
UNIVERSITÀ
DI SIENA
1240

DEPARTMENT OF INFORMATION ENGINEERING AND MATHEMATICS

---

PHD IN INFORMATION ENGINEERING AND SCIENCE
- XXXV CYCLE -

# Improving Decision Making in Real-world Applications by Solving Combinatorial Optimization Problems

PhD Dissertation by Dott. Mario Benini

**Advisor:**

**Prof. Paolo Detti**

**Academic Discipline:**

**MAT/09 - Operations Research**

AY 2021/2022

# Contents

# Introduction

Growth and development are at the basis of our society. Companies and organizations in all industries must grow. It's an imperative that drives all these entities to create new products and services, enter new regions, move into new businesses. The existence of competition among all these actors leads them to pursue a continuous improvement of their operations in order to increase their profits while guaranteeing, or improving, the quality of their products and services, and being subject to the multitude of constraints deriving from the interconnection with authorities and other organizations. In other words, in order to be competitive. As they improve, they inevitably become more complex. Growth creates complexity and complexity may kill growth, if not properly managed. More and more business challenges arise every day and with them the need to make complex decisions involving many factors. It is not possible to optimally make such decisions without proper tools and analysis and that's why Decision Science is becoming more and more important. In general, Decision Science is the collection of quantitative techniques used to inform decision-making. It includes decision analysis, risk analysis, cost-benefit and cost-effectiveness analysis, simulation modeling, as well as microeconomics, statistical inference, management control, computer science and operations research. By focusing on decisions as the unit of analysis, decision science provides a unique framework to face the complexity of the decision-making process in real-world problems.

The motivation for this work is to study complex real-world scenarios and provide tools that can actually improve decision-making in those problems. To do so, we mainly adopt techniques from the fields of Operations Research and Combinatorial Optimization. As it is well known, operations research is a scientific method for providing executive departments (or decision makers) with a quantitative basis for decisions. Its object is, by the analysis of past operations and the data available, to find means of improving future operations. Operations research is important because it creates implementable solutions to complex business challenges. It uses data to create information, which can then be used as insights to improve results and make better decisions about the future of the business. Combinatorial Optimization, in particular, is a subfield of mathematical optimization that consists of finding an optimal solution from a finite set of solutions, where the set of feasible solutions is discrete or can be reduced to a discrete set. This field is strictly interconnected with Linear programming and Integer Programming. In typical combinatorial optimization problems, exhaustive search is not tractable, and so specialized algorithms that quickly rule out large parts of the search space or approximation algorithms must be resorted to instead. Techniques from the field of operations research and combinatorial optimization are very powerful because they can be adopted in a variety of industries and scenarios and want to find a solution to the question "what is the best decision?". These decisions may be very complex to make and they can be of different kinds. Applications of combinatorial optimization include, but are not limited to:

- Logistics.
- Supply chain optimization.
- Developing the best airline network of spokes and destinations.

- Deciding which taxis in a fleet to route to pick up fares.
- Determining the optimal way to deliver packages.
- Allocating jobs to people optimally.
- Designing water distribution networks.
- Earth science problems (e.g. reservoir flow-rates).

Decision making in these application may be so complex that it is generally easier to build methods able to make such decisions for us. The development of solution approaches for combinatorial optimization problems usually requires different phases: the analysis of the problem and the data, the establishment of a model of the problem and the design and implementation of efficient methods able to help taking the best decisions.

In the following section, the organization of this dissertation is presented, together with a brief description of the problems addressed. Most of the material in this work is based on research that has been published or submitted to publication. In the last part of this introduction, the list of publications related to the work presented in the dissertation is reported.

**Contributions and Organization**

In this dissertation we focus on three real-world applications from different industries that can be modeled as combinatorial optimization problems and address them with operations research techniques. The dissertation is divided in chapters, each of which is related to a different topic.

In chapter 1, a problem concerning the transportation of biological samples from draw centers to a main laboratory for analysis is presented. The problem arises from a healthcare application in

Bologna, Italy, where the healthcare authority decided to centralize the analysis of all biological samples of the area to a main laboratory, in order to exploit economies of scales and reduce the costs for samples' analysis. Of course, such an improvement goal also created a new complex problem: all the samples must be transported from draw centers to the main lab. A fleet of vehicle is available for the transportation and must collect the samples from draw centers during given times of the day and deliver them within a certain time, since samples are perishable. Vehicles can also exploit the existence of dedicated centers that can extend the lifespan of the samples and where samples can be transferred from one vehicle to another. It is clear from this brief description how hard it could be to decide which is the routing of all the vehicles which minimizes the traveling costs while delivering all samples on time. For this problem we developed different mixed integer linear programming models, metaheuristic algorithms, and grouping policies for the samples that are able to tackle the complexity of the problem and improve routing decisions. All methods have been tested through an extensive computational campaign using real-world data, showing the effectiveness of the proposed approaches.

In Chapter 2 a problem related to the agricultural industry is presented. The problem arises from a real-world application in Italy and it is that of planning the use of the available land of a farm for a given number of years, given a set of crops that can be grown. The objective is to maximize the farmer's profit, but the farmer is subject to several rules both from an agronomic and from a regulation point of view. In fact, many constraints exist regarding agronomic principles, such as maximum replanting, botanical family constraints and crop rotation issues. One of the goals of this work is indeed that of evaluating the risks and benefits of following or not the best practices regarding crop rotation issues in the Mediterranean pedo-climatic context. Furthermore, we want

to evaluate the effectiveness of public and private initiatives regarding sustainable agriculture. In fact, it is more and more important nowadays to face these challenges in the food supply chain, which is one of the most discussed industries when it comes to sustainability. In particular, we analyze two different initiatives, namely the Common Agricultural Policy by the European Union and "La Carta del Mulino" by Barilla Group S.p.A.. Both initiatives introduce economic incentives for the farmers following virtuous behaviors from a sustainability point of view. Practically, these behaviors are constraints increasing the complexity of the problem and the difficulty in the decision-making process. For this problem, we will give a formal characterization and study its complexity, also analyzing special cases. We will also present a network-flow based model to solve a special case of the problem and integer linear programming models developed to solve three variants accounting for different sustainability scenarios. Real-world data from 23 Italian farms were used in an extensive computational campaign. The analysis of the results shows that the models can be helpful tools for farmers to plan their production and for authorities to evaluate the effectiveness (and efficiency) of their sustainability initiatives.

In Chapter 3 we discuss a problem concerning the sequencing of unreliable jobs on parallel machines. Even if the problem is not taken from a specific application, it may have several applications in real-world scenarios, such as in manufacturing and planning of complex computations on multi-processors computers. In this problem, we have $n$ unreliable jobs providing a reward when successfully completed, but each job has a probability of not being carried out. We have $m$ parallel identical machines at our disposal, and we want to schedule the jobs on the machines in order to maximize the total expected reward. To increase the probability of completing the jobs, we create $m$ copies of each job and schedule each copy on a different machine. For this problem, we will present

a complexity analysis showing that the problem is NP-complete for two machines. For the problem with two machines, we derived some theoretical properties and developed a quadratic integer programming model, a tabu search algorithm, and an upper bound based on the Three-Dimensional Assignment problem. A computational campaign on different sets of instances shows that the tabu search outperforms the model. Then we focused on the general case with $m$ machines. In particular, we developed several heuristics and proved some theoretical results, including the worst case performance guarantee of two heuristics. We also devised a generalized tabu search algorithm and a new, improved, upper bounding scheme based on a relaxation of the problem. Computational experiments are performed for the new methods on the problems with two and three machines. The results show that good optimality gaps are reached on all the instances.

At the end of these three chapters, some conclusions will be drawn.

**List of Publications**

Part of the work of this dissertation has been published and part of it is submitted for publication. In the following, the papers related to this work are reported.

- Benini M., Detti P., Zabalo Manrique de Lara G., Mathematical programming formulations and metaheuristics for biological sample transportation problems in healthcare, Computers & Operations Research, Volume 146, 2022.

- Benini M., Detti P., Zabalo Manrique de Lara G., A milp model for biological sample transportation in healthcare, *In: M. Paolucci et al. (eds), Advances in Optimization and Decision Science for Society, AIRO Springer Series*, 3:81–94, 2019.

- Detti P., Zabalo Manrique de Lara G., and Benini M., A metaheuristic apporach for biological

sample transportation in healthcare, *In: G. Gentile et al. (eds), Graphs and Combinatorial Optimization: from Theory to Applications, AIRO Springer Series*, 5:81–94, 2021.

- Agnetis A., Benini M., Detti P., Hermans B., Pranzo M., Replication and sequencing of unreliable jobs on parallel machines. Computers & Operations Research 139, 105634, 2021.

- Benini M., Blasi E., Detti P., Fosci L., Solving crop planning and rotation problems in a sustainable agriculture perspective, Submitted to Computers & Operations Research.

- Agnetis A., Benini M., Detti P., Hermans B., Pranzo M., Replication and sequencing of unreliable jobs on parallel machines: new results, In preparation.

- The models presented in Chapter 2 have received a positive opinion by the Patent Committee of University of Siena and are currently under patenting process.

# Chapter 1

# Mathematical Programming Formulations and Metaheuristic Algorithms for Biological Sample Transportation in Healthcare

In this chapter, a transportation problem is addressed, arising from a real world healthcare application [5] in Italy. The problem consists in satisfying transportation requests of biological samples: blood and other biological samples must be collected from different blood draw labs (or centers) and routed to the main laboratory, hereafter called HUB, where all samples are analyzed.

Starting in 2012, a reorganization plan for the territorial network of the Bologna Analysis Laboratories was conducted. The reorganization consists in the activation of the Single Metropolitan Laboratory (LUM) that includes the HUB center, 11 laboratories from hospitals of Bologna city and the province of Bologna, called spoke centers, the Ortopedico Rizzoli Institute and 68 sampling points. The main objective of the LUM project is reorganizing and optimizing laboratory processes.

Thus, one of the main strategic decisions of the project is the centralization of the analysis of biological samples in the area. More precisely, the analysys of all samples is entirely moved to the HUB laboratory (Ospedale Maggiore di Bologna), enabling the standardization of procedures and introducing a single control system. If on one hand the centralization and standardization of analysis processes lead to advantageous economies of scales, on the other hand it opens up to a new issue: the reorganization of the transportation of biological sample tubes from the sampling points to the HUB.

Samples are drawn from patients in different centers during morning hours, and have to be transported to the HUB by a fleet of vehicles located in geographically distributed depots. Each sample must be delivered to the HUB within its *lifetime*, in order to be correctly analyzed. However, samples can get an extra lifetime, useful for their on time delivery, by a *stabilization* process that can be performed in dedicated facilities called *spoke centers*, or spokes. Each sample can be stabilized at most once. After the stabilization, samples must be delivered to the HUB within the extra lifetime provided by the stabilization process. In particular, after the stabilization, a sample can be taken in charge by a vehicle that is different from the one that delivered it to the spoke. Hence, spokes may also act as transfer points in which (only) the stabilized samples may be transferred from one vehicle to another.

The problem is to find a transportation plan minimizing the total traveled distance, in such a way that all samples are delivered within their lifetimes to the HUB. Real historical data provided by the Local Healthcare Authority of Bologna, Italy, show that currently around 40% of the samples are not delivered on time, each day.

The problem can be formulated as a new variant of the Vehicle Routing Problem with the

following new features: *i*) Each transportation request has a limited lifetime and must be delivered to the HUB within that time; *ii*) Specific locations exist (i.e., the spokes), devoted to provide extra-lifetimes to the requests; *iii*) At the spokes, requests can be transferred from a vehicle to another.

The problem under study has been first addressed in [4; 8]. In [4], the problem is formally defined and a first mathematical model is provided, while in [8] a simple metaheuristic method is presented based on the Adaptive Large Neighborhood Search (ALNS) framework [26].

In this chapter, three Mixed Integer Linear Programming (MILP) formulations are proposed and compared on benchmark instances. Furthermore, metaheuristic algorithms are also proposed to tackle with real size instances. Two of the MILP formulations employ time indexed variables and the third one uses classical routing variables. A computational campaign on relatively small instances shows that the time indexed formulations outperform the standard formulation and are able to solve instances up to 20 transportation requests and 10 vehicles. As already stated, the metaheuristics have been developed to face with real-life instances. In the algorithms, the Adaptive Large Neighborhood Search method (first proposed by Ropke and Pisinger in [26]) has been combined with techniques used in other metaheuristic frameworks, such as Tabu Search (TS) and Variable Neighborhood Search (VNS). More precisely, while only feasible solutions are typically selected during the search in ALNS approaches, in our *hybrid* algorithms infeasible solutions can be also explored, and infeasibilities are properly taken into account by penalty terms into an evaluation function, as in TS and VNS methods (e.g., see [10], [21]). The hybrid ALNS (H-ALNS) algorithms proposed in this chapter are different and improved versions of the algorithm presented in [8]. The algorithm in [8] only employs four heuristics to generate the neighborhood at each iteration, and the possibility of stabilization is not considered during the search process (but only

evaluated by a post processing procedure at specific algorithm steps). Instead, the proposed new H-ALNS algorithms employ thirteen different heuristics that can also handle sample stabilization at each iteration and have been enriched by new components. As a computational campaign shows, the proposed H-ALNS algorithms are able to find optimal solutions on all the small instances and solutions in which all sample are delivered on time in real-life problems. On the other hand, the algorithm presented in [8] was not able to find feasible solutions for many real-life instances.

Finally, to face with the high dimension of the real instances, in which hundreds of samples must be delivered on time each day, a study is also presented to evaluate different grouping policies. By grouping samples in batches, we get instances with a smaller number of transportation requests, but with stricter lifetime requirements. The aim is to evaluate what are the batching policies on which the H-ALNS algorithms attain the best performances, in terms of solution quality and computational time. Such grouping policies may also give useful insights from an operations point of view.

The work presented in this chapter has been published on Computers & Operations Research [3]. The chapter is organized as follows. In Section 1.1, we review results from the literature. In Section 1.2, a detailed description of the problem is presented. In Section 1.3, the MILP formulations are presented. Section 1.4 describes the H-ALNS algorithms developed to solve real-life instances of the problem. The description of the real-life data and the computational results are reported in Section 1.6. Finally, conclusions are gathered in Section 1.7.

## 1.1 Literature review

Blood transportation and logistics are topics addressed by several works in the literature, and many problems have been modeled as vehicle routing problems (VRPs) [2]. Doerner *et al.* [9] address a vehicle routing problem with multiple interdependent time windows arising from the Austrian Red Cross blood program. A mixed-integer programming formulation and heuristics are proposed for solving the problem. In [1], a biomedical sample transportation problem is modeled as a vehicle routing problem, with precedence constraints and samples' lifetime, and solved by an iterated local search algorithm. Liu *et al.* [16] address a periodic vehicle routing problem concerning the transportation of drugs and blood samples.

Grasas *et al.* [11] present a study to improve the logistics of blood sample collection at two important clinical laboratories in Catalonia, Spain. The collection and transportation problem is solved by a heuristic method based on a genetic algorithm. In [27], a problem concerning the transportation of blood samples from blood mobile draw centers to a central depot is addressed and solved by a matheuristic algorithm. In [12], a metaheuristic algorithm is proposed for a problem in which blood have to be transported between hospitals or donor/client sites. In [28], the problem concerning the allocation of blood units to the hospitals for transfusion is addressed and solved as a multi-objective transportation problem.

Many works in the literature address the management and optimization of supply chain networks in different healthcare contexts. For a review on models and methods for blood supply chain management we refer to [22]. In [13], the problem of designing a blood supply chain network is considered. A multi objective mixed integer mathematical programming model is presented that aims to simultaneously minimize the total cost of the supply chain network and the total envi-

ronmental impact. In [15] a multi-start iterated local search algorithm is proposed for delivering pharmaceutical products to healthcare facilities. Moussavi et al. [20] proposed a matheuristic to find vehicle routings and worker assignments to provide home healthcare services. In [14], the optimization of supply chain for CAR T-cell therapies has been considered, including the optimization of samples transportation. A mixed-integer linear programming model is presented and a General Variable Neighborhood Search to tackle larger problem instances is proposed. Liu *et al.* [17] propose metaheuristic approaches for a special vehicle routing problem with simultaneous delivery and pickup and time windows. The problem concerns the delivery of drugs and medical devices from the home care company's pharmacy to patients' homes, delivery of special drugs from a hospital to patients, pickup of bio samples and unused drugs and medical devices from patients. Shi, Y. *et al.* [30] address a vehicle routing problem with fuzzy demand for scheduling home health care services, and propose a hybrid genetic algorithm. The optimization of patients transportation systems have been considered in many works (e.g., see [7; 31]).

In the problem addressed in this chapter, samples can be transferred from one vehicle to another at the spoke centers. In the literature, transportation problems with "transfers" have been considered by different authors [6; 18; 25]. The usefulness of transshipment is investigated in [25]: The computational results show that transfers can indeed enhance optimization in pick up and delivery problems.

In [18], an Adaptive Large Neighborhood Search algorithm is proposed for the pick up and delivery problem with transfers (PDPT).

In [6], a solution method based on Benders decomposition is presented for Dial-a-Ride problems with transfers, in which passengers may be transferred from one vehicle to another at specific

locations. Note that, unlike the general scheme of the PDPT, in the problem addressed in this chapter only the requests that are stabilized at the spoke centers can be transferred between vehicles.

From the above literature review, we can conclude that none of the previous works attempt to deal with transportation problems where common vehicle routing features simultaneously coexist with perishable item (e.g., biological samples) issues, and restoring and transfer facilities (i.e., the spoke centers). These problem characteristics introduce very challenging difficulties, both at the formulation and solving steps, as it will be clarified in the following sections.

## 1.2 Problem Description

In the addressed problem, a set of blood and biological samples are produced each day in geographically distributed blood draw laboratories, and must be picked up and delivered to a main laboratory (called HUB) by vehicles located in different depots. Each sample is available to be loaded by a vehicle within a time window, starting at the time of its withdrawal from a patient and ending at the closure time of the draw lab. Each sample has a specific lifetime, that is the time beyond which the sample cannot be analyzed anymore. Hence, all the samples have to arrive at the HUB within their lifetime. When this is not possible, samples can be *stabilized* to gain an extra lifetime. Different centers, i.e., the spokes, are devoted to perform the stabilization on the samples. Each sample can be stabilized at most once and, after the stabilization, samples must be delivered within the extra lifetime. Multiple visits of the spoke centers are allowed for the vehicles. Furthermore, after the stabilization, samples can be taken in charge and delivered to the HUB by a vehicle that is different from the one that delivered them to the spoke. It is important to notice that the transfer of samples between vehicles is allowed only for stabilized samples. In other words, the

spoke centers cannot be used for transfer only. A *service time* is needed to load or unload samples at draw labs, spokes or the HUB.

The problem is to define the routing of each vehicle so that all samples are delivered within their lifetime and the total distance traveled by the vehicles is minimized.

In what follows, we refer to samples as to *transportation requests* or simply *requests*. In Figure 1.1, the two possible transportation modes of a request (i.e. a sample) are shown (with or without stabilization).

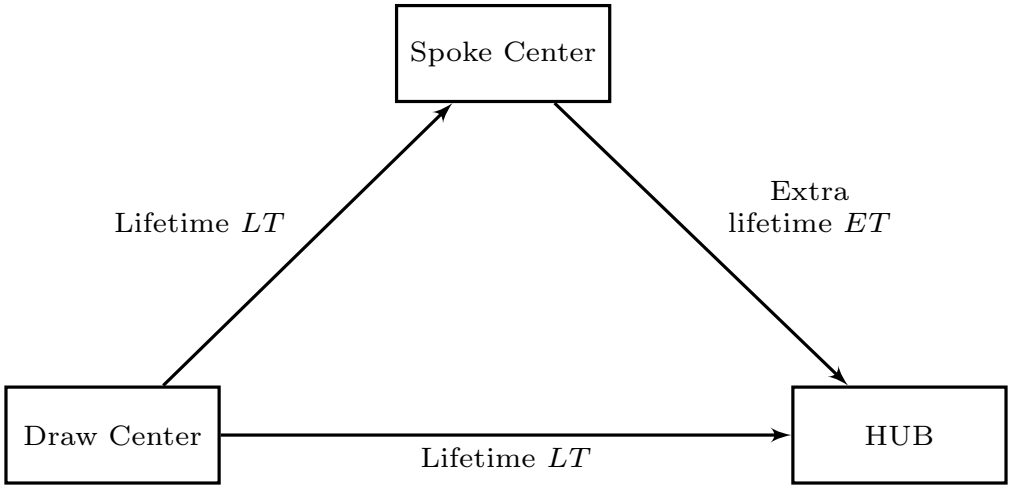



Figure 1.1: Transportation modes of a sample.

A formal definition of the problem is given in the following.

Let $R = \{1,\ldots,n\}$ and $K = \{1,\ldots,m\}$ be the sets of transportation requests and vehicles, respectively. The transportation network can be modeled as a complete directed graph $G = (N,A)$, where $N = \{1,\ldots,n,n+1,n+2,\ldots,n+s+2,n+s+3,\ldots,n+s+m+3\}$ is the node set and $A = \{(i,j) : i,j \in N\}$ is the arc set. The pickup locations of the requests are the nodes in $P = \{1,\ldots,n\}$ and the delivery node is $n+1$, also denoted as $H$, corresponding to the HUB. The spoke centers corresponds to the nodes in $S = \{n+2,\ldots,n+s+2\}$. Finally, nodes in $D =$

$\{n+s+3,\ldots,n+s+m+3\}$ correspond to the depots where vehicles are located. In the following, we denote by $p(r) \in P$ the pickup node of the request $r \in R$ and by $d(k)$ the depot of vehicle $k \in K$. Each request $r \in R$ has a lifetime $LT_r$, and a time window $[e_r, l_r]$. Hence, request $r \in R$ can be picked up from its pickup location $p(r)$ between time $e_r$ and $l_r$. If a vehicle arrives at a pickup location $p(r)$ before $e_r$, then it will wait $e_r$ to begin the load of the sample. The service time to load/unload requests at each node is $st$. If a request $r$ is not stabilized, then it must be delivered to the HUB at a time not bigger than $e_r + LT_r$. If $r$ is stabilized at a spoke center, it must be delivered to the spoke center at a time not bigger than $e_r + LT_r$. $ET_r$ is the extra lifetime gained by $r$ with the stabilization. After the stabilization, request $r$ must be delivered to the HUB within a time $se_r + ET_r$, where $se_r$ denotes the end time of the stabilization process. The time needed to perform the stabilization process is the same for all spokes and requests, and is denoted as $stbt$.

The problem consists in finding a set of at most $m$ routes on $G$ such that:

- each route starts and ends at the same depot;
- each route visits the main hospital right before the arrival to the depot;
- the service at node $p(r) \in P$ begins in the interval $[e_r, l_r]$, for each request $r \in R$;
- each request can be stabilized in a spoke center at most once;
- all the requests arrive to the spokes (if stabilized) and to the main hospital on time;
- the total length of the routes is minimized.

## 1.3 Mixed Integer Linear Programming formulations

In this section, MILP models of the problem are presented. In the formulations, a modeling issue concerns the tracking of two different flows: the flow of vehicles and the flow of requests. The first represents the route followed by each vehicle from its depot to the HUB and back to its depot. The second is related to the path of each request from its origin to its final destination. Since the problem allows transshipment of requests at the spoke centers, in general the two flows can be decoupled at the spokes. So, it is important to keep track of both flows and to make sure that the they are consistent with each other. Another modeling issue is related to the possibility of visiting multiple times a given spoke center by the same vehicle. As an example, multiple visits of a spoke center may occur when a vehicle drops a request at a spoke for stabilization, departs from the spoke and returns to pick up that request after the end of the stabilization. Also requests may visit a spoke multiple times, even if they can be stabilized at most once.

In what follows, three Mixed Integer Linear Programming formulations are presented for the problem. Two of them, denoted as $MILP_1$ and $MILP_2$, use time indexed variables to deal with multiple visits. As explained in Section 1.3.2, the main difference between the two formulations is that $MILP_2$ has a smaller number of variables than $MILP_1$. The third formulation, called $MILP_3$, employs standard routing variables and uses multiple copies of each spoke so that each copy can be visited at most once. The computational campaign on small instances reported in Section 1.6.2 shows that $MILP_1$ and $MILP_2$ are able to solve problems up to 20 requests, while $MILP_3$ fails to find feasible solutions within the time limit in all the instances, suggesting that the time-indexing is more effective to deal with multiple visits at the spokes.

### 1.3.1 Formulation $MILP_1$

As already stated, in $MILP_1$, time indexing is used to properly address routes visiting more than once the same spoke center. At this aim, let $t \in TT$ be the time index, where $TT$ is the time horizon considered. In the formulations $MILP_1$ (and also $MILP_2$), we denote by $loc(i)$ the location of node $i \in N$. In $MILP_1$ the following variables are used:

- $x_{ij}^{kt} \in \{0,1\}$ is equal to 1 if vehicle $k$ traverses arc $(i,j)$ arriving in $j$ at time $t$ and 0 otherwise;
- $y_{ij}^{krt} \in \{0,1\}$ is 1 if vehicle $k$ traverses arc $(i,j)$ carrying request $r$ arriving in $j$ at time $t$ and 0 otherwise;
- $z_j^{krt} \in \{0,1\}$ is equal to 1 if vehicle $k$ arrives in spoke $j$ at time $t$ to drop request $r$ for stabilization and 0 otherwise;
- $w_j^{krt} \in \{0,1\}$ is equal to 1 if vehicle $k$ arrives in spoke $j$ at time $t$ to pick up request $r$ and 0 otherwise.
- $A_{r+n}$ is the arrival time of request $r$ to the HUB.

The objective function is to minimize the total traveled distance:

$$\min \sum_{t \in TT} \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^{kt} \tag{1.1}$$

where $d_{ij}$ is the travel distance between nodes $i$ and $j$ of $N$.

The constraints of the model can be divided into three groups: $(i)$ *Flow Constraints*, modeling both the flow of requests and the flow of vehicles; $(ii)$ *Spoke Constraints*, modeling the dynamic of

the stabilization at spoke centers; $(iii)$ *Time Constraints*, imposing the time windows and lifetime restrictions of the problem as well as the consistency of the time indexing.

The Flow Constraints read as follows.

$$\sum_{t\in TT}\sum_{j\in N, j\neq d(k)} x^{kt}_{d(k)j} \leq 1, \forall k \in K \tag{1.2}$$

$$\sum_{t\in TT} x^{kt}_{Hd(k)} = \sum_{t\in TT}\sum_{i\in N, i\neq H} x^{kt}_{iH}, \forall k \in K \tag{1.3}$$

$$\sum_{t\in TT} x^{kt}_{ij} + \sum_{t\in TT} x^{kt}_{ji} \leq 1, \forall i \in D, \forall j \in S, \forall k \in K \tag{1.4}$$

$$\sum_{t\in TT}\sum_{j\in N, j\neq d(k)} x^{kt}_{d(k)j} = \sum_{t\in TT}\sum_{j\in N, j\neq d(k)} x^{kt}_{jd(k)}, \forall k \in K \tag{1.5}$$

$$\sum_{t\in TT}\sum_{j\in N, j\neq i} x^{kt}_{ij} - \sum_{t\in TT}\sum_{j\in N, j\neq i} x^{kt}_{ji} = 0, \forall k \in K, \forall i \in N \tag{1.6}$$

$$x^{kt}_{p(j)p(i)} = 0,$$

$$\forall k \in K, \forall t \in TT, \forall j,i \in R : loc(p(i)) = loc(p(j)), e_i < e_j \tag{1.7}$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq i} x^{kt}_{ij} = 1, \forall i \in P \tag{1.8}$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq p(r)} y^{krt}_{p(r)j} = 1, \forall r \in R \tag{1.9}$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{i\in N, i\neq H} y^{krt}_{iH} = 1, \forall r \in R \tag{1.10}$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq i} y^{krt}_{ij} - \sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq i} y^{krt}_{ji} = 0, \forall r \in R, \forall i \in S \tag{1.11}$$

$$\sum_{t\in TT}\sum_{j\in N, j\neq i} y^{krt}_{ij} - \sum_{t\in TT}\sum_{j\in N, j\neq i} y^{krt}_{ji} = 0,$$

$$\forall r \in R, \forall k \in K, \forall i \in P \setminus p(r) \tag{1.12}$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq p(r)} y^{krt}_{jp(r)} = 0, \forall r \in R \tag{1.13}$$

$$\sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq p(r)} y^{krt}_{p(r)j} - \sum_{t\in TT}\sum_{k\in K}\sum_{j\in N, j\neq p(r)} y^{krt}_{jp(r)} = 1, \forall r \in R \tag{1.14}$$

$$y^{krt}_{ij} \leq x^{kt}_{ij}, \forall r \in R, \forall k \in K, \forall (i,j) \in A, \forall t \in TT \tag{1.15}$$

Constraints (1.2) and (1.3) impose vehicles to leave at most once from their depot and to return to their depots right after the visit of the HUB, respectively. Constraints (1.4) forbid a vehicle to depart from its depot and consecutively visit a spoke and its depot again, since this would certainly not involve the transportation of requests. Constraints (1.5) impose each route to begin and end at the same depot. Equalities (1.6) ensure the conservation of the flow of vehicles at nodes, while Constraints (1.7) forbid sub-cycles between pickup nodes. Constraints (1.8)–(1.10) state that each transportation request must be served exactly once. Constraints (1.11) and (1.12)–(1.14) ensure the conservation of the flow of requests at pickup nodes and spoke nodes, respectively. Note that Constraints (1.11) allow transfers of requests at spoke nodes. Finally, Constraints (1.15) link the request flow and the vehicle flow.

Now, the Spoke Constraints are presented.

$$\sum_{t\in[e_r,e_r+LT_r]}\sum_{k\in K}\sum_{j\in S} z_j^{krt} \leq 1, \forall r \in R \tag{1.16}$$

$$z_j^{krt} \leq \sum_{i\in N, i\neq j} y_{ij}^{krt}, \forall j \in S, \forall k \in K, \forall r \in R, \forall t \in [e_r, e_r+LT_r] \tag{1.17}$$

$$\sum_{t\in[e_r,l_r+LT_r]}\sum_{k\in K} z_j^{krt} = \sum_{t\in[e_r,e_r+LT_r+stbt+ET_r]}\sum_{k\in K} w_j^{krt}, \forall j \in S, \forall r \in R \tag{1.18}$$

$$x_{ij}^{kt} \leq \sum_{r\in R} z_j^{krt} + \sum_{\tau=t+st}^{TT}\sum_{r\in R} w_j^{kr\tau},$$
$$\forall j \in S, \forall k \in K, \forall i \in N, i \neq j, \forall t \in TT \tag{1.19}$$

$$\sum_{i\in N, i\neq j} y_{ji}^{kr(t+t_{ji})} \geq w_j^{krt},$$
$$\forall r \in R, \forall j \in S, \forall k \in K, \forall t \in [e_r, e_r+LT_r+stbt+ET_r] \tag{1.20}$$

$$z_j^{krt} \leq \sum_{q\in K}\sum_{\tau=t+st+stbt+st}^{e_r+LT_r+stbt+ET_r} w_j^{qr\tau},$$
$$\forall j \in S, \forall k \in K, \forall r \in R, \forall t \in [e_r, e_r+LT_r] \tag{1.21}$$

$$w_j^{qrt} \leq \sum_{q \in K} \sum_{\tau=e_r}^{t-st-stbt} z_j^{qr\tau},$$

$$\forall j \in S, \forall k \in K, \forall r \in R, \forall t \in [e_r, e_r + LT_r + stbt + ET_r] \quad (1.22)$$

$$\sum_{i \in N, i \neq j} \sum_{\tau \geq t+st+t_{ji}} x_{ji}^{k\tau} \geq z_j^{krt} - M[(1 - z_j^{krt})]$$

$$\forall r \in R, \forall k \in K, \forall j \in S, \forall t \in [e_r, e_r + LT_r] \quad (1.23)$$

$$\sum_{i \in N, i \neq j} \sum_{\tau=t+st+t_{ji}}^{t_2-1} x_{ji}^{k\tau} \geq 1 - M[(1 - \sum_{l \in N, l \neq j} x_{lj}^{kt}) + (1 - \sum_{l \in N, l \neq j} x_{lj}^{kt_2}],$$

$$\forall k \in K, \forall j \in S, \forall t, t_2 \in TT, t > e_r, t_2 > t \quad (1.24)$$

$$\sum_{i \in N, i \neq j} \sum_{\tau \geq t+st+t_{ji}} y_{ji}^{kr\tau} \geq \sum_{i \in N, i \neq j} y_{ij}^{krt} - z_j^{krt},$$

$$\forall j \in S, \forall r \in R, \forall k \in K, \forall t \in TT, t > e_r \quad (1.25)$$

$$\sum_{t \in TT} \sum_{j \in S} z_j^{krt} \leq \sum_{h \in N} \sum_{t \in TT} x_{hi(r)}^{kt}, \forall r \in R, i \in P \quad (1.26)$$

Constraints (1.16) impose the maximum of one stabilization per request. Constraints (1.17) link the flow of requests with their possibility of stabilization. Constraints (1.18) impose that a stabilized request must be collected from the spoke where it was processed.

