice for $GRASP$.

In our *GRASP* adaptation for (P), the RCL size is tuned to 5 (based on various tests for values in the interval [1,20]). *Des* is used in the second phase. *GRASP* is summarized in Algorithm 1.

**Algorithm 1** GRASP

```
While a stopping condition is not met do
    1. Create an empty solution $s$
    2. While $s$ does not contain all the jobs do
        (a) Generate the RCL as a set of the 5 best possible insertions
        (b) Randomly pick a member among the RCL, and insert it in $s$
    3. Perform $Des$ on $s$
    4. If $f(s) < f(s^\star)$, set $s^\star = s$
Return the best solution $s^\star$
```

### 1.4.3 Tabu Search

Tabu search [51] relies on the observation that a basic local search often gets stuck in a small portion of the search space. To escape from local optima, when a move is performed from the current solution $x_r$ to the neighbor solution $x_{r+1}$, the reverse move is forbidden (tabu) for *tab* (parameter) iterations. *Diversification* and *intensification* strategies have been proposed to improve performance and find local optima of better quality. An interesting survey on tabu search can be found in [46] with general advices for efficient implementations.

In our tabu search $TS$ for problem (P), the initial solution is provided by $GrE$ and the *moves* consist in reinserting a job somewhere else in the solution (as in the second phase of GRASP). The best move out of a portion of $r\%$ (parameter) of all possible moves is performed at each iteration. Preliminary experiments showed that $r = 50\%$ is an appropriate choice for $TS$.

If a job $j$ is moved somewhere in the schedule at a given iteration, it is forbidden to move $j$ again for *tab* iterations. Preliminary experiments showed that it is reasonable to choose the integer *tab* uniformly at random in $[\frac{n}{10}, \frac{n}{2.5}]$ for small instances ($n < 100$), and in $[\frac{n}{25}, \frac{n}{13}]$ for large instances ($100 \leq n \leq 500$). Note that *tab* values were tested in the interval $[0, 3 \cdot n]$. In addition, a second tabu structure is used: when a job $j$, initially positioned between jobs $j'$ and $j''$, is inserted somewhere else in the schedule, it is then tabu to move $j$ back between $j'$ and $j''$ for $2 \cdot tab$ iterations. As this second tabu structure is less restrictive than the first one, the tabu duration is set to a larger value. Note that when a move improves the best ever visited solution $s^\star$, its two associated tabu durations are increased by a random integer uniformly generated within interval $[\frac{n}{25}, \frac{n}{15}]$. $TS$ contains an aspiration criterion, which allows a tabu move to be performed only if this move improves $s^\star$.

$TS$ also contains diversification and intensification procedures. On the one hand, if after $p_1$ (parameter) iterations, $s^\star$ is not improved, a diversification procedure is triggered, which consists of randomly moving $p_2$ (parameter) jobs in the current solution. Preliminary experiments showed that $p_1 = n \cdot 100$ for small instances ($n < 100$), $p_1 = n + 100$ for larger instances, and $p_2 = \frac{n}{2}$ (for all the instances) are reasonable values. On the other hand, when $s^\star$ is improved, an intensification procedure occurs, and consists in performing a descent with jobs swaps on a per machine basis. Note that parameters $p_1$ and $p_2$ were tested in interval $[0, n^2]$.

$TS$ is summarized in Algorithm 2, where $\mathrm{U}(a, b)$ uniformly returns a random integer in the interval $[a, b]$, and $f(s)$ denotes the objective function value of solution $s$.

### 1.4.4 Adaptive memory algorithm

Unlike in local search methods, a population-based algorithm simultaneously deals with several solutions. This allows to diversify the search and thus increase the probability to reach better local optima. Population-based algorithms fall within the class of evolutionary methods, which include genetic algorithms, ant colonies, swarm algorithms, and adaptive memory methods.

A basic version of the adaptive memory algorithm, first proposed in [110], is summarized in Algorithm 3, where steps (1), (2) and (3) amounts to a *genera-*

**Algorithm 2** Tabu search ($TS$) for problem (P)

```
Generate an initial solution $s$ with $GrE$
While the time limit $t$ is not reached do
    1. Perform a job move on $s$ to reach the best non tabu neighbor solution $s'$
    2. Each time a job $j$ is moved to reach a neighbor solution $s'$, forbid $j$ to
       be moved again for $tab = \mathrm{U}(\frac{n}{25}, \frac{n}{13})$ iterations, and forbid $j$ to move back
       between its previously adjacent jobs for $2 \cdot tab$ iterations
    3. If the move improves $s^\star$, increase both tabu durations by $\mathrm{U}(\frac{n}{25}, \frac{n}{15})$
    4. Set $s = s'$
    5. If $f(s) < f(s^\star)$, set $s^\star = s$ and perform a descent local search (swap jobs)
       on a per machine basis
    6. If $p_1$ iterations have been performed without improving $s^\star$, randomly move
       $p_2$ jobs in the current solution
Return the best solution $s^\star$
```

*tion.*

**Algorithm 3** Adaptive memory algorithm

```
Initialize the central memory $\mathcal{M}$ with solutions
While a stopping condition is not met do
    1. Create an offspring solution $\bar{s}$ from $\mathcal{M}$ by using a recombination operator
    2. Apply a local search operator on $\bar{s}$ (during $I$ iterations) and let $s$ denote
       the resulting solution
    3. Update $\mathcal{M}$ with the use of $s$
Return the best solution encountered
```

To describe our adaptive memory algorithm $AMA$ for problem (P), we need to define the way the population $\mathcal{M}$ is initialized, the recombination operator, the intensification (or local search) operator, and the memory update operator. $\mathcal{M}$ is initialized by randomly generating a few (typically between 10 and 20) solutions and by improving them with a local search. The intensification operator could be tabu search or a descent algorithm. The overall balance between the intensification and the diversification potential of the adaptive memory algorithm has to be tuned with parameter $I$ (which denotes the number of iterations that the intensification operator is run): the smaller it is, the more importance is given to diversification.

In $AMA$, the initialization of the population is performed as follows: while $\mathcal{M}$ contains less than 10 solutions (parameter tested in the interval [1,20]), construct a solution with $GrE$, improve it with $TS$ (without intensification and diversification) for $I$ iterations (parameter tested in the interval [1,2000] and tuned to 500), and finally insert it in $\mathcal{M}$.

The intensification operator is $TS$ (without intensification and diversification), which is run for 500 iterations. Let $s$ denote the resulting solution. In addition, let $s_{worst}$ be the worst solution of $\mathcal{M}$ (i.e., the solution with the largest objective function value). If $f(s) < f(s_{worst})$, the memory update operator replaces $s_{worst}$ with $s$, otherwise the oldest solution of $\mathcal{M}$ is replaced by $s$.

To formally describe the recombination operator, assume that a solution $s_p$ takes the following form: $s_p = \{M_1^p, M_2^p, M_3^p, \ldots, M_m^p\}$ where $M_i^p$ describes the schedule of the machine $i$. The recombination operator works as follows: first, we randomly select in $\mathcal{M}$ a solution $s_p$ and one of its $M_i^p$. We then insert the selected $M_i^p$ (without changing the associated job sequence) in the offspring solution $\bar{s}$. All the jobs of $M_i^p$ are then removed from all the solutions of $\mathcal{M}$ (to prevent scheduling twice the same jobs in $\bar{s}$). The solution $s_p$ cannot be selected in the next step of the recombination operator. The next non considered machine $i$ to be scheduled in $\bar{s}$ is the one associated with the solution $s_q$ of $\mathcal{M}$ such that $|\bar{s} \cup M_i^q|$ is maximized. In other words, at each step of the recombination operator, the number of additional scheduled jobs is maximized. At the end, the remaining jobs that are not scheduled in $\bar{s}$ are greedily inserted in the best positions. This operator is inspired from the very efficient recombination operator for the graph coloring problem [38].

An illustration of the recombination operator is given in Figure 1.5. Twelve jobs are scheduled on three different machines, and $\mathcal{M}$ contains the two solutions $s_1$ and $s_2$. First, a solution and a machine are selected at random and inserted in $\bar{s}$. Here solution $s_1$ and machine 3 are selected, thus $M_3^1$ is inserted in $\bar{s}$. Then the different solution (i.e., solution $s_2$) containing the largest number of additional jobs is selected (i.e., $M_1^2$) and inserted in $\bar{s}$ (note that job 9 is removed, as it is already scheduled in $\bar{s}$). $M_2^1$ is then inserted in $\bar{s}$ (without job 5 which is already scheduled in $\bar{s}$).

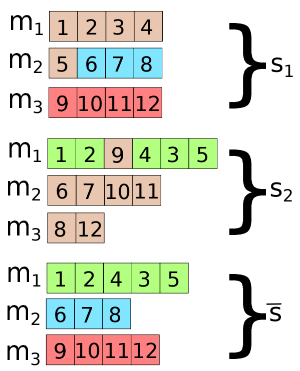

Figure 1.5: Illustration of the recombination operator of $AMA$

## 1.5 Computational results

In this section, we first compare the solutions provided by our tabu search method with those found by solving the MILP formulation with CPLEX 12.4 for small instances. For instances with more than 40 jobs, the MILP formulations turn out to be too challenging and thus only heuristics and metaheuristics will be compared. As the proposed (meta)heuristics have comparable performances with less than 100 jobs, we decided to compare them from $n \geq 100$.

Tests were performed on an Intel Quad-core i7 @ 3.4 GHz with 8 GB DDR3 of RAM memory, and the time limit $t$ is 3600 seconds. Note that considering larger time limits is not relevant from a practical standpoint, as the time actually available is even smaller [90]. $TS_D$ corresponds to $TS$ with the diversification procedure described above, and $TS_{DI}$ corresponds to $TS$ with diversification and intensification procedures. As $TS$, $TS_D$, $TS_{DI}$, and $AMA$ are not methods with restarts, the results are averaged over ten runs.

### 1.5.1 Instances

As (P) is a new problem, we generated dedicated instances inspired from the *Renault* instances proposed in the 2005 ROADEF Challenge `http://challenge.roadef.org/2005/en/`. Each instance is characterized by: the number of jobs $n \in \{10, 20, 30, 40, 100, 200, 300, 400, 500\}$, the integer number of machines $m \in [1, 8]$, the number of families $g = \max\{\lceil 0.02 \cdot n \rceil, 2\}$, the family identifier $f_j$ for each job $j$, the integer processing times $p_j^i$ of each job $j$ on each machine $i$ uniformly generated within $[100, 200]$, the integer setup costs $c_j^i$ uniformly generated within $[30, 50]$ for major setups, and within $[5, 10]$ for minor setups (we choose $s_j^i = 0.5 \cdot c_j^i$), and the smoothing cost $k_f$ of family $f$ uniformly generated within $[40, 60]$. To ensure the respect of the lexicographic order, we took $\alpha = 1,000,000$, $\beta = 1,000$, and $\gamma = 1$.

### 1.5.2 Results on the small instances

The MILP formulation was expressed in the AMPL modeling language and solved using CPLEX 12.4, with a time limit of 10 hours and a memory limit of 7 GB. The results obtained with the MILP formulation and those provided by $TS_D$ are summarized in Table 1.3. Only $TS_D$ is considered in the comparison as we only aim to benchmark a fast and reasonably efficient local search technique with solving the MILP formulation to optimality. In contrast with Section 1.4.3, $TS_D$ is used with the following parameters: $p_1 = n \cdot 10$ and $p_2 = \frac{n}{2}$. The first two columns indicate the values of $n$ and $m$ (observe that there are three instances per couple $(n, m)$). The third column ($f^\star$) corresponds to the objective function value of the best solution ever found by the MILP formulation or by $TS_D$. The fourth column indicates the percentage gap between the solution found by solving the MILP formulation and $f^\star$ (where the percentage gap between a solution value $f(s)$ and $f^\star$ is calculated as $\frac{f(s) - f^\star}{f^\star}$). The fifth column reports the computing time (in seconds) needed to find the best returned solution when solving the MILP formulation within the time limit or the memory limit (with a gap of $1e^{-4}$). The sixth column indicates the type of output provided by the MILP formulation: $O$ stands for *optimal*, $M$ for a *memory-limit exceeded* (when CPLEX required more than 7 GB of memory), and finally $T$ stands for *time-limit exceeded* (when CPLEX required more than 10 hours of computing time). When the type is $M$ or $T$, the solution returned is an upper bound, thus CPLEX can sometimes be defeated by $TS_D$. Note that $TS$ needs about 200 MB of memory on the instances with 500 jobs. The seventh column gives the percentage gap between the solution found by $TS_D$ and $f^\star$. Finally the last column gives the time (in seconds) required by $TS_D$ to find its best solution.

The results show that $TS_D$ is able to find optimal solutions on all the instances with 10 jobs in a computing time comparable to CPLEX. On the instances with 20 jobs, $TS_D$ finds an optimal solution in most of the cases, and usually requires much less computing time than CPLEX to reach its best solution. For instances with 30 or more jobs, $TS_D$ provides remarkably good results in a fair amount of computing time. For example, for the first instance with 30 jobs and 1 machine, CPLEX requires 16,508.63 seconds to find an upper bound whereas $TS_D$ finds the same upper bound in 0.87 seconds. Moreover, on four instances with 40 jobs, $TS_D$ finds a better solution than the upper bound returned by CPLEX. Note that for one instance with $n = 40$ jobs and $m = 1$ machine, CPLEX yields a large percentage gap with respect to the solution found by $TS_D$. The results obtained for the three instances with $n = 40$ and $m = 1$ suggest that the MILP formulations of the instances with $n = 40$ are harder for CPLEX when $m = 1$ than when $2 \leq m \leq 5$.

In summary, Table 1.3 shows that on the small instances, our tabu search algorithm is very competitive when compared to CPLEX and is strongly advised if the allowed computing time is small.

### 1.5.3 Results on the large instances

The results obtained with the (meta)heuristics on larger instances are summarized in Table 1.4. The first two columns indicate the values of $n$ and $m$ (observe that there are five instances per couple $(n, m)$). The third column ($f^\star$) corresponds to the objective function value of the best solution ever found by any of the algorithms. The fourth column reports the percentage gap between the best solution of $GrR$ and $f^\star$. The next columns provide the same information for the other methods. The average results, reported in the last line of Table 1.4, show that $AMA$ performs better in terms of average solution quality than the other algorithms and suggest the following order of decreasing solution quality: $AMA$, $TS_{DI}$, $TS_D$, $TS$, $LDes$, $Des$, $GrE$, $GRASP$, $GrR$, and $GrF$.

Even for instances with the smallest values of $n$, $GrR$ and $GrF$ are dominated by $GrE$. This is likely due to the fact that $GrE$ tries all the possible insertions before performing the best one, whereas $GrR$ selects the next job to be scheduled in a random fashion and $GrF$ selects the least flexible job. On the opposite, the computing time required by $GrE$ is much larger (especially when $n$ grows) than the time required by $GrR$ or $GrF$ to find a feasible solution. The average results show that on the full set of instances, $GrR$ and $GrF$ show similar performance, whereas $GrE$ performs clearly better.

Concerning the local search methods, $Des$ significantly outperforms the greedy procedures. The initial solutions provided by $GrE$ are substantially improved and the learning process added in $LDes$ is relevant, as it leads to slightly better solutions than $Des$. The performance of $GRASP$ is disappointing. This may be due to the fact that, at each step of the construction phase, the next job to be scheduled is randomly selected in the restricted candidate list, which leads to non promising insertions most of the time. Even if $Des$ is triggered as soon as $GRASP$ finds a feasible solution, the solution quality remains poor.

Our tabu search $TS$ compares favorably with the greedy procedures and the descent methods $Des$ and $LDes$. The diversification and intensification procedures improve the average performances of $TS$. The diversification on its own decreases the average percentage gap of $TS$ by 0.18%. The intensification procedure involves a higher computational load, as a descent local search is performed on each machine, but leads to improvements in the average solution quality and thus should be used when computing time is not an issue.

Figure 1.6 shows the evolution of the objective function value when applying $TS$ (without diversification nor intensification) and $Des$ to the first instance of Table 1.4 with 500 jobs and 8 machines. $TS$ decreases much faster the objective function value and outperforms $Des$ after 1.5 seconds. For $TS$, Figure 1.7 shows the variation of the different components of the objective function. Note that the x-axis (time [s]) is log-scaled on both Figures. Slight variations of the main objective (makespan) are shown on Figure 1.7 while the current computing time is less than 10 seconds, but consequent variations are then affected on the objective function value on Figure 1.6. Furthermore, while the objective function decreases after 78 seconds and the makespan remains the same, the smoothing costs decrease and the setup costs increase. The lexicographic order of the objectives easily explains this behavior, as smoothing costs have a higher priority than setup costs. As the three objectives only encounter minor variations for the last 2000 seconds, diversification procedures would probably be useful for $TS$ on this instance. This is numerically confirmed: according to Table 1.4, the diversification procedure improved the results on this instance.

Finally, the $AMA$ algorithm turns out to perform best, as it finds the best averaged results on almost 85% of the instances. This shows the efficiency of the recombination operator, and the good compromise between the intensification and diversification (which relies on a relevant tuning of parameter $I$). For each $n$, Table 1.5 shows the average computing times required by the three tabu search methods and the adaptive memory algorithm to provide their best solutions. Again, $AMA$ is advised as it requires computing times comparable to or lower than the other methods, while leading to better results. When $n \geq 400$, $AMA$ requires significantly less computing times but still provides the best solutions, suggesting that this trend may remain unchanged when $n > 500$.

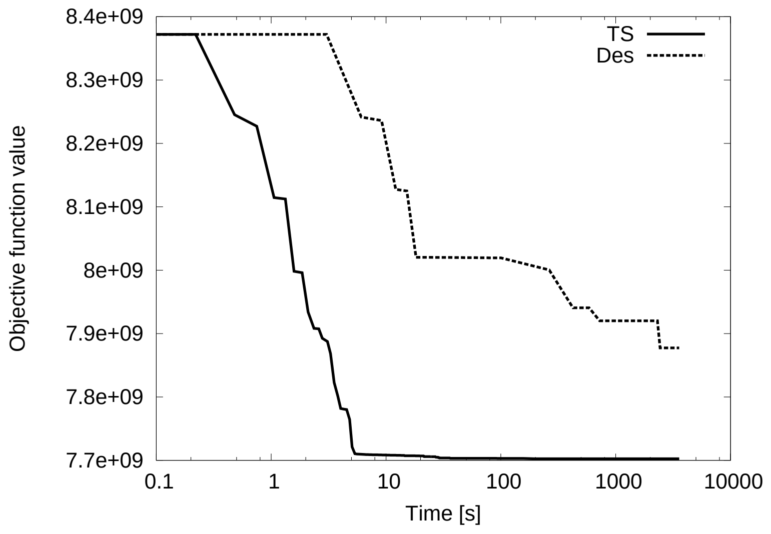

Figure 1.6: Behavior of $TS$ and $Des$ over time

Table 1.3: Results on the small instances ($10 \leq n \leq 40$)

| $n$ | $m$ | $f^\star$ | **MILP gap** | **MILP time** | **Type** | $TS_D$ **gap** | $TS_D$ **time** |
|---|---|---|---|---|---|---|---|
| 10 | 1 | 1,389,816,077 | **0.00** | 0.38 | O | **0.00** | 10.80 |
| 10 | 1 | 1,501,760,081 | **0.00** | 0.99 | O | **0.00** | 0.14 |
| 10 | 1 | 1,508,296,076 | **0.00** | 0.31 | O | **0.00** | 0.00 |
| 10 | 2 | 706,638,111 | **0.00** | 0.07 | O | **0.00** | 1.16 |
| 10 | 2 | 621,604,121 | **0.00** | 0.04 | O | **0.00** | 0.00 |
| 10 | 2 | 727,226,071 | **0.00** | 0.08 | O | **0.00** | 0.32 |
| 10 | 3 | 467,117,092 | **0.00** | 0.07 | O | **0.00** | 0.14 |
| 10 | 3 | 451,096,089 | **0.00** | 0.08 | O | **0.00** | 0.25 |
| 10 | 3 | 462,702,041 | **0.00** | 0.06 | O | **0.00** | 0.05 |
| 10 | 4 | 384,041,063 | **0.00** | 0.08 | O | **0.00** | 0.44 |
| 10 | 4 | 348,000,104 | **0.00** | 0.36 | O | **0.00** | 1.64 |
| 10 | 4 | 369,500,110 | **0.00** | 0.21 | O | **0.00** | 0.76 |
| 10 | 5 | 280,500,081 | **0.00** | 0.06 | O | **0.00** | 5.39 |
| 10 | 5 | 262,500,060 | **0.00** | 0.04 | O | **0.00** | 2.98 |
| 10 | 5 | 272,500,032 | **0.00** | 0.05 | O | **0.00** | 4.00 |
| 20 | 1 | 3,074,792,122 | **0.00** | 6,967.54 | M | **0.00** | 0.14 |
| 20 | 1 | 2,902,836,126 | **0.00** | 8,606.24 | M | **0.00** | 2.79 |
| 20 | 1 | 2,933,730,126 | **0.00** | 7,621.15 | M | **0.00** | 0.15 |
| 20 | 2 | 1,376,640,152 | **0.00** | 2,256.20 | O | **0.00** | 36.59 |
| 20 | 2 | 1,394,548,160 | **0.00** | 391.70 | M | 0.22 | 1,058.63 |
| 20 | 2 | 1,351,600,151 | **0.00** | 1,734.03 | O | **0.00** | 59.26 |
| 20 | 3 | 890,592,124 | **0.00** | 1,059.72 | O | **0.00** | 782.73 |
| 20 | 3 | 909,034,150 | **0.00** | 988.03 | O | **0.00** | 64.71 |
| 20 | 3 | 917,928,184 | **0.00** | 222.85 | O | 0.11 | 69.67 |
| 20 | 4 | 784,676,084 | **0.00** | 6.42 | O | **0.00** | 6.12 |
| 20 | 4 | 681,435,145 | **0.00** | 226.95 | O | **0.00** | 134.70 |
| 20 | 4 | 659,950,145 | **0.00** | 250.44 | O | **0.00** | 250.36 |
| 20 | 5 | 585,88,082 | **0.00** | 1,083.93 | M | **0.00** | 63.01 |
| 20 | 5 | 543,811,165 | **0.00** | 200.69 | O | 0.01 | 529.83 |
| 20 | 5 | 519,976,086 | **0.00** | 1.28 | O | **0.00** | 340.05 |
| 30 | 1 | 4,680,108,172 | **0.00** | 16,508.63 | M | **0.00** | 0.87 |
| 30 | 1 | 4,622,020,175 | **0.00** | 21,706.25 | M | **0.00** | 1.32 |
| 30 | 1 | 4,463,844,175 | **0.00** | 15,116.50 | M | **0.00** | 15.05 |
| 30 | 2 | 2,004,618,202 | **0.00** | 1,387.39 | M | 0.03 | 0.14 |
| 30 | 2 | 2,104,191,199 | **0.00** | 2,523.70 | M | 0.10 | 132.81 |
| 30 | 2 | 2,101,672,207 | **0.00** | 3,095.95 | M | 0.52 | 672.13 |
| 30 | 3 | 1,319,352,222 | **0.00** | 34,038.50 | T | 0.12 | 158.03 |
| 30 | 3 | 1,439,321,141 | **0.00** | 22,320.38 | M | **0.00** | 99.66 |
| 30 | 3 | 1,411,368,252 | **0.00** | 19,519.88 | M | **0.00** | 1,583.47 |
| 30 | 4 | 976,268,222 | **0.00** | 2,968.51 | M | 0.16 | 599.09 |
| 30 | 4 | 1,022,895,165 | **0.00** | 3,208.21 | M | 0.09 | 1,310.22 |
| 30 | 4 | 1,218,267,138 | **0.00** | 31,193.88 | M | **0.00** | 1,233.68 |
| 30 | 5 | 793,171,254 | **0.00** | 8,560.03 | M | 0.18 | 203.07 |
| 30 | 5 | 884,660,138 | **0.00** | 31,299.50 | M | 0.11 | 1,386.85 |
| 30 | 5 | 886,850,189 | **0.00** | 19,188.75 | M | **0.00** | 1,430.92 |
| 40 | 1 | 5,919,821,222 | 5.49 | 558.11 | M | **0.00** | 26.28 |
| 40 | 1 | 6,164,956,122 | 0.04 | 1,941.35 | M | **0.00** | 0.16 |
| 40 | 1 | 5,917,315,221 | 0.04 | 32,548.63 | T | **0.00** | 4.83 |
| 40 | 2 | 2,710,187,249 | **0.00** | 3,220.14 | M | 0.34 | 66.24 |
| 40 | 2 | 2,801,168,252 | **0.00** | 7,532.44 | M | 0.57 | 0.10 |
| 40 | 2 | 2,781,210,253 | **0.00** | 1,189.22 | M | 0.55 | 123.53 |
| 40 | 3 | 1,796,921,255 | **0.00** | 34,612.13 | T | 0.40 | 3.67 |
| 40 | 3 | 1,787,057,254 | **0.00** | 33,496.75 | T | **0.00** | 0.04 |
| 40 | 3 | 1,849,462,226 | **0.00** | 33,937.63 | T | 0.29 | 56.33 |
| 40 | 4 | 1,280,150,276 | **0.00** | 11,304.25 | M | 0.31 | 0.09 |
| 40 | 4 | 1,303,972,390 | 0.20 | 34,138.00 | T | **0.00** | 238.08 |
| 40 | 4 | 1,290,020,292 | **0.00** | 35,021.25 | T | 0.16 | 0.07 |
| 40 | 5 | 1,060,908,249 | **0.00** | 9,977.16 | M | 0.29 | 125.55 |
| 40 | 5 | 1,050,153,273 | **0.00** | 33,779.87 | T | 0.46 | 1,024.50 |
| 40 | 5 | 1,038,747,251 | **0.00** | 8,110.58 | M | **0.00** | 289.05 |
| **Averages** | | | 0.10 | 9,093.72 | | 0.08 | 236.87 |

