Politecnico di Torino

Master's Degree in Aerospace Engineering

Master's thesis

Solving methods for the Multi-item Inventory Routing Problem with Pickup and Delivery

Relators:

Prof. Renzo Arina
Prof.ssa Stefania Scarsoglio
Prof. Patrice Leclaire
Prof. Arthur Bit-Monnot

Candidate:
Filippo Montanari

December 2021

Acknowledgment

First of all, I would like to thank my internship supervisors Prof. Arthur Bit-Monnot and Mr. Mathieu Touchard for the trust they placed in me and their valuable tutoring role. Their guidance was precious and decisive to enhance my working skills and develop the content of this thesis.

I sincerely thank Prof. Patrice Leclaire for helping me take this opportunity and being truly supportive with me on many occasions during the last year. I thank him for his constant professionality, always accompanied by a particular awareness of students' needs and evolvement.

I would like to thank all the great people I had the chance of having at my side along the way, from Turin, to Paris, to Toulouse, in both pleasant and difficult times. A special thank to those whom, despite the distance, have always been there for me. I consider this thesis the conclusion of a 6-years-long journey, and I feel lucky for all the spiritual and concrete support I received in so many occasions. "The road to knowledge is a road that passes through good encounters." (*Spinoza*)

Last, I heartfully thank my family, which is the start and the harbour of every journey I made and will be able to make in my life.

Grazie a tutti voi. *Merci a vous tous.*

Filippo Montanari
17/11/2021

Blank page

Abstract

As industry 4.0, IoT and clouding technologies take hold, real-time scheduling and resource allocation solutions by means of proven optimization techniques have a concrete interest and remarkable market opportunities. Concerning the logistic and production domains, some solid well-designed optimization systems can be a precious ally to *Just-In-Time* philosophy at work. Speaking of logistics, it is a fast-growing sector responsible for a significant share of company costs, and its optimization has a major impact on economic rentability and competitiveness.
The topic of this thesis is the research of dynamic and flexible scheduling methods for a generic indoor logistic support system. More specifically, the sought methods concern the optimal satisfaction of the expressed material needs of an indoor manufacturing facility, by means of a capacitated vehicle fleet. Part I is devoted to bibliographic research, preceded by an introduction to some optimization key-concepts. Part II presents and discusses in detail the problem, its mathematical formulations, and the methods adopted to solve it. A series of MILP formulations are proposed, each with a specific set of hypotheses, the associated mathematical model, and some considerations about applicability and solvability. Then, an ad-hoc heuristic was designed to compare and evaluate the performance of the Discrete-Time 1-mainstock MILP (DT-1ms-MILP) formulation.
The thesis ends with a joint evaluation of two tested methods, with a special appreciation for DT-1ms-MILP by reason of its performances and flexibility. Last, some directions are given for a further development of this thesis work.

Keywords: Inventory Rounting Problem, multi-item IRP, pickup and delivery, dynamic scheduling

Con l'avanzare dell'industria 4.0, dell'IoT, e delle tecnologie cloud, la pianificazione e l'allocazione di risorse in tempo reale attraverso tecniche di ottimizzazione collaudate destano un concreto interesse nel mercato tecnologico. Dei metodi di ottimizzazione robusti ed efficaci costituiscono un valido alleato per la filosofia *Just-In-Time* nei campi della logistica e dei sistemi di produzione. La logistica in particolare è un settore in forte crescita che determina una parte considerevole delle spese di un'azienda, e l'ottimizzazione delle attività che ne fanno parte può portare un considerevole guadagno in termini di redditività e competitività. L'argomento di questa tesi è la ricerca di metodi dinamici e flessibili per la pianificazione di una flotta di veicoli da interno a supporto di una linea di produzione. I suddetti metodi devono permettere di soddisfare in maniera ottimale i bisogni materiali della produzione a mezzo di una flotta a capacità finita. La parte I è dedicata alla ricerca bibliografica e all'introduzione di alcuni concetti chiave dell'ottimizzazione. La parte II presenta e discute in dettaglio il problema e i metodi impiegati per risolverlo. Anzitutto, una serie di formulazioni MILP, ognuna con le proprie ipotesi, il modello matematico associato, e alcune considerazioni riguardo alla applicabilità e alla risolvibilità. Segue la presentazione di un'euristica ad-hoc, sviluppata al fine di comparare e co-valutare la formulazione MILP Discrete-Time 1-mainstock (DT-1ms-MILP).
La tesi si conclude con una valutazione congiunta dei metodi testati, con un particolare apprezzamento per la DT-1ms-MILP in ragione delle performance e della flessibilità dimostrate. Infine, alcuni suggerimenti sono altresì dati per un eventuale continuazione del lavoro presentato.

Content

List of figures

List of tables

Blank page

Introduction

Optimization techniques are a major engineering tool with an impact on several fields, such as transportation, production scheduling, network design and multi-disciplinary systems. Every company that wishes to keep up with times must constantly find new ways to improve productivity and service quality while mitigating costs. Optimization plays a central role in it, firmly backed by calculators and their exponentially increasing processing power. No matter the context, wisely applied optimization techniques grant more efficient processes and services. As industry 4.0, IoT and clouding technologies take hold, real-time data analysis and processing by means of proven optimization techniques has a concrete interest and remarkable market opportunities.
Scheduling and resource allocation problems are among the most studied discrete and combinatorial optimisation issues. A common trait of many practical decision problems it that they are computationally challenging. Solving time is, along with solution quality, one of the most important aspects of an optimization technique, and the criticality of it over the quality of results depends on each specific application cases. Especially in dynamic scheduling, a method yielding a quick acceptable solution is frequently more effective than another slowly converging to global optimality. Concerning logistic and production sectors, some solid well-designed optimization system can be a precious ally to *Just-In-Time* philosophy at work. J-I-T is a cornerstone of lean management, defined by [**1**] as a set of practices to "precisely specify value by specific product, identify the value stream for each product, make value flow without interruptions, let customer pull value from the producer, and pursue perfection." Speaking of logistics, it is a relatively new sector (applied to economic and industrial environments since 1950s'), but its importance has exponentially increased in the last years due to globalization. It does not directly produce value-added; however, it is generally responsible for a significant share of company costs, and optimizing it has a major impact on economic rentability and competitiveness.

This thesis work is based on a 6-months internship at the LAAS-CNRS of Toulouse started on April 2021. The main goal of the internship was the research of dynamic scheduling methods for an indoor logistic support system. More specifically,

the sought method concerns the optimal satisfaction of specific material needs of an indoor manufacturing facility. The fleet in charge of performing logistic operations is composed by a limited number of vehicles, and scheduling methods shall be sufficiently quick to adapt to production changes almost in real-time.
Three are the peculiar and challenging aspects of the problem. First, vehicles can load different kinds of articles, each with a specific lot size. Hence, the method shall record a different value for each loadable article expressed in a unified measure unit. Second, time approximations shall be limited to allow a fine fleet control in accordance with J-I-T principles. Last, the method shall be reactive to production changes and dynamically adapt current fleet plan.
Considering the importance of inventory management aspects, as well as the analogies with routing problems, the core topic of this thesis was identified as *Multi-item Inventory Routing Problem with Pickup and Delivery* (Mi-IRP-PD). Although the primary objective is generally not inventory-and-routing cost minimization, these two terms can still be considered for method piloting or evaluation.

Part I is devoted to bibliographic research and the introduction of some important theoretical concepts met during the internship. Chapter 1 gives a brief insight on some of the most used combinatorial optimization techniques, with a main distinction between exact and approximate methods. Chapter 2 contains a general overview of routing problems' state-of-the-art. The chapter begins with the most basic problems such as the travelling salesman one (TSP), and progressively introduces new elements. The vendor managed inventory (VMI) concept allows to finally reach the definition of inventory routing problem (IRP). Chapter 3 concerns the Mi-IRP-PD itself and makes an overview of similar problems in literature.
Part II introduces and discusses in detail the internship core problem, its solving formulations, and the methods adopted to solve it. First, Chapter 4 presents the *indoor logistic support manager* (ILOM), as well as its generalization and a similar application case in the transportation sector. After that, chapter 5 explains the designed MILP formulations, each with a set hypothesis, the associated mathematical model, and some considerations about applicability and solvability. Linear programming is a powerful optimization technique, but also very time-consuming. The chance of getting a solution in a reasonable amount of time is a delicate issue, thus problems must be wisely formulated, and the choice of a good commercial solver can make the difference. The chapter proposes formulations with different characteristics, with both continuous (CT) and discrete (DT) time domains, and diverse degrees of freedom concerning the key aspects of the problem. A special focus concerns the formulation called *Discrete-Time 1-mainstock Mixed-Integer Linear Program* (DT-1ms-MILP), as it matches with the ILOM problem and was hence developed and tested during the internship. Chapter 6 introduces an ad-hoc heuristic, specifically designed to compare and evaluate the performance of DT-1ms-MILP formulation. This method, simply called *Minimum Penalty Algorithm* (MPA), is based on a direct tree-

search, and has the same input and output data format of DT-1ms-MILP for an easy co-evaluation. Both methods are tested on a set of fictive instances with different sizes. Chapter 7 presents the characteristics of each test instance, then test results followed by some critical comments.

Finally, the thesis ends with some conclusions about the developed methods, with a special focus on DT-1ms-MILP results. Moreover, some future research directions are suggested to the reader.

Blank page

PART I

BIBLIOGRAPHIC RESEARCH AND STATE-OF-THE-ART

Blank page

CHAPTER 1

Introduction to combinatorial optimization

This chapter presents a general outline about optimization methods, with a focus on the combinatorial optimization techniques met and implemented during my studies in France and my internship at the LAAS-CNRS of Toulouse. Although this content might be non-exhaustive, it helped me and could help the reader to have an insight on working principles of a set of optimization methods vastly used in literature and in commercial applications.

1.1 About mathematical optimization

Optimizing consists in finding the best element among a set of available ones by respecting some selection criteria. An optimization problem (also called mathematical programming problem) can be written as follows. Given a function $f: S \to \mathbb{R}$ from a set S to the real numbers, find an element x^* in S such that $f(x^*)$ is an extreme value of $f(x), \forall x \in S$. This problem consists of either a minimization or a maximization of f. Optimization problems are usually stated in terms of minimization, and the expression $f(x^*) \geq f(x) \Leftrightarrow -f(x^*) \leq -f(x), \forall x \in S$ allows the reversion of any problem direction. S is the domain of f, called *search space*; the elements of S are called *feasible solutions* and must respect all problem constraints. f is usually called *objective function*, sometime renamed *loss function*, *cost function*, or *energy function* depending on the application field. The solution x^* that satisfies all the constraints and minimizes (or maximizes) the value of f is called an *optimal solution* [**2, 3**].
Optimization is applied to many technical fields, such as logistics, production systems, economics, computer science and network design. As for other branches of applied mathematics, several analogies make it possible to apply similar optimization problems to very different fields. Frequently the objective functions of different problems contain similar terms, and the search space is often limited by constraints in a codified form (upper and lower bounds, balancing, incompatibility, etc.).

1.2 Combinatorial optimization

Discrete optimization consists in finding one of the best elements in a search space S with at least a component s defined in a discrete domain ($\exists s \in S \mid s \subseteq \mathbb{Z}$). Combinational optimization is a specification of discrete optimization in which some search space components are defined on a discrete and *finite* set. Problems concerning decision making, resource assignment, or any other field that needs a discrete search space require combinatorial optimization techniques to be solved. Some application fields are:

- **Graph theory**, in which the problem can be formulated as a graph exploration problem. (e.g., the Travelling Salesman Problem shown in chapter 2). The topic of this thesis work belongs to this category.
- **Games theory**, that concerns the strategic interactions of decision-makers with the aim of maximizing their performance.
- **Control theory**, that studies the influence and the effects of internal and external agents on a complex system of decision-making units.
- **Multi-disciplinary optimization** employed in the design of complex systems in which different technical aspects participate to the global quality of a product.

The search space of combinatorial optimization problems (COPs) is typically too large to allow an exhaustive search with a brute force algorithm. In fact, these algorithms usually have a factorial complexity and quickly become unsolvable. Some

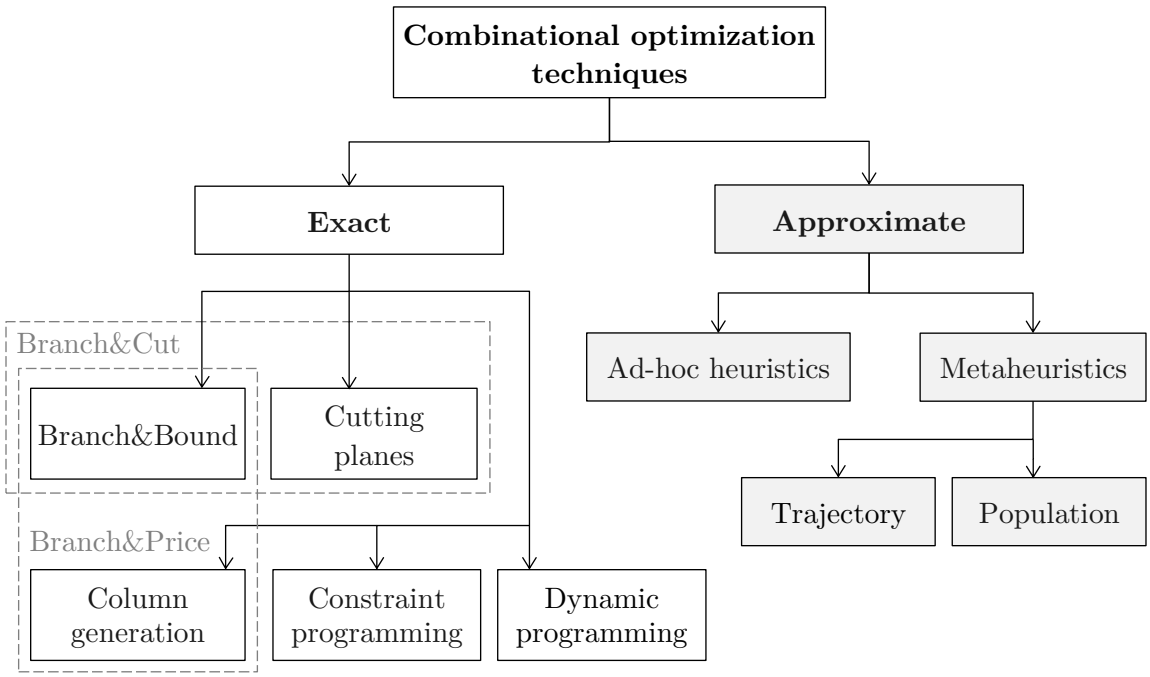


Figure 1.1. Classification of combinational optimization techniques.

combinatorial optimization problems can be solved exactly in polynomial time, for example by a dynamic programming algorithm, or by formulating them with an integer linear programming approach. However, in most cases the problem is NP-hard, and only a small group of methods can effectively find a solution. In practice, the acceptable complexity is often only polynomial; for many instances, approximate methods such as heuristics and metaheuristics are convenient with respect to exact methods. These methods find feasible solutions that approximate the optimum, sometimes with a known marge of error, in a generally reasonable amount of time.

1.3 Exact methods

In operations research (OR) and IT, exact methods consist of algorithms that aim to solve an optimization problem to global optimality. Among the exact methods of combinatorial optimization, the *branch-and-bound* is probably the most universally applied. It consists in the exploration of solution tree, subset by subset, starting from a given feasible solution x_0. The aim is to progressively find a feasible minorant of the upper vertex of solution tree. Some other useful techniques are the *cutting-plane* methods, often combined with *simplex* and *branch-and-bound* methods in a powerful linear-programming solving approach. The joint application of these three exact methods is codified under the name of *branch-and-cut*. It is about exploring the solution tree of the relaxed problem, in which integer constraints are initially lifted, then gradually reformulate them with *cutting-plane* methods until the initial problem is reobtained. In particular, the *branch-and-cut* method is used by all major commercial solvers for ILP and MILP problems (Mixed-Integer Liner Programming), as well as any other type of linearizable formulation. Other techniques to solve exact combinatorial optimization problems are *dynamic programming* – provided that problem optimum is reachable by breaking it down into subproblems and solving them to the optimum – and *constraint programming* – where there is no objective function and solution should just respect all problem constraints. Finally, it is worth mentioning the column generation methods, which apply to large linear problems and make it possible to solve them by considering only a subset of involved variables.

1.3.1 Dynamic programming

Dynamic Programming (DP) applies to broken-down problems for which it can be demonstrated that the optimality of subproblems also grants global optimality. This concept, formalized in the Bellman equation, was introduced by him in [**4**] for solving optimization problems.

Let us consider a generic optimization problem $max\{f(x) \mid x \in S\}$. DP can be applied to optimally solve the problem if the feasible region S can be divided into subsets $S_0 \subset S_1 \subset \cdots \subset S_n = S$, and global optimum can be reached by a set of sequential

solutions of growing size, such that the optimum in S_i is equal to the optimum in S_{i-1} plus the optimum of the problem $max\{f(x) \mid x \in S_i \cup S_{i-1}\}$, for each $i \in \{1, \dots, n\}$:

$$f(x^*) = \begin{cases} f(x, S_0) \\ f(x, S_i) = f(x, S_{i-1}) + f(x, S_i, S_{i-1}) \end{cases} \tag{1.1}$$

1.3.2 Linear programming

"Linear programming (LP) is an optimization method to achieve the best outcome in a mathematical model the requirements of which are represented by linear relationships. In fewer words, linear programming is a technique for the optimization of a linear objective function, subject to linear constraints. Its feasible region is a convex polytope, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function is a real-valued linear function defined in this polytope. A linear programming algorithm finds a point in the polytope where this function has its extreme value (min or max), if such a point exists." [**5**].

In their canonical form, LP problems concern the research of the optimal solution vector $x \in \mathbb{R}^n$ out of the statement:

$$\begin{aligned} &\textit{Maximize} && c^T x \\ &\textit{Subject to} && Ax \leq b, \\ & && x_i \geq 0 \end{aligned} \tag{1.2}$$

$c \in \mathbb{R}^n$ is the column vector of objective function coefficients, $b \in \mathbb{R}^m$ is the column vector of constraint constant terms, and $A \in \mathbb{R}^{m \times n}$ is the matrix of constraint coefficients. "The inequalities $Ax \leq b$ and $x \geq 0$ are the constraints which specify a convex polytope over which the objective function is to be optimized. In this context, two vectors are comparable when they have the same dimensions. If every entry in the first is less-than or equal-to the corresponding entry in the second, then it can be said that the first vector is less-than or equal-to the second vector." [**5**].
Each corner solution of the polytope x' contains *basic* x_i^N and *non-basic* x_i^{NB} variables. Variable $x_i' \in x'$ is said basic with respect to that corner if it assumes a non-zero, non-basic otherwise.

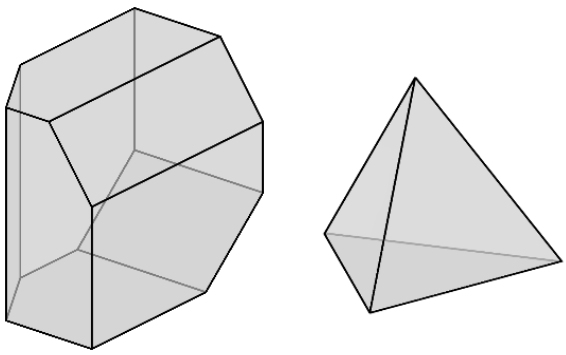


Figure 1.2. A generic 3-d *polytope* and a 3-d *simplex* (simplest polytope).

As shown in (1.2), the canonical form of a *maximization* LP problem requires all constraints to be written as *smaller-or-equal* inequations. However, in real-life problems *grater-or-equal* signs are very often required, and there are some techniques to write a generic LP formulation in its canonical form by adding auxiliary variables $y_k \geq 0$ to every *grater-or-equal* constraint.

$$\begin{array}{llcll} \textit{Maximize} & c^T x & \rightarrow & \textit{Maximize} & c^T x \\ \textit{Subject to} & Ax \geq b, & & \textit{Subject to} & Ax + y \leq b \\ & x \geq 0 & & & x, y \geq 0 \end{array} \tag{1.3}$$

Moreover, other than the canonical form, a LP formulation can be written in its *standard* form, in which all constraints are *equalities*. To do so, some *slack* variables $e_i \geq 0$ are added to each constraint to represent the distance between the hyperplane of the original constraint and the current solution vertex [**6**].

$$\begin{array}{llcll} \textit{Maximize} & c^T x & \rightarrow & \textit{Maximize} & c^T x \\ \textit{Subject to} & Ax \leq b, & & \textit{Subject to} & Ax + e = b \\ & x \geq 0 & & & x, e \geq 0 \end{array} \tag{1.4}$$

1.3.2.1 LP duality

Duality is an important property of mathematical programming. Given a *primal* linear program ($max\ c^T x\ s.t.\ Ax \gtreqless b$), it can be turned into its *dual* problem, ($min\ b^T u\ s.t.\ A^T u \gtreqless c$) by following some conversion rules:

Table 1.1 Primal-dual relationships in a LP.

Primal (dual)			Dual (primal)	
Maximization		→	**Minimization**	
Constraint i	$\leq$	→	Variable u_i	≥ 0
Constraint i	$\geq$	→	Variable u_i	≤ 0
Constraint i	$=$	→	Variable u_i	$\in \mathbb{R}$
Variable x_j	≥ 0	→	Constraint j	$\geq$
Variable x_j	≤ 0	→	Constraint j	$\leq$
Variable x_j	$\in \mathbb{R}$	→	Constraint j	$=$
	Cost c_j	→	Parameter c_j	
	Parameter b_i	→	Cost b_i	
	Coefficient a_{ij}	→	Coefficient a_{ji}	

Theorem of weak duality. Given a linear program, any feasible solution z of the primal formulation (written as a maximization) provides a lower bound for the optimal value w^* of the dual formulation (written as a minimization). Similarly, any feasible solution w of the dual formulation provides an upper bound for the optimal value z^* of the primal formulation. The statement with all signs reversed is also true.

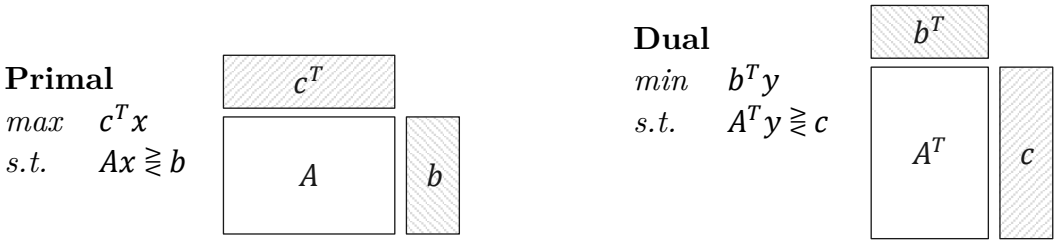


Figure 1.3. Representation of a primal problem and its dual.

Theorem of strong duality. Given a linear program, if its primal formulation has an optimal solution, then the dual formulation also has one, and they have the same value $z^* = w^*$.

1.3.2.2 Simplex algorithm

The simplex algorithm and its variants are widely used to solve LP problems. Their working principle was first introduced by George Dantzig in 1947 as an exact rigorous method to find the corner point of a n-dimensional convex polytope that maximizes an objective function $f: S \to \mathbb{R}$.

The algorithm requires a linear program written in its standard form and a first feasible solution to start. The first solution can be a trivial one (e.g., the origin $x = \vec{0}$) or obtained by solving a modified version of the problem. After that, the simplex algorithm is based on two fundamental considerations:

1. It can be demonstrated that if a value x^* exists in A such that $f(x^*) = \max\{f(x) \mid x \in S\}$, this value is geometrically located in a corner of problem's polytope.
2. It is also true that any linear objective function is weakly monotone along all convex polytope edges. Therefore, it can be stated that each edge connects two extreme points x_1, x_2 such that $f(x_1) \geq f(x_2)$, and that the edge can be 'crossed' by switching a *basic* variable with a *non-basic* one in x.

The simplex algorithm exploits these principles to 'walk' along the polytope edges and arrive, if it exists, to the optimum corner in a finite number of iterations (since finite is also the number of vertices in the polytope) [**7, 8, 9**].

1.3.2.3 Discrete linear programs: ILP and MILP

Even though simplex algorithm is designed to solve LP problems, it is often combined with relaxation techniques and heuristics for solving discrete linear programs. Given an optimization problem, if all its variables have integer domains the problem is called an *integer linear programming* (ILP) problem. Unlikely most LP problems,

the existence of integer or binary constraints makes ILP problems NP-hard, and do not always allow an effective resolution with existing methods.

If only a subset of variables has integer or binary domains, the problem is called a *mixed-integer linear programming* (MILP) problem. These problems are generally NP-hard, too, and are largely encountered in combinatorial optimization applications. Chapter 6 of this thesis is devoted to the developed MILP formulations.

1.3.3 Cutting-plane methods

Cutting-plane are a group of exact optimization methods for integer linear programs. They were first introduced by Ralph E. Gomory in the 1950s' and further developed by Václav Chvátal. "The cutting-plane method is any of a variety of optimization methods that iteratively refine a feasible set or objective function by means of linear inequalities, termed cuts. Such procedures are commonly used to find integer solutions to mixed-integer linear programming (MILP) problems, as well as to solve general, not necessarily differentiable, convex optimization problems." [**10**]. "Cutting plane methods for MILP work by solving a *non*-integer linear program, the *linear relaxation* of the given integer program. The theory of Linear Programming dictates that under mild assumptions (if the linear program has an optimal solution, and if the feasible region does not contain a line), one can always find an extreme point or a corner point that is optimal. The obtained optimum is tested for being an integer solution. If it is not, there is guaranteed to exist a linear inequality that separates the optimum from the convex hull of the true feasible set. Finding such an inequality is the separation problem, and such an inequality is a *cut*, that can be added to the relaxed linear program. This process is repeated until the best integer solution is found." [**10, 11**].

Cutting plane methods can also solve nonlinear problems as long as solution search space is convex and continuous. Kelley's method, Kelley–Cheney–Goldstein method, and bundle methods are commonly used to this purpose. They can solve "non-differentiable convex minimization, where a convex objective function and its subgradient can be evaluated efficiently but usual gradient methods for differentiable optimization cannot be used." [**10**]. The underlying principle of these nonlinear methods is to approximate the feasible region of a convex problem by a set of linear half-spaces enclosed in problem search space.

1.3.3.1 Gomory cut

In practice, the cutting-plane method proposed by Gomory is considered ineffective due to the many rounds often required to progress towards problem solution, besides being subject to numerical instability. Nevertheless, these methods gained in popularity during the 1990s', when Gérard Cornuéjols and his research team

demonstrated the effectiveness of combining cutting-planes with branch-and-bound methods, and developed some techniques to avoid numerical instability. Since then, cutting-plane methods are greatly employed in commercial solvers for discrete and combinatorial optimization. "Gomory cuts are very efficiently generated from a simplex tableau, whereas many other types of cuts are either expensive or even NP-hard to separate. Among other general cuts for MILP, most notably lift-and-project dominates Gomory cuts." [**11, 12**].

In the following, the working principle of Gomory cut is briefly discussed. Let us consider an ILP or MILP problem:

$$\begin{aligned} &\textit{Maximize} && c^T x \\ &\textit{Subject to} && Ax \leq b, \\ & && x \geq 0,\ x_j \textit{ integer} \end{aligned} \tag{1.5}$$

First, integer constraints are dropped, and the problem is solved in its relaxed continuous form. This solution is geometrically located in a vertex of the expanded polytope containing the whole feasible region of the original problem. If this vertex does not satisfy all the integer constraints, then it cannot be the sought solution, and a hyperplane is generated to separate it from the feasible integer points inside the polytope. To do so, an additional linear constraint – a Gomory cut – is added to the relaxed problem to cut the infeasible vertex out of the polytope. The relaxed problem with cutting-plane constraints is then solved, and the process is iterated until a feasible solution is found.

Other than Gomory, other cutting-plane techniques commonly used by commercial solvers are, e.g., *implied bound*, *projected implied bound*, *MIR*, *strong Chvátal-Gomory*, *flow cover*, *zero half*, *RLT*, *relax-and-lift*.

1.3.4 Branch-and-bound

Branch-and-bound (B&B) is the solving-paradigm of several exact algorithms for ILP problems, based on the progressive and systematic exploration of solution search space by splitting it and evaluating the resulting branches. Proposed by Alisa Land and Alison Doig in the 1960s', B&B is today one of the most used exact solving approaches for NP-hard problems, such as the Travelling Salesman Problem (presented in chapter 2) and many other combinatorial optimization applications.

Branch-and-bound algorithms start by considering the solution of a relaxed version of the problem, and as their name suggests, rely on two working principles:

1. Split the search space into smaller pieces by a defined criterion (branching).
2. Find the optimum on each branch, compare the results, and trim the branches for which it is proven they cannot contain the optimal solution (*bound*).

The algorithm below shows the basic working principle of branch-and-bound in the case of a maximization. Let us consider a ILP problem P_0 and its relaxed version P_{0R}. The symbol $\hat{P}$ indicates the problem currently being pointed by the algorithms, which is associated to a node in the search tree. Each problem P – thus each node – is a continuous LP that can be solved with the simplex algorithm. Its optimal solution and optimal solution objective value are indicated with x_P and $f(x_P)$. The algorithm also speaks of *closed* and *non-closed* nodes; in fact, once branches are explored, the algorithm closes and do not visit them anymore [**13**].

Algorithm 1.1. Branch-and-bound general algorithm

1 Initialize root-node $\overline{P} \leftarrow P_{0R}$ and the pointed-node $\hat{P} \leftarrow P_{0R}$
2 Initialize objective bound values $\overline{f} \leftarrow f(x_{P_{0R}})$, $\underline{f} \leftarrow 0$
3 **While** children($\overline{P}$) $\neq \emptyset$ **do**
4 **If** $\hat{P}$ is feasible **and** $f(x_{\hat{P}}) \geq \underline{f}$ **and** has any *non-closed* child
6 **If** $\hat{P}$ breaks any integral constraint of P_0
7 Descend of a level: $\hat{P} \leftarrow$ best non-closed node in children($\hat{P}$) generated with *branching criteria*. Each child of $\hat{P}$ is equal to $\hat{P} \cup$ {additional constraint}.
8 **Else** $\rightarrow$ feasible solution found
9 Close the node $\hat{P}$ (*bound*)
10 **If** $f(x_{P_0}) < f(x_{\hat{P}})$ **or** x_{P_0} still unfound
11 Titular best solution: $x_{P_0} \leftarrow x_{\hat{P}}$
12 New obj. lower bound: $\underline{f} \leftarrow f(x_{\hat{P}})$
13 **End if**
14 Climb of a level: $\hat{P} \leftarrow$ parent($\hat{P}$)
15 **End if**
16 **Else** $\rightarrow \hat{P}$ 'dead branch'
17 Close the node $\hat{P}$ (*bound*)
18 Climb of a level: $\hat{P} \leftarrow$ parent($\hat{P}$)
19 **End if**
20 **If** $\overline{P}$ has only one *non-closed* child
21 New active root-node: $\overline{P} \leftarrow$ *non-closed* child of $\overline{P}$
22 New obj. upper bound: $\overline{f} \leftarrow f(x_{\overline{P}})$
23 **End if**
24 **Continue**

The upper bound at line *2* can also be initialized via a heuristic method. This trick can help accelerate the B&B algorithm.
Concerning the *branching criteria* at line *7*, the simplest classic technique is splitting problem $\hat{P}$ into two subproblems, each with an additional constraint originated from a violated integral condition. More specifically, given an optimal solution $x^*_{\hat{P}}$ for which

the component $x^*_{\hat{P},j}$ is required to be integer but is not, two are the children problems originated from $\hat{P}$: $\hat{P} \cup \{x_{\hat{P},j} \leq \lfloor x^*_{\hat{P},j} \rfloor\}$ and $\hat{P} \cup \{x_{\hat{P},j} \geq \lceil x^*_{\hat{P},j} \rceil\}$.

1.3.5 Branch-and-cut

As it can be inferred from the paragraphs above, *branch-and-cut* algorithms adopt the branch-and-bound exploration approach with the application of cutting-plane methods before each branching step. The employed cutting-plane techniques can vary during the same solving instance. The locally unsatisfied integral conditions give indications about which cutting methods are likely to be the most effective for the algorithm. Although the simplex algorithm is generally quick, the B&B algorithm requires to run it a great number of times, and a good cutting approach has a remarkable impact on branch-and-cut duration.
Two are the main reasons to prefer a branch-and-cut approach over a simple branch-and-bound [**14**]:

1. To reduce the number of nodes to explore and the simplex instances to solve, thus accelerate the overall algorithm.
2. To reduce the number of explicit constraints in case the original problem has too many to be exhaustively enumerated.

1.3.6 Column generation

Column generation (CG) is a technique for solving large LP problems based on the duality and reduced cost concepts. Since duality do not apply to ILP, column generation cannot solve discrete optimization problems. The idea behind it is that many linear programs are too large to allow the exhaustive enumeration of all variables; in addition, it can be assumed that the basic (non-zero) variables in the optimal solution will only be a restricted subset of the whole. CG is hence a method to rewrite a linear program only considering the subset of variables that impacts the objective function [**15**].

First, the initial master problem (MP) is converted into the *restricted master problem* (RMP) only formed by a subset of MP variables. The choice of RMP variables can be made, for example, with a heuristic approach. After that, the RMP and its dual are solved for finding the optimal value of dual variables u_i, needed by CG to generate the *subproblem* (also called *pricing problem*), then check for RMP's optimality. The pricing problem consist in searching for new variables to add to RMP thought the search for the smallest (for a minimization) *reduced cost* in the reduced cost vector $\bar{c}$:

$$\bar{c}^* = min\{c_j - \textstyle\sum_{i=1}^{m} a_{ij} u_i, \quad \forall j \in [1, \dots, n]\} \tag{1.6}$$

If $\bar{c}^*$ is negative, it means that the associated variable can improve the solution, therefore it is added to RMP, which is solved again with its dual and the process repeats until no more negative reduced costs are found [**14**].