Constraints (1.19) ensure that a vehicle can visit a spoke node only if it has either to leave a request for stabilization or to withdraw a stabilized request. Constraint (1.20) links the variables $y$ and $w$ while Constraints (1.21)–(1.22) link the timing between the $z$ and $w$ variables. Constraints (1.23)–(1.24) contribute to define the exit time of a vehicle from a spoke node and to manage the multiple visits at a spoke node by imposing that each vehicle has to depart from a spoke node within the time interval between two different visits of the node. Constraints (1.25) impose that transfers at spoke centers are not allowed for non-stabilized requests. Lastly, Constraints (1.26) state that a request can be left at a spoke node for stabilization only by the vehicle that loaded it from its pickup

node.

Finally, the Timing Constraints are reported in the following.

$$e_r \leq B_p(r) \leq l_r, \forall r \in R \tag{1.27}$$

$$B_j \geq \sum_{k \in K} \sum_{t=t_{d(k)}}^{TT} \sum_{i \in N, i \neq j} tx_{ij}^{kt}, \forall j \in P \tag{1.28}$$

$$tx_{ij}^{kt} \geq B_i + st + t_{ij} - M(1 - x_{ij}^{kt}),$$

$$\forall t \in TT, t > min_r e_r, \forall k \in K, \forall i \in P, \forall j \in N, loc(i) \neq loc(j) \tag{1.29}$$

$$tx_{ij}^{kt} \leq B_i + st + t_{ij} + M(1 - x_{ij}^{kt}),$$

$$\forall t \in TT, t > min_r e_r, \forall i \in P, \forall j \in N, loc(i) \neq loc(j) \tag{1.30}$$

$$tx_{ij}^{kt} \geq B_i - M(1 - x_{ij}^{kt}),$$

$$\forall t \in TT, t > min_r e_r, \forall k \in K, \forall i \in P, \forall j \in N, loc(i) = loc(j) \tag{1.31}$$

$$tx_{ij}^{kt} \leq B_i + M(1 - x_{ij}^{kt}),$$

$$\forall t \in TT, t > min_r e_r, \forall i \in P, \forall j \in N, loc(i) = loc(j) \tag{1.32}$$

$$A_{r+n} = \sum_{t \in TT, t > e_r} \sum_{k \in K} \sum_{i \in N, i \neq H} ty_{iH}^{krt} \forall r \in R \tag{1.33}$$

$$A_{r+n} + st - e_r \leq LT_r(1 - \sum_{t=e_r}^{e_r+LT_r} \sum_{k \in K} \sum_{j \in S} z_j^{krt}) + M \sum_{t=e_r}^{e_r+LT_r} \sum_{k \in K} \sum_{j \in S} z_j^{krt},$$

$$\forall r \in R \tag{1.34}$$

$$A_{r+n} + st - \sum_{t=e_r}^{e_r+LT_r} \sum_{k \in K} \sum_{j \in S} (t + st + stbt) z_j^{krt} \leq$$

$$M(1 - \sum_{t=e_r}^{e_r+LT_r} \sum_{k \in K} \sum_{j \in S} z_j^{krt}) + ET_r \sum_{t=e_r}^{e_r+LT_r} \sum_{k \in K} \sum_{j \in S} z_j^{krt}, \forall r \in R \tag{1.35}$$

$$\sum_{t=e_r}^{e_r+LT_r} \sum_{k \in K} \sum_{j \in S} (t + st) z_j^{krt} - e_r \leq LT_r, \forall r \in R \tag{1.36}$$

Constraints (1.27) impose the compliance with the time windows restriction, while Constraints (1.28) describe the relationship between the beginning of the service at a node and the arrival time at that node. Constraints (1.29)-(1.32) ensure the correct timing between two pickup nodes while Constraints (1.33) define the arrival at the HUB variables. Compliance of lifetimes is ensured by Constraints (1.34) when a request is not stabilized at a spoke, while Constraints (1.35)–(1.36) ensure the lifetime and extra lifetime compliance when a request is stabilized at a spoke node.

### 1.3.2 Formulation $MILP_2$

The formulation $MILP_2$ is basically obtained by removing variables $y_{ij}^{krt}$ from formulation $MILP_1$, and adjusting some constraints accordingly. Hence, $MILP_2$ has a considerably smaller number of variables, since it only employs the variables $x_{ij}^{kt}$, $z_j^{krt}$, $w_j^{krt}$ and $A_{r+n}$ already introduced at the beginning of Section 1.3.1.

In $MILP_1$, variables $y_{ij}^{krt}$ are used to keep track of the flow of the requests by their pickup locations to the HUB. However, it is possible to do it by only employing variables $x_{ij}^{kt}$, $z_j^{krt}$ and $w_j^{krt}$. In particular, let us consider a request $r$ and its pickup location $p(r)$. If $r$ is not stabilized (i.e., $z_j^{krt} = w_j^{krt} = 0 \; \forall j \in S, k \in K, t \in TT$), then the request will simply follow the route of the vehicle picking it at $p(r)$ up to the HUB. On the other hand, if $r$ is stabilized at a given spoke $s$, than it will be: $(i)$ on the route of the vehicle withdrawing it at $p(r)$, from the pickup location $p(r)$ to the spoke $s$; $(ii)$ on the route of the vehicle picking it up at $s$, from the spoke $s$ to the HUB. In all the above cases, the vehicles transporting the request $r$ can be identified by the only variables $x_{ij}^{kt}$, $z_j^{krt}$

and $w_j^{krt}$.

In the following the $MILP_2$ model is presented. Since $MILP_2$ shares many constraints with $MILP_1$, only the new constraints are reported. As for $MILP_1$, also in $MILP_2$ the constraints of the model can be divided into three groups. The Flow Constraints of the model are the Constraints (1.2)–(1.8) of $MILP_1$ and those reported in the following.

$$\sum_{t \in TT} \sum_{j \in N, j \neq i} x_{jH}^{kt} \leq 1, \forall k \in K \tag{1.37}$$

Constraints (1.37) state that a vehicle can visit the HUB at most once.

In $MILP_2$, the set of the Spoke Constraints is composed by the Constraints (1.16), (1.18), (1.19), (1.21)–(1.24) and (1.26) of $MILP_1$ and by the Constraints (1.38)–(1.41) reported in the following.

$$z_j^{krt} \leq \sum_{i \in N, i \neq j} x_{ij}^{kt}, \forall j \in S, \forall r \in R, \forall k \in K,$$
$$\forall t \in [min_{r \in R} e_r, max_{r \in R} e_r + max_{r \in R} LT_r] \tag{1.38}$$

$$\sum_{i \in N, i \neq j} x_{ji}^{kr(t+t_{ji})} \geq w_j^{krt}, \forall j \in S, \forall k \in K, \forall r \in R$$
$$\forall t \in [min_{r \in R} e_r, max_{r \in R} e_r + max_{r \in R} LT_r + st + stbt + max_{r \in R} ET_r] \tag{1.39}$$

$$\sum_{\tau \leq t + st + min_j t_{p(r)j}} z_j^{kr\tau} \leq M(1 - \sum_{i \in N, i \neq p(r)} x_{ip(r)}^{kt}), \forall k \in K, \forall r \in R,$$
$$\forall t \in [e_r, e_r + LT_r + stbt + ET_r - st] \tag{1.40}$$

$$\sum_{i \in N, i \neq H} \sum_{t \in TT} x_{iH}^{kt} \geq \sum_{i \in N, i \neq H} \sum_{t \in TT} x_{ip(r)}^{kt} -$$
$$\sum_{j \in S, i \neq H} \sum_{t \in TT} z_j^{krt} + \sum_{j \in S, i \neq H} \sum_{t \in TT} w_j^{krt}, \forall r \in R, \forall k \in K \tag{1.41}$$

Constraints (1.38) and (1.39) replace (1.17) and (1.20) of $MILP_1$, respectively, ensuring con-

sistency between the flow of vehicles and the download/withdraw of requests at a spoke node. Constraints (1.40) impose the minimum time needed for a request to be left at a spoke node after its withdrawal from its pickup location. Lastly, Constraints (1.41) force a vehicle to reach the HUB if it has withdrawn a request at its pickup location and the request is not stabilized, or if it picks up a request after stabilization at a spoke node.

Finally, the Timing Constraints are

$$(1.27)-(1.32) \text{ and } (1.34)-(1.36)$$

$$A_{r+n} \geq \sum_{i\in N}\sum_{t\in TT} tx_{iH}^{kt} - M(1-\sum_{h\in N}\sum_{t\in TT} x_{hi(r)}^{kt}) - M\sum_{q\in K}\sum_{t\in TT}\sum_{j\in S} z_j^{qrt}, \quad \forall r\in R, \forall k\in K \tag{1.42}$$

$$A_{r+n} \geq \sum_{i\in N}\sum_{t\in TT} tx_{iH}^{kt} - M(1-\sum_{t\in TT}\sum_{j\in S} w_j^{krt}), \quad \forall r\in R, \forall k\in K \tag{1.43}$$

Constraints (1.42)–(1.43) replace (1.33) of $MILP_1$ and set the arrival at the HUB of a request according to the vehicle that will carry it.

### 1.3.3 Formulation $MILP_3$

Formulation $MILP_3$ is similar to the one proposed in [25] for the pickup and delivery problem with transfers, with modifications to fit our problem. As already stated, $MILP_3$ employs multiple copies of the spokes, and allows at most one visit of each copy by each vehicle. More precisely, $2n$

identical copies are considered for each spoke, where $n$ is the number of requests. (In practice, a single vehicle may visit the same spoke center at most $2n$ times, i.e., 2 times for each request.)

The formulation employs the following decision variables:

- $x_{ij}^k \in \{0,1\}$ is equal to 1 if vehicle $k$ traverses arc $(i,j)$ and 0 otherwise;
- $y_{ij}^{kr} \in \{0,1\}$ is 1 if vehicle $k$ traverses arc $(i,j)$ carrying request $r$ and 0 otherwise;
- $s_{jpr}^{kl} \in \{0,1\}$ is equal to 1 if request $r$ arrives at the spoke copy $j$ with vehicle $k$ and departs after stabilization with vehicle $l$ (possibly $k = l$) from spoke copy $p$, and 0 otherwise. Notice that $j$ and $p$ are copies of the same spoke.
- $A_i^k$ is the arrival time of vehicle $k$ at node $i$.
- $B_i^k$ is the beginning of service of vehicle $k$ at node $i$.
- $A_{r+n}$ is the arrival time of request $r$ to the HUB.

The objective function is to minimize the total traveled distance, as in the other two formulations.

Hereafter, the constraints modeling the visits to the spokes are reported, that mainly differentiate this model from the one proposed in [25]. In the following, we denote with $S$ the set of spokes and, given a spoke $s \in S$, with $SC(s)$ the set of copies of $s$.

$$\sum_{j \in N, j \neq i} y_{ji}^{kr} + \sum_{j \in N, j \neq i} y_{hj}^{lr} \leq s_{ihr}^{kl} + 1,$$
$$\forall r \in R, \forall k,l \in K, k \neq l, \forall s \in S, \forall i,h \in SC(s) \tag{1.44}$$

$$\sum_{j \in N, j \neq i} y_{ji}^{kr} \geq \sum_{h \in SC(s)} s_{ihr}^{kl}, \forall r \in R, \forall k,l \in K, \forall s \in S, \forall i \in SC(s) \tag{1.45}$$

$$\sum_{j\in N, j\neq i} y_{ij}^{lr} \geq \sum_{h\in SC(s)} s_{hir}^{kl}, \forall r \in R, \forall k,l \in K, \forall s \in S, \forall i \in SC(s) \tag{1.46}$$

$$\sum_{i\in N, i\neq j} x_{ij}^{k} \leq \sum_{l\in K}\sum_{r\in R}\sum_{h\in SC(s)} s_{jhr}^{kl} + \sum_{l\in K}\sum_{r\in R}\sum_{h\in SC(s)} s_{hjr}^{lk},$$
$$\forall k \in K, \forall s \in S, \forall j \in SC(s) \tag{1.47}$$

$$\sum_{j\in SC(s), l\in K} s_{hjr}^{kl} \leq \sum_{i\in N, i\neq p(r)} x_{ip(r)}^{k},$$
$$\forall r \in R, \forall k \in K, \forall s \in S, \forall h \in SC(s) \tag{1.48}$$

$$\sum_{i\in N} x_{ij}^{k} \leq 1, \forall k \in K, \forall k \in K, \forall s \in S, \forall j \in SC(s) \tag{1.49}$$

$$\sum_{k,l\in K}\sum_{i,h\in SC(s), s\in S} s_{ihr}^{kl} \leq 1, \forall r \in R \tag{1.50}$$

Constraints (1.44) link variables $s$ and $y$ imposing variables $s$ to be equal to 1 when two different vehicles drop and withdraw a request from a spoke. Constraints (1.45) and (1.46) link variables $y$ and $s$ in the other way around. Constraints (1.47) state that a vehicle can visit a spoke only to drop or withdraw a request. Constraints (1.48) impose that only the vehicle that loaded the request at its pick-up location can transport it to a spoke for stabilization. Finally, Constraints (1.49) impose for each vehicle a maximum of one visit of each spoke copy, while Constraints (1.50) state that a request can be stabilized at most once.

## 1.4 Hybrid ALNS algorithms

In this section, the algorithmic framework developed for the problem is presented, hereafter called Hybrid ALNS (H-ALNS). It is based on the ALNS framework proposed in [26] but, differently from [26], in which only feasible solutions are explored, in our approach we also allow the exploration of infeasible solutions. As also proposed in other metaheuristic frameworks from the

literature (e.g., see the Tabu Search [10] and the Variable Neighborhood Search [21] techniques), in H-ALNS infeasibilities are handled by introducing them, as penalty terms, in the evaluation function $f(\cdot)$ used to assess the quality of the solutions (the details are reported in Section 1.4.2). The impact of these penalty terms is then adjusted during the algorithm execution.

Furthermore, ad-hoc developments have been designed for the problem, including heuristics able to deal with spokes and stabilization and a post-processing procedure useful for reducing lifetime violations of infeasible solutions.

The remainder of this section is organized as follows. First, in Section 1.4.1 the general structure of the proposed algorithms is presented. Then, in Section 1.4.2, the evaluation function used to assess the quality of the solutions is introduced. From Section 1.4.3 to 1.4.7, the five phases (briefly described in Section 1.4.1) of the iterative core of the algorithms are shown into detail. Finally, in Section 1.4.8, the procedure for finding an initial solution is presented.

### 1.4.1 General structure of the H-ALNS

The developed H-ALNS algorithms are composed of two main parts. The first part is devoted to the generation of an initial solution $s_0$. The initial solution is built with specific procedures involving several heuristics embedded in an ALNS framework, as explained in Section 1.4.8. The solution $s_0$ resulting from this part is set as current solution $s_{curr}$. The second part is the iterative core of the algorithms, which is performed until a maximum number of iterations is reached. In this part, each iteration is divided into five phases, executed consecutively, that are explained in the following.

- *Phase 1: Destroy*. In this phase the current solution $s_{curr}$ is destroyed by removing a given number of requests from it, resulting in a partial solution $s'$. The removal of such requests is regulated by one of the *destroy heuristics* introduced in Section 1.4.3. The destroy heuristic to employ during each iteration $i$ is chosen with a probability depending on the performances of the heuristic up to $i$. To keep track of these performances some parameters are used, as explained in section 1.4.6

- *Phase 2: Repair*. In this phase the partial solution $s'$ is repaired by reinserting all the unplanned requests, i.e., the requests that were removed in the previous phase, generating a new solution $s_{new}$. The removed requests are inserted by one of the *repair heuristics* introduced in Section 1.4.4. As in Phase 1, in this phase the repair heuristic to employ at a given iteration $i$ is chosen with a probability depending on the performances of the heuristic up to $i$. To keep track of these performances some parameters are used, as explained in section 1.4.6.

- *Phase 3: Evaluate*. In this phase the new solution $s_{new}$ is evaluated. If it is feasible and better than the best known feasible solution, it is kept as a new best feasible solution. Similarly, if $s_{new}$, not necessarily feasible, is better than the best known solution, it is kept as a new best solution. Finally, if $s_{new}$ meets the *acceptance criteria* specified in Section 1.4.5, it is accepted as new current solution for the next iteration.

- *Phase 4: Update*. In this phase, some parameters of the algorithm are adjusted for the next iteration, based on the performances of the current iteration. In particular, two different sets of parameters are updated: $(i)$ *penalty parameters* and $(ii)$ *adaptivity parameters*. The first set of parameters are updated at each iteration according to the ability of the current

iteration to generate a feasible solution. The adaptivity parameters, related to each heuristic, are updated after every *block* of $I$ consecutive iterations according to the performances of the specific destroy/repair heuristic employed during the iteration. This procedure is explained into detail in Section 1.4.6.

- *Phase 5: Post-Segment Procedure*. This phase is not executed at each iteration, but only after every $I$ consecutive iterations. In what follows, a block of $I$ consecutive iterations is called *segment*. In this phase, a procedure designed to reduce the lifetime violations of the current solution $s_{curr}$, if any, is executed. A detailed description of Phase 5 is given in Section 1.4.7.

Algorithm 1 reports a general scheme of our H-ALNS approach.

**Algorithm 1** H-ALNS Algorithm general framework

**First Part: Initial Solution**
Generate the initial solution $s_0$;
Set $s_{best} = s_0$ and $s_{curr} = s_0$ $f(s_{bestFeas}) = +\infty$;
**Second Part: Iterative Core**
$l = 0$;
**Repeat**
Phase 1: Destroy
Select a destroy heuristic;
Use the destroy heuristic to remove $r$ requests from $s_{curr}$:
$s' \leftarrow Destroy(s_{curr})$
Phase 2: Repair
Select a repair Heuristic;
Use the repair heuristic to insert $r$ requests:
$s_{new} \leftarrow Repair(s')$
Phase 3: Evaluate
**If** $s_{new}$ is feasible and $f(s_{new}) < f(s_{bestFeas})$
Set $s_{bestFeas} := s_{new}$
**If** $f(s_{new}) < f(s_{best})$
Set $s_{best} := s_{new}$
**If** $Accept(s_{new}) = true$
$s_{curr} = s_{new}$
Phase 4: Update
Update the *penalty* parameters **and If** $(l = I)$ Update the *adaptivity* parameters;
$l = l + 1$;
**If** $l = I$
Phase 5: Post-Segment Procedure
Execute the Post-Segment Procedure; $l = 0$;
**Until** $It_{max}$ iterations are performed
**return** $s_{best}$ and $s_{bestFeas}$.

### 1.4.2 Evaluation Function

As already stated, the proposed algorithms allow the exploration of infeasible solutions. More precisely, solutions may violate the time window and lifetime constraints, and these violations are properly taken into account in an *evaluation function*. Given a solution $s$ of the problem, the evaluation function is defined as $f(s) = f_1(s) + f_2(s)$, where $f_1(s)$ corresponds to the total traveled distance of all routes, i.e., the problem objective function, and the second component is a penalization term of the form $f_2(s) = \alpha t(s) + \beta w(s)$, where $t(s)$ and $w(s)$ represent the total lifetime and time windows violation of solution $s$, respectively, and $\alpha > 0$ and $\beta > 0$ are *penalty parameters*. The violations are computed as in (1.51) and (1.52). Given a solution $s$, we denote by $[e_i, l_i]$ the time window of a request $i \in P$ and by $p_i$ and $d_i$ the times in which $i$ is collected from its pickup location and is delivered to the HUB, respectively. Let $\delta(i)$ be 1 if $i$ is stabilized in a spoke in solution $s$ and 0 otherwise. In the first case, let $sa_i$ be the arrival time of $i$ at the spoke and let $se_i$ be the end of the stabilization.

Then

$$t(s) = \sum_{i \in P} ((d_i - e_i - LT_i)^+ (1 - \delta(i)) + (sa_i - e_i - LT_i)^+ \delta(i) + (d_i - se_i - ET_i)^+ \delta(i)) \quad (1.51)$$

$$w(s) = \sum_{i \in P} (p_i - l_i)^+ \quad (1.52)$$

(Where $(a)^+ = a$ if $a > 0$ and 0 otherwise.)

In $f_2(s)$, the parameters $\alpha$ and $\beta$ are initially set to given values $\alpha_0$ and $\beta_0$ respectively. (In our experiments (Section 1.6) we set $\alpha_0 = 1$ and $\beta_0 = 1$.) infeasible solutions. At Phase 4 of each

iteration of the algorithms, $\alpha$ and $\beta$ are increased (decreased) if lifetime or time windows constrains are violated (not violated) in solution $s$.

### 1.4.3 Phase 1: Destroy

This phase takes as an input a solution $s_{curr}$ (feasible or not) and is composed by two main steps: First, one destroy heuristic is selected among a pool of six different procedures, with a probability depending on its previous performances, as explained in Section 1.4.6; In the second step, the selected heuristic is used to remove a given number of requests from the solution $s_{curr}$, generating a partial solution $s'$.

When a stabilized request is removed from $s_{curr}$, all the information related to the request and to the spoke in which it is stabilized is deleted and all involved routes are updated.

Three of the six destruction heuristics have already been introduced in the literature and have been adapted for the problem under study. They are the *Worst Removal*, the *Random Removal* [26] and the *Related Removal* [29], and are briefly described in the following.

- The Worst Removal heuristic iteratively removes requests that cause the biggest detour in the current solution in a semi-random way. More precisely, a request is removed from the solution with a probability related to the evaluation function improvement produced by removing that request. The randomization is regulated by a parameter $p \geq 1$, the smaller $p$ is, the more randomness is allowed.

- The Random Removal heuristic randomly selects the requests to remove.

- The Related Removal (or Shaw Removal) heuristic aims to remove requests that are more

related with each other in order to give them the chance to be re-routed more efficiently. A specific measure of relatedness between requests have been developed ad-hoc for the problem under study and is described into detail in Section 1.4.3.

The other three heuristics were developed ad-hoc for the problem and are denoted as *Vehicle Removal*, *Spoke Removal* and *Late Requests Removal*. They are briefly described in the following, and in more detail in Sections 1.4.3–1.4.3:

- The Vehicle Removal heuristic removes all the requests of a randomly selected vehicle.
- The Spoke Removal heuristic removes all the requests stabilized at a randomly selected spoke center.
- The Late Requests Removal removes the requests with the higher lifetime violations.

**Related Removal and relatedness function**

As suggested by Shaw [29], the removal of requests that are very different from each other may not help in improving the solution when re-inserting them. In fact the solution obtained after the re-insertion will probably be close to the previous solution, because the requests will likely be re-inserted in their original positions if not even in worse positions. Therefore, the general idea of this heuristic is to remove requests that are somehow related to each other, in order to give them the chance to be re-routed efficiently. Let $R(i, j)$ be the *relatedness function* of requests $i$ and $j$. The bigger $R(i, j)$ is the more related are the two requests. For the problem under study, we developed an ad-hoc measure of relatedness between a pair of requests, which takes into account three pieces of information: the distance between pick-up locations, the (possible) overlap of the

time windows, and whether they are routed on the same vehicle in the current solution or not. More precisely, the relatedness function employed in the Related Removal heuristic is computed as $R(i,j) = 1/(D_{ij} + E_{ij} + V_{ij})$, where $D_{ij}$ is the distance between request $i$ and request $j$ (normalized according to the maximum distance between locations), $E_{ij} = 1/(100(l_i - e_j))$ if $l_i - e_j \geq 0$ and $E_{ij} = 1 + 1/(100(l_i - e_j))$ if $l_i - e_j \leq 0$, and $V_{ij}$ is equal to 1 if the requests are on the same vehicle and 0 otherwise. The value $l_i - e_j$ is a measure of the overlap between the time windows of requests $i$ and $j$. If the two time windows overlap with each other, then $l_i - e_j \geq 0$ and $E_{ij}$ is smaller, otherwise $l_i - e_j \leq 0$ and $E_{ij}$ is bigger. (Recall that, $l_i$ is the end of the time window of request $i$, and $e_j$ is the beginning of the time window of request $j$.) Hence, $R(i,j)$ is bigger when $D_{ij}$, $E_{ij}$ and $V_{ij}$ are smaller. Observe that, two requests are more related when are not on the same vehicle (i.e., $V_{ij} = 0$), but have close pick-up locations and similar time windows. Hence, requests with higher relatedness are those currently in different vehicles, but that could be more appropriately assigned to the same route.

**Vehicle Removal**

The Vehicle Removal heuristic simply removes a route from the solution. In this way, the requests carried on the removed vehicle have the chance to be re-routed with another vehicle, possibly through a spoke center. In fact, since the stabilization increases the lifetimes of the requests, it is often possible to reduce the number of used vehicles by routing some requests through a spoke center.

More precisely, the heuristic deletes from the solution the route with the largest total violation of the time windows and lifetime constraints is removed. If all routes in the current solution are

feasible, then a random route is removed. Even if the requests will not be reassigned in already existing routes, by removing the route with the largest violations, after the re-insertion of the removed requests the solution will likely be better than before. Note that the number of requests removed by the heuristic is not known *a priori* but depends on the route selected.

**Spoke Removal**

This heuristic is inspired from a neighborhood for a pickup and delivery problem with transfers proposed in [18]. The idea is to remove all the requests using a given spoke center, in order to give them the chance to be re-routed through a different spoke center or without stabilization. Spoke Removal first selects a random spoke among the active ones in the solution. If the number of requests using that spoke is less than or equal to the number of requests to be removed, all these requests are removed from the solution. As a consequence, the spoke is also deleted from the current solution. If the number of requests using the selected spoke is greater than the number of requests to be destroyed, only a subset of requests is removed. This subset is iteratively generated as explained below. At the first iteration, the first request of the subset is selected randomly among the requests using the spoke center and is removed. Then, at each iteration, an already removed request $i$ is randomly selected and the next request to remove, say $j$, is chosen among the requests using the spoke center, with a probability increasing with the proximity of the pickup locations of $i$ and $j$. The randomization is regulated by a parameter $p \geq 1$, the bigger $p$ is, the more randomness is allowed.

**Late Requests Removal**

This heuristic aims at reducing the violations of the lifetime constraints of the current solution. It removes the requests with the highest lifetime violations, if any, in order to give them the chance to be re-routed with no lifetime violation or at least a smaller one. If the solution has no late requests, this heuristic cannot be selected by the ALNS. On the other hand, if the number of late requests of the current solution is smaller than the number of requests to remove, the heuristic removes the late requests only.

### 1.4.4 Phase 2: Repair

This phase takes as an input the partial solution $s'$ generated in Phase 1 and a set of unplanned requests. The phase is divided into two main steps: First, one repair heuristic is chosen among a pool of seven different procedures with a probability depending on its previous performances, as explained in Section 1.4.6; In the second step, the selected heuristic is used to insert all the unplanned requests in the partial solution $s'$, generating a new solution $s_{new}$.

In the algorithms, we implemented both repair heuristics that do not consider stabilization and heuristics in which the use of stabilization at spokes centers is evaluated for the insertion. The five following heuristics, denoted as *Best Insertion* and *Regret-k* (with $k = 2, \ldots, 5$), do not exploit the stabilization and have been already successfully employed in the literature:

- Best Insertion is a construction heuristic in which the insertion cost of each unplanned request is calculated for each possible position. Then, the request with the minimum insertion cost is inserted in its best position. The heuristic terminates when all the unplanned requests are added to the solution [26].

- The Regret-$k$ heuristic have been used for example by Potvin and Rousseau [24] and by Ropke and Pisinger [23] for the Vehicle Routing Problem with Time Windows. Let $R$ be the set of unplanned requests and, for each request $i \in R$, let $\Delta f_i^p$ be the lowest cost of inserting $i$ in the $p$-th best route at the best position. In the Regret-$k$ heuristic, the following procedure is iteratively performed until all the requests are inserted. At each iteration, the request selected for insertion at its best position is

$$i^* = argmax_{i \in R}\{\sum_{p=1}^{k}(\Delta f_i^p - \Delta f_i^p)\},$$

where $\sum_{p=1}^{k}(\Delta f_i^p - \Delta f_i^p)$ is the *regret value* of request $i$. In our algorithm, we consider Regret-$k$ heuristics with values of $k$ between 2 and 5, resulting in four different heuristics.

The following two heuristics have been developed to evaluate the insertion of requests both with and without stabilization at the spoke centers. These heuristics, denoted as *Best Insertion with spokes* and *Best Request-spoke Insertion*, are adaptations of repair heuristics proposed for the pick-up and delivery problem with transfers [18; 19].

**Best Insertion with spokes**

This heuristic evaluates the insertion of each unplanned request both with stabilization (i.e., by delivering it to a spoke center) and without stabilization (i.e., by delivering it directly to the HUB). More precisely, for each unplanned request $i$, first the best insertion cost without the use of stabilization is calculated. Then, the best insertion with stabilization of the request is evaluated. At this aim, the stabilization of request $i$ at a given spoke is modeled by adding two new distinct nodes:

the delivery spoke node, where the request is delivered for the stabilization, and the pick-up spoke node, where the request is picked up after the stabilization. Obviously, the delivery spoke node must be visited before the pick-up spoke node. The best insertion with stabilization is evaluated as follows. For each spoke, we first evaluate the best positions of the pair pick-up node (of the request $i$) and of the delivery spoke of $i$. Of course the two will be on the same route. Then, the best insertion for the pick-up spoke of request $i$ is computed. Note that, recalling that transfers are allowed at the spoke centers, the delivery spoke node and the pick-up spoke node could be inserted in different routes. The best insertion with stabilization is the one involving the spoke that gives the smallest overall insertion cost. On the other hand, the best insertion without stabilization of the request $i$ is evaluated as in the Best Insertion heuristic. Finally, the insertion costs of $i$ with or without stabilization are compared and the insertion with the smallest cost is performed.