Table 1.4: Results (percentage gaps) on the large instances ($100 \leq n \leq 500$)

| $n$ | $m$ | $f^\star$ | $GrR$ | $GrE$ | $GrF$ | $Des$ | $LDes$ | $GRASP$ | $TS$ | $TS_D$ | $TS_{DI}$ | $AMA$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 3 | 4,249,340,634 | 7.56 | 3.07 | 8.83 | 1.87 | 1.87 | 3.51 | 2.66 | 2.66 | 2.66 | **1.29** |
| 100 | 3 | 4,233,412,799 | 6.29 | 0.81 | 6.29 | 0.61 | 0.64 | 1.47 | 0.45 | 0.45 | 0.45 | **0.05** |
| 100 | 3 | 4,415,300,685 | 5.38 | 0.35 | 5.38 | 0.33 | 0.35 | 1.89 | 0.37 | 0.36 | 0.35 | **0.10** |
| 100 | 3 | 4,644,682,485 | 3.00 | 5.50 | 3.24 | 5.49 | 3.84 | 3.33 | 6.60 | 3.87 | 3.24 | **1.32** |
| 100 | 3 | 4,295,080,595 | 7.13 | 0.00 | 6.67 | **0.00** | 0.00 | 2.66 | 0.75 | 0.74 | 0.74 | 0.48 |
| 100 | 4 | 3,373,945,784 | 6.54 | 0.53 | 20.93 | **0.52** | 0.53 | 1.29 | 4.11 | 3.44 | 3.69 | 1.20 |
| 100 | 4 | 3,073,716,830 | 7.97 | 1.35 | 7.97 | 1.35 | 1.35 | 0.70 | 0.58 | 0.58 | 0.58 | **0.30** |
| 100 | 4 | 3,186,260,735 | 8.31 | 0.34 | 7.82 | 0.33 | 0.33 | 1.40 | 0.39 | 0.33 | 0.33 | **0.19** |
| 100 | 4 | 4,631,194,470 | 7.64 | 16.24 | 8.31 | 16.24 | 8.21 | 9.73 | 15.00 | 14.39 | 10.15 | **5.77** |
| 100 | 4 | 4,209,242,697 | 8.50 | 2.27 | 8.50 | 2.26 | 2.26 | 4.58 | 2.44 | 2.44 | 2.45 | **1.28** |
| 200 | 4 | 6,577,040,922 | 10.32 | 2.70 | 8.82 | **0.18** | 0.22 | 6.91 | 0.80 | 0.89 | 0.89 | 0.35 |
| 200 | 4 | 6,457,674,556 | 11.12 | 1.18 | 10.62 | 0.67 | 0.58 | 7.10 | 1.25 | 1.25 | 1.25 | **0.49** |
| 200 | 4 | 7,047,345,176 | 6.82 | 8.32 | 5.90 | 4.35 | 4.33 | 5.05 | 6.06 | 6.18 | 5.68 | **1.06** |
| 200 | 4 | 6,368,356,399 | 9.51 | 0.93 | 9.73 | 0.84 | 0.66 | 6.64 | 0.54 | 0.53 | 0.53 | **0.20** |
| 200 | 4 | 6,364,989,381 | 10.20 | 3.72 | 10.09 | 1.28 | 1.43 | 6.12 | 0.72 | 0.67 | 0.67 | **0.12** |
| 200 | 5 | 5,338,469,163 | 10.70 | 11.82 | 9.26 | 2.03 | 2.02 | 6.20 | 0.69 | 0.86 | 0.85 | **0.35** |
| 200 | 5 | 5,107,140,155 | 14.17 | 11.70 | 12.76 | 7.17 | 8.25 | 9.82 | 4.39 | 4.47 | 4.48 | **1.65** |
| 200 | 5 | 5,014,498,122 | 11.98 | 3.66 | 11.68 | 3.66 | 3.68 | 7.87 | 1.35 | 1.36 | 1.36 | **0.71** |
| 200 | 5 | 4,918,697,112 | 12.23 | 1.48 | 13.17 | 1.35 | 1.29 | 8.42 | 1.79 | 1.80 | 1.80 | **0.73** |
| 200 | 5 | 4,932,777,432 | 11.33 | 1.37 | 12.28 | 0.05 | **0.00** | 6.03 | 0.91 | 0.91 | 0.91 | 0.18 |
| 300 | 5 | 7,546,705,860 | 12.85 | 10.49 | 12.82 | 10.59 | 10.72 | 11.25 | 1.16 | 1.16 | 1.16 | **0.33** |
| 300 | 5 | 8,548,622,664 | 11.43 | 7.37 | 3.91 | 2.59 | 2.79 | 10.85 | 4.06 | 4.30 | 4.57 | **1.15** |
| 300 | 5 | 7,514,486,040 | 15.39 | 3.10 | 26.16 | 2.71 | 2.74 | 12.42 | 2.49 | 2.48 | 2.48 | **0.63** |
| 300 | 5 | 7,845,598,285 | 13.33 | 1.43 | 18.16 | 0.98 | 1.02 | 10.50 | 1.44 | 1.45 | 1.44 | **0.87** |
| 300 | 5 | 7,721,296,294 | 10.47 | 1.45 | 11.85 | 0.94 | 1.24 | 7.99 | 0.41 | 0.47 | 0.47 | **0.10** |
| 300 | 6 | 6,010,630,502 | 14.28 | 0.27 | 12.35 | **0.08** | 0.16 | 9.45 | 0.43 | 0.43 | 0.42 | 0.18 |
| 300 | 6 | 6,787,291,394 | 11.95 | 8.30 | 8.57 | **0.00** | 0.03 | 7.36 | 3.11 | 3.04 | 3.32 | 1.10 |
| 300 | 6 | 7,219,121,497 | 16.55 | 6.27 | 26.08 | 4.23 | 4.23 | 12.97 | 4.00 | 3.98 | 3.98 | **1.51** |
| 300 | 6 | 6,129,353,362 | 14.15 | 0.77 | 13.83 | 0.42 | 0.47 | 9.76 | 0.55 | 0.49 | 0.50 | **0.25** |
| 300 | 6 | 6,225,313,650 | 14.57 | 6.00 | 14.55 | 2.45 | 2.90 | 10.30 | 2.34 | 2.36 | 2.35 | **0.78** |
| 400 | 6 | 8,090,382,439 | 15.95 | 0.85 | 16.07 | 0.73 | 0.69 | 12.53 | 0.77 | 0.75 | 0.75 | **0.56** |
| 400 | 6 | 11,705,162,038 | 12.32 | 1.79 | 16.00 | 1.30 | 1.31 | 9.39 | 1.51 | 1.51 | 1.51 | **0.56** |
| 400 | 6 | 8,748,386,495 | 16.99 | 4.93 | 13.50 | 3.64 | 3.79 | 14.43 | 3.49 | 3.41 | 3.41 | **1.07** |
| 400 | 6 | 9,680,318,902 | 7.84 | 18.39 | 6.02 | 18.38 | 6.56 | 9.71 | 7.57 | 4.48 | 4.20 | **2.42** |
| 400 | 6 | 8,394,069,677 | 14.87 | 3.14 | 15.33 | 3.01 | 3.16 | 11.61 | 1.68 | 1.76 | 1.76 | **0.63** |
| 400 | 7 | 7,332,170,300 | 15.42 | 7.49 | 13.63 | 7.03 | 7.21 | 12.90 | 2.25 | 2.32 | 2.31 | **0.78** |
| 400 | 7 | 7,089,360,402 | 14.41 | 7.99 | 13.76 | 2.88 | 2.34 | 10.73 | 0.67 | 0.65 | 0.65 | **0.16** |
| 400 | 7 | 6,983,157,045 | 16.43 | 0.42 | 16.67 | 0.15 | **0.05** | 11.78 | 0.36 | 0.37 | 0.37 | 0.10 |
| 400 | 7 | 7,276,891,594 | 15.43 | 3.44 | 14.85 | 2.34 | 2.35 | 11.85 | 1.77 | 1.84 | 1.84 | **0.65** |
| 400 | 7 | 7,248,836,517 | 16.93 | 8.35 | 20.61 | 7.46 | 7.47 | 14.35 | 3.18 | 3.13 | 3.20 | **0.86** |
| 500 | 7 | 8,970,000,685 | 16.15 | 2.45 | 16.89 | 2.15 | 1.91 | 12.95 | 0.78 | 0.72 | 0.74 | **0.38** |
| 500 | 7 | 8,955,682,152 | 14.87 | 2.18 | 15.97 | 2.22 | 2.02 | 11.65 | 1.08 | 1.08 | 1.09 | **0.32** |
| 500 | 7 | 8,971,542,025 | 14.62 | 6.37 | 14.54 | 3.95 | 3.53 | 11.74 | 1.13 | 1.15 | 1.15 | **0.30** |
| 500 | 7 | 8,789,686,209 | 15.58 | 2.04 | 16.66 | 0.67 | 0.70 | 13.25 | 0.89 | 0.92 | 0.93 | **0.51** |
| 500 | 7 | 11,647,554,659 | 13.11 | 12.03 | 8.90 | 5.99 | 6.82 | 10.38 | 2.42 | **0.70** | 1.01 | 1.00 |
| 500 | 8 | 7,602,048,722 | 17.01 | 10.08 | 18.02 | 3.31 | 3.78 | 13.52 | 1.65 | 1.49 | 1.47 | **0.61** |
| 500 | 8 | 7,425,937,611 | 15.74 | 1.01 | 15.89 | 0.59 | 0.63 | 12.19 | 0.30 | 0.29 | 0.29 | **0.11** |
| 500 | 8 | 7,410,231,321 | 16.95 | 1.79 | 17.41 | 1.44 | 1.35 | 11.84 | 0.95 | 0.94 | 0.94 | **0.59** |
| 500 | 8 | 7,527,643,588 | 17.49 | 5.03 | 17.40 | 3.57 | 3.84 | 13.67 | 2.61 | 2.70 | 2.69 | **0.78** |
| 500 | 8 | 8,350,425,967 | 17.57 | 8.62 | 17.17 | 7.66 | 8.16 | 13.89 | 3.19 | 2.58 | 3.15 | **0.94** |
| **Averages** | | | **12.35** | **4.62** | **12.84** | **3.08** | **2.72** | **8.96** | **2.20** | **2.02** | **1.94** | **0.76** |

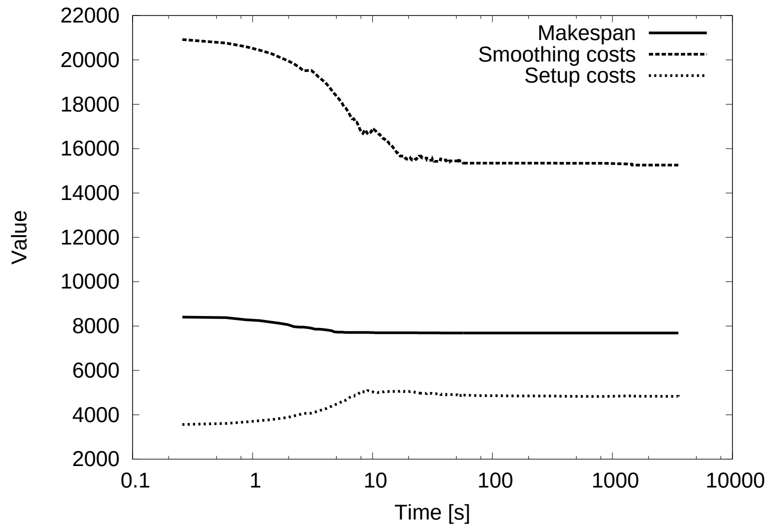

Figure 1.7: Values of the objective functions for the $TS$ run presented in Fig. 1.6

Table 1.5: Average computing time (in seconds) needed to get the best solutions

| $n$ | $TS$ | $TS_D$ | $TS_{DI}$ | $AMA$ |
|---|---|---|---|---|
| 100 | 648.32 | 600.26 | 547.73 | 430.82 |
| 200 | 1,175.79 | 909.67 | 1,478.01 | 1,538.92 |
| 300 | 2,034.97 | 1,416.29 | 1,787.66 | 1,787.36 |
| 400 | 2,329.38 | 2,761.35 | 2,630.58 | 2,170.91 |
| 500 | 2,568.90 | 3,020.97 | 2,598.77 | 1,628.75 |
| **Averages** | 1,751.47 | 1,741.71 | 1,808.55 | 1,511.35 |

## 1.6 Conclusions

We have addressed a new multi-machine multi-objective job scheduling problem with interesting applications for example in the car industry. The problem includes the following realistic features: jobs of various families, different machines (or production resources), makespan minimization, machine eligibility, machine and job dependent setup times and costs, as well as smoothing costs. We gave a MILP formulation and a network interpretation. Small instances can be solved via the MILP formulation. For larger instances, we developed and compared several (meta)heuristics, namely a GRASP, tabu search and an adaptive memory algorithm. When the number of jobs increases, the adaptive memory algorithm provides the best solutions and is very competitive in terms of trade-off between solution quality and computing time.

## Acknowledgments

The authors would like to thank Alain Nguyen (head of an Operations Research team at *Renault* R&D - Paris (France), and member of the organizing committee of the ROADEF 2005 Challenge dedicated to *Renault*) for his help in validating the practical relevance of this study.

# Chapter 2

# Metaheuristics for truck loading in the car production industry

Jean Respen - *University of Geneva, Switzerland*
Nicolas Zufferey - *University of Geneva, Switzerland*

The delivery of goods to car factories is a challenging problem. The French car manufacturer *Renault* is facing daily a complex truck loading problem where various goods must be packed into a truck such that they fulfill different constraints. As trucks can deliver goods to different factories on the same tour, classes of items have been defined, where a class is associated with a delivery point. As the number of items and the standard deviation of the sizes of the items are significant, no exact algorithm is competitive. A tabu search and genetic algorithms are proposed to tackle this problem. The results show that the most effective method is a genetic algorithm. An extension is then proposed, which consists in tackling all the instances within a given time limit. For this extension, results show that a combination of the algorithms is the most powerful strategy.

## 2.1 Introduction

The French car manufacturer *Renault* is facing a complex truck loading problem, where items need to be placed in a truck such that they fulfill different constraints. The items are typically car parts (wheels, brakes, chassis, etc.) used to build the car production. This problem is called here the *Renault* truck loading problem ($RTLP$). *Renault* is dealing on a daily basis with more than a thousand trucks, which have to deliver goods to car factories, making this problem relevant to their production plan. As a single truck can deliver goods to different delivery points, classes of items have been defined, where a class is associated with a delivery point. Each problem instance contains the size of the truck (in millimeters) and the various sizes of all the items that must fit in (in millimeters). The heights of the items can be ignored as they rely on complex factory constraints which are supposed to be already satisfied. At first sight, this problem seems related to a strip-packing 2D problem with rotation, which has been already covered by many research papers. In the strip-packing 2D problem, rectangular items must be packed in a single bin of fixed width and infinite length, with the objective of minimizing the total length of the packing. Relevant surveys on this topic can be found in [60, 74, 92, 109]. This problem is a strongly NP-hard combinatorial problem.

In this paper, we aim at tackling some new features proposed by *Renault*: different *classes* of items, and a significant *number* of items per truck in conjunction with a large *standard deviation* of the sizes of the items. Such elements make the considered problem more relevant to modern and realistic issues. To solve $RTLP$, *Renault* proposes a simple but efficient greedy heuristic ($SG$), and an advanced greedy heuristic called "look-ahead greedy" ($LAG$). *Renault* proposes us to benchmark their algorithms with state-of-the-art metaheuristics, to measure the possible gap between their algorithms and a possibly better solution method. An important aspect is the tradeoff between the computing time and the solution quality.

Based on the problem presented in [105], the contributions of this paper are the following. (1) We present a complex bin-packing problem $RTLP$ faced by a real company. (2) We accurately measure the limitations of exact methods for $RTLP$. (3) We propose advanced metaheuristics to efficiently tackle $RTLP$, which perform better than the ones presented in [105], and we provide interesting features for future bin-packing methods. The obtained solutions indicate to Renault that it is possible to save interesting spaces for some trucks, which could result in the insertion of additional items for such trucks. (4) 597 instances are tackled to ensure the robustness of our methods (only 30 instances are considered in [105]). (5) We propose a global technique which conducts various types of solution methods to efficiently load several trucks according to a predefined computing time limit. Our analysis show the relevance of combining various types of heuristics/metaheuristics to achieve better performances.

The paper is organized as follows. In Section 2.2 is proposed a formal description of the problem and a MILP formulation for $RTLP$. In Section 2.3, a review of the literature concerning the bin-packing problems is depicted. In Section 2.4 are proposed various metaheuristics for $RTLP$, and the obtained results are discussed in Section 2.5. In Section 2.6 strategies are proposed to tackle a set of instances within a given time limit. Section 2.7 concludes the research and proposes some future works.

## 2.2 Description of the problem and MILP formulation

In this section, we start by formally describing $RTLP$. We then propose a MILP formulation.

### 2.2.1 Description of the problem

$RTLP$ can be formally described as follows. A number $n$ of rectangular items need to be placed in a truck (of width $W_t$ and length $L_t$). First, for each item $i$, we know its width $w_i$, its length $l_i$, and its class $C_j$, where $j \in \{1, \ldots, m\}$ with $m \leq n$. Therefore, it is possible to have more than one item of class $C_j$. The height of each item is supposed to respect the height of the truck. In addition, the classes must be placed in an increasing fashion from the front to the rear of the truck. More precisely, the abscissa of the origin item (the item which the lowest abscissa value) which belongs to class $C_j$ (label 1 on Figure 2.1) must be strictly smaller than the abscissa of the extremity of any item of class $C_{j+1}$ (label 2 on Figure 2.1), and such that the abscissa of the extremity item (the closest one to the rear) of class $C_m$ (label 3 on Figure 2.1), denoted as $f$, is minimized. The truck size is a hard constraint to fulfill, as it is not allowed to exceed neither its length nor its width.

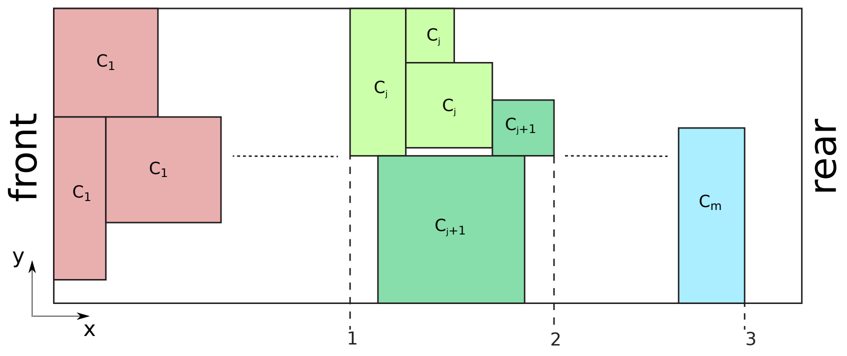

Figure 2.1: A possible solution for $RTLP$

Nowadays, instances of the standard bin-packing problem with up to 100 items can be tackled with exact methods and solvers based on linear programming (e.g., [84]). Unfortunately, $RTLP$ cannot be labeled as a standard bin-packing problem and exact methods are limited to much smaller instances. This is due to the large standard deviation among the sizes of the items (the items have widths or lengths of minimum 570mm and maximum 1810mm, with a standard deviation of 196), and to the class management constraint. In 2009, researchers from *Renault* stated in a conference presentation ([91]) that an exact method relying on a recent version of CPLEX is limited to 6 or 7 items for $RTLP$. In this paper, we first propose an integer linear program relying on CPLEX 12.5 able to solve instances with up to 13 items in about 7 hours on a powerful computer (i.e., Intel Quad-core i7 @ 3.4 GHz with 8 GB DDR3 of RAM memory). This limitation is a first motivation to design (meta)heuristics to tackle the real instances proposed by *Renault* (which contain up to 66 items). Moreover, as the truck loading plan has to be often rebuilt due to constant changes in the production plan, the need for fast and efficient algorithms is mandatory. Therefore, (meta)heuristics appear to be the most appropriate methods for $RTLP$.

### 2.2.2 MILP formulation

An exact model formulation for $RTLP$ is now proposed. The following notation is used: $x_i^l$ and $y_i^l$ are non-negative values representing the coordinates of the bottom left corner of item $i$, whereas $x_i^r$ and $y_i^r$ are non-negative values representing the coordinates of the top right corner. Notation $c_i$ corresponds to the class value $C_j$ of item $i$. $z_i$ is the binary value corresponding to the orientation of item $i$, where $z_i$=0 if the item is not rotated (according to the initially given orientation) and 1 otherwise. Moreover, we use a lower bound $f^{LB}$ of $f$ defined as $\lceil(\sum_n l_i \cdot w_i)/W_t\rceil$. Using the above variables and parameters, the problem can be formulated as the following mixed integer linear program:

minimize $f$, subject to the following constraints:

$$f \geq x_i^r \quad \forall i \tag{2.1}$$

$$x_i^r = x_i^l + (1 - z_i) \cdot w_i + (z_i) \cdot l_i \quad \forall i \tag{2.2}$$

$$y_i^r = y_i^l + (1 - z_i) \cdot l_i + (z_i) \cdot w_i \quad \forall i \tag{2.3}$$

$$x_i^r \leq x_j^l \vee x_j^r \leq x_i^l \vee y_i^r \leq y_j^l \vee y_j^r \leq y_i^l \quad \forall i, j \tag{2.4}$$

$$y_i^r \leq W_t \quad \forall i \tag{2.5}$$

$$x_i^l \leq x_j^r \quad \forall i, j \;\; s.t \;\; c_{i+1} = c_j \tag{2.6}$$

$$f \geq f^{LB} \tag{2.7}$$

$$x_i^l \geq 0, x_i^r \geq 0, y_i^l \geq 0, y_i^r \geq 0 \quad \forall i \tag{2.8}$$

$$z_i \in \{0, 1\} \quad \forall i \tag{2.9}$$

Constraints (2.1) ensure that $f$ is correctly computed as the maximum on all $i$ of value $x_i^r$. Constraints (2.2) and (2.3) ensure a correct computation of the coordinates of each item, whereas constraints (2.4) check that all $x$ and $y$ are getting consistent values. Constraints (2.5) ensure that $y_i^r$ is less than the truck width for all $i$, whereas constraints (2.6) check the class management constraint. Constraints (2.7) define $f^{LB}$ as a lower bound of $f$. Finally, constraints (2.8) forbid negative values for the decision variables.

## 2.3 Related literature

For exact methods, literature often proposes methods to tackle instances proposed in [9] or in [11]. The sizes of the instances are with up to 200 items. Each item has a width or a length in $\{1, \ldots, 100\}$ and a standard deviation of approximately 30 to 40.

If the number of items is less than a hundred, an exact branch-an-bound is proposed in [84] to tackle a problem where a set of $n$ rectangular pieces must be cut from an unlimited number of standardized stock pieces. This problem is a generalization of the 2D *bin-packing* problem (2D-BPP). More recently, in [83], an exact method for the *strip-packing* problem is proposed, solving instances with up to 200 items. The paper proposes a new relaxation, leading to a better lower bound, which is then used with a branch-and-bound algorithm to solve instances to optimality. Lower-bounds and an integer linear formulation for the 2D-BPP with up to 100 items are depicted in [99], where bins have variable sizes and costs, and the objective is to minimize the overall cost of bins used for packing the items. For the *perfect* packing problem, where there is no wasted space, [71] propose an exhaustive approach, relying on a branch-and-bound, able to tackle instance with up to 30 items. Finally, a mixed integer programming formulation for the *three-stages* 2D-BPP is proposed in [103]. In real applications, three stages often occur, where a stage is either a horizontal cut, a vertical cut, or a combination. Recent advances in the strip-packing problem can be found in [27], where significant improvements, regarding to existing algorithms for the strip-packing problem are exposed. Because of the specificity of $RTLP$ (i.e., class management constraint and important standard deviation among the sizes of the items), it is not possible to take advantage of the above exact algorithms to tackle it.

In the area of bin-packing, many researches focus on metaheuristics to tackle problems involving packing of items. Metaheuristics propose to find good solutions (in contrast with optimal solutions for exact methods) in a fair amount of time. A good reference book on metaheuristics can be found in [45], whereas [129] proposes general rules to improve metaheuristics performances according to various criteria and the kind of faced problem. A survey is proposed in [76] on heuristic and metaheuristic methods for the 2D-BPP, with greedy algorithms and a tabu search for four different problems (bin-packing with or without *rotation*, and with *guillotine* cutting required or not, where with guillotine cutting, only orthogonal cuts that bisect one component of the cutting material are allowed). Instances sizes vary from 10 up to 100. The goal of the tabu search is to empty a specific target bin $B$ by performing modifications, which consist in moving subsets of items from $B$ to other bins, where they can be rearranged. A tree-decomposition heuristic for the BPC-2D (2D bin-packing with *conflict*) is proposed in [67] for instances with up to 100 items, where a *conflict graph* $G = (V, E)$ is given, for which each vertex $j \in V$ represents an item, and there is an edge $[j, j'] \in E$ if items $j$ and $j'$ are incompatible. If two incompatible items are loaded in the same bin, a conflict occurs. The goal is to minimize the total number of bins, without creating any conflict. Simple heuristics and a tabu search are proposed to tackle this problem, where for the tabu search, the neighborhood structure consists in moving an item from a bin to another. A tabu search is proposed in [75] to tackle the problem of packing each item into a bin, such that the total number of required bins is minimized. A move also corresponds to relocate an item in the solution. Instances contain at most 100 items. The $TS^2PACK$ algorithm is proposed in [28]. It consists in a two-level tabu search for the 3D bin-packing problem (where the height of the bin has to be also considered). Whereas the first level aims at minimizing the total required number of bins, the second level consists in optimizing the packing of the bins. Instances sizes range from 50 to 200 items. Similarities between the bin-packing and the stock cutting problems are depicted in [74].

Recent advances have been made in genetic algorithms to tackle the 2D-BPP, where a population of solutions are combined and mutated to produce the offspring population. This process allows to diversify the search, and is usually combined with an intensification process, such as tabu search, to improve its performance. [15] proposes a genetic algorithm, where the solutions are not encoded. Instead, fully defined layouts are handled. The orientation is taken into account and extensive computational experiments show the effectiveness of the method. The crossover operator works in the following way: two parents are examined layer by layer, in a descending order, and the best parent layer is always transferred to the offspring. A genetic algorithm and a simulated annealing are described in [119]. Both approaches are compared and show that the genetic algorithm prevails on most of the instances. Here different types of crossover operators are used, namely *order-based crossover* (OBX), *cycle crossover* (CX), *order crossover* (OX), *partially matched crossover* (PMX), *uniform crossover* (UX) and *Stefan Jakobs crossover* (SJX). In the same area, [13] proposes a self-adaptive genetic algorithm and a self-adaptive parallel recombinative simulated annealing, and the genetic algorithm performs best in this case as well. The authors use four different crossover operators, and at each generation select one crossover operator to be performed. The four possible crossovers are: PMX, CX, *partially mapped crossover random locations* (PMXRL), and *preserve location crossover* (PLX). Other relevant works in this area can be found in [59, 66, 72, 73, 100].

The bin-packing problem presents many similarities with the *container loading problem.* However both problems are usually tackled by different communities of researchers. Thus, a careful study of the container loading problem literature has been performed. [42] proposes a tabu search for a routing and container loading problem, where moves consist in interchanging a pair of items between two different vehicles. In this problem, the three-dimensional container loading problem is considered, the loading process is divided into subproblems and is tackled with tabu search. In the same field, [39] proposes a parallelized genetic algorithm, but only a single container is considered. Crossover operators are here *uniform order-based crossover* (UOC), PMX, and CX. This paper was followed by [16], where the authors propose a parallel tabu search algorithm and expose improved results compared to the previous parallel genetic algorithm. The tabu search works on encoded solutions, and two neighbor solutions differ in the position of the items in the encoded solution. [80] are working as well with parallel algorithms, but an hybrid local search is used. A simulated annealing is combined with a tabu search, and results show high solution quality. They work as well with encoded solutions, and two solutions are neighbors if the sequences in which the items are packed differ from exactly one position. Other interesting works, involving complex techniques, can be found in [128, 98, 89].

## 2.4 Solution methods for $RTLP$

Different metaheuristics are now proposed to tackle $RTLP$, based on tabu search and genetic algorithms. All the proposed methods have a stopping condition of $T$ minutes (parameter).

### 2.4.1 Solution space structure

To tackle bin-packing problems, one can work either with *true* solutions or *encoded* solutions. A true solution directly represents a real loading of the items in the truck, which means that the position of each item has to be continuously known. Then, if an item is added to or removed from the truck, the new position of the loaded items is very hard to recompute, which is a major drawback. Working with encoded solutions require the use of a *decoding* algorithm to built the associated true solution and to get its value. Such an approach has the main advantage of being very flexible when adding (resp. removing) an item to (resp. from) the solution. Therefore, a decision has been made to work with *encoded* solutions. More formally, an encoded solution $s$ is a sequence of elements. To build a true solution $\hat{s}$ and compute its value $f(s)$, a *decoding greedy algorithm* ($DGA$) is performed on the encoded solution $s$. To drive $DGA$, information (some are mandatory and others are optional) are carried in each element of $s$, and can contain the item identifier ($ID$, mandatory), the class identifier ($\mathcal{C}$, mandatory), the item orientation ($O$, optional), and the item side ($S$, optional). Thus, component $i$ of the solution $s$ takes the form $s_i = (ID_i, \mathcal{C}_i, O_i, S_i)$, where $ID \in \{1, \ldots, n\}$, $\mathcal{C} \in \{1, \ldots, m\}$, $O \in \{\textit{not rotated}, 90^\circ\ \textit{rotated}\}$, and $S \in \{\textit{left-sided}, \textit{right-sided}\}$. Concerning $S$, left-sided enforces $DGA$ to try every possible position for the item, but only among the positions which are adjacent to the left side of the truck or to the left side of each already loaded item. $DGA$ thus decodes the vector $s$ into a real solution $\hat{s}$ (i.e., a true loading) by inserting in $\hat{s}$ the items from $s$ in a $FIFO$ order, and using the $O$ and $S$ information (if provided) while respecting the class management constraint. At each step, $DGA$ pops the next item $i$ of $s$ and greedily loads it in the truck, while respecting $\mathcal{C}_i$, $O_i$ and $S_i$. If $O_i$ (or $S_i$) is not provided, $DGA$ can decide by itself its value (minimizing the augmentation of $f$), and thus owns more freedom. For instances with $n = 60$, $DGA$ requires less than 0.1 second (on the used computer) to decode a solution into a real loading.

An example is now proposed. Five items, initially oriented as presented in Figure 2.2 (A), have to be placed in a truck. A possible encoded solution is the following: *s=((1,1,not rotated,right-sided), (3,1,*90° *rotated,right-sided), (2,1,*90° *rotated,left-sided), (5,2,?,?), (4,2,?,right-sided)).* The corresponding decoded solution $\hat{s}$ is illustrated in Figure 2.2 (B).

To generate this solution, $DGA$ performs the following steps: it pops the first element $ID_1$ and loads the corresponding item on the right side, without rotation. At that time, the next item, corresponding to $ID_3$, is loaded on the right with a 90° rotation. Then item $ID_2$ is loaded on the left with rotation, whereas item $ID_5$ is inserted at the best possible position tried by $DGA$ while respecting the class management constraint. Finally item $ID_4$ is inserted on the right side but $DGA$ decided its orientation. When $DGA$ is over, it returns the value $f$, which is in this example the extremity of item $ID_4$.

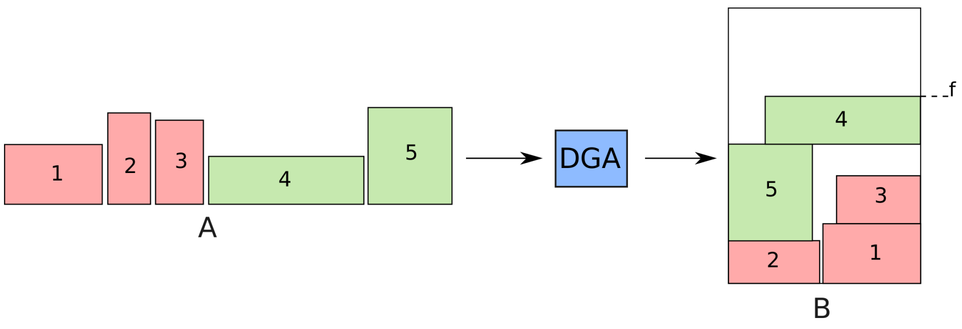

Figure 2.2: (A) Items to be loaded (with initial orientations). (B) Corresponding decoded solution.

Four different local search algorithms are proposed in [105]. They differ in the sense that for the first version (called TS1), only the information $ID$ and $\mathcal{C}$ are contained in each element of the encoded solution. The second version (called TS2) contains $ID$, $\mathcal{C}$ and $O$. The third version (called TS3) contains $ID$, $\mathcal{C}$ and $S$. The fourth version (called TS4) contains $ID$, $\mathcal{C}$, $O$ and $S$. Thus, for the first version, $DGA$ can decide on its own the orientations and the sides (while focusing on the smallest augmentation of $f$). In the second and third version, $DGA$ has the sequence in which the insertions must be made, and the orientation or the side of each item (but not both). Finally, the fourth version constraints $DGA$ at the maximum due to the complete information set contained in each component $s_i$ of the vector $s$. Preliminary experiments (detailed in [105]) have be made on a representative set of 30 instances. For each instance and each version of tabu search was computed the average percentage gap with respect to the best ever obtained solution. TS1, TS2, TS3 and TS4 have respectively the following average gaps: 8.01%, 0.00%, 0.21% and 0.32%. This clearly shows that the most important information to be carried to $DGA$ is the one considered in TS2 (with $ID$, $\mathcal{C}$, and $O$), whereas providing full freedom to $DGA$ (as in TS1) results in poor results. As a consequence, in this paper, we decided to only focus on the solution space structure relying on $ID$, $\mathcal{C}$ and $O$.

### 2.4.2 Tabu search

A local search method starts from an initial solution, and at each iteration explores the neighborhood $N(s)$ of the current solution $s$ to reach a possibly better solution. The *neighborhood* of a solution $s$ is the set of solutions reachable from $s$ by performing a specific modification (called a *move*) on $s$. The neighborhood exploration continues until a stopping criterion is met (usually a number of iterations or a computing time limit), and the best found solution is returned to the user. In a descent local search, the best move is performed at each iteration, and the algorithm stops when a local optimum is reached. Tabu search is a local search with the following specificities: when a move is performed, the reverse move is forbidden (tabu) for *tab* (parameter) iterations. The best non tabu move is usually performed at each iteration.