Reduced cost concept is essential to the column generation method. "Reduced cost $\bar{c}_j$ is the amount by which an objective function coefficient c_j would have to improve (so increase for maximization problem, decrease for minimization problem) before it would be possible for a corresponding variable x_j to assume a positive value in the optimal solution." [**16**].

1.3.7 Branch-and-price

The column generation method can also help solving large ILP and MILP problems if combined to branch-and-bound algorithms. *Branch-and-price* is the name of some algorithms based on tree exploration (like branch-and-bound) and exploiting column generation to solve the large relaxed linear programs before each branching [**14**].

On the other hand, branch-and-price algorithms are not easy to use due to subproblems, sometimes hard to solve, and to the need to find effective branching techniques for avoiding an uncontrolled constraint propagation.

1.4 Heuristic methods

"A *heuristic* technique (from Greek εὑρίσκω 'to find, discover') is any approach to problem-solving that uses a practical method and various shortcuts in order to produce solutions that may not be optimal, but are of a sufficient quality given a limited timeframe or deadline." [**17**]. Heuristic methods are usually problem-dependant, and "are used for quick decisions, especially when finding an optimal solution is either impossible or impractical and when working with complex data." [**17**]. "A heuristic function ranks alternatives at each branching step of a search algorithm, deciding the branch to follow based on available information." [**18**].
Heuristic methods do not backtrack what they progressively find, that means they do not have an improvement phase after the exploration one.

1.5 Metaheuristic methods

As stated above, heuristics are problem-dependent methods. As such, they need to be manually adapted to any new problem to take full advantage of problem characteristics and peculiarities. In addition, unless proven otherwise (e.g., by demonstrating the objective function is convex), heuristic methods usually end their search in local optima and fail, in general, to reach the global optimum.

On the other hand, metaheuristics are problem-independent methods. Their working principles are not as greedy as heuristic techniques, and they generally succeed in reaching better solutions. However, they still need to be tuned at hand depending on the problem to improve, as their working parameters have a remarkable impact on results. In general, they need a solution to start, then explore the search space around them trying to increase solution quality with a trade-off between randomized and local deterministic search. In some cases, they can accept some partial solution deteriorations to get out of a local optimum point (this is the case, e.g., of simulated annealing algorithm, in which adverse solution fluctuations are accepted on a stochastic basis).

A first metaheuristic classification divides metaheuristics into *trajectory-based* and *population-based*. The main methods belonging to these two categories are shown in the chart at figure 1.4. The structural difference between them lies in memory usage. In fact, trajectory-based metaheuristic are *memory-less* algorithms in which "the next state only depends on the information accumulated in the current state of the search process, as a Markov process." [**19**]. On the contrary, population-based methods are also known as *memory-usage* algorithms, as "there is a usage of short and/or long-term memory. Usually, the first keeps track of recently visited solutions (moves), while the second has a wider information storage concerning the entire search process." [**19**].

1.5.1 History of metaheuristic methods

As for most optimization techniques, the history of metaheuristics is coeval with computer evolution. The first landmark came with the development of *evolutionary algorithms* by I. Rechenberg and H. Schwefel in the 1960s'. Some years later, J. Holland proposed the *genetic algorithms* is his seminal book published in 1975. In 1983, S. Kirkpatrick developed the *simulated annealing*, that takes inspiration by metals annealing process, and in the same years F. Glover was working on his tabu

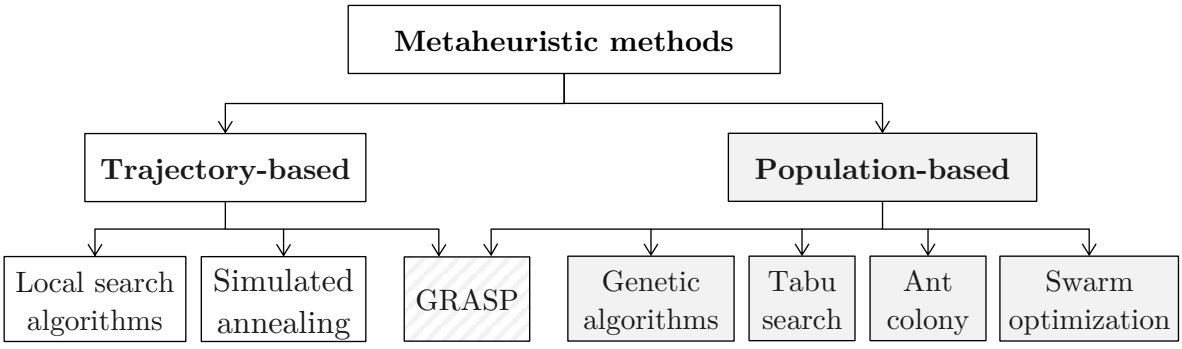


Figure 1.4. Partial classification of metaheuristic methods.

search, then published in 1997. In tabu search algorithms, a tabu list contains the elements to avoid to progressively get to the best feasible one.
The innovative *ant colony optimization* was proposed in 1992 by M. Dorigo in his PhD thesis, and in the same year J. R. Koza published a revolutionary book about genetic programming. Between 1995 and 1997, J. Kennedy and R. C. Eberhart proposed the *particle swarm optimization*, while R. Storn and K. Price developed the vector-based *differential evolution algorithm*, that proved to be better than genetic algorithms in many cases [**20**].
Innovation trend kept on during 21st century. "First, Z. Woo Geem developed the *harmony search algorithm* in 2001, a music-inspired algorithm. Around 2002, a *bacteria foraging algorithm* was developed by Passino. In 2004, S. Nakrani and C. Tovey proposed the *honeybee algorithm* and its application for optimizing internet hosting centres, which was followed by the development of a novel bee algorithm by D. T. Pham and the *artificial bee colony* by D. Karaboga in 2005. In 2009, X. Yang and S. Deb introduced an efficient *cuckoo search algorithm* and proved it to be far more effective than most existing metaheuristic algorithms." [**20**].

1.5.2 Genetic algorithm

Genetic algorithms are abstractions of biological behaviours based on Charles Darwin's theory of natural selection. Many variants of genetic algorithms have been developed and applied to a wide range of optimization problems, such as graph colouring, pattern recognition, for both discrete (e.g., the travelling salesman problem) and continuous systems (e.g., aerospace airfoil design). *Crossover*, *recombination*, *mutation* and *selection* are the essential components of genetic algorithms' solving strategy. "The essence of genetic algorithms involves the encoding of solutions as arrays of bits or character strings (chromosomes), the manipulation of these strings by genetic operators and a selection based on their fitness to find a solution to a given problem." [**20**].
Figure 1.5 shows how each algorithm iteration gives birth to a new generation of chromosomes, that is the current solution population. The objective function is usually encoded in a fixed-length binary or real array. The most critical issues of genetic algorithms concern the choice of proper fitness function and selection criteria, as well as the extent of crossover and mutation operations. Transferring some good chromosomes from a generation to another without much change is called *elitism*. "The basic procedure consists in selecting the best chromosome (in each generation) which will be carried over to the new generation without being modified by the genetic operators. This ensures that a good solution is attained more quickly." [**20**].

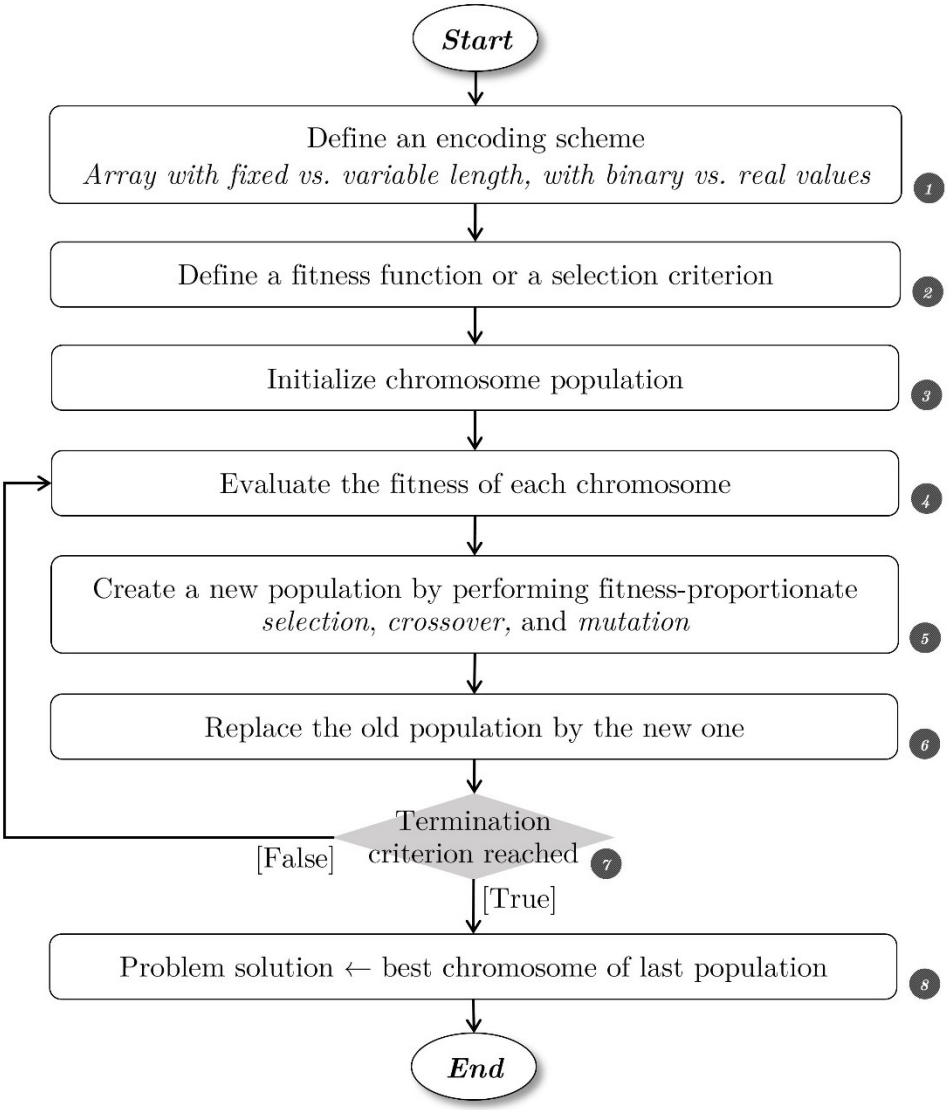


Figure 1.5. Flowchart of a generic genetic algorithm.

1.5.3 Simulated annealing

Simulated annealing (SA) is based on the metal annealing process and the statistical mechanics principles that rule it. It was specifically presented to solve combinatorial optimization problems with an explicit strategy to avoid being trapped in local minima. A well-tuned SA has the potential to converge either to global optimum, or to a very good solution close to it.

Problem solution is iteratively perturbated by some fluctuations similar to those of genetic algorithms (mutation, crossover, etc.). "Essentially, SA is a search along a Markov chain, which converges under appropriate conditions." **[20]**. If the energy of the new system is lower than the previous one (i.e., new solution is 'more optimal'), the new solution is retained. However, the most characteristic aspect of SA is the

acceptance in some circumstances of search moves that do not improve the optimality of current solution. The acceptance of these apparently adverse changes – adverse in the short-term, but likely beneficial to final result – is calculated with the Metropolis-Hastings algorithm as:

$$p = \exp\left(-\frac{\Delta E}{k_B T}\right) \tag{1.7}$$

p is the acceptance probability, and this exponential expression is known as Boltzmann's distribution. T is the temperature that rules SA progression; k_B is the Boltzmann's constant; ΔE is the energy gap between the new and the old solution, that is proportional to objective function variation ($\Delta E \propto \Delta f$). From the Boltzmann's distribution at (1.7), it can be inferred that as temperature increases, the probability of retaining an adverse fluctuation decreases. Also, a wider positive energy gap results in a smaller chance of retaining the new solution. In practice, the adverse solution is retained if the value drawn from a simple uniform distribution $\mathrm{U}(0,1)$ is smaller than

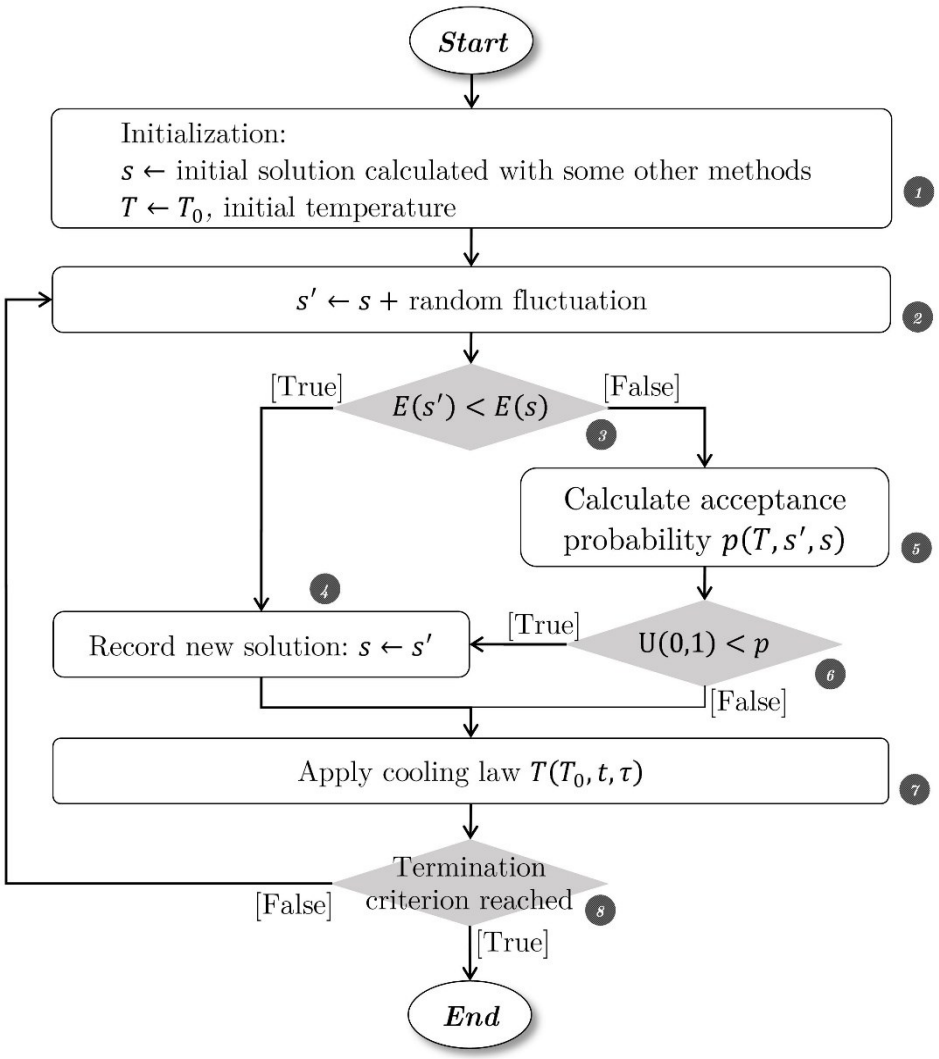


Figure 1.6. Flowchart of simulated annealing.

$\exp(-\Delta E/k_B T)$. After solution evaluation and replacement, a cooling low is applied to decrease temperature T and make the algorithm progress:

$$T = T_0 \cdot \exp(-t/\tau) \tag{1.8}$$

Where T_0 is the initial SA temperature, t is the fictive time that increases by one at each iteration, and τ is the cooling parameter. SA general algorithm is shown in figure 1.6. The termination criterion is usually a lower temperature limit T_{min}, so that SA ends when $T \leq T_{min}$. The values of T_0, τ, T_{min} and the extent of fluctuations must be carefully chosen to obtain good solutions [**20, 21**].

1.5.4 Tabu search

Tabu search (TS) is a memory-based algorithm largely used in combinatorial optimization. Unlikely simulated annealing, TS stores a number of tried solutions in a *tabu list* to avoid circularities, escape local minima, and speed up local search. "studies show that the use of tabu lists with integer programming can save computing effort by at least two orders of magnitude for a given problem, as compared with standard integer programming." [**20**]. As shown at point ❸ of figure 1.7, the tabu list has a limited size called *tabu tenure* and is dynamically filled and emptied as the algorithm progresses, often with a simple FIFO technique. Termination criterion can either be the achievement of a solution with the desired characteristics and quality, or the depletion of local search space $\{s' \mid s' \in \Gamma(s) \setminus TabuList\}$. Given a solution s, $\Gamma(s)$ is the set of neighbour solutions reachable from s with an elementary local search move.

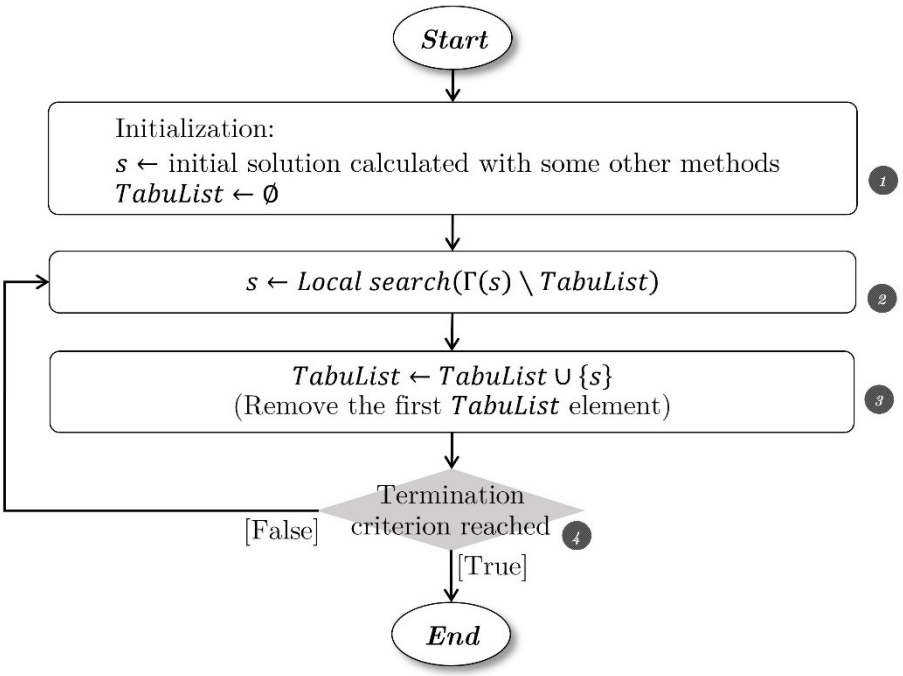


Figure 1.7. Flowchart of a simple tabu search.

It could be argued that keeping in memory a bunch of complete solutions can result inefficient. Therefore, it is not uncommon to only store a set of solution attributes that grant a univocal solution identification. "Attributes are usually components of solutions, moves, or differences between two solutions." [**21**]. Obviously, a special care must be put in the choice of attributes that minimize the risk of ambiguities.

1.5.5 Ant colony optimization

Ant colony optimization (ACO) is inspired by the social behaviour of ants and their interaction with the surrounding environment. "Each ant lays chemical scents called pheromones to communicate with others, and is also able to follow the route marked with pheromone laid by other ants. When an ant finds a food source, it will mark the trail to and from it with a defined quantity of pheromone." [**20**].
Once laid, pheromone has a concentration φ_0 that evaporates with time at the exponential rate ξ shown at (1.9). Pheromone evaporation is crucial to algorithm convergence to a self-organized state.

$$\varphi(t) = \varphi_0 \exp(-\xi t) \tag{1.9}$$

Let P be a CO problem with a finite $n \times m$-dimensional search space $S = \{x_{ij}\}_{i\in\{1,\dots,n\},j\in\{1,\dots,m\}}$, and an objective function $f: S \to \mathbb{R}$. A pheromone value is associated to any possible variable assignment. I.e., a time-dependent pheromone value $\varphi_{ab}^{i}(t)$ exist for each endomorphism $x_{ia} \mapsto x_{ib}$. Given a variable component i with values x_{ia}, an ant chooses the next variable value x_{ib} with a stochastic mechanism. However, if edge (a, b) presents a pheromone track, the probability of choosing it is proportional to pheromone concentration $\varphi_{ab}^{i}(t)$. In turn, $\varphi_{ab}^{i}(t)$ is enhanced by the number of ants crossing (a, b) to search for food on the other side. As the algorithm evolves, the overall favourite path emerges as the most convenient one. This is the basic working principle of any ant colony algorithm [**22**].

1.5.6 Particle swarm optimization

Particle swarm optimization (PSO) is based on the swarm behaviour of some animal species, such as fishes and birds. "PSO has been applied to almost every area in optimization, computational intelligence, and design/scheduling applications. There are at least two dozens of PSO variants, and a much larger number of hybrid algorithms obtained by combining PSO with other optimization techniques, which are increasingly popular." [**19**]. The working principle of PSO consists of piecewise particle trajectories formed by three components (bold letters indicate vectors):

- a deterministic component that tends to attract each particle toward the global best solution $\boldsymbol{X}^*$;
- a deterministic component that tends to attract each particle towards the best solution found for the particle itself $\boldsymbol{x}_i^*$;

- a stochastic component that randomly moves the particle in the search space.

Each particle is a candidate solution for the problem and is associated to a time-dependent positional vector with a position and a velocity [**23**]. "PSO is a metaheuristic, as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. Also, PSO does not use the gradient of the problem being optimized, which means PSO does not require that the optimization problem be differentiable as is required by classic optimization methods such as gradient descent and quasi-newton methods." [**24**].

At each time t, each particle i has a position vector $\boldsymbol{x}_{i,t}$ and velocity vector $\boldsymbol{v}_{i,t}$. For every PS iteration, the new velocity vector of i is calculated as:

$$\boldsymbol{v}_{i,t+1} = \theta \boldsymbol{v}_{i,t} + \phi_s r_s \left(\boldsymbol{X}^* - \boldsymbol{x}_{i,t}\right) + \phi_c r_c \left(\boldsymbol{x}_i^* - \boldsymbol{x}_{i,t}\right) \tag{1.10}$$

θ is the *inertia parameter* that plays a stabilizing role and must be smaller than 1 to prevent swarm divergence. Typically, $\theta \in [0.5,0.9]$. ϕ_s and ϕ_c are respectively called *cognitive* and *social coefficient*, as they determine these two aspects of particle behaviours. In general, ϕ_s, $\phi_c \approx 2$. r_s and r_c are two random coefficients drawn from a uniform distribution $\mathrm{U}(0,1)$ that determine the extent of the component to which they are associated. Consequently, the new position of i is calculate as:

$$\boldsymbol{x}_{i,t+1} = \boldsymbol{x}_{i,t} + \boldsymbol{v}_{i,t+1} \tag{1.11}$$

As the algorithm progresses, the best particle swarm solution $\boldsymbol{X}^*$ could converge toward the global optimum. As for other metaheuristic, the choice of parameters θ, ϕ_s and ϕ_c has a major impact on solution quality, and many research works are focused on PSO parametrization procedures [**19**].

1.5.7 Harmony search

Harmony Search (HS) is a relatively new metaheuristic based on music composition. The HS optimization moves are inspired by the following three musician exercises [**20**].

Musicians	**HS optimization**
Play a piece of music they know	Usage of harmony memory
Play a piece arranged from a known one	Pitch adjustment
Compose new music or random notes	Randomization

The basic HS method is composed of four principal steps [**25**]:

1. The HS memory (HSM) is initialized with a set of m random solutions to the problem: $\left\{x^m \in \mathrm{HSM} \,\middle|\, x^m = \{x_1^m, x_2^m, \dots, x_n^m\}\right\}$.

2. Generate a new solution $x' = \{x'_1, x'_2, \dots, x'_n\}$ from HSM elements. Each component x'_i is randomly chosen from an existing x_j^m and mutated with a *pitching adjust rate*. This step of HS is similar to genetic algorithm's crossover and mutation.
3. If x' is better than the worst solution in HSM, it takes its place, elsewise it is discarded.
4. Repeat steps 2 and 3 until a termination criterion is met. For example, a solution in HSM has the desired quality, or the maximum number of iterations is reached.

1.5.8 GRASP

The *Greedy Randomized Adaptive Search Procedure* (GRASP) is a multi-start iterative metaheuristic in which each iteration presents two steps:

1. *Construction*, in which a solution is built with a greedy heuristic.
2. *Local search*, that tries to repair the solution obtained at point 1 in case it is infeasible. If local search cannot reach feasibility, the solution is discarded and a new initial solution is sought. Otherwise, the feasible solution is kept, and a new neighbourhood is recursively searched until a local minimum is reached.

GRASP is based on the multiple resolution of the problem with a logic improvement pattern. The *randomized* component of GRASP lies in its multi-start approach, as the initial solutions sought at point 1 are obtained with different greedy parameters. Once the algorithm is over, the best solution found during its execution is picked as problem solution [**26**].

GRASP was introduced by [**27**] in 1989, and its first application concerned the *set covering problem*. Today this flexible algorithm is widely used in a number of hybrid applications, often combined with other metaheuristic such as tabu search, simulated annealing, and genetic algorithms.

Blank page

CHAPTER 2

Overview on Routing Problems

Vehicle Routing Problems (VRPs) are an important class of problems in operations research and combinatorial optimization. The common goal is to determine the routes of a vehicle fleet to perform some tasks at a certain number of space-distributed stations under a series of constraints. The objective is minimizing the cost of problem solution based on many factors, such as distance and penalties. This kind of problem is generally applied to logistic and transport science problems; however, they can easily be adapted to other fields due to the strong analogies they have with other organizational problems, such as circuits (electrical, hydraulic, etc.) and computer networks design.

This chapter shows the basic concepts of VRPs and progressively adds complexity elements that are relevant to the topic of this thesis.

2.1 Travelling Salesman Problem

The statement of the TSP (Traveling Salesman Problem) is as follows. Given a set of n points (also called *cities*) and the pairwise distances between them, find the closed path in which each city is visited once and only once and the total travelled distance is minimal. Formally, the problem consists in finding the shortest Hamiltonian cycle in a graph $G = (V, E, c)$, where V is the set of vertices (cities), E is the set of edges, and c is the set of costs associated to each edge. To make sure the problem is solvable, an associated Hamiltonian graph must exist [**28**].

This problem could look easy, but is still one of the most studied in combinatorial optimization and belongs to the list of 21 NP-complete problems of Karp. The TSP is NP-complete as no known method can grant an effective resolution for any problem instance. *Non-deterministic Polynomial-time* (NP) problems are a class of computational decision problems solvable by a non-deterministic Turing machine in polynomial time. NP-complete problems are defined as the hardest NP problems [**29**].
The TSP is said *symmetrical* if all edges are bidirectional and both directions have the same weight, *asymmetrical* otherwise.

A brute-force algorithm for solving this problem has a factorial complexity $O(n!)$, more precisely equal to $(n-1)!/2$. Thus, problem resolution with this kind of algorithms becomes soon impractical also with a few cities [**28**]. Held and Karp demonstrated that dynamic programming algorithms can solve the TSP with a time complexity of the order $O(n^2 2^n)$ [**30**]. "Among the methods known today, the best for the resolution of the TSP have proved to be linear optimization and approximate methods such as heuristics and metaheuristics. Linear programming can solve considerably large problems, even if solving times can be important."
As long as problem size is reasonable, the TSP can be solved with a ILP formulation; given a set of cities $\mathcal{C} = \{\text{dep}, 1, \dots, n\}$ such that the travelling cost c_{ij} is known for each pair (i,j), find the best set of binary connection variables x_{ij} such that each city is visited once and only once, and the total problem cost is minimized:

$$\begin{array}{ll} \textit{Minimize} & \sum_i^n \sum_{j \neq i}^n c_{ij} x_{ij} \\ \textit{Subject to} & \sum_i^n (x_{ij} + x_{ji}) = 1,\ \forall j \in \mathcal{C} \text{ (visit unicity constraints)} \\ & x_{ij} \geq x_{jk},\ \forall i \in \mathcal{C} \setminus \{\text{dep}\}, \forall k \in \mathcal{C} \text{ (route continuity constraints)} \\ & x_{ij} \in \{0,1\} \text{ (binary constraints)} \end{array} \tag{2.1}$$

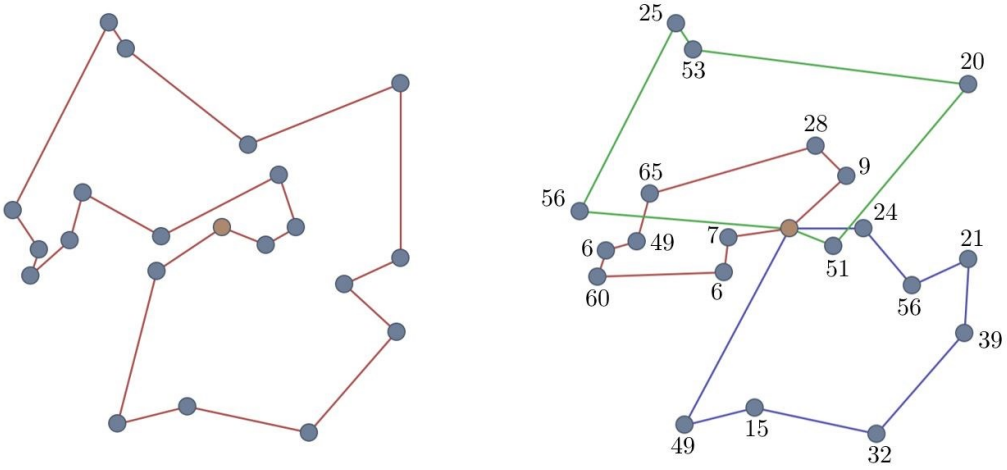
2.2 Vehicle Routing Problems

The TSP is a Vehicle Routing Problem in its simplest form, where a single vehicle has to visit a set of cities through the shortest path. More generally, VRP family is about finding the most efficient way to move passengers or goods in a network of stations, subject to a set of constraints. The optimization objective is generally about minimizing travelled distance, but other terms can integrate it or even take its place in some cases. As stated before, the most classic instance is the Traveling Salesman Problem (TSP), where the aim is finding the shortest path that starts at a central depot, visits all destinations once and only once, and finally returns to starting point.

Some other common VRP variants are known as CVRP (Capacitated VRP), VRP-TW (VRP with Time Windows), TD-VRP (Time-Dependent VRP), and VRP-PD (VRP with Pickup and Delivery). Book [**31**] presents a vast collection of VRP variants and their solving methods.

2.2.1 Capacitated VRP

The *Capacitated Vehicle Routing Problem* (CVRP) adds a major logistic issue to the VRP, that is the quantification and the constraining of moved goods or people. This is a basic aspect of all practical VRP problems. In fact, there is no interest in moving a vehicle without considering the transported items.
CVRP was first presented in 1959 in a paper called '*The Truck Dispatching Problem*', "concerned with the optimum routing of a fleet of gasoline delivery trucks between



TSP **CVRP**
vehicles with capacity $Q_k = 250$

Figure 2.1. Visual comparison between TSP and CVRP.

a bulk terminal and a large number of service stations supplied by the terminal. The routes between any two points in the system are given, and a demand for one or several products is specified for every station within the distribution system." [**32**] The aim was creating a set of truck routes capable of satisfying station demands such that the overall travelled distance was minimized. The problem was solved with a near-optimal LP formulation and tested on some fictive instances.
In general, the capacity of problem vehicles does not allow to find a single route that satisfies the needs of all the cities. As shown in figure 2.1, CVRP solutions usually involve more than a route, to be covered either by an equal number of vehicles in parallel, or sequentially by a smaller set of vehicles.

The CVRP is widely studied, and several heuristic and metaheuristic solving approaches are present in the literature. Exact optimization is also largely spread, mainly as MILP formulations. Assuming the number of available vehicles is known, a CVRP formulation can be written by adapting the linear program at (2.1) with some additional parameters, variables, and equations:

Additional parameters and variables:

d_i (parameter) demand of city i, must be greater than zero,
Q_k (parameter) capacity of vehicle $k \in \mathcal{V} = \{1,2,\dots,m\}$,
q_0^k (parameter) initial content of vehicle k, usually equal to zero,
q_i^k (variable) content of vehicle k after visiting i.

Additional constraints:

$$q_i^k + d_i = q_j^k \Leftrightarrow x_{ij} = 1, \forall i,j \in \mathcal{C}, k \in \mathcal{V} \text{ (content evolution constraints)} \quad (2.2)$$
$$q_i^k \leq Q_k, \ \forall k \in \mathcal{V} \text{ (capacity constraints)}$$
$$q_i^k \in \mathbb{R}.$$

2.2.2 VRP with Time Windows

The *VRP with Time Windows* (VRP-TW) adds further complexity to the VRP and CVRP. It is about introducing some time-related aspects to the formulation, namely by imposing that each delivery is completed within a specified time-window. Thus, time dimension enters the VRP model. The VRP-TW is particularly useful to represent the situations in which a delivery deadline is imposed, or when destinations are not open at all time. There could also be some start-times before which a delivery shall not be made, for example to avoid overstock at a destination. To this purpose, vehicles could be given the change to wait at some location if there is no delivery that can be made in that moment.
These time-related features can either enter the model as constraints, or as objective function terms by putting a penalty on retards. The second option can sometimes relax the model and enhance solvability in presence of excessively rigid constraints.

2.2.3 Time-Dependent VRP

In real-life problems, especially those in urban or congested areas, travelling times not only depend on distance, but also on traffic conditions at passing hour. The *Time-Dependent VRP* (TD-VRP) considers this aspect by associating a different travelling cost c_{ij} to each (i, j) connections as a function of the time at which they are crossed. E.g., crossing a sector of Paris' ring-road can take much longer at 5 p.m. with respect to the same stretch at 3 p.m. Traffic aspects are also very important to *green* variants of VRP aiming at minimizing energy waste. Anyway, a traffic time-model is required in input to any TD-VRP.
[33] makes some key considerations about the gap between VRP models and real-life situations, especially focusing on travel time variability. It also presents a GRASP heuristic for the TD-VRP with time windows applied to Turin's road network and demonstrates that polynomial functions are suitable for representing real speed fluctuations along roads.