**Best Request-spoke Insertion**

The Best Request-spoke Insertion heuristic inserts each unplanned request with stabilization. In fact, even if in some cases the insertion without stabilization could be cheaper, in terms of traveled distance, the insertion of one or more spokes in the solution may help future insertion operations.

More precisely, for each unplanned request $i$, the heuristic evaluates and performs the best insertion with stabilization as in the heuristic Best Insertion with spokes. It is important to notice that the stabilization of unplanned requests can be performed in spokes already used in the solution.

### 1.4.5 Phase 3: Evaluate

This phase takes as an input the solution $s_{new}$ obtained in Phase 2. As already mentioned, if $s_{new}$ is feasible and better than the best known feasible solution, it is kept as the new best feasible solution. Similarly, if $s_{new}$, not necessarily feasible, is better than the best known solution, it is kept as the new best solution. Then, the solution $s_{new}$ is evaluated to check whether it should be accepted as the new current solution for the next iteration. More precisely, the current solution is updated if $s_{new}$ meets given *acceptance criteria*. To escape local minima, we employ the *simulated annealing* acceptance criterion used in [26].

### 1.4.6 Phase 4: Update

This phase is composed of two main steps. First, the *penalty parameters* used in the evaluation function are updated. Recall from Section 1.4.2 that these parameters are denoted as $\alpha$ and $\beta$, and that they are initially set to the values $\alpha_0 = 1$ and $\beta_0 = 1$. As already mentioned in Section 1.4.2, the parameter $\alpha$ regulates the weight that total lifetime violation have on the evaluation function, while $\beta$ is used to weigh the total time windows violation. If the solution $s_{new}$ resulting from Phase 3 is feasible, $\alpha$ and $\beta$ are decreased. On the other hand, if $s_{new}$ violates both lifetime and time windows constraints, then $\alpha$ and $\beta$ are increased, in order to further penalize the violations in the objective function. Finally, if $s_{new}$ only violates lifetime constraints (time windows constraints) parameter $\alpha$ is increased and parameter $\beta$ is decreased (parameter $\beta$ is increased and parameter $\alpha$ is decreased).

In the second step of Phase 4, the *adaptivity parameters* are updated, which are used to adapt the algorithm to the performances of the heuristics. These parameters are only updated at a end of each block of $I$ consecutive iterations, called *segment*. Each heuristic has two *adaptivity parameters*: the

*weight* and the *score*. Recall that, at the beginning of Phase 1 (Destroy) and Phase 2 (Repair), the heuristics to employ are selected with a probability that depends on their historical performances during the previous iterations.

As proposed in [26], the selection is made using the *roulette wheel selection principle* where, at each iteration, the heuristic $j$ is chosen with probability

$$\frac{w_j}{\sum_{i=1}^{k} w_i},$$

where $w_j$ is the *weight* of heuristic $j$. These weights are updated according to the performances of the heuristics. As in [26], we keep track of these performances by assigning *scores* to the heuristics. At the beginning of the iterative core of the algorithms, all heuristics have the same scores and weights. At each iteration, in Phase 4, the scores of the destroy and repair heuristics selected in Phases 1 and 2 are increased by $\sigma_1$, $\sigma_2$ or $\sigma_3$, as explained below.

Let $s_{new}$ be the solution produced by the pair of destroy and repair heuristics used in the last iteration. If $s_{new}$ is better than the best known solution, their score is increased by $\sigma_1$; if $s_{new}$ has not been accepted yet and is better than the current solution, then the scores are increased by $\sigma_2$; finally, the scores are increased by $\sigma_3$ if $s_{new}$, completely new, has not been accepted yet and is worse than the current solution. The scores of the destroy and repair heuristics used during an iteration are updated equally, since we cannot give the credit for the new solution to just one between the destroy or repair heuristic used. In our experiments, we set $\sigma_1$, $\sigma_2$ and $\sigma_3$ as in [26]. As already mentioned, the iterative core of the algorithms is divided into *segments* of $I$ consecutive iterations (we set $I = 100$ in our experiments). At the beginning of the first segment all the heuristics have

the same weight.

At the end of each segment, Phase 4 updates the weight of each heuristic, on the basis of the score gained by that heuristic during the segment, and sets all the scores to zero.

More precisely, the weight that will be used for heuristic $i$ in the segment $j+1$ reads as follows,

$$w_{i,j+1} = w_{ij}(1-r) + r\frac{\pi_i}{\theta_i}$$

where $\pi_i$ is the score obtained by the the $i$-th heuristic during the last segment and $\theta_i$ is the number of times the heuristic was used. Parameter $r$ controls how fast the algorithm reacts to the weight change. If $r = 0$ the same weights will be used during all the algorithm, instead if $r = 1$ the scores attained in the last segment will entirely determine the weights to use in the following segment.

### 1.4.7 Phase 5: Post-Segment Procedure

This phase is executed at the end of each segment of $I$ iterations. A procedure, called *Spoke Insertion* heuristic, is executed with the aim of reducing lifetime violations of the current solution $s$, if any. Spoke Insertion tries to stabilize some requests in order to remove or reduce lifetime violations. For each route, the last request $r_l$ violating the lifetime constraints and not already stabilized in a spoke center is detected, if any. Afterward, the spoke center $j$ that minimizes the sum of the distances to the pick-up node of $r_l$ and to the subsequent node in the route is chosen. The spoke is then inserted in the route right after the pickup-node of $r_l$. In practice, the insertion of $j$ in the route may generate new lifetime violations in the requests that were picked up by the vehicle before $r_l$. Hence, also such requests, if not already stabilized in a spoke, are re-routed

through $j$ for stabilization. Once all the routes with lifetime violations have been processed, the Spoke Insertion procedure terminates and a check on the output solution is performed. If the output solution is better than the input solution (i.e., the current solution $s$) in terms of evaluation function, it is taken as a new current solution. Otherwise, the current solution is not changed. Furthermore, if the solution produced by Spoke Insertion is also better than the best known solution (best known feasible solution), the best solution (best known feasible solution) is updated.

### 1.4.8 Initial Solution

Different Large Neighborhood Search (LNS) procedures have been used to produce an initial solution with small lifetime violations. In fact, for the problem under study, we observed that a too high total lifetime violation of the initial solution may negatively affect the performances of the overall algorithm. At this aim, different starting solutions are generated by employing each of the repairing heuristics described in Section 1.4.4. In this case, the repair heuristics are used as constructive heuristics for routing all the requests to the vehicles. Then, a 5-iterations LNS is applied to each starting solution generated as described above, with the aim of reducing the lifetime violations, if any. More precisely, at each iteration of the LNS procedures, the destruction heuristic Late Requests Removal (see Section 1.4.3) is first applied to remove late requests. Then, the solution is repaired with an heuristic randomly chosen among the pool of repairing heuristics and, finally, the post processing procedure Spoke Insertion (introduced in Section 1.4.7) is applied. At the end of each iteration, the new generated solution is kept only if it is better than the previous solution. In a preliminary experimental campaign, it has been observed that these small LNS procedures are able to sensibly reduce the lifetime violations of the starting solutions. Obviously,

the LNS procedures are not applied to starting solutions with no lifetime violation. After applying the LNS to each starting solution with lifetime violations, the best solution obtained so far is chosen as initial solution of the ALNS algorithm.

## 1.5 Real-world data and instance description

In 2011, the ASL (Local Healthcare Authority) of Bologna, Italy, began a reorganization involving all processes related to the management of blood and biological samples collected in the draw labs of the metropolitan area of Bologna. In this context, in 2015, a single Laboratory for the analysis of all samples was established, called "Laboratorio Unico Metropolitano" (or HUB), located in Bologna.

Currently, the metropolitan area of Bologna includes 46 blood draw labs, with different opening days and hours (ranging from 7 am to 10 am, from Monday to Saturday). Real data show that, during working hours, a new sample is drew from a patient approximately every 3 minutes in each draw lab. In Table 1.1, the number of draw labs opened each day and the related average opening hours (in minutes) are given. Each sample must be delivered to the main hospital or to a spoke center within 120 minutes (i.e., the sample lifetime is 120 minutes). For the transportation 26 and 16 vehicles are available on weekdays and on Saturday, respectively, located in 8 different depots. Each day, the stabilization process can be performed in 12 spoke centers and requires about 30 minutes. The extra lifetime gained by a stabilized sample is of 90 minutes (starting from the end of the stabilization process). The service time required by a vehicle to load or unload samples at a draw labs or a spoke is about 10 minutes.

Figure 1.2 reports the geographical distribution of the draw labs, spokes and depots in the

| | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
|---|---|---|---|---|---|---|
| # Draw labs | 31 | 33 | 30 | 36 | 30 | 16 |
| Av. Time (min) | 100.65 | 100.00 | 100.00 | 99.17 | 100.00 | 108.75 |

Table 1.1: Number of draw labs and average opening hours (in minutes).

metropolitan area of Bologna. More precisely, each area highlighted in grey contains at least one of the above mentioned centers. The location of the main hospital (HUB), denoted by $H$, is also shown.

Small and large instances have been generated from real data. More precisely, 10 small instances have been produced in order to compare the performances of the MILP formulations introduced in Section 1.3 and the H-ALNS algorithms. Furthermore, 32 large instances have been generated for solving the real problem. In the small instances, the number of transportation requests (i.e., samples) and vehicles range from 5 to 20 and from 3 to 10, respectively, while the number of spokes is one or two. The details of the ten small instances are reported in Table 1.2, where Column 2 reports the time step used in the two time-indexed formulations $MILP_1$ and $MILP_2$ (i.e., one or five minutes). Obviously, the bigger the time step is, the smaller the number of variables of the formulations is. Columns 3–5 respectively report the number of requests $|R|$, vehicles $|K|$ and spokes $|S|$ of the instances. These instances have been designed to also test the different transportation modes of the samples (i.e., directly to the main hospital, or first to a spoke and then to the hospital). For example, in Instance 2 a feasible solution can be obtained only by using stabilization at the spokes, while in Instance 7 an optimal solution exists that does not require sample stabilization.

In the large instances, all vehicles, draw labs and spokes of the real case have been considered. Furthermore, since the daily number of samples to manage is extremely high (i.e., more than 500 and 1000 on Saturday and weekdays, respectively), different instances have been generated in which

samples have been grouped in batches according to different time spans. More precisely, samples have been grouped in batches every either 30, 45 or 60 minutes of activity of a draw lab. Hence, as an example, recalling that a biological sample is produced every 3 minutes at each draw center, by grouping them using a time span of 30 minutes, we get batches containing about 10 samples. The earliest limit of the time window assigned to each batch is set equal to the production time of the youngest sample of the batch. The latest limit of the time window is always set to 30 minutes after the closing time of the draw lab. Thus, the time window width of a batch varies depending on the batch dimension.

| Id | Time step | $\|R\|$ | $\|K\|$ | $\|S\|$ |
|---|---|---|---|---|
| 1 | 1 | 5 | 3 | 1 |
| 2 | 1 | 8 | 3 | 1 |
| 3 | 1 | 9 | 4 | 1 |
| 4 | 5 | 9 | 4 | 1 |
| 5 | 5 | 12 | 6 | 1 |
| 6 | 5 | 12 | 6 | 2 |
| 7 | 5 | 12 | 7 | 2 |
| 8 | 5 | 15 | 8 | 1 |
| 9 | 5 | 15 | 9 | 2 |
| 10 | 5 | 20 | 10 | 2 |

Table 1.2: Characteristics of the small instances.


((a)) Spoke Centers


((b)) Depots


((c)) Draw labs

Figure 1.2: Locations of draw centers, spoke Centers and Depots.

The lifetime assigned to each batch is set to 120 minutes after the production time of the oldest

sample of the batch, i.e., the sample of the batch produced first. Hence, if a batch is delivered within its lifetime to the HUB, every sample contained in the batch is delivered on time, too. Note that, the bigger the time span used for grouping the samples is, the shorter the lifetimes of the batches are, since the cardinality of each batch is bigger.

In general, given a time span, the number of batches generated at a draw lab may vary according to the opening hours of the lab and, if the time span and the opening hours are not divisible numbers, to the rounding procedure used. Formally, let $ts$ be the time span used and $o_k$ be the opening hours (in minutes) of the draw lab $k$. If $ts$ does not divide $o_k$, the batches generated may be $\lceil o_k/ts \rceil$ or $\lfloor o_k/ts \rfloor$. In the first case (second case), the last batch generated at each draw center may contain a smaller (bigger) number of samples, resulting in a batch with a longer (shorter) lifetime. As an example, let us assume that the draw lab $k$ has a opening hours $o_k = 300$ minutes and that $ts = 45$ minutes. Hence, $\lceil o_k/ts \rceil = 7$ and $\lfloor o_k/ts \rfloor = 6$. Recalling that a sample is approximately produced every 3 minutes, in the first case, we get 7 batches: the first 6 batches have a lifetime of 78 minutes and contain 15 samples each, while the last batch contains 25 samples and has a shorter lifetime of 48 minutes. In the second case, we get 6 batches: the first 5 batches contain 15 samples and have a lifetime of 78 minutes, and the last batch contains 10 samples and has a lifetime of 93 minutes.

According to the above criteria, for each time span $ts \in \{30, 45, 60\}$, we generated two sets of large instances, denoted as Set $A$ and Set $B$, containing $\lceil o_k/ts \rceil$ and $\lfloor o_k/ts \rfloor$ batches for each draw lab $k$, respectively. Table 1.3 shows the number of requests/batches in the instances of Set $A$ and Set $B$, generated by a time span $ts$ of 60, 45 and 30 minutes, for each day of the week. As already stated, instances of Set $A$ generally contain a bigger number of batches than the instances of Set $B$, but possibly with longer lifetimes. Furthermore, the number of requests varies according to the day

of the week, since the opening hours of the draw labs depend on the day. Finally, note that, when a time span of 30 minutes is considered, the number of requests in the instances of Set *A* and Set *B* related to Monday, Wednesday, Friday and Saturday are the same, resulting in exactly the same instances. Hence, sets *A* and *B* contain 32 different instances.

| Set | Time Span ($ts$) | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
|---|---|---|---|---|---|---|---|
| $A_1$ | 60 | 61 | 65 | 60 | 71 | 59 | 34 |
| $A_2$ | 45 | 76 | 81 | 72 | 87 | 73 | 42 |
| $A_3$ | 30 | 104 | 111 | 100 | 120 | 100 | 58 |
| $B_1$ | 60 | 43 | 45 | 40 | 48 | 41 | 24 |
| $B_2$ | 45 | 62 | 66 | 61 | 72 | 60 | 35 |
| $B_3$ | 30 | 104 | 109 | 100 | 118 | 100 | 58 |

Table 1.3: Number of requests/batches per time span and day in instances of Sets *A* and *B*.

## 1.6 Experimental Results

In this section, the experimental results on the instances described in the previous section are presented. All tests have been performed on a PC equipped with Intel i5 processor and 8 Gb of RAM. This section is organized as follows. In Section 1.6.1, a tuning experimental campaign devoted to properly set parameters and some components of the H-ANLS algorithms is presented. In Sections 1.6.2 and 1.6.3 , the computational results on small and large instances are respectively reported.

### 1.6.1 Tuning of the H-ALNS metaheuristic

In the tuning experimental campaign, two test phases have been carried out. In the first test phase, preliminary experiments have been performed to set the main parameters of the algorithms. This phase showed that most of the parameters proposed in [26] performs well, i.e., $\sigma_1 = 33$,

$\sigma_2 = 9$, $\sigma_3 = 19$, $C_{rate} = 0.99975$ and $w = 0.05$. On the other hand, in this phase, it turned out that the performances of the H-ALNS algorithms improve when the percentage of requests removed and reinserted at each iteration, denoted as *rem* in the following, depends on the number of the transportation requests $|R|$. In fact, we observed that the algorithm benefits of large values of *rem* when $|R|$ is small and (sensibly) smaller *rem* values for larger $|R|$. As also noted in [26], such a behavior could be due to the fact that insertion heuristics work fairly well when they must insert a limited number of requests into a partial solution (while, when a fixed *rem* is used, the number of requests to insert becomes large when $|R|$ is large). As a consequence, in our experiments we set *rem* as follows: $rem = 0.7$ if $|R| \leq 20$, $rem = 0.3$ if $20 < |R| \leq 40$ and $rem = 0.05$ if $|R| > 40$. Another aspect affecting the performances of the algorithm is how the weights assigned to each heuristic change during the algorithm. In the algorithm, such a feature is controlled by parameter $r \in (0,1)$: the smaller $r$ is the more importance is given to the whole performance history of the heuristics during the algorithm when updating their weights. In the first test phase, it turned out that the algorithms attain better performances when $r$ is high. As a consequence, we set $r = 0.8$ in the second test phase of the tuning experiments.

In the second test phase, a set of experiments has been performed to evaluate how some new ad-hoc developed components affect the performances of the H-ALNS algorithms. In particular, we test different variants of the insertion heuristics *Best Insertion with spokes* and *Best Request-spoke Insertion* (see Section 1.4.4), designed to insert requests both with and without stabilization at the spoke centers, and the procedure called at the end of each segment (see Section 1.4.7). Furthermore, in this second phase, we also evaluated H-ALNS configurations in which the weights of all the heuristics are re-initialized to their starting values at given points during the algorithm

(similarly to what is done with the scores gained by each heuristic, that are set to 0 at end of each segment) and configurations using the Post Segment Procedure (Section 1.4.7) or not.

The different variants of the insertion heuristics *Best Insertion with spokes* and *Best Request-spoke Insertion* have been developed in order to evaluate how different accuracy levels in the insertion of the requests affect the algorithm performances. In fact, since the requests can be transferred from one vehicle to another at the spoke centers, a change in a route may produce changes in other routes (called *affected routes*). As an example, let us suppose that request $i$ is delivered by vehicle $k$ to a spoke $sp$ and is picked up at $sp$ by a vehicle $q$ after the stabilization. Then, a change in the route $k$ may lead to request $i$ ending the stabilization process at spoke $sp$ earlier or later than before, implying a possible timing change in the route $q$. In practice, taking into account the changes on the affected routes during the insertion of requests increases the accuracy of local insertion decisions but, at the same time, increases the complexity of the heuristics. More precisely, the following two variants of the heuristics *Best Insertion with spokes* and *Best Request-spoke Insertion* have been considered:

- Variant 1 ($V_1$): The changes in the affected routes are never taken into account during the evaluation of the insertion cost of a request. Hence, heuristics evaluate the insertion of a request in a route $k$ considering route $k$ only.

- Variant 2 ($V_2$): The changes in the affected routes are always taken into account during the insertion of a request. Hence, the insertion of a request in a route takes into account how that insertion affects other routes.

The two variants have been evaluated either alone or together. Hence, three algorithm configurations have been developed, two in which either variant $V_1$ or $V_2$ has been used for the heuristics,

| Configurations<br>Features | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ | $C_9$ | $C_{10}$ | $C_{11}$ | $C_{12}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $V_1$ | X | X | X | X | | | X | | X | | X | X |
| $V_2$ | X | X | | | X | X | | X | | X | X | X |
| *PostSeg* | | | | | | | X | X | X | X | X | X |
| $WI$ | X | | | X | X | | | X | X | | | X |

Table 1.4: Algorithm configurations of the second test phase.

and one in which both $V_1$ and $V_2$ have been included in the algorithm as repair heuristics.

In total, 12 algorithm configurations have been evaluated in the second tuning phase. Table 1.4 summarizes the different configurations components. In the table, *PostSeg* is the post segment procedure presented in Section 1.4.7, and $WI$ is the re-initialization of the weights of the heuristics to 1 after $It_w$ iterations, where $It_w$ has been set to 1000 in the experiments.

The configurations have been tested on 4 real-world instances, i.e., Wednesday and Thursday of set $A_2$, and Friday and Saturday of Set $A_3$. Five runs of the algorithm have been performed for each setting. Hence, 240 experiments have been performed in total. The maximum number of iterations, $It_{max}$, has been set to $20,000$ in all the runs. The main indicators used to evaluate the performances of the different configurations are the number of feasible solutions found, the solution values, and the computation time. The results are reported in Tables 1.5 and 1.6, where "Av" and "best" report the average and the best solution values found over the 5 runs, "# feas" is the number of times a feasible solution is found (out of the 5 runs), and $t$ is the average computational time in seconds. As the results show, not all configurations are able to find feasible solutions on some instances, i.e., Thursday of set $A_2$ and Friday of Set $A_3$.

| | Wednesday $A_2$ | | | | Thursday $A_2$ | | | |
|---|---|---|---|---|---|---|---|---|
| Conf. | Av | best | # feas | $t$ | Av | best | # feas | $t$ |
| $C_1$ | 1787.80 | 1703 | 5 | 465.38 | 2402 | 2257 | 5 | 786.28 |
| $C_2$ | 2112.50 | 2068 | 2 | 187.80 | - | - | 0 | 196.47 |
| $C_3$ | 2074.67 | 2070 | 3 | 199.82 | - | - | 0 | 173.27 |
| $C_4$ | 1716.80 | 1644 | 5 | 317.73 | 2307 | 2229 | 2 | 496.53 |
| $C_5$ | 1933.20 | 1883 | 5 | 988.10 | - | - | 0 | 1444.13 |
| $C_6$ | 2247.50 | 2198 | 2 | 812.70 | - | - | 0 | 788.77 |
| $C_7$ | 2159 | 2070 | 4 | 175.04 | - | - | 0 | 234.28 |
| $C_8$ | 1938.60 | 1762 | 5 | 836.94 | 2614 | 2614 | 1 | 1698.52 |
| $C_9$ | 1800.60 | 1644 | 5 | 315.51 | 2466.80 | 2216 | 5 | 491.38 |
| $C_{10}$ | 2195 | 2160 | 2 | 191.18 | - | - | 0 | 257.16 |
| $C_{11}$ | 2195 | 2160 | 2 | 191.16 | - | - | 0 | 257.42 |
| $C_{12}$ | 1858.40 | 1722 | 5 | 439.48 | 2437.33 | 2358 | 3 | 714.46 |

Table 1.5: Tuning results on instances Wednesday and Thursday of Set $A_2$.

| | Friday $A_3$ | | | | Saturday $A_3$ | | | |
|---|---|---|---|---|---|---|---|---|
| Conf. | Av | best | # feas | $t$ | Av | best | # feas | $t$ |
| $C_1$ | 1980.60 | 1816 | 5 | 668.59 | 1227.20 | 1165 | 5 | 236.05 |
| $C_2$ | - | - | 0 | 258.59 | 1437.40 | 1250 | 5 | 165.21 |
| $C_3$ | - | - | 0 | 185.12 | 1465 | 1373 | 5 | 153.21 |
| $C_4$ | 2051.20 | 2022 | 5 | 296.37 | 1234.20 | 1106 | 5 | 119.29 |
| $C_5$ | 2148.50 | 1984 | 4 | 964.46 | 1274.60 | 1221 | 5 | 291.32 |
| $C_6$ | - | - | 0 | 757.08 | 1439 | 1318 | 5 | 545.66 |
| $C_7$ | - | - | 0 | 295.75 | 1420.80 | 1309 | 5 | 150.04 |
| $C_8$ | 2059 | 1866 | 5 | 1383.01 | 1205 | 1148 | 5 | 338.93 |
| $C_9$ | 1999.60 | 1848 | 5 | 329.85 | 1217.60 | 1142 | 5 | 114.72 |
| $C_{10}$ | - | - | 0 | 249.40 | 1462.20 | 1407 | 5 | 547.09 |
| $C_{11}$ | - | - | 0 | 250.17 | 1440.60 | 1250 | 5 | 160.44 |
| $C_{12}$ | 2052.20 | 1928 | 5 | 685.63 | 1181.80 | 1053 | 5 | 188.42 |

Table 1.6: Tuning results on instances Saturday and Friday of Set $A_3$.

The results of the second tuning phase are summarized in Table 1.7. They show that $C_1$ and $C_9$ are the only configurations able to find feasible solutions in all the runs. Regarding the quality of the solutions, both $C_1$ and $C_9$ attain solutions of good quality when compared to the other configurations. Note that $C_1$ requires higher computational times than $C_9$, on average, since it employs the time consuming spoke insertion heuristics of variant $V_2$.

| Conf. | Av | best | # feas | $t$ |
|---|---|---|---|---|
| $C_1$ | 1849.40 | 1735.25 | 5 | 539.08 |
| $C_2$ | 1774.95 | 1659 | 1.75 | 202.02 |
| $C_3$ | 1769.83 | 1721.50 | 2 | 177.85 |
| $C_4$ | 1827.30 | 1750.25 | 4.25 | 307.48 |
| $C_5$ | 1785.43 | 1696 | 3.50 | 922 |
| $C_6$ | 1843.25 | 1758 | 1.75 | 726.05 |
| $C_7$ | 1789.90 | 1689.50 | 2.25 | 213.78 |
| $C_8$ | 1954.15 | 1847.50 | 4 | 1064.35 |
| $C_9$ | 1871.15 | 1712.50 | 5 | 312.87 |
| $C_{10}$ | 1828.60 | 1783.50 | 1.75 | 311.21 |
| $C_{11}$ | 1817.80 | 1705 | 1.75 | 214.80 |
| $C_{12}$ | 1882.43 | 1765.25 | 4.50 | 507 |

Table 1.7: Average results of the tuning phase.

In the following sections, computational experiments are presented in which configurations $C_1$ and $C_9$ are tested on all benchmark instances.

### 1.6.2 Results on small instances

The MILP formulations and the H-ALNS algorithms selected in the tuning phase have been tested on the ten small instances introduced in Section 1.5. As already mentioned, these instances have been generated in order to test the formulations on different aspects of the problem, e.g., sample stabilization, multiple visits of the vehicles at spoke centers. The mathematical formulations have been solved by Gurobi 9.0 on a 3.2 GHz computer with 12 cores, 64 Gb of RAM with a time limit of 16 hours. The H-ALNS algorithms $C_1$ and $C_9$ have been executed on a 2.5 GHz Quad-core processor and 16 Gb of RAM with $It_{max} = 20,000$. It is worth noting that $MILP_3$ was not able to find any feasible solution within the time limit in all the 10 small instances, suggesting that the time-indexing introduced in $MILP_1$ and $MILP_2$ is more effective to deal with multiple visits. As a consequence, Table 1.8 reports a summary of the results of $MILP_1$, $MILP_2$ and of the H-ALNS algorithms. In the table, for each instance, Columns 2–4 and 5–7 respectively report the results related to $MILP_1$ and $MILP_2$. More precisely, "LB" is the lower bound found at the root

| | $MILP_1$ | | | $MILP_2$ | | | $C_1$ | | | | $C_9$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| id | LB | Sol | $t$ | LB | Sol | $t$ | Av. | best | # Opt. | $t$ | Av. | best | # Opt. | $t$ |
| 1 | 227 | 227 | 601.53 | 184 | 227 | 526.83 | 227 | 227 | 5 | 2.42 | 227 | 227 | 5 | 2.58 |
| 2 | 366 | 366 | 1.024.73 | 322 | 366 | 1,171.99 | 366 | 366 | 5 | 6.99 | 366 | 366 | 5 | 5.86 |
| 3 | 385 | 385 | 1,862.24 | 385 | 385 | 1,432.54 | 385 | 385 | 5 | 3.11 | 385 | 385 | 5 | 3.26 |
| 4 | 328 | 328 | 19.02 | 260 | 328 | 19.14 | 328 | 328 | 5 | 4.32 | 328 | 328 | 5 | 4.87 |
| 5 | 572 | 572 | 39.11 | 420 | 572 | 55.86 | 572 | 572 | 5 | 4.73 | 572 | 572 | 5 | 6.42 |
| 6 | 572 | 572 | 187.8 | 311 | 572 | 57600* | 572 | 572 | 5 | 6.82 | 572 | 572 | 5 | 7.13 |
| 7 | 572 | 572 | 231.62 | 322 | 572 | 57600* | 572 | 572 | 5 | 7.17 | 572 | 572 | 5 | 9.67 |
| 8 | 569 | 569 | 87.03 | 569 | 569 | 90.2 | 569 | 569 | 5 | 7.57 | 569 | 569 | 5 | 11.35 |
| 9 | 528 | 528 | 604.88 | 390 | 638 | 57600* | 528 | 528 | 5 | 10.62 | 533.6 | 528 | 3 | 17.09 |
| 10 | 302 | 302 | 1,463.58 | 211 | 302 | 42,088.83 | 307.4 | 302 | 1 | 71.92 | 305 | 302 | 2 | 182.77 |

Table 1.8: Results on the small instances.

node, “Sol” is the value of the best solution and ”time” is the computation time in seconds. A “*” indicates that Gurobi was not able to certify the optimality of the solution within the time limit. Finally, Columns 8–10 show the results of the H-ALNS algorithms, in which, for each instance, “Av.” is the average solution found over the 5 runs of the algorithms, “# Opt.” is the number of times the optimal solution is found out of the five runs and “time” is the average computational time in seconds.

The results show that $MILP_1$ is able to find the optimal solution in all the small instances. In most of the cases the optimality is certified at the root node. The computation time required by $MILP_1$ generally depends on the instance dimension, and ranges from a few dozens of seconds to about half an hour. It also seems to depend on other characteristics of the instances. In fact, when the optimal solution requires multiple visits of a vehicle at a spoke center, the computation time seems to be higher. Also, the number of spokes seems to affect the computation time more than the number of requests or vehicles. $MILP_2$ performs similarly on the 5 smallest instances but, in general, as the size of the instances increases, it starts to struggle in finding the optimal solution. In three of the ten instances, $MILP_2$ was not able to certify the optimal solution within the time limit. In two of these cases the best solution found is actually the optimal solution, but the optimality gap

was not filled up. Such a behavior could be mainly due to the strength of the linear relaxation: In fact, the best lower bound found at root node of $MILP_1$ always equals the optimal solution value, while the one provided by $MILP_2$ is generally smaller. As for the H-ALNS algorithms, they are able to find the optimal solution on all the instances within the five runs with computational times ranging from 3 to 180 seconds. Instance 10 (and 9 for $C_9$) results the hardest instance, on which $C_1$ and $C_9$ are able to find the optimal solution in only one and two runs out of five, respectively. However, as the average shows, even the not optimal solutions are in general very close to the optimum.

### 1.6.3 Computational results on real-life instances

In this section, the experimental results of the H-ALNS algorithms on the real-life instances of Sets $A$ and $B$ are presented. We point out that, even on the smallest instances of these sets, Gurobi running on $MILP_1$ and $MILP_2$ was not able to find a feasible solution in 16 hours of computation.

Tables 1.9 and 1.10 report the results on the two sets. We recall that the instances of Monday, Wednesday, Friday and Saturday of Sets $A_3$ and $B_3$ are the same. On each instance, five runs have been performed employing the parameters and the two algorithm configurations selected in the tuning phase presented in Section 1.6.1, i.e., $C_1$ and $C_9$. The maximum number of iterations $It_{max}$ has been set to 20,000 in all the experiments.