For $RTLP$, the seven following moves can be proposed:

(1) move an item $j$ from position $s_i$ to position $s_{i'}$ is $s$;

(2) move an item $j$ from position $s_i$ to position $s_{i'}$ is $s$ while switching to the opposite value its orientation $O$;

(3) move an item $j$ from position $s_i$ to position $s_{i'}$ is $s$ while switching to the opposite value its side $S$;

(4) move an item $j$ from position $s_i$ to position $s_{i'}$ is $s$ while switching $S$ and $O$ to the opposite values;

(5) for an item $j$, switch $O$ to the opposite value;

(6) for an item $j$, switch $S$ to the opposite value;

(7) for an item $j$, switch $O$ and $S$ to the opposite values.

As mentioned in Section 2.4.1, the solution space considering $ID$, $\mathcal{C}$ and $O$ is the most powerful. Therefore, in the proposed tabu search $TS$, only the moves considering moving an item in the encoded solution and switching $O$ are considered at each iteration, while respecting the class management constraint (ties are broken randomly). In other words, only moves of type (1), (2) and (5) are considered in $TS$ (because the other moves are associated with less efficient solution space structures).

The initial solution used by $TS$ is found in the following way: for a time $T' = \lambda \cdot T$ (where $\lambda$ is a parameter tuned to 10% with $\lambda$ tested in interval $[0\%, 60\%]$), a random solution is generated and then improved with a descent local search. As long as $T'$ is not reached, restarts are performed. When $T'$ is reached, the best found solution is returned and is used as the initial solution for $TS$. This initialization operator is called $INIT$ in the following.

In tabu search, it is common that, at each iteration, only a random fraction $\upsilon$ of the possible moves are generated. This restriction in the neighborhood size allows to perform more iterations for a same time limit, and helps in bringing more diversification. At each iteration, the tabu tenure *tab* is set to a uniformly distributed value between $a$ and $b$ (parameters). Preliminary experiments shows that the following parameter setting is appropriate for the considered instances: $(\upsilon,\ a,\ b) = (50\%,\ 25,\ 55)$, with $\upsilon$ tested in interval $[1\%, 100\%]$, whereas $a$ and $b$ were tested in interval $[5, 200]$. Note that more refined strategies were tested for the neighborhood size and the tabu durations, but without leading to better results. Thus, the simpler settings are favored. This observation holds for all the parameters described in this paper.

The following diversification mechanism is also proposed. Let $s^\star$ be the best encountered solution during the search process. When a number $\gamma$ (parameter tuned to 300, with $\gamma$ tested in interval $[50, 1000]$) of iterations without improving $s^\star$ is reached, a diversification procedure $DIV$ is triggered, which consists of reversing the encoded solution, while respecting the class management constraint (i.e., for each class, the sequence of the items is reversed). This simple process allows big jumps in the solution space.

### 2.4.3 Genetic algorithm

In contrast with local search methods, genetic algorithms maintain a population $\mathcal{M}$ of solutions, instead of working on a single solution. As with human genetic, where chromosomes are used to encode our DNA, a genetic algorithm manages a set of chromosomes (solutions), and each chromosome is a list of genes (variables). At each generation of the algorithm, the complete population is renewed with $|\mathcal{M}|$ offspring solutions. To generate a new offspring solution, two parent solutions are selected, based on their ability to produce a good offspring solution. An offspring is then generated by recombining the two parent solutions. A mutation operator, which introduces random modifications to the offspring, is triggered after the recombination methods to ensure that the offspring is different enough compared to its parents. Finally the offspring is added to the new population. Thus, to design a genetic algorithm, one has to carefully define the following three different operators: the parent selection process, the recombination operator, and the mutation. At the end of a generation, the new population replaces the parent population, and the algorithm continues the same process until a stopping criterion is met.

To introduce the genetic algorithm for $RTLP$ (denoted $GA$), we first define the *distance* between two solutions $s_1$ and $s_2$ as

$$d(s_1, s_2) = \sum_{i=1}^{n} \left| s_1^L(i) - s_2^L(i) \right| + \sum_{i=1}^{n} \left| s_1^W(i) - s_2^W(i) \right| \tag{2.10}$$

where $s_j^W(i)$ represents the width (measured according to the front-rear axis) of the item $i$ of solution $s_j$, and $s_j^L(i)$ the corresponding length. This distance function is easy to compute and allows to measure the difference of shape between two true solutions, which is more relevant than to measure the structural difference between two encoded solutions. Figure 2.3 proposes an illustration. For the true solutions (A) and (B), the associated encoded solutions could be the same following one: $(1, 2, 3, 4)$. In other words, even if the loading sequences of (A) and (B) are the same, the resulting loadings are very different, and this difference can be accurately measured by the proposed distance function. In contrast, the true solutions (A) and (C) have similar shapes even if the encoded solutions could respectively be $(1, 2, 3, 4)$ and $(4, 1, 3, 2)$. Actually, the distance between (A) and (C) is 0, considering that items 1, 2 and 4 have the same dimensions. In other words, even if the loading sequences of (A) and (C) are very different, the resulting shapes of the loadings could be the same, and this similarity is perfectly detected by the proposed distance function.

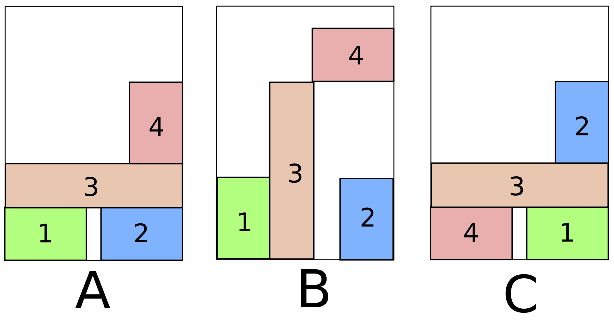

Figure 2.3: Illustration of the distance between solutions

In $GA$, $N$ (parameter tuned to 10, with $N$ tested in interval $[1, 100]$) initial solutions are first generated similarly to $INIT$. For a time $T' = \lambda \cdot T$ (where $\lambda$ is a parameter tuned to 10% with $\lambda$ tested in interval $[0\%, 60\%]$), a random solution is generated and then improved with a descent local search. As long as $T'$ is not reached, restarts are performed. When $T'$ is reached, the $N$ best visited solutions are selected for the second phase, which consists of improving each solution with $I$ (parameter tuned to 100, with $I$ tested in interval $[20, 600]$) iterations of $TS$. Finally the $N$ resulting solutions are inserted in $\mathcal{M}$.

As long as $T$ is not reached, the selection operator intends to select the two most interesting solutions in $\mathcal{M}$. The first solution $s_1$ is stochastically selected according to its rank in $\mathcal{M}$, assuming that the solutions in $\mathcal{M}$ are ranked from the best to the worst (according to $f$). Thus, the best solution has $|\mathcal{M}|$ times more chance to be selected than the worst, whereas the second best has $|\mathcal{M}|$-1 times more chance, etc. The second solution $s_2$ is then selected as the one having the largest distance with $s_1$ (according to Equation (2.10)). This technique is likely to generate an offspring solution different enough from the solutions in $\mathcal{M}$.

The recombination operator, denoted $RECOMB$, works as follows to produce the offspring solution $\bar{s}$ from two parent encoded solutions $s_1$ and $s_2$.
If $m = 1$, half of each parent solution is given to $\bar{s}$ as follows: each parent encoded

solution is split in two, and the operator randomly selects which parent gives the first part. The other parent then gives the missing items in the order they appear in it.
If $m > 1$, then the operator works in two distinct phases. In the first phase, namely the *constructive phase*, it selects the next block (a complete class in the *constructive phase*, then a part of a class in the *reconstructive phase*) belonging to a class randomly selected from a parent and inserts it in $\bar{s}$, until all the classes are filled in $\bar{s}$. After the initial *constructive phase* and any subsequent *reconstructive phase*, a check is performed, and as long as each parent does not give at least 40% and at most 60% of information to $\bar{s}$, a *reconstructive phase* is performed as follows. The largest block of $\bar{s}$ from the parent having provided the largest number of items is split in two, then one part of the block is given from a parent (randomly selected) and the other part is given by the other parent. This process allows to ensure that a fair amount of information is transmitted to $\bar{s}$ by each parent. Of course the procedure forbids duplicate items in the offspring by selecting only the items that are not already present in $\bar{s}$.

Figure 2.4 proposes an illustration of the $RECOMB$ operator. Nine items, with two different classes, have to be packed. The first class contains two items (labeled 1 and 2), whereas the second class contains seven items. In the first phase, it randomly selects which solution provides class 1 (here $s_1$). Then $s_2$ provides the second class. After this *constructive phase*, a check is performed and it results that $s_1$ gives 22.2% of its solution, whereas $s_2$ gives 77.7% of its solution. The algorithm then splits in two the block which has the largest number of items (here the corresponding block is the complete second class, with items 7, 9, 4, 5, 6, 8, 3), and each part of the block is provided by the two different solutions. Then, the first part of the second class is provided by $s_2$ (items 7, 9 and 4), whereas the second part is provided by $s_1$ (the missing items are 5, 6, 8, 3, but they appear as 3, 5, 6, 8 in $s_1$). Again, a check is performed, and $s_1$ gives 66.6% of its solution, whereas $s_2$ provides 33.3% of its solution. The last block, provided by $s_1$ (containing items 3, 5, 6, 8) for class 2, is the biggest block, thus the algorithm splits it in two. After the third step, $s_1$ provides 44.4% of its solution, whereas $s_2$ provides 55.5% of its solution. The algorithm then stops and returns $\bar{s}$, as each parent provides at least 40% and at most 60% of data from its solution.

In the most efficient genetic algorithms currently used to tackle combinatorial

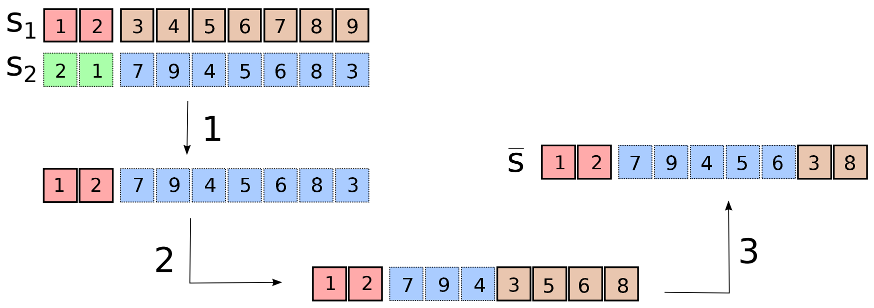

Figure 2.4: Illustration of the recombination operator of *GA*

problems, called hybrid genetic algorithms, the mutation operator is often performed by a local search method in order to improve the offspring solution. In *GA*, *TS* without diversification is used as the underlying local search method. Thus, after the generation of the offspring solution $\bar{s}$, *TS* is used to improve $\bar{s}$ for $I$ iterations.

In *GA*, only one offspring $\bar{s}$ is provided at each generation, and is used to replace only one solution in the population. The worst solution $s_{worst}$ of $\mathcal{M}$ is replaced with $\bar{s}$ (the worst solution being the one with the longest loading). But if $\bar{s}$ is worse than $s_{worst}$, then the solution $s_{close}$ having the minimum distance value with the set $\mathcal{M} \setminus s_{close}$ is replaced with $\bar{s}$. More precisely $s_{close}$ minimizes

$$D(s) = \frac{1}{|\mathcal{M}| - 1} \sum_{s' \in \mathcal{M} \setminus s} d(s, s') \tag{2.11}$$

where $d(s, s')$ is defined according to Equation (2.10). This technique allows to add more diversity in $\mathcal{M}$.

The pseudo-code of *GA* is given in Algorithm 4.

## 2.5 Results

In this section are compared all the methods presented in this paper on a set of 40 real benchmark instances provided by *Renault*. Note that a total

**Algorithm 4** Genetic algorithm $GA$ proposed in this paper

```
Set $\mathcal{M} = \emptyset$

While $|\mathcal{M}| < N$, do
    1. Generate a solution $s$ using $INIT$
    2. Perform $I$ iterations of $TS$ on $s$
    3. Put $s$ in $\mathcal{M}$
While the time limit $T$ is not reached, do
    1. Select two parents based on the selection operator
    2. Perform the recombination operator to produce $\bar{s}$
    3. Perform $I$ iterations of $TS$ on $\bar{s}$
    4. If $f(\bar{s}) < f(s_{worst})$, replace $s_{worst}$ with $\bar{s}$. Otherwise, replace $s_{close}$ with $\bar{s}$.
Return the best visited solution $s^\star$
```

of 597 instances exist (with the same difficulty), and are available on `http://operations-research.unige.ch/downloads.html`. Tests were performed on an Intel Quad-core i7 @ 3.4 GHz with 8 GB DDR3 of RAM memory. The time limit $T$ is set to 3600 seconds. Such a time limit was confirmed to be a relevant value, as the goal of this project is to benchmark the *Renault* algorithms by finding the best possible solutions. Thus, the computation time is not a hard constraint in this case. As tabu search and genetic algorithm are not methods with restarts, their results are averaged over ten runs. For analysis purposes, the 40 instances are split in five different groups in the result tables. Let $Z$ be the full set of 597 instances provided by *Renault*. We decided to consider 40 representative instances of $Z$ which are partitioned into 4 instances types, ranging from (A) to (D), and described below:

(A) large $m$ values ($m \geq 6$), given that the average $m$ value in $Z$ is 2.36;

(B) medium $m$ values ($m = 4$);

(C) $m = 1$;

(D) instances for which *Renault* was not able to find a feasible solution prior to this project.

Table 2.1 summarizes the fourteen different instances. For each instance are given the truck identifier ID, the number $n$ of items, the number $m$ of classes, and the type to which the instance belongs.

Table 2.1: Presentation of the forty instances

| **ID** | $n$ | $m$ | **Type** |
|---|---|---|---|
| 1 | 26 | 7 | (A) |
| 2 | 36 | 7 | (A) |
| 3 | 37 | 7 | (A) |
| 4 | 27 | 6 | (A) |
| 5 | 25 | 7 | (A) |
| 6 | 20 | 7 | (A) |
| 7 | 26 | 7 | (A) |
| 8 | 30 | 9 | (A) |
| 9 | 38 | 9 | (A) |
| 10 | 21 | 7 | (A) |
| 11 | 18 | 4 | (B) |
| 12 | 20 | 4 | (B) |
| 13 | 13 | 4 | (B) |
| 14 | 23 | 4 | (B) |
| 15 | 37 | 4 | (B) |
| 16 | 25 | 4 | (B) |
| 17 | 36 | 4 | (B) |
| 18 | 33 | 4 | (B) |
| 19 | 32 | 4 | (B) |
| 20 | 30 | 4 | (B) |
| 21 | 16 | 1 | (C) |
| 22 | 16 | 1 | (C) |
| 23 | 19 | 1 | (C) |
| 24 | 26 | 1 | (C) |
| 25 | 44 | 1 | (C) |
| 26 | 18 | 1 | (C) |
| 27 | 11 | 1 | (C) |
| 28 | 14 | 1 | (C) |
| 29 | 22 | 1 | (C) |
| 30 | 17 | 1 | (C) |
| 31 | 38 | 2 | (D) |
| 32 | 34 | 3 | (D) |
| 33 | 42 | 2 | (D) |
| 34 | 31 | 3 | (D) |
| 35 | 25 | 3 | (D) |
| 36 | 27 | 2 | (D) |
| 37 | 29 | 2 | (D) |
| 38 | 31 | 2 | (D) |
| 39 | 35 | 3 | (D) |
| 40 | 37 | 1 | (D) |

*Renault* proposes the two following greedy heuristics to tackle *RTLP*, denoted *SG* (for Simple Greedy) and *LAG* (for Look-Ahead Greedy). *SG* builds a solution from scratch, and at each iteration, selects an item (following different possible rules) from a list $A$ of non already inserted items, and adds it to the solution at minimum $f$ value (i.e., which minimizes the augmentation of $f$, label 3 of Figure 2.1). This process stops when $A$ is empty. In $LAG$, at each iteration, the algorithm tries each item $j$ of $A$, and for each $j$, tries the next $p$ (parameter tuned to 5, with $p$ tested in interval $[1, 10]$ insertions following this possible insertion of $j$ (look-ahead process). At the end of the iteration, the item $j$ that would involve the lowest cost after the next $p$ iterations is selected and inserted at the best position. As before, this process continues until $A$ becomes empty. Both processes are fast (a few seconds per run), and therefore relevant to *Renault* as they are dealing with more than a thousand trucks to operate on a daily basis. As [105] show that $LAG$ is the most efficient greedy heuristic, only the latter is used for comparison purposes, and it performs restarts as long as the time limit $T$ is not reached. When $T$ is reached, the best generated solution is returned.

Table 2.2: Results on the (A) instances

| **ID** | $f^\star$ | $LAG$ | **gap** | $TS$ | **gap** | **time** | $TS + DIV$ | **gap** | **time** | $GA$ | **gap** | **time** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 11,820 | **11,820** | 0.00% | 11,939.20 | 1.01% | 1,515.70 | 11,873.00 | 0.45% | 936.57 | 11,872.20 | 0.44% | 641.96 |
| 2 | 13,510 | 13,670 | 1.18% | **13,510.00** | 0.00% | 444.93 | **13,510.00** | 0.00% | 39.60 | **13,510.00** | 0.00% | 69.12 |
| 3 | 13,950 | 14,090 | 1.00% | 13,974.00 | 0.17% | 1,267.58 | 13,964.00 | 0.10% | 1,685.37 | **13,950.00** | 0.00% | 414.23 |
| 4 | 13,048 | 13,068 | 0.15% | 13,074.20 | 0.20% | 1,475.82 | 13,051.00 | 0.02% | 1,161.32 | **13,048.00** | 0.00% | 171.13 |
| 5 | 13,372 | **13,372** | 0.00% | **13,372.00** | 0.00% | 721.31 | **13,372.00** | 0.00% | 36.14 | **13,372.00** | 0.00% | 736.85 |
| 6 | 13,192 | 13,230 | 0.29% | 13,208.20 | 0.12% | 1,011.17 | **13,192.00** | 0.00% | 535.09 | **13,192.00** | 0.00% | 156.91 |
| 7 | 13,012 | **13,012** | 0.00% | 13,111.00 | 0.76% | 1,049.59 | 13,110.00 | 0.75% | 29.11 | 13,110.00 | 0.75% | 0.00 |
| 8 | 13,950 | **13,950** | 0.00% | 13,998.40 | 0.35% | 2,249.02 | 13,965.00 | 0.11% | 306.06 | 13,967.00 | 0.12% | 1,273.15 |
| 9 | 13,110 | 13,170 | 0.46% | 13,189.00 | 0.60% | 948.03 | 13,170.00 | 0.46% | 371.21 | **13,140.00** | 0.23% | 1,710.92 |
| 10 | 14,290 | 14,310 | 0.14% | 14,298.00 | 0.06% | 1,855.08 | **14,290.00** | 0.00% | 178.48 | **14,290.00** | 0.00% | 131.61 |
| **AVG** | | | 0.32% | | 0.33% | 1,253.82 | | 0.19% | 527.89 | | 0.15% | 530.59 |

Table 2.2 shows the results on the (A) instances. The first column (ID) gives the instances identifier. The second column ($f^\star$) provides the best found solution returned by any of the algorithms (i.e., the best found solution can be achieved on a run of an algorithm, but its average could be higher, thus the value of $f^\star$ can be lower than all the results provided for this instance). Column three and four respectively gives the best results returned by $LAG$ and the corresponding percentage gap with $f^\star$. The next three columns provide the average (over ten runs) result found by $TS$, the corresponding gap with $f^\star$, and finally the average time (in seconds) to reach the best solution. The next columns provide the same information for the other methods ($TS$ with diversification and $GA$). The last row provides averaged values for each corresponding gap and time. For $TS$, $TS + DIV$ and $GA$, a time lower than $T'$ indicates that the best solution

was found while performing $INIT$. Note that no time is given for $LAG$ as it is a restarting procedure. Thus, its best solution within $T$ seconds can be generated at any time. On average, $GA$ tends to perform best on this kind of instances, not far from $TS + DIV$. $LAG$ and $TS$ show similar performances, but $TS$ needs more time to reach its best solution compared to $TS + DIV$ and $GA$. The diversification ability associated with $GA$ seems to pay back, as it gives very good results with a competitive computing time. The diversification mechanism of $TS$ also shows interesting improvements.

Table 2.3: Results on the (B) instances

| **ID** | $f^\star$ | $LAG$ | **gap** | $TS$ | **gap** | **time** | $TS + DIV$ | **gap** | **time** | $GA$ | **gap** | **time** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 11 | 13,295 | **13,295** | 0.00% | **13,295.00** | 0.00% | 1,534.66 | **13,295.00** | 0.00% | 4.65 | **13,295.00** | 0.00% | 23.19 |
| 12 | 13,325 | **13,325** | 0.00% | **13,325.00** | 0.00% | 650.61 | **13,325.00** | 0.00% | 11.92 | **13,325.00** | 0.00% | 12.98 |
| 13 | 12,600 | **12,600** | 0.00% | **12,600.00** | 0.00% | 1.01 | **12,600.00** | 0.00% | 0.14 | **12,600.00** | 0.00% | 0.00 |
| 14 | 12,902 | 12,916 | 0.11% | 12,912.20 | 0.08% | 1,652.50 | **12,904.00** | 0.02% | 941.47 | 12,906.00 | 0.03% | 573.16 |
| 15 | 13,660 | **13,660** | 0.00% | 13,719.60 | 0.44% | 1,335.28 | 13,716.00 | 0.41% | 765.21 | 13,716.00 | 0.41% | 1,152.94 |
| 16 | 12,350 | **12,350** | 0.00% | 12,390.40 | 0.33% | 1,534.97 | **12,350.00** | 0.00% | 653.06 | **12,350.00** | 0.00% | 99.43 |
| 17 | 13,160 | **13,160** | 0.00% | 13,232.00 | 0.55% | 1,194.06 | 13,220.00 | 0.46% | 676.90 | 13,208.00 | 0.36% | 1,037.55 |
| 18 | 12,590 | 12,642 | 0.41% | 12,614.00 | 0.19% | 1,739.00 | **12,590.00** | 0.00% | 797.56 | 12,637.00 | 0.37% | 527.58 |
| 19 | 14,040 | **14,040** | 0.00% | **14,040.00** | 0.00% | 1,005.18 | **14,040.00** | 0.00% | 115.69 | **14,040.00** | 0.00% | 120.15 |
| 20 | 13,010 | **13,010** | 0.00% | 13,037.00 | 0.21% | 554.08 | **13,010.00** | 0.00% | 1,337.56 | **13,010.00** | 0.00% | 381.74 |
| **AVG** | | | 0.05% | | 0.18% | 1,120.13 | | 0.09% | 530.42 | | 0.12% | 392.87 |

Table 2.3 shows the results on the (B) instances. $LAG$ shows similar performances compared to more sophisticated metaheuristics, such as $TS + DIV$. Here, the look-ahead mode seems to provide enough knowledge of the impacts of each insertion, and provide interesting results. $GA$ seems to obtain promising results (e.g., close to the values of $LAG$), but sees its average gap value raising only because of instances 15, 17, and 18. These three instances are the ones with an important number of items. It tends to prove that $TS$, $TS + DIV$ and $GA$ are in general less competitive when the number of items is raising up (but exceptions can occur, see for example instances 2 and 3).

For all the methods, Table 2.4 shows similar average performances on the (C) instances. The only instance without a zero-value gap is the number 25, which is the instance with the biggest number of items (44 items) in this set. For this instance, $TS$, $TS + DIV$ and $GA$ show the same gap, which are approximately half the gap of $LAG$. On the one hand, these results tend to show that with a large number of items, the performances of the algorithms decrease when the number of items is important. On the other hand, these results show that with a single class, all the algorithms are competitive. Note however that $GA$ is much

Table 2.4: Results on the (C) instances

| ID | $f^{\star}$ | $LAG$ | gap | $TS$ | gap | time | $TS + DIV$ | gap | time | $GA$ | gap | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 12,675 | **12,675** | 0.00% | **12,675.00** | 0.00% | 111.81 | **12,675.00** | 0.00% | 0.55 | **12,675.00** | 0.00% | 0.00 |
| 22 | 13,370 | **13,370** | 0.00% | **13,370.00** | 0.00% | 0.18 | **13,370.00** | 0.00% | 0.24 | **13,370.00** | 0.00% | 0.00 |
| 23 | 12,800 | **12,800** | 0.00% | **12,800.00** | 0.00% | 524.33 | **12,800.00** | 0.00% | 4.86 | **12,800.00** | 0.00% | 0.00 |
| 24 | 14,060 | **14,060** | 0.00% | **14,060.00** | 0.00% | 219.81 | **14,060.00** | 0.00% | 4.92 | **14,060.00** | 0.00% | 0.00 |
| 25 | 13,414 | 13,544 | 0.97% | **13,473.20** | 0.44% | 1,048.02 | 13,475.20 | 0.46% | 1,220.49 | 13,474.00 | 0.45% | 0.00 |
| 26 | 12,634 | **12,634** | 0.00% | **12,634.00** | 0.00% | 242.20 | **12,634.00** | 0.00% | 30.21 | **12,634.00** | 0.00% | 58.03 |
| 27 | 13,300 | **13,300** | 0.00% | **13,300.00** | 0.00% | 0.02 | **13,300.00** | 0.00% | 0.02 | **13,300.00** | 0.00% | 0.00 |
| 28 | 13,792 | **13,792** | 0.00% | **13,792.00** | 0.00% | 0.79 | **13,792.00** | 0.00% | 0.62 | **13,792.00** | 0.00% | 0.00 |
| 29 | 12,530 | **12,530** | 0.00% | 12,531.00 | 0.01% | 814.99 | **12,530.00** | 0.00% | 2.02 | **12,530.00** | 0.00% | 0.00 |
| 30 | 12,975 | **12,975** | 0.00% | **12,975.00** | 0.00% | 38.51 | **12,975.00** | 0.00% | 1.28 | **12,975.00** | 0.00% | 0.00 |
| **AVG** | | | 0.10% | | 0.04% | 300.07 | | 0.05% | 126.52 | | 0.04% | 5.80 |

quicker on these instances, with an average time of 5.8. Note that instance 26 has an average time much larger than the other ones, which can be explained because of its much larger standard deviation over the dimensions of the items (which is 485, versus 330 for the other instances of this set). Thus, instance 26 is more challenging to load in an efficient way. Again, the diversity provided by the population seems to pay off, as the average times are negligible on the other nine instances (a value of 0.00 indicates that the best value was found during $INIT$, and very small time values were rounded to 0.00 due to the two significant digits).

Table 2.5: Results on the (D) instances

| ID | $f^{\star}$ | $LAG$ | gap | $TS$ | gap | time | $TS + DIV$ | gap | time | $GA$ | gap | time |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 13,220 | 13,304 | 0.64% | 13,317.90 | 0.74% | 2,021.19 | **13,277.00** | 0.43% | 2,255.11 | 13,297.40 | 0.59% | 3,036.93 |
| 32 | 13,416 | 13,518 | 0.76% | 13,498.30 | 0.61% | 2,070.64 | 13,462.80 | 0.35% | 1,813.51 | **13,439.20** | 0.17% | 1,922.99 |
| 33 | 13,650 | 13,780 | 0.95% | 13,712.00 | 0.45% | 1,538.84 | 13,695.00 | 0.33% | 1,671.72 | **13,674.00** | 0.18% | 1,693.30 |
| 34 | 13,299 | 13,335 | 0.27% | 13,318.00 | 0.14% | 1,613.95 | 13,306.00 | 0.05% | 1,398.05 | **13,303.20** | 0.03% | 706.30 |
| 35 | 13,684 | **13,684** | 0.00% | **13,684.00** | 0.00% | 829.54 | **13,684.00** | 0.00% | 193.45 | **13,684.00** | 0.00% | 290.08 |
| 36 | 14,000 | **14,000** | 0.00% | **14,000.00** | 0.00% | 1.88 | **14,000.00** | 0.00% | 2.98 | **14,000.00** | 0.00% | 0.00 |
| 37 | 13,388 | 13,544 | 1.17% | 13,497.30 | 0.82% | 1,768.86 | 13,458.30 | 0.53% | 2,021.53 | **13,407.80** | 0.15% | 1,109.25 |
| 38 | 13,291 | 13,372 | 0.61% | 13,328.10 | 0.28% | 1,338.26 | 13,320.00 | 0.22% | 2,180.31 | **13,296.60** | 0.04% | 1,041.22 |
| 39 | 13,380 | **13,380** | 0.00% | 13,413.00 | 0.25% | 2,057.55 | 13,427.20 | 0.35% | 2,090.85 | 13,413.00 | 0.25% | 1,820.81 |
| 40 | 13,318 | 13,410 | 0.69% | 13,391.30 | 0.55% | 1,802.11 | **13,356.00** | 0.29% | 1,846.37 | 13,398.80 | 0.61% | 2,223.43 |
| **AVG** | | | 0.51% | | 0.38% | 1,504.28 | | 0.25% | 1,547.39 | | 0.20% | 1,384.43 |

The results on the (D) instances are presented in Table 2.5. The testbed used in this paper has greatly more computation power compared to the testbed previously used by *Renault*, and it is interesting to see that some instances can now be solved by $LAG$. This instance set was selected to measure the improvement potential of advanced metaheuristics (namely $TS$, $TS + DIV$ and $GA$) over more simpler heuristics (such as $LAG$). Note that the truck length is usually of 13,400mm, which is the case of instance 40. For this instance, $LAG$ was not able to solve the instance in one hour. On the opposite, $TS$, $TS + DIV$ and $GA$ were all three able to find a feasible solution. Thus, sophisticated metaheuristics could be tried when $LAG$ is close but not able to solve an instance. The instances included in this set have a relatively low value for $m$ but interestingly, high values for $n$ (which is usually a sign of poor results for the metaheuristics presented here). Even here, the results provided show very good performances for $TS + DIV$ and $GA$ for comparable required computation times.