2.2.4 VRP with Pickup and Delivery

The *VRP with Pickup and Delivery* (VRP-PD) allows both loading and unloading operations in a same route. In general, each delivered item shall be previously loaded at some point in the route. The VRP-PD is crucial to optimize fleet operations in a model composed by multiple sources and destinations. In fact, for a company the rentability of vehicles is partially proportional to the average fill rate, and only-load/unload routes seldom allow a fill rate higher than 50%. In parallel, a higher fill rate decreases gas emission and has an important role in *green* routing problems.
The VRP-PD was investigated in **[34]**. The paper proposes a cooperative approach in which a logistic operator delivers goods and collects waste simultaneously. The

problem is solved with a heuristic method, tuned on some fictive instances, and successfully tested on a real-life application case.

2.3 Inventory Routing Problems

The *Inventory Routing Problems* (IRPs) combine VRPs with the *Vendor-managed Inventory* (VMI) model. In opposition to traditional inventory management, in which the retailer asks for a precise order quantity, in the VMI the information about stock levels is shared with the vendor (supplier), so that the vendor is the decision-maker about order quantity and time. This inventory management system promotes buyer-vendor cooperation and showed operational advantages and cost reduction. The supplier benefits from a wider knowledge about clients' inventories and can guarantee a better delivery service, while the retailer saves the resources previously employed for active inventory monitoring.

[35] makes a clear and practical distinction between IRPs and classical VRPs. It states that "VRPs occur when customers place orders and the delivery company, on any given day, assigns the orders for that day to routes for trucks. In inventory routing problems, the delivery company, not the customer, decides how much to deliver to which customers each day. There are no customer orders. Instead, the delivery company operates under the restriction that its customers are not allowed to run out of product. Another difference is the planning horizon. VRPs typically deal with a single day, and the only requirement is that all orders have to be delivered by the end of the day. Inventory routing problems deal with a longer horizon. Each day the delivery company makes decisions about which customers to visit and how much to deliver to each of them, while keeping in mind that decisions made today impact what has to be done in the future. The objective is to minimize the total cost over the planning horizon while making sure no customers run out of product." **[35]**.

2.3.1 Basic IRP statement

Let $G(\mathcal{R}, \mathcal{D})$ be a graph composed by a set of vertices (1 supplier and R retailers) $\mathcal{R} = \{\text{supplier}, 1, \dots, R\}$ and a set of arcs (connections) $\mathcal{D} = \{(i,j) \mid i,j \in \mathcal{R}, i \neq j\}$. In a discrete-time IRP formulation, the planning horizon T_P is divided into a set of same-length timeframes $\hat{T} = \{0,1, \dots, T\}$. For each $t \in \hat{T}$, every retailer i has a consumption $\gamma_i(t)$ and a storage cost s_i (cost·content^{-1}·time^{-1}). Each arc in $\mathcal{D}$ has a distance d_{ij}. Let $\mathcal{V} = \{1,2, \dots, V\}$ be a set of vehicles, each with a capacity Q_k and a transportation cost c_k (cost·load^{-1}·distance^{-1}) **[36]**. The IRP aims at finding a set of distribution routes, one for each vehicle, such that there is no stockout and the total inventory-transportation cost is minimal. For each delivery, three interdependent decisions must be taken:

1. when to deliver,

2. how much to deliver,
3. how to insert the delivery in a route.

Other than this basic version, real-life applications of IRP present many more aspects to consider and integrate within the model. [**37**] identifies seven characteristics that affect IRP model building. They are listed in the table below.

Table 2.1. IRP characteristics and variations.

Characteristic	**Alternatives**			
Time	Instant	Finite	Infinite	
Demand	Stochastic	Deterministic		
Topology	One-to-one	One-to-many	Many-to-many	
Routing	Direct	Multiple	Continuous	
Inventory type	Fixed	Stock-out	Lost sale	Back-order
Fleet	Homogeneous	Heterogeneous		
Fleet size	Single	Multiple	Unconstrained	

The basic IRP stated above is NP-hard as any problem derived from the TSP. Some exact approaches are found in literature, but heuristic methods are used for the most to solve IRPs.

2.3.2 Production Routing Problem

The *Production Routing Problem* (PRP) is an extension of the IRP in which a superior supplying echelon (a production plant or another supplier) is added before the direct vendor.
Paper [**38**] studies the case of a single supplier in charge of distributing different products to a set of retailers (see figure 2.2). Both the supplier and the retailers have limited storage capacity, and the supplier can place orders to the plant at a fixed cost in order to get the required articles. The objective of the problem is minimizing the total cost of plant orders, inventory holding, and distribution.

2.4 Bus Routing Problem

The *Bus Routing Problem* (BRP) is a specific multi-objective version of the VRP with some interesting characteristics. “The problem of scheduling and routing school buses deals with the important question of how to transport pupils to and from schools in the safest, most economical and most convenient manner.” [**39**]. One peculiar aspect of the BRP is that transporting pupils is a delicate and demanding task. In fact, each pupil would like to be transported in the shortest possible time, and its

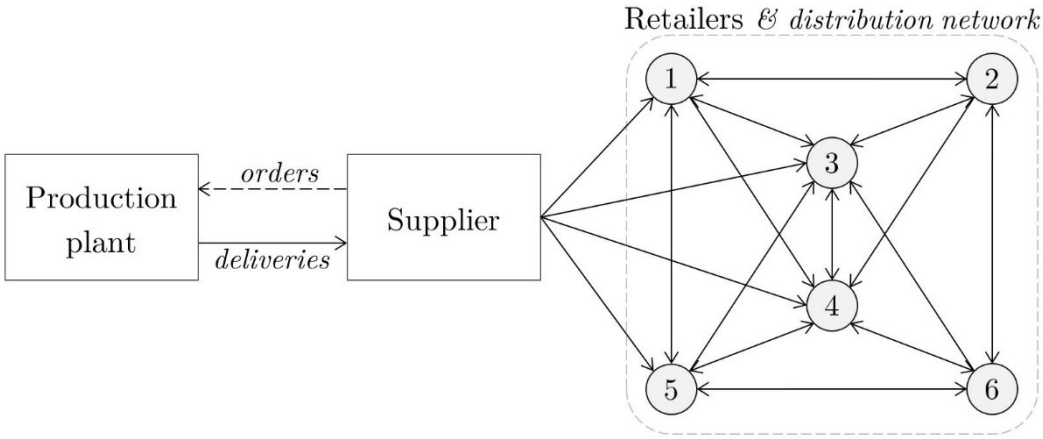


Figure 2.2. Graphic scheme of the Production Routing Problem.

impatience is likely to increase more-than-linearly as the service takes longer. In the BRP there are three aspects to minimize **[39, 40]**:

- the cost of the service (usually proportional to total distance),
- the number of buses required,
- the dissatisfaction of pupils (function of the time that each pupil spends onboard).

Blank page

CHAPTER 3

The Multi-item IRP with Pickup and Delivery

The core topic of this thesis is the *Multi-item Inventory Routing Problem with Pickup and Delivery*, abbreviated as Mi-IRP-PD in the following. As the name suggests, it is an extended version of the Inventory Routing Problem with both pickups and deliveries of many diverse items. Basically, the problem is composed of a set of stocks and a set of capacitated vehicles. Each stock can only contain a defined subset of items and has some consumption values as functions of time. The objective is finding a set of vehicle routes that satisfies every stock demand and does not break any capacity, time-coherence, or item-coherence constraints.

This chapter presents some cases of Multi-item IRP and IRP-PD found in literature, then makes a prior introduction to the Mi-IRP-PD version discussed in this thesis.

3.1 Multi-product IRP

The multi-product (or multi-item) aspect is the first to consider for modelling the Mi-IRP-PD. The previous chapter presents many variants of the VRP and the IRP in which product variety is not mentioned. However, many practical applications of the IRP require the distribution of more than a unique type of article. E.g., the works at **[41, 42]** discuss the *Multiproduct Multivehicle Inventory Routing* (MMIRP) and propose two mixed-integer linear programs to solve it exactly.

[41] shows the case of the Indonesian LPG supply chain, with two different articles to distribute: the 5.5. kg and the 12 kg LPG cylinders. A two-vehicles fleet is in charge of dispatching the products over a total of 46 clients scattered in Malang-City. The exact solving procedure yields for each vehicle a Monday-to-Friday schedule composed of six closed daily routes. Hence, every vehicle serves an ordered subset of clients each day, and the delivered quantities of each product are decision variables, too. The objective function includes three terms: the fuel cost per kilometre, the

fuel cost per kilogram onboard, and the customer inventory cost. Although MILP solution includes more vehicle routes with respect to the planning traditionally used by the company (i.e., more visits to the central stock), it allows to reduce fuel consumption by 13.39% and inventory cost by 16.01%.
On the other hand, paper [**42**] discusses the MMIRP, too, and introduces some *consistency features* to increase service quality. These features have the purpose of making the model more realistic and be compliant with some concerns about workforce management and regularity of service. (E.g., establishment of a trust relationship between some customers and a specific driver; existence of a reasonable non-visit time to a same customer after a delivery.)

In a linear program, the multi-product aspect can be handled by adding a further article index, hence splitting and specifying all the load-related variables and constraints. For example, let $\mathcal{V} = \{k \mid k = 1, \ldots, K\}$ be a set of vehicles with capacity Q_k. Also, let $\mathcal{A} = \{\theta \mid \theta = 1, \ldots, \Theta\}$ be a set of products. Content ($q^{k,\theta}$) and loading ($L^{k,\theta}$) variables are defined for each pair (k, θ). $q^{k,\theta}$ indicates the quantity of θ on k at a certain moment, while $L^{k,\theta} \geq 0$ indicates the quantity of θ loaded by k (thus, $\Delta q^{k,\theta}$) at a certain moment. The following expressions can describe capacity and loading constraints for a MMIRP:

$$\sum_{\theta \in \mathcal{A}} q^{k,\theta} \leq Q_k, \ \forall k \in \mathcal{V}, \text{ at any moment (capacity constraints)} \tag{3.1}$$

$$[q^{k,\theta} + L^{k,\theta}]_{\text{before}} - [q^{k,\theta}]_{\text{after}} = 0, \ \forall k \in \mathcal{V}, \theta \in \mathcal{A} \text{ (loading constraints)} \tag{3.2}$$

3.1.1 Multicompartment IRP

A specification of the Multiproduct IRP is the Multicompartment IRP (MCIRP). In the MCIRP each vehicle has devoted compartments of fixed capacity for each product. "There are many industrial fields in which the multicompartment vehicles are employed. The first example arises in the process of supplying fuels, where a lot of vehicles or ships with some tanks of various capacities are used to settle the problem. Another example is the transportation of food, where different degrees of refrigeration goods are stored in different compartments in one vehicle." [**43**], slightly adapted. In the same paper, the MCVRP is solved with a hybrid ACO algorithm.

3.1.2 Multi-compatibility and site-dependency

Paper [**44**] introduces *multi-compatibility* and *site-dependency* in the MMIRP applied to a food distribution service. The capacity of vehicles is divided into three zones with different temperatures (multi-compatibility of the products with compartments), and each vehicle can only visit a subset of destinations due to the specifications of unloading facilities (site-dependency). Then, the problem is solved with a three-stage *math-heuristic* based on the cluster-first and route-second method.

For a Multi-compatibility MCIRP linear program, the expressions (3.1) and (3.2) must be modified to consider the capacity of every compartment. Let each vehicle k have a set of compartments $\mathcal{Q}_k = \{\sigma \mid \sigma = 1, \dots, \Sigma_k\}$, each of capacity $Q_{k,\sigma}$. Content variables must be rewritten with the additional index σ; thus, $q^{k,\sigma,\theta}$ is the quantity of θ in compartment σ of vehicle k. Consequently, loading variables become $L^{k,\sigma,\theta}$. The expressions (3.1) and (3.2) are rewritten as:

$$\sum_{\theta \in \mathcal{A}} q^{k,\sigma,\theta} \leq Q_{k,\sigma}, \ \forall k \in \mathcal{V}, \sigma \in \mathcal{Q}_k, \text{ at any moment} \tag{3.3}$$

$$[q^{k,\sigma,\theta} + L^{k,\sigma,\theta}]_{\text{before}} - [q^{k,\sigma,\theta}]_{\text{after}} = 0, \ \forall k \in \mathcal{V}, \sigma \in \mathcal{Q}_k, \theta \in \mathcal{A} \tag{3.4}$$

These expressions can be simplified by assuming that each article is compatible with at most one compartment onboard of each vehicle. This compatible-articles set is defined as $\mathcal{A}_\sigma$, and (3.3), (3.4) become as follows:

$$\sum_{\theta \in \mathcal{A}_\sigma} (q^{k,\theta}) \leq Q_{k,\sigma}, \ \forall k \in \mathcal{V}, \sigma \in \mathcal{Q}_k, \text{ at any moment} \tag{3.5}$$

$$[q^{k,\theta} + L^{k,\theta}]_{\text{before}} - [q^{k,\theta}]_{\text{after}} = 0, \ \forall k \in \mathcal{V}, \sigma \in \mathcal{Q}_k, \theta \in \mathcal{A}_\sigma \tag{3.6}$$

Index σ disappeared from variables as the compartment is implicitly specified by the article $\theta \in \mathcal{A}_\sigma$.

3.2 IRP with Pickup and Delivery

The *pickup-and-delivery* aspect is the second main pillar of Mi-IRP-PD. The IRP with Pickup and Delivery is an important variant of the IRP with a great number of practical applications.
Paper [**45**] discusses the problem in detail and presents a discrete-time mathematical model solved with a branch-and-cut algorithm, while [**46**] concerns the IRP-PD with time windows and perishability constraints.

By reference to the model briefly introduced in the previous chapter, the pickup-and-delivery aspect can be integrated in a linear program by allowing negative values of $L^{k,\theta}$. As a consequence, vehicle content must be lower-bounded, and the constraint set is rewritten as:

$$\sum_{\theta \in \mathcal{A}} q^{k,\theta} \leq Q_k, \ \forall k \in \mathcal{V}, \text{ at any moment (upper capacity constraints)} \tag{3.7}$$

$$\sum_{\theta \in \mathcal{A}} q^{k,\theta} \geq 0, \ \forall k \in \mathcal{V}, \text{ at any moment (lower capacity constraints)} \tag{3.8}$$

$$[q^{k,\theta} + L^{k,\theta}]_{\text{before}} - [q^{k,\theta}]_{\text{after}} = 0, \ \forall k \in \mathcal{V}, \theta \in \mathcal{A} \text{ (loading constraints)} \tag{3.9}$$

3.3 IRP with Transshipment

In logistics, *transshipment* is a term coming from the maritime sector that "means the unloading of goods from one ship and its loading into another to complete a journey to a further destination, even when the cargo may have to remain ashore

some time before its onward journey. The term can also be applied more generally to other transport modes, such as freight transport by road or rail or air, or any combination of them." [**47**].
From an OR point of view, the use of a transshipment in IRPs can bring a considerable problem relaxation, as well as a general lead time reduction and an improvement of distribution performance. "The optimization problem must find the best configuration of vehicles, routes, pickups, deliveries and transshipments for each period, so as to minimize the total cost of supply chain operations." [**49**]. Depending on the problem, transshipment can rely on suppliers, retailers, suppliers-and-retailers, or even on devoted intermediate warehouses.

Two literature cases are shown at [**48, 49**]. The first presents an adaptive large neighbourhood search heuristic to solve the IRP with transshipment, while the second proposes an exact MILP formulation and tests it on instances of different size (with up to 15 stations and 12 scheduling periods).

3.4 The Mi-IRP-PD of this thesis

The Mi-IRP-PD discussed in this thesis presents some structural differences with respect to the IRPs usually found in literature. In fact, the *application case* for which this Mi-IRP-PD version was developed presents three characterizing aspects:

1. Each vehicle has a starting point, where it is not obliged to get back at the end of the period. Every vehicle can finish its route anywhere on problem ground.
2. Although the problem is about routing and inventory management, the minimization of distribution and storage costs are generally not a priority. As shown in part II of this thesis, the priority is given to the overall inventory-demand satisfaction.
3. Time-control shall be fine, of the order of some tens of seconds to some minutes. Therefore, the assumption for which any route is completed within a timeframe hardly matches with the here-presented Mi-IRP-PD.

How these three characteristic where handled is one of the main topics of part II, in which the aforementioned application case is also presented.

PART II

DEVELOPEMNT
OF SOLVING MODELS

Blank page

Chapter 4

Mi-IRP-PD formalization and application cases

This chapter discusses the origin and the evolution of the model proposed to solve the *Multi-item Inventory Routing Problem with Pickup and Delivery* (Mi-IRP-PD). The models and formulations described in this thesis were partially developed during my 6-months internship at the Laboratory of Analysis and Architecture of Systems (LAAS-CNRS) in Toulouse. The chapter starts by outlining its goals and ambitions, as well as all the elements that play a relevant role in it.

First, the proposed class structure allowed to adapt different problems to some common solving approaches developed during the internship. The main application case is the *Indoor logistic operation manager*. It consists in finding a viable scheduling of an indoor vehicle fleet to support the production of a manufacturing facility. The chapter deals with some variants of this problem, and an additional application case is introduced, too.

4.1 Case 1: Indoor logistic operation manager

This internship at LAAS-CNRS focused on the development of an operation scheduling system for an indoor vehicle fleet in charge of supporting the logistic needs of a manufacturing facility. The goal of the fleet is to collect and dispatch items (components, subassemblies, waste, tools) to manufacturing facility workstations so as to maintain acceptable stock levels at all times, all along the production line. Moreover, the scheduling system shall be able to quickly react to unforeseen production perturbations. If an unexpected event changes production schedule, the logistic support system shall also reschedule in response to the new needs. Consequently, system scheduling methods shall be able to give a new viable solution in a very short time, quantified at a few seconds. The *Indoor logistic operation manager* problem is also indicated with the acronym ILOM.

In its simplest form, this problem involves a set of stocks dislocated along the production facility, allowed to store a single article reference each, and associated to a provisional consumption profile. A main stock is also present on the production ground to generate and absorb articles that enter and quit it. However, not all the articles are involved in this in-out process; the main stock is a source of assembly components and a sink for defective pieces and waste. Semi-assembled parts can be transferred from a station to another but are not allowed to leave the production ground through the main stock. Forecasts about consumptions come from production planning or production simulation. The first method consists in scheduling production activities at workstations by collecting client orders and their deadlines. Once each workstation has its operations scheduled, material consumptions can be derived though the bill of materials. Alternatively, simulating production activities in a digital environment can also provide material consumptions profiles at each workstation.

The involved vehicle fleet in charge of transferring items from a stock to another can consist of heterogeneous vehicles, each with its own capacity and speed. The content of the vehicles shall be expressed in a generic *load unit* related to a univocal physical quantity, such as a fraction of the supported weight or of the available surface onboard. Stock contents and consumptions are expressed in load units, too; therefore, all item quantities shall be scaled from their real size to the unified one. E.g., let us consider a production line that assembles car wheels. Among other parts, vehicles are also called to transport bolts and tyres, that have a very different size. Making the hypotheses that:

- the line produces one wheel per minute,
- each wheel requires a tyre and six bolts,
- the space taken by a tyre can host 2000 bolts,
- the load unit (LU) is equal to the space taken by a tyre,

consumptions γ can be calculated as follows:

$$\gamma_{\text{tyres}} = 1\frac{\text{wheel}}{\text{min}} \cdot 1\frac{\text{tyre}}{\text{wheel}} \cdot 1\frac{\text{LU}}{\text{tyre}} = 1\frac{\text{LU}}{\text{min}} \tag{4.1}$$

$$\gamma_{\text{bolts}} = 1\frac{\text{wheel}}{\text{min}} \cdot 6\frac{\text{bolts}}{\text{wheel}} \cdot 1\frac{\text{LU}}{\text{tyre}} \cdot \left(2000\frac{\text{bolts}}{\text{tyre}}\right)^{-1} = 0.003\ \frac{\text{LU}}{\text{min}} \tag{4.2}$$

The same kind of calculation must be made for stored contents and lot sizes, respectively indicated by the letters C and λ in the following chapters.

Finally, item collection and delivery are made according to a full-lot policy. Every article reference has a lot size expressed in load units that corresponds to the minimum divisor of loadable/unloadable quantities.

4.1.1 Use cases and data of the logistic support system

Les us consider a set of buffer stocks $\mathcal{B}$, a main stock MS, and a set of vehicles $\mathcal{V}$. The following table shows system interaction scenarios in their most essential form.

Table 4.1. Use cases and scenarios of the ILOM.

Use case	Scenario	Action
1. $b \leftarrow$ MS	Items required at buffer b and absent on production ground.	Items shall be retrieved at main stock MS and brought to b.
2. $b \leftarrow b'$	Items required at buffer b and present at buffer b'.	Items shall be retrieved at b' and brought to b.
3. $b \leftarrow v$	Items required at buffer b and present on vehicle v.	Vehicle v shall deliver the items at b.
4. $b \rightarrow$ MS	Items to remove at buffer b and no more needed on production ground.	Items shall be collected at b and brought to main stock MS.
5. $b \rightarrow b'$	Items to remove at buffer b and needed at buffer b'.	Items shall be collected at b and brought to b'.
6. $b \rightarrow v$	Items to remove at buffer b and soon needed on production ground, or not *absorbable* by MS.	Items shall be collected at b by vehicle v and kept onboard.

Listing these use cases made it possible to identify the fundamental classes of the problem. However, to refine their formal definition it was then necessary to investigate their interdependencies and the relevant data belonging to each class. The version described in this paragraph was not the first to be developed during the internship. Indeed, the initial structure underwent several changes – usually simplifications – until reaching the current version. The final data structure features fours object classes: *Ground*, *Stocks*, *Vehicles,* and *Tasks*, hence called GSVT structure. The diagram presented at Figure 6.1 shows the aforementioned structure divided into three areas:

1. **Spatial data of production ground**. It contains a *static* class called *Ground* that stocks every useful topological information of the manufacturing facility.
2. **Production forecast data**, that lied outside the scope of the internship. Nevertheless, production is the most crucial data source of the problem. Its function is to forecast consumption (or production) profiles of materials at each workstation, and thus to allow the calculation of buffer needs overtime. Client orders are processed through MPS (Master Production Schedule) and

MRP (Material Requirements Planning) in order to obtain workstation-level activities and consumptions.

3. **Data actively handled by scheduling methods**. This area contains the classes of objects that have a direct role in solving procedures. In fact, the applied scheduling methods can read, process, and overwrite the objects belonging to these *dynamic* classes. After several simplifications and some research, the number of dynamic classes was reduced to three: *Stocks*, *Vehicles*, and *Tasks*.

The most important part of problem results is the list of *Tasks* generated by solving methods. The evolution of buffer and vehicle contents is also a relevant part of the solution. It is particularly useful for making a detailed analysis of result quality and balance, since it allows to easily identify local overloaded spots along the scheduling.

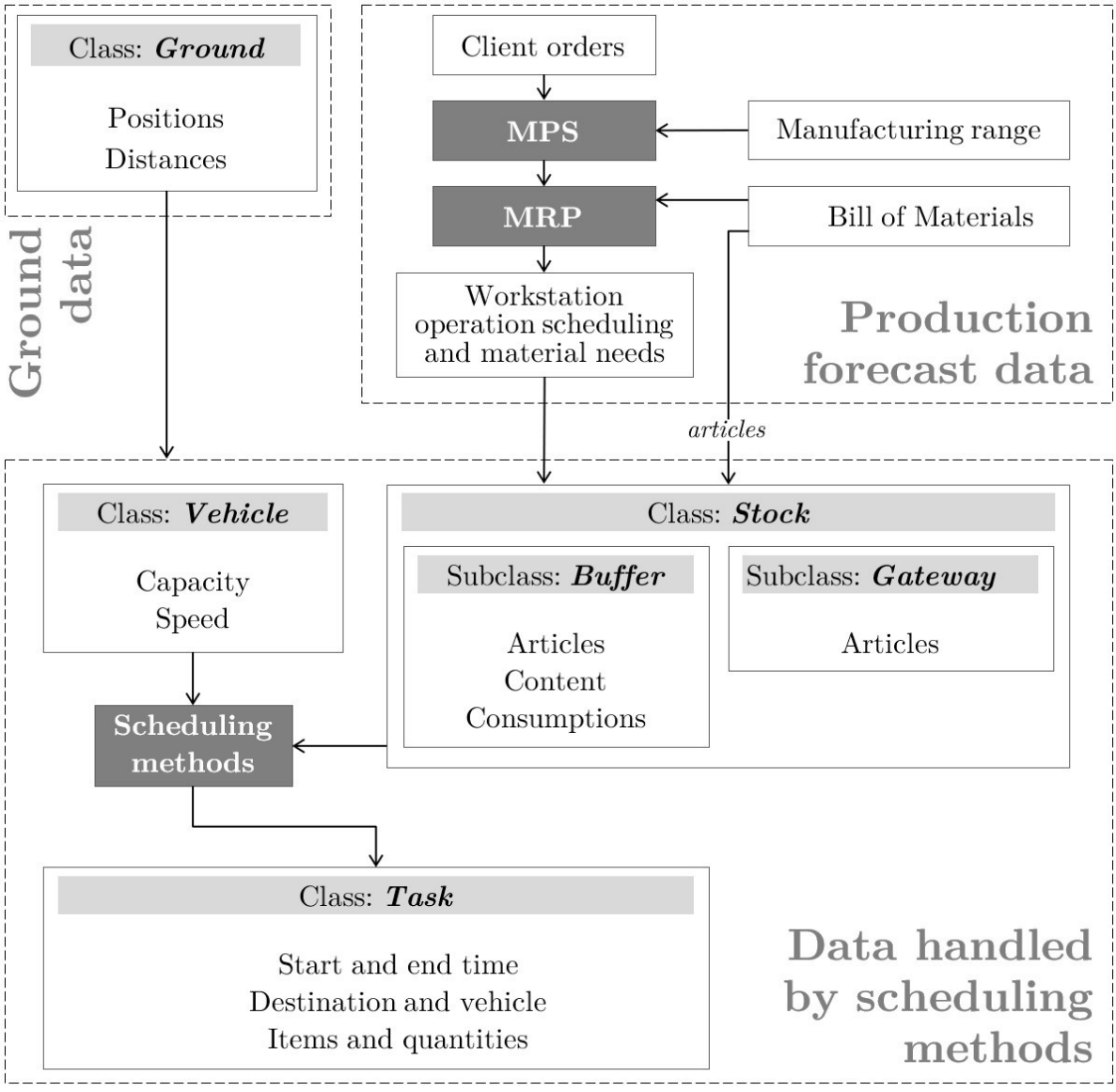


Figure 4.1. Class structure of the GSVT model.

4.1.2 Class definitions

- **Ground***:* set of coordinates and pairwise distances of all the remarkable points inside the manufacturing facility. Distances correspond to the actual directional paths between each pair of points, including tight corners and permanent ground obstacles.
- **Stock***:* a physical storage unit that is located somewhere within the production ground. Two subclasses derive from the *Stock*: *Buffers* and *Gateways*. The first one can contain a single article reference and has lower and upper content limits. Moreover, each *buffer* has a provisional consumption profile that allows to track content evolution overtime. Instead, *gateways* have the function of generating and/or absorbing a specific subset of items. In the simplest case, a single *gateway* called 'main stock' is the source of every supply article and the sink of waste material.
- **Vehicle***:* object that can load, contain, transfer, and unload items. Each *vehicle* has a capacity and a constant speed. Vehicles carry out *tasks*.
- **Task***:* activity with a start and an end time. It has a single destination and a single *vehicle* in charge of its completion. A *task* only ends when the *vehicle* that performs it is free of leaving *task*'s destination.

4.1.3 ILOM problem statement with a single main stock

Let us consider a manufacturing facility during a period T_P for which the production scheduling is known. The following sets of elements are deployed on the production ground.

- A set of article references $\mathcal{A} = \{a_1, a_1, \dots, a_\Theta\}$, indicated by index θ, such that each article has a unique lot size λ_θ.
- A set of buffers $\mathcal{B} = \{b_1, b_2, \dots, b_\mathrm{I}\}$, with index i, such that each buffer has the following parameters:

θ_i	the only allowed article reference,
$\overline{C_i}$	upper content limit,
$\underline{C_i}$	lower content limit,
$C_{i,t} = f(t)$	set of content values overtime,
$C_{i,0}$	initial content value,
$\gamma_{i,t} = f(t)$	set of consumptions, a value of $\gamma_{i,t}^{\theta}$ below zero represents a production,
d_{ij}	travel distance from i to any other fixed element j,
τ_i	forfeit time to complete onsite operations (load/unload).

- A main stock $\mathbb{i}$ with the following parameters:

$\mathcal{A}_\mathbb{i} \subseteq \mathcal{A}$	set of article references allowed in $\mathbb{i}$,

$d_{\mathbb{i}j}$	travel distance from $\mathbb{i}$ to any other fixed element j,
$\tau_{\mathbb{i}}$	forfeit time to complete onsite operations (load/unload).

- A set of vehicles $\mathcal{V} = \{v_1, v_2, \dots, v_K\}$, with index k, such that each vehicle has the following parameters:

Q_k	capacity,
s_k	speed,
$q_t^{k,\theta} = f(t)$	set of content values overtime for each article $\theta \in \mathcal{A}$,
$q_0^{k,\theta}$	set of initial content values for each article $\theta \in \mathcal{A}$,
$d_{\mathbb{0}j}^k$	distance between vehicle initial condition $\mathbb{0}$ and any fixed element j,
$t_{\mathbb{0}}^k$	unavailability time of the vehicle, that cannot move or do any task as long as $t < t_{\mathbb{0}}^k$.

The main objective is finding the set of feasible vehicle routes (encoded in a list of tasks) that minimizes the sum of buffer content values exceeding the limits $\underline{C_i}$, $\overline{C_i}$, $\forall i \in \mathcal{B}$. Travelled distance minimization could have an interest, too, for some specific cases.
This basic formulation can be generalized in a more complete one, in which sources and sinks of articles are not centralized in a single main stock. Furthermore, some *multi-item* auxiliary buffers can be added to the model as logistic supports for the other elements, therefore allowing a more flexible scheduling.

4.1.4 Generalized ILOM problem statement

The ILOM problem with multiple sources/sinks and multi-item-buffers is the generalization of what exposed in the previous chapter. Statements about articles, vehicles, and vehicles initial conditions do not change. Buffers and main stock statements are replaced by the following.

- A set of *multi-item* buffers $\mathcal{B} = \{b_1, b_2, \dots, b_I\}$, with index i, such that each buffer has the following parameters:

$\mathcal{A}_i \subseteq \mathcal{A}$	set of allowed article references,
$\overline{C}_i$	buffer capacity,
$\underline{C_i^\theta}$	set of lower content limits for each article $\theta \in \mathcal{A}_i$,
$C_{i,t}^\theta = f(t)$	set of content values overtime for each article $\theta \in \mathcal{A}_i$,
$C_{i,0}^\theta$	set of initial content values for each article $\theta \in \mathcal{A}_i$,
$\gamma_{i,t}^\theta = f(t)$	set of consumptions for each article $\theta \in \mathcal{A}_i$, a value of $\gamma_{i,t}^\theta$ below zero represents a production, and a $\gamma_{i,t}^\theta = 0$ represents a logistic support buffer,
d_{ij}	travel distance from i to any other fixed element j,
τ_i	forfeit time to complete onsite operations (load/unload),

w_i^θ	penalty weight associated to the importance of avoiding stockout of article θ at i.

- A set of source/sink points $\mathcal{P} = \{p_1, p_2, \dots, p_\Psi\}$, with index ψ, such that each s/s point has the following parameters:

$\mathcal{A}_\psi \subseteq \mathcal{A}$	set of allowed article references,
$d_{\psi j}$	travel distance between ψ and any other fixed element j,
τ_ψ	forfeit time to complete onsite operations (load/unload).

The objective of the problem remains the same: finding the set of feasible vehicle routes that minimizes the weighted sum of buffer content values exceeding the limits $\underline{C_i}$, $\overline{C_i}$, $\forall i \in \mathcal{B}$. Minimizing the overall travelled distance can be part of the objective, too.

4.2 Case 2: Supply-chain network manager

The *Multi-item Inventory Routing Problem with Pickup and Delivery* (Mi-IRP-PD) can be tailored to different application cases for which deterministic consumption previsions are known. Solving model is flexible; its constraints permit to obtain a realistic and customizable solution, and scalable time granularity is a crucial aspect of the model. In fact, time detail can easily be chosen as a function of the fundamental time unit of the problem (minutes, hours, days, etc.).

For the *Supply-chain network manager* (SNM) problem, let us consider a network of geographically distributed factories, warehouses and shops (as shown in figure 4.2), each producing, storing, or selling a certain number of different articles. The production output of factories is known, as well as shops provisional sales and return rates. The proposed Mi-IRP-PD model can help to manage warehouse and shop inventories, as well as to organize delivery, transfer and returning operations.

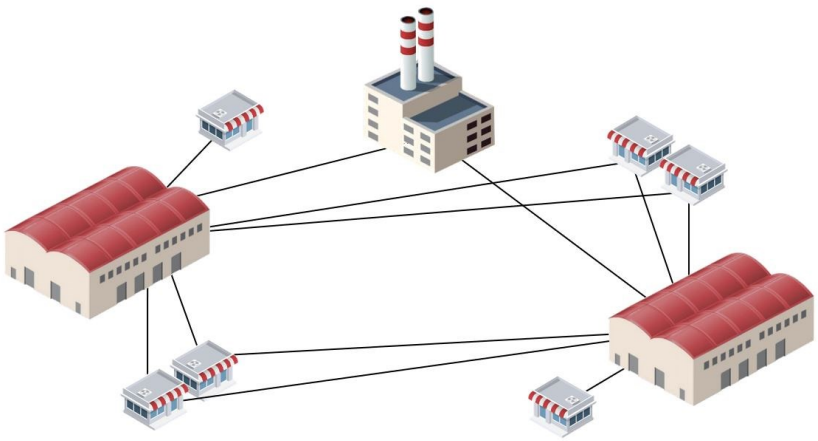


Figure 4.2. Graphic example of a supply chain network.