The objective of the experimental campaign is twofold: (1) to assess the ability of the H-ALNS algorithms to tackle big real-life instances; (2) to evaluate the best policy for batching samples in such a way that all samples can be delivered on time. Recall that, instances of Sets $A$ and $B$ present different characteristics: instances in Set $A$ usually have a bigger number of requests, but

with longer lifetimes, while instances in Set $B$ are smaller but with shorter lifetimes of the requests. We point out that the algorithm presented in [8] is not able to find any feasible solution (i.e., in which all samples are delivered on time) in all the instances of Sets $A$ and $B$, except for instances of Saturday of Set $A_3$ (and $B_3$), on which it attains solutions of worse quality (with total distance about 58% bigger) than the solutions found by algorithms $C_1$ and $C_9$. Tables 1.9 and 1.10 show the results of the H-ALNS algortihms $C_1$ and $C_9$ on the instances of Sets $A$ and $B$, respectively. In the tables, "Av" and "best" are the average and the best travel distance, respectively, of the solutions found on the 5 runs of the algorithms, "#feas." is the number of times, out of 5, a feasible solution is found, and $t$ is the average computational time in seconds. A "-" in "Av" and "best" indicates that no feasible solution is found in the five runs. The last row of each table reports on average results.

First of all, note that the H-ALNS $C_1$ and/or $C_9$ are able to find feasible solutions (in which lifetime and time windows requirements are satisfied) for all the instances of the Sets $A_1$, $A_2$, $B_2$ and $B_3$. In instances of Set $A_3$ the algorithms find feasible solutions except for the biggest instances, i.e., Thursday and Tuesday. On the other hand, the H-ALNS algorithms are not able to find feasible solutions for the instances of Set $B_1$, except for Saturday (for which they find the best feasible solutions for this day). However, note that the instances in Set $B_1$ contain batches with very short lifetimes, so that the existence of a feasible solution is not guaranteed.

The two H-ALNS configurations attain similar results in terms of solution quality, with $C_1$ ($C_9$) performing slightly better than $C_9$ ($C_1$) on instances of Set $B$ (Set $A$). On the other hand, $C_9$ requires in general shorter computational times than $C_1$: about 300 secs and 490 secs on average for $C_1$ and $C_9$, respectively.

Regarding the batching policy, the results show that the most reliable policy for grouping sam-

| | | $C_1$ | | | | $C_9$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| *Set* | | Av | best | # feas. | $t$ | Av | best | # feas. | $t$ |
| $A_1$ ($ts = 60$) | Monday | 1580 | 1470 | 5 | 378.49 | 1640.80 | 1559 | 5 | 271.16 |
| | Tuesday | 1985.40 | 1863 | 5 | 476.74 | 1911 | 1815 | 5 | 303.05 |
| | Wednesday | 2134.75 | 2079 | 4 | 146.95 | 1745 | 1702 | 5 | 251.39 |
| | Thursday | - | - | 0 | 196.39 | 2200 | 1964 | 5 | 424.02 |
| | Friday | 1504.40 | 1353 | 5 | 348.66 | 1536.80 | 1368 | 5 | 240.70 |
| | Saturday | 873.60 | 849 | 5 | 170.95 | 904.40 | 838 | 5 | 112.13 |
| $A_2$ ($ts = 45$) | Monday | 2049.60 | 1943 | 5 | 489.95 | 2051.20 | 1991 | 5 | 367.94 |
| | Tuesday | 2306.20 | 2141 | 5 | 680.16 | 2270.40 | 2118 | 5 | 437.59 |
| | Wednesday | 1787.80 | 1703 | 5 | 465.38 | 1800.60 | 1644 | 5 | 315.51 |
| | Thursday | 2402 | 2257 | 5 | 786.28 | 2466.80 | 2216 | 5 | 491.38 |
| | Friday | 1731.20 | 1644 | 5 | 471.89 | 1731.80 | 1538 | 5 | 289.28 |
| | Saturday | 1038.40 | 958 | 5 | 200.57 | 971.60 | 937 | 5 | 136.10 |
| $A_3$ ($ts = 30$) | Monday | 2156.60 | 2108 | 5 | 872.13 | 2053.40 | 1938 | 5 | 320.56 |
| | Tuesday | - | - | 0 | 732.19 | - | - | 0 | 411.59 |
| | Wednesday | 2280.60 | 2122 | 5 | 810.04 | 2216.20 | 2110 | 5 | 443.36 |
| | Thursday | - | - | 0 | 1158.78 | - | - | 0 | 589.55 |
| | Friday | 1980.60 | 1816 | 5 | 668.58 | 1999.60 | 1848 | 5 | 330.31 |
| | Saturday | 1227.20 | 1165 | 5 | 234.92 | 1217.60 | 1142 | 5 | 114.78 |
| Av | | 1802.56 | 1698.07 | 4.11 | 516.03 | 1794.83 | 1670.50 | 4.44 | 325.03 |

Table 1.9: H-ALNS results on the instances of Set $A$.

ples is to use a time span of 45 minutes. In fact, on these sets (i.e., $A_2$ and $B_2$) both the H-ALNS algorithms find five feasible solutions (out of the five runs) for each instance, while on the sets with time span of 60 and 30 minutes the average number of feasible solutions is 2.6 and 3.6, respectively. In terms of solution quality, the H-ALNS algorithms attain the best results on the instances of Sets $A_1$, $A_2$ and $B_2$, suggesting that using a time span of 45 or 60 minutes is the best batching policy when a feasible solution is found. On these instances, it turns out that $C_9$ finds always feasible solutions, while $C_1$ is not able to find any solution on Thursday of Set $A_1$, only.

Comparing results on instances $A_2$ and $B_2$, in which a feasible solution is found in all the runs, it turns out that the H-ALNS algorithms attain the best results on instances of $B_2$, both in terms of solution quality and computational times. Finally, regarding the biggest instances of Sets $A_3$ and $B_3$, we observe that the algorithms find feasible solutions in all the runs except for the instances of Thursdays and Tuesdays. As already stated, this fact could mainly due to the big sizes of these instances, with a number of requests ranging from 109 to 120 (see Table 1.3).

| | | $C_1$ | | | | $C_9$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| *Set* | | Av | best | # feas. | $t$ | Av | best | # feas. | $t$ |
| | Monday | - | - | 0 | 326.38 | - | - | 0 | 255 |
| | Tuesday | - | - | 0 | 325.79 | - | - | 0 | 235.45 |
| $B_1$ | Wednesday | - | - | 0 | 313.99 | - | - | 0 | 226.68 |
| ($ts = 60$) | Thursday | - | - | 0 | 341.95 | - | - | 0 | 221.54 |
| | Friday | - | - | 0 | 313.86 | - | - | 0 | 228.79 |
| | Saturday | 804.20 | 792 | 5 | 153.47 | 836.40 | 802 | 5 | 106.35 |
| | Monday | 1744.00 | 1687.00 | 5.00 | 374.22 | 1697.00 | 1496.00 | 5.00 | 254.16 |
| | Tuesday | 2077.80 | 1983.00 | 5.00 | 454.48 | 2096.60 | 2003.00 | 5.00 | 350.16 |
| $B_2$ | Wednesday | 1567.00 | 1461.00 | 5.00 | 400.47 | 1596.60 | 1463.00 | 5.00 | 314.38 |
| ($ts = 45$) | Thursday | 2123.80 | 2005.00 | 5.00 | 480.27 | 2152.80 | 2040.00 | 5.00 | 357.66 |
| | Friday | 1530.20 | 1392.00 | 5.00 | 346.70 | 1604.00 | 1493.00 | 5.00 | 215.51 |
| | Saturday | 901.60 | 835.00 | 5.00 | 184.10 | 951.80 | 898.00 | 5.00 | 115.64 |
| | Monday | 2156.60 | 2108.00 | 5.00 | 871.66 | 2053.40 | 1938.00 | 5.00 | 320.69 |
| | Tuesday | 2369.00 | 2283.00 | 2.00 | 910.77 | 2534.67 | 2441.00 | 3.00 | 628.46 |
| $B_3$ | Wednesday | 2280.60 | 2122.00 | 5.00 | 810.04 | 2216.20 | 2110.00 | 5.00 | 443.36 |
| ($ts = 30$) | Thursday | 2501.00 | 2501.00 | 1.00 | 972.50 | 2546.00 | 2546.00 | 1.00 | 692.30 |
| | Friday | 1980.60 | 1816.00 | 5.00 | 668.58 | 1999.60 | 1848.00 | 5.00 | 330.31 |
| | Saturday | 1227.20 | 1165.00 | 5.00 | 234.92 | 1217.60 | 1142.00 | 5.00 | 114.78 |
| Av | | 1789.51 | 1703.85 | 3.22 | 471.34 | 1807.90 | 1709.23 | 3.28 | 300.62 |

Table 1.10: H-ALNS results on the instances of Set $B$.

Summarizing, the developed H-ALNS algorithms are able to find solutions in which all the samples are delivered on time, using different grouping time spans with limited computational efforts. Such a result improves the current real case related to the Local Healthcare Authority of Bologna, in which the specified time requirements are not fully respected. Furthermore, a deeper analysis of the solutions found by the H-ALNS algorithms provides information that could help to take decisions at a tactical and/or strategic level. At this aim, we considered the best solutions found by the H-ALNS Algorithm $C_9$ on the six instances of Set $A_1$, one of the sets in which $C_9$ always finds feasible solutions (see Table 1.9). Table 1.11 reports the spokes used by these best solutions: An "X" in the table indicates that the corresponding spoke (labeled as $sp_1, sp_2, \ldots, sp_{12}$) is visited by at least one vehicle in the solution. The last column of the table reports on the total number of spokes used by the solutions. Note that, the number of used spokes ranges from 7 to 9, implying that around 30% of the available spokes is not used. Furthermore, it can be noticed that Spokes $sp_3$, $sp_9$ and $sp_{12}$ are not used by any solution. At a managerial level, such a fact could have a relevant

| Spoke | $sp_1$ | $sp_2$ | $sp_3$ | $sp_4$ | $sp_5$ | $sp_6$ | $sp_7$ | $sp_8$ | $sp_9$ | $sp_{10}$ | $sp_{11}$ | $sp_{12}$ | # spokes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Monday | X | X | | X | X | X | X | X | | X | X | | 9 |
| Tuesday | X | X | | X | X | X | X | X | | X | X | | 9 |
| Wednesday | | X | | X | | X | X | X | | X | X | | 7 |
| Thursday | X | X | | X | X | X | X | X | | X | X | | 9 |
| Friday | | X | | X | X | X | X | X | | X | X | | 8 |
| Saturday | | X | | X | X | X | X | X | | X | X | | 8 |

Table 1.11: Number of spokes used by the best solutions of H-ALNS $C_9$ on Set $A_1$.

economic and organizational impact, since it informs healthcare managers that some spokes can be removed without affecting the system performances. A similar analysis has been performed on the number of vehicles used by the solutions. It shows that the solutions often employ all the available vehicles, suggesting that the performances of the system could be sensible to a reduction in the number of vehicles.

## 1.7 Conclusions

In this chapter, a transportation problem arising from a real-world healthcare application has been presented, dealing with the transportation of biological samples from blood draw labs to a main laboratory. The problem can be modeled as a Vehicle Routing Problem with time-windows and lifetime constraints, in which transfers and multiple possible visits at specific nodes (i.e., the spokes) are allowed to get extra lifetimes of the transportation requests. Mixed Integer Linear Programming formulations and hybrid ALNS algorithms have been proposed for the problem. Computational experiments on different sets of instances, based on real-life data provided by the Local Healthcare Authority of Bologna, Italy, have been also presented. A comparison between the solutions obtained by MILP formulations and the H-ALNS algorithms on small instances shows the effectiveness of the proposed metaheuristics and assesses the superiority of the two time-indexed

MILP models with respect to the simple MILP one. On real-life instances, due to the big number of samples to deliver each day, we proposed an experimental study to evaluate different grouping policies. The computational results show that the H-ALNS algorithms are able to find solutions in which all samples are delivered on time, while, in the real case, the lifetime requirements of the samples are not currently satisfied.

# Chapter 2

# Solving Crop Planning and Rotation Problems in a Sustainable Agriculture Perspective

Climate change patterns and the uncertainty of the availability of global value chain commodities, due to socio-economic aspects, call for a viable development of resilient agricultural processes able to ensure sustainability at various levels: economic (including food security), environmental (relating to resource efficiency, soil and water quality and threats to habitats and biodiversity) and social (territorial and community level).

Sustainable agriculture aims to address the above challenges highlighting the benefits of crop planning decisions that include the rotation of crops across growing seasons [38] and crop diversification strategies. As highlighted in [51], crop rotation increases crop revenues, improves soil structure and decreases farming costs owing to reduced need for fertilizers and pesticides.

Many countries and organizations promoted sustainable agriculture by ad-hoc policies followed by specific regulations. In particular, crop rotation was included as a new commitment into the Good Agricultural Environmental Conditions (GAEC) list published in the last Common Agricultural Policy (CAP) proposal of the European Union (EU), and several times it has been included as a tool of sustainability in production schemes used by farmers, food producers and retailers [49]. As a result, attention is increasing for decision support tools that can be used to assess the impact of common policies and private initiatives from one side, and to help farmers to maximize their revenues, while respecting sustainability principles [41; 39].

In this chapter, the problem of planning the allocation of crops to arable lands is addressed, taking into account crop rotation principles and diversification strategies promoted by sustainable agriculture, and optimization models are proposed for solving multi-period planning problems. From one side, the models can be embedded in decision support tools able to help farmers maximizing their revenues in compliance with ecological transition pathways indications. From the other side, they can be used by decision makers to assess the effectiveness of current regulations and to design future rules to promote sustainability principles in arable land use, as well as to evaluate farmers production factors allocation choices.

The optimization models proposed in this chapter allow to decide how to allocate the available farmland among different crops in each growing season to maximize the total expected profit over a finite planning horizon. The allocation decisions are made considering the crop rotation benefits and the sustainable requirements stated by regulations. The proposed models, differently from other works in the literature (e.g., see [50]), allow crop rotation schemes that do not follow the best agronomic practices, by taking into account and quantifying the agronomic costs and constraints

deriving from the adoption of the same crop species/type on the same plot. The models' outputs will provide information that could be useful to assess, via farm level quantitative economics indicators, the viability of crop rotation schemes taking into account the potential economic impact on agriculture value chain actors in different pedo-climatic areas.

Numerical experiments have been conducted with real data coming from structured professional farms specialized in arable crops, located in one of the most intensive agricultural areas of production in Europe, the plain area around the Po river basin, called "Pianura Padana" Valley in northern Italy [43]. In this pedo-climatic context, farmers plan the use of their arable land by distinguishing winter crops from summer crops. The types of crops that can be selected change among farms and across seasons according to structural features (irrigation, machines and work availability), costs and benefits prediction (potential yields, product prices, input availability, premium prices and subsidies) and specific pedo-climatic constraints (temperatures, pluviometry trends, type of soil).

The main contributions of this work are listed below:

- A formal characterization of the crop planning and rotation problem is given, where the rotation schemes are based on sequences of $k$ consecutive crops. The risks/benefits of all possible $k$-crop rotation schemes are assessed, including those not following the traditional agronomic practices.
- A complexity analysis is performed, showing that the problem is strongly NP-hard when $k \geq 3$.
- Polynomial network flow approaches for special cases are proposed.
- A real world application is presented, based on data from Italian farms and current sustainability rules coming from public (i.e., CAP) and private initiatives (promoted by the Barilla Group). So, the models are settled up for taking into account constraints and incomes coming from the adoption of different production schemes.

- Integer Linear Programming (ILP) models are developed for the case under study (Italian farms of the Pianura Padana Valley) where $k$ is set to 3, as a consequence of best practices in Mediterranean pedo-climatic contexts.
- An experimental campaign on real data shows that the models are able to optimally solve the planning problems under different sustainability scenarios for all the real-life instances, in reasonable computational times.

The chapter is organized as follows. In Section 2.1, a literature review on crop planning and solution approaches is presented. In Section 2.2, crop rotation issues and other sustainable requirements are described into detail. A formal definition of the problem and a complexity analysis is reported in Section 2.3. In Section 2.4, a polynomial network flow approach for solving a special class of crop planning problems is proposed. In Section 2.5, the ILP models related to the case under study are presented. Sections 2.6 and 2.7 are devoted to describe the real data and the results of the experimental analysis, respectively. Conclusions follow in Section 2.8.

## 2.1 Literature review

In the literature, crop planning and rotation problems have received considerable attention from the operations management and agricultural economics communities. We refer the reader to [33; 45] for reviews on the topic.

In particular, Mathematical Programming approaches have been proposed by different authors for solving crop planning and rotation problems. Haneveld and Stegeman [50] proposed linear programming models for farm production planning with crop rotation constraints, where crop succession information is only given in the form of a set of inadmissible successions of crops. The

decision variables represent the areas where a certain admissible sequence of crops is cultivated. In Alfandari *et al.* [34], the authors focused on the Madagascan context, where the minimization of cultivated space contributed to the sustainable development of the primary forest in the long term. An ILP formulation and a graph formulation are provided, and a branch and price and cut approach is adopted to solve the problem of minimizing the cultivated land necessary to satisfy demand constraints. Given the different pedo-climatic context, the constraints of the problem are quite different from the ones considered in this paper. Also crop rotation is handled differently and, in general, only some rotation schemes are allowed in their problem. In Santos *et al.* (2015) [58], an ILP model is presented for the problem of finding the minimum land necessary to satisfy crop demand. Crop rotation is only considered by imposing a green manuring crop and a fallow period in the crop schedules. In particular, they use binary variables and assignment constraints to model the crop schedules on single plots of land and adopt integer variables to minimize the total number of base-area plots assigned to the different schedules. A Branch and Price and Cut approach is presented, able to solve instances up to 20 crops and a 2 years planning horizon.

Santos *et al.* (2010) [56] also presented a binary linear programming model for the Crop Rotation Scheduling Problem (CRSP) aiming to maximize the plots occupation considering demand constraints. The authors proposed a column generation procedure to solve the model. A similar procedure was also employed in [57] to solve a variant of CRSP without demand constraints. In [57], the authors were the first to introduce adjacency constraints for the CRSP, preventing crops of the same botanical family to be cultivated at the same time in neighboring plots. In [53], a crop planning problem in the Brazilian area is addressed, with the objective of maximizing plots' occupation and profit while dealing with adjacency constraints. The authors propose improvements in

the mathematical model presented by [57] and a detailed set of instances based on real-world data. In order to find bounds and solutions for the improved model, five different relaxation approaches are proposed. The results show improvements from the model proposed by [57].

In Fikry *et al.* [46], ILP models are presented to assign crops to plots to periods with the objective of maximizing the total profit. The crop rotation is performed by simply forbidding that two crops of the same family are consecutively planted. However, in the models, the profits earned by each crop are considered as *static*, i.e., do not depend by the succession of the previous crops grown on the same piece of land as in our model.

Boyabatli [38] *et al.* proposed a stochastic dynamic program model for crop planning in sustainable agriculture, taking into account crop rotation issues. However, only succession between two crops (i.e., corn and soybeans) are considered in the study. Detlefsen and Jensen [42] consider the problem of finding an optimal crop rotation for a given selection of crops on a given piece of land. The problem is modeled and solved as a minimum cost network flow problem for the case in which sequences of at most three crops are considered. Bachinger and Zander [35] presents a decision support tool called ROTOR, a static rule-based model in which a set of annual crop production activities are assembled semi-automatically. However, ROTOR does not include the crop rotation benefits in the decision process, but it allow to only perform a *what if* static analysis.

Works from the literature also focus on different planning and coordination problems in agriculture. Filippi *et al.* [47] addressed the problem of crop-mix selection with the objective of maximizing the farmer's profit. In particular, they focus on the working phases (and their costs) required by each crop, rather than on crop rotation issues, and develop two integer programming models: the first one to solve the problem of maximizing the farmer's profit, setting costs and prices

values as the historical average; the second one where consider the risk aversion of the farmer using the maximization of the Conditional Value-at-Risk as objective function and looking for the crop-mix that allows to maximize the average expected profit under a predefined quantile of worst realizations. The models are applied to the real case of a single farm located in Italy.

Volte *et al.* [60] considered the Differential harvesting problem, consisting in optimizing the harvest of different grape qualities in a vineyard so that a minimum quantity of good quality grapes is harvested and directed to a specific bin, while minimizing the total harvesting time. The authors exploit the analogies with the capacitated vehicle routing problem to develop efficient exact methods using column generation and VRPSolver$^{TM}$ based models.

In Yan *et al.* [62], the authors propose a method to coordinate a fresh agricultural product supply chain with the consideration of strategic consumer behavior. In particular, they developed a decision-making model established on the basis of a two-period newsvendor model under the centralized and decentralized chain.

Machine learning techniques have also been used in the literature to deal with problems in the agricultural supply chain. The reader may refer to [59] for a systematic review of such approaches in the agricultural context.

## 2.2 Crop planning issues and sustainability requirements in the European agricultural context

In this section, the concept of crop rotation and the main issues connected to it are introduced. Then, different variants of the crop planning problem are described, accounting for different sus-

tainability scenarios. More precisely, real constraints derived from agricultural policies and agri-food value chain initiatives have been analyzed. The sustainability scenarios are defined on the basis of the current CAP regulations and the rules of private initiatives promoted by food producers. As it will be shown in Section 2.7, optimal solutions of the different scenarios can be used to evaluate the impact of different sustainability policies on the farmers' revenues.

Crop planning is the problem of deciding the crops to cultivate on a farmland for a given planning horizon. When dealing with this problem in sustainable agriculture, decision makers, e.g., farmers, have to take into account two main aspects: $(i)$ the crop rotation principles, and $(ii)$ the constraints deriving from legislative frameworks and policies, especially promoted by international agencies or governments (e.g., CAP for European Union countries) to mitigate natural resources overexploitation.

In agriculture, crop rotation refers to the succession of different crops on the same piece of land over consecutive seeding periods. In fact, monoculture schemes, in which the same crop is assigned to the entire farmland over the whole planning horizon, lead to a series of issues (such as loss of yield and soil fertility, increase of weed and pest diseases) that can be mitigated by applying suitable rotation of crops. The problems caused by intensive agriculture and monoculture practices have been the subject of numerous research over the years, both technical-agronomic and socio-economic [40; 55]. To overcome these problems, movements of scholars, technicians and civil society proposed more sustainable arable land use patterns, inspired to agroecology principles [61]. The principles of the International Federation of Organic Agriculture Movements (IFOAM) were used worldwide to define specific regulatory framework for organic agriculture. In Europe, since the publication of the first Regulation on organic farming in 1992, the importance of crop rotation

has been translated into rotation schemes that can be adopted by farmers. Indeed, it was highlighted the importance of the inclusion of leguminous plants or crops whose management had improving effects on the structure and, more generally, on the natural fertility of the soil [36; 63]. For instance, in Italy, the viable crop rotation schemes for organic agriculture are defined by ministerial decree [44]. The decree states that the same crop species can come back on the same land after two cycles of different crop species, that must include a nitrogen-fixing crop.

In the Mediterranean pedo-climatic context, which is the case considered in this study, the best crop rotation practice consists in the consecutive succession of three different types of crop, namely, a *renewal* crop, an *impoverishing* crop and an *improver* crop, in this order [36]. The characteristics of renewal, impoverishing and improver crops are described in the following:

- Renewal: these crops (e.g., corn, sugar beet, potato, tomato, sunflower, etc.) require particular care consisting in excellent soil preparation and balanced organic fertilizations, which has a positive effect on the structure of the soil. However, in some specific contexts, also a fallow period could be considered as renewal soil quality practice.

- Impoverishing: these crops exploit the nutritional elements present in the soil and deplete it. Crops in this class are wheat, oats, barley, rye, rice, corn, sorghum and generally all grain cereals.

- Improver: these crops increase the fertility of the soil, enriching it with nutrients. Improver crops mainly are legumes, such as alfalfa or clover, which are able to fix atmospheric nitrogen.

As explained above, proper crop rotation schemes may bring many advantages to the farm, both

of an agronomic and economic-managerial perspective [37]. On the other hand, when crop rotation does not follow the best agronomic practice, in order to keep their yield stable over time, farmers will have to compensate for the loss of soil fertility and the increase in weeds and pests' risks by employing technical inputs, such as herbicide fertilizers and pesticides. The use of technical inputs leads to an increase in production costs, directly related on the specific crop succession performed in the rotation scheme.

Summarizing, all crop successions are viable in general, but additional production costs may arise when crop patterns do not follow the best agronomic practices [36]. As an example, Figure 2.1 reports 3-crop rotations following and not following best practices. As a consequence of what we have seen, in the Mediterranean pedo-climatic context it is generally assumed that the production cost and the profit derived by cultivating a crop on a plot of land is linked to the *two* previous crops grown on the same plot.

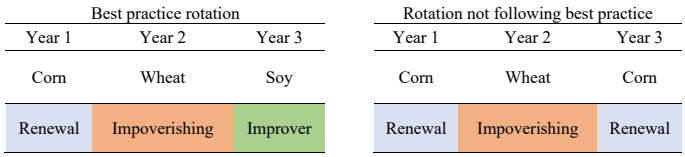



Figure 2.1: 3-crop rotations following and not following best practices.

Another issue strongly related to the crop rotation that must be taken into account in crop planning is the *maximum crop replanting*, namely the number of times a given crop family can repeat itself on a plot in a certain period. In fact, although a cost increase occurs when crop rotation does not follow the best agronomic practice, crops cannot indefinitely be replanted on the same land, if their succession is not interrupted by other crops, also when this solution is technically viable.

In this work, we address the problem of assigning crops to the arable land of a farm over a given

time horizon, taking into account the crop rotation issues described above, with the objective of maximizing the total profit. In the problem, we also consider the impact of regulations arising from public sustainable policies (e.g., CAP regulation in EU) and private supply chain initiatives, that already affect European farmers' choices on arable land allocation through economic incentives. At this aim, three different production scenarios are introduced, based on different greening and sustainability constraints and different incentive levels, leading to different farmers entrepreneurship approaches. The three scenarios are denoted as "Pure farmer", "CAP farmer" and "CAP+SVC farmer" and are described in what follows.

### 2.2.1 Pure farmer

The farmer is not involved in any production scheme and does not follow any sustainable regulation. Therefore, her/his entrepreneurial choices are dictated by the objective of maximizing income and maintaining land capital. In this case, crop planning decisions are only taken on the basis of agronomic knowledge, technical features, and machinery.

### 2.2.2 CAP farmer

The farmer is involved in the greening production schemes of the Common Agricultural Policy (CAP), promoted by the European Union following the EU regulation 1307/2013. In this scenario, farmers receive payments based on a set of standard requirements defined at EU level for obtaining different kinds of incomes which, summed up, generate the CAP economic incentive. The incentive is mainly based on environmental and climate issues and aims to promote practices which are good for the environment (soil and biodiversity in particular) and for climate. More precisely, the CAP

establishes the following rules:

- *Crop diversification* - Farms with more than 10 ha of arable land have to grow at least two crops each year, while at least three crops are required on farms with more than 30 ha. Furthermore, a single crop can not cover more than 75% of the arable land, and, for farms with more than 30 ha, the land assigned to two crops must not exceed 95% of the arable land.

- *Ecological Focus Area* - Farmers with arable land exceeding 15 ha must ensure that at least 5% of their land is an *Ecological Focus Area* (EFA). Ecological focus areas may include different kind of features linked to landscapes, grasslands or improving biodiversity crops. For the farms in the case study, located in an area characterized by the production specialization of high-income arable crops, we have chosen to introduce EFAs devoted to nitrogen fixing crops.

### 2.2.3 CAP+SVC farmer

In this scenario, an income related to a Sustainable Value Chain initiative (SVC) is added to the CAP incentive. More precisely, in the case study, CAP+SVC farmers are also involved in an initiative called *Carta del Mulino* (CdM), introduced by the international Group "Barilla" to promote crop diversification and biodiversity and to support the efforts of farmers towards sustainability through economic incentives [52]. In fact, the CdM incentive is an additional premium price paid for wheat, in the following denoted as CdM crop, cultivated in compliance with rules involving strong agroecological principles.

The CdM constraints are of three types:

- $(i)$ Greening constraints: when a CdM crop is sown on a hectare, then at least one legume or oilseed must be cultivated in the next four crops on the same hectare.

- $(ii)$ Diversification constraints: if a CdM crop is assigned to a hectare, then at least two different crops must be cultivated in the next four crops on the same hectare.

- $(iii)$ Repetition constraints: The CdM regulation define a set of crops, that we denoted as $CR$, subject to the repetition constraints. They state that, if a CdM crop $v$ is assigned to a hectare, then at most one repetition of crops in $CR$ can be performed in the crop sequence containing $v$ and the next four succeeding crops (namely, three crops in $CR$ can not be consecutively assigned in the sequence of five crops that starts from $v$).

Finally, to get the CdM incentives, at least 3% of the agricultural area devoted to wheat must be dedicated to flower strips. This constraint will be implicitly included in the real data provided by the farmers and, thus, not explicitly imposed by the models.

## 2.3 Notation, problem definition and complexity

In this section, we formally define the problem of planning crop production on a farmland taking into account the crop rotation requirements reported in Section 2.2 and the constraints related to environmental sustainability arising from the CAP regulation and the CdM initiative presented in Sections 2.2.2 and 2.2.3, respectively.

In what follows we formally define the problem by generally considering $k$-crop sequences for any $k \geq 1$. Namely, due to the rotation issues introduced in Section 2.2, we assume that the profit earned by cultivating a crop is *affected* by the $k-1$ crops preceding it on the same piece of land.

As highlighted in Section 2.2, we focus on the case of $k = 3$ characterizing the Mediterranean pedo-climatic context, from which the real case arise.

Let a farmland be defined as a set of hectares $H = \{1, \ldots, m\}$ to be cultivated. Let $C = \{c_1, c_2, \ldots, c_n\}$ be the set of crops available to be sown on farmland's hectares during a planning horizon $T = \{1, \ldots, p\}$ composed of $p$ seeding periods. As we are mainly considering the European context, each period in $T$ correspond to a "fall" (starting in October, devoted to winter crops) or to a "spring" (starting in April, devoted to summer crops) semester. Odd (even) periods in $T$ denote the first (the second) semester of a year. We denote by $T_o$ the set of odd periods and by $T_e$ the set of even periods in $T$, with $T = T_e \cup T_o$. The crops in $C$ are classified according to their seeding periods and time requirements as *annual*, *first semester* and *second semester* crops. Let $C_A$, $C_F$ and $C_S$ be the disjoint sets of annual, first semester and second semester crops, respectively, with $C = C_A \cup C_F \cup C_S$. These three sets are formally defined in the following. The crops in $C_A$ can only be sown in first semester periods $t \in T_o$, and need to stay on the land also in the related second semester $t + 1 \in T_e$. Moreover, each annual crop $c \in C_A$ has a *duration* $d_c$ ($2dc$) indicating the number of years (periods) it need to stay on the land. The crops in $C_F$ can only be sown at the beginning of a first semester periods $t \in T_o$ and the harvest is gathered at the end of the semester. Finally, the crops in $C_S$ can only be sown in a second semester period $t \in T_e$ and only on hectares where a crop in $C_F$ was cultivated in the first semester $t - 1 \in T_o$. Given a crop $c \in C$ and a period $t \in T$, we say that $c$ and $t$ are *compatible* if $t$ is a possible seeding period for $c$. We assume that a crop in $C$ must be always assigned to each odd period $t \in T_o$ on each hectare (fallow periods can be simply modeled by introducing a *dummy fallow crop* in $C$, belonging to a type depending on specific cases). On the other hand, we assume that an even period $t$ must not be necessarily

assigned to a second semester crop, when $t-1 \in T_o$ is assigned to a first semester crop. (Note that, when $t-1 \in T_o$ is assigned to an annual crop, $t$ is not available for a second semester crop.)