Table 2.6: Average performances of the algorithms

| | $LAG$ | $TS$ | $TS + DIV$ | $GA$ |
|---|---|---|---|---|
| Average gaps on the 40 instances set | 0.25% | 0.23% | 0.14% | 0.13% |
| Average gaps on the 597 instances set | 0.19% | 0.23% | 0.13% | 0.09% |
| Average times on the 40 instances set | | 1,044.58 | 683.05 | 578.42 |
| Average times on the 597 instances set | | 855.22 | 635.87 | 574.82 |
| Number of solved instances on the 597 instances set | 350 | 347 | 347 | 350 |

To summarize the performances of the algorithms, Table 2.6 presents the average gaps and times of the algorithms on the set of 40 instances and on the full set of 597 instances. On the set of 40 instances, averages are calculated based on the results of Tables 2.2, 2.3, 2.4, and 2.5. On the set of 597 instances, results are presented with a single one-hour run per instance. Not surprisingly, $GA$ shows the best average gap, followed by $TS + DIV$, $LAG$ and $TS$. The four algorithms provide similar results when comparing the 597 instances set and the 40 instances set, which show the relevance of the 40 instances set for benchmarking purposes. Note that $GA$ requires as well the smallest average times to perform best. This method is then well advised both for performance and time constraints. $TS + DIV$ proves that the diversification procedure, even if it is really simple to implement, improves significantly the performances of $TS$. $TS$ shows on average similar performances to $LAG$, and is only overpowered by $LAG$ when dealing with the (B) instances. The last line provides the number of feasible loadings for each algorithm (i.e., instances for which $f \leq 13{,}400$, which means that all the items fit in the truck). Note that the number of solved instances for $LAG$ and $GA$ are equivalent, even if they do not solve the same instances on the complete set. These results are used in Section 2.6 to show that even if they are both able to solve the same amount of instances, a relevant combination of different algorithms leads to better results.

Table 2.7: Exact method vs (meta)heuristics

| **ID** | **OPL** | $LAG$ | $TS$ | $TS+DIV$ | $GA$ |
|---|---|---|---|---|---|
| 1 | 11,860 | 0.34% | -0.67% | -0.11% | -0.10% |
| 2 | 13,510 | -1.18% | 0.00% | 0.00% | 0.00% |
| 3 | NaN | NaN | NaN | NaN | NaN |
| 4 | 13,058 | -0.08% | -0.12% | 0.05% | 0.08% |
| 5 | 13,352 | -0.15% | -0.15% | -0.15% | -0.15% |
| 6 | 12,870 | -2.80% | -2.63% | -2.50% | -2.50% |
| 7 | 13,050 | 0.29% | -0.47% | -0.46% | -0.46% |
| 8 | NaN | NaN | NaN | NaN | NaN |
| 9 | 13,170 | 0.00% | -0.14% | 0.00% | 0.23% |
| 10 | 14,300 | -0.07% | 0.01% | 0.07% | 0.07% |
| 11 | 13,295 | 0.00% | 0.00% | 0.00% | 0.00% |
| 12 | 13,325 | 0.00% | 0.00% | 0.00% | 0.00% |
| 13 | 12,600 | 0.00% | 0.00% | 0.00% | 0.00% |
| 14 | 12,900 | -0.12% | -0.09% | -0.03% | -0.05% |
| 15 | NaN | NaN | NaN | NaN | NaN |
| 16 | 12,350 | 0.00% | -0.33% | 0.00% | 0.00% |
| 17 | NaN | NaN | NaN | NaN | NaN |
| 18 | 12,636 | -0.05% | 0.17% | 0.36% | -0.01% |
| 19 | NaN | NaN | NaN | NaN | NaN |
| 20 | 13,040 | 0.23% | 0.02% | 0.23% | 0.23% |
| 21 | 12,675 | 0.00% | 0.00% | 0.00% | 0.00% |
| 22 | 13,370 | 0.00% | 0.00% | 0.00% | 0.00% |
| 23 | 12,800 | 0.00% | 0.00% | 0.00% | 0.00% |
| 24 | 14,090 | 0.21% | 0.21% | 0.21% | 0.21% |
| 25 | NaN | NaN | NaN | NaN | NaN |
| 26 | 12,634 | 0.00% | 0.00% | 0.00% | 0.00% |
| 27 | 13,300 | 0.00% | 0.00% | 0.00% | 0.00% |
| 28 | 13,792 | 0.00% | 0.00% | 0.00% | 0.00% |
| 29 | 12,530 | 0.00% | -0.01% | 0.00% | 0.00% |
| 30 | 12,975 | 0.00% | 0.00% | 0.00% | 0.00% |
| 31 | 13,500 | 1.45% | 1.35% | 1.65% | 1.50% |
| 32 | 13,476 | -0.31% | -0.17% | 0.10% | 0.27% |
| 33 | NaN | NaN | NaN | NaN | NaN |
| 34 | 13,406 | 0.53% | 0.66% | 0.75% | 0.77% |
| 35 | 13,686 | 0.01% | 0.01% | 0.01% | 0.01% |
| 36 | 14,000 | 0.00% | 0.00% | 0.00% | 0.00% |
| 37 | 13,758 | 1.56% | 1.89% | 2.18% | 2.55% |
| 38 | 13,562 | 1.40% | 1.72% | 1.78% | 1.96% |
| 39 | NaN | NaN | NaN | NaN | NaN |
| 40 | 13,519 | 0.81% | 0.94% | 1.21% | 0.89% |

Table 2.7 presents the results of the exact method (see Section 2.2.2) compared to the different solution methods. The exact method was implemented in the OPL modeling language, using CPLEX 12.5 as the underlying solver. A time limit of 10 hours was given to CPLEX to tackle each instance. The OPL results are compared to the best values of $LAG$, and to the average values for the other methods. A NaN value indicates that the exact method was not able to return a feasible solution, which often happens if the number of items is important.
We can observe that CPLEX failed on 8 instances (with $n$ ranging from 30 to 44). CPLEX was able to optimally solve only instances 13 and 27, and to propose feasible upper bounds for 30 instances (with $n$ ranging from 16 to 38). Instance 13 (with $n = 13$ items and $m = 4$ classes) was solved in 461 seconds whereas instance 27 (with $n = 11$ and $m = 1$) was solved in 24,066 seconds. This seems to indicate that CPLEX can optimally solve instances with more items if the number of classes is larger (which is straightforward as the search can be split proportionally to the number of classes).
A positive gap means that an algorithm performs better and a negative gap means that the upper bound returned by OPL is better than the best found solution of all the solution methods. The average gaps on the forty instances are 0.06%, 0.07%, 0.17% and 0.17% for $LAG$, $TS$, $TS + DIV$, and $GA$, respectively. The exact method could be used if enough computational time is available or to solve a specific important instance. Note however that as the exact method needs hours to complete, it is not appropriate for *Renault*.

To further compare the performances of the metaheuristics, Figure 2.5 present the evolution of the average objective function best value for each algorithm over the 40 instances and the ten runs. The horizontal line is the length of a truck. $GA$ tends to perform best and has the lowest objective function curve over time. The bumps of the $GA$ curve are explained by the fact that often a generation requires more than a minute, and thus the average for each minute takes into account only runs which have a value for the corresponding minute. $TS$ and $TS + DIV$ shows similar performances, but $TS + DIV$ still improves $TS$ values, which again shows the relevance of the diversification procedure.

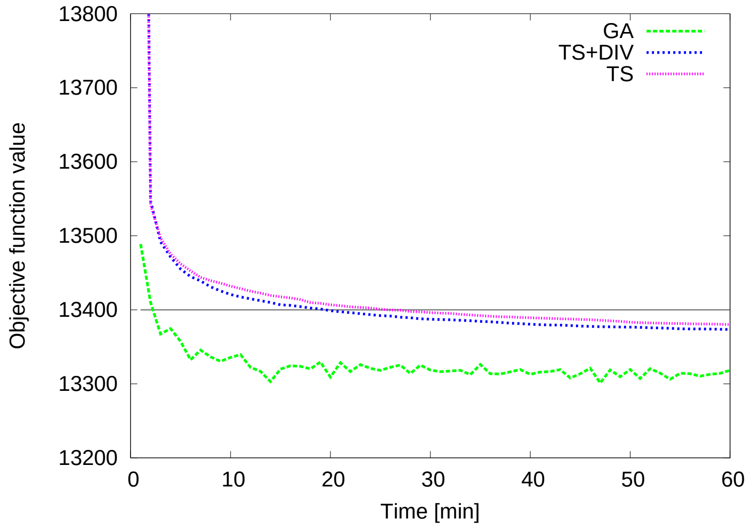

Figure 2.5: Evolution of the best objective function value over time, for each algorithm

## 2.6 Steering strategies

An extension is now proposed, which consists in testing different steering strategies when, instead of having $T$ seconds to tackle each instance, one has to tackle the full set of instances before time $T$ is reached.

Having a time $T$ to tackle a full set $U = \{u_1, u_2, \ldots, u_v\}$ of truck instances points out the question of knowing when to stop the search on a certain instance to tackle the next one. To answer this question, different strategies are first described and then tested with exposed results. In the following, let $T'$ represents the current time.

The first strategy, called *fair strategy* ($FS$), works as follows: a time $\frac{T}{v}$ is given to $LAG$ to tackle each instance, as this method is very fast and usually gets satisfying results. A second strategy, called *fair strategy with genetic* ($FSG$), is an extension of $FS$. It starts with a procedure called $LAGSORT$, which consists in giving a time $\frac{1}{3} \cdot \frac{T}{v}$ per instance to $LAG$ (i.e., a total time of $\frac{T}{3}$ is provided to test all the instances). Each time an instance $u_i$ is solved (i.e., the loading fits in the truck), it is removed from $U$ and moved to a set $P$ of solved instances. After this procedure, during a time of $\frac{T-\frac{1}{3}\cdot\frac{T}{v}}{|U|}$ per instance, $GA$ is used to tackle the remaining instances (i.e., the ones that have not been previously solved by $LAG$), and using as initial solutions the best solution provided by $LAGSORT$.

*Renault* pointed out an important goal of this problem which is trying to solve the largest number of trucks, regardless of the solution quality (i.e., as soon as $f \leq$ 13,400mm, the instance is considered as solved and keeping minimizing $f$ is irrelevant. This stopping criterion seems the most straightforward, but for benchmarking purposes, we decided to not include it in both $FS$ and $FSG$). Thus, a third steering algorithm is then proposed, called *fit in strategy* ($FIS$). First, as for $FS$, $LAGSORT$ is performed. After this first phase, the algorithm tackles the next instance $u_i \in U$ using $GA$ for a maximum time of $T_{u_i} = \frac{T-T'}{|U|}$. $GA$ moves to the next instance $u_{i+1}$ of $U$ if the time $T_{u_i}$ is reached or when the loading fits in the truck. Each time an instance $u_i$ is solved or the time $T_{u_i}$ dedicated to it is reached, $u_i$ is removed from $U$ and moved to $P$. As $U$ is dynamically updated, so is the time dedicated to an instance. If instance $u_i$ is given a time limit $T_{u_i}$ and is solved before $T_{u_i}$ is reached, then the next instance $u_{i+1}$ of $U$ has a time $T_{u_{i+1}} > T_{u_i}$, and $T_{u_{i+1}} = T_{u_i}$ otherwise. An extended version of this strategy, called *extended fit it strategy* ($EFIS$), performs a sorting of the instances according to the number $n$ of items in an increasing fashion, after $LAGSORT$ has been performed (thus, only the instances of $U$ are sorted). Therefore, $GA$ has then more chance to be able to solve the first instances of $U$ and spare some interesting time for the next ones, which have larger $n$ values.

The next strategy, called *sort and perform* ($SP$), first uses $LAGSORT$ (but for which $LAG$ is stopped if the loading fits in the truck). Then for a maximum time $T_{u_i} = \frac{1}{2} \cdot \frac{T-T'}{|U|}$ for each instance $u_i \in U$, $TS$ (without diversification) is used to tackle as many instances as possible (again, $TS$ can move to the next instance if it is able to solve the instance before the time limit). In addition, the instances that were not solved but which have a gap of more than $\sigma\%$ (parameter tuned to 2, with $\sigma$ tested in interval [1, 30]) between $f_i$ (best encountered value of $f$ for instance $u_i$) and $L_t$ (length of the truck) are removed from $U$ and added to $P$, as they are likely to be unsolvable. The remaining instances in $U$ are then sorted (in a negligible computing time) in an increasing order of the remainder $f_i - L_t$, and then the $TS$ (without diversification) is performed on each $u_i \in U$ for a time $T_{u_i} = \frac{1}{2} \cdot \frac{T-T'}{|U|}$. Again, $TS$ can move directly to the next instance

$u_{i+1} \in U$ if it has been able to solve an instance $u_i$ before time $T_{u_i}$ is reached. In $SPG$, $SP$ is used but $GA$ is used instead of $TS$ for the two last phases. Finally, in $SPTG$, $SP$ is used, where the second phase is performed by $TS$ and the last phase by $GA$.

Please note that for every strategy that requires $GA$, $N$ is set to 5 instead of 10, due to shorter time limits (when compared to the experimental conditions of Section 2.5).

Table 2.8: Results of the steering strategies

| Strategy | Number of solved instances |
|---|---|
| $FS$ | 336 |
| $FSG$ | 338 |
| $FIS$ | 341 |
| $EFIS$ | 341 |
| $SP$ | 340 |
| $SPG$ | 338 |
| $SPTG$ | 346 |

Results are provided in Table 2.8 for a time limit of $T = 1791$ minutes (i.e., an average of 3 minutes for each of the 597 instances). For comparison purposes only, note that 328 instances are solved by $LAG$ when one minute is allocated to each instance. $FS$ solved 336 instances. Thus, eight instances are solved in the two additional minutes allocated to $FS$. When $GA$ is used in $FSG$, two additional instances can be solved compared to $FS$. $FIS$ shows the relevance of moving to the next instance when an instance is solved, which save some computing time for the remaining instances. Interestingly, $EFIS$ does not provide any improvement over $FIS$. This is probably due to the fact that the prior sorting regarding the number $n$ of items in each instance is not relevant for the remaining instances. The last three steering strategies, namely $SP$, $SPG$ and $SPTG$, show the relevance of the combination of two different methods. Using together $TS$ and $GA$ shows the best results as it can solve up to 346 instances. Note that without steering techniques, $LAG$ is given 10 minutes per instance to be able to solve 346 instances. Thus, $LAG$ requires a total time of 5970 minutes $(597 \cdot 10)$, which is almost 100 hours. In contrast, $SPTG$ requires 1135 minutes $(597{\cdot}1+(597-328){\cdot}2)$, which is almost 19 hours. Therefore, $SPTG$ is more than five times faster compared to $LAG$ without steering strategy. This shows the relevance of using advanced steering strategies when having a hard deadline to tackle all the instances. Experiments show that on the full set of 597 instances, almost 250 instances are not solvable (i.e., $f > 13{,}400$mm). In Section 2.5, 350 instances only are solved by $GA$ and $LAG$ with a time limit of $T = 60$ minutes per instance (which is a total time of 597 hours). Thus 346 instances in 19 hours seems an interesting result and shows the relevance of the more refined steering strategies. Such a result is relevant from a practical standpoint: *Renault* has now insights to improve their approach when dealing with a complete set of instances with a hard time limit.

## 2.7 Conclusion

In this paper, we have investigated a practical application faced by the French car manufacturer *Renault* on a daily basis. We proposed and compared a MILP-based approach and various metaheuristics (a tabu search with an optional diversification mechanism, as well as an hybrid genetic algorithm). To benchmark the different algorithms, we extracted a relevant set $Z$ of 40 instances from the 597 real instances provided by *Renault*. We compared the *Renault* algorithms to the proposed metaheuristics on the set $Z$ and showed that the genetic algorithm is the most powerful and fast algorithm, even when compared to refined and well-tuned greedy algorithms with look-ahead process. The conclusions remain unchanged when comparing the methods on the full set of 597 instances. Furthermore, we proposed an extension where a total time $T$ is allocated to tackle all the instances. A steering strategy involving a combination of greedy algorithms, tabu search and genetic metaheuristics leads to the best results. This strategy was proved to be five time faster than working with the best *Renault* algorithm without using a steering strategy.

### Acknowledgements

The authors would like to thank Alain Nguyen and Jean-Philippe Brenaut, both from the *Renault* operations research team, for their availability and relevant advices on the elaboration of the algorithms.

# Chapter 3

# Impact of Online Tracking on a Vehicle Routing Problem with Dynamic Travel Times

JEAN RESPEN - *University of Geneva, Switzerland*
NICOLAS ZUFFEREY - *University of Geneva, Switzerland*
JEAN-YVES POTVIN - *University of Montreal, Canada*

This paper analyzes the impact of vehicle tracking devices, such as global positioning systems, on a vehicle routing problem with time windows in the presence of dynamic customer requests and dynamic travel times. It is empirically demonstrated that substantial improvements are achieved over a previously reported model which does not assume such tracking devices. We also analyze how the system handles dynamic perturbations to the travel times that lead to earliness or lateness in the planned schedule.

## 3.1 Introduction

Dynamic vehicle routing is attracting a growing attention in the research community. In these problems, some data are not known in advance, but is rather revealed in real-time while the routes are executed. Dynamically occurring customer requests have often been considered, but also dynamic customer demands and dynamic travel times. Since our work deals with dynamic customer requests and dynamic travel times, we provide a non exhaustive review of these variants in the following. Note that general considerations as well as exhaustive surveys on different types of dynamic vehicle routing problems can be found in [44, 96, 102].

In [41], the authors propose a parallel tabu search heuristic for a vehicle routing problem with soft time windows in the presence of dynamic customer requests. In this work, a central dispatch office manages the planned routes. Furthermore, the vehicles are not aware of their planned routes and are informed of their next destination only when they have reached their current customer location. The optimization procedure runs in background and is interrupted when a vehicle reaches a customer or when a new customer request is received. At this point, the best known solution is returned and updated, based on the new information received, and a new optimization task is launched on the updated solution. An adaptive memory is also combined with the parallel tabu search to maintain a pool of interesting solution alternatives. It is shown that this algorithm improves over simple greedy heuristics when the optimization tasks can run long enough before they are interrupted. This work was later extended in [40] to address a courier service application where each new customer request is made of a pick-up and a delivery location, with a precedence constraint between the two locations.

The impact of diversion has also been studied in the literature. It consists in diverting a vehicle to a newly occurring customer request, close to the vehicle's current location, while en route to another destination. In [61], diversion is integrated within the tabu search heuristic reported in [41], and is shown to provide substantial improvements. Diversion is also considered in [48] where two different approaches are compared. The first approach, called sample-scenario planning, provides high-quality solutions, but at the expense of large computation times. At each step, a sample of likely-to-occur future customer requests is generated to obtain a number of scenarios. Robust planned routes are then computed based on these scenarios. The second method, called anticipatory-insertion heuristic, incorporates information about expected future customer requests when each new request is inserted into the current planned routes.

As illustrated by the last method, the myopy of methodologies developed for static problems can be alleviated by exploiting any probabilistic knowledge about the occurrence of future customer requests, either implicitly or explicitly. Different approaches are based on waiting and relocation strategies. In [12], for example, the vehicles can either wait at their current customer location or at any other site, to answer customer requests that are likely to occur in their vicinity. A similar idea is also found in [63]. Here, dummy customers in the planned routes stand for future, likely-to-occur, customer requests which are replaced by true requests when they occur. Another approach reported in the literature uses a short-term and a long-term objective, where the latter tends to introduce waiting times in the planned routes to facilitate the inclusion of future requests [86].

Dynamic travel times, where times can change due to road congestion, have also raised the attention of the research community. In [35], for example, a traffic management system forecasts the travel times, based on road conditions, and transmits this information to the dispatch office. The latter then takes appropriate actions in the context of a pickup and delivery problem, assuming that the communication between the dispatch office and the drivers is possible at all time. The authors also describe a general framework to account for dynamic travel times and report results based on traffic information from the city of Berlin, Germany.

The authors in [101] consider a vehicle routing problem with time windows and dynamic travel times. The latter have three different components: static long-term forecasts (often referred to as time-dependent travel times in the literature), short-term forecasts, where the travel time on a link is modified with a random uniform value to account for any new information available when a vehicle is ready to depart from its current location, and dynamic perturbations caused by unforeseen events that might occur while traveling on a link (e.g., an accident causing sudden congestion). A modification to a planned route is only possible when the vehicle is at a customer location. Hence, a planned route cannot be reconsidered while the vehicle is traveling on a link between two customer locations. An extension to this model is proposed in [77]. In this work, the position of each vehicle can be obtained when a vehicle reaches its lateness tolerance limit or when a new customer request occurs. Based on this information, the planned route of each vehicle is reconsidered, including the possibility of diversion (i.e., redirecting a vehicle en route to its current destination). The results show that the setting of an appropriate lateness tolerance limit can provide substantial improvements. Here, we propose a further extension by assuming that the position of each vehicle is known at all time. This assumption allows the system to detect perturbations to the travel times and take appropriate actions much earlier.

This paper is organized as follows. A description of the problem is provided in Section 3.2. Then, Section 3.3 describes the two models in [77, 101] and explain the extension proposed here. Section 3.4 introduces travel time perturbations that lead to earliness in the planned schedule. The results obtained with the model in [77] and the new extension are then compared in Section 3.5. Finally, Section 3.6 concludes the paper and proposes future research avenues.

## 3.2 Problem description

The description of the problem is based on [77] where a fleet of vehicles performs routes, starting from and ending at a central depot, to collect goods at customer locations. Each customer must be visited exactly once by a vehicle within a (soft) time window. Some customer requests are said to be static, because they are known in advance and can be used to create initial planned routes. Other requests occur dynamically through the day and must be incorporated in real-time into the current solution. The ratio between the number of static requests and the total number of requests (static plus dynamic) is known as the degree of dynamism and is denoted $d_{od}$ in the following [79]. Additional details on this topic can be found in [96].

Formally, let us consider a complete undirected graph $G = (V, E)$ with a set of vertices $V = \{0, 1, 2, \ldots, n\}$, where vertex 0 is the depot, and a set of edges

$E$. Each edge $(i, j) \in E$ is characterized by a travel time $t_{ij}$. Also, each vertex $i \in V \setminus \{0\}$ has a time window $[e_i, l_i]$. A vehicle can arrive before the lower bound $e_i$ but must wait to start the service. Conversely, a vehicle can arrive after the upper bound $l_i$, but a (linear) penalty is incurred in the objective. We assume that $K$ vehicles of virtually infinite capacity are available. Each vehicle performs a single route which must end before an upper bound $l_0$, otherwise another penalty is incurred in the objective.

The objective function $f$ takes into account (1) the travel time, (2) the sum of lateness at customer locations and (3) the lateness at the depot. Denoting $t_{i^k}$ the arrival time of vehicle $k$ at customer $i \in V \setminus \{0\}$ (assuming that customer $i$ is served by vehicle $k$) and by $t_0^k$ the return time of vehicle $k$ at the depot 0, the objective can be written as:

$$\begin{aligned} f(S) &= \sum_{k \in K} f(S^k) \\ &= \sum_{k \in K} \left( \alpha \sum_{p=1}^{m_k} t_{i_{p-1}^k, i_p^k} + \beta \sum_{p=1}^{m_k - 1} max\{0, t_{i_{p-1}^k} - l_{i_{p-1}^k}\} + \gamma \max\{0, t_0^k - l_0\} \right) \end{aligned} \quad (3.1)$$

where $S = \bigcup_{k \in K} S^k$ represents a solution (a set of routes) and $S^k = \{i_0^k, i_1^k, \ldots, i_{m_k}^k\}$ is the route of vehicle $k \in K$, with $i_0^k = i_{m_k}^k = 0$. The weights $\alpha$, $\beta$ and $\gamma$ are used to put more or less emphasis on travel time or lateness.

With regard to the static, time-dependent, component of the travel time, we do as in [62], we split the operations day in three time periods for the morning, lunch time and afternoon. With each period is associated a coefficient that multiplies the average travel time (namely, 1.25 for the morning, 0.5 for the lunch time, and 1.25 for the afternoon. Therefore, morning and afternoon coefficients induce that there is important traffic as people are going or leaving their work, and lunch time is a rather low traffic condition due to people having lunch breaks). To guarantee that a vehicle leaving earlier from some customer location also arrives earlier at destination, which is known as the FIFO property, the travel times are adjusted when a boundary between two time periods is crossed.

The travel times also suffer dynamic perturbations due, for example, to unexpected congestion. A dynamic perturbation is thus included based on a normal probability law with mean 0 and different standard deviations $\sigma$. Perturbations with negative values, leading to earliness, are reset to 0 in the first implementation, so that only lateness in the planned schedule can occur (as it is done in [77, 101]). In a second implementation, perturbations with negative values are also considered.

## 3.3 Models

Three related models are presented in this section. The third model is an extension of the two previous ones.

### 3.3.1 Model 1

In Model 1 [101], a central dispatch office manages the planned route of each vehicle. It is assumed that communication between the drivers and the dispatch office takes place only at customer locations. When a driver has finished serving a customer, he communicates with the dispatch office to know his next destination. Hence, the drivers are not aware of their planned route, but only of the next customer to be served. The static requests are first used to construct initial routes through an insertion heuristic where, at each iteration, a customer is selected and inserted at the best possible place in the current routes (i.e., with minimum increase in the objective value). At the end, a local search-based improvement procedure is triggered using CROSS exchanges [124], where sequences of customers are moved from one route to another. Finally, another local search-based improvement procedure is applied to each individual route, based on the relocation of each customer. Whenever a new dynamic request is received, the same insertion and reoptimization procedures are applied to update the planned routes.

Since travel times are dynamic, a lateness tolerance limit $TL$ is defined, which is the maximum acceptable delay to a vehicle's planned arrival time at its current destination before some reassignment action is considered. For example, if we assume that $s^k$ is the current destination of vehicle $k$ and its planned arrival time is $t_{s^k}$, then $t_{s^k} + TL$ defines the tolerance time limit $TTL^k$ of vehicle $k$.

That is, if vehicle $k$ has not reached customer $s^k$ at time $TTL^k$, $s^k$ is removed from its planned route and inserted in the planned route of some other vehicle $l$ (note that vehicle $k$ is not aware of this change and will continue toward $s^k$, as communication between the dispatch office and vehicles only take place at customer locations). If it happens that vehicle $k$ still reaches $s^k$ before vehicle $l$ and while $l$ is en route to $s^k$, then vehicle $k$ serves $s^k$, but vehicle $l$ will only know when reaching $s^k$. A major drawback of this model thus relates to the limited communication scheme between the dispatch office and the vehicles.

### 3.3.2 Model 2

In [77], Model 1 was extended by adding diversion to allow any vehicle to be redirected to another customer, while en route to its current destination (if it provides some benefit with regard to the objective). When (a) a new customer request is received or (b) some vehicle $k$ has reached its tolerance time limit, it is assumed that the dispatch office can obtain the current location of each vehicle to evaluate the benefits of a diversion. In case (a), a pure diversion of vehicle $k$ to serve the newly occurring customer request is considered whereas, in case (b), the current destination of vehicle $k$ is reassigned to another vehicle $l \neq k$. Vehicle $k$ is then redirected to the customer that immediately follows (what was) its current destination in the planned route. Figure 3.1 illustrates these two cases. In Figure 3.1 (a), a new customer request $s$ occurs while vehicle $k$ is located at position $x$ between vertices $i^k_{p-1}$ and $i^k_p$. In this case, vehicle $k$ will serve $s$ before $i^k_p$ if it is beneficial to do so. In Figure 3.1 (b), vehicle $k$ has reached its tolerance time limit. Thus, its current destination $s = i^k_p$ is removed from its planned route and is reassigned to another vehicle $l$, while vehicle $k$ is redirected to $i^k_{p+1}$. The results reported in [77] show that Model 2 significantly outperforms Model 1. Also, empirical results demonstrated that the best $TL$ value is 0. That is, an appropriate action must be considered as soon as a perturbation to the planned schedule is detected.

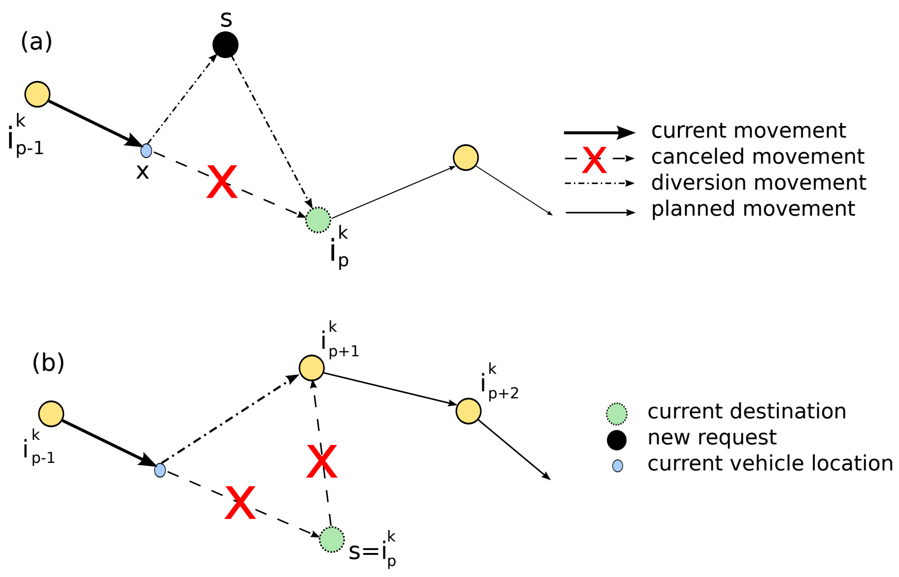

Figure 3.1: Diversion and reassignment actions

### 3.3.3 Model 3

The new model proposed here extends Model 2 by assuming that the position of each vehicle is known at all time, not only when the two types of situations described above for Model 2 occur. To this end, we first assume that the dynamic perturbation component of the travel time is distributed uniformly along a link. Then, as soon as it is impossible for vehicle $k$ to arrive at its current destination at time $TTL^k$, a reassignment action is considered. For example, let us assume that vehicle $k$ departs from $i$ to $j$ at time $t$, with a travel time $t_{ij} = 5$ and a dynamic perturbation $\Delta t_{ij} = 5$. That is, the vehicle is planned to arrive at $j$ at time $t$+5, but will in fact arrive only at time $t$+10. If $TL = 0$, then $TTL^k = t$+5 and Model 2 will consider a reassignment at time $t+5$ when it is observed that vehicle $k$ has not yet reached $j$. On the other hand, by tracking the current position of each vehicle, Model 3 can detect the problem much earlier. For example, at time $t+1$, vehicle $k$ has still to cover $\frac{9}{10}$ of the distance, so that the planned arrival time at location $j$ could be updated to $t+1+\frac{9}{10}\cdot 5 = t+5.5$ (assuming no more perturbation on the remainder of the link) which already exceeds $TTL^k$. This assumption holds if we assume the availability of tracking devices, which are now widely available at competitive prices, as well as a mobile network coverage of the service area (note that preliminary results regarding this model have been presented in [107]).