4.2.1 SNM problem statement

This problem is an adapted version of the ones presented in the *Indoor logistic operation manager* chapter. The model is composed of a set of factory inventories (*buffers* with negative consumptions), a set of shop inventories (*buffers* with positive consumptions), and a set of intermediate warehouses (*buffers* with no consumption). Moreover, the model can include provisional return fluxes originating from shops. In that case, shops can have negative consumptions for *return articles*, and a gateway is required to allow them to quit the area.

Let us consider a supply chain network during a period T_P for which production and sales are known. The following sets of elements are deployed in the network.

- A set of article references $\mathcal{A} = \{a_1, a_1, \dots, a_\Theta\}$, indicated by index θ, such that each article has a unique lot size λ_θ.

- A set of *facilities* (factories, shops, and warehouses) $\mathcal{B} = \{b_1, b_2, \dots, b_\mathrm{I}\}$, indicated by index i, such that each of them has the following parameters:

$\mathcal{A}_i \subseteq \mathcal{A}$	set of allowed article references,
$\overline{C_i}$	inventory capacity,
$\underline{C_i^\theta}$	set of lower content limits for each article $\theta \in \mathcal{A}_i$,
$C_{i,t}^\theta = f(t)$	set of content values overtime for each article $\theta \in \mathcal{A}_i$,
$C_{i,0}^\theta$	set of initial content values for each article $\theta \in \mathcal{A}_i$,
$\gamma_{i,t}^\theta = f(t)$	set of consumptions for each article $\theta \in \mathcal{A}_i$, $\gamma_{i,t}^\theta$ is the parameter that marks the difference between *factories*, *shops*, and *warehouses*. *Factories* have values $\gamma_{i,t}^\theta \leq 0$ as they produce articles; *shops* have values of $\gamma_{i,t}^\theta \geq 0$ as articles are sold, thus consumed; *warehouses* have values of $\gamma_{i,t}^\theta = 0$ as they are logistic support elements.
d_{ij}	travel distance from i to any other fixed element j,
τ_i	forfeit time to complete onsite operations (load/unload),
w_i^θ	penalty weight associated to the importance of avoiding stockout of article θ at i.

- A set of *return points* $\mathcal{P} = \{p_1, p_2, \dots, p_\Psi\}$, with index ψ, such that each point has the following parameters:

$\mathcal{A}_\psi \subseteq \mathcal{A}$	set of allowed article references,
$d_{\psi j}$	travel distance between ψ and any other fixed element j,
τ_ψ	forfeit time to complete onsite operations (load/unload).

- A set of *cargo vehicles* $\mathcal{V} = \{v_1, v_2, \dots, v_\mathrm{K}\}$, with index k, such that each vehicle has the following parameters:

Q_k	capacity,
s_k	speed,
$q_t^{k,\theta} = f(t)$	set of content values overtime for each article $\theta \in \mathcal{A}$,
$q_0^{k,\theta}$	set of initial content values for each article $\theta \in \mathcal{A}$,
$d_{\mathbb{0}j}^k$	distance between vehicle initial condition $\mathbb{0}$ and any fixed element j,
$t_{\mathbb{0}}^k$	unavailability time of the vehicle, that cannot move or do any task as long as $t < t_{\mathbb{0}}^k$.

The objective of the *SC network manager* problem is expressed as a two-terms minimization; first, minimize the inventory exceeding content limits at each facility, and second, minimize the overall travelled distance of the fleet. It can be proven that the second term has the heaviest impact on exact methods solving time. However, while finding the global optimum can take a very long time, it frequently happens that a very good solution is already found at an early solving stage. A technical trick could be the introduction of progressive distance brackets for evaluating the quality of a solution. The problem is hence relaxed, and optimality precision is known a priori. As shown in figure 4.3, this trick is equivalent to defining a staircase or piece-wise objective function. The precision of solution optimality is given by the width of each step. (E.g., if a solution has a distance included in the step (d_1, d_2) and is assigned the average value $(d_1 + d_2)/2$, the maximum error is then equal to $|d_1 - d_2|/2$.)

4.3 Considerations about applicability

Other than the ILOM (*indoor logistic operation manager*) and the SNM (*supply-chain network manager*), the generalized Mi-IRP-PD can be applied to any problem with the following characteristics:

- a set of scattered batch elements the content of which changes overtime,
- a set of items that can selectively enter and exit the global system, and can be picked-up from or delivered to batch elements,

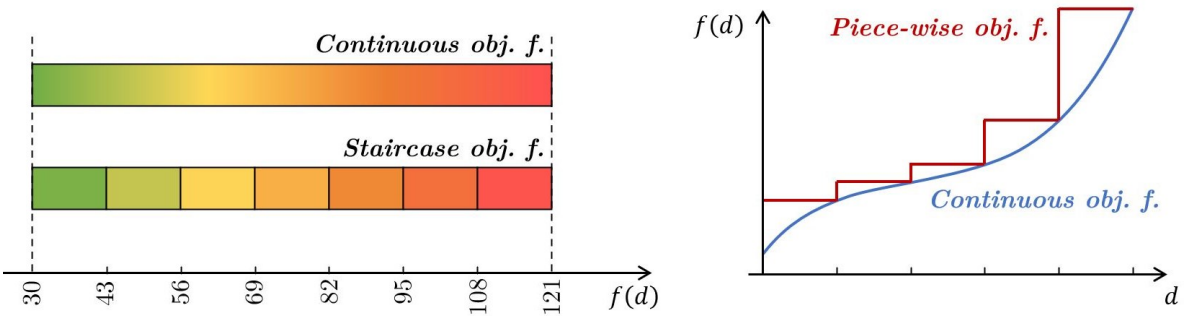


Figure 4.3. Graphic example of staircase and piece-wise objective functions.

- a set of capacitated transporters, each with a set of supported items.

However, the following conditions restrain the applicability field:

- the time spent for onsite operations at each batch is constant and does not depend on loaded/unloaded quantities,
- travel time does not change overtime. In other words, it is calculated as the product of distance and vehicle speed, and does not take account of any traffic condition or speed fluctuation.

Chapter 5

MILP formulations

This chapter presents the *Mixed Integer Linear Programming* (MILP) formulations developed to solve the problems of Chapter 4. First, a *continuous-time* formulation CT-MILP is introduced with its advantages, disadvantages, and applicability conditions. Next, some *discrete-time* formulations DT-MILP are presented. As explained in this chapter, DT formulations allow the exploration of a wider search space and give better quality results with respect to CT formulations. On the other hand, CT formulations are simpler to solve, and generally require a much shorter solving time.

5.1 Common standards and considerations

All the MILP formulations of this chapter have some common standards concerning the definition of sets, indexes, and units of measure. In general, all stocks are indexed with letters i and j, all vehicles with letter k, all article references with Greek letter θ, and time with letter t. However, each formulation paragraph explicitly reports its indexing system to avoid any misunderstanding.
Also, in the following pages, unless otherwise specified, the word *article* is used to indicate an *article reference* – a SKU – rather than the unitary item. This allows to reduce verbiage and display the models in a more readable way.
The content of *Buffers* and *Vehicles* is expressed in a generic measure unit called *Load Unit* (LU). In a real problem, a LU corresponds to a physical capacity unit, such as a weight in kilograms, a surface in m^2, or a volume in m^3. The number of Load Units on a vehicle or inside any inventory corresponds to the number of article units multiplied for the LU taken by one of them: $C[\text{LU}] = C[\text{units}] \cdot \text{LU taken by one article [LU/unit]}$. Consequently, content values, provisional consumptions, and stock content limits must undergo this procedure before entering MILP formulations, as shown at paragraph 4.1.

5.2 CT-FOQ-MILP formulation

As its name suggests, the *Continuous-Time Fixed-Order-Quantity MILP* (CT-FOQ-MILP) at this paragraph adopts the hypotheses of *fixed order quantity* pick-ups and deliveries. It means that a vehicle visiting a buffer can only load or unload a predefined quantity that depends on the buffer. This assumption permits to write the problem as a *continuous-time* model where pickup-delivery target times are pre-calculated parameters; target times come out from the integration of consumption profiles overtime.

5.2.1 Pre-calculation of pickups and deliveries

Let us consider a period of length T_P. The punctual logistic needs are the outcome of a simple integration algorithm. Algorithm input data includes buffer initial content $C_{i,0}$, buffer limits $\underline{C_i}$ and $\overline{C_i}$, and consumption rate function $\gamma_i(t)$: consumption/dt(t). First, consumptions are integrated in time and give content evolution profiles, i.e., provisional content level at each moment t'.

$$C_i(t') = C_{i,0} - \int_0^{t'} \gamma_i(t)dt \tag{5.1}$$

If $\gamma_i(t)$ is a piecewise function defined in D domains $\{\Gamma_0, \Gamma_1, \dots, \Gamma_{D-1}\}$, the expression can be converted in a form which is closer to numerical *discrete-time* models:

$$C_i(t')|_{t'\in\Gamma'} = C_{i,0} - \sum_{\Gamma=\Gamma_0}^{\Gamma<\Gamma'} \left(\int_{\Gamma} \gamma_i(t)dt \right) - \int_{\inf\Gamma'}^{t'} \gamma_i(t)dt, \quad \forall \Gamma' \in \{\Gamma_0, \Gamma_1, \dots, \Gamma_{D-1}\} \tag{5.2}$$

These equations shall be combined to the *fixed order quantity* approach to get to the *pickup-delivery target times*. In fact, each buffer has a fixed *order quantity* $\mathcal{OQ}_i$, that is often equal to the allowed content gap of the buffer $(\overline{C_i} - \underline{C_i})$. This quantity is not directly integrated in CT-FOQ-MILP models, as the pre-calculation algorithm sums it to buffer content $C_i(t)$ whenever the buffer reaches its limits. The times at which this happens are the *pickup-delivery target times*.

Algorithm 5.1 Pre-calculation of Pickup-Delivery target times

```
1  For each buffer i
2     Initialise pickup-delivery target times set T_i = ∅
3     t' = 0, C_i = C_{i,0}
4     While t' < T do
5        While C_i ∈ [C_i, C̄_i] do
6           Integrate consumptions C_i ← C_i − ∫_{t'}^{t'+dt'} γ_i(t)dt
7           Make time progress t' ← t' + dt'
8        Continue
```

9 $\mathcal{T}_i \leftarrow \mathcal{T}_i \cup \{t'\}$
10 $C_i \leftarrow C_i + \mathcal{OQ}_i$
11 **Continue**
12 **Next** i

The algorithm returns many sets of target times $\mathcal{T}_i$, one for each buffer. At this point, $\mathcal{T}_i$ and $\mathcal{OQ}_i$ values converge into a unique set of *pickup-delivery* elements $\mathcal{N} = \{n_1, n_2, \dots, n_N\}$, also called *needs*, that adopts the i-index formerly used to indicate buffers. For the rest of this paragraph, index i marks *pickup-delivery* objects that have a position, a demanded quantity, and a target time.

5.2.2 Parameters and variables

About formulation indexes:

- Buffer needs (pickup-deliveries) are indexed with letters i, j and l; vehicles with letter k; articles with Greek letter θ.
- The main stock is indicated with symbol $\mathbb{i}$, and the initial conditions of vehicles are indicated with symbol $\mathbb{o}$. The sets $\mathcal{N} \cup \{\mathbb{i}\}$, $\mathcal{N} \cup \{\mathbb{o}\}$ are also written $\mathcal{N}_\mathbb{i}$, $\mathcal{N}_\mathbb{o}$.
- For sake of readability, the elements of sets $\mathcal{N}$, $\mathcal{V}$, $\mathcal{A}$ are directly written as their indexes, as reported in the table below at the column 'Simplified'.

Table 5.1. Sets of CT-FOQ-MILP.

Set			**Set element notations**	
Name	Symbol	Index	Complete	Simplified
Needs	$\mathcal{N}$	i, j, l	$\{n_i\}_{i\in\{1,2,\dots,N\}}$	$\{1,2,\dots,N\}$
Vehicles	$\mathcal{V}$	k	$\{v_k\}_{k\in\{1,2,\dots,K\}}$	$\{1,2,\dots,K\}$
Articles	$\mathcal{A}$	θ	$\{a_\theta\}_{\theta\in\{1,2,\dots,\Theta\}}$	$\{1,2,\dots,\Theta\}$

Table 5.2. Parameters of CT-FOQ-MILP.

	Description	**Index domains**
$\mathcal{OQ}_i$	order quantity of need i	$i \in \mathcal{N}$
$\dot{t}_i$	target time of need i	$i \in \mathcal{N}$
d_{ij}	Distance to go from i-need location to j-need location	$i, j \in \mathcal{N}_\mathbb{i}, i \neq j$
τ_i	time required to perform onsite operations for i	$i \in \mathcal{N}_\mathbb{i}$
Q_k	capacity of vehicle k	$k \in \mathcal{V}$

s_k	speed of vehicle k	$k \in \mathcal{V}$
$q_0^{k,\theta}$	initial content of vehicle k	$k \in \mathcal{V}$, $\theta \in \mathcal{A}$
$t_{\mathbb{0}}^k$	time from which vehicle k is available	$k \in \mathcal{V}$
$d_{\mathbb{0}i}^k$	distance from k initial position to i-need location	$i \in \mathcal{N}_{\mathbb{i}}$, $k \in \mathcal{V}$
M	sufficiently big number $= \max_{k \in \mathcal{V}}\{T_P, Q_k\}$	

Model variables are specific to each formulation. In the following, they are written in bold to distinguish from constant parameters.

Table 5.3. Variables of CT-FOQ-MILP.

	Description	**Index domains**
$\boldsymbol{x_{ij}^k}$	Boolean var := $\begin{cases} 1 \text{ if vehicle } k \text{ goes from } i \text{ to } j, \\ \quad \text{without passing from main stock } \mathbb{i} \\ 0 \text{ otherwise} \end{cases}$	$i \in \mathcal{N}_{\mathbb{0}}$, $j \in \mathcal{N}$, $i \neq j$, $k \in \mathcal{V}$
$\boldsymbol{y_{ij}^k}$	Boolean var := $\begin{cases} 1 \text{ if vehicle } k \text{ goes from } i \text{ to } j, \\ \quad \text{passing from main stock } \mathbb{i} \\ 0 \text{ otherwise} \end{cases}$	$i \in \mathcal{N}_{\mathbb{0}}$, $j \in \mathcal{N}$, $i \neq j$, $k \in \mathcal{V}$
$\boldsymbol{t_i}$	Time at which the need i is satisfied	$i \in \mathcal{N}$
$\boldsymbol{q_i^{k,\theta}}$	content of vehicle k in terms of θ after satisfying i	$i \in \mathcal{N}, k \in \mathcal{V}$, $\theta \in \mathcal{A}$
Auxiliary variable		
$\boldsymbol{\delta_i}$	$\geq \lvert \dot{t}_i - \boldsymbol{t_i} \rvert$	$i \in \mathcal{N}$

5.2.3 CT-FOQ-MILP model

The CT-FOQ-MILP model is formulated as follows:

$$\textit{Minimize } f = \sum_{i \in \mathcal{N}} \boldsymbol{\delta_i} + \alpha_d \left(\sum_{i \in \mathcal{N}_{\mathbb{0}}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{V}} d_{ij} \boldsymbol{x_{ij}^k} + \sum_{i \in \mathcal{N}_{\mathbb{0}}} \sum_{j \in \mathcal{N}} \sum_{k \in \mathcal{V}} \left(d_{i\mathbb{i}} + d_{\mathbb{i}j}\right) \boldsymbol{y_{ij}^k} \right) \tag{5.3}$$

Subject to:

$$\boldsymbol{q_i^{k,\theta}} \geq 0 \qquad \forall i \in \mathcal{N},\ k \in \mathcal{V},\ \theta \in \mathcal{A} \tag{5.4}$$

$$\sum_{\theta \in \mathcal{A}} \boldsymbol{q_i^{k,\theta}} \leq Q_k \qquad \forall i \in \mathcal{N},\ k \in \mathcal{V} \tag{5.5}$$

$$\sum_{j \in \mathcal{N}} \left(\boldsymbol{x}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}} + \boldsymbol{y}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}} \right) \leq 1 \qquad \forall k \in \mathcal{V} \tag{5.6}$$

$$\sum_{i \in \mathcal{N}_{\mathbb{o}}} \left(\boldsymbol{x}^{\boldsymbol{k}}_{\boldsymbol{ij}} + \boldsymbol{y}^{\boldsymbol{k}}_{\boldsymbol{ij}} \right) \geq \sum_{l \in \mathcal{N}} \left(\boldsymbol{x}^{\boldsymbol{k}}_{\boldsymbol{jl}} + \boldsymbol{y}^{\boldsymbol{k}}_{\boldsymbol{jl}} \right) \qquad \forall j \in \mathcal{N},\ k \in \mathcal{V} \tag{5.7}$$

$$\sum_{i \in \mathcal{N}_{\mathbb{o}}} \sum_{k \in \mathcal{V}} \left(\boldsymbol{x}^{\boldsymbol{k}}_{\boldsymbol{ij}} + \boldsymbol{y}^{\boldsymbol{k}}_{\boldsymbol{ij}} \right) = 1 \qquad \forall j \in \mathcal{N},\ i \neq j \tag{5.8}$$

$$\boldsymbol{t_i} + d_{ij}/s_k + \tau_j \leq \boldsymbol{t_j} + M\left(1 - \boldsymbol{x}^{\boldsymbol{k}}_{\boldsymbol{ij}}\right) \qquad \forall\, i,j \in \mathcal{N},\ k \in \mathcal{V} \tag{5.9}$$

$$t^k_{\mathbb{o}} + d^k_{\mathbb{o}j}/s_k + \tau_j \leq \boldsymbol{t_j} + M\left(1 - \boldsymbol{x}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}}\right) \qquad \forall j \in \mathcal{N},\ k \in \mathcal{V} \tag{5.9'}$$

$$\boldsymbol{t_i} + d_{i\mathbb{i}}/s_k + \tau_{\mathbb{i}} + d_{\mathbb{i}j}/s_k + \tau_j \leq \boldsymbol{t_j} + M\left(1 - \boldsymbol{y}^{\boldsymbol{k}}_{\boldsymbol{ij}}\right) \qquad \forall\, i,j \in \mathcal{N},\ k \in \mathcal{V} \tag{5.10}$$

$$t^k_{\mathbb{o}} + d^k_{\mathbb{o}\mathbb{i}}/s_k + \tau_{\mathbb{i}} + d_{\mathbb{i}j}/s_k + \tau_j \leq \boldsymbol{t_j} + M\left(1 - \boldsymbol{y}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}}\right) \qquad \forall i \in \mathcal{N},\ k \in \mathcal{V} \tag{5.10'}$$

$$\boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{i}} + \mathcal{O}\mathcal{Q}_j \leq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} + M\left(1 - \boldsymbol{x}^{\boldsymbol{k}}_{\boldsymbol{ij}}\right) \qquad \forall\, i,j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \tag{5.11}$$

$$q^{k,\theta}_{\mathbb{o}} + \mathcal{O}\mathcal{Q}_j \leq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} + M\left(1 - \boldsymbol{x}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}}\right) \qquad \forall\, j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \tag{5.11'}$$

$$\boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{i}} + \mathcal{O}\mathcal{Q}_j \geq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} - M\left(1 - \boldsymbol{x}^{\boldsymbol{k}}_{\boldsymbol{ij}}\right) \qquad \forall i,j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \tag{5.12}$$

$$q^{k,\theta}_{\mathbb{o}} + \mathcal{O}\mathcal{Q}_j \geq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} - M\left(1 - \boldsymbol{x}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}}\right) \qquad \forall j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \tag{5.12'}$$

$$\boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{i}} + \mathcal{O}\mathcal{Q}_j \leq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} + M\left(1 - \boldsymbol{y}^{\boldsymbol{k}}_{\boldsymbol{ij}}\right) \qquad \forall\, i,j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \setminus \mathcal{A}_{\mathbb{i}} \tag{5.13}$$

$$q^{k,\theta}_{\mathbb{o}} + \mathcal{O}\mathcal{Q}_j \leq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} + M\left(1 - \boldsymbol{y}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}}\right) \qquad \forall\, j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \setminus \mathcal{A}_{\mathbb{i}} \tag{5.13'}$$

$$\boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{i}} + \mathcal{O}\mathcal{Q}_j \geq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} - M\left(1 - \boldsymbol{y}^{\boldsymbol{k}}_{\boldsymbol{ij}}\right) \qquad \forall\, i,j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \setminus \mathcal{A}_{\mathbb{i}} \tag{5.14}$$

$$q^{k,\theta}_{\mathbb{o}} + \mathcal{O}\mathcal{Q}_j \geq \boldsymbol{q}^{\boldsymbol{k},\boldsymbol{\theta}}_{\boldsymbol{j}} - M\left(1 - \boldsymbol{y}^{\boldsymbol{k}}_{\mathbb{o}\boldsymbol{j}}\right) \qquad \forall\, j \in \mathcal{N}, k \in \mathcal{V}, \theta \in \mathcal{A}_i \setminus \mathcal{A}_{\mathbb{i}} \tag{5.14'}$$

$$\boldsymbol{\delta_i} \geq \dot{t}_i - \boldsymbol{t_i} \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \tag{5.15}$$

$$\boldsymbol{\delta_i} \geq \boldsymbol{t_i} - \dot{t}_i \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \tag{5.16}$$

$$\boldsymbol{x}^{\boldsymbol{k}}_{\boldsymbol{ij}} \in \{0,1\} \qquad \forall i \in \mathcal{N}_{\mathbb{o}},\ j \in \mathcal{N},\ k \in \mathcal{V} \tag{5.17}$$

$$\boldsymbol{y}^{\boldsymbol{k}}_{\boldsymbol{ij}} \in \{0,1\} \qquad \forall i \in \mathcal{N}_{\mathbb{o}},\ j \in \mathcal{N},\ k \in \mathcal{V} \tag{5.18}$$

The objective function contains two terms. The first aims at minimizing the overall gap between pickup-delivery target times and real pickup-delivery times. The second is multiplied for a weight $\alpha_d \geq 0$ and aims at minimizing the global travelled distance of all vehicles.

The first set of constraints (5.4) imposes to every vehicle content to be greater or equal to zero, as negative contents have no physical meaning. On the other hand, (5.5) ensures that vehicle capacities are not exceeded. Expressions (5.6), (5.7) and (5.8) bind connection variables $\boldsymbol{x_{ij}^k}$ and $\boldsymbol{y_{ij}^k}$. (5.6) imposes that if vehicles start their route, they start it once and in a single way (not-visiting the main stock - $\boldsymbol{x_{\mathbb{0}j}^k} = 1$ *XOR* visiting it - $\boldsymbol{y_{\mathbb{0}j}^k} = 1$). Expression (5.7) imposes a series of chain dependencies between connection variables. In fact, no *need* location can be left it is not reached first. Finally, (5.8) states that each *need* location must be visited – thus each *need* must be satisfied – once and only once. The expression (5.6) is still necessary as (5.8) does not forbid the departure of a same vehicle from its initial condition $\mathbb{0}$ to two different *need* locations; more specifically, (5.8) does not forbid the combinations of the kind $\left(\boldsymbol{x_{\mathbb{0}1}^1}, \boldsymbol{x_{\mathbb{0}2}^1}\right) = (1,1)$.
Expressions from (5.9) to (5.10') are the time constraints of the formulation. (5.9) states that if vehicle k crosses the arc ij, then the time at which k can leave j must be greater or equal to the time at which k left i, plus the travel time from i to j, plus the duration of onsite operations at j. (5.10) imposes the same time relationships in case the main stock is visited between i and j. These two sets of constraints are written like in (5.9') and (5.10') if $i = \mathbb{0}$.
Expressions from (5.11) to (5.14') rule the evolution of vehicle content. (5.11) and (5.12) state that if vehicle k crosses the arc ij, then its content increases exactly of $\mathcal{OQ}_j$; otherwise, content values $\boldsymbol{q_i^{k,\theta}}$ and $\boldsymbol{q_j^{k,\theta}}$ are unbound. (5.13) and (5.14) impose the same content relationships in case the main stock is visited between i and j, and if the involved article cannot be exchanged with the main stock ($\theta \in \mathcal{A}_i \setminus \mathcal{A}_{\mathbb{i}}$). In fact, if $\boldsymbol{y_{ij}^k} = 1$ and θ can be exchanged with the main stock, $\boldsymbol{q_i^{k,\theta}}$ and $\boldsymbol{q_j^{k,\theta}}$ are unbound. These four sets of constraints are written like in (5.11'), (5.12'), (5.13') and (5.14') if $i = \mathbb{0}$.
Constraints (5.15) and (5.16) rule the auxiliary variables $\boldsymbol{\delta_i}$ that must satisfy the inequation $\boldsymbol{\delta_i} \geq |\dot{t}_i - \boldsymbol{t_i}|$. Finally, (5.17) and (5.18) express the Boolean domain of variables $\boldsymbol{x_{ij}^k}$ and $\boldsymbol{y_{ij}^k}$.

5.3 DT-1ms-MILP formulation

The *Discrete-Time 1-main-stock MILP* (DT-1ms-MILP) formulation can be used to solve to the Mi-IRP-PD with a unique main stock that generates supply material and absorbs waste. This formulation is based on the classes of GSVT structure (*Ground*, *Stock*, *Vehicles*, *Tasks*) adapted to the *indoor logistic operations with a unique main-stock* problem.

This formulation is *time-discrete*; the scheduling period of length T_P is converted into a discrete set of equal timeframes $\hat{T} = \{\hat{t}_0, \hat{t}_1, \hat{t}_2, \dots, \hat{t}_T\}$. This means that the exact state of the problem is only known in a finite set of instants. Conventionally, the t-

index found in the formulation indicates the exact state of the system at instant t, as well as its approximate state during the interval $[t, t{+}1)$. Consequently, problem parameters and variables involving time are expressed in terms of $(*)$ per timeframe, where $(*)$ is a generic physical quantity, and timeframe duration is defined as $\Delta\hat{t} = (\hat{t}_{t+1} - \hat{t}_t)$. $\Delta\hat{t}$ can be chosen freely; nevertheless, a too (and sometimes unnecessarily) small value of $\Delta\hat{t}$ adds a greater number of constraints and complexifies the resolution procedure.
About formulation indexes:

- Buffers are indexed with letters i and j; vehicles with letter k; articles with Greek letter θ; time with letter t.
- The main stock is indicated with symbol $\mathbb{i}$, and vehicles initial conditions are indicated with symbol $\mathbb{o}$. The sets $\mathcal{B} \cup \{\mathbb{i}\}$, $\mathcal{B} \cup \{\mathbb{o}\}$, $\mathcal{B} \cup \{\mathbb{o}, \mathbb{i}\}$ are also written $\mathcal{B}_{\mathbb{i}}$, $\mathcal{B}_{\mathbb{o}}$, $\mathcal{B}_{\mathbb{o}\mathbb{i}}$.
- For sake of readability, the elements of $\mathcal{B}$, $\mathcal{V}$, $\mathcal{A}$, $\hat{T}$ are directly written as their indexes, as reported in the table below at the column 'Simplified'.

Table 5.4. Sets of DT-1ms-MILP.

Set			**Set element notations**	
Name	Symbol	Index	Complete	Simplified
Buffers	$\mathcal{B}$	i, j	$\{b_i\}_{i\in\{1,2,\dots,B\}}$	$\{1,2,\dots,B\}$
Vehicles	$\mathcal{V}$	k	$\{v_k\}_{k\in\{1,2,\dots,K\}}$	$\{1,2,\dots,K\}$
Articles	$\mathcal{A}$	θ	$\{a_\theta\}_{\theta\in\{1,2,\dots,\Theta\}}$	$\{1,2,\dots,\Theta\}$
Timeframes	$\hat{T}$	t	$\{\hat{t}_t\}_{t\in\{0,1,2,\dots,T\}}$	$\{0,1,2,\dots,T\}$

5.3.1 Parameters and variables

The nomenclature of model parameters reflects what stated for the *1-main-stock indoor logistic operations* problem.

Table 5.5. Parameters of DT-1ms-MILP.

	Description	**Index domains**
λ_θ	load units in a lot of θ	$\theta \in \mathcal{A}$
$\underline{C_i}$	lower content limit of buffer i	$i \in \mathcal{B}$
$\overline{C_i}$	upper content limit of buffer i	$i \in \mathcal{B}$
$C_{i,0}$	initial content of buffer i	$i \in \mathcal{B}$
$\gamma_{i,t}$	consumption of buffer i during t	$i \in \mathcal{B}$, $t \in \hat{T} \setminus \{T\}$

d_{ij}	travelling distance from i to j	$i,j \in \mathcal{B}_{\mathbb{I}}, i \neq j$
τ_i	time required to perform onsite operations at i	$i \in \mathcal{B}_{\mathbb{I}}$
Q_k	capacity of vehicle k, in load units	$k \in \mathcal{V}$
s_k	speed of vehicle k	$k \in \mathcal{V}$
$q_0^{k,\theta}$	initial content of vehicle k	$k \in \mathcal{V}$, $\theta \in \mathcal{A}$
$t_{\mathbb{O}}^k$	time from which vehicle k is available	$k \in \mathcal{V}$
$d_{\mathbb{O}i}^k$	distance from k initial position to i	$i \in \mathcal{B}_{\mathbb{I}}$, $k \in \mathcal{V}$
$\Pi_{ij,t}^k$	rounded-up integer job finish-time $= \lceil t + d_{ij}/s_k + \tau_j \rceil$	$i,j \in \mathcal{B}_{\mathbb{I}}, i \neq j$, $k \in \mathcal{V}$, $t \in \hat{T} \setminus \{T\}$
M	sufficiently big number $= \min\left\{ \max_{k \in \mathcal{V}, \theta \in \mathcal{A}} \{Q_k/\lambda_\theta\}, \max_{i \in \mathcal{B}_{\mathbb{I}}} \{\overline{C}_i/\lambda_{\theta_i}\} \right\}$	

Variables are written in bold. The *do* and *wait* states mentioned for $\boldsymbol{x_{i,t}^k}$ and $\boldsymbol{y_{i,t}^k}$ variables are better explained in the following paragraph.

Table 5.6. Variables of DT-1ms-MILP.

	Description	**Index domains**
$\boldsymbol{x_{i,t}^k}$	Boolean var $:=$ $\begin{cases} 1 \text{ if vehicle } k \text{ is in } do \text{ state during } t \\ 0 \text{ otherwise} \end{cases}$	$i \in \mathcal{B}_{\mathbb{I}}$, $k \in \mathcal{V}$, $t \in \hat{T}$
$\boldsymbol{y_{i,t}^k}$	Boolean var $:=$ $\begin{cases} 1 \text{ if vehicle } k \text{ is in } wait \text{ state during } t \\ 0 \text{ otherwise} \end{cases}$	$i \in \mathcal{B}_{\mathbb{I}\mathbb{O}}$, $k \in \mathcal{V}$, $t \in \hat{T}$
$\boldsymbol{C_{i,t}}$	content of buffer i at instant t	$i \in \mathcal{B}$, $t \in \hat{T} \setminus \{0\}$
$\boldsymbol{q_t^{k,\theta}}$	content of vehicle k in terms of θ at instant t	$k \in \mathcal{V}$, $\theta \in \mathcal{A}$, $t \in \hat{T} \setminus \{0\}$
$\boldsymbol{L_{i,t}^{k,\theta}}$	number of lots of θ, loaded at i by k at instant t	$i \in \mathcal{B}_{\mathbb{I}}$, $k \in \mathcal{V}$, $\theta \in \mathcal{A}_i$, $t \in \hat{T} \setminus \{0\}$
Auxiliary variable		
$\boldsymbol{\epsilon_{i,t}}$	$\geq \max\{0, \boldsymbol{C_{i,t}} - \overline{C}_i, \underline{C}_i - \boldsymbol{C_{i,t}}\}$	$i \in \mathcal{B}$, $t \in \hat{T} \setminus \{0\}$

5.3.2 Standards applied to states and transitions

This paragraph defines the relationships between variables as a function of time, first recalling that all variables with the same t must represent the exact state of the system in that moment. Once the problem is solved, the values assumed by state-variables ($x_{i,t}^k$ and $y_{i,t}^k$) encode vehicle routes and allow state transitions that are crucial to make other variables change.
States and transitions follow some guidelines:

- A vehicle k can be in two different states:

Table 5.7. State variables of DT formulations.

State	Variables	Description
do	$(x_{i,t}^k, y_{i,t}^k) = (1,0)$	Vehicle k is going to or doing onsite operations at buffer i during timeframe t.
wait	$(x_{i,t}^k, y_{i,t}^k) = (0,1)$	Vehicle k is waiting at buffer i during timeframe t.

These two states cannot coexist; each vehicle must be in *one and only one* state at a time, for *one and only one* value of i.

- Three state transitions are defined, depending on state-variable changes from t to $t+1$. In the tables below, only the variables involved in transitions have a non-blank value case.

1. *do-i* → *do-j*:
   At instant $t+1$, vehicle k ends the job it was doing for buffer i and starts a new job for buffer j. In this case, i and j must have different values.

k-ij variables at t				*k-ij* variables at $t+1$			
$x_{i,t}^k$	$y_{i,t}^k$	$x_{j,t}^k$	$y_{j,t}^k$	$x_{i,t+1}^k$	$y_{i,t+1}^k$	$x_{j,t+1}^k$	$y_{j,t+1}^k$
1		0		0		1	

2. *do-i* → *wait-i*:
   At instant $t+1$, vehicle k ends the job it was doing for buffer i and starts waiting at the same buffer.

k-ij variables at t				*k-ij* variables at $t+1$			
$x_{i,t}^k$	$y_{i,t}^k$	$x_{j,t}^k$	$y_{j,t}^k$	$x_{i,t+1}^k$	$y_{i,t+1}^k$	$x_{j,t+1}^k$	$y_{j,t+1}^k$
1	0			0	1		

3. *wait-i* → *do-j*:
   At instant $t+1$, vehicle k stops waiting at buffer i and starts a job for buffer j. In this case, i is allowed to be equal to j.

k-ij variables at t				k-ij variables at t+1			
$x^k_{i,t}$	$y^k_{i,t}$	$x^k_{j,t}$	$y^k_{j,t}$	$x^k_{i,t+1}$	$y^k_{i,t+1}$	$x^k_{j,t+1}$	$y^k_{j,t+1}$
	1	0			0	1	

wait-i $\rightarrow$ *wait-j* transitions do not exist, since a vehicle cannot move from a buffer i to a buffer j (with $i \neq j$) without passing through a *do* state.