As reported in Section 2.2, crops are also classified according to their types, i.e., *impoverishing*, *renewal* and *improver*. We denote the disjoint sets of impoverishing, renewal and improver crops as $C_D$, $C_R$ and $C_M$, respectively, with $C = C_D \cup C_R \cup C_M$. The type is related to the effect produced by a crop on the soil, and affects the costs, profits and yields of crop successions. In fact, as stated in Section 2.2, the profit obtained by cultivating a crop $c \in C$ on one hectare depends on the *crop rotation scheme*, i.e., on the types of the crops immediately preceding $c$, on that hectare. In this work, we consider a $k$-crop rotation scheme, in which the profit obtained by assigning a crop $c$ to a hectare $h$ depends on $c$ and on the types of the $k-1$ crops preceding $c$ on $h$, if $k-1$ preceding crops exist. (If the crops preceding $c$ are smaller than $k-1$, then the profit depends on the types of all the crops cultivated before $c$). Furthermore, according to the *maximum crop replanting* issue (see Section 2.2), limits exist on the number of consecutive repetitions of some crop on the same piece of land. More precisely, crops are divided into *homogeneous replanting groups*, and crops in the same group can not be consecutively repeated on the same hectare a number of times bigger than the *maximum replanting value* of the group.

In what follows, we formally define a crop assignment for our problem.

**Definition 2.3.1** *A* crop assignment *consists in:*

- *assigning, on each hectare, an annual or a first semester crop to each odd period in $T_o$;*
- *deciding whether to assign or not, to each hectare, a second semester crop to each empty even period $t$, i.e., the periods $t \in T_e$ such that $t-1$ is assigned to a first semester crop.*

Note that a crop assignment defines a *crop sequence* $\sigma = \langle c_1, c_2, \ldots, c_q \rangle$ on each hectare, whose length (i.e., the number of crops in the sequence) depends on how many even periods are used for second semester crops. In fact, it is easy to see that the *longest crop sequence* assigns one crop to each period in $T$, and, hence, alternates first and second semester crops. On the other hand, the *shortest crop sequence* does not assign second semester crops (i.e., contains $p/2$ annual or first semester crops, where $p$ is the last period of the time horizon $T$).

Let $MAX_c$ and $MIN_c$ respectively be the maximum and the minimum number of hectares that can be assigned to crop $c$ in each period in $T$, for all $c \in C$. Recalling the requirements on the maximum replanting, a *feasible crop assignment* can be formally defined as follows.

**Definition 2.3.2** *A crop assignment is* feasible *if:*

- *on each hectare, the related crop sequence $\sigma$ does not contain a subsequence in which crops of the same homogeneous replanting group are consecutively repeated a number of times bigger than the maximum replanting value;*
- *the number of hectares assigned to a crop $c$ in each compatible period in $T$ belongs to the interval $[MIN_c, MAX_c]$, for all $c \in C$;*
- *each annual crop $c \in C_A$ sown on a hectare in $t \in T_o$ stays on that hectare for at least $2d_c$ periods from $t$ (its required duration).*

We now formally define the profit of a crop sequence $\sigma = \langle c_1, c_2, \ldots, c_q \rangle$ assigned to a given hectare $h$. For each crop $c \in \sigma$, let $\sigma_k(c)$ be the subsequence of $\sigma$ containing the crop $c$ and the $k-1$ crops immediately preceding $c$ in $\sigma$, if any. If less than $k-1$ crops precede $c$, then $\sigma_k(c)$ will contain all the preceding crops. We denote by $p_h(\sigma_k(c))$ the profit earned by growing crop $c$ on

hectare $h$ when the at most $k-1$ crops immediately preceding $c$ on $h$ are those given in $\sigma_k(c)$. The total profit $\pi_h(\sigma)$ earned by cultivating the crop sequence $\sigma$ on hectare $h$ is then given by

$$\pi_h(\sigma) = \sum_{c \in \sigma} p_h(\sigma_k(c)) \tag{2.1}$$

As an example, assuming that $k = 3$, the total profit (2.1) produced by the sequence $\sigma = \langle c_1, c_2, c_3, c_4 \rangle$ on a hectare $h$ is given by

$$\pi_h(\sigma) = p_h(\langle c_1 \rangle) + p_h(\langle c_1, c_2 \rangle) + p_h(\langle c_1, c_2, c_3 \rangle) + p_h(\langle c_2, c_3, c_4 \rangle).$$

The $k$-sequence Crop Rotation Problem (CRP-$k$) can be now formally defined.

> $k$ CROP ROTATION PROBLEM (CRP-$k$):
>
> *Given* a set $C$ of crops, a set $H$ of hectares, a set $T$ of periods, the profits per hectare $p_h(\sigma_k(c))$ for all $c \in C$ and $h \in H$, a minimum $MIN_c$ and a maximum $MAX_c$ number of hectares to assign to each crop $c \in C$ in each period, the maximum replanting values of the homogeneous groups and the duration of annual crops;
>
> *Find* a feasible crop assignment maximizing the total profit $\sum_{h \in H} \pi_h(\sigma)$.

Note that, by definition, CRP-$k$ corresponds to the crop planning problem in the "Pure Farmer" scenario introduced in Section 2.2.1. i.e., sustainability requirements are not considered.

As an example, let us consider a CRP-$k$ instance with $k = 3$, three crops, $C = \{c_1, c_2, c_3\}$, ten periods, $T = \{1, 2, \ldots, 10\}$, corresponding to five years, and one hectare $H = \{1\}$. Let us assume that $c_1$ and $c_2$ are annual crops, $c_3$ a first semester crop, with $MIN_c = 0$ and $MAX_c = 1$ for all $c \in C$, and suppose that each crop belongs to a different homogeneous replanting group with maximum replanting equal to two. Hence, the seeding periods are all the odd periods in $T$, i.e., $T_o = \{1, 3, \ldots, 9\}$.

Let us consider the solution in which the crops $c_1, c_1, c_2, c_3, c_2$ are respectively assigned to the odd periods in $T_o$ on the available hectare, producing the crop sequence $\sigma = \langle c_1, c_1, c_2, c_3, c_2 \rangle$. Note that the crop assignment is feasible, since it satisfies the maximum replanting requirement (no crop is consecutively repeated more than twice), and the constraints on the minimum and maximum number of hectares are trivially satisfied. By definition, the total profit of the solution is given by $\pi_1(\sigma) = p_1(\langle c_1 \rangle) + p_1(\langle c_1, c_1 \rangle) + p_1(\langle c_1, c_1, c_2 \rangle) + p_1(\langle c_1, c_2, c_3 \rangle) + p_1(\langle c_2, c_3, c_2 \rangle)$.

The following theorem shows that CRP-$k$ with $k = 3$ is strongly NP-hard.

**Theorem 2.3.1** *CRP-3 is strongly NP-hard.*

*Proof.* The proof is by reduction from the strongly NP-hard axial 3-index assignment problem (3IAP) [48]. 3IAP can be defined as follows.

> *Instance:* We are given 3 sets $P$, $Q$ and $R$ of equal size $n$, a profit $a_{i,j,l}$ associated to each triple $(i, j, l) \in P \times Q \times R$.
>
> *Problem:* Find $n$ triples such that: $i$) each element of $P$, $Q$ and $R$ belongs to exactly one triple, $ii$) the total profit of the selected triples is maximum.

Given an instance of 3IAP, we build an instance of CRP-3 with a time horizon of 3 years (six periods), a set $C$ with $n$ first semester crops, each crop belonging to a different replanting group, and a farmland with $n$ hectares. Hence, crops can only be assigned to the odd periods 1, 3 and 5. Let the maximum replanting value of each crop be equal to 3, i.e., each crop $c \in C$ can be replanted in all odd periods on any hectare. Then, we set the maximum and minimum number of hectares to assign to each crop in each year equal to 1, i.e., $MAX_c = 1$ and $MIN_c = 1$. Hence, each crop $c \in C$ must be assigned to exactly one hectare in each odd period. Note that, a crop assignment produces

a crop sequence of length three on each hectare, say $\langle i, j, l \rangle$, with $i, j, l \in C$, that we associate to the triple $(i, j, l) \in P \times Q \times R$ of the 3IAP instance. Furthermore, since a crop $c$ can be assigned to exactly one hectare in each odd period ($MAX_c = MIN_c = 1$), a feasible crop assignment of the CRP-3 instance corresponds to a set of triples such that each element of $P$, $Q$ and $R$ belongs to exactly one triple of the set.

The profits $p_h(\langle i \rangle)$, $p_h(\langle i, j \rangle)$ and $p_h(\langle i, j, l \rangle)$ obtained by assigning the crop sequence $\langle i, j, l \rangle$ on hectare $h$ are defined in such way that the total profit of the sequence is $\pi_h(\langle i, j, l \rangle) = p_h(\langle i \rangle) + p_h(\langle i, j \rangle) + p_h(\langle i, j, l \rangle) = 2a_{i,j,l}$. Note that, this is always possible by setting:

$p_h(\langle i \rangle) = \min_{u \in Q, v \in R}\{a_{i,u,v}\}$;

$p_h(\langle i, j \rangle) = \min_{v \in R}\{a_{i,j,v}\}$;

$p_h(\langle i, j, l \rangle) = 2a_{i,j,l} - p_h(\langle i \rangle) - p_h(\langle i, j \rangle)$.

Hence, by definition of the profits, a feasible solution of 3IAP of total profit $M$ corresponds to a feasible solution of CRP-3 with total profit $2M$, and vice versa. Such correspondence holds even for an optimal solution of 3IAP, and the thesis follows. □

### 2.3.1 Variants of the problem accounting for sustainable scenarios

According to the scenarios introduced in Sections 2.2.1–2.2.3, CRP-$k$ basically corresponds to the "Pure Farmer" scenario, since no greening and/or sustainable constraint is explicitly considered. However, the "CAP Farmer" scenario can be formally stated by properly redefining a feasible crop assignment (given in Definition 2.3.2), in such a way that the CAP constraints specified in Section 2.2.2 are taken into account. Similarly, the "CAP+SVC farmer" scenario can be formally defined by changing Definition 2.3.2, in such a way that the CdM rules defined by the *Carta del Mulino*

initiative (see Section 2.2.3) are satisfied. In order to introduced the ILP models, Formal notation for this two latter scenarios is introduced in Section 2.5.

## 2.4 A network flow model for solving a variant of CRP-$k$

In this section, we introduce a simplified version of CRP-$k$, denoted as SCRP-$k$, and show that it can be polynomially solved as a minimum cost network flow problem when $k = 3$ and, in general, for any $k$, with $k$ fixed. SCRP-$k$ is defined as CRP-$k$ in which:

- restrictions on the number of hectares to assign to each crop in each period do not exist (i.e., $MIN_c = 0$ and $MAX_c = \infty$ for all $c \in C$);
- the maximum replanting for each group and the duration $d_c$ of each annual crop $c$ is at most $k-1$.

In Section 2.4.1, we show that the network flow approach described in the following for SCRP-3 can be easily extended to polynomially solve SCRP-$k$, when $k > 3$ is fixed.

SCRP-3 can be modeled and solved as a minimum cost network flow problem, as explained in the following. Recall that $n$ is the number of crops in $C$ and $m$ is the number of hectares in $H$. Let $G = (V,E)$ be a graph where $V$ and $E$ are the node and the arc sets, respectively. On $G$, we define a minimum cost network flow problem with one source and one demand node, denoted as $s$ and $t$, respectively. An amount of flow equal to $m$ has to be sent from $s$ to $t$. The nodes in $V$ are partitioned into layers, say $L_0, L_1, \ldots, L_l$, where $L_0$ and $L_l$ only contain the source and the demand node, respectively. The arcs in $E$ only connect the nodes of two adjacent layers $L_i$, $L_{i+1}$, for $i = 0, \ldots, l-1$, or a node of a layer with the demand node. A minimum and a maximum capacity

equal to 0 and $m$, respectively, is assigned to each arc. Each node of an intermediate layer $L_i$, for $i = 1, \ldots, l-1$, is denoted as $(a,b)^{p,q}$ and models the assignment of the two crops $a$ and $b$ in $C$ on the two compatible periods $p$ and $q$, respectively, with $p < q$, when no other crop is assigned to periods $p+1, p+2, \ldots, q-1$.

An arc between nodes of layers $L_i$ and $L_{i+1}$, with $1 \leq i < l-1$, models a sequence of four crops, possibly not all distinct, over four periods, as detailed in the following. Let us denote as $(a,b)^{p,q}$ and $(c,d)^{u,v}$ two nodes of layers $L_i$ and $L_{i+1}$, respectively, with $1 \leq i < l-1$. The flow on the arc from $(a,b)^{t,q}$ to $(c,d)^{u,v}$ represents the number of hectares on which the crop sequence $\langle a,b,c,d \rangle$ is assigned to the compatible set of periods $\{t,q,u,v\}$. Observe that, the arc from $(a,b)^{t,q}$ to $(c,d)^{u,v}$ allows to keep track of the two crops preceding $c$ and $d$, i.e., $\langle a,b \rangle$ and $\langle b,c \rangle$, respectively. Given a node $(c,d)^{u,v}$ of an intermediate layer, it may happen that the crop $d$ does not exist (e.g., when $u = p$ is the last period in $T$, or when $u = p-1$ and $c$ is an annual crop). In such cases, the node is written as $(c,-)^{u,-}$. The profit assigned to the arc from $(a,b)^{t,q}$ to $(c,d)^{u,v}$ is equal to the profit obtained by assigning to one hectare, say $h$, the crops $c$ and $d$ to the periods $u$ and $v$ respectively, when the two preceding crops of $c$ and $d$ are those given in the $k = 3$ crop sequences $\langle a,b,c \rangle$ and $\langle b,c,d \rangle$ respectively, i.e.,

$$p_h(\langle a,b,c \rangle) + p_h(\langle b,c,d \rangle).$$

The profit assigned to an arc from the source node $s$ to a node $(a,b)^{t,q}$ of the first intermediate Layer $L_1$ is equal to the total profit obtained by assigning to one hectare, say $h$, the sequence $\langle a,b \rangle$ to periods $\{t,q\}$, i.e.,

$$p_h(\langle a \rangle) + p_h(\langle a,b \rangle).$$

In what follows, we derive the length of shortest and longest paths on $G$. Note that, given a node $(a,b)^{t,q}$ of $G$, we have either $q = t+1$ or $q = t+2$. In fact, if $a$ and $b$ are a first and a second semester crop, respectively, then $t$ is an odd period and $q = t+1$. While if nor $a$ nor $b$ are second semester crops, then $t$ and $q$ are odd periods and $q = t+2$. By the above observation, and recalling that the length of a crop sequence in a crop assignment (from period 1 to period $p$) depends on how may even periods are assigned to second semester crops, we can easily determine the length of a shortest and of a longest path on $G$, in terms of number of nodes. In fact, a path on $G$ can not be shorter than a path only containing nodes $(a,b)^{t,q}$, in which $a$ and $b$ are annual or first semester crops (and consequently in which $t$ and $q$ are consecutive odd periods). Hence, such a path is a *shortest path* on $G$. On the other hand, a path on $G$ can not be longer than a path only containing nodes $(a,b)^{t,q}$ representing a succession of first and second semester crops in which $q = t+1$. Hence, such a path is a *longest path* on $G$.

The two following lemmas provide the number of layers and the number of nodes of each layer in $G$.

**Lemma 2.4.1** *The number of node layers in graph $G$ is $\frac{p}{2}+2$, where $p$ is the last period of the time horizon $T$.*

*Proof.* Recall that the time periods in the time horizon $T = \{1,\ldots,p\}$ correspond to first and second semesters of a given number of consecutive years, where period 1 is the first semester of the first year and the last period $p$ is the second semester of the last year. Since each node of a path in $G$ belongs to a different layer, the number of layers is equal to the length of a longest path on $G$. As already observed, a path visiting nodes $(a,b)^{t,q}$, where $a$ and $b$ are first and second semester crops, respectively, assigned to periods $t \in T_o$ and $q = t+1$, is a longest path on $G$. Then, it is easy to see

that a longest path from the source node to the demand node must contain $p/2+2$ nodes, that by definition of $G$ belong to distinct layers. □

The following lemma provides the maximum number of nodes contained in each intermediate layer of $G$. (Recall that the first and the last layer only contain one node.)

**Lemma 2.4.2** *The maximum number of nodes contained in each Layer $L_i$ of $G$ is $4in^2$ where $n$ is the number of crops in $C$, for $i = 1,\ldots,\frac{p}{2}+1$.*

*Proof.* By definition, a node $(a,b)^{t,q}$ in each Layer $L_i$, for $i = 1,\ldots,l-1$, is related to a crop pair $a,b \in C$ and to a pair of periods $t,q \in T$, with $q \in \{t+1,t+2\}$. We first compute the number of possible period pairs $t,q$ associated to nodes in each layer $L_i$. At this aim, given a node $(a,b)^{t,q}$ of layer $L_i$, we determine the minimum and the maximum values taken by $t$ and $q$.

Recall that any arc connecting nodes of two intermediate layers $L_i$ and $L_{i+1}$ connects a node $(a,b)^{t,q}$ with a node $(c,d)^{u,v}$. As already observed, there exists on $G$ a shortest path $P_s$ only containing nodes related to annual and first semester crops. Let $(a,b)^{t,q}$ and $(c,d)^{u,v}$ be the two (consecutive) nodes on path $P_s$ belonging to layers $L_i$ and $L_{i+1}$, respectively. We have that $a$, $b$, $c$ and $d$ are first semester or annual crops, and $t,q,u,v$, are consecutive odd periods. On such a path, an arc from $(a,b)^{t,q}$ to $(c,d)^{u,v}$ "moves" from period $t \in T_o$ to $v = t+6 \in T_o$. Note that, no other succession of four consecutive crops produces a move bigger than this, in terms of number of periods. As a consequence, the node of layer $L_i$ belonging to path $P_s$, i.e., $(a,b)^{t,q}$, has the biggest period values $t$ and $q$. In fact, we have $t = 4i-3$ and $q = 4i-1$.

Similarly, as already observed, a longest path exists on $G$, say $P_l$, only containing nodes related to first and second semester crops. Hence, given two consecutive nodes $(a,b)^{t,q}$ and $(c,d)^{u,v}$ on $P_l$,

we have that $t, q, u, v$ are consecutive periods (with $q = t+1$, $u = t+2$, and $v = t+3$), and the arc from $(a,b)^{t,q}$ to $(c,d)^{u,v}$ "moves" from period $t$ to $v = t+3$. Note that, no other succession of four consecutive crops produces a move smaller than this, in terms of number of periods. Hence, the minimum values taken by the periods associated to nodes of layer $L_i$ are the periods $t$ and $q$ of the node $(a,b)^{t,q}$ on $P_l$, for which we have $t = 2i-1$ and $q = 2i$, respectively.

Let $(i,j)^{t_1,t_{max}}$ and $(f,g)^{t_2,t_{min}}$ be the nodes of the intermediate layer $L_i$ belonging to the shortest path $P_s$ and to the longest path $P_l$ of $G$, respectively. By the above discussion, we have that $t_{max} = 4i-1$ and $t_{min} = 2i$ (hence, $t_{max} - t_{min} = 2i-1$). In fact, $t_{max} = 4i-1 \in T_o$ is the seeding period of the $2i$-th crop of the shortest crop sequence (containing annual or first semester crops only), while $t_{min} \in T_e$ is the seeding period of the $2i$-th crop of the longest crop sequence (alternating first and second semester crops). As a consequence, the last period $q$ of each node $(a,b)^{t,q}$ in Layer $L_i$ can take at most $2i$ values, i.e., $2i \leq q \leq 4i-1$. Since, in $(a,b)^{t,q}$, the first period $t \in \{q-2, q-1\}$, the number of possible period pairs associated to nodes of $L_i$ is $2 \times 2i$. Recalling that, in the intermediate layer $L_i$, at most two nodes exist for each crop pair $a,b$ and period pair $t,q$, i.e., $(a,b)^{t,q}$ and $(b,a)^{t,q}$, the nodes in $L_i$ are at most $4in^2$. □

Lemma 2.4.3 provides the total number of nodes in $G$.

**Lemma 2.4.3** *The number of nodes in $G$ is $O(n^2p^2)$ for $f = 1, \ldots, \tau$, where $n$ is the number of crops in $C$ and $p$ is the number of time periods in $T$.*

*Proof.* The thesis easily follows by Lemmas 2.4.1 and 2.4.2 and since $\sum_{i=1}^{p/2} 4in^2 = 4n^2(p/4)(p/2+1)$. □

A consequence of Lemma 2.4.3 is that, when $MIN_c = 0$ and $MAX_c = \infty$ for all $c \in C$, CRP-3 can be

optimally solved in polynomial time by solving a minimum cost network flow problem on graph $G$.

In Figure 2.2, the graph $G$ is reported for a SCRP-3 instance with a three year planning horizon $T = \{1,2,3,4,5,6\}$ and 3 crops $C = \{a,b,c\}$, where $a$ is an annual crop and $b$ and $c$ are first and second semester crops, respectively. Hence, $a$ and $b$ can be assigned to periods in $T_o = \{1,3,5\}$, and $c$ to periods in $T_e = \{2,4,6\}$. Note that, a shortest path on $G$ alternating annual or first semester crops is $s \to (a,b)^{1,3} \to (a,-)^{5,-} \to t$. On the other hand, a longest path on $G$, alternating first and second semester crops, is $s \to (b,c)^{1,2} \to (b,c)^{3,4} \to (b,c)^{5,6} \to t$.

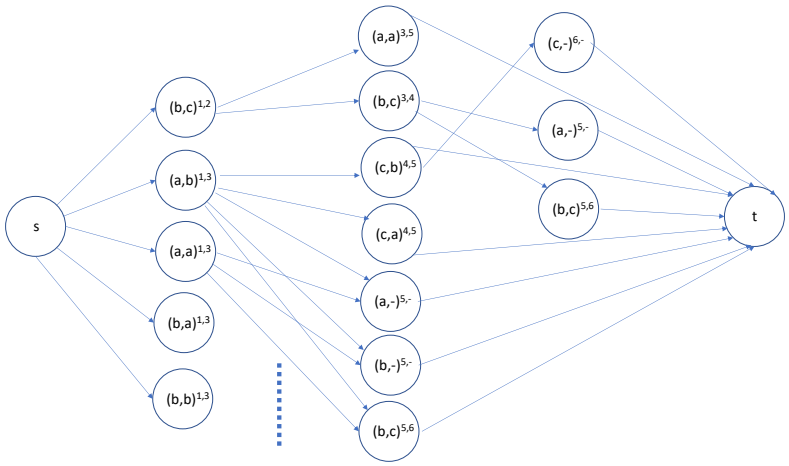



Figure 2.2: The network flow of the SCRP-3 instance.

### 2.4.1 Extending the network flow approach to SCRP-$k$

The graph $G$ described in the above section can be suitably modified to model SCRP-$k$ when $k > 3$. The graph $G$ used for modeling SCRP-$k$ is a multi-layer graph too, with the difference that a node in each intermediate layer models the assignment of a subsequence of $k-1$ crops to $k-1$

compatible periods. Such a node can be denoted as $(c_1, c_2, \ldots, c_{k-1})^{t_1,t_2,\ldots,t_{k-1}}$.

By exploiting the concepts of shortest and longest paths on the graph $G$ introduced for SCRP-3, it can be proved that the number of layers and nodes in each layer is polynomial with respect to the number of crops and periods. This implies that SCRP-$k$ can be polynomially solved as a minimum cost network flow problem when $k$ is fixed, i.e., it is not part of the input.

## 2.5 An Integer Linear Programming formulation for CRP-3

In this section, an ILP formulation for CRP-$k$ with $k = 3$ is presented. As already stated in Section 2.2, the rationale of considering 3-crop rotation schemes is a consequence of the best agronomic practice in mediterranean pedo-climatic contexts [36], that is based on the consecutive succession of three crop types. Hence, in what follows we assume that the profits earned by cultivating a crop on a piece of land are affected by the types of the two crops preceding it on the same piece of land.

Recall that each crop can be classified into the *crop types*: renewal (set $C_R$), impoverishing (set $C_D$) or improver (set $C_M$). As reported in Section 2.2, all possible successions of crop types are allowed, but for each assigned crop $c$ a profit loss may occur depending on the types of $c$ and of the two immediately preceding crops, if they exist. Hence, since $k = 3$, to determine the profit of a crop $c$ assigned to a given time period $t \geq 3$ we have to take into account the types of the two crops immediately preceding $c$, if any. When $c$ is only preceded by one crop $b$, the profit of $c$ is determined by considering the two crop sequence $\langle b, c \rangle$, only.

From now on, we call *3-rotation* (*2-rotation*) the succession of three (two) crop types. Let $R_3$ and $R_2$ be the sets of all possible 3-rotations and 2-rotations, respectively. Obviously, the number of

possible 3-rotations (2-rotations) of the three (two) types is 27 (is 9). Given a three crop sequence $\sigma_3 = \langle c_1, c_2, c_3 \rangle$, let $r = \langle \alpha, \beta, \gamma \rangle \in R_3$ be the corresponding 3-rotation, where $\alpha, \beta, \gamma$ are the types of the crops $c_1$, $c_2$ and $c_3$, respectively. Similarly, given a two crop sequence $\sigma_2 = \langle c_1, c_2 \rangle$, let $r = \langle \alpha, \beta \rangle \in R_2$ be the corresponding 2-rotation, where $\alpha, \beta$ are the types of crops $c_1$ and $c_2$, respectively. In what follows, given a three crop sequence $\sigma_3 = \langle c_1, c_2, c_3 \rangle$ and the corresponding 3-rotation $\langle \alpha, \beta, \gamma \rangle \in R_3$, we denote as $C_\alpha$, $C_\beta$ and $C_\gamma$ the sets of all crops of type $\alpha, \beta, \gamma$, respectively. Similarly, we denote as $C_\alpha$ and $C_\beta$ the sets of types $\alpha$ and $\beta$, respectively, corresponding to the 2-rotation $\langle \alpha, \beta \rangle \in R_2$.

Given a three crop sequence $\sigma_3 = \langle c_1, c_2, c_3 \rangle$ and its corresponding 3-rotation $r = \langle \alpha, \beta, \gamma \rangle \in R_3$, we define as $L^3_{r,c_3}$ the loss, in terms of profit, on crop $c_3$ when it *activates* the rotation scheme $r \in R_3$, i.e., the profit loss for a crop of type $\gamma$ when is preceded by two crops of types $\alpha$ and $\beta$. Similarly, given a two crop sequence $\sigma_2 = \langle c_1, c_2 \rangle$ and its corresponding 2-rotation $r = \langle \alpha, \beta \rangle \in R_2$, we define as $L^2_{r,c_2}$ the loss, in terms of profit, on crop $c_2$ when it *activates* the rotation scheme $r \in R_2$. Note that, since we are assuming that a crop is necessarily assigned to each odd period in $T_o$, a 3-rotation is activated by each crop assigned to a period $t \geq 5$. On the other hand, when $t < 5$, a 3-rotation or a 2-rotation may occur, depending whether second semester crops are assigned or not to the even periods not bigger than $t$.

In the following, the decision variables used in the ILP model are listed:

- $x^t_{ch} \in \{0,1\}$ equal to 1 if crop $c \in C$ is assigned to hectare $h \in H$ in the period $t$ and 0 otherwise.

- $y^t_{rch} \in \{0,1\}$ equal to 1 if crop $c \in C$ is assigned to hectare $h \in H$ in an odd period $t \geq 5$, activating the 3-rotation scheme $r \in R_3$ and 0 otherwise.

- $y_{rch} \in \{0,1\}$ equal to 1 if crop $c \in C$ is assigned to hectare $h \in H$ in period 3 activating the 2-rotation scheme $r \in R_2$ and 0 otherwise.

- $w^t_{rch} \in \{0,1\}$ equal to 1 if the second semester crop $c \in C_S$ is assigned to hectare $h \in H$ in the even period $t$, with $t \geq 4$, activating the 3-rotation scheme $r \in R_3$ and 0 otherwise.

- $w_{rch} \in \{0,1\}$ equal to 1 if the second semester crop $c \in C_S$ is assigned to hectare $h \in H$ in period 2, activating the 2-rotation scheme $r \in R_2$ and 0 otherwise.

Let $P_c$ be the *nominal* profit attained by assigning crop $c \in C$ on a hectare to one period (or two periods in the case of annual crops) when the best crop rotation practice is followed (i.e. no profit loss occurs). Given a sequence $\sigma_3 = \langle c_1, c_2, c_3 \rangle$, let $r \in R_3$ be the 3-rotation activated by the sequence. Then, the profit $p_h(\sigma_3)$ earned by crop $c_3 \in C$ when it is preceded by crops $\langle c_1, c_2 \rangle$ on hectare $h$ is $p_h(\sigma_3) = P_{c_3} - L^3_{r,c_3}$. Similarly, we can define the profit earned by assigning the last crop $c_2 \in C$ of a 2-crop sequence $\sigma_2 = \langle c_1, c_2 \rangle$ activating the 2-rotation $r \in R_2$ as $p_h(\sigma_2) = P_{c_2} - L^2_{r,c_2}$.

Since in CRP-3 the objective is to maximize the total profit, the objective function is:

$$\max \sum_{t \in T} \sum_{h \in H} \sum_{c \in C} P_c x^t_{ch} - \sum_{h \in H} \sum_{c \in C} \sum_{r \in R_3} L^3_{r,c} \Big( \sum_{t \in T_o : t \geq 5} y^t_{rch} + \sum_{t \in T_e : t \geq 4} w^t_{rch} \Big) - \sum_{h \in H} \sum_{c \in C} \sum_{r \in R_2} L^2_{r,c} (y_{rch} + w_{rch}) \tag{2.2}$$

The constraints of the problem are reported in the following.

$$\sum_{c \in C} x^t_{ch} \leq 1, \forall h \in H, \forall t \in T \tag{2.3}$$

$$\sum_{h \in H} x^t_{ch} \leq MAX_c \ \ \forall c \in C, \forall t \in T \tag{2.4}$$

$$\sum_{h \in H} x^t_{ch} \geq MIN_c \ \ \forall c \in C, \forall t \in T \tag{2.5}$$

$$\sum_{c \in C_F \cup C_A} x_{ch}^t = 1, \forall h \in H, \forall t \in T_o \tag{2.6}$$

$$x_{ch}^t = 0, \forall c \in C_F, \forall h \in H, \forall t \in T_e \tag{2.7}$$

$$x_{ch}^t = 0, \forall c \in C_S, \forall h \in H, \forall t \in T_o \tag{2.8}$$

$$x_{ch}^t \leq \sum_{k \in C_F} x_{kh}^{t-1}, \forall c \in C_S, \forall h \in H, \forall t \in T_e \tag{2.9}$$

$$x_{ch}^{t+1} \geq x_{ch}^t \;\; \forall c \in C_A, \forall h \in H, \forall t \in T_o \tag{2.10}$$

$$x_{ch}^{t+1} \leq x_{ch}^t \;\; \forall c \in C_A, \forall h \in H, \forall t \in T_o \tag{2.11}$$

$$x_{ch}^{t+2q} \geq x_{ch}^t - x_{ch}^{t-1} \;\; \forall c \in C_A, \forall h \in H, \forall t \in T_o, \forall q = 1, \ldots, d_c - 1 \tag{2.12}$$

Constraints (2.3) state that, in each period $t$, i.e., semester, at most one crop can be assigned to each hectare. Constraints (2.4) and (2.5) impose the minimum and maximum number of hectares assigned to each crop in each period. Constraints (2.6) state that a first semester or annual crop must be assigned to each odd period on each hectare. Constraints (2.7)-(2.11) set the seeding periods for annual, first semester and second semester crops. In particular, Constraints (2.10) and (2.11) impose that an annual crop occupies both the semesters $t$ and $t+1$, with $t \in T_o$, of a given year. Constraints (2.12) state that if the annual crop $c$ is assigned to period $t$ but not to period $t-1$, then it must be assigned to periods $\{t, t+1, \ldots, 2d_c - 2\}$ (it must stay on the hectare for its duration, corresponding to $2d_c$ consecutive periods).