## 3.4 Earliness

As stated earlier, only positive perturbations to the travel times that lead to lateness in a vehicle schedule were considered in [77], by resetting any negative value to 0. Negative perturbation values, leading to earliness in a vehicle schedule (i.e., the vehicle will arrive earlier than expected at its current destination) have been tested here for both Models 2 and 3. If some vehicle $l$ is late, then the earliness in the schedule of another vehicle $k$ will automatically be exploited by the optimization procedure. That is, the current destination of vehicle $l$ will likely be transferred to vehicle $k$. The benefits of Model 3 over Model 2 in this situation are the same as those mentioned for positive perturbation values: it will be possible to detect the earliness and lateness in the vehicle schedules before the vehicles reach their current destination and, consequently, react more promptly. Figure 3.2 illustrates this capability. In the figure, vehicle $k$ is currently traveling between customers $i_{p-1}^k$ and $i_p^k$ and is ahead of its schedule. Similarly, another vehicle $l$ is traveling between customers $i_{p-1}^l$ and $i_p^l$ and is late. Then, vehicle $k$ can be redirected to $i_p^l$ and vehicle $l$ to $i_{p+1}^l$ while both vehicles are en route.

## 3.5 Computational results

Tests were performed on a 3.4 GHz Intel Quad-core i7 with 8 GB of DDR3 RAM memory. The Euclidean 100-customer Solomon's benchmark instances [121] were used to compare Models 2 and 3. Any dynamic customer request $i$ was set to occur at time $e_i \cdot r$, where $e_i$ is the lower bound of the time window at customer $i$ and $r$ is a random number between 0 and 1. Parameters $\alpha$, $\beta$ and $\gamma$ were set to 1 in the objective. For these experiments, only the three classes of instances $R2$, $C2$ and $RC2$ with 11, 8 and 8 instances, respectively, were considered due to their large time horizon which allows for many customers

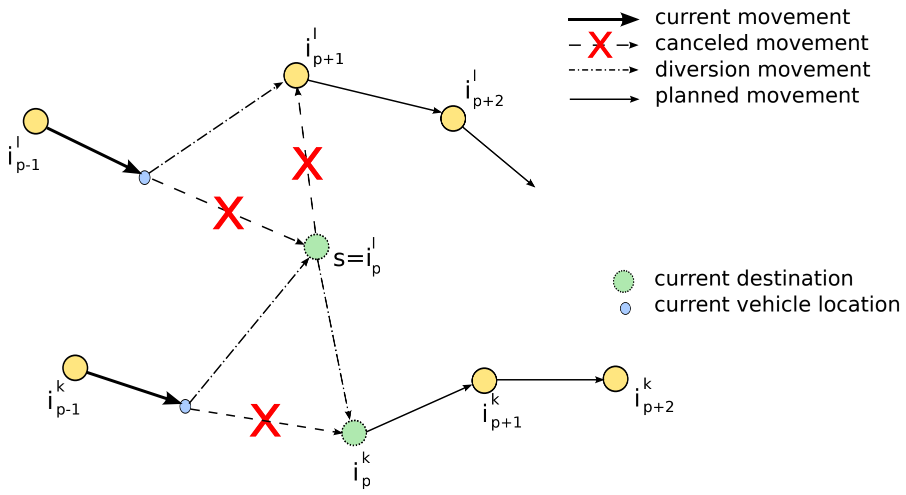

Figure 3.2: Integrating earliness into the model

per route. Note that customers are randomly generated in $R2$, clustered in $C2$ or both clustered and randomly generated in $RC2$. Note also that the computing times are not commented given that the optimization takes place within a fraction of a second.

Tables 3.1 to 3.3 show the results on classes $R2$, $C2$, and $RC2$ respectively, for various tolerances $TL$ and $\sigma$ values in Euclidean units (where $\sigma$ is the variance of the dynamic perturbations to the travel times). Each entry in these tables correspond to the average objective value over ten runs, using ten different seeds, and over each instance of a given class. There is also a pair of numbers between parentheses: the first number is the average number of times a reassignment action was considered (per instance) and the second number is the percentage of reassignments that were undertaken because they proved to be beneficial. The last row with $TL = 1000 \cdot \sigma$ is an extreme case where no action is taken. That is, the planned routes are followed whatever the perturbation. The degree of dynamism $d_{od}$ was set to 0.5 and only lateness with regard to the current schedule was allowed (i.e., negative perturbations to the travel times were reset to 0).

These results indicate that small $TL$ values lead to better results, with the best value being $TL = 0$ in all cases. In other words, a reactive action should be considered as soon as a perturbation to the current schedule is detected. This

Table 3.1: Results of Model 3 on class R2

| $TL$ | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| 0 | 1548.57 (52.76,8.86%) | 2099.08 (54.65,15.77%) | 4779.91 (67.24,45.86%) | 7322.89 (77.35,65.35%) |
| $0.5 \cdot \sigma$ | 1595.78 (31.95,8.14%) | 2233.27 (32.64,13.59%) | 6283.63 (35.56,39.72%) | 13491.44 (36.91,56.11%) |
| $1 \cdot \sigma$ | 1623.17 (15.96,7.86%) | 2288.34 (16.24,13.89%) | 6641.03 (16.58,34.10%) | 15201.17 (15.05,44.69%) |
| $2 \cdot \sigma$ | 1634.21 (2.05,9.73%) | 2326.33 (2.04,12.50%) | 6945.38 (1.75,18.75%) | 15211.18 (1.33,27.40%) |
| $3 \cdot \sigma$ | 1640.09 (0.07,25.00%) | 2335.04 (0.07,0.00%) | 6925.79 (0.04,0.00%) | 15260.43 (0.04,0.00%) |
| $1000 \cdot \sigma$ | 1641.17 (0.00,-) | 2335.04 (0.00,-) | 6925.79 (0.00,-) | 15260.43 (0.00,-) |

Table 3.2: Results of Model 3 on class C2

| $TL$ | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| 0 | 2145.41 (51.65,6.53%) | 2746.84 (51.68,7.21%) | 6047.75 (53.70,12.90%) | 10865.41 (58.20,26.29%) |
| $0.5 \cdot \sigma$ | 2362.23 (31.60,7.83%) | 2972.81 (31.53,8.88%) | 6432.07 (32.15,13.61%) | 11914.13 (32.68,25.78%) |
| $1 \cdot \sigma$ | 2419.65 (15.80,8.39%) | 3054.61 (15.75,9.52%) | 6722.66 (15.48,15.51%) | 12808.44 (12.50,28.60%) |
| $2 \cdot \sigma$ | 2696.53 (2.03,11.11%) | 3311.12 (2.03,9.88%) | 7119.54 (1.75,18.57%) | 13263.69 (0.90,33.33%) |
| $3 \cdot \sigma$ | 2725.66 (0.10,50.00%) | 3349.47 (0.10,25.00%) | 7197.18 (0.10,25.00%) | 13326.74 (0.00,-) |
| $1000 \cdot \sigma$ | 2728.99 (0.00,-) | 3359.17 (0.00,-) | 7200.55 (0.00,-) | 13326.74 (0.00,-) |

Table 3.3: Results of Model 3 on class RC2

| $TL$ | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| 0 | 1515.47 (52.03,6.78%) | 1878.14 (53.98,13.06%) | 3975.85 (63.75,36.39%) | 6205.70 (73.58,56.81%) |
| $0.5 \cdot \sigma$ | 1547.14 (31.58,6.25%) | 1952.08 (32.45,11.86%) | 4815.76 (34.40,30.31%) | 10187.55 (34.33,47.20%) |
| $1 \cdot \sigma$ | 1572.37 (15.85,4.73%) | 2006.65 (16.03,10.30%) | 5313.23 (15.38,27.80%) | 11370.48 (13.75,36.91%) |
| $2 \cdot \sigma$ | 1590.77 (1.98,3.80%) | 2037.86 (1.98,7.59%) | 5490.76 (1.88,13.33%) | 11641.22 (1.25,26.00%) |
| $3 \cdot \sigma$ | 1591.66 (0.10,0.00%) | 2041.68 (0.10,0.00%) | 5495.05 (0.10,25.00%) | 11685.73 (0.08,66.67%) |
| $1000 \cdot \sigma$ | 1591.66 (0.00,-) | 2041.68 (0.00,-) | 5501.21 (0.00,-) | 11734.49 (0.00,-) |

observation is in line with the results reported in [77]. Also, the percentage of reassignments that provide an improvement increases with $\sigma$. This is not surprising, given that the current plan is likely to be improved when large perturbations are encountered.

Table 3.4 shows the objective values as well as the percentage of improvement of Model 3 over Model 2 when $d_{od}$ ranges from 0.1 to 0.9 with $TL = 0$. Although we show only these results, additional experiments with other $TL$ values led to the same observation, namely, that Model 3 is clearly superior to Model 2 due to its ability to detect perturbations to the current plan much earlier. The improvement is quite substantial in the case of $R2$ and $RC2$, and can even reach 30% for large $\sigma$ values. The results are less impressive in the case of $C2$ (Model 3 is even worse than Model 2 for $d_{od} = 0.9$ and $\sigma = 16$). This observation can be explained by the geographical clustering of customers which seriously limits the benefit of redirecting a vehicle, for example to serve a distant customer in another cluster. Table 3.5 summarizes the improvements obtained over all $d_{od}$ values for each class of instances, as well as over all classes of instances.

Table 3.4: Improvement of Model 3 over Model 2 with $TL = 0$ for different $d_{od}$ values

| $d_{od}$ | | | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|---|---|
| | **R2** | Model 2 | 1183.59 | 1653.41 | 4827.73 | 8606.15 |
| 0.1 | | Model 3 | 1168.82 | 1598.71 | 4097.30 | 6545.20 |
| | | % Imprv. | 1.26% | 3.42% | 17.83% | 31.49% |
| | **C2** | Model 2 | 1195.68 | 1536.89 | 4073.00 | 8578.14 |
| | | Model 3 | 1171.77 | 1533.58 | 3920.30 | 8230.87 |
| | | % Imprv. | 2.04% | 0.22% | 3.89% | 4.22% |
| | **RC2** | Model 2 | 1267.60 | 1557.40 | 4002.04 | 7499.71 |
| | | Model 3 | 1259.96 | 1524.96 | 3512.06 | 5805.93 |
| | | % Imprv. | 0.61% | 2.13% | 13.95% | 29.17% |
| | **R2** | Model 2 | 1381.69 | 1933.86 | 5334.02 | 9228.19 |
| 0.3 | | Model 3 | 1346.87 | 1853.73 | 4520.57 | 7097.35 |
| | | % Imprv. | 2.59% | 4.32% | 17.99% | 30.02% |
| | **C2** | Model 2 | 1645.09 | 2014.47 | 5269.25 | 9705.95 |
| | | Model 3 | 1642.23 | 1970.29 | 5103.47 | 9466.89 |
| | | % Imprv. | 0.17% | 2.24% | 3.25% | 2.53% |
| | **RC2** | Model 2 | 1460.75 | 1857.16 | 4521.94 | 8105.48 |
| | | Model 3 | 1442.85 | 1775.30 | 3815.26 | 6338.51 |
| | | % Imprv. | 1.24% | 4.61% | 18.52% | 27.88% |
| | **R2** | Model 2 | 1604.03 | 2245.95 | 5891.80 | 9636.36 |
| 0.5 | | Model 3 | 1548.57 | 2099.08 | 4779.92 | 7322.89 |
| | | % Imprv. | 3.58% | 7.00% | 23.26% | 31.59% |
| | **C2** | Model 2 | 2193.26 | 2769.02 | 6221.36 | 11652.32 |
| | | Model 3 | 2145.41 | 2746.84 | 6047.76 | 10865.42 |
| | | % Imprv. | 2.23% | 0.81% | 2.87% | 7.24% |
| | **RC2** | Model 2 | 1551.26 | 1949.16 | 4621.90 | 8426.49 |
| | | Model 3 | 1515.47 | 1878.14 | 3975.86 | 6205.70 |
| | | % Imprv. | 2.36% | 3.78% | 16.25% | 35.79% |
| | **R2** | Model 2 | 2014.79 | 2823.41 | 6553.96 | 10366.10 |
| 0.7 | | Model 3 | 1932.65 | 2599.74 | 5292.18 | 8037.42 |
| | | % Imprv. | 4.25% | 8.60% | 23.84% | 28.97% |
| | **C2** | Model 2 | 2444.12 | 2806.64 | 6075.49 | 11937.20 |
| | | Model 3 | 2340.94 | 2790.25 | 5809.04 | 10929.91 |
| | | % Imprv. | 4.41% | 0.59% | 4.59% | 9.22% |
| | **RC2** | Model 2 | 1805.32 | 2343.49 | 5311.32 | 9153.11 |
| | | Model 3 | 1747.96 | 2185.97 | 4708.65 | 6948.25 |
| | | % Imprv. | 3.28% | 7.21% | 12.80% | 31.73% |
| | **R2** | Model 2 | 2182.42 | 2994.81 | 6961.94 | 11386.24 |
| 0.9 | | Model 3 | 2044.20 | 2769.36 | 5728.86 | 8902.86 |
| | | % Imprv. | 6.76% | 8.14% | 21.52% | 27.89% |
| | **C2** | Model 2 | 3498.62 | 4337.93 | 8702.07 | 15191.83 |
| | | Model 3 | 3414.02 | 4216.22 | 8941.81 | 14441.63 |
| | | % Imprv. | 2.48% | 2.89% | -2.68% | 5.19% |
| | **RC2** | Model 2 | 2097.39 | 2782.93 | 6743.54 | 10730.58 |
| | | Model 3 | 2008.42 | 2554.65 | 5759.39 | 8285.90 |
| | | % Imprv. | 4.43% | 8.94% | 17.09% | 29.50% |

Table 3.5: Improvement of Model 3 over Model 2 with $TL = 0$ over all $d_{od}$ values

| | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| $R2$ | 3.69% | 6.30% | 20.89% | 29.99% |
| $C2$ | 2.27% | 1.35% | 2.38% | 5.68% |
| $RC2$ | 2.38% | 5.33% | 15.72% | 30.81% |
| Overall | 2.78% | 4.33% | 13.00% | 22.16% |

Tables 3.6 to 3.8 are similar to Tables 3.1 to 3.3 and report the results of Model 3 for various tolerance $TL$ and $\sigma$ values with $d_{od} = 0.5$ when negative perturbations to the travel times are allowed (leading to earliness in the schedule). Tables 3.9 and 3.10 are also similar to Tables 3.4 and 3.5 and report the improvement of Model 3 over Model 2 with $TL = 0$ for $d_{od}$ values between 0.1 and 0.9 when negative perturbations are allowed. Not surprisingly, the trends are the same as those observed previously but are somewhat accentuated, in particular for large $\sigma$ values.

Table 3.6: Results of Model 3 on class R2 including negative perturbations

| $TL$ | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| 0 | 1408.23<br>(53.13,8.80%) | 1576.70<br>(54.82,14.23%) | 2951.06<br>(66.20,42.85%) | 4156.69<br>(76.67,62.81%) |
| $0.5 \cdot \sigma$ | 1451.92<br>(32.22,7.62%) | 1675.40<br>(32.73,12.89%) | 4164.83<br>(35.77,38.81%) | 9962.26<br>(37.57,55.17%) |
| $1 \cdot \sigma$ | 1474.64<br>(16.20,7.58%) | 1719.48<br>(16.35,14.17%) | 4840.45<br>(16.71,34.33%) | 12027.22<br>(15.36,45.03%) |
| $2 \cdot \sigma$ | 1488.51<br>(2.12,8.58%) | 1753.46<br>(2.10,10.82%) | 5176.51<br>(1.90,25.36%) | 12765.64<br>(1.39,40.52%) |
| $3 \cdot \sigma$ | 1490.47<br>(0.13,14.29%) | 1761.16<br>(0.12,15.38%) | 5160.00<br>(0.09,10.00%) | 12485.61<br>(0.07,12.50%) |
| $1000 \cdot \sigma$ | 1491.05<br>(0.00,-) | 1762.37<br>(0.00,-) | 5161.13<br>(0.00,-) | 12489.44<br>(0.00,-) |

Table 3.7: Results of Model 3 on class C2 including negative perturbations

| $TL$ | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| 0 | 2208.83<br>(52.21,6.22%) | 2480.35<br>(52.43,7.20%) | 4934.58<br>(54.14,12.65%) | 8706.28<br>(57.91,24.15%) |
| $0.5 \cdot \sigma$ | 2349.66<br>(31.93,7.01%) | 2593.29<br>(32.06,8.38%) | 5153.91<br>(32.63,13.26%) | 9852.66<br>(32.64,24.01%) |
| $1 \cdot \sigma$ | 2433.25<br>(15.90,8.18%) | 2697.11<br>(15.95,9.01%) | 5550.78<br>(15.54,14.88%) | 10736.38<br>(12.54,26.32%) |
| $2 \cdot \sigma$ | 2592.38<br>(2.09,10.18%) | 2877.84<br>(2.08,12.65%) | 5923.32<br>(1.73,21.01%) | 11454.67<br>(0.89,29.58%) |
| $3 \cdot \sigma$ | 2639.34<br>(0.14,27.27%) | 2932.43<br>(0.14,9.09%) | 5999.76<br>(0.11,22.22%) | 11449.34<br>(0.04,0.00%) |
| $1000 \cdot \sigma$ | 2638.97<br>(0.00,-) | 2933.91<br>(0.00,-) | 6018.05<br>(0.00,-) | 11449.34<br>(0.00,-) |

Table 3.8: Results of Model 3 on class RC2 including negative perturbations

| $TL$ | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| 0 | 1444.06 (53.05,7.28%) | 1570.02 (54.51,12.27%) | 2602.30 (65.19,37.03%) | 3583.52 (74.21,55.47%) |
| $0.5 \cdot \sigma$ | 1472.57 (32.01,6.44%) | 1630.52 (32.78,11.94%) | 3387.50 (34.94,30.88%) | 7333.01 (34.93,48.21%) |
| $1 \cdot \sigma$ | 1489.27 (15.96,6.50%) | 1669.12 (16.15,12.00%) | 3931.03 (15.61,29.78%) | 8636.15 (13.38,39.81%) |
| $2 \cdot \sigma$ | 1503.83 (2.06,8.48%) | 1712.14 (2.10,16.07%) | 4122.88 (1.83,24.66%) | 9111.62 (1.39,28.83%) |
| $3 \cdot \sigma$ | 1508.63 (0.14,0.00%) | 1722.23 (0.14,0.00%) | 4150.32 (0.14,18.18%) | 9185.94 (0.05,25.00%) |
| $1000 \cdot \sigma$ | 1508.63 (0.00,-) | 1722.23 (0.00,-) | 4145.58 (0.00,-) | 9184.10 (0.00,-) |

Finally, Figure 3.3 illustrates the average objective values of Models 2 and 3 with $TL = 0$ and $d_{od} = 0.5$ for a large number of $\sigma$ values taken between 1 and 32 using the instances of class $RC2$. In this figure, negative perturbations to the travel times are allowed. This figure clearly shows that the gap between the two models sharply increases with $\sigma$.

Table 3.9: Improvement of Model 3 over Model 2 with $TL = 0$ for different $d_{od}$ values, including negative perturbations

| $d_{od}$ | | | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|---|---|
| | **R2** | Model 2 | 1093.38 | 1238.16 | 3091.47 | 5400.52 |
| 0.1 | | Model 3 | 1082.36 | 1208.02 | 2531.94 | 3723.15 |
| | | % Imprv. | 1.02% | 2.50% | 22.10% | 45.05% |
| | **C2** | Model 2 | 1180.54 | 1361.29 | 3160.62 | 6975.13 |
| | | Model 3 | 1167.54 | 1343.24 | 3095.66 | 6527.79 |
| | | % Imprv. | 1.11% | 1.34% | 2.10% | 6.85% |
| | **RC2** | Model 2 | 1205.74 | 1322.11 | 2749.25 | 4765.88 |
| | | Model 3 | 1199.42 | 1300.70 | 2298.89 | 3178.00 |
| | | % Imprv. | 0.53% | 1.65% | 19.59% | 49.96% |
| | **R2** | Model 2 | 1255.76 | 1445.00 | 3417.23 | 5638.62 |
| 0.3 | | Model 3 | 1229.63 | 1377.47 | 2679.75 | 3874.91 |
| | | % Imprv. | 2.12% | 4.90% | 27.52% | 45.52% |
| | **C2** | Model 2 | 1626.13 | 1767.72 | 3902.18 | 7793.65 |
| | | Model 3 | 1607.29 | 1735.47 | 3651.05 | 7159.94 |
| | | % Imprv. | 1.17% | 1.86% | 6.88% | 8.85% |
| | **RC2** | Model 2 | 1372.78 | 1541.12 | 3208.57 | 5321.58 |
| | | Model 3 | 1356.16 | 1481.81 | 2532.72 | 3579.10 |
| | | % Imprv. | 1.23% | 4.00% | 26.68% | 48.68% |
| | **R2** | Model 2 | 1454.72 | 1668.62 | 3741.13 | 5996.84 |
| 0.5 | | Model 3 | 1408.23 | 1576.70 | 2951.06 | 4156.69 |
| | | % Imprv. | 3.30% | 5.83% | 26.77% | 44.27% |
| | **C2** | Model 2 | 2269.21 | 2496.09 | 5110.58 | 9508.50 |
| | | Model 3 | 2208.83 | 2480.35 | 4934.58 | 8706.28 |
| | | % Imprv. | 2.73% | 0.63% | 3.57% | 9.21% |
| | **RC2** | Model 2 | 1478.08 | 1634.76 | 3338.04 | 5278.44 |
| | | Model 3 | 1444.06 | 1570.02 | 2602.30 | 3583.52 |
| | | % Imprv. | 2.36% | 4.12% | 28.27% | 47.30% |
| | **R2** | Model 2 | 1803.15 | 2053.27 | 4159.67 | 6545.16 |
| 0.7 | | Model 3 | 1728.48 | 1909.71 | 3264.42 | 4409.19 |
| | | % Imprv. | 4.32% | 7.52% | 27.42% | 48.44% |
| | **C2** | Model 2 | 2310.80 | 2422.14 | 4852.43 | 9672.17 |
| | | Model 3 | 2217.70 | 2336.18 | 4573.30 | 8745.50 |
| | | % Imprv. | 4.20% | 3.68% | 6.10% | 10.60% |
| | **RC2** | Model 2 | 1669.42 | 1853.11 | 3614.90 | 5818.96 |
| | | Model 3 | 1625.96 | 1747.68 | 2805.90 | 3807.39 |
| | | % Imprv. | 2.67% | 6.03% | 28.83% | 52.83% |
| | **R2** | Model 2 | 1965.55 | 2283.00 | 4631.17 | 7386.65 |
| 0.9 | | Model 3 | 1861.09 | 2099.15 | 3610.27 | 4990.11 |
| | | % Imprv. | 5.61% | 8.76% | 28.28% | 48.03% |
| | **C2** | Model 2 | 3323.86 | 3621.42 | 6403.54 | 12274.79 |
| | | Model 3 | 3261.67 | 3446.48 | 6610.18 | 11123.78 |
| | | % Imprv. | 1.91% | 5.08% | -3.13% | 10.35% |
| | **RC2** | Model 2 | 2112.12 | 2381.04 | 5101.59 | 7644.64 |
| | | Model 3 | 2016.67 | 2195.41 | 3943.92 | 5532.53 |
| | | % Imprv. | 4.73% | 8.46% | 29.35% | 38.18% |

Table 3.10: Improvement of Model 3 over Model 2 with $TL = 0$ over all $d_{od}$ values, including negative perturbations

| | $\sigma = 1$ | $\sigma = 4$ | $\sigma = 16$ | $\sigma = 32$ |
|---|---|---|---|---|
| $R2$ | 3.28% | 5.90% | 26.42% | 46.26% |
| $C2$ | 2.22% | 2.52% | 3.10% | 9.17% |
| $RC2$ | 2.30% | 4.85% | 26.55% | 47.39% |
| Overall | 2.60% | 4.42% | 18.69% | 34.28% |

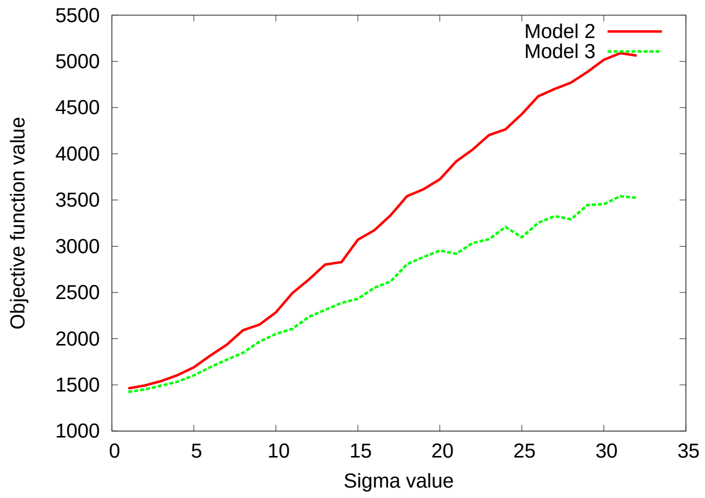

Figure 3.3: Average objective values of Models 2 and 3 on class $RC2$ with $TL = 0$, $d_{od} = 0.5$ and different $\sigma$ values

## 3.6 Conclusion

This paper has investigated the impact of on-line vehicle tracking devices on solution quality for a dynamic vehicle routing problem with time windows. The dynamic characteristics relate to the occurrence of new customer requests and perturbations to the travel times. We empirically demonstrated that a reactive action should be contemplated as soon as a perturbation is detected in the current planned routes. Our model was also shown to be robust and to behave well under different degrees of dynamism. It also proved to be significantly superior to another model where the location of each vehicle is only known at specific moments during the operations day. In the future, we plan to investigate the impact of other probability distributions to model the travel time perturbations. We also want to consider other ways to distribute a perturbation to the travel time along a link.

# Chapter 4

# Three-level inventory deployment for a luxury watch company facing various perturbations

Jean Respen - *University of Geneva, Switzerland*
Nicolas Zufferey - *University of Geneva, Switzerland*
Philippe Wieser - *EPFL, Switzerland*

A well-known Swiss watch brand, active in the top-end luxury market, is facing a complex inventory deployment problem where watches of different models (more than 100 different models) must be dispatched first to wholesalers to finally reach the shops where clients come in. Along the way, different perturbations are expected at three levels (production plan, demand, and dispatching), and accurate reactions must be taken to fit to these uncertainties. An exact method is proposed to model the problem without perturbations. Solution methods are proposed to solve realistic instances, where the exact method in not able to provide solutions in a fair amount of time. Results show the relevance of

the methods and the robustness of the solutions. Managerial insights are also given.

## 4.1 Introduction

In this paper, a well-known Swiss watch brand (denoted *SWB*), which is active in the luxury industry, is facing a multi-level inventory deployment problem (called (P)), where watches (called *items*) of different models (called *products*) are produced, and must be sent to the final shops while satisfying various constraints. Inventory deployment is often regarded as a very complex problem by companies, as different constraints and perturbations often raise along the different levels (i.e., from the production to the final clients), and the dispatching system must react accordingly. Here, the supply chain is composed of three different levels, namely a single factory, the wholesalers and the final destinations, which are the shops where the goods are sold to the clients. The factory is the only place where the goods are produced. For each considered geographical region, there exists at least one wholesaler which then delivers to its associated set of shops. A shop can only order the items to its single wholesaler. As far as *SWB* is concerned, the shops are the final points of interest, and the final client, who actually buys an item, is not integrated in the system as the shops are often not owned by *SWB*. Thus, in the following, the *customers* of *SWB* are the wholesalers, and the *clients* are the persons actually buying the items at the shops level. Unfortunately, in (P), three different perturbations are encountered along the supply chain: first at the production plan level, then at the dispatching level (wholesalers), and finally on the demand (i.e., the actual demand is different from the forecasted demand). These perturbations are respectively due to the suppliers, the behaviors of the wholesalers (on which *SWB* has not a total control), and the behaviors of the clients. (P) is new and there is no literature on it.

As the watches produced by *SWB* are expensive, the suppliers have a key role, because the raw material used to produce the luxury items is expensive and rare. Thus, there is no long list of suppliers which can provide the raw material. The number of products and the number of items yearly produced are both increasing with the demand, which makes the plant reach its production capacity limit. From a *SWB* perspective, the client wants his/her watch when he/she has decided to buy it, not later. Thus a perturbation of the production plan due to suppliers has a strong impact and is hard to manage, as every item that is not produced due to unreliable suppliers must be produced later, and therefore disturbs the production plan of the following weeks.

The planning horizon is a year, and the time unit is a week. (P) is a three-level inventory deployment problem, where the main decision to make is the number of items of each product to send (from the plant) to each shop on a per week basis. A *solution* is an aggregation of these numbers for a whole year. The objective function involves three different components: shortages, rarity and inventory. Most of the literature involves only two levels. There is no literature dealing with such a complex problem (P) with unreliable behaviors of suppliers, wholesalers and clients. Moreover, the perturbation on the production plan can result in an unfeasible solution, and thus a specific repair procedure is requested.

This paper aims at proposing dedicated solution methods and a simulator to improve the quality of the inventory deployment of *SWB*. The simulator is used to evaluate the quality of a solution according to the above described random events, and is helpful to conduct a sensitivity analysis, allowing to identify the key parameters. One of the proposed solution methods shows that the use of relevant distance functions can be very powerful for tackling (P), where exact methods are not appropriate on realistic instances. The best method is a matheuristic, which combines the accuracy of an exact method with the speed of a heuristic. Finally we highlight some managerial insights, as solution methods are effective but are not affected by the intuition or the knowledge of the involved decision makers.

The paper is organized as follows. In Section 4.2, an exhaustive literature review is provided. In Section 4.3, the three different perturbations faced by *SWB* are first exposed, then a formal description of (P) is proposed, and finally an ILP (integer linear programming) formulation in depicted. Section 4.4 describes the different solution methods used to provide a solution $S$, and the simulator designed to evaluate the quality of a solution $S$. The experiments are presented in Section 4.5. Section 4.6 concludes the paper and identifies possible future works.