- Consumption values $\gamma_{i,t}$ refer to the number of load units consumed during timeframe t. The whole consumption is conventionally counted once the t-frame is over, i.e., at instant t+1. Thus, $\boldsymbol{C_{i,t}}$ is not deprived yet of the quantity $\gamma_{i,t}$ with respect to $\boldsymbol{C_{i,t-1}}$.
  Numerical example:

t	0	1	2	3	4	5	6	7	8
$\gamma_{i,t}$	0	3	3	3	0	0	2	2	0
$C_{i,t}$	20	20	17	14	11	11	11	9	7

  N.B.: since $\boldsymbol{C_{i,t}}$ represents buffer content during the whole t time frame, positive consumptions values could be right shifted of a $-\Delta\hat{t}$ ($\gamma_{i,t} \mapsto \gamma_{i,t-1}$) to ensure that real buffer content never runs below zero.
  E.g., in the small table below, buffer content could go negative for $t \geq 2$ in real world operations. In fact, passing to a continuous domain and considering a constant consumption rate between $t2$ and $t3$, buffer content at $t2.\overline{9}$ is equal to -0.9. Nevertheless, the solution is formally acceptable.

t	0	1	2	3	4
$\gamma_{i,t}$	0	1	1	1	…
$C_{i,t}$	1.1	1.1	0.1	0.1	0.1
$\sum L_{i,t}$	0	0	1	1	1

- When a *do-i* $\rightarrow *$ transition occurs, the involved vehicles and buffer can exchange their content. Content value changes and article loading / unloading all happen at instant t+1, when the transition occurs.

5.3.3 DT-1ms-MILP model

The DT-1ms-MILP model is formulated as follows:

$$\textit{Minimize } f = \sum_{i \in \mathcal{B}} \sum_{t \in \hat{T} \setminus \{0\}} \boldsymbol{\epsilon_{i,t}} \tag{5.19}$$

Subject to:

$$\boldsymbol{C_{i,t}} \geq 0 \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \qquad (5.20)$$

$$\boldsymbol{C_{i,t+1}} = \boldsymbol{C_{i,t}} - \gamma_{i,t} - \lambda_{\theta_i} \cdot \sum_{k \in \mathcal{V}} \boldsymbol{L_{i,t+1}^{k,\theta_i}} \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0, T\} \qquad (5.21)$$

$$\boldsymbol{C_{i,1}} = C_{i,0} - \gamma_{i,0} - \lambda_{\theta_i} \cdot \sum_{k \in \mathcal{V}} \boldsymbol{L_{i,1}^{k,\theta_i}} \qquad \forall i \in \mathcal{B} \qquad (5.21')$$

$$\boldsymbol{q_t^{k,\theta}} \geq 0 \qquad \forall k \in \mathcal{V}, \theta \in \mathcal{A}, t \in \hat{T} \setminus \{0\} \qquad (5.22)$$

$$\sum_{\theta \in \mathcal{A}} \boldsymbol{q_t^{k,\theta}} \leq Q_k \qquad \forall k \in \mathcal{V}, t \in \hat{T} \setminus \{0\} \qquad (5.23)$$

$$\boldsymbol{q_{t+1}^{k,\theta}} = \boldsymbol{q_t^{k,\theta}} + \sum_{i \in \mathcal{B} \cap \{i' | \theta \in \mathcal{A}_{i'}\}} \lambda_\theta \boldsymbol{L_{i,t+1}^{k,\theta}} \qquad \forall k \in \mathcal{V}, \theta \in \mathcal{A}, t \in \hat{T} \setminus \{0, T\} \qquad (5.24)$$

$$\boldsymbol{q_1^{k,\theta}} = q_0^{k,\theta} + \sum_{i \in \mathcal{B} \cap \{i' | \theta \in \mathcal{A}_{i'}\}} \lambda_\theta \boldsymbol{L_{i,1}^{k,\theta}} \qquad \forall k \in \mathcal{V}, \theta \in \mathcal{A} \qquad (5.24')$$

$$M \boldsymbol{x_{i,t}^k} + \boldsymbol{L_{i,t+1}^{k,\theta}} \geq 0 \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{T\} \qquad (5.25)$$

$$-M \boldsymbol{x_{i,t}^k} + \boldsymbol{L_{i,t+1}^{k,\theta}} \leq 0 \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{T\} \qquad (5.26)$$

$$M\left(1 - \boldsymbol{x_{i,t+1}^k}\right) + \boldsymbol{L_{i,t+1}^{k,\theta}} \geq 0 \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{T\} \qquad (5.27)$$

$$-M\left(1 - \boldsymbol{x_{i,t+1}^k}\right) + \boldsymbol{L_{i,t+1}^{k,\theta}} \leq 0 \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{T\} \qquad (5.28)$$

$$\sum_{i \in \mathcal{B} \cup \{\mathbb{i}\}} \boldsymbol{x_{i,t}^k} + \sum_{i \in \mathcal{B} \cup \{\mathbb{o}, \mathbb{i}\}} \boldsymbol{y_{i,t}^k} = 1 \qquad \forall k \in \mathcal{V}, t \in \hat{T} \qquad (5.29)$$

$$1 - \left(\boldsymbol{x_{i,t}^k} + \boldsymbol{y_{i,t}^k}\right) - \boldsymbol{x_{j,t+1}^k} + \boldsymbol{x_{j,\pi}^k} \geq 0 \qquad \begin{array}{r} \forall i, j \in \mathcal{B}_{\mathbb{i}}, i \neq j, k \in \mathcal{V} \\ \forall t \in \hat{T} \setminus \{T\}, \pi \in \left[t + 2, \Pi_{ij,t}^k\right] \end{array} \qquad (5.30)$$

$$1 - \boldsymbol{y_{\mathbb{o},t}^k} - \boldsymbol{x_{j,t+1}^k} + \boldsymbol{x_{j,\pi}^k} \geq 0 \qquad \begin{array}{r} \forall j \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, \\ \forall t \in \hat{T} \setminus \{T\}, \pi \in \left[t + 2, \Pi_{ij,t}^k\right] \end{array} \qquad (5.30')$$

$$\boldsymbol{x_{i,t}^k} + \boldsymbol{y_{i,t}^k} \geq \boldsymbol{y_{i,t+1}^k} \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, t \in \hat{T} \setminus \{T\} \qquad (5.31)$$

$$\boldsymbol{y_{\mathbb{o},t}^k} \geq \boldsymbol{y_{\mathbb{o},t+1}^k} \qquad \forall k \in \mathcal{V}, \hat{t} \in \hat{T} \setminus \{T\} \qquad (5.31')$$

$$1 - \boldsymbol{y_{i,t}^k} \geq \boldsymbol{x_{i,t+1}^k} \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, t \in \hat{T} \setminus \{T\} \qquad (5.32)$$

$$\boldsymbol{y_{\mathbb{o},t}^k} = 1 \qquad \forall k \in \mathcal{V}, t \in [0, t_{\mathbb{o}}^k) \qquad (5.33)$$

$$\sum_{i \in \mathcal{B} \cup \{\mathbb{o}, \mathbb{i}\}} \boldsymbol{y_{i,T}^k} = 1 \qquad \forall k \in \mathcal{V} \qquad (5.34)$$

$$\boldsymbol{\epsilon_{i,t}} \geq 0 \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \qquad (5.35)$$

$$\boldsymbol{\epsilon_{i,t}} \geq \boldsymbol{C_{i,t}} - \overline{C}_i \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \qquad (5.36)$$

$$\boldsymbol{\epsilon_{i,t}} \geq \underline{C_i} - \boldsymbol{C_{i,t}} \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \qquad (5.37)$$

$$\boldsymbol{x_{i,t}^{k}} \in \{0,1\} \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, t \in \hat{T} \qquad (5.38)$$

$$\boldsymbol{y_{i,t}^{k}} \in \{0,1\} \qquad \forall i \in \mathcal{B}_{\mathbb{oi}}, k \in \mathcal{V}, t \in \hat{T} \qquad (5.39)$$

$$\boldsymbol{L_{i,t}^{k,\theta}} \in \mathbb{Z}_0 \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{0\} \qquad (5.40)$$

The objective function $min\ f = \sum_i \sum_t \boldsymbol{\epsilon_{i,t}}$ aims at minimizing the overall portion of buffer contents exceeding the desired limits $[C_i, \overline{C}_i]$.

Buffer contents are constrained by inequations (5.20), (5.21) and (5.21'). Content values cannot take negative values (5.20) since the resulting solution would be physically impossible. Expression (5.21) handles buffer content evolutions overtime, and expression (5.21') binds the set of first variable content values ($\boldsymbol{C_{i,1}}$) to the set of initial conditions $C_{i,0}$. These two sets of equations show that at each time step $t \to t+1$, buffer contents are deprived of the quantity consumed during t-frame and the overall quantity loaded by every vehicle at instant $t+1$. These constraints only exist for $\theta = \theta_i$. (Indeed, for all $\theta \neq \theta_i$ the $\boldsymbol{L_{i,t}^{k,\theta}}$ variable do not exist.) Moreover, for $i = \mathbb{i}$ these constraints do not exist either, because the main stock has no content variable.

The constraints (5.22), (5.23), (5.24) and (5.24') bind vehicle content evolution to the quantities exchanged with vehicles. First, (5.22) impose to every $\boldsymbol{q}$-value to be greater or equal to zero since negative contents have no physical meaning. Second, expression (5.23) ensures that vehicle capacities are not exceeded. Last, (5.24) manages vehicle content evolutions overtime and (5.24') binds the set of first variable content values ($\boldsymbol{q_1^{k,\theta}}$) to the initial contents $q_0^{k,\theta}$.

Inequations (5.25), (5.26), (5.27) and (5.28) have a crucial role in allowing content exchanges between vehicles and buffers. In fact, the lot exchange variables $\boldsymbol{L_{i,t}^{k,\theta}}$ can be different from zero only if a *do-*$i \to *$ transition occurs at the same time. These four constraint sets must ensure the following relationships:

do-$i \to *$ transition at instant $t+1$	$\boldsymbol{x_{i,t}^{k}}$	$\boldsymbol{x_{i,t+1}^{k}}$		$\boldsymbol{L_{i,t+1}^{k,\theta_i}}$
no	0	0		0
no	0	1		0
yes	1	0	$\Rightarrow$	*unbound*
no	1	1		0

In other terms, the following expression must be verified:

$$x_{i,t}^{k} = 1 \vee x_{i,t}^{k} = 0 \iff x_{i,t}^{k} - x_{i,t+1}^{k} < 1 \Rightarrow L_{i,t+1}^{k,\theta} = 0, \quad \begin{matrix} \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V} \\ \forall \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{T\} \end{matrix}$$

The M parameter in (5.25), (5.26), (5.27), (5.28) must take the smallest value that can overcome any possible $\left|L_{i,t+1}^{k,\theta}\right|$. It is calculated as:

$$M = \min\left\{ \max_{k \in \mathcal{V}, \theta \in \mathcal{A}} \{Q_k / \lambda_\theta\}, \max_{i \in \mathcal{B}_{\mathbb{i}}} \{\overline{C}_i / \lambda_{\theta_i}\} \right\}$$

Constraints (5.29) to (5.34) have the function of binding state variables $x_{i,t}^{k}$, $y_{i,t}^{k}$ overtime. Namely, they define the consistent set of rules that apply to vehicle route making. Expression (5.29) constrains each vehicle to lie in *one and only state* at *one and only buffer* at a time.

The set of constraints expressed at (5.30) deserves a special attention. In fact, it permits to schedule *multi-timeframe do*-jobs, and thus to freely scale model time granularity. On the other hand, by increasing time granularity the formulation complexifies and becomes harder to solve; a fair trade-off must be found. This set of constraints states that after a $* \rightarrow$ *do-j* transition at instant t, the following $x_{j,*}^{k}$ are equal to 1 until the *do*-job is not over, i.e., as long as $t \leq \Pi_{ij,t}^{k} = \left\lceil t + d_{ij}/s_k + \tau_j \right\rceil$. The Π parameters are fundamental for writing these constraints, as it corresponds to the rounded-up integer finish time of each hypothetical *do*-job performed by k, starting at t, and going from buffer i to buffer j. Let us see the origin of this constraint set; the $* \rightarrow$ *do-j* transition that triggers the constraint only exists if $(x_{j,t}^{k}, x_{j,t+1}^{k}) = (0,1)$. Since $x_{j,t}^{k} = 0$, there must be a buffer $i \neq j$ for which $(x_{i,t}^{k} + y_{i,t}^{k}) = 1$. Let $\pi \in \left\{ t^* \mid t^* \in \hat{T}, t{+}1 < t^* \leq \Pi_{ij,t}^{k} \right\}$, the relationships to respect are:

$* \rightarrow$ *do-j* transition at instant $t{+}1$	$x_{i,t}^{k} + y_{i,t}^{k}$	$x_{j,t+1}^{k}$		$x_{j,\pi}^{k}$
no	0	0		*unbound*
no	0	1		*unbound*
no	1	0	$\Rightarrow$	*unbound*
yes	1	1		1

In other terms, the following expression must be verified:

$$\left(x_{i,t}^{k} + y_{i,t}^{k}\right) = 1 \wedge x_{j,t+1}^{k} = 1 \iff \iff \left(x_{i,t}^{k} + y_{i,t}^{k}\right) + x_{j,t+1}^{k} = 2 \Rightarrow x_{j,\pi}^{k} = 1, \quad \begin{matrix} \forall i, j \in \mathcal{B}_{\mathbb{i}}, i \neq j, k \in \mathcal{V} \\ \forall t \in \hat{T} \setminus \{T\}, \pi \in \left[t + 2, \Pi_{ij,t}^{k}\right] \end{matrix}$$

For $i = \mathbb{o}$, the expression becomes as in (5.30'), since the initial condition $\mathbb{o}$ cannot be a *do*-job destination, and consequently variables $x_{\mathbb{o},t}^{k}$ do not exist.

Expression (5.31) states that vehicles can pass to *wait* state only at the buffer they visited last. Therefore, the $y_{i,t+1}^{k}$ variable can be set to 1 *if and only if* either $x_{i,t}^{k}$ or $y_{i,t}^{k}$ is equal to 1. For $i = \mathbb{o}$, the expression becomes as in (5.31'). Expression (5.32) states that a vehicle waiting at a buffer cannot start a *do*-job toward the same buffer: $y_{i,t}^{k} = 1 \rightarrow x_{i,t+1}^{k} = 0$. Constraint (5.33) is to be used in case some vehicles are not immediately available to start *do*-jobs. They are put in *wait* state at their initial

condition $\mathbb{o}$ as long as their availability time $t_{\mathbb{o}}^{k}$ is not reached. Finally, (5.34) grants that no vehicle can finish the scheduling period in a *do* state. In fact, a *do*-job cannot start if it cannot be finished, too.
Constraints (5.35), (5.36), (5.37) rule the auxiliary variables $\boldsymbol{\epsilon_{i,t}}$, that must satisfy the inequation $\boldsymbol{\epsilon_{i,t}} \geq \max\{0, \boldsymbol{C_{i,t}} - \overline{C}_i, \underline{C}_i - \boldsymbol{C_{i,t}}\}$.
Finally, (5.38) and (5.39) express the Boolean domain of variables $\boldsymbol{x_{i,t}^{k}}$ and $\boldsymbol{y_{i,t}^{k}}$, while (5.40) imposes the integral domain of $\boldsymbol{L_{i,t}^{k,\theta}}$. All the other variables have continuous domains.

5.3.4 DT-1ms-MILP results post-processing

The DT-1ms-MILP formulation was chosen as the most appropriate for solving the ILOM at the basis of this thesis and the internship. Therefore, the variables forming DT-1ms-MILP solution must be interpreted so as to be understood, evaluated, compared, and visualized. The following algorithm creates a set of tasks $\mathcal{Z} = \{z(t_s, t_e, i, k, L^{\theta}) \mid (i,k,\theta)^T \in (\mathcal{B}, \mathcal{V}, \mathcal{A})^T, 0 \leq t_s \leq t_e \leq T_P\}$, where each task z has a start time t_s, an end time t_e, an appointed vehicle k, a destination i, and a set of quantities to pick-up L^{θ}.

Standards:

1. In pseudocode, MILP solution variables are called with the expression $\mathcal{S}$[<var.name>][<indexes>].
2. Tasks are of two types: *do* and *wait* tasks. A wait-task differs from a do-task as its destination does not change with respect to the current buffer. Moreover, wait-tasks always have pick-up quantities L^{θ} set at zero.
3. To make it simpler, let us assume a $\Delta\hat{t} = 1$.
4. The symbol $\star$ indicates a yet undefined quantity, specified later by the algorithm.

Algorithm 5.2. Conversion of DT-1ms-MILP solution variables into tasks

Initialize tasks set $\mathcal{Z} \leftarrow \emptyset$
CREATION OF TASKS:
For each triplet (i,k,t)
  If $\mathcal{S}[\boldsymbol{x}][i,k,t] = 1$ (i.e., k in a *do* state toward i at time t)
    If there is a do-task $z_{do} \in \mathcal{Z}$ such that $(i,k,t_e)_{z_{do}}^{T} = (i,k,t)^T$
      Update the end time of z_{do}: $t_e^{z_{do}} \leftarrow t_e^{z_{do}} + 1$
    Else
      Create a new do-task: $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{z(t, t+1, i, k, \star\}$
    End if
  Elsif $\mathcal{S}[\boldsymbol{y}][i,k,t] = 1$ (i.e., k in a *wait* state at i at time t)
    If there is a wait-task $z_{wait} \in \mathcal{Z}$ such that $(i,k,t_e)_{z_{wait}}^{T} = (i,k,t)^T$
      Update the end time of z_{wait}: $t_e^{z_{wait}} \leftarrow t_e^{z_{wait}} + 1$

12 **Else**
13 Create a new wait-task: $\mathcal{Z} \leftarrow \mathcal{Z} \cup \{z(t, t+1, i, k, 0\}$
14 **End it**
15 **End if**
16 **Next** (i, k, t)
DEFINITION OF LOADED/UNLOADED QUANTITIES:
17 **For** each do-task $z \in \mathcal{Z}$
18 **For** each article $\theta \in \mathcal{A}$
19 $L^{\theta,z} \leftarrow$ value of variable $\mathcal{S}[L][i^z, k^z, \theta, t_e^z]$
20 **Next** θ
21 **Next** z

5.4 DT-Ms²-Mib-MILP formulation

The *Discrete-Time Multiple-source/sinks Multi-item-buffers MILP* (DT-Ms²-Mib-MILP) formulation is the generalisation of DT-1ms-MILP. It can be applied to the homonymous problem, in which there is not a unique main stock and each buffer capacity can be shared by more than a single allowed article. Furthermore, vehicles can be completely heterogeneous and can possibly support only a limited set of articles. The standards adopted for this formulation are the same seen for DT-1ms-MILP. However, concerning the involved elements there are some differences:

- Buffers are *multi-item* and can contain many different item references until a common content limit is reached. The set of supported articles at buffer i is indicated with $\mathcal{A}_i \subseteq \mathcal{A}$.
- Some articles can enter and exit the problem via *source/sink* elements. These elements behave like buffers with endless content and capacity, and the joint set of buffers and source/sinks is indicated by symbol $\mathcal{BS}$. The set $\mathcal{BS} \cup \{\mathbb{0}\}$ is also written $\mathcal{BS}_{\mathbb{0}}$.
- Vehicles can possibly support a limited set of articles indicated with $\mathcal{A}_k \subseteq \mathcal{A}$. This feature can be useful in case articles have different physical state or properties (for example, solid and liquid products, type of packaging, items stored at different temperatures, etc.).
  The set $\mathcal{A}_i \cap \mathcal{A}_k$ is also written $\mathcal{A}_{ik}, \forall i \in \mathcal{B}, k \in \mathcal{V}$.

Table 5.8. Sets of DT-Ms²-Mib-MILP.

Set			Set element notations	
Name	Symbol	Index	Complete	Simplified
Buffers	$\mathcal{B}$	i, j	$\{b_i\}_{i\in\{1,2,\dots,B\}}$	$\{1,2,\dots,B\}$
Source/sinks	$\mathcal{S}$	i, j	$\{s_i\}_{i\in\{1,2,\dots,S\}}$	$\{1,2,\dots,S\}$
Buffers & s/s	$\mathcal{BS}$	i, j	$\{bs_i\}_{i\in\{1,2,\dots,BS\}}$	$\{1,2,\dots,BS\}$

Vehicles	$\mathcal{V}$	k	$\{v_k\}_{k\in\{1,2,\dots,K\}}$	$\{1,2,\dots,K\}$
Articles	$\mathcal{A}$	θ	$\{a_\theta\}_{\theta\in\{1,2,\dots,\Theta\}}$	$\{1,2,\dots,\Theta\}$
Timeframes	$\hat{T}$	t	$\{\hat{t}_t\}_{t\in\{0,1,2,\dots,T\}}$	$\{0,1,2,\dots,T\}$

5.4.1 Parameters and variables

The nomenclature of model parameters is conceptually the same shown for DT-1ms-MILP, and the most modified parts are the 'Index domains'.

Table 5.9. Parameters of DT-Ms²-Mib-MILP.

	Description	Index domains
λ_θ	load units in a lot of θ	$\theta \in \mathcal{A}$
$\overline{C}_i$	upper content limit (capacity) of buffer i	$i \in \mathcal{B}$
$\underline{C}_i^\theta$	lower content limit of buffer i for article θ	$i \in \mathcal{B}$, $\theta \in \mathcal{A}_i$
$C_{i,0}^\theta$	initial content in buffer i of article θ	$i \in \mathcal{B}$, $\theta \in \mathcal{A}_i$
$\gamma_{i,t}^\theta$	consumption of article θ at buffer i during t	$i \in \mathcal{B}$, $\theta \in \mathcal{A}_i$ $t \in \hat{T} \setminus \{T\}$
d_{ij}	travelling distance from i to j	$i,j \in \mathcal{BS}, i \neq j$
τ_i	time required to perform onsite operations at i	$i \in \mathcal{BS}$
$\overline{w}_i$	penalty weight of overstock at i	$i \in \mathcal{B}$
$\underline{w}_i^\theta$	penalty weight of stockout of θ at i	$i \in \mathcal{B}$, $\theta \in \mathcal{A}_i$

All vehicle parameters are identical to those in DT-1ms-MILP formulation, except for article index domains that change from $\mathcal{A}$ to $\mathcal{A}_k$.

Table 5.10. Variables of DT-Ms²-Mib-MILP.

	Description	Index domains
$\boldsymbol{x_{i,t}^k}$	Boolean var := $\begin{cases}1 \text{ if vehicle } k \text{ is in } do \text{ state during } t\\ 0 \text{ otherwise}\end{cases}$	$i \in \mathcal{BS}$, $k \in \mathcal{V}$, $t \in \hat{T}$
$\boldsymbol{y_{i,t}^k}$	Boolean var := $\begin{cases}1 \text{ if vehicle } k \text{ is in } wait \text{ state during } t\\ 0 \text{ otherwise}\end{cases}$	$i \in \mathcal{BS} \cup \{\textcircled{0}\}$, $k \in \mathcal{V}$, $t \in \hat{T}$
$\boldsymbol{C_{i,t}^\theta}$	quantity of θ at buffer i at instant t	$i \in \mathcal{B}$, $\theta \in \mathcal{A}_i$ $t \in \hat{T} \setminus \{0\}$

$\boldsymbol{q_t^{k,\theta}}$	quantity of θ on vehicle k at instant t	$k \in \mathcal{V}$, $\theta \in \mathcal{A}_k$, $t \in \hat{T} \setminus \{0\}$
$\boldsymbol{L_{i,t}^{k,\theta}}$	number of lots of θ, loaded at i by k at instant t	$i \in \mathcal{BS}$, $k \in \mathcal{V}$, $\theta \in \mathcal{A}_i \cap \mathcal{A}_k$, $t \in \hat{T} \setminus \{0\}$
Auxiliary variables		
$\overline{\boldsymbol{\epsilon}}_{i,t}$	$\geq \max\{0, \sum_{\theta \in \mathcal{A}_i} \boldsymbol{C_{i,t}^{\theta}} - \overline{C}_i\}$	$i \in \mathcal{B}$, $t \in \hat{T} \setminus \{0\}$
$\underline{\boldsymbol{\epsilon}}_{i,t}^{\boldsymbol{\theta}}$	$\geq \max\{0, \underline{C}_i^{\theta} - \boldsymbol{C_{i,t}^{\theta}}\}$	$i \in \mathcal{B}$, $\theta \in \mathcal{A}_i$, $t \in \hat{T} \setminus \{0\}$

5.4.2 DT-Ms²-Mib-MILP model

The DT-Ms²-Mib-MILP model is formulated as follows:

$$Minimize\ f = \sum_{i \in \mathcal{B}} \sum_{t \in \hat{T} \setminus \{0\}} \overline{w}_i \overline{\boldsymbol{\epsilon}}_{i,t} + \sum_{i \in \mathcal{B}} \sum_{\theta \in \mathcal{A}_i} \sum_{t \in \hat{T} \setminus \{0\}} \underline{w}_i^{\theta} \underline{\boldsymbol{\epsilon}}_{i,t}^{\boldsymbol{\theta}} \tag{5.41}$$

Subject to:

$$\boldsymbol{C_{i,t}^{\theta}} \geq 0 \qquad \forall i \in \mathcal{B}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{0\} \tag{5.42}$$

$$\boldsymbol{C_{i,t+1}^{\theta}} = \boldsymbol{C_{i,t}^{\theta}} - \gamma_{i,t}^{\theta} - \sum_{k \in \mathcal{V} \cap \{k' | \theta \in \mathcal{A}_{k'}\}} \lambda_{\theta} \boldsymbol{L_{i,t+1}^{k,\theta}} \qquad \forall i \in \mathcal{B}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{0, T\} \tag{5.43}$$

$$\boldsymbol{C_{i,1}^{\theta}} = C_{i,0}^{\theta} - \gamma_{i,0}^{\theta} - \sum_{k \in \mathcal{V} \cap \{k' | \theta \in \mathcal{A}_{k'}\}} \lambda_{\theta} \boldsymbol{L_{i,1}^{k,\theta}} \qquad \forall i \in \mathcal{B} \tag{5.43'}$$

$$\boldsymbol{q_t^{k,\theta}} \geq 0 \qquad \forall k \in \mathcal{V}, \theta \in \mathcal{A}_k, t \in \hat{T} \setminus \{0\} \tag{5.44}$$

$$\sum_{\theta \in \mathcal{A}_k} \boldsymbol{q_t^{k,\theta}} \leq Q_k \qquad \forall k \in \mathcal{V}, t \in \hat{T} \setminus \{0\} \tag{5.45}$$

$$\boldsymbol{q_{t+1}^{k,\theta}} = \boldsymbol{q_t^{k,\theta}} + \sum_{i \in \mathcal{B} \cap \{i' | \theta \in \mathcal{A}_{i'}\}} \lambda_{\theta} \boldsymbol{L_{i,t+1}^{k,\theta}} \qquad \forall k \in \mathcal{V}, \theta \in \mathcal{A}_k, t \in \hat{T} \setminus \{0, T\} \tag{5.46}$$

$$\boldsymbol{q_1^{k,\theta}} = q_0^{k,\theta} + \sum_{i \in \mathcal{B} \cap \{i' | \theta \in \mathcal{A}_{i'}\}} \lambda_{\theta} \boldsymbol{L_{i,1}^{k,\theta}} \qquad \forall k \in \mathcal{V}, \theta \in \mathcal{A}_k \tag{5.46'}$$

$$M \boldsymbol{x_{i,t}^{k}} + \boldsymbol{L_{i,t+1}^{k,\theta}} \geq 0 \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, \theta \in \mathcal{A}_{ik}, t \in \hat{T} \setminus \{T\} \tag{5.47}$$

$$-M\boldsymbol{x}_{i,t}^{k} + \boldsymbol{L}_{i,t+1}^{k,\theta} \leq 0 \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, \theta \in \mathcal{A}_{ik}, t \in \hat{T} \setminus \{T\} \qquad (5.48)$$

$$M\left(1 - \boldsymbol{x}_{i,t+1}^{k}\right) + \boldsymbol{L}_{i,t+1}^{k,\theta} \geq 0 \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, \theta \in \mathcal{A}_{ik}, t \in \hat{T} \setminus \{T\} \qquad (5.49)$$

$$-M\left(1 - \boldsymbol{x}_{i,t+1}^{k}\right) + \boldsymbol{L}_{i,t+1}^{k,\theta} \leq 0 \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, \theta \in \mathcal{A}_{ik}, t \in \hat{T} \setminus \{T\} \qquad (5.50)$$

$$\sum_{i \in \mathcal{B} \cup \{\mathbb{i}\}} \boldsymbol{x}_{i,t}^{k} + \sum_{i \in \mathcal{B} \cup \{\mathbb{o}, \mathbb{i}\}} \boldsymbol{y}_{i,t}^{k} = 1 \qquad \forall k \in \mathcal{V}, t \in \hat{T} \qquad (5.51)$$

$$1 - \left(\boldsymbol{x}_{i,t}^{k} + \boldsymbol{y}_{i,t}^{k}\right) - \boldsymbol{x}_{j,t+1}^{k} + \boldsymbol{x}_{j,\pi}^{k} \geq 0 \qquad \begin{matrix} \forall i, j \in \mathcal{B}_{\mathbb{i}}, i \neq j, k \in \mathcal{V} \\ \forall t \in \hat{T} \setminus \{T\}, \pi \in \left[t+2, \Pi_{ij,t}^{k}\right] \end{matrix} \qquad (5.52)$$

$$1 - \boldsymbol{y}_{\mathbb{o},t}^{k} - \boldsymbol{x}_{j,t+1}^{k} + \boldsymbol{x}_{j,\pi}^{k} \geq 0 \qquad \begin{matrix} \forall j \in \mathcal{B}_{\mathbb{i}}, i \neq j, k \in \mathcal{V} \\ \forall t \in \hat{T} \setminus \{T\}, \pi \in \left[t+2, \Pi_{ij,t}^{k}\right] \end{matrix} \qquad (5.52')$$

$$\boldsymbol{x}_{i,t}^{k} + \boldsymbol{y}_{i,t}^{k} \geq \boldsymbol{y}_{i,t+1}^{k} \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, t \in \hat{T} \setminus \{T\} \qquad (5.53)$$

$$\boldsymbol{y}_{\mathbb{o},t}^{k} \geq \boldsymbol{y}_{\mathbb{o},t+1}^{k} \qquad \forall k \in \mathcal{V}, \hat{t} \in \hat{T} \setminus \{T\} \qquad (5.53')$$

$$1 - \boldsymbol{y}_{i,t}^{k} \geq \boldsymbol{x}_{i,t+1}^{k} \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, t \in \hat{T} \setminus \{T\} \qquad (5.54)$$

$$\boldsymbol{y}_{\mathbb{o},t}^{k} = 1 \qquad \forall k \in \mathcal{V}, t \in [0, t_{\mathbb{o}}^{k}) \qquad (5.55)$$

$$\sum_{i \in \mathcal{B} \cup \{\mathbb{o}, \mathbb{i}\}} \boldsymbol{y}_{i,T}^{k} = 1 \qquad \forall k \in \mathcal{V} \qquad (5.56)$$

$$\overline{\boldsymbol{\epsilon}}_{i,t} \geq 0 \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \qquad (5.57)$$

$$\overline{\boldsymbol{\epsilon}}_{i,t} \geq \sum_{\theta \in \mathcal{A}_i} \boldsymbol{C}_{i,t}^{\theta} - \overline{C}_i \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \qquad (5.58)$$

$$\underline{\boldsymbol{\epsilon}}_{i,t}^{\theta} \geq 0 \qquad \forall i \in \mathcal{B}, t \in \hat{T} \setminus \{0\} \qquad (5.59)$$

$$\underline{\boldsymbol{\epsilon}}_{i,t}^{\theta} \geq \underline{C}_i^{\theta} - \boldsymbol{C}_{i,t}^{\theta} \qquad \forall i \in \mathcal{B}, \theta \in \mathcal{A}_i, t \in \hat{T} \setminus \{0\} \qquad (5.60)$$

$$\boldsymbol{x}_{i,t}^{k} \in \{0,1\} \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, t \in \hat{T} \qquad (5.61)$$

$$\boldsymbol{y}_{i,t}^{k} \in \{0,1\} \qquad \forall i \in \mathcal{BS} \cup \{\mathbb{o}\}, k \in \mathcal{V}, t \in \hat{T} \qquad (5.62)$$

$$\boldsymbol{L}_{i,t}^{k,\theta} \in \mathbb{Z}_0 \qquad \forall i \in \mathcal{BS}, k \in \mathcal{V}, \theta \in \mathcal{A}_{ik}, t \in \hat{T} \setminus \{0\} \qquad (5.63)$$

The objective function $min\, f = \sum_i \sum_t \overline{w}_i \overline{\boldsymbol{\epsilon}}_{i,t} + \sum_i \sum_\theta \sum_t \underline{w}_i^{\theta} \underline{\boldsymbol{\epsilon}}_{i,t}^{\boldsymbol{\theta}}$ aims at minimizing the weighted overstock and stockout for each article at each buffer. Constraints can be understood by reading the DT-1ms-MILP model description.

5.4.3 DT-Ms²-Mib-MILP with distance minimization

A few modifications of the model permit to integrate distance minimization in the objective function. Even though distance minimization has a secondary role in the *indoor logistic operation manager* problem, it is crucial to effectively solve the *supply-chain network* one. On the other hand, adding this feature to the model can sensibly complexify it and result in much longer solving times.