The Constraints (2.13) and (2.14) reported below impose the requirements on the maximum replanting. In the constraints, let $ng$ be the number of homogeneous replanting groups, $RG_i$ be the set of crops of the $i$-th group and $rep_i$ be the maximum replanting value of $RG_i$, for $i = 1, \ldots, ng$. Hence, $rep_i$ is the maximum number of times that crops in $RG_i$ can be consecutively assigned to the same hectare. More precisely, Constraints (2.13) are related to homogeneous replanting groups with no second semester crops,

i.e., when $RG_i \cap C_S = \emptyset$. Given an odd period $t$, the constraints are active when no second semester crop is assigned to even periods in $\{t+1,\ldots,t+2rep_i-1\}$ (i.e., when the sum in the right hand side is 0). Similarly, (2.14) impose the maximum replanting constraint for homogeneous replanting groups containing also second semester crops. More precisely, in the left hand side of (2.14), the first term is related to crops of $RG_i$ that are annual or first semester crops (assigned to odd periods), and the second term is related to second semester crops of $RG_i$ (assigned to even periods).

$$\sum_{c\in RG_i}\sum_{q=t:q\in T_o}^{t+2rep_i} x_{ch}^q \leq rep_i + \sum_{c\in C_S}\sum_{q=t+1:q\in T_e}^{t+2rep_i-1} x_{ch}^q \forall i : RG_i \cap C_S = \emptyset, i = 1\ldots,ng, t \in T_o \tag{2.13}$$

$$\sum_{c\in RG_i\setminus C_S}\sum_{q=t:q\in T_o}^{t+p} x_{ch}^q + \sum_{c\in RG_i\cap C_S}\sum_{q=t:q\in T_e}^{t+p} x_{ch}^q \leq rep_i + \sum_{c\in C\setminus RG_i}\sum_{q=t}^{t+p} x_{ch}^q,$$
$$\forall i : RG_i \cap C_S \neq \emptyset, i = 1\ldots,ng, p = rep_i+1,\ldots,2rep_i+2, \forall h \in H, t \in T \tag{2.14}$$

In the following, the constraints on the crop rotation are presented. Recall that a 3-rotation $r \in R_3$ is the succession $r = \langle \alpha,\beta,\gamma\rangle$ of three crop of types $\alpha,\beta$ and $\gamma$, respectively, while a 2-rotation $r \in R_2$ is the succession $r = \langle \alpha,\beta\rangle$ of two crop of types $\alpha$ and $\beta$. These constraints force variables $y$ and $w$ to be 1 when specific rotation schemes occur, accounting for different cases depending on how many second semester crops are cultivated in a given crop sequence. In particular, Constraints (2.15)–(2.17) consider odd periods $t \in T_o$ only, and account for all combinations of the two preceding crops. As an example, Constraints (2.15) are related to the case in which the two crops preceding crop $c$ in $t$ are annual or first semester crops (i.e., no second semester crop is cultivated in even periods $t-1$ and $t-3$) assigned to periods $t-2$ and $t-4$. On the other hand, Constraints (2.18) and (2.19) consider even periods $t \in T_e$ only, and consider all possible combinations of the two preceding crops. As an example, Constraints (2.18) impose the related variable $w$ to be equal to 1 when there is a second semester crop $c \in C_S$ in $t \in T_e$ and the two crops preceding $c$ are in $t-1$ (first semester crop) and $t-3$ (annual or first semester crop), i.e., there is no second semester crop cultivated

in $t-2$. Finally, Constraints (2.20) and (2.21) are used to activate the variables related to the 2-rotations: Constraints (2.20) cover the case in which there is a 2-rotation given by crops cultivated in period $t=3$ and $t-2$ (i.e., there is no second semester crop in $t-1$), while Constraints (2.21) are related to the case of a 2-rotation given by crops cultivated in periods $t=2$ (second semester crop) and $t-1$.

$$y^t_{rch} \geq x^t_{ch} + \sum_{i \in C_\alpha} x^{t-4}_{ih} + \sum_{i \in C_\beta} x^{t-2}_{ih} - \sum_{i \in C_S} x^{t-1}_{ih} - \sum_{i \in C_S} x^{t-3}_{ih} - 2,$$
$$\forall r = \langle \alpha, \beta, \gamma \rangle \in R_3, \forall c \in C_\gamma, \forall h \in H, t \in T_o, t \geq 5 \tag{2.15}$$

$$y^t_{rch} \geq x^t_{ih} + \sum_{i \in C_\beta} x^{t-1}_{ih} + \sum_{i \in C_\alpha} x^{t-2}_{ih} + \sum_{i \in C_S} x^{t-1}_{ih} - 3,$$
$$\forall r = \langle \alpha, \beta, \gamma \rangle \in R_3, \forall c \in C_\gamma, \forall h \in H, \forall t \in T_o, t \geq 3 \tag{2.16}$$

$$y^t_{rch} \geq x^t_{ih} + \sum_{i \in C_\beta} x^{t-2}_{ih} + \sum_{i \in C_\alpha} x^{t-3}_{ih} + \sum_{i \in C_S} x^{t-3}_{ih} - \sum_{i \in C_S} x^{t-1}_{ih} - 3,$$
$$\forall r = \langle \alpha, \beta, \gamma \rangle \in R_3, \forall c \in C_\gamma, \forall h \in H, \forall t \in T_o, t \geq 5 \tag{2.17}$$

$$w^t_{rch} \geq x^t_{ch} + \sum_{i \in C_\beta} x^{t-1}_{ih} + \sum_{i \in C_\alpha} x^{t-3}_{ih} - \sum_{i \in C_S} x^{t-2}_{ih} - 2,$$
$$\forall r = \langle \alpha, \beta, \gamma \rangle \in R_3, \forall c \in C_\gamma, c \in C_S, \forall h \in H, \forall t \in T_e, t \geq 4 \tag{2.18}$$

$$w^t_{rch} \geq x^t_{ch} + \sum_{i \in C_\beta} x^{t-1}_{ih} + \sum_{i \in C_\alpha} x^{t-2}_{ih} + \sum_{i \in C_S} x^{t-2}_{ih} - 3,$$
$$\forall r = \langle \alpha, \beta, \gamma \rangle \in R_3, \forall c \in C_\gamma, c \in C_S, \forall h \in H, \forall t \in T_e, t \geq 4 \tag{2.19}$$

$$y_{rce} \geq x^t_{ch} + \sum_{i \in C_\alpha} x^{t-2}_{ih} - \sum_{i \in C_S} x^{t-1}_{ih} - 1,$$
$$\forall r = \langle \alpha, \beta \rangle \in R_2, \forall c \in C_\beta, \forall h \in H, t = 3 \tag{2.20}$$

$$w_{rce} \geq x^t_{ch} + \sum_{i \in C_\alpha} x^{t-1}_{ih} - 1, \forall r = \langle \alpha, \beta \rangle \in R_2, \forall c \in C_\beta, c \in C_S, \forall h \in H, t = 2 \tag{2.21}$$

### 2.5.1 Constraints of the CAP farmer scenario

As already mentioned in Section 2.2.2, farmers observing CAP regulation have benefits and incentives, but have to follow strict ecological and diversification rules. These rules are described into detail in Section

2.2.2.

The diversification constraints can be written as

$$\sum_{h \in H} x^t_{ch} \leq 0.75|H|, \forall c \in C, \forall t \in T_o. \tag{2.22}$$

Constraints (2.22) state that at least two different (annual or first semester) crops must be cultivated each year, i.e., in each odd period, and that no crop can be assigned to more than the 75% of the farmland. Recall that, as reported in Section 2.2.2, such constraints hold only to farms with total land between 10 and 30 hectares. Note that, since we assume that a crop is assigned to each odd period, Constraints (2.22) also imply that at least two crops will be cultivated in each year (which is another CAP requirement).

Let $C_{leg}$ be the set of legume crops, the following Constraints (2.23) and (2.24) only exist for farmlands with more than 30 hectares.

$$\sum_{h \in H} (x^t_{c_1 h} + x^t_{c_2 h}) \leq 0.95|H|, \forall c_1, c_2 \in C, c_1 \neq c_2, \forall t \in T_o \tag{2.23}$$

$$\sum_{c \in C_{leg}} \sum_{h \in H} x^t_{ch} \geq 0.05|H|, \forall t \in T_o \tag{2.24}$$

Constraints (2.23) state that the two most cultivated crops can not exceed the 95% of the farmland, each year. The Constraints (2.24) impose a minimum of 5% of the whole land to be cultivated in legumes each year.

### 2.5.2 CAP+SVC scenario

In this scenario, in addition to the CAP constraints, farmers have to respect the policies introduced by the Carta del Mulino initiative (see Section Section 2.2.3). CdM identifies the set of crops of interest of the initiative (i.e., wheat), in what follows denoted as $C_{CdM}$.

Recall that, the CdM initiative impose three types of requirements: $(i)$ greening constraints, $(ii)$ diversification constraints, $(iii)$ repetition constraints. For the three types, each set of constraints covers a specific case, characterized by the set of (even) periods assigned to second semester crops. In the following, as an example, one constraint set for each of the above three types is presented. The constraints related to the other cases are reported in the appendix in Section 2.9.

$$\sum_{i\in C_L}\sum_{q=t+1}^{t+4} x_{ih}^{q} \geq x_{ch}^{t} + \sum_{i\in C_S} x_{ih}^{t+1} + \sum_{i\in C_S} x_{ih}^{t+3} - 2, \forall c \in C_{CdM}, \forall h \in H, \forall t \in T_o \tag{2.25}$$

Constraints (2.25) are greening constraints related to the case in which a CdM crop is assigned to period $t$ and second semester crops are cultivated in $t+1$ and $t+3$. In (2.25), $C_{leg}$ and $C_{oil}$ are the sets of legume and oil crops, respectively, and $C_L = C_{leg} \cup C_{oil}$. They impose that at least a crop in $C_L$ must be assigned on $h$ from period $t+1$ to $t+4$.

$$\sum_{i\in\{c,v\}} (x_{ih}^{t+1} + x_{ih}^{t+2} + x_{ih}^{t+4} + x_{ih}^{t+6}) \leq 3 + (1 - x_{vh}^{t}) + (1 - \sum_{i\in C_S} x_{ih}^{t+1}) + \sum_{c\in C_S} x_{ih}^{t+3} + \sum_{i\in C_S} x_{ih}^{t+5},$$
$$\forall v \in C_{CdM}, c \in C, h \in H, t \in T_o \tag{2.26}$$

Constraints (2.26) are the CdM diversification constraints related to the case in which a CdM crop $v$ is assigned to hectare $h$ in the odd period $t$ and a second semester crop is assigned in $t+1$, but no second semester crop is assigned to $t+3$ and $t+5$. Hence, the seeding periods for the four crops succeeding $v$ are $t+1, t+2, t+4, t+6$. The constraints state that, for each crop $c \in C$, the crops $v$ and $c$ can not be assigned in more than 3 periods in $\{t+1, t+2, t+4, t+6\}$ (allowing the assignment of a crop different from $v$ and $c$ in these periods).

$$\sum_{c\in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + (1 - \sum_{i\in C_S} x_{ih}^{t+1}) + \sum_{c\in C_S} x_{ih}^{t+3} + \sum_{i\in C_S} x_{ih}^{t+5} + (1 - x_{vh}^{t}),$$

$$\forall v \in C_{CdM}, \forall h \in H, \forall t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega \tag{2.27}$$

Constraints (2.27) are the CdM repetition constraints covering the case in which a CdM crop $v$ is assigned to hectare $h$ in period $t$, and second semester crops are cultivated in $t+1$, but not in $t+3$ and $t+5$. In (2.27), let $CR$ be the set of crops subject to the repetition constraints. The constraints impose that crops in $CR$ can be repeated at most once in the seeding periods $t, t+1, t+2, t+4, t+6$. In the constraints, $\Omega$ denotes the set of all triples $\{t_1, t_2, t_3\}$ of 3 consecutive periods chosen among the seeding periods considered in each constraint (i.e., $\{t, t+1, t+2, t+4, t+6\}$ in this specific case). Hence, in Constraints (2.27), $\Omega$ is the set of triples $\{\{t, t+1, t+2\}, \{t+1, t+2, t+4\}, \{t+2, t+4, t+6\}\}$.

## 2.6 Instance description and real data

The real data used in this study have been collected from a panel of Italian farms involved in the CdM [52] initiative (described in Section 2.2.3 into detail), promoted by the Barilla Group in collaboration with WWF Italy, University of Bologna, University of Tuscia and Open Fields. The farms have been identified in the Italian provinces with the greater arable land cultivated with soft wheat for the year 2018 (1st year of the CdM initiative), compared to the total arable land involved in crop rotation. In total, 23 farms have been selected, with a land surface ranging from 19 to 229 hectares. In the experimental campaign, the set of crops $C$ considered for each farm includes the crops already grown during the years of data collection and other crops that could potentially be cultivated. In fact, our models can be also employed to evaluate the potential benefits arising from the cultivation of crops currently not considered by the farmers. In the instances, the number of crops available for each farm ranges from 5 to 10. The details of each instance, in terms of number of considered crops and number of hectares of the arable land, are reported in the Columns 1–3 of Table 2.2.

Since the real data collected from the farms include the crops already planned during the first year of

the CdM initiative (i.e., year 2018), two different sets of experiments have been conducted for the CAP and CAP+SVC scenarios (denoted as *Set 1* and *Set 2* in the following). In Set 1, crop planning decisions are only taken from the second year on, keeping the planning of the first year fixed according to the real data. In fact, the seeding plan for the first year was established and declared by the farmers, in order to satisfy the CAP regulations and join the CdM initiative. In Set 2, also the first year is included in the crop planning decision process. Note that, since the Pure farmer scenario is not actually followed by any of the selected farms, no real data are available for the first year of planning. Hence, in all the experiments on this scenario we always include the first year in the decision process.

The analysis of the "Used Agricultural Area" of the farms showed differences in terms of production specialization of the arable land. These differences are due to the different cultivation areas and the different cultivation techniques adopted. All the farms are specialized in cereals production (wheat and corn), but all of them have a legume crop or an oil crop. However, the distribution of crop areas within farms is not uniform.

In order to reconstruct, for each farm, the profits obtained by the cultivation of the arable land with the available crops, economic surveys and official data sources have been used. More precisely, the crop profits are determined by factors such as crop prices, yields, cultivation costs and incentives. The main data sources exploited are summarized below. The crop prices were obtained by using: annual prices available on official commodities exchange (AGER – Bologna [32]); prices annually fixed by contracts for industrial crops; official Italian Ministry of Agriculture decree. Yields are set according to the National Agricultural Information System [54]. The system records all the crops yields in Italy for year and municipality. For the CAP incentives, the real farms' data for all the years have been used. For the cultivation costs, composed by technical inputs and operation costs, farms' real figures collected during surveys have been employed.

As mentioned in Section 3, when crop rotation does not follow the best agronomic practices, a cost increase occurs depending on the specific crop succession in the rotation scheme. In our experiments, we

assume a cost increase ranging from 0% (best agronomic practice) to 30%, depending on the specific rotation of crop types (i.e., renewal, improver, impoverishing). The cost increases considered in the experimental campaign are reported in Table 2.1. As an example, when three impoverishing crops are consecutively assigned to the same plot (the first rotation in Table 2.1), a cost increase of 30% (20%) occurs when the third (second) crop is cultivated.

| Crop Rotation scheme | | | Cost Increase | | Crop Rotation scheme | | | Cost Increase | |
|---|---|---|---|---|---|---|---|---|---|
| Period 0 | Period 1 | Period 2 | Period 1 | Period 2 | Period 0 | Period 1 | Period 2 | Period 1 | Period 2 |
| Impoverishing | Impoverishing | Impoverishing | 20% | 30% | Renewal | Renewal | Improver | 10% | 0 |
| Impoverishing | Impoverishing | Renewal | 20% | 0% | Renewal | Improver | Impoverishing | 0 | 0 |
| Impoverishing | Impoverishing | Improver | 20% | 0% | Renewal | Improver | Renewal | 0 | 0 |
| Impoverishing | Renewal | Impoverishing | 0 | 10% | Renewal | Improver | Improver | 0 | 10% |
| Impoverishing | Renewal | Renewal | 0 | 10% | Improver | Impoverishing | Impoverishing | 0 | 20% |
| Impoverishing | Renewal | Improver | 0 | 0 | Improver | Impoverishing | Renewal | 0 | 0 |
| Impoverishing | Improver | Impoverishing | 0 | 0 | Improver | Impoverishing | Improver | 0 | 0 |
| Impoverishing | Improver | Renewal | 0 | 0 | Improver | Renewal | Impoverishing | 0 | 0 |
| Impoverishing | Improver | Improver | 0 | 10% | Improver | Renewal | Renewal | 0 | 10% |
| Renewal | Impoverishing | Impoverishing | 0 | 20% | Improver | Renewal | Improver | 0 | 0 |
| Renewal | Impoverishing | Renewal | 0 | 10% | Improver | Improver | Impoverishing | 10% | 0 |
| Renewal | Impoverishing | Improver | 0 | 0 | Improver | Improver | Renewal | 10% | 0 |
| Renewal | Renewal | Impoverishing | 10% | 0 | Improver | Improver | Improver | 0 | 10% |
| Renewal | Renewal | Renewal | 10% | 20% | | | | | |

Table 2.1: Cost Increase of the rotations.

## 2.7 Experimental results

In this section, the results of the computational campaign on the MIP models introduced in Section 2.5 are presented on the real instances described in the previous section. The experiments have been performed using the Gurobi Optimizer version 9.0.1 on a 2.5 GHz Quad-Core computer equipped with 16 GB of RAM. In particular, all the real world instances have been solved in the three scenarios of the problem: Pure farmer, CAP farmer and CAP+SVC farmer (see Sections 2.2.1–2.2.3). The aim of the experimental campaign is twofold: (1) to evaluate the effectiveness of the sustainable policies proposed by the CAP and CdM regulations, with respect to the case in which no sustainable initiative is followed (recall that, this last case corresponds to the Pure farmer scenario); (2) to assess the ability of the proposed MIP models to solve

real-life instances.

A five year planning horizon has been considered in all the instances. Hence, $T = \{1, 2, \ldots, 10\}$. As already stated in the above section, two sets of experiments have been performed for both the CAP and CAP+SVC scenarios. In Set 1, the crop planning of the first year is already given and set according to the real data. In Set 2, the CAP and CAP+SVC scenarios were solved on the overall 5-year planning horizon, including the first year in the planning process. Recall that, in the experiments, the crop planning of the first year in the Pure farmer scenario is always included in the decision process, i.e., not given.

Table 2.2 reports the results of the Pure farmer scenario and those of Set 1 for the CAP and CAP+SVC farmer scenarios. Columns 1, 2 and 3 of the table respectively report the data of the instances: id, number of hectares and number of crops of each farm. Columns 4–6, 7–10 and 11–12 report the results for the three scenarios of the problem. For each scenario, the optimal solution value (the profit in Euros), called "Solution", the computing time (in seconds) and the number of Branch & Bound nodes explored by Gurobi are reported. Furthermore, for the CAP and CAP+SVC scenarios, we also report the profit increases (in %) with respect to the Pure farmer. Such increases are denoted as $\Delta(P-C)$ and $\Delta(P-SVC)$ and computed as $(Solution_{CAP} - Solution_{Pure})/Solution_{Pure} \times 100$ and $(Solution_{CAP+SVC} - Solution_{Pure})/Solution_{Pure} \times 100$, respectively.

In the table, the instances are ordered by increasing number of hectares, ranging from 19 to 229 hectares. Note that the number of the available crops does not vary as much in the instances, ranging from 5 to 10, implying that the instance dimension is mainly determined by the number of hectares. The last row of the table reports the average values on all the instances.

In Table 2.2, a comparison of the "Solution" values (i.e., the profits) of the three scenarios and of the profit increases $\Delta(P-C)$ and $\Delta(P-SVC)$ show that the CAP and CAP+SVC attain higher profits than the Pure farmer. Note that, with respect to the Pure farmer, the profits obtained with the CAP farmer (CAP+SVC farmer) are 79% (94%) higher in Set 1, on average. This results show how the incentives introduced by

sustainable policies, such as the Common Agricultural Policy by the European Union and a Sustainable Value Chain private initiative, can be profitable for farmers and encourage them to join such initiatives.

Regarding the computational times, they generally increase with the number of hectares, even though there are some exceptions that may depend on the particular structure of some instances. On average, the computing time increases by 470% going from the Pure farmer scenario to the CAP farmer scenario and by a further 121% in the CAP+SVC farmer case. These huge increases are probably due to the higher number of constraints added to model the CAP and CAP+SVC scenarios, as shown in Section 2.5. As expected, a general increase can be also observed in the number of nodes explored by the Branch & Bound procedure performed by Gurobi. The maximum computation time is registered on the instance "EMR_BO1" for the CAP+SVC scenario, solved in about 27 minutes, while the maximum computation times for the CAP and the Pure farmer scenarios are around 8 minutes and 1 minute, respectively. Note that these computation times are reasonable, proving a good ability of the models to solve this kind of problems for real-life instances.

Table 2.3 shows the results of the experiments of Set 2 on the CAP and CAP+SVC scenarios (to simplify the reading we also retrieve the results of the Pure farmer scenario already reported in Table 2.2). Recall that, in these experiments, the CAP and CAP+SVC scenarios are solved with the crop planning of the first year not set, but included in the decision process. From one side, these results allow to evaluate the benefits of fully joining sustainability initiatives. In fact, since the first year is included in the decision process, it is possible to establish the actual convenience of joining them. On the other side, the results of Set 2 allow to assess the additional computational effort required for solving larger instances. As expected, the objective solution values (i.e., the profits) in Set 2 are higher than in Set 1, since the problem is less constrained, being the first year included in the decision process. More precisely, as shown by the $\Delta(P-C)$ and $\Delta(P-SVC)$ values, in Set 2 the CAP and the CAP+SVC scenarios attain profits that are, on average, 94% and 95% higher than those obtained with the Pure farmer scenario. These results suggest that the optimization models can help farmers to increase their profits when they attempt to join sustainability programs.

The computational times of CAP and CAP+SVC in Set 2 are 77% and 6% higher, respectively, than those of CAP and CAP+SVC in Set 1. Nevertheless, all the instances in Set 2 are optimally solved with maximum and average computational times smaller than 20 minutes and 150 seconds, respectively, meaning that the models are still able to solve real world instances in reasonable time. Note that, in Set 2, the solution values of some instances are the same in the CAP and CAP+SVC scenarios. This happens when the CdM crop is never cultivated in the optimal solution of the CAP+SVC scenario (recall that the CdM crop is the only crop that affects the decision process in the CAP+SVC scenario), implying that the optimal solution of the two scenarios is indeed the same.

To conclude, the models perform well on real-life instances, implying that the models could be embedded in a decision support tool that can be practically used by farmers to plan their production and by regulatory bodies to tune the incentives of sustainability programs.

| Instance | | | Pure Farmer | | | CAP Farmer | | | | CAP+SVC Farmer | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | # Hectares | # Crops | Solution | Time | B&B nodes | Solution | Time | $\Delta(P-C)(\%)$ | B&B nodes | Solution | Time | $\Delta(P-SVC)(\%)$ | B&B nodes |
| PIE_TO1 | 19 | 5 | 75,009.91 | 0.22 | 1 | 106,746.82 | 0.3 | 42 | 1 | 107,852.47 | 0.34 | 53 | 1 |
| EMR_FE3 | 23 | 7 | 89,596.04 | 0.29 | 1 | 136,005.36 | 0.28 | 52 | 0 | 140,252.79 | 0.28 | 53 | 0 |
| EMR_FE4 | 27 | 9 | 102,535.20 | 11.59 | 1014 | 171,966.34 | 9.18 | 68 | 1122 | 177,139.09 | 3.94 | 74 | 892 |
| EMR_PR1 | 29 | 10 | 174,676.28 | 1.43 | 1 | 159,307.75 | 3.11 | -9 | 143 | 166,551.05 | 17.06 | 24 | 1223 |
| PIE_CN2 | 33 | 10 | 103,718.36 | 0.62 | 0 | 175,718.90 | 8.19 | 69 | 279 | 177,563.58 | 6.72 | 79 | 1 |
| VEN_RO1 | 49 | 6 | 69,893.60 | 1.27 | 1 | 186,722.88 | 7.1 | 167 | 1111 | 189,916.02 | 24.28 | 179 | 8472 |
| LMB_MI2 | 51 | 7 | 127,933.50 | 0.5 | 0 | 288,669.32 | 5.52 | 126 | 331 | 290,319.04 | 6.84 | 131 | 329 |
| LMB_MI1 | 53 | 8 | 103,948.90 | 0.62 | 0 | 256,200.01 | 2.58 | 146 | 45 | 257,995.15 | 5.16 | 166 | 868 |
| PIE_AL3 | 54 | 8 | 127,296.15 | 11.09 | 1102 | 254,348.29 | 27.19 | 100 | 2859 | 260,300.14 | 21.84 | 105 | 2592 |
| PIE_CN1 | 62 | 7 | 223,110.72 | 0.99 | 1 | 501,914.97 | 4.16 | 125 | 1 | 458,095.24 | 6 | 137 | 1 |
| LMB_MN2 | 66 | 9 | 210,945.24 | 24.67 | 3251 | 410,982.30 | 6.21 | 95 | 1 | 418,645.00 | 23.99 | 98 | 1280 |
| EMR_BO3 | 72 | 7 | 186,914.88 | 0.67 | 1 | 340,864.76 | 0.77 | 82 | 0 | 347,632.28 | 3.71 | 88 | 35 |
| PIE_AL4 | 84 | 5 | 236,728.23 | 7.23 | 1257 | 380,739.48 | 9.39 | 61 | 126 | 402,581.16 | 3.54 | 79 | 1 |
| LMB_MN3 | 84 | 10 | 1,047,807.99 | 42.81 | 1187 | 1,319,890.44 | 101.83 | 26 | 1185 | 1,418,083.42 | 12.41 | 51 | 1 |
| LMB_MN1 | 86 | 10 | 980,863.14 | 10.23 | 1 | 1,085,012.45 | 4.56 | 11 | 1 | 1,092,714.29 | 5.56 | 25 | 1 |
| EMR_FE2 | 89 | 9 | 285,202.28 | 1.1 | 1 | 454,570.68 | 31.94 | 59 | 1086 | 465,987.04 | 66.28 | 65 | 1223 |
| PIE_AL2 | 120 | 7 | 553,195.20 | 4.16 | 1 | 1,186,593.58 | 43.47 | 114 | 2137 | 1,256,405.84 | 22.74 | 132 | 60 |
| VEN_RO2 | 134 | 8 | 2,246,992.67 | 2.53 | 0 | 2,512,784.63 | 122.17 | 12 | 1133 | 2,702,264.75 | 65.19 | 30 | 1250 |
| PIE_AL1 | 154 | 10 | 1,069,265.12 | 10.08 | 1 | 1,496,261.38 | 171.6 | 40 | 2381 | 1,588,214.07 | 54.18 | 66 | 65 |
| EMR_BO1 | 161 | 10 | 494,902.73 | 36.79 | 302 | 1,033,833.48 | 42.49 | 109 | 296 | 1,086,321.08 | 1,638.96 | 131 | 33506 |
| EMR_FE1 | 177 | 9 | 653,204.34 | 2.71 | 1 | 1,089,875.38 | 290.41 | 67 | 1375 | 1,092,432.94 | 276.78 | 073 | 1183 |
| VEN_VE1 | 211 | 7 | 735,681.04 | 7.47 | 1 | 1,506,860.26 | 33.37 | 105 | 1 | 1,669,041.69 | 99.85 | 152 | 1051 |
| EMR_BO2 | 229 | 10 | 667,741.10 | 72.4 | 1631 | 1,670,885.66 | 508.08 | 150 | 1177 | 1,723,333.04 | 806.39 | 168 | 1447 |
| Average | | | 459,441.85 | 10.93 | 424.17 | 727,250.22 | 62.34 | 79 | 730.04 | 760,419.18 | 137.91 | 94 | 2,412.26 |

Table 2.2: Results of Set 1 of experiments.

| Instance | | | Pure Farmer | | | CAP Farmer | | | | CAP+SVC Farmer | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Id | # Hectares | # Crops | Solution | Time | B&B nodes | Solution | Time | $\Delta(P-C)(\%)$ | B&B nodes | Solution | Time | $\Delta(P-SVC)(\%)$ | B&B nodes |
| PIE_TO1 | 19 | 5 | 75,009.91 | 0.22 | 1 | 114,458.93 | 0.34 | 44 | 1 | 114,458.93 | 0.34 | 53 | 1 |
| EMR_FE3 | 23 | 7 | 89,596.04 | 0.29 | 1 | 137,409.17 | 0.44 | 57 | 1 | 140,463.11 | 0.35 | 57 | 1 |
| EMR_FE4 | 27 | 9 | 102,535.20 | 11.59 | 1014 | 178,211.10 | 24.54 | 73 | 5071 | 183,513.55 | 10.83 | 79 | 1526 |
| EMR_PR1 | 29 | 10 | 174,676.28 | 1.43 | 1 | 216,608.02 | 4.19 | -5 | 161 | 218,126.03 | 18.53 | 25 | 7248 |
| PIE_CN2 | 33 | 10 | 103,718.36 | 0.62 | 0 | 185,834.70 | 2.92 | 71 | 1 | 187,629.20 | 6.38 | 81 | 5 |
| VEN_RO1 | 49 | 6 | 69,893.60 | 1.27 | 1 | 195,225.70 | 17.19 | 172 | 2373 | 195,225.70 | 40.62 | 179 | 4236 |
| LMB_MI2 | 51 | 7 | 127,933.50 | 0.50 | 0 | 295,682.29 | 8.16 | 127 | 1386 | 295,739.17 | 30.10 | 131 | 1696 |
| LMB_MI1 | 53 | 8 | 103,948.90 | 0.62 | 0 | 276,281.55 | 9.69 | 148 | 2241 | 276,292.83 | 110.59 | 166 | 25585 |
| PIE_AL3 | 54 | 8 | 127,296.15 | 11.09 | 1102 | 260,415.19 | 35.65 | 104 | 2163 | 266,395.87 | 64.02 | 109 | 1542 |
| PIE_CN1 | 62 | 7 | 223,110.72 | 0.99 | 1 | 528,035.50 | 8.92 | 105 | 538 | 491,986.96 | 9.49 | 121 | 1 |
| LMB_MN2 | 66 | 9 | 210,945.24 | 24.67 | 3251 | 417,575.48 | 39.69 | 98 | 2144 | 426,448.08 | 108.44 | 102 | 2092 |
| EMR_BO3 | 72 | 7 | 186,914.88 | 0.67 | 1 | 351,828.16 | 4.90 | 86 | 141 | 353,861.36 | 4.43 | 89 | 7 |
| PIE_AL4 | 84 | 5 | 236,728.23 | 7.23 | 1257 | 423,140.48 | 4.43 | 70 | 1 | 428,269.91 | 4.31 | 81 | 1 |
| LMB_MN3 | 84 | 10 | 1,047,807.99 | 42.81 | 1187 | 1,587,311.80 | 86.69 | 35 | 1144 | 1,589,201.22 | 232.93 | 52 | 4731 |
| LMB_MN1 | 86 | 10 | 980,863.14 | 10.23 | 1 | 1,226,554.67 | 21.08 | 11 | 394 | 1,233,986.27 | 45.48 | 26 | 1049 |
| EMR_FE2 | 89 | 9 | 285,202.28 | 1.10 | 1 | 469,323.98 | 47.37 | 63 | 3381 | 479,495.57 | 293.33 | 68 | 2123 |
| PIE_AL2 | 120 | 7 | 553,195.20 | 4.16 | 1 | 1,283,437.44 | 51.31 | 127 | 343 | 1,289,863.20 | 93.38 | 133 | 1059 |
| VEN_RO2 | 134 | 8 | 2,246,992.67 | 2.53 | 0 | 2,931,218.51 | 60.84 | 20 | 2413 | 2,931,218.51 | 64.29 | 30 | 2144 |
| PIE_AL1 | 154 | 10 | 1,069,265.12 | 10.08 | 1 | 1,778,159.32 | 352.50 | 49 | 7950 | 1,778,159.32 | 240.08 | 66 | 2146 |
| EMR_BO1 | 161 | 10 | 494,902.73 | 36.79 | 302 | 1,143,805.91 | 790.44 | 120 | 1378 | 1,148,847.93 | 289.54 | 132 | 717 |
| EMR_FE1 | 177 | 9 | 653,204.34 | 2.71 | 1 | 1,130,892.09 | 223.34 | 67 | 1165 | 1,130,892.09 | 369.50 | 73 | 1252 |
| VEN_VE1 | 211 | 7 | 735,681.04 | 7.47 | 1 | 1,856,487.73 | 281.39 | 127 | 1132 | 1,860,657.17 | 268.33 | 153 | 1074 |
| EMR_BO2 | 229 | 10 | 667,741.10 | 72.40 | 1631 | 1,786,250.55 | 467.94 | 158 | 2163 | 1,810,098.89 | 1,088.13 | 171 | 1293 |
| Average | | | 459.441,85 | 10,93 | 424,17 | 816.267,32 | 110,61 | 84 | 1.638,48 | 818.731,78 | 147,54 | 95 | 2.675,17 |

Table 2.3: Results of Set 2 of experiments.