## 4.2 Literature review

As explained in [49], the Swiss industry lost control on the watch industry in the 90's, due to the different actors gaining power in the technological field. But, as of today, Swiss luxury watches are selling in an increasing fashion each year [93], which shows how the Swiss luxury industry was able to overcome its past drawbacks to return to expansion. Switzerland is not the only country facing luxury expansion. In [19], the authors state that in Italy, the luxury segment was worth 26 billion euros per year in 2006, and is increasing. The paper aims at investigating 12 luxury Italian firms of different luxury segments, to define the behaviors of each segment in the supply chain strategies. Extending the previous paper, the authors of [20] propose to derive different clusters of companies, and identify which supply chain strategy is applied in each cluster. In [21], a study is proposed on how luxury companies are adept of pretending to sell rare and exclusive products, even if the quantity of each product is so important that it does not belong to rarity anymore.

Usually, the actors of the Swiss watch industry started with a small family-sized company and gained world-class profits, focusing on the products and having only basic knowledge of the supply chain management. This trend tends now to be reversed and many companies are trying to optimize their supply chain systems. In [58], a two-level assembly problem is tackled using a multi-objective approach based on genetic algorithms. Uncertainties appear in lead times and the main objective is to maximize the service level from the client point of view. In [116], a known demand and uncertain lead times are expected, where the perturbations of the lead times depend on the season. In such a situation, heuristics are proposed and their performances are analyzed. Extending the previous paper, perturbations on the lead times are again expected in [117], but the company faces a planned shutdown period. Simulation is used to assess the quality of the proposed solution methods.

As also discussed in the above papers, to correctly assess the quality of the models and methods, simulation has to be used. With regard to simulation, different interesting surveys can be found in [5, 69, 125]. A reverse problem is tackled in [65], where defective products must be brought back to the factory from the client level. The simulator is used to assess the robustness of the presented model. A simulator is used in [29] to tackle an inventory deployment problem using a genetic algorithm. The robustness of the algorithm is tested with different supply chain settings.

In [81], a multi-level inventory management problem is tackled, where suppliers, warehouses and retailers are considered, and transportation costs are encountered. Centralized and decentralized ordering models are assessed through numerical experiments. A two-level inventory problem with one warehouse and many retailers is proposed in [36], where the retailers face different compound Poisson demand processes, and the facilities apply order-up-to-S replenishment policies. A very similar problem is proposed in [3], where simple recursive procedures are proposed to evaluate the shortage costs. In [4], distribution systems with stochastic demands are analyzed using simulation. Advantages of echelon stock [24] and installation stock [70, 87] are assessed. In [114], a mathematical formulation is presented for a capacity constrained multistage inventory and production control problem. Numerical experiments for a trivial version of the problem are presented, and links among other problems are proposed (e.g., Kanban [7], Cover-Time planning [113]). In [127], three different inventory strategies are designed for a two-level inventory deployment problem, and managerial insights are provided. In [122], the authors propose methods to assess the performance of different multi-level inventory strategies.

In [85], an interesting survey is proposed on facility location and supply chain management, in addition to recent advances in optimization techniques for these problems and for inventory deployment. Finally, a compelling survey can be found in [97], where algorithms and systems are discussed, whereas [129] proposes general guidelines on metaheuristics to efficiently design them depending on the encountered problem.

## 4.3 Presentation of problem (P)

In this section, the different perturbation types are first explained, followed by a formal problem description of (P), and finally an ILP model.

### 4.3.1 Perturbations

As explained earlier, the complex process of conceiving watches suffers multiple uncertainties. Three different perturbations are exposed, namely: perturbations on the demand, perturbations on the production plan, and finally perturbations on the dispatching of the items at the wholesalers' level. The intervals in which the perturbations parameters belong to are set thanks to the *SWB* deep knowledge of their supply chain. The sensitivity analysis performed in Section 4.5 will assess the robustness of such assumptions.

#### Perturbations on the demand

There is obviously a gap between the forecasted and the actual demands encountered at the shops' level. For *SWB*, the clients are wealthy persons, who wants the item at the time they decide to buy it. A relevant way to model this behavior is to use a normal distribution with different means and standard deviations. Thus, each shop has its own mean and standard deviation for each product, used to generate the demand for each week. The means and standard deviations are set with the help of *SWB*, to accurately reflect the reality.

#### Perturbations on the production plan

For various reasons (suppliers being unreliable, problems in raw material deliveries, etc.), two different types of perturbation on the production plan can occur every week. The first one sees some products being not produced at all (because the associated raw material was not delivered), whereas the second sees some items of some products being not produced (as the production was slower than expected). Let $\hat{p}_t^i$ be the planned number of items of product $i$ to produce at week $t$, and $p_t^i$ be the corresponding actual number (i.e., with the perturbation).

Each week, a maximum of $\delta_1$ percent of models are not produced, due to non deliveries from suppliers (i.e., $p_t^i = 0$ for these models, instead of $p_t^i = \hat{p}_t^i$). The perturbation affects two weeks, and the corresponding production must be performed later. Thus $p_t^i = p_{t+1}^i = 0$, and from a practical standpoint, we assume $p^i_{t+2} = \hat{p}^i_{t+2} + \hat{p}^i_t + \hat{p}^i_{t+1}$. *SWB* proposes that $\delta_1$ belongs to interval $[5, 15]$.

On the remaining $1-\delta_1$ percent of the models (i.e., the ones which are produced), each model has $\delta_2$ percent of risk to be slowed down. Slowed down means that at most $\delta_3$ percent of $\hat{p}^i_t$ is not produced at week $t$ but at week $t+1$. At the end of week $t$, the total number of items not produced due to slowed down performances is $Q_t = \hat{P}_t - P_t$, where $\hat{P}_t = \sum_i \hat{p}^i_t$ and $P_t = \sum_i p^i_t$. On the total number $\hat{P}_t$ of items expected to be produced at week $t$, a given threshold of $\varepsilon$ percent of items are allowed to be slowed down, meaning that if $Q_t \leq \varepsilon \cdot \hat{P}_t$, then the factory does not take any action. But if $Q_t > \varepsilon \cdot \hat{P}_t$, then $C_t = Q_t - \varepsilon \cdot \hat{P}_t$ watches must be compensated (meaning that items of different models are produced instead of planned ones) with the following rule: models which are not slowed down at week $t$ and planned at week $t+1$ are eligible for compensation, and the models which were not expected to be produced at week $t$ (i.e., $\hat{p}^i_t = 0$), but are expected to be produced at week $t+1$, can be eligible for compensation as well. For the eligible models, some watches (randomly selected, one at a time) have to be produced at week $t$, until quantity $C_t$ is fulfilled, as explained further in Subsection 4.4.2. *SWB* proposes the following intervals for the above discussed parameters: $\delta_2 \in [45, 55]$, $\delta_3 \in [5, 15]$, $\varepsilon \in [5, 15]$.

#### Perturbations on the dispatching from the wholesalers

As shown in Figure 4.1, the supply chain network linking the factory, the different wholesalers (WS) and the shops is a three-level inventory model. The wholesalers belong to *SWB*, and are the only customers regarding *SWB*. It is expected that the different wholesalers manage the received inventory and deliver items to their corresponding shops in a reliable manner, but unfortunately perturbations occur. The perturbation which can happen is the following: *SWB* estimates a target inventory for each final shop, and sends the corresponding items to the associated wholesaler. Each shop $j$ has a priority $w_j$ and let $w^{max}$ be the largest possible priority (i.e., the priority of the more important shops). Any wholesaler can perturb the *SWB* estimations by sending the items dedicated to a given shop $j$ to a different shop $j'$. *SWB* assumes that $\delta_4$ percent of the items does not go to the expected shop. On these $\delta_4$, $\delta_5$ percent would go to a shop with a lower priority. *SWB* proposes to use $\delta_4 \in [30, 50]$ and $\delta_5 \in [60, 80]$.

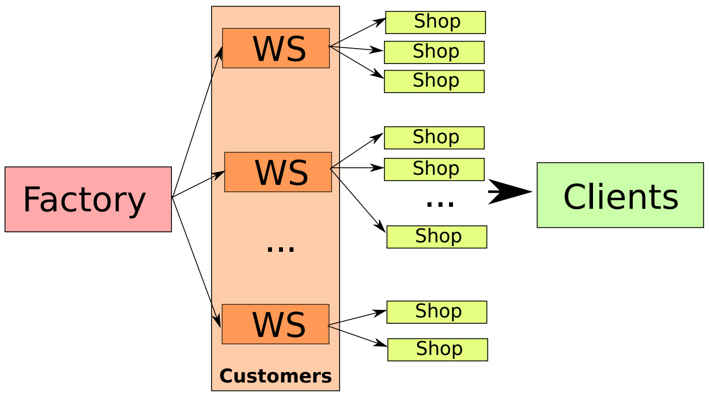

Figure 4.1: Inventory deployment network

### 4.3.2 Formal description

To deeply understand (P), we first propose a formal description, followed by an exact method which could be used to solve (P) without perturbation. $N$ models (i.e., products) are produced in the factory. The number of weeks in a year is $W$. Each week $t$, $p_t^i$ items of model $i$ arrive to the different wholesalers around the world (i.e., all items are sent every week, none are kept in the factory inventory). $J$ shops must be supplied by the different wholesalers, and each shop $j$ is associated with exactly one wholesaler $m_j$ (thus, there can be different $j, j'$ for which $m_j = m_{j'}$). The total number of wholesalers is $M$. Let $I_t^{i,m_j}$ be the on-hand inventory of wholesaler $m_j$ regarding model $i$ at week $t$. Each wholesaler delivers to its shops and tries to empty its inventory. Let $\hat{x}_t^{i,j}$ be a decision variable describing the number of items of model $i$ supposed to arrive at shop $j$ at week $t$ from the corresponding wholesaler $m_j$. As explained in Subsection 4.3.1, each variable $\hat{x}_t^{i,j}$ suffers a perturbation, and let $x_t^{i,j}$ be the corresponding variable with the perturbation. In other words, $\hat{x}_t^{i,j}$ (resp. $x_t^{i,j}$) is an expected (resp. actual or observed) value. Moreover, a lead time of $L_{m_j}$ weeks occurs from the plant to the wholesaler $m_j$. Another lead time of $L_j^{m_j}$ weeks is then required to reach the shop $j$ from the corresponding wholesaler $m_j$. For each week $t$, a demand $\hat{d}_t^{i,j}$ is forecasted for model $i$ at shop $j$, and is subject to the perturbation depicted in Subsection 4.3.1. Thus, let $d_t^{i,j}$ be the corresponding actual demand. As soon as the items $x_t^{i,j}$ reaches the shop $j$, the corresponding on-hand inventory of shop $j$ is updated: $I_t^{i,j} = \max(0, I_{t-1}^{i,j} - d_{t-1}^{i,j} + x_t^{i,j})$ (we also have the expected updating: $\hat{I}_t^{i,j} = \max(0, \hat{I}_{t-1}^{i,j} - \hat{d}_{t-1}^{i,j} + \hat{x}_t^{i,j})$). Thus, the inventory at period $t$ is set as the inventory at period $t-1$ augmented by the number of delivered items and decreased by the demand at period $t-1$. A solution $\hat{S}$ can be denoted $\hat{S} = (\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_t, \ldots, \hat{S}_W)$, which is a per week list of $\hat{S}_t$'s, where $\hat{S}_t = (\hat{S}_t^1, \hat{S}_t^2, \ldots, \hat{S}_t^i, \ldots, \hat{S}_t^N)$ is the corresponding solution for week $t$ for each product $i$. $\hat{S}_t^i = (\hat{x}_t^{i,1}, \hat{x}_t^{i,2}, \ldots, \hat{x}_t^{i,j}, \ldots, \hat{x}_t^{i,J})$ is a vector of decision variables for each shop $j$, each week $t$ and each product $i$.

The overall objective function $f$ involves three different components $f_1$, $f_2$ and $f_3$ to minimize in a lexicographic order (i.e., no deterioration on $f_i$ can be compensated by improvements on $f_{i+1}$), as motivated in [123]. $f_1$ is the expected shortage penalty of item $i$ at week $t$ in shop $j$, which happens if $\hat{I}_t^{i,j} < \hat{d}_t^{i,j}$. Let $\hat{B}_t^{i,j} = \max(0, \hat{d}_t^{i,j} - \hat{I}_t^{i,j})$ be the expected shortage quantity of model $i$ at week $t$ in shop $j$, and $\hat{v}_t^{i,j}$ be a binary value which is 0 if $\hat{x}_t^{i,j} > 0$, and 1 otherwise. Then, $f_1$ can be described as in Equation (4.1). It is the sum of the shortage penalties, where a shortage which has started a long time ago costs more than a recent shortage. $SWB$ assumes that if a watch of model $i$, which is assumed to be in shortage, arrives in a shop $j$ at week $t$, then the considered shortage penalty is reset to 0 from that time period, even if the demand is above the resulting on-hand inventory.

$$f_1 = \sum_{j=1}^{J} w_j \sum_{t=1}^{W} \sum_{t'=1}^{t-1} (t - t') \sum_{i=1}^{N} \hat{B}_{t'}^{i,j} \cdot \hat{v}_t^{i,j} \tag{4.1}$$

Let $\hat{y}_t^{i,j} = 1$ if $\hat{I}_t^{i,j} = 0$, and $\hat{y}_t^{i,j} = 0$ otherwise. Objective $f_2$ is described in Equation (4.2), which is a measure of the rarity of each model $i$ in each shop $j$ for each week $t$.

$$f_2 = \frac{1}{J \cdot W \cdot N} \cdot \sum_{t=1}^{W} \sum_{j=1}^{J} \sum_{i=1}^{N} \hat{y}_t^{i,j} \tag{4.2}$$

The last objective $f_3$ is described in Equation (4.3), which is a weighted inventory penalty.

$$f_3 = \sum_{j=1}^{J} \left[ (w^{max} + 1) - w_j \right] \left[ \sum_{t=1}^{W} \sum_{i=1}^{N} \hat{I}_t^{i,j} \right] \tag{4.3}$$

To ensure a better understanding of the components involved in the objective function, we now propose an example. Consider an instance with $J = 1$ shop (with $w_1 = 1$ and thus $w^{max} = 1$), $N = 1$ product, $W = 4$ weeks, and the initial inventory at the shop is empty (i.e., $\hat{I}_0^{1,1} = 0$). Furthermore, consider solution $\hat{S}$ given by $\hat{x}^{1,1} = (1, 2, 0, 3)$. Assume that the forecasted demand is $\hat{d}^{1,1} = (1, 3, 0, 1)$. Thus, $f_1 = 0 + 1 + 2 + 0 = 3$, as the shortage of week 2 costs double at week 3, and is reset to 0 in week 4 as a unit of this product reaches the shop. We can easily compute that $f_2 = (\frac{1}{1 \cdot 1 \cdot 4}) \cdot (1 + 1 + 1 + 0) = \frac{3}{4}$ and $f_3 = (2 - 1) \cdot (0 + 0 + 0 + 2) = 2$.

### 4.3.3 Exact model

$f_1$ contains the multiplication of two decision variables, and is thus nonlinear. In order to use a linear solver, such as CPLEX or Gurobi, and to benchmark our results, we propose now a linear model for (P), which necessitates a slight modification of $f_1$ to be linearized as proposed in Equation (4.4). $f_1'$ is based on $f_1$ but does not reset the penalty if at least one item arrives at week $t$. Therefore, it is a relevant approximation of the shortage penalty.

$$f_1' = \sum_{j=1}^{J} w_j \sum_{t=1}^{W} \sum_{t'=1}^{t-1} (t - t') \sum_{i=1}^{N} \hat{B}_{t'}^{i,j} \tag{4.4}$$

Let $\alpha, \beta$ and $\gamma$ be the coefficients which guarantee the lexicographic approach. The objective function can then be formulated in Equation (4.5). The constraints are given in Equations (4.6) to (4.10).

$$\min f = \alpha \cdot f_1 + \beta \cdot f_2 + \gamma \cdot f_3 \tag{4.5}$$

$$\hat{B}_t^{i,j} \geq \hat{d}_t^{i,j} - \hat{I}_t^{i,j} \quad \forall\, t, i, j \tag{4.6}$$

$$\hat{I}_t^{i,j} \geq 1 - \hat{y}_t^{i,j} \quad \forall\, t, i, j \tag{4.7}$$

$$\hat{I}_t^{i,j} \geq \hat{I}_{t-1}^{i,j} - \hat{d}_{t-1}^{i,j} + \hat{x}_t^{i,j} \quad \forall\, t, i, j \tag{4.8}$$

$$\sum_j \hat{x}_t^{i,j} \leq \hat{p}^i_{t-L_{m_j}-L_j^{m_j}} + \sum_j \hat{I}^{i,m_j}_{t-L_j^{m_j}} \quad \forall\, t, i \tag{4.9}$$

$$\hat{x}_t^{i,j}, \hat{B}_t^{i,j}, \hat{I}_t^{i,j} \in \mathbb{N} \quad \forall\, t, i, j \tag{4.10}$$

Constraints (4.6) ensure a correct shortage computation by setting $\hat{B}_t^{i,j}$ above $\hat{d}_t^{i,j} - \hat{I}_t^{i,j}$. Thanks to the minimization of $f$, this replaces the formulation of

$\hat{B}_t^{i,j} = \max(0, \hat{d}_t^{i,j} - \hat{I}_t^{i,j})$. Constraints (4.7) and (4.8) ensure a correct inventory computation. The inventory is updated using $\hat{d}_{t-1}^{i,j}$ and $\hat{x}_t^{i,j}$, and is used to set $\hat{y}_t^{i,j}$ correctly ($\hat{y}_t^{i,j} = 1$ if $\hat{I}_t^{i,j} = 0$, and using the minimization of $f$, $\hat{y}_t^{i,j} = 0$ otherwise). Constraints (4.9) ensure that non existent items are not created. Finally, Constraints (4.10) ensure that $\hat{x}_t^{i,j}, \hat{B}_t^{i,j}$ and $\hat{I}_t^{i,j}$ are not below 0.

## 4.4 Solution methods and simulator

To evaluate the actual value of a solution, it is mandatory to develop a simulator. The input data is the solution, the production plan, the initial inventories (at each shop and each wholesaler, for each product), and the forecasted demand. As shown in Figure 4.2, a solver is first used to provide a solution $S$, which is then evaluated with the simulator. At the end, a manager of $SWB$ can modify and evaluate again the provided solution, based on his/her expertise and the non modeled elements. To produce a solution, two types of methods are available in the solver: various constructive heuristics and the exact model (proposed in Subsection 4.3.3) using CPLEX. Unfortunately, the exact model is limited to instances with up to $N = 24$ models of watches, $J = 30$ shops and $M = 4$ wholesalers, due to exponential solving times. Therefore, for realistic instances, only heuristics are able to produce solutions in a fair amount of time. Note that there is no uncertainty in the solver. Therefore, all the data is deterministic at that stage. Only the simulator deals with random events by adding the disturbances proposed in Subsection 4.3.1.

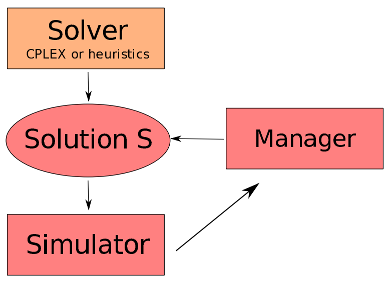

Figure 4.2: Sketch of the decision analysis tool

An appropriate time-limit to get a solution (including its simulation), regardless of the method, was set to 30 minutes by $SWB$. Therefore, it forbids the use of advanced metaheuristics which would require more time to perform well. We will see in Subsection 4.4.3 that a simulation requires roughly three minutes on the used computer. For these reasons, constructive methods are very appropriate to tackle (P).

In the remaining part of this section, three solution methods are proposed in Subsection 4.4.1, the simulator is presented in Subsection 4.4.2, and important computing time aspects are discussed in Subsection 4.4.3.

### 4.4.1 Solution methods

We design three solution methods, which rely on a population of *pop* (parameter) solutions and use a constructive heuristic called *SmartPriorityGreedy* (denoted $SPG$) as an underlying method. $SPG$ is depicted in Algorithm 5, where $A_t^{i,w}$ is a set that contains all the shops with priority $w$ that have a strictly positive demand for week $t$ and model $i$, and $A$ contains all the shops with the highest priority $w^{max}$ (regardless of the demand). For each week $t$ and each model $i$, the available items are the sum of the produced items in the factory and the items available at the wholesalers' level. At each step, the algorithm attributes one item to a shop. After intensive tests, the most efficient method (according to speed and results' quality) consists in randomly selecting the shop to which an item is assigned in the considered set $A_t^{i,w}$. Using more advanced methods (using $f_1$, $f_2$ and $f_3$ for example) dramatically slows down the overall process, without improving the quality of the solutions. As an example, on an instance with $N = 143$, $W = 47$, $J = 192$ and $M = 9$, a method which computes $f_1$, $f_2$ and $f_3$ at each step to attribute the best item to the best shop requires more than 15 hours to return a solution! Here, the efficiency of $SPG$ consists in dealing with the $A_t^{i,w}$'s. As $f_1$ and $f_3$ are very influenced by the weights of the shops, creating the $A_t^{i,w}$'s allows to very quickly and efficiently minimize these objectives. If all the demands of model $i$ of all the shops have been fulfilled for the considered week $t$, the last phase of $SPG$ is triggered (see the last part of step (3c) in Algorithm 5). This phase consists in attributing the remaining items to the shops with priority $w^{max}$.

**Algorithm 5** SmartPriorityGreedy (*SPG*)

```
Input: production plan, forecasted demand, inventory levels (at both shops'
and wholesalers' levels), shops' priorities, set $A$

For each week $t$ and each model $i$, do
    1. Compute the sets $A_t^{i,w}$ (for each $w$).
    2. Select the set $A_t^{i,w}$ with the highest priority $w$ as the current set $\mathcal{C}$.
    3. For each available item, do
        (a) Assign the item to a randomly chosen member of $\mathcal{C}$.
        (b) If a member of $\mathcal{C}$ has its demand fulfilled, remove it from $\mathcal{C}$.
        (c) If $\mathcal{C} = \emptyset$, go to step (2), except if all the $A_t^{i,w}$'s have been emptied,
            set $\mathcal{C} = A$.

Output: inventory deployment solution $S$.
```

The first solution method, called *BestSolution* (denoted *BS*), generates a set of *pop* solutions using *SPG*, and returns the best one (according to $f$). Following this method, *BestSolutionSimulated* (denoted *BSS*) generates a set of *pop* solutions with *SPG*, simulates the $k$ (parameter) best ones (according to $f$), and finally returns the solution with the best simulated value. Finally, *BestSolutionSimulatedClique* (denoted *BSSC*) is a matheuristic [17].

To properly understand *BSSC*, let us first introduce a distance function $dist(\hat{S}_1, \hat{S}_2)$ which returns the distance between two solutions $\hat{S}_1$ and $\hat{S}_2$. $dist(\hat{S}_1, \hat{S}_2) = \sum_{t,i,j} w_j \cdot \lambda_t^{i,j}$, assuming that $\hat{x}_t^{i,j}(\hat{S})$ is the value of $\hat{x}_t^{i,j}$ in solution $\hat{S}$, and $\lambda_t^{i,j}$ is set as in Equation (4.11). Therefore, this distance function assumes that two solution components are distant if one is zero and the other is strictly positive (regardless of the amplitude of the positiveness), which is relevant in the context of *SWB*, which wants at least one item of each product available in each shop.

$$\lambda_t^{i,j} = \begin{cases} 1 \text{ if } \left[(\hat{x}_t^{i,j}(\hat{S}_1) = 0) \text{ and } (\hat{x}_t^{i,j}(\hat{S}_2) > 0)\right] \text{ or if } \left[(\hat{x}_t^{i,j}(\hat{S}_2) = 0) \text{ and } (\hat{x}_t^{i,j}(\hat{S}_1) > 0)\right] \\ 0 \text{ otherwise} \end{cases} \tag{4.11}$$

Let $G = (V, E)$ be a graph with vertex set $V$ and edge set $E$. A clique $C$ of $G$ is a subset of $V$ such that its vertices are pairwise adjacent. In addition, let $w(v, v')$

be the weight associated with edge $(v, v') \in E$. We define here the weight $w(C)$ of a clique $C$ by $\frac{1}{2} \cdot \sum_{v,v' \in C} w(v, v')$. $\frac{1}{2}$ ensures that the weights are not counted twice. Further, the *k-clique with maximum weight* problem consists in searching the clique $C$ of size $k$ in $G$ which maximizes $w(C)$. Surveys on graph theory and cliques can be found in [18, 95, 111]. This problem can be exactly solved with an integer linear program (ILP) using CPLEX. Let $x_v = 1$ if vertex $v \in C$, and $x_v = 0$ otherwise. Let $y_{v,v'} = 1$ if $x_v = x_{v'} = 1$, and $y_{v,v'} = 0$ otherwise. The ILP model is given in Equations (4.12-4.15). Constraints (4.13) and (4.14) are used to compute correctly the value of $y_{v,v'}$, and Constraints (4.15) ensure that the clique size is $k$.

$$\max \sum_{(v,v') \in E} w(v, v') \cdot y_{v,v'} \tag{4.12}$$

$$x_v + x_{v'} - 1 \leq y_{v,v'} \quad \forall\, v, v' \tag{4.13}$$

$$2 \cdot y_{v,v'} \leq x_v + x_{v'} \quad \forall\, v, v' \tag{4.14}$$

$$\sum_{v} x_v = k \tag{4.15}$$

*BSSC* starts by generating a set of *pop* solutions with *SPG*, then builds a set $B$ with the best $b$ (parameter) generated solutions, and on this set, computes distance between each pair of solutions. The above ILP clique problem is then solved to optimality with CPLEX (a vertex $v$ represents a solution $S_v \in B$ and $w(v, v') = dist(S_v, S_{v'})$). It finally simulates only the $k$ solutions associated with the provided clique. At the end, *BSSC* returns the solution with the best simulated value. The clique approach used in *BSSC* allows to select the solutions that differ the most with respect to their structures. In addition to being efficient, this technique allows decision makers to select a solution among highly different solutions.

## 4.4.2 Simulator

The simulator returns a value $f(S)$ for a solution $S$ after having added the different perturbations to the corresponding solution without perturbation $\hat{S}$ (as explained in Subsection 4.3.1) in that order: generate the demand, perturb the production plan and the dispatching. Algorithm 6 proposes a sketch of the different simulation steps. After the second step, which perturbs the production plan, the solution $\hat{S}$ could possibly not be feasible anymore, as the $\hat{p}^i_t$'s could be different from the $p^i_t$'s (i.e., some items are unfortunately not produced, and others are produced in counterpart). Thus, a repairing step is performed, which ensures that the resulting solution remains feasible. To repair a solution, $f^j = \alpha \cdot f^j_1 + \beta \cdot f^j_2 + \gamma \cdot f^j_3$ is computed for each shop $j$, and the resulting $f^j$'s are sorted in increasing fashion. If for a given week, a $\hat{x}^{i,j}_t$ has to be reduced (because of the production plan perturbations, some watches do not exist anymore), then they are one by one removed from the shops with the smallest $f^j$ value. On the opposite, if some items must be added (because of the compensation production), then they are added one by one to the shops with the largest $f^j$ value. A pseudo-code of the repair procedure is proposed in Algorithm 7. Note that even after the perturbations on the dispatching, solution $S$ remains feasible because the same set of watches is managed.

At the very end, the simulator returns a value $f(S)$ for the input solution $\hat{S}$. To properly assess the operating mode of the simulator, let us propose a small example. Assume that an instance involves one product, two weeks, one wholesaler, and two shops (with $w_1 = 2$, $w_2 = 1$, and thus $w^{max} = 2$). All initial inventories are empty, and all the lead times are zero. The production plan is $\hat{p}^1 = (2, 3)$ and the forecasted demand $\hat{d}^{1,1} = (1, 3)$ and $\hat{d}^{1,2} = (1, 1)$. The solver, based on the expected production plan and demand, proposes a solution $\hat{S}$ with $\hat{x}^{1,1} = (1, 3)$ and $\hat{x}^{1,2} = (1, 0)$, as it fulfills in priority the demand for the first shop. Then, the demand is generated as $d^{1,1} = (1, 2)$ and $d^{1,2} = (2, 1)$, whereas the production plan is generated as $p^1 = (3, 3)$ (we produce more than expected, due to a perturbation which occurred before the current events). The solution then needs to be repaired, and the system returns a modified solution $\hat{x}^{1,1} = (1, 2)$ and $\hat{x}^{1,2} = (2, 1)$, which is optimal. But the behavior of the wholesaler perturbs the solution such that $x^{1,1} = (1, 3)$ and $x^{1,2} = (2, 0)$. Finally the objective function is computed as $f(S) = 1 + \frac{1}{4} + 1 = \frac{9}{4}$ (using $\alpha = \beta = \gamma = 1$).

In order to be robust, *sim* (parameter) runs of the above described simulator are performed with solution $\hat{S}$ as input. The value $f(S)$ is then the average value over the *sim* runs. Let $f_{sim}(S)$ be the average value of $f(S)$ after *sim* runs of the simulator. Parameter *sim* was tuned such that $f_{sim}(S)$ and $f_{sim-1}(S)$ differ by less than 1‰. After intensive tests on various instances, we have set

$sim = 40$.

**Algorithm 6** Sketch of the simulator

```
Input: a solution $\hat{S}$, the forecasted demands and the corresponding normal distributions, the inventories of the wholesalers and shops, the production plan, the shops' priorities.
    1. Generate the demands (using a normal distribution) for each shop/model/week.
    2. Perturb the production plan.
    3. If $\hat{S}$ is not feasible, perform the repair procedure (see Algorithm 7).
    4. Perturb the dispatching (i.e., generate $S$ from $\hat{S}$).
    5. Compute $f(S)$.
Output: $f(S)$.
```

**Algorithm 7** Repair procedure

```
Input: a non feasible solution $\hat{S}$, the perturbed production plan.
For each week $t$ and each product $i$ do
    1. Get the number of watches $E_t^i = p_t^i - \hat{p}_t^i$ that does not fit the perturbed production plan.
    2. If $E_t^i < 0$, remove watches (one by one) from the shops with the lowest $f^j$ value (i.e., reduce some $\hat{x}_t^{i,j}$ values to generate the corresponding $x_t^{i,j}$).
    3. If $E_t^i > 0$, add watches (one by one) to the shops with the highest $f^j$ value (i.e., augment some $\hat{x}_t^{i,j}$ values to generate the corresponding $x_t^{i,j}$).
Output: a repaired solution $S$ (feasible).
```

### 4.4.3 Computing time considerations

Remind that the time limit allowed by *SWB* is 30 minutes. A typical instance of *SWB* contains 47 weeks, between 7 and 9 wholesalers, between 160 and 250 shops, and between 100 and 200 models (these intervals are voluntarily consequent, to respect a $SWB$ non-disclosure agreement). $SPG$ requires half a second to get a solution. A complete simulation (i.e., $sim = 40$ runs of the simulator on one solution) requires around three minutes.