Let us consider the DT-Ms²-Mib-MILP formulation. Each new transfer from i to j begins with a $* \rightarrow$ *do-j* transition. A way to identify these transitions already exists at constraint (5.30), and a new set of Boolean auxiliary variables $\boldsymbol{z_{ij,t}^{k}}$ can be used to mark any time t at which a vehicle k at i starts a new do-task toward j:

$$\boldsymbol{z_{ij,t}^{k}} \text{ Boolean var} := \begin{cases} 1 \text{ if vehicle } k \text{ begins } i \rightarrow j \text{ transfer at } t \\ 0 \text{ otherwise} \end{cases} \qquad \begin{matrix} i \in \mathcal{BS}, \\ k \in \mathcal{V},\ t \in \hat{T} \end{matrix}$$

The table hereafter reports the conditions to respect.

$* \rightarrow$ *do-j* transition at instant t+1	$\boldsymbol{x_{i,t}^{k} + y_{i,t}^{k}}$	$\boldsymbol{x_{j,t+1}^{k}}$		$\boldsymbol{z_{ij,t+1}^{k}}$
no	0	0		*unbound*
No	0	1	$\Rightarrow$	*unbound*
No	1	0		*unbound*
Yes	1	1		1

The following expression resumes the content of the table.

$$\begin{aligned} &\left(\boldsymbol{x_{i,t}^{k} + y_{i,t}^{k}}\right) = 1 \ \wedge \ \boldsymbol{x_{j,t+1}^{k}} = 1 \ \Leftrightarrow \\ &\qquad \Leftrightarrow \ \left(\boldsymbol{x_{i,t}^{k} + y_{i,t}^{k}}\right) + \ \boldsymbol{x_{j,t+1}^{k}} = 2 \ \Rightarrow \ \boldsymbol{z_{ij,t+1}^{k}} = 1, \end{aligned} \qquad \begin{matrix} \forall i, j \in \mathcal{BS}, i \neq j \\ \forall k \in \mathcal{V}, t \in \hat{T} \setminus \{T\} \end{matrix}$$

The MILP model presents therefore some additional constraints:

$$1 - \left(\boldsymbol{x_{i,t}^{k} + y_{i,t}^{k}}\right) - \ \boldsymbol{x_{j,t+1}^{k}} + \boldsymbol{z_{ij,t+1}^{k}} \geq 0 \qquad \begin{matrix} \forall i \in \mathcal{BS} \cup \{\mathbb{0}\}, j \in \mathcal{BS}, i \neq j \\ \forall k \in \mathcal{V}, t \in \hat{T} \setminus \{T\} \end{matrix} \qquad (5.64)$$

$$1 - \boldsymbol{y_{\mathbb{0},t}^{k}} - \ \boldsymbol{x_{j,t+1}^{k}} + \boldsymbol{z_{\mathbb{0}j,t+1}^{k}} \geq 0 \qquad \begin{matrix} \forall j \in \mathcal{BS}, k \in \mathcal{V} \\ \forall t \in \hat{T} \setminus \{T\} \end{matrix} \qquad (5.64')$$

$$\boldsymbol{z_{ij,t}^{k}} \in \{0,1\} \qquad \begin{matrix} \forall i \in \mathcal{BS} \cup \{\mathbb{0}\}, j \in \mathcal{BS}, i \neq j \\ \forall k \in \mathcal{V}, t \in \hat{T} \end{matrix} \qquad (5.65)$$

Finally, the objective function with distance minimization is written as:

$$\textit{Minimize } f = \sum_{i \in \mathcal{B}} \sum_{t \in \hat{T} \setminus \{0\}} \overline{w}_i \overline{\boldsymbol{\epsilon}}_{i,t} + \sum_{i \in \mathcal{B}} \sum_{\theta \in \mathcal{A}_i} \sum_{t \in \hat{T} \setminus \{0\}} \underline{w}_i^{\theta} \underline{\boldsymbol{\epsilon}}_{i,t}^{\boldsymbol{\theta}} + \qquad (5.66)$$

$$+\alpha_d \left(\sum_{i \in \mathcal{BS}} \sum_{j \in \mathcal{BS}} \sum_{k \in \mathcal{V}} \sum_{t \in \hat{T} \setminus \{0\}} d_{ij} \boldsymbol{z}^{\boldsymbol{k}}_{\boldsymbol{ij,t}} + \sum_{j \in \mathcal{BS}} \sum_{k \in \mathcal{V}} \sum_{t \in \hat{T} \setminus \{0\}} d^{k}_{\mathbb{0}j} \boldsymbol{z}^{\boldsymbol{k}}_{\mathbb{0}\boldsymbol{j,t}} \right)$$

5.5 DT-1ms-PR-MILP formulation

The DT-1ms-MILP formulation is a versatile mathematical model that allows a flexible operation scheduling with a fine time control. However, its size grows in a polynomial fashion with problem size, and it risks becoming impractical with higher numbers of buffers, vehicles, articles, and timeframes.
The *Discrete-Time 1-main-stock with Predefined Routes MILP* (DT-1ms-PR-MILP) aims at reducing the number of variables and constraints by making two assumptions:

1. **Vehicle routes are pre-defined**, and decision-making does not involve their building. This assumption can be particularly reasonable for the ILOM problem involving some flow-shop manufacturing facilities. In fact, part consumptions are generally proportional along a flow-shop, thus supplying and waste collection operations can be done with the same frequency. The whole assembly line is thus decomposed in different zones, possibly based on production models (open-shop, flow-shop, job-shop), and for each of them a circuit tying all buffers is defined. If the manufacturing plant permits it, placing loading points after unloading ones could be beneficial for solution optimality. Also, the *twin-buffers* (pair of buffers containing the same subassembly article but with opposite consumption sign) must be put in the same route, as their content cannot be unloaded at the main stock.
   With the predefined-routes assumption, the decision variables are no more about how to build pickup-delivery routes, but rather which vehicles launch on each route at the beginning of each subperiod.

2. **Any pickup-delivery route is completed in a subperiod** (discrete timeframe). This second assumption is complementary to the first, as it allows to avoid trans-period carryover of vehicle contents. I.e., each route begins and ends at the main stock during the same subperiod, therefore there is no need to stock information about vehicle content when passing from a subperiod to the following.

The solution of the DT-1ms-PR-MILP is clearly suboptimal compared to the route-building version DT-1ms-MILP. However, under some circumstances it can give good results in a much shorter time.

The DT-1ms-PR-MILP is a *discrete-time* model in which the scheduling period of length T_P is converted into a discrete set of equal subperiods $\hat{P} = \{\hat{p}_0, \hat{p}_1, \hat{p}_2, \dots, \hat{p}_P\}$.

The exact state of the system within a subperiod is not entirely known, but the solution gives to each vehicle a list of pickup-deliveries to do during each subperiod. About formulation indexes:

- Buffers are indexed with letters i and j; routes with letter w; vehicles with letter k; subperiods with letter p.
- The main stock is indicated with symbol $\mathbb{i}$, and vehicles initial conditions are indicated with symbol $\mathbb{o}$. The set $\mathcal{B} \cup \{\mathbb{i}\}$ is also written $\mathcal{B}_{\mathbb{i}}$.
- For sake of readability, the elements of $\mathcal{B}$, $\mathcal{R}$, $\mathcal{V}$, $\hat{P}$ are directly written as their indexes, as reported in the table below at the column 'Simplified'.

Table 5.11. Sets of DT-1ms-PR-MILP.

Set			Set element notations	
Name	Symbol	Index	Complete	Simplified
Buffers	$\mathcal{B}$	i, j	$\{b_i\}_{i\in\{1,2,\dots,B\}}$	$\{1,2,\dots,B\}$
Routes	$\mathcal{R}$	w	$\{r_w\}_{w\in\{1,2,\dots,W\}}$	$\{1,2,\dots,W\}$
Vehicles	$\mathcal{V}$	k	$\{v_k\}_{k\in\{1,2,\dots,K\}}$	$\{1,2,\dots,K\}$
Subperiods	$\hat{P}$	p	$\{\hat{p}_p\}_{p\in\{1,2,\dots,P\}}$	$\{1,2,\dots,P\}$

5.5.1 Route building and subperiod length

In accordance with the second assumption made above, the length of subperiods $\Delta\hat{p}$ must be large enough to grant the completion of any predefined route within its duration. To calculate it, the following data is necessary:

Ω_w ordered set of buffers in route w, $\forall w \in \mathcal{R}$,
d_{ij} travelling distance between each couple of buffers $i, j \in \mathcal{B}_{\mathbb{i}}$, $i \neq j$,
τ_i time required to perform onsite operations at i, $\forall i \in \mathcal{B}_{\mathbb{i}}$,
s_k speed of vehicle k, $\forall k \in \mathcal{V}$,
$\bar{s}$ maximum speed of any vehicle $= \max\{s_k \mid k \in \mathcal{V}\}$.

The minimum subperiod length necessary to fully cover any route is calculated as:

$$\Delta\hat{p} = \max_{w\in\mathcal{R}} \left\{ 2\tau_{\mathbb{i}} + \bar{s}\cdot\left(d_{\mathbb{i},\inf\Omega_w} + d_{\sup\Omega_w,\mathbb{i}}\right) + \sum_{i\in\Omega_w\setminus\{\sup\Omega_w\}} \left(\tau_i + \bar{s}\cdot d_{i,i+1}\right) + \tau_{\sup\Omega_w} \right\} \quad (5.67)$$

where $\tau_{\mathbb{i}}$ is counted twice, once at the beginning, and once at the end of the route.

The duration of subperiods is also necessary to calculate the consumptions values of $\gamma_{i,p}$ during each period p. A fair strategy for calculating γ has a positive impact on scheduling reliability, since a finer content control within a subperiod is not permitted by the DT-1ms-PR-MILP formulation. In other terms, since it is not possible to know the exact moment at which a buffer will be visited within a period, content

and consumptions value shall be calculated in a form that grants to really prevent stockouts.

5.5.2 Parameters and variables

Table 5.12. Parameters of DT-1ms-PR-MILP.

	Description	Index domains
λ_i	load units in a lot of the articles at buffer i	$i \in \mathcal{B}$
$\underline{C_i}$	lower content limit of buffer i	$i \in \mathcal{B}$
$\overline{C_i}$	upper content limit of buffer i	$i \in \mathcal{B}$
$C_{i,0}$	initial content of buffer i	$i \in \mathcal{B}$
$\gamma_{i,p}$	consumption of buffer i during p	$i \in \mathcal{B}$, $p \in \hat{P}$
Q_k	capacity of vehicle k, in load units	$k \in \mathcal{V}$
$p_{\mathbb{0}}^k$	period from which vehicle k is available	$k \in \mathcal{V}$
M	sufficiently big number $= \min\left\{\max_{k\in\mathcal{V},i\in\mathcal{B}}\{Q_k/\lambda_i\}, \max_{i\in\mathcal{B}_{\mathbb{i}}}\{\overline{C}_i/\lambda_i\}\right\}$	
Route subsets		
Ω_w	ordered set of buffers in route w	$w \in \mathcal{R}$
ω_w	subset of Ω_w containing the buffers that can exchange their content with the main stock $\mathbb{i}$.	$w \in \mathcal{R}$
$\widetilde{\omega}_w$	subset of Ω_w containing the buffers that cannot exchange their content with the main stock $\mathbb{i}$.	$w \in \mathcal{R}$

As shown in figure 5.1, for all $w \in \mathcal{R}$ the sets ω_w and $\widetilde{\omega}_w$ are disjoint ($\omega_w \cap \widetilde{\omega}_w = \emptyset$) and one the complement of the other with respect to Ω_w ($\omega_w \cup \widetilde{\omega}_w = \Omega_w$).

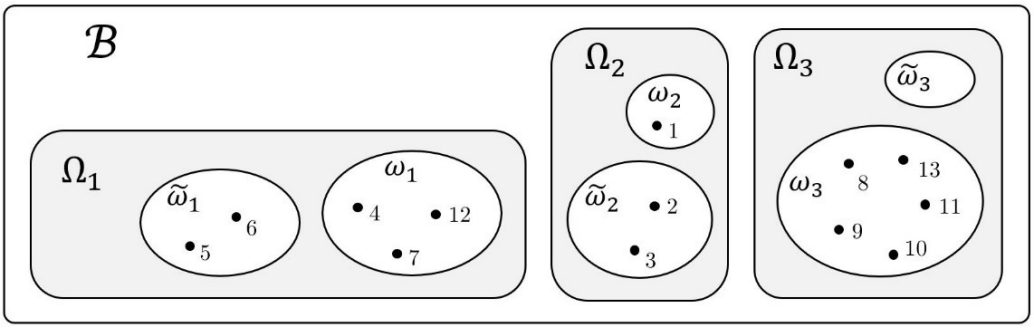


Figure 5.1. Visual example of buffer repartition in the DT-1ms-PR-MILP

Table 5.13. Variables of DT-1ms-PR-MILP.

	Description	Index domains
$\boldsymbol{x_{w,p}^{k}}$	Boolean var := $\begin{cases} 1 \text{ if } k \text{ covers routes } w \text{ during } p \\ 0 \text{ otherwise} \end{cases}$	$w \in \mathcal{R}$, $k \in \mathcal{V}$ $p \in \hat{P}$
$\boldsymbol{C_{i,p}}$	content of buffer i at the end of period p	$i \in \mathcal{B}$, $p \in \hat{P}$
$\boldsymbol{L_{i,p}^{k}}$	number of lots loaded at i by k during p	$i \in \mathcal{B}_{\mathbb{I}}$, $k \in \mathcal{V}$, $p \in \hat{P}$
Auxiliary variable		
$\boldsymbol{\epsilon_{i,p}}$	$\geq \max\{0, \boldsymbol{C_{i,p}} - \overline{C}_i, \underline{C}_i - \boldsymbol{C_{i,p}}\}$	$i \in \mathcal{B}$, $p \in \hat{P}$

5.5.3 DT-1ms-MILP model

The DT-1ms-MILP model is formulated as follows:

$$\textit{Minimize } f = \sum_{i \in \mathcal{B}} \sum_{p \in \hat{P} \setminus \{1\}} \boldsymbol{\epsilon_{i,p}} \tag{5.68}$$

Subject to:

$$\boldsymbol{C_{i,p}} \geq 0 \qquad \forall i \in \mathcal{B}, p \in \hat{P} \tag{5.69}$$

$$\boldsymbol{C_{i,p+1}} = \boldsymbol{C_{i,p}} - \gamma_{i,p+1} - \lambda_i \cdot \sum_{k \in \mathcal{V}} \boldsymbol{L_{i,p+1}^{k}} \qquad \forall i \in \mathcal{B}, p \in \hat{P} \setminus \{P\} \tag{5.70}$$

$$\boldsymbol{C_{i,1}} = C_{i,0} - \gamma_{i,1} - \lambda_i \cdot \sum_{k \in \mathcal{V}} \boldsymbol{L_{i,1}^{k}} \qquad \forall i \in \mathcal{B} \tag{5.70'}$$

$$\sum_{i \in \Omega_w}^{j} \lambda_i \boldsymbol{L_{i,p}^{k}} \geq -Q_k \qquad \forall j \in \Omega_w, w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P} \tag{5.71}$$

$$\sum_{i \in \Omega_w}^{j} \lambda_i \boldsymbol{L_{i,p}^{k}} \leq Q_k \qquad \forall j \in \Omega_w, w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P} \tag{5.72}$$

$$\sum_{i \in \widetilde{\omega}_w}^{j} \lambda_i \boldsymbol{L_{i,p}^{k}} \geq 0 \qquad \forall j \in \Omega_w, w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P} \tag{5.73}$$

$$\sum_{i \in \widetilde{\omega}_w} \lambda_i \boldsymbol{L_{i,p}^{k}} = 0 \qquad \forall w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P} \tag{5.74}$$

$$M\boldsymbol{x}_{w,p}^{k} - \boldsymbol{L}_{i,p}^{k} \geq 0 \qquad \forall i \in \Omega_w, w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P} \qquad (5.75)$$

$$-M\boldsymbol{x}_{w,p}^{k} - \boldsymbol{L}_{i,p}^{k} \leq 0 \qquad \forall i \in \Omega_w, w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P} \qquad (5.76)$$

$$\sum_{w \in \mathcal{R}} \boldsymbol{x}_{w,p}^{k} \leq 1 \qquad \forall k \in \mathcal{V}, p \in \hat{P} \qquad (5.77)$$

$$\boldsymbol{x}_{w,p}^{k} = 0 \qquad \forall w \in \mathcal{R}, k \in \mathcal{V}, p \in [1, p_{\mathbb{0}}^{k}) \qquad (5.78)$$

$$\boldsymbol{\epsilon}_{i,p} \geq 0 \qquad \forall i \in \mathcal{B}, p \in \hat{P} \qquad (5.79)$$

$$\boldsymbol{\epsilon}_{i,p} \geq \boldsymbol{C}_{i,p} - \overline{C}_i \qquad \forall i \in \mathcal{B}, p \in \hat{P} \qquad (5.80)$$

$$\boldsymbol{\epsilon}_{i,p} \geq \underline{C}_i - \boldsymbol{C}_{i,p} \qquad \forall i \in \mathcal{B}, p \in \hat{P} \qquad (5.81)$$

$$\boldsymbol{x}_{w,p}^{k} \in \{0,1\} \qquad \forall w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P} \qquad (5.82)$$

$$\boldsymbol{L}_{i,p}^{k} \in \mathbb{Z}_0 \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, k \in \mathcal{V}, p \in \hat{P} \qquad (5.83)$$

Similarly to DT-1ms-MILP, the objective function $min\, f = \sum_i \sum_p \boldsymbol{\epsilon}_{i,p}$ aims at minimizing the overall portion of buffer contents exceeding the desired limits $[C_i, \overline{C}_i]$.

(5.69) prevents buffer contents from going negative, while the expressions (5.70) and (5.70') constrain buffer content evolution.
The inequations (5.71) to (5.74) limit vehicle content evolution within a period. (5.71) and (5.72) impose that vehicle content remains between $-Q_k$ and Q_k after each visited buffer in Ω_w. If $\sum_{i \in \Omega_w} \lambda_i \boldsymbol{L}_{i,p}^{k} < 0$ it means that the vehicle shall load some articles at the main stock before starting the route. (5.73) and (5.74) add two additional rules for the articles that are not exchangeable with the main stock; (5.73) imposes that the sum of loaded/unloaded articles at buffers in $\widetilde{\omega}_w$ is always greater than zero, as these articles cannot be loaded at the main stock; (5.74) imposes that, for each route, the end balance of loaded/unloaded articles in $\widetilde{\omega}_w$ is equal to zero.
Constraints (5.75) and (5.76) set at zero the $\boldsymbol{L}_{i,p}^{k}$ variables for the subperiods in which vehicle k does not cover the route containing buffer i. The following relationships are imposed:

Route w covered by k during p ($\boldsymbol{x}_{w,p}^{k} = 1$?)	$i \in \Omega_w$		$\boldsymbol{L}_{i,p}^{k}$
no	*no*		*unbound*
no	*yes*		0
yes	*no*	$\Rightarrow$	*unbound*
yes	*yes*		*unbound*

Hence, the following expression must be verified:

$$\boldsymbol{x}_{w,p}^{k} = 0 \;\wedge\; i \in \Omega_w \;\Rightarrow\; \boldsymbol{L}_{i,p}^{k} = 0, \qquad \forall i \in \mathcal{B}_{\mathbb{i}}, w \in \mathcal{R}, k \in \mathcal{V}, p \in \hat{P}$$

The M parameter is calculated as:

$$M = \min\left\{\max_{k\in\mathcal{V},i\in\mathcal{B}}\{Q_k/\lambda_i\}, \max_{i\in\mathcal{B}_{\mathbb{i}}}\{\overline{C}_i/\lambda_i\}\right\}$$

(5.77) imposes to each vehicle to only cover a single route per subperiod. (5.78) is to be used in case some vehicles are only available starting from a period $p^k_{\mathbb{0}}$. Constraints (5.79), (5.80), (5.81) rule the auxiliary variables $\boldsymbol{\epsilon_{i,p}}$, that must satisfy the inequation $\boldsymbol{\epsilon_{i,p}} \geq \max\{0, \boldsymbol{C_{i,p}} - \overline{C}_i, \underline{C_i} - \boldsymbol{C_{i,p}}\}$. Finally, (5.82) expresses the Boolean domain of $\boldsymbol{x^k_{w,p}}$, and (5.83) the integral domain of $\boldsymbol{L^k_{i,p}}$.

NB: no constraint is imposed to limit the number of vehicles covering the same route during a period; if necessary, more than a vehicle can be simultaneously launched on a route.

Blank page

CHAPTER 6

Heuristic method: minimum penalty algorithm

The DT-1ms-MILP formulation proposed in the previous chapter demonstrated good solving performances for the *indoor logistic operations manager*. However, having the results of another method can help evaluate the performances of DT-1ms-MILP in a more significant way. To this purpose, an ad-hoc heuristic algorithm called *Minimum Penalty Algorithm* (MPA) was developed, coded, and tested. The input data of the heuristic are the same of DT-1ms-MILP, with the same GSVT (Ground, Stocks, Vehicles & Tasks) data structure. Solutions are also syntactically similar and comparable with MILP ones. In accordance with its heuristic nature, the main advantage of MPA compared to DT-1ms-MILP is a shorter solving time.

The MPA finds solutions in a semi-discrete time domain; as for DT-MILP formulations, the scheduling period T_P is divided into T timeframes of the same duration. However, unlike DT-MILP, the MPA allows some events to happen within a timeframe instead of its extremities.

6.1 Standards and notations

The notations used in this chapter are mainly the same of the previous one. Problem elements are the following:

$\mathcal{B}$	set of buffers,
$\mathcal{V}$	set of vehicles,
$\mathcal{A}$	set of article references,
$C_i(t)$	content of buffer i at time t (in continuous domain),
$\gamma_i(t)$	consumption of buffer i at time t (in continuous domain),
d_{ij}	distance from point i to point j,
τ_i	duration of onsite operations at i,
$q^{k,\theta}$	load units of article θ currently onboard of vehicle k,
t_k	availability time of vehicle k,

z	*task* object, with:	n_z	task number in a sorted task set $\mathcal{Z}$ $n_z \in [0,1,\dots,+\infty)$,
		$t_{s,z}$	start time of z,
		$t_{e,z}$	end time of z (equal to $t_{s,z}$ plus task travelling distance plus τ_{i_z}),
		i_z	destination of z,
		k_z	vehicle appointed to perform z,
		L_z^θ	lots of θ charged at i_z by k_z, $\forall\theta \in \mathcal{A}$.

6.2 Heuristic method design

The heuristic method shall have the same objective of DT-1ms-MILP formulation, i.e., to grant the satisfaction of system material needs by picking-up and delivering the required articles. The design of an algorithm directly based on that seems complicated, as stock consumptions are not stationary, and shortcomings as a function of delivery decisions are not easy to forecast. In other terms, there is no direct decision parameter (such as distance, delay, etc) that grants objective satisfaction. Moreover, the effects of a decision taken upstream can remarkably affect the system downstream in a hardly predictable way.
Despite that, the fleet and the system have some other direct parameters that can relate to the fact of 'being doing a good supporting job'. Since the fleet is composed of a limited number of vehicles, it can be logically inferred that fleet efficiency increases if pickup and deliveries are as complete as possible, and useless fleet movements are reduced.

The *Minimum Penalty Algorithm* was designed as a direct step-by-step tree-exploration heuristic, in which a decision is taken at each tree-level. It is not led by time progression, but rather system (stocks and vehicles) asynchronous evolution. Let us consider a graph $G = (V, E, \Pi)$, where V is the set of vertices representing different system states, E is the set of edges representing state evolutions (i.e., tasks) and Π is the set of penalties associated to each edge. System state s is defined by the following elements and data:

$$s = \left\{ \begin{array}{ll} \{(C_{i,0}, \boldsymbol{C_i}(t), \gamma_i(t)) & \mid i \in \mathcal{B}, t \in [0, T_P]\} \\ \{(q_0^{k,\theta}, \boldsymbol{q^{k,\theta}}, \boldsymbol{t_k}) & \mid k \in \mathcal{V}, \theta \in \mathcal{A}, 0 \le t_k \le T_P\} \end{array} \right\} \tag{6.1}$$

The elements written in bold are manipulated by the MPA, as they are part of problem solution once the algorithm is over. Another important data container of the MPA is set $\mathcal{Z}$ containing *Task* objects z:

$$\mathcal{Z} = \{\boldsymbol{z}(t_s, t_e, i, k, L^\theta) \mid (i,k,\theta)^T \in (\mathcal{B}, \mathcal{V}, \mathcal{A})^T, 0 \le t_s < t_e \le T_P\} \tag{6.2}$$

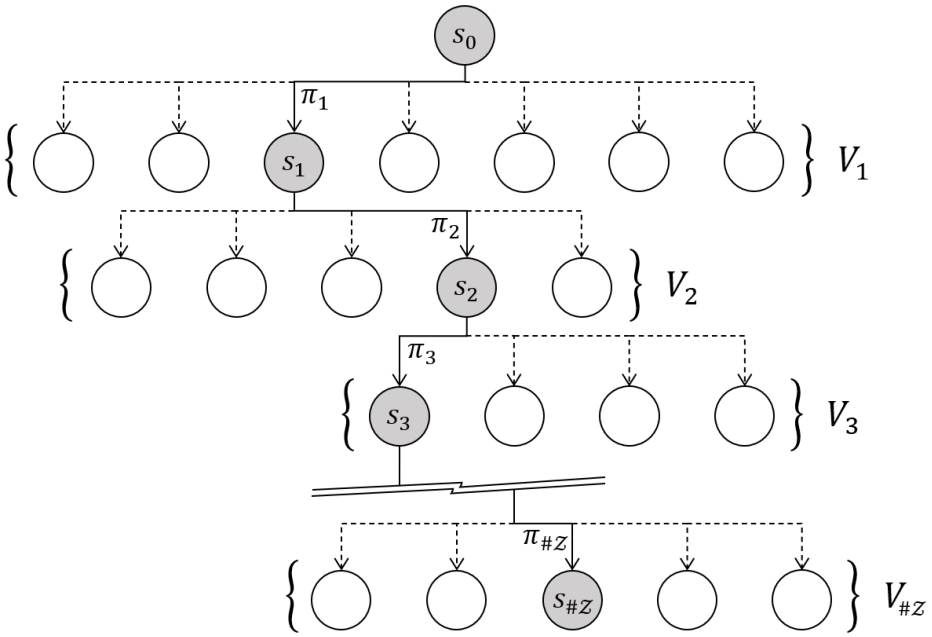


Figure 6.1. Example of MPA exploration tree.

Tasks are defined by a start time t_s, and end time t_e, an appointed vehicle k, a destination buffer i, and a set of article quantities to pick-up (or deliver) L^θ. Tasks are collected in set $\mathcal{Z}$ as the algorithm progresses.
As shown in the following, the algorithm ends when time variables t_k are equal to T_P for all vehicles $k \in \mathcal{V}$.

The graph $G(V, E, \Pi)$ is a simply connected graph the origin of which is vertex $s_0 \in V$, that represents system initial state. Vertex s_0 is connected to a series of other vertices $\{s_1', s_1'', \dots\} \in V_1 \subseteq V$ by the edges $(s_0, s_1) \in E_1 \subseteq E$. V_1 contains all system states reachable from s_0 with the execution of one and only one *Task* (*Task* object as defined in chapter 4). Namely, the involved task z could either be of type *go and pickup/delivery* or *wait a* Δt. In graph exploration, each task is associated to an edge, and allows the calculation of the associated penalty $\pi_z \in \Pi$. The penalty function only considers the current state and the states directly reachable from it – the algorithm is not regret-based. It can feature different terms and shall permit the choice of the best task in each circumstance. Therefore, an adapted and well-parameterized penalty function is essential to MPA success.
As stated before, graph exploration is step-by-step, and method visibility is limited to the current exploration level. Once the choice of s_1 is made in the minimum-penalty sense, the algorithm calculates the set of states $\{s_2', s_2'', \dots\} \in V_2 \subseteq V$ reachable by s_1 via the edges $(s_1, s_2) \in E_2$. A second task is chosen and added to $\mathcal{Z}$, then the procedure is repeated from s_2 until the end condition is reached. A graphic representation of MPA exploration tree is shown at figure 6.1.

The step-by-step approach makes the algorithm very lean in both runtime and memory usage, since it does not calculate the unexplored and useless graph regions, children of the non-selected state vertices.

MPA solution is contained in the final state s. System evolution can also be read along the elementary path traced by the algorithm in $G(V, E, \Pi)$. Once more, the algorithm is not led by system time progression. Time goes on asynchronously for every vehicle as the algorithm progresses, and buffer data is consequently modified with task creation. Indeed, MPA progression is granted by the step-by-step decision-making process that adds tasks to $\mathcal{Z}$ and updates system state.

6.2.1 MPA flowchart

The Minimum Penalty Algorithm flowchart is shown in figure 6.2.
First, system objects (buffers and vehicles) are initialized ❶, along with penalty function Π and its parameter set α ❷. The algorithm features two nested loops. The outer and main one is charged of vertical exploration of state tree. It starts with the initialization of a feasible task set $\mathrm{f}\mathcal{Z}$ ❸ that only exists inside this loop. After that, the inner loop begins. It performs the horizontal tree exploration by evaluating the feasibility and the penalty value of all tasks involving each *vehicle-buffer* combination ❹…❶❶. Once all feasible tasks are explored, the minimum-penalty one is picked ❶❷. If its penalty is smaller than the maximum allowed penalty ❶❸, then the task enters the resulting task set $\mathcal{Z}$ ❶❹. Otherwise, the picked task is discarded, and a new *wait* task is added to $\mathcal{Z}$ ❶❺. In both cases, system state is updated with the evolutions produced by the last added task ❶❻. Finally, the algorithm checks the arising of the exit-condition ❶❼. If it is verified, the algorithm ends by returning the task list $\mathcal{Z}$, else it goes back to ❸ and repeats the outer loop. Generally, the exit-condition consists in the fulfilment of all buffer needs over the period, or in the exhaustion of fleet availability (i.e., all vehicles are completely scheduled: $t_k = T_P \; \forall k \in \mathcal{V}$).

Problem constraints mentioned at ❺ and ❽ are related to loadable/unloadable quantities. In general, feasibility conditions of a task z are:

- vehicle k_z must be able to load/unload at least a lot at time $t_{e,z} - \tau_{i_z}$,
- buffer i_z must the able to yield/accept at least a lot at time $t_{e,z} - \tau_{i_z}$.

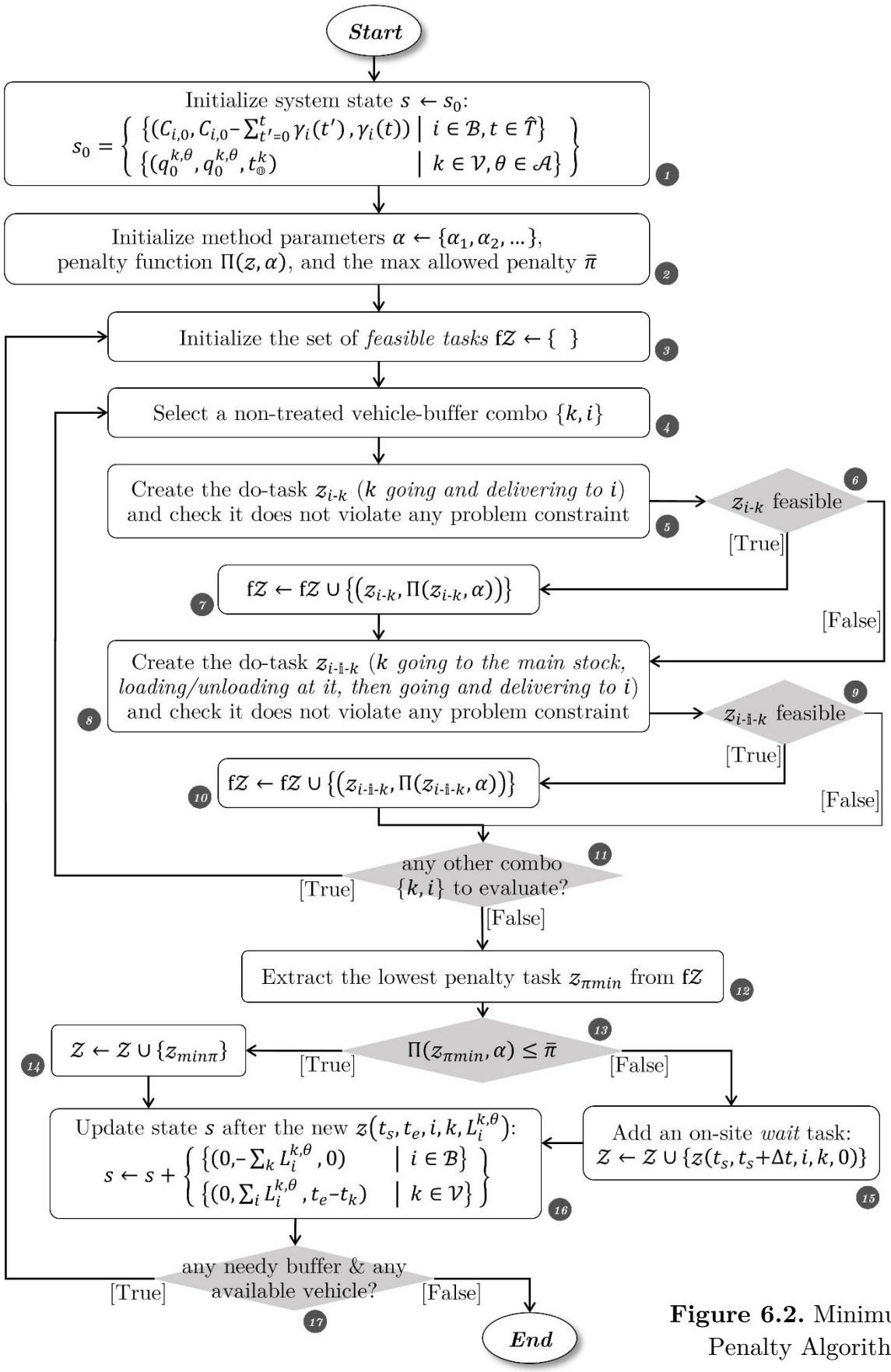


Figure 6.2. Minimum Penalty Algorithm.