## 2.8 Conclusions

In this chapter, the decision problem of crop planning in sustainable agriculture taking into account crop rotation benefits across growing seasons is considered. A formal problem characterization is given and a complexity analysis is performed. A polynomial minimum cost network flow approach is proposed for special cases. Integer Linear Programming models including all the problem characteristics have been developed for the case in which the rotation is based on sequences of $k = 3$ crops, and tested on real data. Recall that, as stated by classical agronomic literature, $k = 3$ is identified as best agronomic practice in Mediterranean pedo-climatic contexts and prescribed by Italian regulations on organic agriculture [44].

An experimental campaign on real-world instances shows that the proposed approaches could be embedded in a decision support tool that can be practically used by farmers to plan their production and to evaluate the convenience of joining sustainability initiatives. On the other hand, the models can also be employed by regulatory bodies to tune the incentives of such initiatives. The policy makers will be able to use the results to assess the suitability of rules for farm sustainability transition. In fact, the comparison of the farmers' profitability among different scenarios will facilitate the design of adequate business tools by agri-food

value-chain actors aimed to pull farmers into more sustainable and profitable arable land management.

Future research directions include $(i)$ the extension of the network flow approach to more general cases (e.g., considering the constraints arising from the CAP regulation), $(ii)$ the development of ad-hoc models for solving CRP-$k$ in other pedo-climatic contexts.

## 2.9 Appendix: remaining constraints of the CAP+SVC scenario

The greening constraints read as follows:

$$\sum_{i\in C_L}\sum_{q=t+1}^{t+5} x_{ih}^{q} \geq x_{ch}^{t} + \sum_{i\in C_S} x_{ih}^{t+1} - \sum_{i\in C_S} x_{ih}^{t+3} + \sum_{i\in C_S} x_{ih}^{t+5} - 2, \forall c \in C_{CdM}, \forall h \in H, \forall t \in T_o \tag{2.28}$$

$$\sum_{i\in C_L}\sum_{q=t+1}^{t+5} x_{ih}^{q} \geq x_{ch}^{t} - \sum_{i\in C_S} x_{ih}^{t+1} + \sum_{i\in C_S} x_{ih}^{t+3} + \sum_{i\in C_S} x_{ih}^{t+5} - 2, \forall c \in C_{CdM}, \forall h \in H, \forall t \in T_o \tag{2.29}$$

$$\sum_{i\in C_L}\sum_{q=t+1}^{t+6} x_{ih}^{q} \geq x_{ch}^{t} - \sum_{i\in C_S} x_{ih}^{t+1} - \sum_{i\in C_S} x_{ih}^{t+3} + \sum_{i\in C_S} x_{ih}^{t+5} - 1, \forall c \in C_{CdM}, \forall h \in H, \forall t \in T_o \tag{2.30}$$

$$\sum_{i\in C_L}\sum_{q=t+1}^{t+7} x_{ih}^{q} \geq x_{ch}^{t} - \sum_{i\in C_S} x_{ih}^{t+1} - \sum_{i\in C_S} x_{ih}^{t+3} - \sum_{i\in C_S} x_{ih}^{t+5} + \sum_{i\in C_S} x_{ih}^{t+7} - 1, \forall c \in C_{CdM}, \forall h \in H, \forall t \in T_o \tag{2.31}$$

$$\sum_{i\in C_L}\sum_{q=t+1}^{t+8} x_{ih}^{q} \geq x_{ch}^{t} - \sum_{i\in C_S} x_{ih}^{t+1} - \sum_{i\in C_S} x_{ih}^{t+3} - \sum_{i\in C_S} x_{ih}^{t+5} - \sum_{i\in C_S} x_{ih}^{t+7}, \forall c \in C_{CdM}, \forall h \in H, \forall t \in T_o \tag{2.32}$$

The diversification constraints read as follows:

$$\sum_{i\in\{c,v\}} (x_{ih}^{t+1} + x_{ih}^{t+2} + x_{ih}^{t+4} + x_{ih}^{t+5}) \leq 3 + (1 - x_{vh}^{t}) + (1 - \sum_{i\in C_S} x_{ih}^{t+1}) + \sum_{c\in C_S} x_{ih}^{t+3} + (1 - \sum_{i\in C_S} x_{ih}^{t+5}) \tag{2.33}$$
$$\forall v \in CdM, \forall c \in C, \forall h \in H, t \in T_o$$

$$\sum_{i\in\{c,v\}} (x_{ih}^{t+1} + x_{ih}^{t+2} + x_{ih}^{t+3} + x_{ih}^{t+4}) \leq 3 + (1 - x_{vh}^{t}) + (1 - \sum_{i\in C_S} x_{ih}^{t+1}) + (1 - \sum_{c\in C_S} x_{ih}^{t+3}) \tag{2.34}$$

$$\forall v \in CdM, \forall c \in C, \forall h \in H, t \in T_o$$

$$\sum_{i \in \{c,v\}} (x_{ih}^{t+2} + x_{ih}^{t+3} + x_{ih}^{t+4} + x_{ih}^{t+6}) \leq 3 + (1 - x_{vh}^{t}) + \sum_{i \in C_S} x_{ih}^{t+1} + (1 - \sum_{c \in C_S} x_{ih}^{t+3}) + \sum_{i \in C_S} x_{ih}^{t+5} \quad (2.35)$$
$$\forall v \in CdM, \forall c \in C, \forall h \in H, t \in T_o$$

$$\sum_{i \in \{c,v\}} (x_{ih}^{t+2} + x_{ih}^{t+3} + x_{ih}^{t+4} + x_{ih}^{t+5}) \leq 3 + (1 - x_{vh}^{t}) + \sum_{i \in C_S} x_{ih}^{t+1} + (1 - \sum_{c \in C_S} x_{ih}^{t+3}) + (1 - \sum_{i \in C_S} x_{ih}^{t+5}) \quad (2.36)$$
$$\forall v \in CdM, \forall c \in C, \forall h \in H, t \in T_o$$

$$\sum_{i \in \{c,v\}} (x_{ih}^{t+2} + x_{ih}^{t+4} + x_{ih}^{t+5} + x_{ih}^{t+6}) \leq 3 + (1 - x_{vh}^{t}) + \sum_{i \in C_S} x_{ih}^{t+1} + \sum_{c \in C_S} x_{ih}^{t+3} + (1 - \sum_{i \in C_S} x_{ih}^{t+5}) \quad (2.37)$$
$$\forall v \in CdM, \forall c \in C, \forall h \in H, t \in T_o$$

$$\sum_{i \in \{c,v\}} (x_{ih}^{t+2} + x_{ih}^{t+4} + x_{ih}^{t+6} + x_{ih}^{t+7}) \leq 3 + (1 - x_{vh}^{t}) + \sum_{i \in C_S} x_{ih}^{t+1} + \sum_{c \in C_S} x_{ih}^{t+3} + \sum_{i \in C_S} x_{ih}^{t+5} + (1 - \sum_{i \in C_S} x_{ih}^{t+7}) \quad (2.38)$$
$$\forall v \in CdM, \forall c \in C, \forall h \in H, t \in T_o$$

$$\sum_{i \in \{c,v\}} (x_{ih}^{t+2} + x_{ih}^{t+4} + x_{ih}^{t+6} + x_{ih}^{t+8}) \leq 3 + (1 - x_{vh}^{t}) + \sum_{i \in C_S} x_{ih}^{t+1} + \sum_{c \in C_S} x_{ih}^{t+3} + \sum_{i \in C_S} x_{ih}^{t+5} + \sum_{i \in C_S} x_{ih}^{t+7} \quad (2.39)$$
$$\forall v \in CdM, \forall c \in C, \forall h \in H, t \in T_o$$

The repetition constraints are reported in the following.

$$\sum_{c \in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + (1 - \sum_{i \in C_S} x_{ih}^{t+1}) + \sum_{c \in C_S} x_{ih}^{t+3} + (1 - \sum_{i \in C_S} x_{ih}^{t+5}) + (1 - x_{vh}^{t}) \quad (2.40)$$

$$\forall v \in CdM, \forall h \in H, t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega$$

$$\sum_{c \in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + (1 - \sum_{i \in C_S} x_{ih}^{t+1}) + (1 - \sum_{c \in C_S} x_{ih}^{t+3}) + (1 - x_{vh}^{t}) \tag{2.41}$$
$$\forall v \in CdM, \forall h \in H, t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega$$

$$\sum_{c \in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + \sum_{i \in C_S} x_{ih}^{t+1} + (1 - \sum_{c \in C_S} x_{ih}^{t+3}) + \sum_{i \in C_S} x_{ih}^{t+5} + (1 - x_{vh}^{t}) \tag{2.42}$$
$$\forall v \in CdM, \forall h \in H, t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega$$

$$\sum_{c \in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + \sum_{i \in C_S} x_{ih}^{t+1} + (1 - \sum_{c \in C_S} x_{ih}^{t+3}) + (1 - \sum_{i \in C_S} x_{ih}^{t+5}) + (1 - x_{vh}^{t}) \tag{2.43}$$
$$\forall v \in CdM, \forall h \in H, t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega$$

$$\sum_{c \in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + \sum_{i \in C_S} x_{ih}^{t+1} + \sum_{c \in C_S} x_{ih}^{t+3} + (1 - \sum_{i \in C_S} x_{ih}^{t+5}) + (1 - x_{vh}^{t}) \tag{2.44}$$
$$\forall v \in CdM, \forall h \in H, t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega$$

$$\sum_{c \in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + \sum_{i \in C_S} x_{ih}^{t+1} + \sum_{c \in C_S} x_{ih}^{t+3} + \sum_{i \in C_S} x_{ih}^{t+5} + (1 - \sum_{i \in C_S} x_{ih}^{t+7}) + (1 - x_{vh}^{t}) \tag{2.45}$$
$$\forall v \in CdM, \forall h \in H, t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega$$

$$\sum_{c \in CR} (x_{ch}^{t_1} + x_{ch}^{t_2} + x_{ch}^{t_3}) \leq 2 + + \sum_{i \in C_S} x_{ih}^{t+1} + \sum_{c \in C_S} x_{ih}^{t+3} + \sum_{i \in C_S} x_{ih}^{t+5} + \sum_{i \in C_S} x_{ih}^{t+7} + (1 - x_{vh}^{t}) \tag{2.46}$$
$$\forall v \in CdM, \forall h \in H, t \in T_o, \forall \{t_1, t_2, t_3\} \in \Omega$$

# Chapter 3

# Replication and Sequencing of Unreliable Jobs on Parallel Machines

Scheduling problems in which *machine breakdowns* may occur have received a good deal of attention in the literature [76]. Typically, breakdowns refer to the fact that processing resources (machines) may unexpectedly become unavailable, either because of a technical failure or because they are claimed by some higher-priority process. In this context, the problem is how to schedule the jobs in advance, taking into account the fact that breakdowns may occur. Many models in the literature assume that the machine(s) breakdown has a finite (either deterministic or stochastic) duration, so that after repairing it the schedule can be resumed (either in a preempt-repeat or a preempt-resume fashion [72]). In such scenarios, classical scheduling objectives are considered, such as total (weighted) flow time or total tardiness. Some studies address the optimality of the SPT rule with respect to expected total flow time [64] or in other post-disruption management situations [78].

Here we take a different view, namely we assume that breakdowns are *unrecoverable*, i.e., once a resource is withheld, it will not be returned before the end of the scheduling horizon. This scenario has been

particularly addressed in the computer science literature [70]. If there are chances that not all jobs will be carried out, rather than optimizing completion-time-related functions, it may be more sensible to focus on objectives depending on whether or not each job is successfully carried out. More precisely, this chapter addresses the following scenario. A set of *n jobs* is to be processed on identical parallel machines. Machines are subject to failures, i.e., when a machine breaks down, the remaining jobs scheduled on the machine (including the job currently being processed) cannot be performed. Job $J_j$ is characterized by a *success probability* $p_j$, i.e., the probability that the machine does not break down during the execution of the job. If job $J_j$ is successfully carried out, a *revenue* $r_j$ is gained. The problem is to allocate the jobs to the machines and sequence them on each machine so that the expected total revenue is maximized. Notice that the value $r_j$ may indeed represent the cost of outsourcing job $J_j$ to some external provider, so that maximizing expected revenue corresponds to minimizing expected outsourcing cost. This is also called *rejection cost* in the context of scheduling problems with rejection [77]. The problem of sequencing unreliable jobs to maximize expected total revenue has been addressed for the first time in [**?** ], where it was shown that the single-machine version of this problem can be solved by a simple sequencing rule. In [66] it is proved that for $m \geq 2$ the problem is NP-hard. It has been recently shown that, for $m = 2$, the list scheduling algorithm provides a 0.8535-approximate solution [68] and for the general $m$-machine case a 0.8531-approximate solution [69]. More specific results have been found for the special case in which machine breakdowns are exponentially distributed [67].

In this work, we investigate a common strategy which is adopted, when possible, to deal with unrecoverable interruptions, namely *job replication* [70]. The idea is that if identical copies of the same job are run on different machines, this increases the chance that *at least one copy* is successfully completed. An example is a complex scientific computation. We can easily replicate it on different machines, and we are interested that at least *one* copy completes successfully.

In this chapter, we address a scheduling problem on machines subject to unrecoverable breakdowns

characterized by the possibility of job replication. More precisely, there are *m identical copies* of each job, and each copy has to be scheduled on exactly one of the $m$ identical machines. On each of these machines, the corresponding copy of job $J_j$ has a probability $p_j$ to be successfully completed, and this probability is independent among the machines and also from the outcome of other jobs. We say that job $J_j$ is *carried out* if *at least one* of the $m$ copies of $J_j$ is successfully completed.

For the sake of simplicity, and when it does not create confusion, we write "job $J_j$ on machine $M_k$" to refer to "the copy of job $J_j$ allocated to machine $M_k$". The problem addressed in this chapter is the following.

*Expected Revenue Maximization with m Machines (ERMm) – Given n jobs* $\{1,2,\ldots,n\}$*, each having success probability* $p_j$ *and a revenue* $r_j$ *which is attained if* $J_j$ *is carried out, and m identical machines* $M_1, M_2, \ldots, M_m$*, find a sequence of the n copies of the jobs on each machine so that the expected revenue is maximized.*

Throughout the chapter, we let $P = \prod_{j=1}^{n} p_j$ and assume that $p_j < 1$ for all $j$ (jobs such that $p_j = 1$ can be obviously processed in the first positions of the schedule of any machine, with no consequences on the other jobs).

The following results are reported in this chapter. We first investigate the two-machine case. We show that *ERM2* is NP-complete, and propose a metaheuristic algorithm and a quadratic, mixed-integer formulation for its exact solution. The solution approaches are tested via extensive computational experiments, in which the quality of the solutions obtained is compared with an upper bound based on the solution of a 3-dimensional assignment problem. Then, the general case *ERMm* is investigated. For the special case of *ERMm* with two jobs, we establish that an optimal schedule can be derived in constant time, and we use this result to show that, in case of two jobs, the marginal increase in the expected revenue thanks to an additional machine decreases in the number of machines. For *ERMm* with $n$ jobs, we propose three heuristics, for two of which we prove a worst-case approximation bound, and a new, improved upper bound obtained by solving a variant of *ERMm*. We also propose a tabu search approach generalized to the case with $m$ machines.

Extensive computational experiments are performed to assess the quality of the introduced upper bound and heuristics, focusing on the problems *ERM2* and *ERM3*. Finally, some empirical results are also shown for a generic number of machines.

Part of this work has been published on Computers & Operations Research [65]. Another paper regarding the newest results is in preparation. The chapter is organized as follows. Section 3.1 presents some basic results which are needed by subsequent developments. Section 3.2 deals with *ERM2*. In particular, in Section 3.2.1 basic properties of *ERM2* are established, and its NP-hardness is proved. In Section 3.2.2 a quadratic integer formulation is proposed. A tabu search algorithm and an upper bounding scheme are proposed in Sections 3.2.3 and 3.2.4 respectively. Computational experiments for *ERM2* are discussed in Section 3.2.5. Concerning *ERMm*, the problem is addressed in Section 3.3. In particular, Section 3.3.1 focuses on *ERMm* with two jobs, while Section 3.3.2 focuses on the upper bounding schemes. Sections 3.3.3 to 3.3.6 present the three heuristic and the tabu search algorithm while Section 3.3.7 presents the computational experiments for the new bound and heuristics. Finally, some conclusions are drawn in Section 3.4.

## 3.1 Preliminaries

In what follows, we represent a *sequence* by a bijection $\sigma\colon N \to \{1,\ldots,n\}$, such that job $i \in N$ comes before job $j \in N$ in the sequence $\sigma$ if and only if $\sigma(i) < \sigma(j)$. Here and below, we do not distinguish between a job and its copies when it is unlikely to cause confusion. The probability that job $j$ is successful under the sequence $\sigma$ then equals

$$P_j(\sigma) = \prod_{i \in N\colon\, \sigma(i) \le \sigma(j)} p_i.$$

A *solution* for *ERM* will be represented by a multiset $S = \{\sigma_1, \ldots, \sigma_m\}$ of $m$ sequences (i.e., one for each machine). The corresponding expected revenue then equals

$$ER(S) = \sum_{j \in N} \left[ 1 - \prod_{\sigma \in S} (1 - P_j(\sigma)) \right] r_j. \tag{3.1}$$

Recall in this respect that job $j \in N$ is carried out, and thus leads to a revenue $r_j$, if and only if the job is successful on at least one machine. This latter event occurs with a probability equal to one minus the probability that the job is unsuccessful on all machines.

In the special case where there is only one machine (i.e., $m = 1$), a solution for *ERM* consists of a single sequence $\sigma$. Denoting the expected revenue of such a sequence by $ER(\sigma)$, Equation (3.1) simplifies to

$$ER(\sigma) = \sum_{j \in N} [1 - (1 - P_j(\sigma))] r_j = \sum_{j \in N} P_j(\sigma) r_j. \tag{3.2}$$

It is well-known that for this special case an optimal sequence can be found by following the so-called *Z-rule* [79; 66]. More specifically, define the *Z-ratio* of a job $j \in N$ as

$$Z_j = \frac{p_j r_j}{1 - p_j},$$

then it holds that a sequence is optimal if and only if it schedules the jobs in non-increasing order of their Z-ratio. The proof uses a standard pairwise-interchange argument, which for the sake of completeness is repeated below.

**Theorem 3.1.1 ([79; 66])** *Consider an* ERM *instance with a single machine and a sequence* $\sigma^\star$. *It then holds that* $ER(\sigma^\star) = \max_\sigma ER(\sigma)$ *if and only if* $Z_i > Z_j$ *implies* $\sigma^\star(i) < \sigma^\star(j)$ *for all* $i, j \in N$.

*Proof.* Consider a sequence $\sigma^\star$ that maximizes the expected revenue and, for arbitrary jobs $i, j \in N$ with

$\sigma^\star(j) = \sigma^\star(i) + 1$, let $\sigma^\star_{ij}$ be the sequence obtained from $\sigma^\star$ by interchanging jobs $i$ and $j$. Defining $Q$ such that $P_i(\sigma^\star) = p_i Q$, it then holds that $P_j(\sigma^\star) = P_i(\sigma^\star_{ij}) = p_i p_j Q$ and $P_j(\sigma^\star_{ij}) = p_j Q$. For all other jobs $k \in N \setminus \{i, j\}$, in turn, we have that $P_k(\sigma^\star) = P_k(\sigma^\star_{ij})$. Equation (3.2) then yields that

$$\begin{aligned} ER(\sigma^\star) - ER(\sigma^\star_{ij}) &= P_i(\sigma^\star) r_i + P_j(\sigma^\star) r_j - P_i(\sigma^\star_{ij}) r_i - P_j(\sigma^\star_{ij}) r_j \\ &= p_i Q r_i + p_i p_j Q r_j - p_i p_j Q r_i - p_j Q r_j \\ &= (1 - p_j) p_i Q r_i - (1 - p_i) p_j Q r_j \\ &= (1 - p_i)(1 - p_j)(Z_i - Z_j) Q, \end{aligned} \tag{3.3}$$

which is non-negative if and only if

$$Z_i = \frac{p_i r_i}{1 - p_i} \geq \frac{p_j r_j}{1 - p_j}.$$

This proves that scheduling the jobs in non-increasing order of their Z-ratio is a necessary condition for optimality. To establish sufficiency as well, consider an arbitrary sequence $\sigma^Z$ that schedules the jobs in non-increasing order of their Z-ratio. By the above argument, $\sigma^Z$ and $\sigma^\star$ can only differ in how they order jobs having the same Z-ratio. Equation (3.3) implies that interchanging jobs with the same Z-ratio does not affect the expected revenue, and thus $\sigma^Z$ must be optimal as well. □

## 3.2 *ERM* with two machines

Let us consider problem *ERM2*. We are given machines $M_1$ and $M_2$, and on each machine we must sequence a copy of the jobs $1, 2, \ldots, n$. We denote the two schedules as $(\sigma_1, \sigma_2)$ and the corresponding expected revenue as $ER(\sigma_1, \sigma_2)$. Also, we write $i \prec_1 j$ ($i \prec_2 j$) if $J_i$ precedes $J_j$ in $\sigma_1$ ($\sigma_2$). Given schedules $(\sigma_1, \sigma_2)$, let $P^1_j$ and $P^2_j$ denote the cumulative probabilities of job $J_j$ on the two machines respectively. At

least one copy of $J_j$ is successfully carried out with probability

$$P_j^1 + P_j^2 - P_j^1 P_j^2. \tag{3.4}$$

Hence, for given sequences $\sigma_1$ and $\sigma_2$ on the two machines, the expected revenue is given by

$$ER(\sigma_1, \sigma_2) = \sum_{j=1}^{n} r_j (P_j^1 + P_j^2 - P_j^1 P_j^2). \tag{3.5}$$

### 3.2.1 Properties and complexity

We now analyze some properties of the problem. First, we state an auxiliary result which will turn out to be useful in the analysis of *ERM2*. Suppose that a job sequence $\bar{\sigma}_1$ has been fixed on $M_1$, and let $\bar{P}_j^1$ denote the cumulative probability of job $J_j$ on $M_1$ in this sequence, i.e.

$$\bar{P}_j^1 = p_j \prod_{k:k \prec_1 j} p_k.$$

Moreover, given $\bar{\sigma}_1$, we define the *modified Z-ratio* of job $J_j$, denoted by $Z'_j$, as

$$Z'_j = Z_j (1 - \bar{P}_j^1). \tag{3.6}$$

Given a sequence $\bar{\sigma}_1$ on $M_1$, we consider the problem of finding the job sequence on $M_2$ so that the expected revenue is maximized. The following lemma holds:

**Lemma 3.2.1** *Given two machines and two copies of each job, if a job sequence $\bar{\sigma}_1$ is fixed on $M_1$, the expected revenue is maximized by sequencing the jobs on $M_2$ by nonincreasing values of the modified Z-ratios $Z'_j$.*

Proof. The proof uses an interchange argument. Consider a sequence $\sigma_2$ on $M_2$, and recall that $P_i^2$ is the cumulative probability of job $J_i$ in $\sigma_2$. Given a fixed sequence $\bar{\sigma}_1$ on $M_1$, assume that in $\sigma_2$ there are two *consecutive* jobs $J_j$ and $J_i$ such that $j \prec_2 i$ and $Z_i' > Z_j'$. Let $\sigma_2'$ be the sequence obtained swapping $J_i$ and $J_j$ in $\sigma_2$, and denote with $Q$ the product of the probabilities of the jobs preceding $J_i$ and $J_j$ on $M_2$. The expected revenue of $(\bar{\sigma}_1, \sigma_2')$ can be expressed as

$$ER(\bar{\sigma}_1, \sigma_2') = A + r_i(Qp_i + \bar{P}_i^1 - Qp_i\bar{P}_i^1) + r_j(Qp_ip_j + \bar{P}_j^1 - Qp_ip_j\bar{P}_j^1) + B$$

while

$$ER(\bar{\sigma}_1, \sigma_2) = A + r_j(Qp_j + \bar{P}_j^1 - Qp_j\bar{P}_j^1) + r_i(Qp_jp_i + \bar{P}_i^1 - Qp_jp_i\bar{P}_i^1) + B,$$

where $A$ and $B$ denote the contribution of jobs preceding and, respectively, following $J_i$ and $J_j$ on $M_2$ in the two schedules. Now, the difference

$$ER(\bar{\sigma}_1, \sigma_2') - ER(\bar{\sigma}_1, \sigma_2) = Q[r_ip_i - r_ip_i\bar{P}_i^1 + r_jp_ip_j - r_jp_ip_j\bar{P}_j^1 - (r_jp_j - r_jp_j\bar{P}_j^1 + r_ip_jp_i - r_ip_jp_i\bar{P}_i^1)]$$

is strictly positive if and only if

$$r_ip_i(1-\bar{P}_i^1)(1-p_j) > r_jp_j(1-\bar{P}_j^1)(1-p_i),$$

and hence

$$Z_i(1-\bar{P}_i^1) > Z_j(1-\bar{P}_j^1),$$

which holds since $Z_i' > Z_j'$. By repeatedly applying the above argument, the thesis follows. □

A consequence of Lemma 3.2.1 is the following.

**Lemma 3.2.2** *Consider an instance of ERM2 in which $Z_j = 1$ for all jobs $J_j$ (with $j = 1,\ldots,n$). Then any schedule in which the jobs are reversely sequenced on the two machines is optimal.*

Proof. Consider any schedule $\bar{\sigma}_1$ on $M_1$. If $J_i$ precedes $J_j$ on $M_1$, by definition of cumulative probability, $\bar{P}_i^1 > \bar{P}_j^1$. Since $Z_j = 1$, we have that, for all $j$, $r_j = (1-p_j)/p_j$ and $Z'_j = (1-\bar{P}_j^1)$, so from Lemma 3.2.1 the best schedule on $M_2$ is the one in which jobs are reversely sequenced. Recalling (3.4) and denoting $\prod_{j=l}^{u} p_j = 1$ if $l > u$, we obtain that the expected revenue is given by

$$\begin{aligned}&\sum_{i=1}^{n} \frac{1-p_i}{p_i}\left[\prod_{j=1}^{i} p_j + \prod_{j=i}^{n} p_j - \left(\prod_{j=1}^{i} p_j\right)\left(\prod_{j=i}^{n} p_j\right)\right] \\ &= \sum_{i=1}^{n}(1-p_i)\left(\prod_{j=1}^{i-1} p_j + \prod_{j=i+1}^{n} p_j - \prod_{j=1}^{n} p_j\right). \end{aligned} \tag{3.7}$$

Now observe that

$$\sum_{i=1}^{n}(1-p_i)\prod_{j=1}^{i-1} p_j = \sum_{i=1}^{n}\prod_{j=1}^{i-1} p_j - \sum_{i=1}^{n}\prod_{j=1}^{i} p_j = \sum_{i=0}^{n-1}\prod_{j=1}^{i} p_j - \sum_{i=1}^{n}\prod_{j=1}^{i} p_j = 1 - \prod_{j=1}^{n} p_j$$

and, similarly,

$$\sum_{i=1}^{n}(1-p_i)\prod_{j=i+1}^{n} p_j = \sum_{i=1}^{n}\prod_{j=i+1}^{n} p_j - \sum_{i=1}^{n}\prod_{j=i}^{n} p_j = \sum_{i=2}^{n+1}\prod_{j=i}^{n} p_j - \sum_{i=1}^{n}\prod_{j=i}^{n} p_j = 1 - \prod_{j=1}^{n} p_j.$$

Recalling that $P = \prod_{j=1}^{n} p_j$, we can then simplify (3.7) to

$$(1-P) + (1-P) - \left(n - \sum_{j=1}^{n} p_j\right)P = 2 - \left(n + 2 - \sum_{j=1}^{n} p_j\right)P. \tag{3.8}$$

Since, from (3.8), the expected revenue does not depend on the sequence on $M_1$, we can conclude that any schedule in which the jobs are reversely ordered on the two machines is optimal. □

We are now in the position of proving the main result of this section. In the proof we use the following

problem, which is strongly NP-hard [75].

*PRODUCT PARTITION: Given n integers $a_1, a_2, \ldots, a_n$, is there a partition $S_1, S_2$ such that*

$$\prod_{j \in S_1} a_j = \prod_{j \in S_2} a_j \ ?$$

**Theorem 3.2.3** *ERM2 is NP-hard.*

Proof. Given an instance of *PRODUCT PARTITION*, define an instance of *ERM2* containing $n+1$ jobs. The first $n$ jobs (called *regular*) correspond to the $n$ numbers of *PRODUCT PARTITION*, and are defined so that

$$p_i = \frac{1}{a_i} \quad r_i = \frac{1-p_i}{p_i}$$

while for the last job, called $J_K$ in the following, we let

$$p_K = \frac{\hat{P}-\varepsilon}{\hat{P}(1-\varepsilon)} \quad r_K = \frac{1-p_K}{p_K}(1-\varepsilon), \tag{3.9}$$

where $\hat{P} = \prod_{j=1}^{n} p_j$ is the product of all regular jobs' probabilities and $\varepsilon$ is any number such that $\varepsilon < \hat{P}$. As usual, $P$ denotes the product of all jobs' probabilities, i.e., $P = p_K \prod_{j=1}^{n} p_j$. Note that $Z_j = 1$ for all regular jobs $J_j$ (with $j = 1, \ldots, n$), while $Z_K = 1 - \varepsilon$.