Based on such constraining time considerations, preliminary experiments lead to the following parameter setting: $pop = 100$, $b = 25$, and $k = 5$. With such values, CPLEX requires about 5 minutes to optimally solve the ILP clique problem (see Equations (4.12) to (4.15)). $BSS$ and $BSSC$ need between 20 and 30 minutes to complete, whereas $BS$ requires only 15 seconds.

## 4.5 Results

Tests were performed on an Intel Quad-core i7 @ 3.4 GHz with 8 GB DDR3 of RAM memory. Subsection 4.5.1 describes the instances used to evaluate the quality of the different solution methods. Subsection 4.5.2 shows the experiments on the realistic instances, whereas Subsection 4.5.3 discusses the results of the exact method proposed in Subsection 4.3.3. Subsection 4.5.4 conducts a sensitivity analysis of the perturbation parameters. Finally, Subsection 4.5.5 highlights some managerial insights.

### 4.5.1 Generation of the instances

Based on the various data provided by $SWB$, we have developed an instance generator, which is able to quickly generate some realistic instances. An instance is composed of the following parameters: the number $N$ of models, the number $W$ of weeks in a year, the number $J$ of shops, the number $M$ of wholesalers, the perturbation parameters ($\delta_1$, $\delta_2$, $\delta_3$, $\delta_4$, $\delta_5$ and $\varepsilon$), the production plan, the demand (following a normal distribution with mean 0 and standard deviation $\sigma$), the lead times (from the factory to the wholesalers, and from the wholesalers to the shops), the shop priorities, the links wholesalers-shops, the inventory levels at the wholesalers and at the shops. The parameters were reasonably fixed after discussions with $SWB$. In the *reference* instance, we use $N = 143$, $W = 47$, $J = 192$, $M = 9$, $w^{max} = 3$, $\delta_1 = 10$, $\delta_2 = 50$, $\delta_3 = 10$, $\delta_4 = 40$, $\delta_5 = 70$, $\varepsilon = 10$.

We have generated a total of 112 instances around the *reference* instance by varying $\delta_1$, $\delta_2$, $\delta_3$, $\delta_4$, $\delta_5$ and $\varepsilon$.

### 4.5.2 Results on realistic instances

Table 4.1 shows the results for the three solution methods when $\delta_1$, $\delta_2$, $\delta_3$, $\delta_4$, $\delta_5$, and $\varepsilon$ vary, and $\sigma = 2$ (larger values of $\sigma$ are totally not realistic for *SWB*). Two different values are presented for each parameter, the first is lower than the *reference* value, and the second is the *reference* value. The goal is to compare the performances of the algorithms and the impact of lower perturbations on the objectives. The first six columns give the values of each parameter. The seventh column, called $f_1^\star$, is the best $f_1$ value returned by one of the algorithm on the considered instance. The same applies for the next two columns regarding $f_2$ and $f_3$. Starting at column ten, the next three columns show the gaps between the $f_i^\star$'s and the $f_i$'s for method *BS*. The same information is then given for *BSS* and *BSSC*. The last two rows show the number of times that the corresponding algorithm found a gap of zero, followed by the average values of the different gaps (provided in ‰). Note that the first row is the instance with the lowest perturbations, and the last row shows the instance with the largest perturbations (which is the *reference* instance).

The results shows that *BSSC* outperforms *BSS* on both $f_1$ and $f_3$, whereas *BS* is significantly overpowered by both *BSS* and *BSSC*. Regarding $f_2$, *BSS* finds zero gaps 29 times, versus 15 times for *BSSC*, but the gaps for *BSSC* are so small when *BSS* finds zero gaps that no conclusion can come out. Moreover, as $f_1$ has the priority over $f_2$, it is not surprising that if *BSSC* beats *BSS* on $f_1$, then *BSS* can sometimes outperform *BSSC* on $f_2$. When comparing the last row, the averages over the complete set of instances clearly show that *BSSC* is the most efficient method, as it shows the lower percentages for $f_1$ and $f_3$, and very low gaps for $f_2$.

Regarding the impact of a single parameter variation (for example $\delta_1 \in \{5, 10\}$), and leaving the other parameters unchanged, we can see that the impacts on $f_1$ and $f_3$ are significant, but not on $f_2$. The reasons to this are detailed in Section 4.5.4. The parameters that induce the greater reductions on $f_1$ and

$f_3$ are obviously $\delta_4$ and $\delta_5$. Both parameters are used when encountering the perturbations involved by the wholesalers' behaviors. The improvements are such that it would be very important for $SWB$ to reduce the perturbations by either finding more reliable suppliers (for $\delta_1, \delta_2, \delta_3$) or by having a better control on the wholesalers (for $\delta_4, \delta_5$). To properly assess the impact of such reductions, Subsection 4.5.4 proposes a more detailed sensitivity analysis.

To measure the impact of the perturbation parameters on a solution $\hat{S}$, we propose to use $Dist(\hat{S}, S) = \sum_{t,i,j} \lambda_t^{i,j}$, with $\lambda_t^{i,j}$ defined as in Subsection 4.4.1. Function $Dist$ measures the structural difference between the expected solution $\hat{S}$ and the actual solution $S$ (after simulation). To better capture the impact of the perturbations, we propose to compute such a distance value for each priority level. More precisely, at the end of each single run of the simulation, the distance difference (in %) between $\hat{S}$ and the returned actual solution is computed for each priority level (i.e., each gap is divided by $J^w \cdot W \cdot N$, where $J^w$ is the number of shops with priority $w$). These gaps are then averaged over the $sim$ runs. Such a test was performed 100 times on two different instances and the results are averaged for each priority level. The used instances are $(\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, \epsilon) \in \{(5, 40, 5, 30, 60, 5), (10, 50, 10, 40, 70, 10)\}$ (i.e., the instances with the smallest and largest perturbations, respectively). The resulting gaps are presented with the notation ($w^1$,$w^2$,$w^3$), where $w^i$ is the gap corresponding to the shops with priority $i$. The obtained gaps are (2.92%, 3.11%, 3.69%) for the least perturbed instance, and (3.15%, 3.50%, 4.49%) for the *reference* instance. As an example, the shops with priority 3 have a gap of 4.49% on the *reference* instance. The results show that the larger gaps concern the most important shops, which is consistent as such shops are the least represented in the distribution network.

### 4.5.3 Results on smaller instances

To ensure that our algorithms are reliable, we propose to conduct a comparison with the exact ILP method (see Subsection 4.3.3) tested with AMPL and CPLEX 12.5. The ILP is compared with *BSSC* (which also uses $f_1'$ as the first objective, instead of $f_1$) on 25 different instances, with $W = 47$ (as we cannot change the number of weeks in a year). CPLEX is tested with a time limit of 3 hours per objective and a memory limit of 7 GB. Note that the memory limit

Table 4.1: Results (percentage gaps) on realistic instances

| $\delta_1$ | $\delta_2$ | $\delta_3$ | $\delta_4$ | $\delta_5$ | $\varepsilon$ | $f_1^\star$ | $f_2^\star$ | $f_3^\star$ | $BSf_1$ | $BSf_2$ | $BSf_3$ | $BSSf_1$ | $BSSf_2$ | $BSSf_3$ | $BSSCf_1$ | $BSSCf_2$ | $BSSCf_3$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 40 | 5 | 30 | 60 | 5 | 93,856,809.67 | 103.84 | 6,426,820.70 | 0.42% | 0.12% | 0.04% | **0.00%** | **0.00%** | 0.53% | 0.11% | 0.04% | **0.00%** |
| 5 | 40 | 5 | 30 | 60 | 10 | 93,611,706.42 | 104.04 | 6,393,880.08 | **0.00%** | 0.13% | **0.00%** | 0.66% | **0.00%** | 0.33% | 0.24% | **0.00%** | 0.70% |
| 5 | 40 | 5 | 30 | 70 | 5 | 104,748,487.28 | 103.00 | 8,210,743.70 | **0.00%** | **0.00%** | 0.18% | 0.08% | 0.05% | 0.07% | 0.11% | 0.02% | **0.00%** |
| 5 | 40 | 5 | 30 | 70 | 10 | 104,160,132.28 | 103.05 | 8,188,112.60 | 0.26% | **0.00%** | **0.00%** | **0.00%** | 0.04% | 0.53% | 0.66% | 0.15% | 0.88% |
| 5 | 40 | 5 | 40 | 60 | 5 | 104,584,290.88 | 100.48 | 9,127,778.05 | 1.29% | 0.14% | **0.00%** | 0.25% | **0.00%** | 0.20% | **0.00%** | 0.11% | 0.29% |
| 5 | 40 | 5 | 40 | 60 | 10 | 104,521,898.40 | 100.56 | 9,044,377.57 | 1.00% | **0.00%** | 1.10% | 0.49% | 0.02% | **0.00%** | **0.00%** | 0.03% | 0.74% |
| 5 | 40 | 5 | 40 | 70 | 5 | 118,030,025.88 | 100.49 | 11,122,331.10 | 0.47% | 0.09% | 0.07% | 0.03% | 0.04% | 0.39% | **0.00%** | **0.00%** | **0.00%** |
| 5 | 40 | 5 | 40 | 70 | 10 | 117,299,231.35 | 100.42 | 11,082,133.78 | 1.24% | **0.00%** | 1.12% | 0.82% | 0.06% | 1.03% | **0.00%** | 0.31% | **0.00%** |
| 5 | 40 | 10 | 30 | 60 | 5 | 94,229,919.08 | 103.87 | 6,400,873.60 | 0.12% | 0.18% | 1.46% | **0.00%** | **0.00%** | 0.92% | 0.20% | 0.15% | **0.00%** |
| 5 | 40 | 10 | 30 | 60 | 10 | 93,813,698.90 | 103.90 | 6,434,238.35 | 0.44% | 0.10% | **0.00%** | **0.00%** | **0.00%** | 0.28% | 0.32% | 0.02% | 0.84% |
| 5 | 40 | 10 | 30 | 70 | 5 | 104,678,949.62 | 102.99 | 8,241,697.45 | 0.33% | **0.00%** | **0.00%** | 0.32% | 0.07% | 0.46% | **0.00%** | 0.05% | 0.49% |
| 5 | 40 | 10 | 30 | 70 | 10 | 104,376,624.28 | 102.87 | 8,229,631.92 | 0.82% | **0.00%** | 0.48% | 0.61% | 0.24% | 0.26% | **0.00%** | 0.06% | **0.00%** |
| 5 | 40 | 10 | 40 | 60 | 5 | 104,522,811.25 | 100.53 | 9,098,167.30 | 0.58% | **0.00%** | 0.46% | 0.42% | 0.11% | 0.29% | **0.00%** | 0.14% | **0.00%** |
| 5 | 40 | 10 | 40 | 60 | 10 | 104,718,207.03 | 100.62 | 9,051,360.93 | 0.32% | **0.00%** | **0.00%** | **0.00%** | 0.06% | 0.72% | 0.10% | 0.02% | 0.90% |
| 5 | 40 | 10 | 40 | 70 | 5 | 117,686,732.45 | 100.35 | 11,145,245.72 | 0.36% | 0.17% | **0.00%** | 0.21% | **0.00%** | 0.38% | **0.00%** | 0.13% | **0.00%** |
| 5 | 40 | 10 | 40 | 70 | 10 | 117,642,362.15 | 100.46 | 11,148,556.55 | **0.00%** | **0.00%** | **0.00%** | 0.27% | 0.02% | 0.16% | 0.29% | 0.01% | 0.21% |
| 5 | 50 | 5 | 30 | 60 | 5 | 93,974,135.67 | 103.95 | 6,444,403.80 | 0.03% | 0.07% | 0.21% | **0.00%** | **0.00%** | 0.05% | 0.17% | 0.04% | **0.00%** |
| 5 | 50 | 5 | 30 | 60 | 10 | 94,002,498.80 | 103.86 | 6,443,839.08 | 0.35% | 0.14% | **0.00%** | **0.00%** | **0.00%** | 0.38% | 0.45% | 0.11% | 0.57% |
| 5 | 50 | 5 | 30 | 70 | 5 | 104,492,704.28 | 103.09 | 8,241,887.65 | **0.00%** | 0.08% | 0.49% | 0.25% | 0.03% | **0.00%** | 0.32% | **0.00%** | 0.09% |
| 5 | 50 | 5 | 30 | 70 | 10 | 103,862,333.53 | 102.99 | 8,162,655.60 | **0.00%** | 0.24% | **0.00%** | 0.53% | **0.00%** | 1.16% | 1.05% | 0.24% | 1.01% |
| 5 | 50 | 5 | 40 | 60 | 5 | 105,574,759.25 | 100.44 | 9,125,329.62 | 0.10% | **0.00%** | 0.68% | **0.00%** | 0.10% | 0.49% | 0.08% | 0.18% | **0.00%** |
| 5 | 50 | 5 | 40 | 60 | 10 | 104,700,360.65 | 100.56 | 9,070,600.65 | 0.04% | **0.00%** | **0.00%** | 0.79% | 0.05% | 0.55% | **0.00%** | 0.09% | 0.35% |
| 5 | 50 | 5 | 40 | 70 | 5 | 117,369,288.78 | 100.43 | 11,145,628.20 | 0.58% | 0.07% | **0.00%** | 0.40% | **0.00%** | 0.24% | **0.00%** | 0.02% | 0.45% |
| 5 | 50 | 5 | 40 | 70 | 10 | 116,791,994.05 | 100.37 | 11,100,587.30 | 0.51% | 0.24% | 0.63% | 0.71% | **0.00%** | 0.44% | **0.00%** | 0.24% | **0.00%** |
| 5 | 50 | 10 | 30 | 60 | 5 | 94,075,385.40 | 103.88 | 6,457,223.20 | 0.68% | 0.11% | **0.00%** | 0.41% | **0.00%** | 0.90% | **0.00%** | 0.27% | 0.13% |
| 5 | 50 | 10 | 30 | 60 | 10 | 93,606,039.33 | 103.84 | 6,455,417.00 | 0.58% | 0.14% | 0.02% | **0.00%** | **0.00%** | 0.04% | 0.52% | 0.11% | **0.00%** |
| 5 | 50 | 10 | 30 | 70 | 5 | 104,844,657.33 | 102.96 | 8,250,414.83 | 0.14% | 0.07% | 0.32% | 0.21% | **0.00%** | 0.24% | **0.00%** | 0.17% | **0.00%** |
| 5 | 50 | 10 | 30 | 70 | 10 | 104,136,960.10 | 102.93 | 8,248,901.70 | 1.19% | **0.00%** | 0.50% | 0.50% | 0.08% | 0.21% | **0.00%** | 0.13% | **0.00%** |
| 5 | 50 | 10 | 40 | 60 | 5 | 104,825,043.78 | 100.46 | 9,103,612.25 | 0.39% | **0.00%** | 0.63% | **0.00%** | 0.13% | 0.16% | 0.07% | 0.19% | **0.00%** |
| 5 | 50 | 10 | 40 | 60 | 10 | 105,599,244.03 | 100.42 | 9,136,612.82 | 0.06% | 0.18% | 0.09% | 0.30% | **0.00%** | 0.48% | **0.00%** | 0.16% | **0.00%** |
| 5 | 50 | 10 | 40 | 70 | 5 | 118,241,676.47 | 100.28 | 11,146,366.88 | **0.00%** | 0.26% | 0.65% | 0.01% | **0.00%** | 0.88% | 0.15% | 0.18% | **0.00%** |
| 5 | 50 | 10 | 40 | 70 | 10 | 117,588,781.15 | 100.39 | 11,113,507.18 | 0.25% | **0.00%** | **0.00%** | **0.00%** | 0.13% | 0.18% | 0.22% | 0.11% | 0.21% |
| 10 | 40 | 5 | 30 | 60 | 5 | 94,207,031.50 | 104.34 | 6,529,899.92 | 0.47% | **0.00%** | 1.08% | **0.00%** | **0.00%** | **0.00%** | 0.37% | 0.03% | 0.25% |
| 10 | 40 | 5 | 30 | 60 | 10 | 94,340,721.88 | 104.24 | 6,557,394.03 | 0.31% | 0.11% | 0.31% | **0.00%** | **0.00%** | **0.00%** | 0.15% | 0.21% | 0.29% |
| 10 | 40 | 5 | 30 | 70 | 5 | 105,345,990.90 | 103.23 | 8,249,574.90 | 0.10% | **0.00%** | 0.99% | 0.05% | 0.13% | 1.41% | **0.00%** | 0.11% | **0.00%** |
| 10 | 40 | 5 | 30 | 70 | 10 | 104,650,005.45 | 103.35 | 8,269,697.12 | 0.53% | 0.08% | **0.00%** | 0.50% | 0.13% | 0.32% | **0.00%** | **0.00%** | 0.16% |
| 10 | 40 | 5 | 40 | 60 | 5 | 105,322,713.28 | 100.74 | 9,104,621.30 | **0.00%** | 0.11% | **0.00%** | 0.23% | 0.27% | 0.24% | 0.79% | **0.00%** | 0.23% |
| 10 | 40 | 5 | 40 | 60 | 10 | 105,729,258.72 | 100.90 | 9,052,801.07 | **0.00%** | **0.00%** | **0.00%** | 0.05% | **0.00%** | 1.59% | 0.24% | 0.02% | 0.81% |
| 10 | 40 | 5 | 40 | 70 | 5 | 117,661,305.95 | 100.75 | 11,080,102.28 | 0.49% | 0.15% | 0.72% | **0.00%** | **0.00%** | **0.00%** | 0.62% | 0.02% | 0.21% |
| 10 | 40 | 5 | 40 | 70 | 10 | 117,698,666.75 | 100.61 | 11,088,766.53 | 0.97% | 0.33% | 0.55% | 0.72% | **0.00%** | 0.05% | **0.00%** | 0.28% | **0.00%** |
| 10 | 40 | 10 | 30 | 60 | 5 | 94,554,839.78 | 104.21 | 6,566,525.05 | **0.00%** | 0.09% | 0.16% | 0.06% | 0.09% | 0.74% | 0.04% | **0.00%** | **0.00%** |
| 10 | 40 | 10 | 30 | 60 | 10 | 94,712,670.17 | 104.24 | 6,595,042.62 | 0.02% | **0.00%** | 0.39% | 0.14% | **0.00%** | 0.44% | **0.00%** | 0.03% | **0.00%** |
| 10 | 40 | 10 | 30 | 70 | 5 | 104,941,038.42 | 103.18 | 8,315,693.42 | 1.02% | 0.23% | 0.18% | 0.05% | **0.00%** | **0.00%** | **0.00%** | **0.00%** | 0.73% |
| 10 | 40 | 10 | 30 | 70 | 10 | 104,981,703.83 | 103.33 | 8,312,080.92 | 0.28% | 0.03% | 0.45% | **0.00%** | 0.06% | **0.00%** | 0.39% | **0.00%** | 0.18% |
| 10 | 40 | 10 | 40 | 60 | 5 | 104,064,078.90 | 100.93 | 9,057,680.45 | 1.37% | 0.01% | 0.98% | **0.00%** | **0.00%** | **0.00%** | 1.43% | **0.00%** | 0.55% |
| 10 | 40 | 10 | 40 | 60 | 10 | 104,990,602.78 | 100.83 | 9,092,775.85 | 1.30% | 0.01% | 0.84% | **0.00%** | 0.05% | **0.00%** | 0.68% | **0.00%** | 1.06% |
| 10 | 40 | 10 | 40 | 70 | 5 | 117,748,578.03 | 100.65 | 11,059,492.68 | 1.26% | 0.20% | 0.91% | 0.70% | **0.00%** | 0.71% | **0.00%** | 0.18% | **0.00%** |
| 10 | 40 | 10 | 40 | 70 | 10 | 117,612,171.58 | 100.50 | 11,066,311.35 | 1.38% | **0.00%** | 1.36% | **0.00%** | 0.16% | **0.00%** | 0.39% | 0.34% | 0.65% |
| 10 | 50 | 5 | 30 | 60 | 5 | 94,552,063.95 | 104.09 | 6,500,546.62 | 0.09% | 0.17% | 0.73% | 0.07% | 0.36% | **0.00%** | **0.00%** | **0.00%** | 1.51% |
| 10 | 50 | 5 | 30 | 60 | 10 | 94,624,449.00 | 104.19 | 6,559,273.08 | **0.00%** | **0.00%** | **0.00%** | 0.06% | 0.07% | 0.26% | 0.13% | 0.10% | 0.23% |
| 10 | 50 | 5 | 30 | 70 | 5 | 104,841,454.97 | 103.34 | 8,332,678.75 | 1.23% | **0.00%** | 0.36% | **0.00%** | 0.05% | **0.00%** | 0.68% | 0.03% | 0.42% |
| 10 | 50 | 5 | 30 | 70 | 10 | 104,644,649.20 | 103.26 | 8,311,685.03 | 0.92% | 0.09% | 0.12% | **0.00%** | 0.12% | **0.00%** | 0.35% | **0.00%** | 0.53% |
| 10 | 50 | 5 | 40 | 60 | 5 | 105,256,370.90 | 100.74 | 9,124,679.62 | 0.70% | 0.22% | 0.60% | 0.64% | **0.00%** | 0.47% | **0.00%** | 0.28% | **0.00%** |
| 10 | 50 | 5 | 40 | 60 | 10 | 104,892,459.38 | 100.72 | 9,148,605.15 | 1.41% | 0.08% | **0.00%** | 0.59% | 0.03% | 0.34% | **0.00%** | **0.00%** | 0.25% |
| 10 | 50 | 5 | 40 | 70 | 5 | 118,134,879.38 | 100.80 | 11,101,943.82 | 0.25% | 0.14% | 0.70% | 0.21% | **0.00%** | 0.91% | **0.00%** | 0.03% | **0.00%** |
| 10 | 50 | 5 | 40 | 70 | 10 | 118,136,339.90 | 100.75 | 11,154,865.12 | 0.89% | 0.07% | 0.07% | 0.31% | **0.00%** | **0.00%** | **0.00%** | 0.01% | 0.67% |
| 10 | 50 | 10 | 30 | 60 | 5 | 94,337,018.88 | 104.08 | 6,558,153.50 | 1.37% | **0.00%** | 0.56% | 0.65% | 0.01% | 1.04% | **0.00%** | 0.44% | **0.00%** |
| 10 | 50 | 10 | 30 | 60 | 10 | 94,388,692.80 | 104.12 | 6,560,253.45 | 0.66% | **0.00%** | 0.36% | 0.28% | 0.08% | 1.16% | **0.00%** | 0.19% | **0.00%** |
| 10 | 50 | 10 | 30 | 70 | 5 | 105,347,982.90 | 103.16 | 8,353,232.15 | **0.00%** | 0.17% | **0.00%** | 0.56% | **0.00%** | 1.01% | 0.63% | 0.07% | 0.70% |
| 10 | 50 | 10 | 30 | 70 | 10 | 104,673,098.12 | 103.26 | 8,288,574.45 | 0.72% | 0.20% | 0.84% | **0.00%** | 0.18% | **0.00%** | 0.64% | **0.00%** | 1.53% |
| 10 | 50 | 10 | 40 | 60 | 5 | 105,867,424.58 | 100.85 | 9,153,754.32 | 0.34% | **0.00%** | 0.94% | 0.47% | 0.05% | 0.34% | **0.00%** | 0.22% | **0.00%** |
| 10 | 50 | 10 | 40 | 60 | 10 | 105,854,191.80 | 100.83 | 9,166,580.05 | 0.27% | 0.12% | **0.00%** | 0.61% | 0.03% | 1.45% | **0.00%** | **0.00%** | 0.03% |
| 10 | 50 | 10 | 40 | 70 | 5 | 117,234,709.35 | 100.67 | 11,094,372.15 | 1.45% | 0.21% | 0.50% | **0.00%** | **0.00%** | **0.00%** | 0.84% | 0.17% | 0.39% |
| 10 | 50 | 10 | 40 | 70 | 10 | 117,928,718.10 | 100.79 | 11,107,442.60 | 0.66% | **0.00%** | 0.96% | **0.00%** | 0.17% | **0.00%** | 0.54% | 0.01% | 0.12% |
| **Number of zero gaps** | | | | | | | | | **11** | **25** | **21** | **23** | **29** | **17** | **30** | **15** | **27** |
| **Averages** | | | | | | | | | **5.23‰** | **0.84‰** | **3.92‰** | **2.45‰** | **0.53‰** | **4.02‰** | **2.29‰** | **1.03‰** | **3.01‰** |

has never been reached. The objectives are solved sequentially. The model is first solved by ignoring $f_2$ and $f_3$. Then, $f_1^{(opt)}$ is set as a constraint and the model is again solved minimizing $f_2$ only. Finally, $f_2^{(opt)}$ is set as a constraint, and the model is solved minimizing $f_3$ only. Table 4.2 shows the results of the exact method compared with *BSSC*. The first three columns indicate the number $N$ (resp. $J$, $M$) of models (resp. shops, wholesalers). The fourth column gives the output of ILP, where $O$ stands for optimal (i.e., CPLEX finds the optimal solution), and $T$ indicates that the time-limit was reached without getting an optimum. The next three columns give the different objective values returned by CPLEX. A "$-$" means that CPLEX is not able to find an upper bound for this objective. In this case, no further experiment is performed on that instance. If for a given objective value, a (B) is indicated next to it, it means that CPLEX failed to return the optimal value and returns only an upper bound. The last three columns give the gaps of *BSSC* with respect to the results returned by CPLEX. *BSSC* is very competitive compared with CPLEX, and it can sometimes find a better upper bound (i.e., negative gaps), as in the fourth block of results. Note also that the computation time of *BSSC* is much smaller. For example, *BSSC* requires only 580 seconds for the instance with $(N, J, M) = (50, 70, 6)$. For the above reasons, we can safely conclude that *BSSC* is effective on small instances, and is likely to be trusted on the realistic instances, for which CPLEX cannot even find an interesting upper bound in a reasonable amount of time.

Table 4.2: Results (percentage gaps) of the exact methods compared with *BSSC*

| $N$ | $J$ | $M$ | *Output* | $f_1^\star$ | $f_2^\star$ | $f_3^\star$ | ***BSSC***$(f_1)$ | ***BSSC***$(f_2)$ | ***BSSC***$(f_3)$ |
|---|---|---|---|---|---|---|---|---|---|
| 12 | 15 | 2 | O | 6 | 5.39 | 25,813 | 0.00% | 0.00% | 0.00% |
| 12 | 15 | 2 | O | 6 | 6.04 | 27,985 | 0.00% | 0.00% | 0.00% |
| 12 | 15 | 2 | O | 5 | 4.91 | 21,394 | 0.00% | 0.00% | 0.00% |
| 12 | 15 | 2 | O | 6 | 5.43 | 25,454 | 0.00% | 0.00% | 0.00% |
| 12 | 15 | 2 | O | 7 | 5.05 | 26,345 | 0.00% | 0.00% | 0.00% |
| 24 | 30 | 4 | O | 69 | 7.18 | 98,426 | 0.00% | 0.00% | 0.00% |
| 24 | 30 | 4 | O | 65 | 6.98 | 91,945 | 0.00% | 0.00% | 0.00% |
| 24 | 30 | 4 | O | 81 | 6.75 | 93,443 | 0.00% | 0.00% | 0.00% |
| 24 | 30 | 4 | O | 56 | 6.87 | 101,434 | 0.00% | 0.00% | 0.00% |
| 24 | 30 | 4 | O | 61 | 6.04 | 91,343 | 0.00% | 0.00% | 0.00% |
| 30 | 50 | 4 | T | 131 | 10.13 (B) | - | 0.00% | 0.00% | - |
| 30 | 50 | 4 | T | 133 | 10.34 (B) | - | 0.00% | 0.00% | - |
| 30 | 50 | 4 | T | 145 | 11.23 (B) | - | 0.00% | 0.00% | - |
| 30 | 50 | 4 | T | 112 | 11.24 (B) | - | 0.00% | -0.05% | - |
| 30 | 50 | 4 | T | 131 | 11.34 (B) | - | 0.00% | 0.02% | - |
| 50 | 70 | 6 | T | 440 (B) | - | - | 0.00% | - | - |
| 50 | 70 | 6 | T | 453 (B) | - | - | 0.00% | - | - |
| 50 | 70 | 6 | T | 398 (B) | - | - | -0.03% | - | - |
| 50 | 70 | 6 | T | 343 (B) | - | - | -0.01% | - | - |
| 50 | 70 | 6 | T | 459 (B) | - | - | 0.00% | - | - |
| 60 | 80 | 7 | T | - | - | - | - | - | - |
| 60 | 80 | 7 | T | - | - | - | - | - | - |
| 60 | 80 | 7 | T | - | - | - | - | - | - |
| 60 | 80 | 7 | T | - | - | - | - | - | - |
| 60 | 80 | 7 | T | - | - | - | - | - | - |

### 4.5.4 Sensitivity analysis

In this subsection, a deeper sensitivity analysis of the perturbation parameters is conducted. Figures 4.3 to 4.8 show a wider variation of each perturbation parameter, (for $\delta_1, \delta_2, \delta_3, \delta_4, \delta_5$ and $\varepsilon$, respectively). For each figure, the other involved parameters are set to the *reference* values.

Figure 4.3 shows that $\delta_1$ has an impact on both $f_1$ and $f_2$ (as $f_1$ decreases by almost 1% if $\delta_1$ decreases by 4%). Finding more reliable suppliers could lead to such a $\delta_1$ reduction. Figures 4.4 and 4.5 show that the $f_i$'s are steady even if $\delta_2$, $\delta_3$ and $\varepsilon$ are significantly reduced. Therefore, these three parameters do not play a crucial role in the supply chain. On the opposite, Figures 4.6 and 4.7 show that both $\delta_4$ and $\delta_5$ are the most sensitive parameters, as they significantly reduce $f_1$ and $f_3$ (even if in Figure 4.6, it leads to a small augmentation of $f_2$).