6.2.2 Penalty function and acceptance conditions

The choice of a proper penalty function Π and its coefficients α is crucial to solution quality. The objective of MPA is solving the ILOM scheduling problem by granting a set of buffer pickup-deliveries that can effectively cope the provisional needs of the supported production system. Given a task $z(t_s, t_e, i, k, L)$, the function used to calculate penalty and set priority order is composed of the following terms.

1. **Gap between completion time and deadline**

$$\pi_{z,dline} = \alpha_{dline}(\hat{t}_i - t_{e,z}) \tag{6.3}$$

Proportional to the difference between the time $\hat{t}_i$ at which the buffer reaches its content threshold and the earliest time $t_{e,z}$ at which the appointed vehicle can complete the task.

2. **Relative gap between delivered and demanded**

$$\pi_{z,qty} = \alpha_{qty} \frac{D^{\theta}_{i,t_{e,z}} + \lambda_{\theta} L_z}{\overline{C}_i - \underline{C}_i} \tag{6.4}$$

$D^{\theta}_{i,t_{e,z}}$ is the demand of buffer i at task delivery time $t_{e,z}$. λ_{θ} is the lot size of article θ, and L_z is the number of lots that can be delivered by performing task z. The fact that the vehicle cannot fully satisfy buffer demand counts as a penalty; as stated above, the aim is maximizing the picked-up/delivered quantities. If the weight α_{qty} is equal to 1, $\pi_{z,qty}$ corresponds to the percentage of buffer demand that cannot be satisfied by the vehicle, scaled on the maximum number of load units that the buffer can exchange (i.e., $\overline{C}_i - \underline{C}_i$).
NB, pay attention to signs: $D^{\theta}_{i,t_{e,z}} > 0$ if the buffer has a positive content need, and $L_z > 0$ if the buffer is emptied of some content. The minimum $\pi_{z,qty}$ is reachable when $D^{\theta}_{i,t_{e,z}} = -L_z$. In addition, it is always true that $|D^{\theta}_{i,t_{e,z}}| \geq |L_z|$.

3. **Distance to destination**

$$\pi_{z,dist} = \alpha_{dist} d_z \tag{6.5}$$

Proportional to the distance d_z travelled by the vehicle doing task z.

4. ***Flag:* task completed late**

$$\pi_{z,late} = \begin{cases} 1 \text{ if } C_{i,t_{e,z}} \notin [\underline{C}_i, \overline{C}_i] \\ 0 \text{ otherwise} \end{cases} \tag{6.6}$$

Tasks in advance have the priority over late tasks, and a task is late if its completion is achieved when its content is out of the allowed limits. $\pi_{z,late}$ is a Boolean flag that stores this information and has an impact on task priority order. The developer can choose whether using this parameter on not for priority definition. Some tests have shown that considering $\pi_{z,late}$ can improve

solution quality of mildly constrained problems. In contrast, it seems to deteriorate solution quality for more rigid ones (shortage of vehicles).

5. ***Flag:*** **content runs below zero**

$$\pi_{z,below0} = \begin{cases} 1 \text{ if } C_{i,t_e^z} < 0 \\ 0 \text{ otherwise} \end{cases} \tag{6.7}$$

Since a negative buffer content makes the solution physically unfeasible, the tasks that allow a buffer to keep its content above zero have the priority over the others. $\pi_{z,below0}$ is a Boolean flag that stores this information and has an impact on task priority order.

Task priority decisions are based on the five parameters listed above, considered in the following order:

- Tasks with $\pi_{z,late} = 0$ have the priority,
- $\pi_{z,late}$ being equal, tasks with $\pi_{z,below0} = 1$ have the priority.
- $\pi_{z,late}$ and $\pi_{z,below0}$ being equal, the task with the smallest $(\pi_{z,dline} + \pi_{z,qty} + \pi_{z,dist})$ has the priority.

Another important parameter of MPA is the maximum allowed penalty value $\bar{\pi}$. Tasks can be added to set $\mathcal{Z}$ if and only if their penalty is smaller than $\bar{\pi}$. In fact, it could happen that at some point the most convenient decision is making a vehicle wait at its current location rather than starting a do-task.
The choice of the vehicle to put on hold can follow two strategies: either it will be the vehicle with the most advanced *timing* (k with the greatest t_k), or the one with the least advanced *timing* (k with the smallest t_k). Some tests showed that the solutions obtained with these two strategies do not present significant differences in quality. Nevertheless, choosing the first strategy (the most advanced *timing*) causes a greater imbalance of vehicles' activity ratio, while the second strategy allocates tasks more uniformly across the entire fleet.

6.2.3 Visits to main stock

When a vehicle visits the main stock, the algorithm forces it to unload as many articles as possible (every article compatible with the main stock). Then, the vehicle is filled of *unspecified* articles until its capacity is reached. These special *unspecified* articles are later converted either into real ones, or into empty space. After the MPA ends, a post-processing algorithm runs back through the sorted task set to specify which articles were actually loaded during main-stock visits. This aspect of the MPA makes it more effective and introduces some further visibility to task creation.

Example: an empty vehicle of capacity 5 load units (LU) visits the main stock. There it loads 5 LU of *unspecified* '□' ($q_0 =$ □□□□□). Then, it visits a buffer $b1$

where it loads 2 LU of ◩ ($\rightarrow q_1 =$ ◩◩□□□); then buffer $b2$ where it unloads 2 LU of ◨ ($\rightarrow q_2 =$ ◩◩□); finally, $b3$ where it unloads 3 LU of ◩ ($\rightarrow q_3 = \emptyset$). By a backward analysis through the sorted task set, the algorithm finds that the real vehicle content after visiting the main stock must have been $q_0 =$ ◩◨◨.

6.3 Post-processing algorithms

MPA results consist in a *vehicle-time*-sorted list of tasks. However, articles loaded at each main-stock visit are still to be specified. The following algorithm was designed to this purpose.

Algorithm 6.1. Article specification of main stock visits

1 **For** each vehicle $k \in \mathcal{V}$
2 Define the subset $\mathcal{Z}_k \coloneqq \{z \mid z \in \mathcal{Z}, k_z = k\}$
3 Sort tasks in $\mathcal{Z}_k$, *key* : greatest start-time first
4 Initialize *delivered article quantities* for vehicle k: $\mathcal{D}_\theta \leftarrow 0$, $\forall \theta \in \mathcal{A}$
5 **For** each task z in $\mathcal{Z}_k$
6 **If** z is a do-task and $i_z \neq \mathbb{i}$
7 $\mathcal{D}_\theta \leftarrow \mathcal{D}_\theta - L_z^\theta$, $\forall \theta \in \mathcal{A}$
8 **Elsif** z is a do-task and $i_z = \mathbb{i}$
9 **For** each article in $\{\theta | \theta \in \mathcal{A}_\mathbb{i}, \mathcal{L}_\theta > 0\}$
10 Calculate *real loaded units* $\mathcal{U}_\theta = \min\{L_z^{unsp}, \mathcal{D}_\theta\}$
11 $L_z^{unsp} \leftarrow L_z^{unsp} - \mathcal{U}_\theta$
12 $L_z^\theta \leftarrow \mathcal{U}_\theta$
13 $\mathcal{D}_\theta \leftarrow \mathcal{D}_\theta - \mathcal{U}_\theta$
14 **Next** θ
15 **End if**
16 **Next** z
17 Remove any residual L_z^{unsp} for any $z \in \mathcal{Z}_k$
18 **Next** k

Given any point of algorithm 6.1, $\mathcal{D}_\theta$ corresponds to the units of θ already delivered by vehicle k for which a source was not specified yet. L_z^θ is the loaded quantity of θ at the end of task z. For this reason, they have opposite signs in the expressions at line *7*. For example, if a vehicle loads in a row 1 and 2 units of θ at two different buffers, $\mathcal{D}_\theta$ is then equal to –3, which means that a source must be found for them. When the algorithm arrives to a main-stock-task (line *9*) it can convert the quantity L_z^{unsp} of unspecified loaded articles into specified L_z^θ. $\mathcal{U}_\theta$ is the number of θ units that can be exchanged with $\mathbb{i}$, equal to the minimum value between the still *unspecified units* loaded during z (L_z^{unsp}) and the delivered units of θ that still have not a source ($\mathcal{D}_\theta$). Thus, L_z^θ is set equal to $\mathcal{U}_\theta$, which is also subtracted from L_z^{unsp} and $\mathcal{D}_\theta$.

Once the article-specification algorithm is completed, some tasks may remain empty (no loaded/unloaded articles). In that case, the do-task is called *useless*, otherwise it is said *useful*. A second post-processing algorithm is required to remove useless tasks and adapt the scheduling.

Algorithm 6.2. Useless-tasks removal

1 **For** each vehicle $k \in \mathcal{V}$
2 Define the subset $\mathcal{Z}_k := \{z | z \in \mathcal{Z}, k_z = k\}$
3 Sort tasks in $\mathcal{Z}_k$, *key* : greatest $t_{z,e}$ (finish-time) first
4 Initialize *last useful task index* $n_{\hat{z}} \leftarrow 0$
5 **While** $n_z <$ cardinality$(\mathcal{Z}_k)$ **do**
6 **If** z is a do-task and $L_z \neq 0$ (useful task)
7 New *last useful task index* $n_{\hat{z}} \leftarrow n_z$
8 **Elsif** z is a do-task and $L_z = 0$ (useless task)
9 **While** z is not a useful do-task **do**
10 $n_z \leftarrow n_z + 1$
11 **Continue**
12 Remove all the tasks between $\hat{z}$ and z from $\mathcal{Z}_k$ ($\hat{z}$, z excluded)
13 Update the end-time of $\hat{z}$: $t_{e,\hat{z}} \leftarrow t_{s,\hat{z}} + d_{i_{\hat{z}} i_z} + \tau_{i_z}$
14 Add a new wait-task to fill the time gap in $\mathcal{Z}_k$: $z_{wait}(t_{e,\hat{z}}, t_{s,z}, i_{\hat{z}}, k, \emptyset)$
15 **End if**
16 **Continue**
17 **Next** k

6.4 MPA parametrization

MPA results drastically vary depending on the chosen parameters α_{dline}, α_{qty}, α_{dist} (called α_1, α_2, α_3 in the following) and $\bar{\pi}$ (π_{max}). The choice of considering or not $\pi_{z,late}$ impacts on solution quality, too. In general, it has been observed through several tests that good solutions are located on a diagonal band of the (α_2, π_{max}) chart, divided into two main clusters as shown by figure 6.3. Combining high values

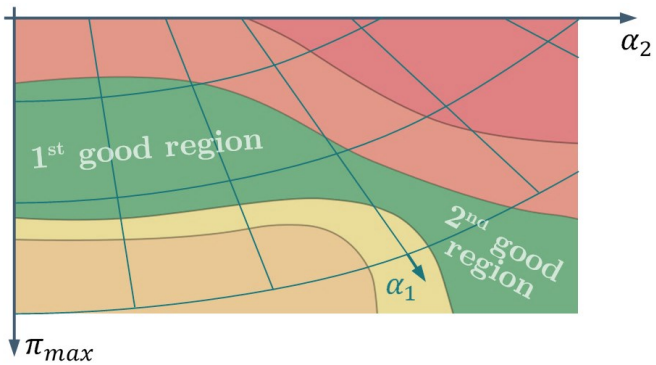


Figure 6.3. Good solution regions on the α_2-π_{max} chart.

of α_2 with low values of π_{max}, and vice versa, does not produce good solutions. Moreover, increasing α_1 has a down-right stretching effect on the chart. The following empirical formula can sometimes find an effective value of α_1 for a good heuristic solution:

$$\alpha_1 = \max\left\{\frac{5.3 \cdot \Delta t}{(\#\mathcal{V})^{2.2}},\ 0.5\right\} \tag{6.8}$$

Δt is the elementary timeframe duration, and $\#\mathcal{V}$ is the cardinality of $\mathcal{V}$, hence the number of available vehicles.

In conclusion, the relationship between MPA parameters and result quality should be further investigated. In fact, a clear dependency pattern was not found; nevertheless, a quick and effective parametrizing procedures can have a remarkable impact on MPA usefulness and operational speed.

CHAPTER 7

MILP and heuristic results

This chapter concerns the side-by-side testing of the *Discrete-Time 1-main-stock MILP* formulation and the heuristic *Minimum Penalty Algorithm*. First, three test instances of growing size are presented. Then, the chapter shows test results for both methods, followed by a critical evaluation and some comments.

7.1 Test instances

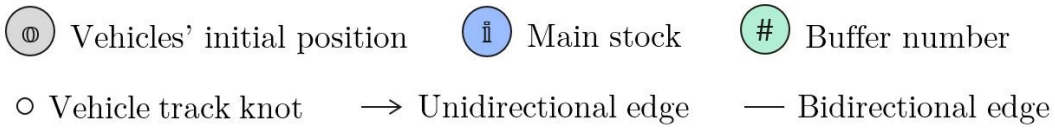
DT-1ms-MILP and MPA were jointly tested on a set of instances based on three fictive workshops. Each workshop plant has a different size; it contains a single main stock, a set of buffers with a unique allowed article reference, and a directional network of vehicle tracks. Articles are either *parts* (raw materials or assembly components), *subassemblies*, or *waste*. Parts are collected at the main stock and delivered at buffers; subassemblies are to be moved from a buffer to another (called twin-buffer); waste is a particular kind of article that is collected at some devoted buffers and must be disposed at the main stock.
For each plant there are nine test instances with different combinations of vehicle fleet size and production intensity. Taking up the notation of chapter 4, production intensity depends on the values of consumptions γ along scheduling period.

	$\#\mathcal{V}_0$		$\#\mathcal{V}_{-1}$		$\#\mathcal{V}_{-2}$	
$\gamma_{\times 1.0}$						
$\gamma_{\times 1.5}$						
$\gamma_{\times 2.0}$						

In the following, the three instances used to test the two methods are shown and explained in detail. Each instance is scheduled over a period T_P of 20 minutes, divided into a set $\hat{T}$ of 20 timeframes of one minute each ($\Delta t = 60$ seconds). The articles allowed in each buffer are identified by minuscule letters. Input data was chosen such that the average base-value of γ is 0.3 load units per minute, absorbed or produced by the production line.

$$\bar{\gamma}_{base} = \frac{\sum_i \sum_t |\gamma_{i,t}|}{\#\mathcal{B} \cdot (\#\hat{T} - 1)} = 0.3 \text{ LU} \tag{7.1}$$

Each test instance has a devoted subparagraph that shows the graph of stocks (*vertices*), vehicle tracks (*edges*) and distances between each pair of knots (*weights*), as well as all the initial and static characteristics of the involved elements. The legend of graph elements is shown below; each buffer has its allowed elements written aside, and the small bidirectional edges that connect each buffer to the rest of the graph have a distance equal to 1.



7.1.1 Instance A - 8 buffers

The first test instance consists of a single main stock and 8 buffers. Figure 7.1 shows the disposition of ground elements in *test instance A* and their connections, while the table below contains their characteristics. $\Delta t = 60$ seconds.

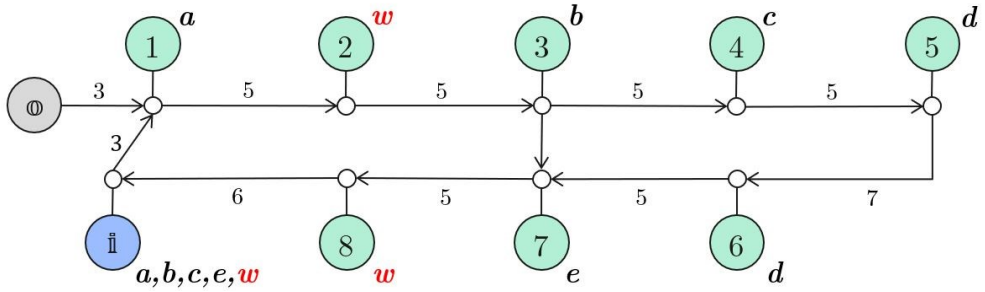


Figure 7.1. Graph of test instance A.

Main stock ⅱ:
$\tau_{ⅱ}[\Delta t] = 1.4$; $\mathcal{A}_{ⅱ} = \{a,b,c,e,w\}$

Table 7.1. Buffer data of instance A.

i	type	$C_{i,0}$[LU]	$\underline{C_i}$[LU]	$\overline{C_i}$[LU]	$\tau_i[\Delta t]$	θ_i	λ_θ[lot/LU]
1	Part	3.5	2	5	.7	a	1
2	Waste	2	0	4	.7	w	1
3	Part	3.5	2	5	.7	b	1
4	Part	3.5	2	5	.7	c	1
5	Subassy	3.5	2	5	.7	d	1
6	Subassy	3.5	2	5	.7	d	1
7	Part	3.5	2	5	.7	e	1
8	Waste	2	0	4	.7	w	1

Table 7.2. Vehicle data of instance A.

k	s_k[m/Δt]	$q^{\theta}_{k,0}$	$t^k_{\mathbb{0}}$[Δt]	notes
1	60	∅	0	
2	60	∅	0	
3	60	∅	0	*Removed on 2nd and 3rd test column*
4	60	∅	0	*Removed on 2nd test column*

7.1.2 Instance B - 14 buffers

The second test instance consists of a single main stock and 14 buffers. Figure 7.2 shows the disposition of ground elements in *test instance B* and their connections, while the table below contains their characteristics. $\Delta t = 60$ seconds.

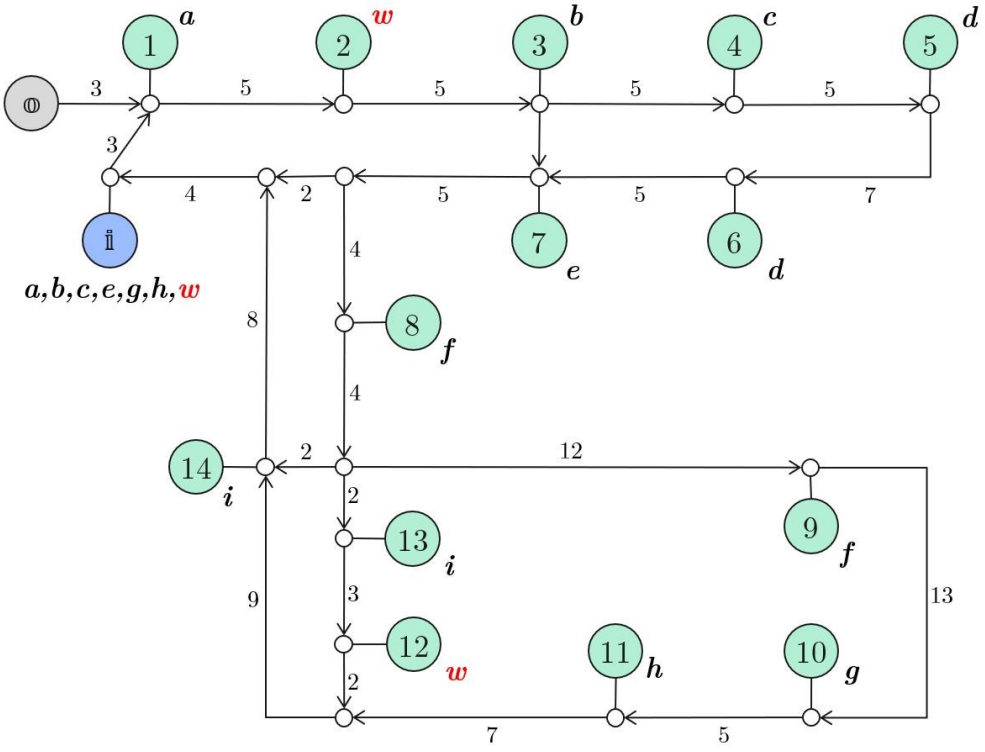


Figure 7.2. Graph of test instance B.

Main stock 1:
$\tau_1[\Delta t] = 1.4$; $\mathcal{A}_1 = \{a,b,c,e,g,h,w\}$

Table 7.3. Buffer data of instance B.

i	type	$C_{i,0}$[LU]	$\underline{C}_i$[LU]	$\overline{C}_i$[LU]	τ_i[Δt]	θ_i	λ_θ[lot/LU]
1	Part	3.5	2	5	.7	a	1
2	Waste	2	0	4	.7	w	1
3	Part	3.5	2	5	.7	b	1
4	Part	3.5	2	5	.7	c	1

5	Subassy	3.5	2	5	.7	*d*	1
6	Subassy	3.5	2	5	.7	*d*	1
7	Part	3.5	2	5	.7	*e*	1
8	Subassy	3.5	2	5	.7	*f*	1
9	Subassy	3.5	2	5	.7	*f*	1
10	Part	3.5	2	5	.7	*g*	1
11	Part	3.5	2	5	.7	*h*	1
12	Waste	2	0	4	.7	*w*	1
13	Subassy	3.5	2	5	.7	*i*	1
14	Subassy	3.5	2	5	.7	*i*	1

Table 7.4. Vehicle data of instance B.

k	s_k[m/Δt]	$q^{\theta}_{k,0}$	$t^{k}_{⌀}$[Δt]	notes
1	60	∅	0	
2	60	∅	0	
3	60	∅	0	
4	60	∅	0	
5	60	∅	0	*Removed on 2nd and 3rd test column*
6	60	∅	0	*Removed on 2nd test column*

7.1.3 Instance C - 20 buffers

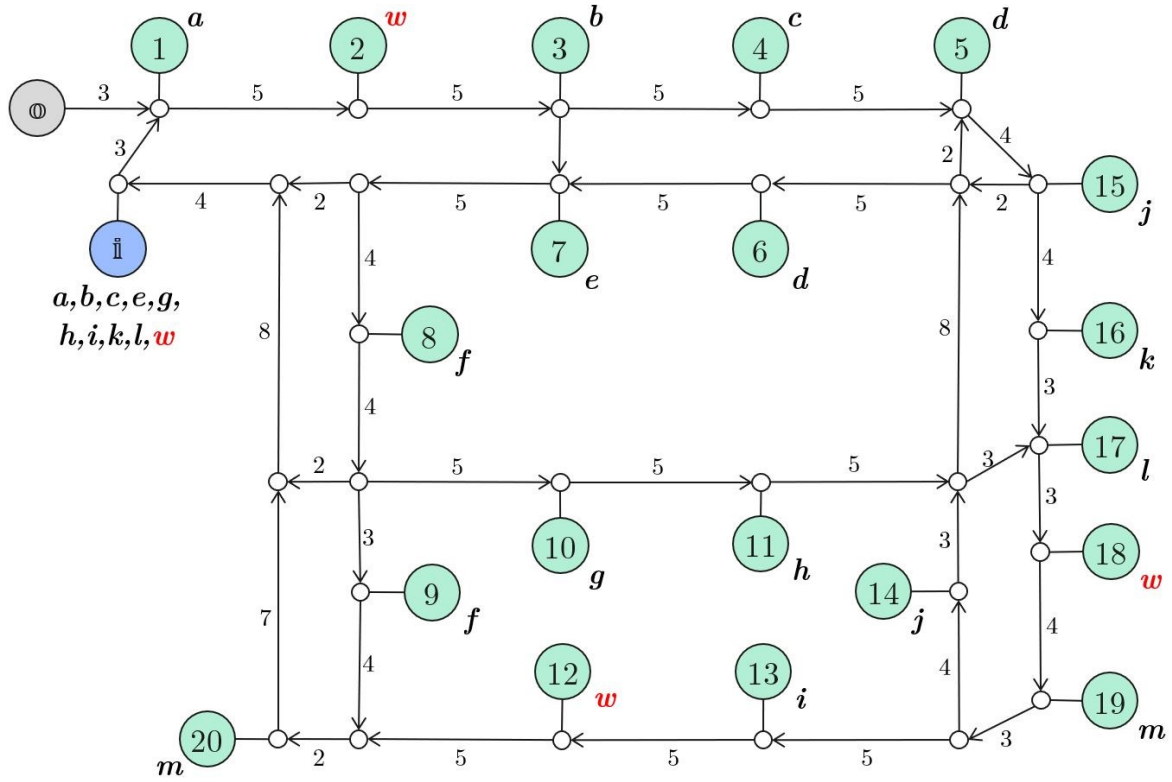


Figure 7.3. Graph of test instance C.

The third and last test instance consists of a single main stock and 20 buffers. Figure 7.3 shows the disposition of ground elements in *test instance C* and their connections, while the table below contains their characteristics. $\Delta t = 60$ seconds.

Main stock 𝕚:
$\tau_{\mathbb{i}}[\Delta t] = 1.4$; $\mathcal{A}_{\mathbb{i}} = \{a,b,c,e,g,h,i,k,l,w\}$

Table 7.5. Buffers of instance C.

i	type	$C_{i,0}$[LU]	$\underline{C_i}$[LU]	$\overline{C_i}$[LU]	$\tau_i[\Delta t]$	θ_i	λ_θ[lot/LU]
1	Part	3.5	2	5	.7	a	1
2	Waste	2	0	4	.7	w	1
3	Part	3.5	2	5	.7	b	1
4	Part	3.5	2	5	.7	c	1
5	Subassy	3.5	2	5	.7	d	1
6	Subassy	3.5	2	5	.7	d	1
7	Part	3.5	2	5	.7	e	1
8	Subassy	3.5	2	5	.7	f	1
9	Subassy	3.5	2	5	.7	f	1
10	Part	3.5	2	5	.7	g	1
11	Part	3.5	2	5	.7	h	1
12	Part	3.5	2	5	.7	i	1
13	Waste	2	0	4	.7	w	1
14	Subassy	3.5	2	5	.7	j	1
15	Subassy	3.5	2	5	.7	j	1
16	Part	3.5	2	5	.7	k	1
17	Part	3.5	2	5	.7	l	1
18	Waste	2	0	4	.7	w	1
19	Subsassy	3.5	2	5	.7	m	1
20	Subassy	3.5	2	5	.7	m	1

Table 7.6. Vehicle data of instance C.

k	s_k[m/Δt]	$q^{\theta}_{k,0}$	$t^k_{\mathbb{o}}[\Delta t]$	notes
1	60	∅	0	
2	60	∅	0	
3	60	∅	0	
4	60	∅	0	
5	60	∅	0	
6	60	∅	0	
7	60	∅	0	*Removed on 2nd and 3rd test column*
8	60	∅	0	*Removed on 2nd test column*

7.1.4 Some considerations about test instances

Before showing test results, here are some considerations about how relevant these instances are to the evaluation of DT-1ms-MILP and MPA. In particular, the *just-in-time* aspect is investigated.
First, some numerical considerations:

- the average gap between the allowed content limits of all buffers is generally around 3.2 LU:

$$\overline{C_{gap}} = \frac{\sum_i (\overline{C}_i - \underline{C}_i)}{\mathrm{card}(\mathcal{B})} \cong 3.2 \text{ LU} \tag{7.2}$$

- as written in (7.1), the average basic consumption value is around 0.3 LU.

From these two values it can be inferred that, in average, each buffer should be visited and totally refilled/emptied every 10-11 minutes ($3.2\,\mathrm{LU}/0.3\,\mathrm{LU}$) to respect content limits. Also, by applying a gamma-multiplier $\gamma_\times = 2$, visit frequency doubles, too, passing from 10-11 minutes to little more than 5 minutes. This quick but meaningful consideration helps to understand the coherence between the proposed test instances and the *just-in-time* approach.

7.2 Test results

For each test instance and both methods, nine test cases were carried out with decreasing fleet size $\#\mathcal{V}$ and increasing consumption multiplier $\gamma_\times$. For practical reasons, MILP solving time was limited to 1200 seconds (20 minutes), which is also equal to the length of the scheduling.

Machine: DELL Vostro 5481, Windows 10 64bit-Professional, Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.9) GHz, RAM 8.00 GB.
Solver: Gurobi Optimizer, version 9.1.2 build v9.1.2rc0 (win64), academic license.
Programming language: Python (MPA coding and Gurobi's API).

7.2.1 Evaluation metrics

The following metrics were used to evaluate instance results. Their calculated values are shown for each test in the tables below.

- $\mathcal{st}$[s]: solving time.
  Concerning the MPA, solving time does not include *best-parameters search*, as it still needs further research and improvement.
  Concerning the DT-1ms-MILP, two solving times are shown. First, the time necessary to get to the best solution, then, in brackets, the time after which a first feasible solution was found.

- $\mathcal{O}bj$[LU]: value of the objective function to *minimize*, i.e., the sum of all the portions of $C_{i,t}$ values exceeding the imposed limits $\overline{C_i}$ and $\underline{C_i}$. The $\mathcal{O}bj$ value in the cell is referred to the best solution the methods could find.
- $b0$[LU]: sum of all $-C_{i,t}$ values such that $C_{i,t} < 0$. As stated above, if $b0 > 0$ the solution is impractical.
- $ar \in [0,1]$: average activity rate of the fleet, calculated as the sum of all travel and load/unloading times, divided by the overall scheduling period duration. This metric quickly points out whether resources are underexploited; however, it does not provide any information about the effectiveness of carried-out activities. In fact, a scheduling in which vehicles roam around in a poorly effective way still has a high ar value.
- $to \in [0,1]$: average item turnover on vehicles. This is the most important value for evaluating scheduling effectiveness; a higher value of to is a univocal sign that the fleet operates in a more effective way. A unitary to value indicates that, at each timeframe, each vehicle of the fleet fully unloads then loads a quantity equal to its capacity. E.g., considering a vehicle k with $Q_k = 4$ LU, $to = 1$ if k unloads 4 LU and loads as many at each timeframe $t \in [1,2,\dots,T]$.

Grey text indicates unfeasible test cases. Concerning DT-1ms-MILP, a case is unfeasible if no solution was found by the solver in the available time. For MPA, a solution is unfeasible if it presents any negative content value.

7.2.2 Results of instance A

Table 7.7

		$\#\mathcal{V}$ (fleet size)					
		4		3		2	
$\gamma_\times$	metrics	MPA	MILP	MPA	MILP	MPA	MILP
1.0	st[s]	0.03	1.3(0)	0.03	1.3(0)	0.01	2.6(0)
	$\mathcal{O}bj$[LU]	0.1	0	0	0	12.6	0
	$b0$[LU]	-	-	-	-	-	-
	ar—	0.64	0.74	0.83	0.57	1.00	0.90
	to—	0.10	0.14	0.13	0.14	0.21	0.17
1.5	st[s]	0.01	3.8(1)	0.03	11.0(1)	0.01	573(2)
	$\mathcal{O}bj$[LU]	3.9	0	29.8	0	148.1	8.55
	$b0$[LU]	-	-	**2.5**	-	**83.5**	-
	ar—	0.78	0.88	0.93	0.70	0.80	1.00
	to—	0.16	0.17	0.18	0.18	0.27	0.27
2.0	st[s]	0.02	17.6(1)	0.01	798(8)	0.01	**1200**
	$\mathcal{O}bj$[LU]	54.4	0	134.5	4.7	339.3	-
	$b0$[LU]	**0.6**	-	**54.3**	-	**229.5**	-
	ar—	0.84	0.95	0.82	1.00	0.85	-
	to—	0.23	0.20	0.22	0.25	0.28	-

7.2.3 Results of instance B

Table 7.8

		$\#\mathcal{V}$ (fleet size)					
		6		5		4	
$\gamma_\times$	metrics	MPA	MILP	MPA	MILP	MPA	MILP
1.0	st[s]	0.08	4.41(3)	0.07	4.69(2)	0.03	6.97(2)
	Obj[LU]	8.0	0	1.6	0	14.9	0
	$b0$[LU]	-	-	-	-	**0.6**	-
	ar—	0.63	0.62	0.77	0.68	0.85	0.84
	to—	0.11	0.12	0.14	0.13	0.16	0.16
1.5	st[s]	0.06	26.9(14)	0.05	51,9(6)	0.03	322(7)
	Obj[LU]	74.3	0	156.6	0	255.1	2.22
	$b0$[LU]	**19.0**	-	**55.9**	-	**115.6**	-
	ar—	0.86	0.75	0.84	0.92	0.83	0.98
	to—	0.17	0.16	0.17	0.20	0.17	0.23
2.0	st[s]	0.05	545(309)	0.04	**1200**	0.03	**1200**
	Obj[LU]	285.8	0.4	449.0	-	530.3	-
	$b0$[LU]	**106.6**	-	**200.1**	-	**300.7**	-
	ar—	0.78	0.95	0.82	-	0.65	-
	to—	0.18	0.22	0.18	-	0.17	-

7.2.4 Results of instance C

Table 7.9

		$\#\mathcal{V}$ (fleet size)					
		8		7		6	
$\gamma_\times$	metrics	MPA	MILP	MPA	MILP	MPA	MILP
1.0	st[s]	0.14	25.1(11)	0.11	10.7(6)	0.09	13.7(7)
	Obj[LU]	14.3	0	4.1	0	22.3	0
	$b0$[LU]	-	-	-	-	**2.1**	-
	ar—	0.71	0.71	0.69	0.81	0.82	0.82
	to—	0.11	0.13	0.12	0.15	0.15	0.16
1.5	st[s]	0.18	134(74)	0.09	100(28)	0.07	1109(162)
	Obj[LU]	41.3	0	241.4	0	314.9	0.15
	$b0$[LU]	**0.3**	-	**107.7**	-	**131.9**	-
	ar—	0.88	0.90	0.75	0.94	0.84	0.95
	to—	0.17	0.19	0.17	0.20	0.16	0.21
2.0	st[s]	0.14	**1200**	0.13	**1200**	0.07	**1200**
	Obj[LU]	606.7	-	673.8	-	749.8	-
	$b0$[LU]	**261.0**	-	**306.8**	-	**370.2**	-
	ar—	0.70	-	0.72	-	0.69	-
	to—	0.16	-	0.15	-	0.16	-

7.3 Results evaluation and comments

In the three tables above, the first glance at each cell is to determine if a feasible solution was found. Unfeasible test cases can help to determine the capability limits of each method in terms of problem size and criticality of resources.
If a test case was feasible, the first relevant metric is solving time st. It gives an indication about problem complexity, especially for MILP tests. Other than that, the most important metric of each test is the turnover to. In fact, it can be considered as the scheduling efficiency index, since it is proportional to the number of articles picked-up and delivered.
In the following, a specific test cell is identified with the wording: <instance>(<$\#\mathcal{V}$>, <$\gamma_{\times}$>, <method>). E.g., **A(7, 1.5, MPA)** indicates the result of instance A, tested with a fleet size $\#\mathcal{V} = 7$, a gamma-multiplier $\gamma_{\times} = 1.5$, and solved with MPA.

Comment about Minimum Penalty Algorithm:

- It could quickly find feasible solutions for all the proposed instances as long as production intensity was limited and the number of vehicles was reasonable (neither too high, nor too low).
- The maximum activity rate ar is around 85%. Exceptional peak of 100% at A(2, 1.0, MPA).
- Solution turnover to rarely exceeds 17%. As for ar, the limits on the value of to are methodological, as they are mainly due to the reduced visibility horizon of MPA. Exceptional peak value reached at A(2, 1.0, MPA), where $to = 21\%$.
- The only advantage of this heuristic method compared to DT-1ms-MILP is the smaller solving time, which is of the order of 0.1 seconds (excluding *best-parameters search*).
- On the other hand, a major disadvantage of heuristics is precisely its sensitivity to parameterization and the still unclear correlation between parameter values and the quality of results.

Comment about Discrete-Time 1-mainstock MILP:

- It could find a solution in almost all cases, with a resolution time that rises very quickly as production intensifies and fleet size decreases. In general, it can be inferred that the factor that makes a problem harder is logistic resource shortage (too much production and/or too few vehicles) rather than global instance size (8 vs. 14 vs. 20 buffers). Even for medium-sized problems (20 buffers), solving time remains very reasonable if the fleet is sufficiently large.
- Since DT-1ms-MILP has a visibility horizon that coincides with the whole scheduling period, the activity rate ar of vehicles can potentially reach 100%. $ar = 100\%$ at A(3, 2.0, MILP).
- For the same reason, the turnover to can reach highest values in MILP solutions. to max was found at A(3, 2.0, MILP), and is equal to 26%.

In practice, assuming a capacity $Q_k = 4$ for all vehicles, a value of $\mathit{to} = 26\%$ means that, on average, each vehicle exchanges 2.08 LU of its content at every 1-minute timeframe.

- Given the discrete-time approximations, DT-1ms-MILP's limitations are rather related to the increasing rigidity of the problem (more articles to move, less vehicles to do it) which determines a steep growth of solving time.

In general, it can be concluded that DT-1ms-MILP formulation programmed and solved with Gurobi Optimizer shows significantly better performance with respect to the heuristic *Minimum Penalty Algorithm*. When the heuristics was able to find a feasible solution, MILP model also found one, of equal or better quality, and in the order of a few tens of seconds. However, if a high solving speed is required, the MPA can schedule the fleet in less than a second and, in some cases, it proved to give a good and feasible solution up to 17 seconds ahead of DT-1ms-MILP.

Table 7.10. Resume and comparison of results.

	MPA	DT-1ms-MILP	Variation
Solving time	**0.01÷0.2 s**	5÷600 s	+500÷3000%
Max activity rate (excluding except. peaks)	85%	**100%**	+15%
Max turnover	17%	**26%**	+9%
Requires parametrization	Yes	**No**	
Hardest problem solved	C(7, 1, MPA)	**C(6, 1.5, MILP)**	–1 vehicle $+0.5\gamma_{\times}$

7.4 Period decomposition

The instances shown in the paragraphs above present a planning period of 20 minutes. Nevertheless, the actual planning system should be able to schedule longer periods, of the order of 3-4 hours. Thus, it is necessary to define a split strategy to divide the planning in subperiods and solve with a sliding horizon approach. Subperiods duration is defined upstream, as well as the portion of each subperiod that is overlapped to the previous one. The subperiods are scheduled one by one, and the results of the previous one are the input data for the next, as shown in figure 7.4. Subperiods overlapping is necessary to give a longer horizon of visibility to MILP methods. For example, optimization is done by knowing what happens in the next 20 minutes, but recorded operations are only those planned for the first 15. The remaining 5 minutes will be rescheduled with the next subperiod. This allows a more fluid transition between a subperiod solution and the next.

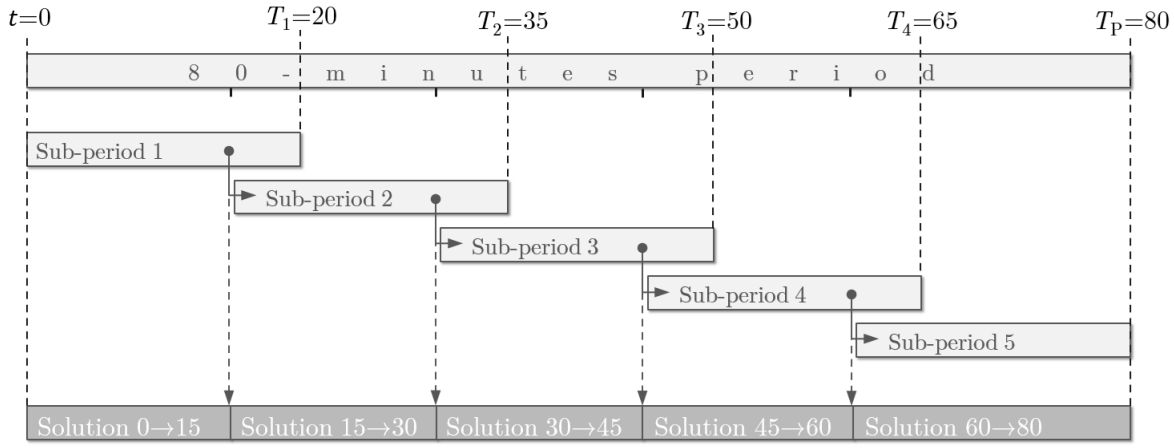


Figure 7.4. Example of a problem decomposed in subperiods.

7.5 Visual representation of the solution

The outcome of problem resolution and post-processing is a list of instructions for vehicles. However, a list of tasks is not the best form for an easy and detailed analysis of the outcome. The charts presented below were developed during the internship to visualize the results in a more understandable and significant way. They contain the following information:

- Usage rate of vehicles,
- Travelled distances,
- Buffer content evolution along the period,
- Scheduling of each vehicle (resource diagram).

Figures 7.4 to 7.11 contain these charts for test instance A with $\#\mathcal{V} = 3$, $\gamma_{\times} = 1.5$, and 15 timeframes, solved with both MPA and DT-1ms-MILP.

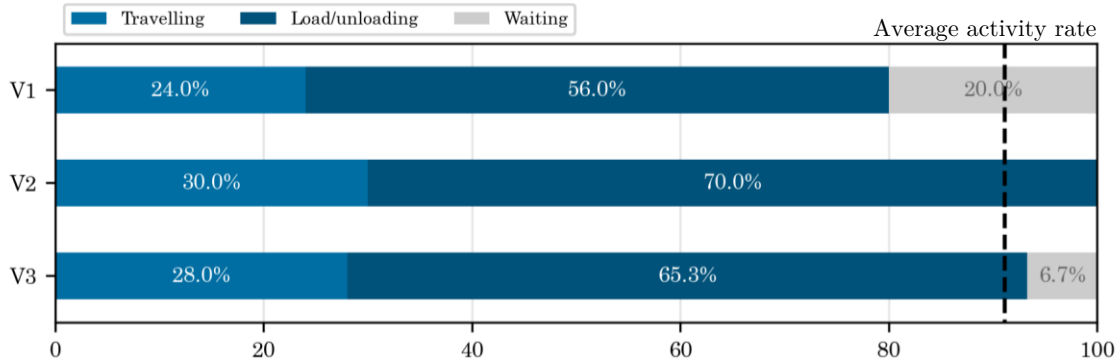


Figure 7.5. Usage rates, instance A (8b|3v|15t), MPA.

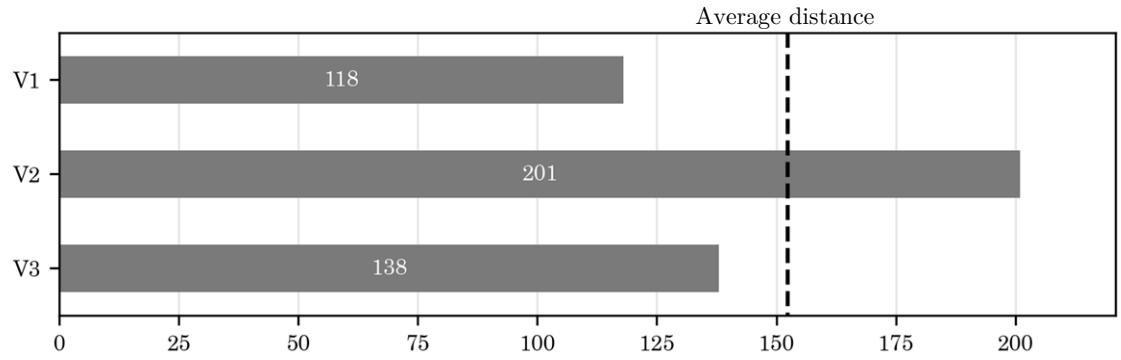


Figure 7.6. Travelled distances, instance A (8b|3v|15t), MPA.

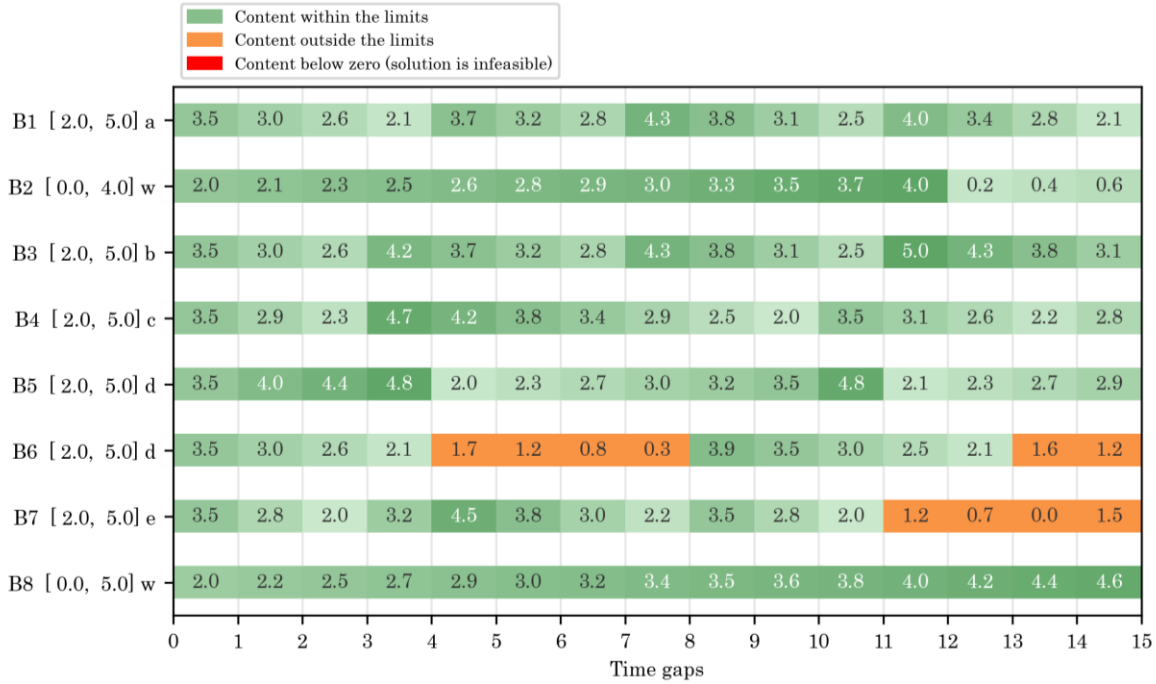


Figure 7.7. Buffer content evolution, instance A (8b|3v|15t), MPA.[1]

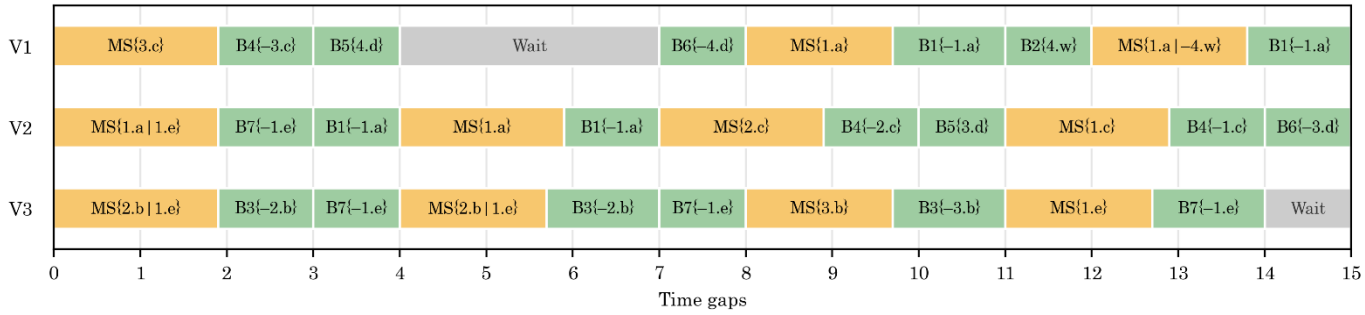


Figure 7.8. Task scheduling, instance A (8b|3v|15t), MPA.[2]

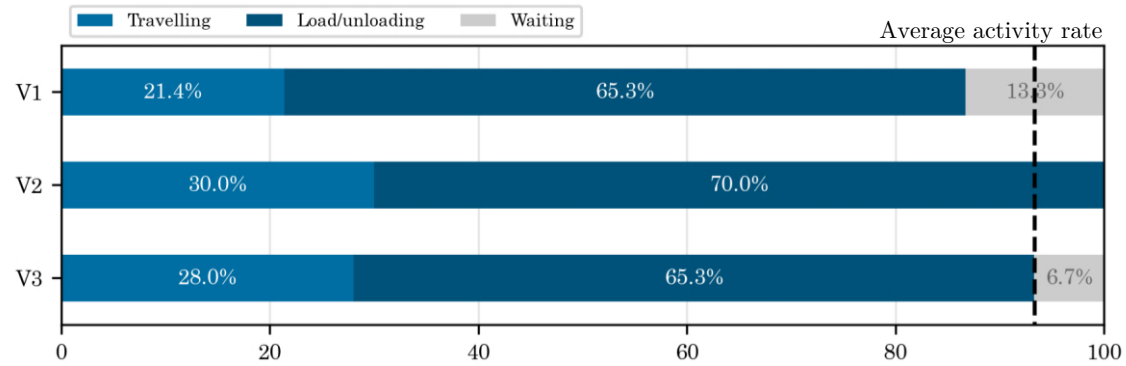


Figure 7.9. Usage rates, instance A (8b|3v|15t), DT-1ms-MILP.

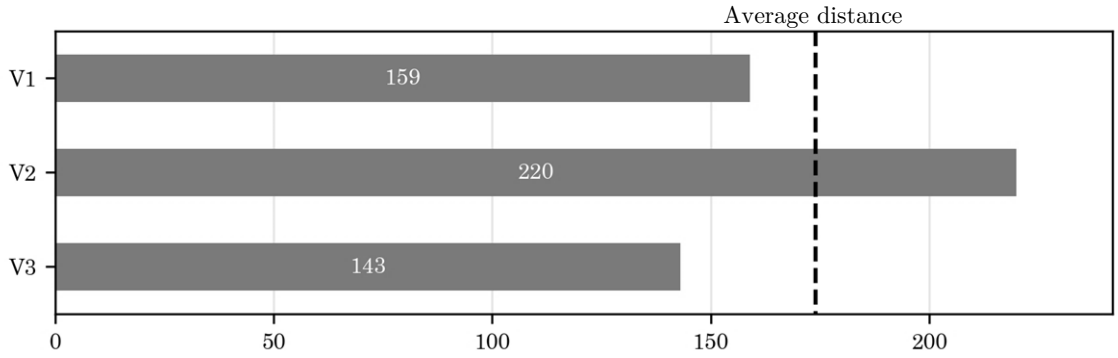


Figure 7.10. Travelled distances, instance A (8b|3v|15t), DT-1ms-MILP.

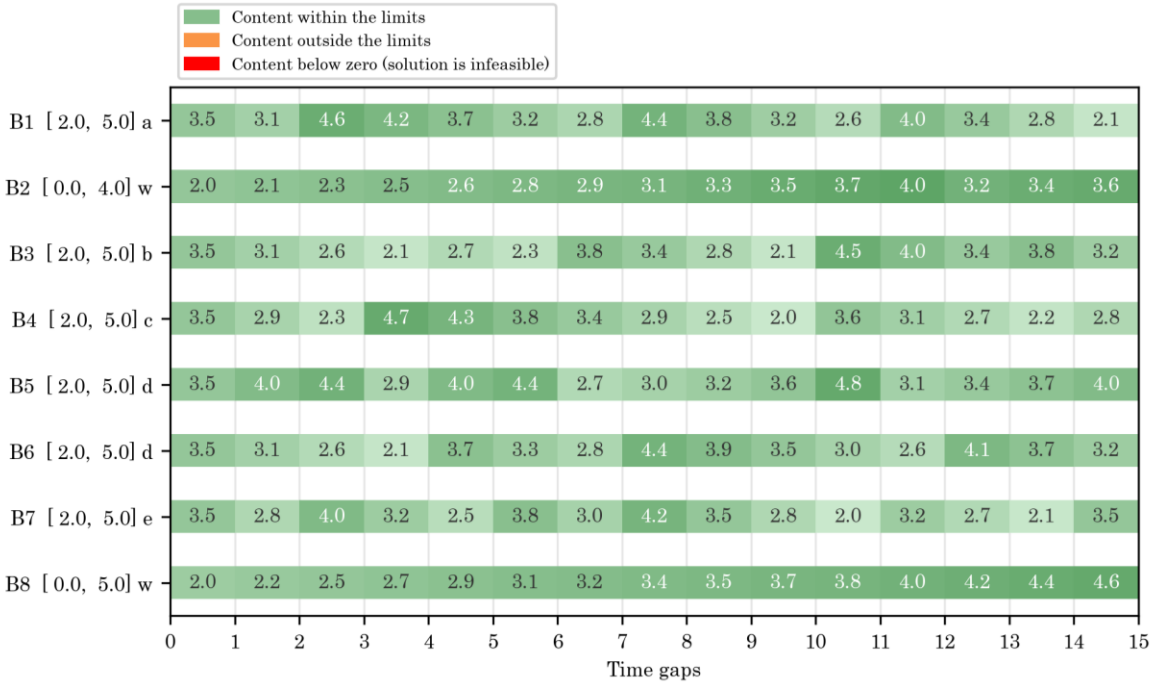


Figure 7.11. Buffer content evolution, instance A (8b|3v|15t), DT-1ms-MILP.[1]

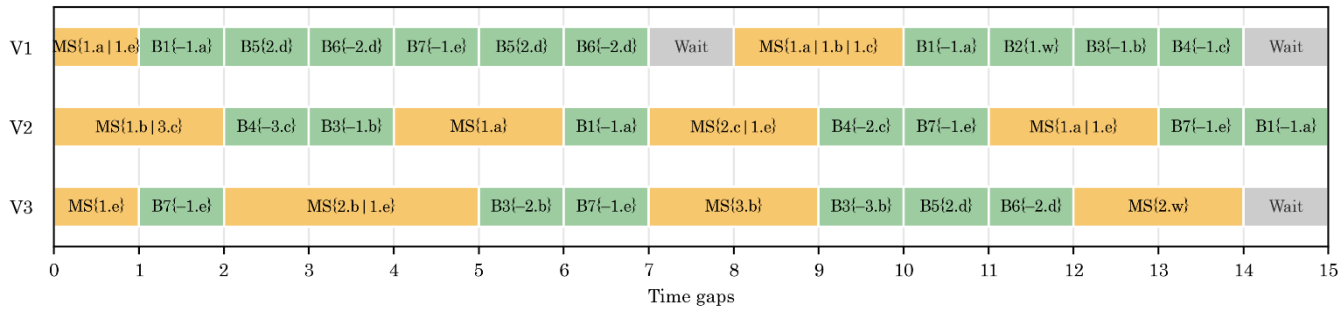


Figure 7.12. Task scheduling, instance A (8b|3v|15t), DT-1ms-MILP.[2]

[1] Format of y-labels in figures 7.7 and 7.11:
<buffer> [<lower bound>, <upper bound>] <supported article>
The number inside each cell indicates buffer content at the beginning of timeframe.

[2] Format of cell-labels in figures 7.8 and 7.12:

<task destination> { <article1>.<loaded lots> | <article2>.<loaded lots> | … }

MS: main stock.

Conclusion

The thesis ends with a final comment divided into three main parts. First, the compliance of developed and tested methods with respect to application cases is verified, especially for the *indoor logistic operation manager*. Scheduling objectives are analysed and checked one by one. Second, a quantitative evaluation of 'how just-in-time' these methods are is presented. Last, a list of potential future works closes the thesis. This list contains all the complementary aspects that could not be treated during the 6-months internship at LAAS-CNRS.

Compliance of solving methods with application cases

Objective	**Compliance**
1. Logistic support to guarantee production operability.	Direct mathematical objective of the developed methods. Granted by both MPA and DT-1ms-MILP if logistic resources are sufficient.
2. Minimization of on-ground inventory (stock reduction is a major aim of the *just-in-time* approach). **3.** Minimization of logistic resources necessary to effectively support the production line.	These two objectives are concurrently satisfied prior to direct method resolution. Once the lower and upper limits of each buffer are fixed, the minimum fleet size necessary to support production activities is searched off-line by analysing the results of several fictitious and representative problem instances. In accordance with a robust programming approach, it would be appropriate to consider the worst case which could reasonably occur. For example, the fleet shall be dimensioned to also cope an unforeseen event or a moderate increase of production intensity. A slightly over-dimensioned fleet has also some advantages: it makes it possible to have a

	surplus of inactive vehicles during nominal operations that can be employed to process and satisfy punctual and explicit needs (e.g., a production operator that asks for a specific item for some unforeseen reason). In general, less capacious buffers require a larger vehicle fleet to grant system operability, and vice versa.
4. Reactivity to unforeseen events	The responsiveness of the system to unforeseen events is granted by the solving speed of methods, especially the minimum penalty algorithm (MPA). This grants an important dynamic aspect to the support system, that can quickly react to contingencies by selectively adapting or rescheduling operations.

Just-in-time evaluation

The following calculus aims at proving that the obtained results are compliant with *just-in-time* principles by making it possible to supply the production line with low inventory levels.
Let us consider the hardest instance solved with DT-1ms-MILP. It is the instance C presented in chapter 7, with 20 buffers, 6 vehicles, and an average absolute consumption $\bar{\gamma} = 0.45$ LU per minute. Vehicles have a capacity $Q_k = 4$ LU, and the average allowed content gap of all buffers is equal to 3.2 LU (as calculated in paragraph 7.1.4). The average survival time of each buffer can be calculated as:

$$\text{av. survival time} = \frac{\text{av. allowed content gap}}{\text{av. absolute consumption}} = \frac{3.2\ \text{LU}}{0.45\ \text{LU}} \cong 7.1\ \text{min} \qquad \text{(C.1)}$$

Moreover, the number of available vehicles per buffer is equal to:

$$\frac{\#\mathcal{V}}{\#\mathcal{B}} = \frac{6}{20} = 0.3\,\text{vehicles/buffer} \qquad \text{(C.2)}$$

Hence, the average frequency at which each vehicle should visit a buffer and completely refill or empty it is calculated as the product of the two values above:

$$\text{av. fill-empty frequency} = 7.1\ \text{min} \cdot 0.3\,\text{vehicles/buffer} = 2.1\,\frac{\text{min} \cdot \text{vehicles}}{\text{buffer}} \qquad \text{(C.3)}$$

To summarize, the hardest instance for which DT-1ms-MILP could find a solution presents the following quantitative characteristics:

- 20 buffers and 6 vehicles,

- an average buffer content gap of 3.2 load units, equal to 80% of any vehicle capacity (4 LU),
- an average buffer survival time of 7.1 minutes,
- a fleet of vehicles in which every unit completely refills or empties a buffer every 2.1 minutes.

Similarly, the hardest instance solved with MPA (20 buffers, 7 vehicles, $\bar{\gamma} = 0.3$ LU/min) has an average buffer survival time of 10.7 minutes, and an average fill-empty frequency of 3.7 minutes.

In both cases, but especially for DT-1ms-MILP, it can be concluded that the system operates in accordance with *just-in-time* principles.

Future works

Many complementary aspects were left aside during the internship. They are enumerated in this paragraph to give some further development prompts.

1. Adapt the application cases and mathematical models to be more compliant with real instances; review the work of this thesis and validate the assumptions regarding core aspects and data handling. Applying the developed methods to real-life instances could provide some important information to improve models and algorithms.

2. Search for other evaluation metrics by focusing on more pertinent key aspects of a real workshop (distances travelled by vehicles, average rate of deliveries and quantities exchanged, adaptability and reactivity demonstrated in simulations, etc.).

3. Concerning the *Minimum Penalty Algorithm*, investigate the relationship between problem data, parameterization, and results, and search for quick and effective parameterization methods.
Also, look for a more effective penalty function, possibly with a more direct dependency with problem primary objective.

4. Design and develop a metaheuristic to improve the solutions given by the MPA. For example, a *genetic algorithm* or a *swarm optimization algorithm* to improve an existing MPA solution, or even making feasible a solution of MPA which is not. A deterministic component could target unfeasible scheduling parts, while a second stochastic component modifies them with crossover and mutation operations. Moreover, unfeasible test cases show a lower activity rate compared to feasible ones (see table at paragraphs 7.2.2 to 7.2.4). A metaheuristic algorithm could also aim at adding new *do*-tasks to improve the overall vehicle exploitation.

Considering search space complexity and the many components involved, a population-based metaheuristic would be more suitable to effectively solve the problem. In fact, population-based algorithms make a deeper neighbourhood search with respect to trajectory-based ones. As such, they often grant a quicker convergence in presence of complex search spaces with wide optimality regions.

References

[**1**] J. P. Womack, D. T. Jones, (2003). Lean Thinking: Banish Waste and Create Wealth in Your Corporation. *Simon and Schuster*, p. 10.

[**2**] The Nature of Mathematical Programming, (2014, March 5). *Published on Wayback Machine, Mathematical Programming Glossary, INFORMS Computing Society.*

[**3**] Wikipedia contributors, (2021, October 25). Mathematical optimization. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**4**] R. Bellman, (1957). Dynamic programming and the numerical solution of variational problems. *Operations Research*, 5, pp. 277-288.

[**5**] Wikipedia contributors, (2021, October 22). Linear programming. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**6**] LP: Canonical form and standard form, (2021). *Published on complex-systems-ai.com.*

[**7**] Bradley, Hax, and Magnanti, (1977). Applied Mathematical Programming. Addison-Wesley. *Published on web.mit.edu.*

[**8**] Wikipedia contributors, (2021, September 18). Simplex algorithm. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**9**] Murty, G. Katta, (1983). Linear programming. *New York: John Wiley & Sons*, pp. xix+482.

[**10**] Wikipedia contributors, (2021, July 6). Cutting-plane method. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**11**] P. C. Gilmore, R. E. Gomory, (1961). A linear programming approach to the cutting stock problem. *Operations Research*, 9, pp. 849–859.

[**12**] P. C. Gilmore, R. E. Gomory, (1963). A linear programming approach to the cutting stock problem-Part II. *Operations Research*, 11, pp. 863-888.

[**13**] G. Nemhauser, L. Wolsey, (1988). Integer and Combinatorial Optimization. *Wiley Interscience.*

[**14**] H. Toussaint, (2013). Introduction au Branch Cut and Price et au solveur SCIP (Solving Constraint Integer Programs). *Rapport de recherche LIMOS/RR-13-07.*

[**15**] Wikipedia contributors, (2020, November 24). Column generation. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**16**] Interpreting LP Solutions - Reduced Cost. *Published on Courses.psu.edu*, retrieved in October 2021. (https://www.courses.psu.edu/for/for466w_mem14/Ch11/HTML/Sec4/ch11sec4_RC.htm)

[**17**] J. Chen, (2021). Heuristics. *Published on Investopedia*, retrieved in October 2021. (https://www.investopedia.com/terms/h/heuristics.asp)

[**18**] J. Pearl, (1984). Heuristics: intelligent search strategies for computer problem solving. *United States: Addison-Wesley Pub. Co.*, p. 3.

[**19**] S. Consoli, (2006). Combinatorial Optimization and Metaheuristics. *Operational research report: combinatorial optimization and metaheuristics*, TR/01/06, p. 7.

[**20**] X. Yang, (2011). Metaheuristic Optimization. *Published on Scholarpedia*, 8(9), p. 11472 (http://www.scholarpedia.org/article/Metaheuristic_Optimization)

[**21**] C. Blum and A. Roli, (2001). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, pp. 274-284.

[**22**] M. Dorigo, M. Birattari and T. Stützle, (2006). Ant Colony Optimization. *IEEE Computational Intelligence Magazine*, pp. 29-31.

[**23**] J. Kennedy and R. C. Eberhart, (1995). Particle swarm optimization. *IEEE International Conference on Neural Networks*, Piscataway, NJ. pp. 1942–1948.

[**24**] Wikipedia contributors, (2021, October 19). Particle swarm optimization. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**25**] X. Z. Gao, V. Govindasamy, H. Xu, X. Wang, and K. Zenger, (2015, March 21). Harmony Search Method: Theory and Applications, *Hindawi Publishing Corporation*, v.2015, ID.258491.

[**26**] M. G. C. Resende and C. C. Ribeiro, (2008). GRASP. *AT&T Labs Research Technical Report.*

[**27**] T. A. Feo and M. G. C. Resende, (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8, pp. 67–71.

[**28**] Wikipedia contributors, (2021, October 28). Travelling salesman problem. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**29**] Wikipedia contributors, (2021, May 28). NP-hardness. *Wikipedia, The Free Encyclopedia*, retrieved in October 2021.

[**30**] M. Held and R. M. Karp, (1962). A Dynamic Programming Approach to Sequencing Problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1), pp. 196–210.

[**31**] P. Toth, and D. Vigo, (2014). Vehicle Routing. *Mathematical Optimization Society-SIAM.*

[**32**] G. B. Dantzig and J. H. Ramser, (1959). The Truck Dispatching Problem. *Management Science*, 6(1), pp. 80–91.

[**33**] S. Mancini, (2014). Time dependent travel speed vehicle routing and scheduling on a real road network: the case of Torino. *Transportation Research Procedia 3.*

[**34**] J. Dethloff, (2001). Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum*, 23(1), pp. 79-96.

[**35**] A. M. Campbell, L. W. Clarke and M. W. P. Savelsbergh, (2002). The Vehicle Routing Problem. *SIAM*, pp.309-330.

[**36**] L. C. Coelho, J. Cordeau, G. Laporte, (2012). Thirty Years of Inventory-Routing. *CIRRELT.*

[**37**] H. Andersson, A. Hoff, M. Christiansen, G. Hasle, A. Løkketangen, (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research*, 37(9), pp. 1516-1536.

[**38**] P. Chandra, (1993). A Dynamic Distribution Model with Warehouse and Customer Replenishment Requirements. *Journal of the Operational Research Society*, 44(7), pp. 681-692.

[**39**] A. Corberán, M. Laguna, E. Fernandez, R. Marti, (2002). Heuristic solutions to the problem of routing school buses with multiple objectives. *Journal of the Operational Research Society*, 53, pp. 427-435.

[**40**] L. Y. O. Li, Z. Fu, (2002). The school bus routing problem: A case study. *Journal of the Operational Research Society*, 53, pp. 552-558.

[**41**] I. Masudin, Z. Maghfur, Mudrifah, F. Zulfikarijah, and D. P. Restuputri, (2019). Multi-Product Multi-Vehicle Inventory Routing Problem With Mixed Integer Linear Programming. *International Conference on Industrial Engineering and Operations Management.*

[**42**] L. C. Coelho, G. Laporte, (2012). A Branch-and-Cut Algorithm for the Multi-Product Multi-Vehicle Inventory-Routing Problem. *CIRRELT*, 2012-53.

[**43**] N. Guo, B. Qian, R. Hu, H. P. Jin, and F. H. Xiang, (2020). A Hybrid Ant Colony Optimization Algorithm for Multi-Compartment Vehicle Routing Problem. *Hindawi*, v.2020, ID.8839526.

[**44**] S. Ohmori, K. Yoshimoto, (2021). Multi-product multi-vehicle inventory routing problem with vehicle compatibility and site dependency: A case study in the restaurant chain industry. *Uncertain Supply Chain Management*, 2021(9), pp. 351–362.

[**45**] C. Archetti, M. Christiansen, M. G. Speranza, (2017). Inventory Routing with Pickups and Deliveries. *European Journal of Operational Research.*

[**46**] I. Hssini, N. Meskens, F. Riane, (2016). Blood Products Inventory Pickup and Delivery Problem under Time Windows Constraints. *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems (ICORES 2016)*, pp. 349-356.

[**47**] Glossary:Transshipment, (2021, July 7). *Published on ec.europa.eu, Eurostat statistics explained.*

[**48**] L. C. Coelho, J. Cordeau, G. Laporte, (2012). The inventory routing problem with transshipment. *Computers & Operations Research*, 39(11), pp. 2537-2548.

[**49**] S. M. J. Mirzapour Al-e-hashem, Y. Rekik, (2014). Multi-product multi-period Inventory Routing Problem with a transshipment option: A green approach. *International Journal of Production Economics*, 157, pp. 80-88.