This is due to the wholesalers, which are likely to sacrifice the considered shop and favor higher priority shops. More precisely, as the shops with the lowest priority are the most present, the watches will rather reach shops of priority 2 or 3, and therefore decrease $f_1$ and $f_3$ (as shortages are less penalized if more watches arrive, and shops of higher priorities are less penalized on inventories). Therefore, $f_2$ increases for the shops of lower priorities, which are more present. These conclusions, regarding the wholesalers, were acknowledged by *SWB*.

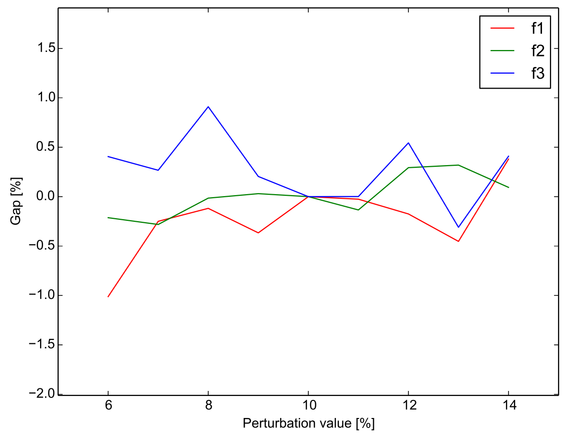

Figure 4.3: Sensitivity analysis on $\delta_1$

### 4.5.5 Managerial insights

The three solution methods used to tackle the problem faced by *SWB* are designed such that the best simulated solution is returned. But sometimes, due to other information, the decision-makers could decide to select another solution, which is not necessarily the best one, but could be another one based on intuition and experience. In this context, Table 4.3 proposes, using *BSSC*, to simulate and return the values of the $k$ best solutions, such that a decision-maker can select one among $k = 5$ different solutions.

Moreover we propose a new measure *rob*, which is a robustness indicator of a simulation, defined as the standard deviation of $f$ over the *sim* runs. In realistic conditions, the lower $rob(\hat{S})$ is, the higher is the probability that, if $\hat{S}$

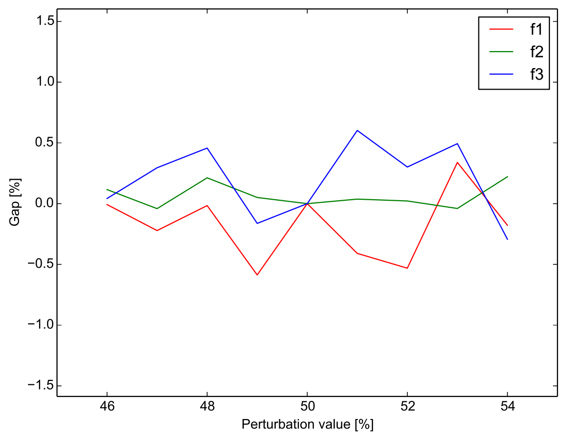

Figure 4.4: Sensitivity analysis on $\delta_2$

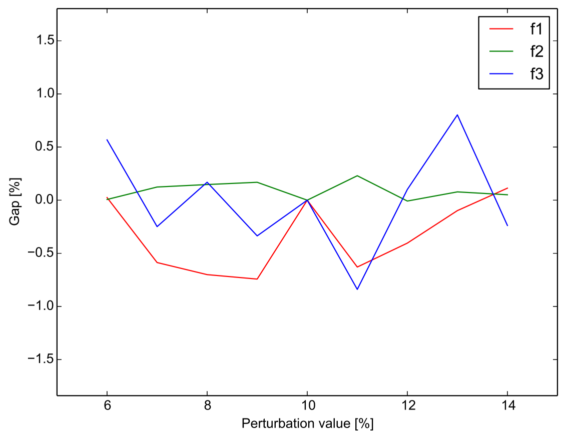

Figure 4.5: Sensitivity analysis on $\delta_3$

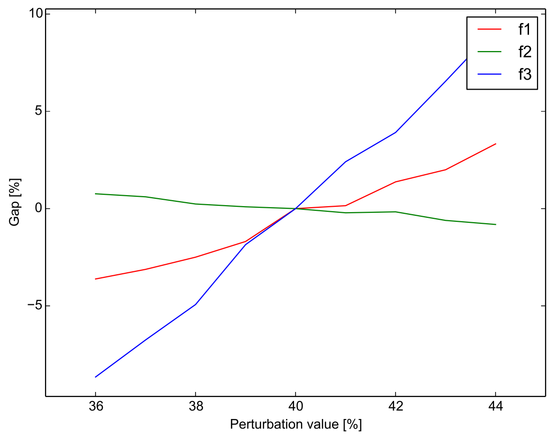

Figure 4.6: Sensitivity analysis on $\delta_4$

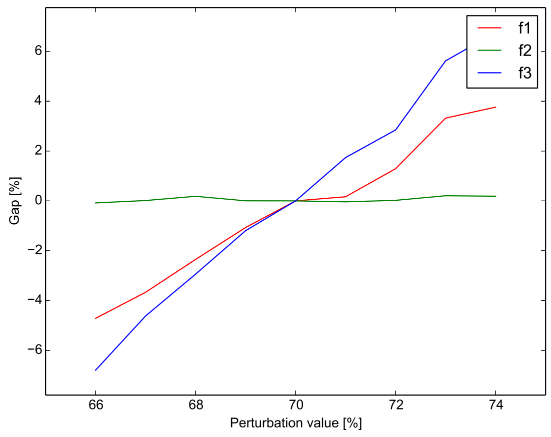

Figure 4.7: Sensitivity analysis on $\delta_5$

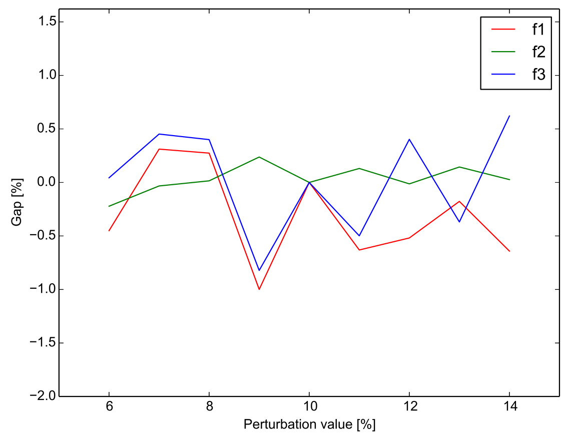

Figure 4.8: Sensitivity analysis on $\varepsilon$

is used for multiple consecutive years, the corresponding actual solutions (with perturbations) are similar to each others. Moreover, it results in a good control on the costs.

In Table 4.3, solution $S_3$ has the best $f_1$ value (as the indicated gap is 0), but it has a rather high *rob* value. The manager could decide to sacrifice 0.25% on $f_1$ and select solution $S_4$, which has the lowest $f_3$ value and a much better *rob* value. Alternatively, he/she could select solution $S_2$ by sacrificing 0.05% on $f_1$ and get the best proposed values of $f_2$ and *rob*. This example is used to show that a manager could decide to sacrifice a small gap on one top objective and favor a lower level objective on the robustness indicator. Depending on the situation, the manager could also decide to select the solution with the lowest *rob* value, even if it is outperformed on the other objectives by the other provided solutions. In any case, the provided five solutions are competitive, as they are the most promising among the $pop = 100$ solutions.

Table 4.3: Management insights using *BSSC*

| **Objective** | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
|---|---|---|---|---|---|
| $f_1$ | 1.31% | 0.05% | 0.00% | 0.25% | 0.87% |
| $f_2$ | 0.04% | 0.00% | 0.12% | 0.01% | 0.18% |
| $f_3$ | 1.58% | 0.63% | 0.75% | 0.00% | 1.25% |
| $rob$ | 11.08% | 0.00% | 16.22% | 9.34% | 18.33% |

## 4.6 Conclusion and future works

In this paper, we propose a relevant and complex problem (P) faced by a well-known Swiss watch brand. We propose powerful solution methods to tackle (P), and showed the superiority of the proposed matheuristic compared with standard heuristics. Moreover, we highlight the most sensitive parameters on which luxury companies should focus on. Finally we propose managerial insights, which allow decision makers to select a solution not only based on its quality. Future works include modeling a more complete problem, which integrates more types of perturbations. Moreover, *SWB* could evaluate and optimize an integrated supply chain where suppliers, production and dispatching would be centralized in a common system. This would allow *SWB* to reduce the perturbation parameters by gaining control over the complete supply chain.

## Acknowledgements

The authors would like to thank the team of *SWB* for their availability and their relevant advices. As we are under a non-disclosure agreement, their names cannot be mentioned, but they will probably recognize themselves in these lines.

# General conclusion

In opposition to regular PhD thesis, where a single subject is deeply discussed, this thesis proposes four different topics. Each topic is relevant to a problem faced by various industries. Even if the Chapters 1 and 3 were only applied to academic instances, they show the relevance of modern academic techniques to practical problems, such as online vehicle routing with constant position monitoring, which becomes very familiar with the appearance of cheap GPS devices. Techniques developed in both chapters can be easily transposed to practical applications, and each chapter proposes such possible applications.

In Chapters 2 and 4, two complex problems were proposed by two different companies, and are therefore completely related to practical aspects. Even if an accurate cost saving is hard to measure (as we do not have the required private data to compute it), outputs of both projects show that this thesis has been interesting to both industries.

In Chapter 2, the bin-packing problem faced by *Renault* is so important to their industry that they contacted us to benchmark their currently used algorithms, in order to ensure their quality. Results show that the latter could be improved with more advanced algorithms, such as genetic algorithms. A correct loading is very important, as the cost of transportation of a single item can increase significantly if the loading is inefficient. This could lead to important cost savings in the transportation of car parts to factories. The algorithms designed by *Renault* were improved thanks to our research, and are now widely used and trusted by the company. Moreover it led to possible future works in the case a number of instances has to be tackled jointly within a time limit.

In Chapter 4, *SWB* contacted us for a dispatching problem, without any prior method to tackle it. At the end of the project, at the time of exposing our results, *SWB* was very satisfied to realize that the simulator is very closely reflecting the situations they are facing daily. They intend to use the simulator and the solver to improve their supply chain management. The simulator and the solver can now help *SWB* to efficiently improve the dispatching of the items around the globe, in addition to testing new dispatching management methods. With little extra work, additional objectives could be added (such as cost components).

In the supply chain optimization field, only dedicated methods to tackle a single problem are relevant. The problems and the methods exposed in this thesis, which are either new or recent, are therefore very relevant to modern industries. This thesis was focusing on practical applications which were tackled with state-of-the-art academic methods. Extensive literature reviews are proposed and modern methods are described to tackle each issue. This shows the relevance of such a combination for current industrial problems, and the relevance of advanced methods dedicated to specific applications.

# Bibliography

[1] A. Allahverdi, C.T. Ng, T.C.E. Cheng, and M. Kovalyov. A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3):985 – 1032, 2008.

[2] A. Anglani, A. Grieco, E. Guerriero, and R. Musmanno. Robust scheduling of parallel machines with sequence-dependent set-up costs. *European Journal of Operational Research*, 161(3):704 – 720, 2005.

[3] S. Axsäter. Optimization of order-up-to-s policies in two-echelon inventory systems with periodic review. *Naval Research Logistics*, 40(2):245–253, 1993.

[4] S. Axsäter and L. Juntti. Comparison of echelon stock and installation stock policies for two-level inventory systems. *International Journal of Production Economics*, 45(1):303–310, 1996.

[5] J. Banks. *Handbook of simulation*. Wiley Online Library, 1998.

[6] P. Baptiste and C. Le Pape. Scheduling a single machine to minimize a regular objective function under setup constraints. *Discrete Optimization*, 2(1):83 – 99, 2005.

[7] J. F. Bard and B. Golany. Determining the number of kanbans in a multiproduct, multistage production system. *International Journal of Production Research*, 29(5):881–895, 1991.

[8] J. Bautista, J. Pereira, and B. Adenso-Díaz. A grasp approach for the extended car sequencing problem. *Journal of Scheduling*, 11(1):3–16, 2008.

[9] J. E. Beasley. Or-library. http://people.brunel.ac.uk/m̃astjjb/jeb/info.html.

[10] C. Becker and A.Scholl. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168(3):694 – 715, 2006.

[11] Bengston. *OR-Library*, http://www.ibr.cs.tu-bs.de/alg/packlib/xml/b-prpha-82-xml.shtml, 1982.

[12] R. Bent and P. Van Hentenryck. Waiting and Relocation Strategies in Online Stochastic Vehicle Routing. In *International Joint Conference on Artificial Intelligence*, pages 1816–1821, 2007.

[13] K. J. Binkley and M. Hagiwara. Applying self-adaptive evolutionary algorithms to two-dimensional packing problems using a four corners heuristic. *European Journal of Operational Research*, 183(3):1230–1248, 2007.

[14] C. Blum and A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35 (3):268–308, 2003.

[15] A. Bortfeldt. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. *European Journal of Operational Research*, 172(3):814–837, 2006.

[16] A. Bortfeldt, H. Gehring, and D. Mack. A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing*, 29(5):641–662, 2003.

[17] M. A. Boschetti, V. Maniezzo, M. Roffilli, and A. B. Röhler. Matheuristics: Optimization, simulation and control. In *Hybrid Metaheuristics*, pages 171–177. Springer, 2009.

[18] A. Brandstädt, J. P. Spinrad, et al. *Graph classes: a survey*, volume 3. Siam, 1999.

[19] A. Brun, F. Caniato, M. Caridi, C. Castelli, G. Miragliotta, S. Ronchi, A. Sianesi, and G. Spina. Logistics and supply chain management in luxury fashion retail: Empirical investigation of Italian firms. *International Journal of Production Economics*, 114(2):554–570, 2008.

[20] F. Caniato, M. Caridi, C. Castelli, and R. Golini. Supply chain management in the luxury industry: a first classification of companies and their strategies. *International Journal of Production Economics*, 133(2):622–633, 2011.

[21] B. Catry. The great pretenders: the magic of luxury goods. *Business Strategy Review*, 14(3):10–17, 2003.

[22] S. Chopra and P. Meindl. *Supply chain management. Strategy, planning & operation.* Springer, 2007.

[23] M. Christopher. *Logistics and supply chain management.* Pearson UK, 2012.

[24] A. J. Clark and H. Scarf. Optimal policies for a multi-echelon inventory problem. *Management Science*, 6(4):475–490, 1960.

[25] J.-F. Cordeau, M. Gendreau, G. Laporte, J.-Y. Potvin, and F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53 (5):512–522, 2002.

[26] J-F. Cordeau, G. Laporte, and F. Pasin. Iterated tabu search for the car sequencing problem. *European Journal of Operational Research*, 191(3):945 – 956, 2008.

[27] J.-f Côté, M. Dell'Amico, and M. Iori. Combinatorial benders cuts for the strip packing problem. *Technical report, DISMI, University of Modena and Reggio Emilia*, 2013.

[28] T. G. Crainic, G. Perboli, and R. Tadei. TS2PACK: A two-level tabu search for the three-dimensional bin packing problem. *European Journal of Operational Research*, 195(3):744–760, 2009.

[29] J. Daniel and C. Rajendran. A simulation-based genetic algorithm for inventory optimization in a serial supply chain. *International Transactions in Operational Research*, 12(1):101–127, 2005.

[30] M. Dorigo. *Optimization, learning and natural algorithms (in Italian).* PhD thesis, Politecnico di Milano, Dipartimento di Elettronica, Italy, 1992.

[31] M. Ehrgott. *Multicriteria optimization.* Springer Science & Business Media, 2006.

[32] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spectrum*, 22:425–460, 2000.

[33] B. Estellon, F. Gardi, and K. Nouioua. Two local search approaches for solving real-life car sequencing problems. *European Journal of Operational Research*, 191(3):928 – 944, 2008.

[34] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[35] B. Fleischmann, S. Gnutzmann, and E. Sandvoß. Dynamic Vehicle Routing based on Online Traffic Information. *Transportation Science*, 38:420–433, 2004.

[36] R. Forsberg. Optimization of order-up-to-S policies for two-level inventory systems with compound Poisson demand. *European Journal of Operational Research*, 81(1):143–153, 1995.

[37] B. Gacias, C. Artigues, and P. Lopez. Parallel machine scheduling with precedence constraints and setup times. *Computers & Operations Research*, 37(12):2141 – 2151, 2010.

[38] P. Galinier, A. Hertz, and N. Zufferey. An Adaptive Memory Algorithm for the Graph Coloring Problem. *Discrete Applied Mathematics*, 156:267–279, 2008.

[39] H. Gehring and A. Bortfeldt. A parallel genetic algorithm for solving the container loading problem. *International Transactions in Operational Research*, 9(4):497–511, 2002.

[40] M. Gendreau, F. Guertin, J.-Y. Potvin, and R. Séguin. Neighborhood Search Heuristics for a Dynamic Vehicle Dispatching Problem with Pick-ups and Deliveries. *Transportation Research Part C: Emerging Technologies*, 14:157–174, 2006.

[41] M. Gendreau, F. Guertin, J.-Y. Potvin, and É. Taillard. Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching. *Transportation Science*, 33:381–390, 1999.

[42] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350, 2006.

[43] M. Gendreau, G. Laporte, and J.-Y. Potvin. *The Vehicle Routing Problem*, chapter Metaheuristics for the VRP, pages 129–154. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 2002.

[44] M. Gendreau and J.-Y. Potvin. Dynamic Vehicle Routing and Dispatching. In T.G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 115–126. Kluwer, 1998.

[45] M. Gendreau and J.-Y. Potvin. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science. Springer, 2010.

[46] M. Gendreau and J.-Y. Potvin. Tabu search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 41–59. Springer US, 2010.

[47] M. Gendreau, J.-Y. Potvin, O. Bräumlaysy, G. Hasle, and A. Lokketangen. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces Series*, pages 143–169. Springer, 2008.

[48] G. Ghiani, E. Manni, and B. W. Thomas. A Comparison of Anticipatory Algorithms for the Dynamic and Stochastic Traveling Salesman Problem. *Transportation Science*, 46:374–387, 2012.

[49] A. Glasmeier. Technological discontinuities and flexible production networks: The case of Switzerland and the world watch industry. *Research Policy*, 20(5):469–485, 1991.

[50] F. Glover. Future paths for integer programming and linkage to artificial intelligence. *Computers & Operations Research*, 13:533–549, 1986.

[51] F. Glover. Tabu search - part I. *ORSA Journal on Computing*, 1:190–205, 1989.

[52] F. Glover, D. Klingman, and N. Phillips. Netform Modeling and Applications. *Interfaces*, 20(4):7–27, 1990.

[53] F. Glover, C. McMillan, and D. Klingman. The netform concept: A more effective model form and solution procedure for large scale nonlinear problems. In *Proceedings of the 1977 annual conference*, pages 283–289. ACM, 1977.

[54] D. E. Goldberg. Genetic algorithms in search, optimization and machine learning. *Addison-Wesley, Reading, MA*, 1989.

[55] B.L. Golden, S. Raghavan, and E.A. Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer, 2008.

[56] P. Hansen, J. Kuplinsky, and D. de Werra. Mixed Graph Coloring. *Mathematical Methods of Operations Research*, 45:145–169, 1997.

[57] A. Hertz, D. Schindl, and N. Zufferey. A solution method for a car fleet management problem with maintenance constraints. *Journal of Heuristics*, 15 (5):425–450, 2009.

[58] F. Hnaien, X. Delorme, and A. Dolgui. Multi-objective optimization for inventory control in two-level assembly systems under uncertainty of lead times. *Computers & Operations Research*, 37(11):1835–1843, 2010.

[59] E. Hopper and B.C.H. Turton. A genetic algorithm for a 2D industrial packing problem. *Computers & Industrial Engineering*, 37:375–378, 1999.

[60] E. Hopper and B.C.H. Turton. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. *European Journal of Operational Research*, 128(1):34–57, 2001.

[61] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Diversion Issues in Real-Time Vehicle Dispatching. *Transportation Science*, 34:426–438, 2000.

[62] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle Dispatching with Time-Dependent Travel Times. *European Journal of Operational Research*, 144:379–396, 2003.

[63] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Exploiting Knowledge about Future Demands for Real-Time Vehicle Dispatching. *Transportation Science*, 40:211–225, 2006.

[64] D.F. Jones, S.K. Mirrazavi, and M. Tamiz. Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, 137(1):1 – 9, 2002.

[65] S. Kara, F. Rugrungruang, and H. Kaebernick. Simulation modelling of reverse logistics networks. *International Journal of Production Economics*, 106(1):61–69, 2007.

[66] M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, and H. Nagamochi. Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research*, 198(1):73–83, 2009.

[67] A. Khanafer, F. Clautiaux, and E.-G. Talbi. Tree-decomposition based heuristics for the two-dimensional bin packing problem with conflicts. *Computers & Industrial Engineering*, 39(1):54–63, 2012.

[68] G. Laporte. Fifty Years of Vehicle Routing. *Transportation Science*, 43(4):408–416, November 2009.

[69] A. M. Law and W. D. Kelton. *Simulation modeling and analysis*, volume 2. McGraw-Hill New York, 1991.

[70] H. L. Lee and K. Moinzadeh. Two-parameter approximations for multi-echelon repairable inventory models with batch ordering policy. *IIE Transactions*, 19(2):140–149, 1987.

[71] N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher. Exhaustive approaches to 2D rectangular perfect packings. *Information Processing Letters*, 90(1):7–14, 2004.

[72] R. Lewis, X. Song, K. Dowsland, and J. Thompson. An investigation into two bin packing problems with ordering and orientation implications. *European Journal of Operational Research*, 213(1):52–65, 2011.

[73] D.S. Liu, K.C. Tan, S.Y. Huang, C.K. Goh, and W.K. Ho. On solving multiobjective bin packing problems using evolutionary particle swarm optimization. *European Journal of Operational Research*, 190(2):357–382, 2008.

[74] A. Lodi, S. Martello, and M. Monaci. Two-dimensional packing problems: A survey. *European Journal of Operational Research*, 141(2):241–252, 2002.

[75] A. Lodi, S. Martello, and D. Vigo. Approximation algorithms for the oriented two-dimensional bin packing problem. *European Journal of Operational Research*, 112(1):158–166, 1999.

[76] A. Lodi, S. Martello, and D. Vigo. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing*, 11(4):345–357, April 1999.

[77] S. Lorini, J.-Y. Potvin, and N. Zufferey. Online Vehicle Routing and Scheduling with Dynamic Travel Times. *Computers & Operations Research*, 38:1086–1090, 2011.

[78] T. Loukil, J. Teghem, and D. Tuyttens. Solving multi-objective production scheduling problems using metaheuristics. *European Journal of Operational Research*, 161(1):42–61, 2005.

[79] K. Lund, O.B.G. Madsen, and J.M. Rygaard. *Vehicle Routing Problems with Varying Degrees of Dynamism*. IMM-REP. Technical report, 1996.

[80] D. Mack, A. Bortfeldt, and H. Gehring. A parallel hybrid local search algorithm for the container loading problem. *International Transactions in Operational Research*, 11(5):511–533, 2004.

[81] A. Madadi, M. E. Kurz, and J. Ashayeri. Multi-level inventory management decisions with transportation cost consideration. *Transportation Research Part E: Logistics and Transportation Review*, 46(5):719–734, 2010.

[82] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[83] S. Martello, M. Monaci, and D. Vigo. An exact approach to the strip-packing problem. *INFORMS Journal on Computing*, 15:310–319, 2003.

[84] S. Martello and D. Vigo. Exact solution of the two-dimensional finite bin packing problem. *Management Science*, 44(3):388–399, 1998.

[85] M. T. Melo, S. Nickel, and F. Saldanha-da Gama. Facility location and supply chain management–a review. *European Journal of Operational Research*, 196(2):401–412, 2009.

[86] S. Mitrović-Minić, R. Krishnamurti, and G. Laporte. Double-Horizon Based Heuristics for the Dynamic Pickup and Delivery Problem with Time Windows. *Transportation Research Part B: Methodological*, 38:669–685, 2004.

[87] K. Moinzadeh and H. L Lee. Batch size and stocking levels in multi-echelon repairable systems. *Management Science*, 32(12):1567–1581, 1986.

[88] G. Moslehi and M. Mahnam. A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1):14–22, 2011.

[89] A. Moura and J.F. Oliveira. A GRASP approach to the container-loading problem. *Intelligent Systems*, 20(4):50–57, 2005.

[90] A. Nguyen. Private communication. *Renault R&D - Paris*, 2013.

[91] A. Nguyen and J-Ph. Brenaut. A truck loading algorithm for Renault's supply chain system. In *23rd EURO Conference, Bonn, Germany*, July 5-8, 2009.

[92] N. Ntene and J.H. van Vuuren. A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem. *Discrete Optimization*, 6(2):174–188, 2009.

[93] Federation of the Swiss Watch Industry FH. Swiss watch export: World. *2014 Report*, 2014.

[94] I. H. Osman and G. Laporte. Metaheuristics: A Bibliography. *Annals of Operations Research*, 63:513–623, 1996.

[95] P. M. Pardalos and J. Xue. The maximum clique problem. *Journal of Global Optimization*, 4(3):301–328, 1994.

[96] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia. A Review of Dynamic Vehicle Routing Problems. *European Journal of Operational Research*, 225:1 – 11, 2013.

[97] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems, third edition.* Prentice Hall, 2008.

[98] D. Pisinger. Heuristics for the container loading problem. *European Journal of Operational Research*, 141(2):382–392, 2002.

[99] D. Pisinger and M. Sigurd. The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, 2(2):154–167, 2005.

[100] S. Polyakovsky and R. M'Hallah. An agent-based approach to the two-dimensional guillotine bin packing problem. *European Journal of Operational Research*, 192(3):767–781, 2009.

[101] J.-Y. Potvin, Y. Xu, and I. Benyahia. Vehicle Routing and Scheduling with Dynamic Travel Times. *Computers & Operations Research*, 33:1129 – 1137, 2006.

[102] H.N. Psaraftis. Dynamic Vehicle Routing: Status and Prospects. *Annals of Operations Research*, 61:143–164, 1995.

[103] J. Puchinger and G. R. Raidl. Models and algorithms for three-stage two-dimensional bin packing. *European Journal of Operational Research*, 183(3):1304–1327, 2007.

[104] M.G.C. Resende and C.C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 283–319. Springer US, 2010.

[105] J. Respen and N. Zufferey. A Renault truck loading problem: form benchmarking to improvements. *Proceedings of the 14th EU/ME Workshop, Hamburg, Germany*, pages 79–84, 2013.

[106] J. Respen, N. Zufferey, and E. Amaldi. Heuristics for a multi-machine multi-objective job scheduling problem with smoothing costs. In *1st IEEE International Conference on Logistics Operations Management 2012*, 2012.

[107] J. Respen, N. Zufferey, and J.-Y. Potvin. Online Vehicle Routing and Scheduling with Continuous Vehicle Tracking. In *Proceedings of the ROADEF 2014, Bordeaux, France. February 26 to 28.*, 2014.

[108] C. C. Ribeiro, D. Aloise, T. F. Noronha, C. Rocha, and S. Urrutia. A hybrid heuristic for a multi-objective real-life car sequencing problem with painting and assembly line constraints. *European Journal of Operational Research*, 191(3):981–992, 2008.

[109] M.C. Riff, X. Bonnaire, and B. Neveu. A revision of recent approaches for two-dimensional strip-packing problems. *Engineering Applications of Artificial Intelligence*, 22:823–827, 2009.

[110] Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.

[111] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[112] D. Schindl and N. Zufferey. Solution methods for fuel supply of trains. *Information Systems and Operational Research*, 51 (1):22 – 29, 2013.

[113] A. Segerstedt. *Cover-time Planning: An Alternative to MRP*. PhD thesis, Linköping University, 1991.

[114] A. Segerstedt. A capacity-constrained multi-level inventory and production control problem. *International Journal of Production Economics*, 45(1):449–461, 1996.

[115] H. D. Sherali and P. J. Driscoll. On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems. *Operations Research*, 50(4):656–669, 2002.

[116] E. A. Silver and N. Zufferey. Inventory control of raw materials under stochastic and seasonal lead times. *International Journal of Production Research*, 43:5161–5179, 2005.

[117] E. A. Silver and N. Zufferey. Inventory control of an item with a probabilistic replenishment lead time and a known supplier shutdown period. *International Journal of Production Research*, 49(4):923–947, 2011.

[118] D. Simchi-Levi. *Designing and managing the supply chain.* Mcgraw-Hill College, 2005.

[119] A. Soke and Z. Bingul. Hybrid genetic algorithm and simulated annealing for two-dimensional non-guillotine rectangular packing problems. *Engineering Applications of Artificial Intelligence*, 19(5):557–567, 2006.

[120] C. Solnon, V.D. Cung, A. Nguyen, and C. Artigues. The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEFï¿½ 2005 challenge problem. *European Journal of Operational Research*, 191(3):912–927, 2008.

[121] M.M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research*, 35:254–265, 1987.

[122] A. Svoronos and P. Zipkin. Estimating the performance of multi-level inventory systems. *Operations Research*, 36(1):57–72, 1988.

[123] SWB. Private communication. *Switzerland*, 2014.

[124] É. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science*, 31:170–186, 1997.

[125] S. Terzi and S. Cavalieri. Simulation in the supply chain context: a survey. *Computers in Industry*, 53(1):3–16, 2004.

[126] S. Webster, P. D. Jog, and A. Gupta. A genetic algorithm for scheduling job families on a single machine with arbitrary earliness/tardiness penalties and an unrestricted common due date. *International Journal of Production Research*, 36 (9):2543–2551, 1998.

[127] D.-Q. Yao, X. Yue, S. K Mukhopadhyay, and Z. Wang. Strategic inventory deployment for retail and e-tail stores. *Omega*, 37(3):646–658, 2009.

[128] D. Zhang, Y. Peng, and S. C. H. Leung. A heuristic block-loading algorithm based on multi-layer search for the container loading problem. *Computers & Operations Research*, 39(10):2267–2276, 2012.

[129] N. Zufferey. Metaheuristics: Some principles for an efficient design. *Computer Technology and Application*, 3:446–462, 2012.

[130] N. Zufferey, M. Studer, and E. A. Silver. Tabu search for a car sequencing problem. *Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference, FLAIRS 2006, Melbourne, USA, May 11-13*, pages 457–462, 2006.