First of all we want to establish the structure of an optimal solution of the instance of *ERM2*. Given an optimal solution $\sigma^*$ of *ERM2*, for notation simplicity let us number regular jobs according to their ordering on $M_1$ in $\sigma^*$. Let $S_1 = \{1, 2, \ldots, h\}$ and $S_2 = \{h+1, h+2, \ldots, n\}$ denote the job set preceding and, respectively, following $J_K$ on $M_1$, so that $J_h$ and $J_{h+1}$ denote the predecessor and, respectively, the successor of $J_K$ on $M_1$. (This definition also covers the cases in which $J_K$ is the first scheduled job on $M_1$, in which case we assume

$h = 0$, or it is the last scheduled job on $M_1$, in which case $h = n$.) Note that as a consequence of the above definitions, $P_K^1 = P_h^1 p_K$ ($P_K^1 = p_K$ if $h = 0$).

We observe that, since all regular jobs have $Z_j = 1$, the modified Z-ratio of job $J_j$ is $Z'_j = 1 - P_j^1$ and so, if $J_i$ precedes $J_j$ on $M_1$, then $Z'_j > Z'_i$. By Lemma 3.2.1, once the ordering of the regular jobs on $M_1$ is given, their ordering on $M_2$ is reversed with respect to $M_1$, and is unique.

We next claim that in $\sigma^*$, job $K$ immediately follows $J_{h+1}$ and immediately precedes $J_h$ on $M_2$. This is equivalent to showing that

$$1 - P_h^1 < Z'_K < 1 - P_{h+1}^1,$$

i.e., since $Z'_K = Z_K(1 - P_K^1)$ and $P_K^1 = P_h^1 p_K$,

$$\frac{1 - P_h^1}{1 - P_h^1 p_K} < Z_K < \frac{1 - P_h^1 p_K p_{h+1}}{1 - P_h^1 p_K}. \tag{3.10}$$

Recalling that $p_K = (\hat{P} - \varepsilon)/(\hat{P}(1 - \varepsilon))$, one has

$$\frac{1 - P_h^1}{1 - P_h^1 \frac{\hat{P} - \varepsilon}{\hat{P}(1-\varepsilon)}} = \frac{(1 - P_h^1)\hat{P}(1 - \varepsilon)}{\hat{P} - \hat{P}\varepsilon - P_h^1\hat{P} + P_h^1\varepsilon} < \frac{(1 - P_h^1)\hat{P}(1 - \varepsilon)}{\hat{P} - P_h^1\hat{P}} = 1 - \varepsilon,$$

where the inequality follows since $\hat{P} < P_h^1$. This proves the first inequality of (3.10). The second inequality of (3.10) simply follows from the observation that the right hand side is strictly greater than 1. We can therefore conclude that the structure of an optimal solution $\sigma^*$ is necessarily the following:

$M_1$: | $S_1$ $h$ | $K$ | $h+1$ $S_2$ |

$M_2$: | $S_2$ $h+1$ | $K$ | $h$ $S_1$ |

where the jobs in $S_1$ and $S_2$ are reversely ordered on the two machines. Notice that due to Lemma 3.2.2, the value of the solution only depends on the partition $(S_1, S_2)$, as sequencing within the two sets is immaterial (as long as they are reversely ordered on the two machines). For this reason we denote the value of such a

solution as $ER(S_1, S_2)$.

Let us therefore compute $ER(S_1, S_2)$. Recalling that $r_j = (1 - p_j)/p_j$ for all regular jobs and using an analogous derivation as the one leading to Equation (3.8), we obtain that the expected revenue of all jobs in $S_1$ is equal to

$$\begin{aligned}
&\sum_{i=1}^{h} \frac{1-p_i}{p_i} \left[ \prod_{j=1}^{i} p_j + p_K \prod_{j=i}^{n} p_j - \left( \prod_{j=1}^{i} p_j \right) \left( p_K \prod_{j=i}^{n} p_j \right) \right] \\
&= \sum_{i=1}^{h} (1-p_i) \left( \prod_{j=1}^{i-1} p_j + p_K \prod_{j=i+1}^{n} p_j - p_K \prod_{j=1}^{n} p_j \right) \\
&= \left( 1 - \prod_{j \in S_1} p_j \right) + \left( p_K \prod_{j \in S_2} p_j \right) \left( 1 - \prod_{j \in S_1} p_j \right) - \left( |S_1| - \sum_{j \in S_1} p_j \right) P \\
&= 1 - \prod_{j \in S_1} p_j + p_K \prod_{j \in S_2} p_j - \left( |S_1| + 1 - \sum_{j \in S_1} p_j \right) P.
\end{aligned}$$

where, in the last expression, we used $P = p_K \prod_{j \in S_1} p_j \prod_{j \in S_2} p_j$. By symmetry, we also find that the expected revenue of all jobs in $S_2$ is equal to

$$1 + p_K \prod_{j \in S_1} p_j - \prod_{j \in S_2} p_j - \left( |S_2| + 1 - \sum_{j \in S_2} p_j \right) P.$$

The expected revenue of job $K$, finally, equals

$$\begin{aligned}
&r_K \left[ p_K \prod_{j \in S_1} p_j + p_K \prod_{j \in S_2} p_j - \left( p_K \prod_{j \in S_1} p_j \right) \left( p_K \prod_{j \in S_2} p_j \right) \right] \\
&= r_K p_K \left( \prod_{j \in S_1} p_j + \prod_{j \in S_2} p_j - P \right).
\end{aligned}$$

Hence, if we denote

$$C = 2 - \left( n + 2 + r_K p_K - \sum_{j=1}^{n} p_j \right) P,$$

then we find that the total expected revenue of all jobs equals

$$ER(S_1,S_2) = C + (r_K p_K - (1-p_K)) \left( \prod_{j \in S_1} p_j + \prod_{j \in S_2} p_j \right). \tag{3.11}$$

Now the key observation is that due to the definition of $r_K$ in (3.9), one has that $r_K p_K - (1-p_K) < 0$, so indeed maximizing (3.11) is equivalent to minimizing:

$$\prod_{j \in S_1} p_j + \prod_{j \in S_2} p_j. \tag{3.12}$$

This expression is always greater or equal to $2\sqrt{\hat{P}}$, with the equality attained if $(S_1,S_2)$ is such that

$$\prod_{j \in S_1} p_j = \prod_{j \in S_2} p_j = \sqrt{\hat{P}},$$

and this in turn occurs if and only if

$$\frac{1}{\prod_{j \in S_1} a_j} = \frac{1}{\prod_{j \in S_2} a_j},$$

i.e., if and only if there exists a partition in the instance of *PRODUCT PARTITION*. We can therefore conclude that there is a partition $(S_1,S_2)$ of the regular jobs such that the value of the expected revenue is

$$C + 2\sqrt{\hat{P}}(R_K p_K - (1-p_K)),$$

if and only if the same partition also defines a partition of integers in *PRODUCT PARTITION* such that

$$\prod_{j \in S_1} a_j = \prod_{j \in S_2} a_j.$$

□

We note that while the reduction in Theorem 3.2.3 requires polynomial time, it implies the specification of very small positive numbers. In fact, the value $\hat{P}$ is the inverse of the *product* of all numbers of *PRODUCT PARTITION*, and such a product is not bounded by a polynomial in the maximum integer of *PRODUCT PARTITION*. As a consequence, even though *PRODUCT PARTITION* is indeed strongly NP-complete, Theorem 3.2.3 only proves the NP-hardness of *ERM2*, which is open for strong NP-hardness.

### 3.2.2 An integer formulation for *ERM2*

In this section we introduce a quadratic integer programming formulation of the problem *ERM2*. In the following, $P_j^1$ and $P_j^2$ are continuous variables, representing the cumulative probability of job $J_j$ on $M_1$ and $M_2$ respectively, while $s_{ij}^1$ and $s_{ij}^2$ are binary variables such that, for $k \in \{1,2\}$, one has that $s_{ij}^k = 1$ if $J_i$ precedes $J_j$ on machine $M_k$ and 0 otherwise. Problem *ERM2* can be formulated as follows.

$$\max \quad \sum_{j=1}^{n} r_j(P_j^1 + P_j^2 - P_j^1 P_j^2) \tag{3.13a}$$

$$\text{s.t.} \quad s_{ij}^1 + s_{ji}^1 = 1 \qquad \forall i,j\colon i \neq j \tag{3.13b}$$

$$s_{ij}^2 + s_{ji}^2 = 1 \qquad \forall i,j\colon i \neq j \tag{3.13c}$$

$$P_j^1 \leq p_j \qquad \forall\, j \tag{3.13d}$$

$$P_j^1 \leq p_j P_i^1 + 1 - s_{ij}^1 \qquad \forall i,j\colon i \neq j \tag{3.13e}$$

$$P_j^2 \leq p_j \qquad \forall\, j \tag{3.13f}$$

$$P_j^2 \leq p_j P_i^2 + 1 - s_{ij}^2 \qquad \forall i,j\colon i \neq j \tag{3.13g}$$

$$s_{ij}^1, s_{ij}^2 \in \{0,1\} \qquad \forall i,j \in N \tag{3.13h}$$

The objective function (3.13a) represents the expected revenue (from (3.5)). Contraints (3.13b) and (3.13c) define sequencing variables on the two machines, while constraints (3.13d) and (3.13e) define the

meaning of variables $P_j^1$. In fact, notice that if a job $J_i$ precedes $J_j$ on $M_1$ (i.e., $s_{ij}^1 = 1$), the variable $P_j^1$ cannot exceed the cumulative probability of $J_i$ multiplied by $p_j$. If $J_j$ is in first position on $M_1$, then $s_{ij}^1 = 0$ for all $J_i$ and (3.13d) implies that $P_j^1 = p_j$. Otherwise, the most binding among constraints (3.13e) determines the value of $P_j^1 = p_j P_i^1$, where $J_i$ is the job that immediately precedes $J_j$ in the schedule on $M_1$ (maximization implies that the corresponding constraint (3.13e) is active in an optimal solution). Constraints (3.13f) and (3.13g) have the same meaning for $M_2$. Notice that we do not need to impose transitivity constraints, in fact if $s_{ij}^1 = 1$ and $s_{jk}^1 = 1$, then constraints (3.13d) and (3.13e) imply that $s_{ki}^1 = 0$ (and similarly for $M_2$). This quadratic integer formulation has been employed in the computational experiments in Section 3.2.5.

### 3.2.3 A tabu search heuristic

In this section, a Tabu Search (TS) algorithm designed for solving *ERM2* is presented.

The algorithm exploits the result of Lemma 3.2.1, i.e., for a fixed schedule $\bar{\sigma}_1$ on machine $M_1$, the expected revenue is maximized by sequencing the jobs on $M_2$ by nonincreasing values of the modified Z-ratios $Z'_j$ (3.6). Hence, a solution of *ERM2* can be represented by only specifying the order in which the jobs are sequenced on machine $M_1$, since the optimal order on $M_2$ follows by Lemma 3.2.1.

The TS algorithm has two main phases. In the first phase an initial solution is generated. Then, a TS scheme is applied to iteratively improve the incumbent solution. In the first phase, the initial solution is found by scheduling jobs on $M_1$ by nonincreasing values of the Z-ratios (**??**). In the second phase, at each iteration $t$, starting from the current solution $(\sigma_1, \sigma_2)$, a solution neighborhood $N_t$ is generated, by performing all possible swaps of two jobs in $\sigma_1$. More precisely, if $\sigma'_1$ is a sequence on $M_1$ obtained by swapping two jobs in $\sigma_1$, and $\sigma'_2$ the corresponding sequence on $M_2$ by nonincreasing $Z'_j$, the new solution generated in the neighborhood is $(\sigma'_1, \sigma'_2)$. A tabu list of *jobs* is used to avoid considering solutions already visited. Every time a new current solution $(\sigma'_1, \sigma'_2)$ is selected from the neighborhood of $(\sigma_1, \sigma_2)$, the first of the two jobs

swapped on $M_1$ is inserted in the tabu list. All the jobs in the tabu list cannot be swapped, thus avoiding to return in previously visited solutions. When the tabu list is full and a new job has to be inserted, the job at the head of the list is removed from the tabu list and the new job is added to its tail.

At the end of iteration $t$, the best solution in $N_t$ is selected and chosen as the new current solution and the tabu list is updated. The whole process is repeated until a maximum number of iterations $It_{max}$ is reached.

A scheme of the algorithm is reported in Algorithm 2.

**Algorithm 2** Scheme of the Tabu Search algorithm.

Generate the initial solution $(\sigma_1,\sigma_2)^0$, set $t=0$, $(\sigma_1,\sigma_2)^{best} := (\sigma_1,\sigma_2)^0$.
**While** $t < It_{max}$ **do**
**begin**
  Generate the neighborhood $N_t$ of the current solution $(\sigma_1,\sigma_2)^t$ by swapping all job pairs in $\sigma_1$.
  Select a solution $(\sigma_1,\sigma_2)^*$ in $N_t$ that is not tabu and has maximal expected revenue.
  Set $(\sigma_1,\sigma_2)^t := (\sigma_1,\sigma_2)^*$.
    **If** $ER(\sigma_1,\sigma_2)^* > ER(\sigma_1,\sigma_2)^{best}$
      Set $(\sigma_1,\sigma_2)^{best} := (\sigma_1,\sigma_2)^*$.
  Update the tabu list, set t:=t+1.
**end**
**Return** the best solution $(\sigma_1,\sigma_2)^{best}$.

### 3.2.4 An upper bound for *ERM2*

In view of the intrinsic difficulty of the problem *ERM2*, in order to evaluate the quality of a solution, we propose an upper bounding scheme, based on the solution of an instance of the three-dimensional assignment problem (3AP):

$$\max \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk} \tag{3.14a}$$

$$\text{s.t.} \quad \sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \qquad \forall i = 1,\ldots,n \tag{3.14b}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \qquad \forall j = 1,\ldots,n \tag{3.14c}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} = 1 \qquad \forall k = 1,\ldots,n \tag{3.14d}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j,k \tag{3.14e}$$

where we refer to coefficients $c_{ijk}$ as *utilities*. Even though 3AP is itself a strongly NP-hard problem [73], commercial solvers are able to solve medium-sized instances in limited computation times, which is sufficient for our analysis.

Consider a schedule $(\sigma_1, \sigma_2)$, and any job $J_j$. Recalling (3.5), the contribution of $J_j$ to the expected revenue is $r_j(P_j^1 + P_j^2 - P_j^1 P_j^2)$. Now let $P(u)$ denote the product of the $u$ *largest* success probabilities. Suppose that a job $J_j$ is scheduled in position $u$ on $M_1$ and $v$ on $M_2$. Since the function $f(x,y) = x + y - xy$ is monotonically increasing for any $x \in [0,1]$ and $y \in [0,1]$, an upper bound on the contribution of $J_j$ to the expected revenue is given by

$$r_j(P(u) + P(v) - P(u)P(v)). \tag{3.15}$$

Actually, this expression can be slightly refined. In fact, if $p_j$ is *not* one of the $u$ ($v$) largest success probabilities, $P(u)$ ($P(v)$) in (3.15) can be replaced with $p_j P(u-1)$ ($p_j P(v-1)$).

Hence, given an instance of *ERM2*, we get an upper bound on its optimal value by solving an instance of 3AP in which each job is assigned a position on $M_1$ and a position on $M_2$. The utility of assigning a job $J_j$ to position $u$ on $M_1$ and $v$ on $M_2$ is

$$c_{juv} = r_j(\hat{P}_j(u) + \hat{P}_j(v) - \hat{P}_j(u)\hat{P}_j(v)),$$

where

$$\hat{P}_j(h) = \min\{p_j P(h-1), P(h)\}.$$

In fact, notice that if $p_j$ is one of the $h-1$ largest probabilities, then $P(h) \leq p_j P(h-1)$, otherwise $P(h) \geq p_j P(h-1)$.

In the special case that all jobs have an equal success probability $p$, the 3AP instance *exactly* yields the optimal solution. In fact, for every job $J_i$ and every schedule $(\sigma_1, \sigma_2)$ that schedules job $J_i$ in position $j$ on $M_1$ and $k$ on $M_2$, it holds that $P_i^1 = P(j) = p^j$ and $P_i^2 = P(k) = p^k$. Hence, we can obtain an optimal schedule by solving the corresponding 3AP instance with $c_{ijk} = r_i p^j p^k$ for every $i, j, k = 1, \ldots, n$. Observe that, for each $i, j, k = 1, \ldots, n$, the objective function coefficient $c_{ijk}$ is the product of three numbers that depend only on $i$, $j$ and $k$, respectively. The resulting structure is therefore actually a special case of the axial three-dimensional assignment problem (A3AP); see e.g., [71]. Although A3AP is NP-hard [71], the complexity of *ERM2* in the special case where all jobs have an equal success probability is still open.

### 3.2.5 Computational results

In this section, we present a computational campaign on random instances carried out to test the quadratic program, the tabu search and the 3AP-based upper bound.

We have carried out two different experiments. In the first experiment, both job probabilities and job revenues are generated from uniform distributions. The second experiment deals with two special cases, namely $(i)$ the jobs have identical revenues and $(ii)$ the time to failure follows an exponential distribution (identical for the two machines).

#### Uniform instances

In the first experiment, the instances have been generated by considering different numbers of jobs $n$ (i.e., $n \in \{10, 20, 30, 40, 50\}$) and three different intervals of probability $I_p$ ($I_p \in \{[0.9, 1], [0.5, 1], [0.1, 1]\}$), thus obtaining 15 possible sets. For each set we randomly generated 20 instances with job success probabilities uniformly distributed in $I_p$, while job revenues are integers uniformly distributed in $[10, 100]$. Hence, the experiments have been performed on 300 instances.

From a preliminary test campaign, it turned out that the TS algorithm (Section 3.2.3) is most effective by

setting the size of the tabu list to 20% of the number $n$ of jobs, and the number of iterations to $It_{max} = 100$. In fact, in most cases the quality of the solutions does not increase at all by increasing the maximum number of iterations over such limit. So, the TS algorithm was performed using these settings in all the experiments.

All the algorithms have been run on a 1.7 GHz CPU with 2 cores and 8 GB RAM. More in details, the TS algorithm has been implemented as a single-thread Python 3.7 program. The quadratic programming formulation (3.13a)–(3.13h) and the 3AP-based bound presented in Section 3.2.4 have been implemented and solved by the Gurobi 9.1 solver using all the available cores and the default tolerance parameters. We set the Gurobi parameter *NumericFocus* to 3, corresponding to maximum numerical accuracy. A time limit of 20 minutes was set for Gurobi, as preliminary experiments showed that beyond such limit the improvements in the optimality gap are extremely slow.

In Table 3.1, we report the results from our tests for each set of instances. Each row of the table refers to the average results over the 20 randomly generated instances of each set (one for each $n$ and $I_p$). In Column 3, the CPU time of the TS algorithm is reported. In Column 4, we report the average gap between the solutions obtained with the Tabu Search and the solutions obtained by Gurobi with the QP model (3.13a)–(3.13h). In particular, for each instance we compute this gap as $(Sol_{TS} - Sol_{QP})/Sol_{TS}$, where $Sol_{TS}$ and $Sol_{QP}$ are the values of the Tabu Search and the Gurobi solution, respectively. In Column 5 the CPU time needed to compute the upper bound $UB_{3AP}$ by solving the three-dimensional assignment problem is reported. Column 6 reports the average gap between $UB_{3AP}$ and the best upper bound found by Gurobi on the QP model, denoted as $UB_{QP}$. For each instance, this gap is computed as $(UB_{QP} - UB_{3AP})/UB_{QP}$ and so, when the value reported in Column 6 is negative, it means that $UB_{QP}$ is better than $UB_{3AP}$, on average. Finally, Column 10 reports the average gap between the Tabu Search solution and the best bound $UB_{Best}$ between $UB_{3AP}$ and $UB_{QP}$, computed as $(UB_{Best} - Sol_{TS})/UB_{Best} \times 100$ for each instance.

As the results show, the solutions obtained through the mathematical program turn out to be generally dominated by the Tabu Search solutions. In fact, Gurobi is not able to find optimal solutions within the time

| $n$ | $I_p$ | TS time | Avg gap TS-QP (%) | $UB_{3AP}$ time | Avg gap $UB_{QP}$-$UB_{3AP}$ (%) | Avg gap $UB_{Best}$-TS (%) |
|---|---|---|---|---|---|---|
| 10 | 0.1-1 | 0.24 | 0.00 | 0.22 | -14.78 | 0.46 |
| 20 | 0.1-1 | 3.76 | 0.15 | 1.05 | 20.44 | 16.93 |
| 30 | 0.1-1 | 11.80 | 0.79 | 1.60 | 32.36 | 22.10 |
| 40 | 0.1-1 | 27.00 | 2.46 | 4.44 | 35.38 | 26.75 |
| 50 | 0.1-1 | 55.92 | 3.48 | 13.95 | 44.21 | 26.42 |
| 10 | 0.5-1 | 0.25 | 0.04 | 0.94 | -7.95 | 4.98 |
| 20 | 0.5-1 | 3.31 | 0.36 | 1.02 | 17.92 | 21.75 |
| 30 | 0.5-1 | 11.47 | 2.12 | 1.52 | 30.21 | 26.57 |
| 40 | 0.5-1 | 26.46 | 4.60 | 3.98 | 35.14 | 28.46 |
| 50 | 0.5-1 | 55.08 | 6.48 | 9.78 | 39.98 | 31.19 |
| 10 | 0.9-1 | 0.24 | 0.00 | 0.97 | -0.46 | 0.85 |
| 20 | 0.9-1 | 3.64 | 0.08 | 1.07 | 2.57 | 4.13 |
| 30 | 0.9-1 | 10.84 | 1.18 | 1.51 | 5.24 | 7.82 |
| 40 | 0.9-1 | 25.81 | 2.83 | 4.81 | 8.37 | 10.53 |
| 50 | 0.9-1 | 54.68 | 5.22 | 10.66 | 9.97 | 14.52 |

Table 3.1: Computational results for the uniform instances.

limit on all the instances with $n > 10$, and, as shown in Column 4, the TS solution is better than the solution found by Gurobi and the gap increases as $n$ increases. For $n = 10$, Gurobi finds the optimal solution on 31 instances, and the optimality gap is below 0.1% on all other instances. On these 31 instances, the optimality is certified in an average time of 277 seconds. The TS algorithm is able to find the optimal solution on 25 of these 31 instances, while in the other instances the gap between $Sol_{TS}$ and $Sol_{QP}$ is close to zero. As for the upper bounds, the best bounds are consistently obtained by the 3AP approach except when $n = 10$, in which $UB_{QP}$ is better than $UB_{3AP}$ on average.

As the instance size increases, the optimality gap (reported in Column 7 of the table) gets worse. However, for $I_p = [0.9, 1]$, it does not exceed 15% even on larger instances, while for $I_p = [0.5, 1]$ or $I_p = [0.1, 1]$ a sharp increase of the optimality gaps can be observed already for $n = 20$. Somewhat surprisingly, there is not a big difference between instances with $I_p = [0.5, 1]$ and $I_p = [0.1, 1]$.

Finally, concerning computation times, we observe that both the TS and the 3AP do not require large CPU times, even for the largest instances.

From a practical viewpoint, one may wonder whether it pays off to invest in an additional machine such that jobs can be duplicated. In order to get some insight, for each instance of this experiment, we compare the value of expected revenue attained if there is a single machine ($m = 1$) with the expected revenue if there

| $n$ | $I_p = [0.1, 1]$ | $I_p = [0.5, 1]$ | $I_p = [0.9, 1]$ |
|---|---|---|---|
| 10 | 37.84 | 38.16 | 14.37 |
| 20 | 40.89 | 37.68 | 21.36 |
| 30 | 39.77 | 40.61 | 28.11 |
| 40 | 35.55 | 38.62 | 31.42 |
| 50 | 38.97 | 37.62 | 34.75 |

Table 3.2: Average percentage increase in the expected revenue for $m = 2$ instead of $m = 1$.

are two machines ($m = 2$). (Recall from Section 3.1 that when $m = 1$ the optimal schedule is attained by sequencing the jobs in nonincreasing order of their Z-ratio.)

Table 3.2 reports the average percentage increase in expected revenue thanks to the second machine. In these computations, we compare for each instance the best available (but not necessarily optimal) schedule for $m = 2$ with the optimal solution for $m = 1$. Table 3.2 suggests that the value of an additional machine is limited if $I_p = [0.9, 1]$ and $n$ is small, but for $I_p = [0.5, 1]$ and $I_p = [0.1, 1]$, or for increasing $n$, the average increase becomes more significant. One possible explanation is that, if the success probabilities are smaller or if there are more jobs, then it becomes more likely that not all jobs can be carried out on a single machine. Adding an extra machine therefore adds more value. This effect, however, appears to tail off when comparing $I_p = [0.5, 1]$ and $I_p = [0.1, 1]$. Indeed, between these settings, the additional value of a machine does not seem to differ substantially, and it also does not seem to depend on the number of jobs $n$.

Section 3.3.1 characterizes in more detail the marginal increase in the expected revenue thanks to an additional machine for $m > 2$, but when there are only two jobs.

### Special cases

The second experiment deals with two special cases. The first special case is when we want to maximize the expected *number* of achieved jobs, i.e., $r_j = 1$ for all $j$. In this case, each job is only characterized by its success probability. The second special case is when a job is characterized by a certain duration $t_j$, and the machine time to failure follows an exponential distribution. In this case, the success probability of a job is

given by $e^{-\lambda t_j}$, where $\lambda$ is the parameter of the exponential distribution. In this context, we let $r_j = t_j$ for all $j$, i.e., we want to maximize the expected amount of work done, as in [67; 70]. As before, we have run 20 instances for each value of $n = 10, 20, 30, 40, 50$. Processing times have been uniformly generated in [1,100], and $\lambda = 0.01$.

*Unit revenues* – The results on these instances (Table 3.3) do not seem substantially different from the uniform case. In fact, the overall average gap $(UB_{Best} - Sol_{TS})/UB_{Best} \times 100$ computed over all the instances is about 16% both in the uniform and unit-revenue instance sets. Similarly as before, a larger probability range negatively affects the gap between the best upper bound and the TS solution. The fact of having identical revenues does not result in any apparent structure for the optimal solution, and we conjecture that even this special case is hard.

*Exponential breakdowns* – Also in this case the computational times are not an issue for the Tabu Search and the 3AP upper bound, while Gurobi is never able to certify the optimal solution within the time limit. As in the other experiments, the Tabu Search performs better than Gurobi on the QP in all cases, with relatively small gaps even for larger instances (column 3 of Table 3.4). However, in this special case the upper bounds are less effective with respect to the other scenarios. The 3AP bound is better than the Gurobi bound except for $n = 10$. The gap between the TS solution and the best upper bound, however, is quite large, and it gets larger as the number of jobs increases, occasionally reaching 89% for an instance with 50 jobs. This may be due to the fact that the jobs with larger (smaller) revenue have a smaller (larger) success probability, which possibly makes it harder for Gurobi to discriminate among different schedules and therefore to find good upper bounds.

The increased gaps, as well as the fact that no apparent optimality property can be easily established, leads us to conjecture that even this special case is hard.

| $n$ | $I_p$ | TS time | Avg gap TS-QP (%) | $UB_{3AP}$ time | Avg gap $UB_{QP}$-$UB_{3AP}$ (%) | Avg gap $UB_{Best}$-TS (%) |
|---|---|---|---|---|---|---|
| 10 | 0.1-1 | 0.29 | 0.00 | 0.88 | -9.54 | 2.38 |
| 20 | 0.1-1 | 2.32 | 0.10 | 0.14 | 23.73 | 17.06 |
| 30 | 0.1-1 | 8.78 | 0.43 | 0.71 | 40.77 | 19.15 |
| 40 | 0.1-1 | 20.65 | 1.61 | 2.31 | 19.10 | 45.57 |
| 50 | 0.1-1 | 36.80 | 2.36 | 11.62 | 51.60 | 21.62 |
| 10 | 0.5-1 | 0.32 | 0.05 | 0.02 | -4.86 | 7.05 |
| 20 | 0.5-1 | 2.31 | 0.29 | 0.13 | 22.47 | 19.47 |
| 30 | 0.5-1 | 8.35 | 1.43 | 0.59 | 37.58 | 21.84 |
| 40 | 0.5-1 | 20.06 | 3.27 | 2.31 | 43.57 | 23.39 |
| 50 | 0.5-1 | 40.25 | 4.03 | 6.31 | 48.00 | 24.79 |
| 10 | 0.9-1 | 0.28 | 0.02 | 0.02 | -0.19 | 1.04 |
| 20 | 0.9-1 | 2.51 | 0.15 | 0.15 | 3.01 | 0.15 |
| 30 | 0.9-1 | 7.00 | 1.58 | 0.53 | 6.48 | 8.79 |
| 40 | 0.9-1 | 19.79 | 3.37 | 2.17 | 11.44 | 12.59 |
| 50 | 0.9-1 | 38.23 | 4.24 | 4.87 | 13.50 | 14.79 |

Table 3.3: Computational results for the instances with unit revenue.

| $n$ | TS time | Avg gap TS-QP (%) | $UB_{3AP}$ time | Avg gap $UB_{QP}$-$UB_{3AP}$ (%) | Avg gap $UB_{Best}$-TS (%) |
|---|---|---|---|---|---|
| 10 | 0.24 | 0.16 | 0.02 | -36.68 | 16.11 |
| 20 | 2.09 | 0.33 | 0.16 | 23.47 | 54.23 |
| 30 | 7.55 | 1.97 | 0.75 | 45.06 | 62.46 |
| 40 | 18.74 | 4.19 | 3.29 | 54.03 | 67.16 |
| 50 | 36.81 | 4.74 | 12.26 | 58.87 | 71.07 |

Table 3.4: Computational results for the instances with exponentially distributed failures.

## 3.3 *ERM* with m machines

In this section, the general *ERMm* problem is addressed. For simplicity, we will also write *ERM* to refer to *ERMm*. For this problem, we provide the following results.

- A new, improved upper bound obtained by solving a variant of *ERMm*. This upper bound is used to assess the quality of heuristic solutions;
- Three new heuristics, for two of which we prove a worst-case approximation bound, respectively of $H_m/m$ (where $H_m$ is the $m$-th harmonic number) and $1-(\frac{m-1}{m})^m$;
- An extensive computational campaign concerning the three heuristics as well as a tabu search approach for *ERMm*, showing that it is possible to quickly produce solutions with a small optimality gap even for large values of $m$.

This section is organized as follows. First, in Section 3.3.1 the problem with two jobs is analyzed. Then, we

focus on the general case with $n$ jobs. In Section 3.3.2, a procedure to compute an upper bound for *ERMm* is described. Sections 3.3.3, 3.3.4, and 3.3.5 deal with three different heuristic approaches, while Section 3.3.6 presents a metaheuristic approach. Computational experiments are described and discussed in Section 3.3.7.

### 3.3.1 *ERMm* with 2 jobs

In this section we focus on solving *ERMm* with $n = 2$ jobs efficiently. Besides showing that this special case can be solved in constant time, we wish to assess how the number of machines affects the expected revenue. In particular, we show that the marginal increase in the expected revenue thanks to an additional machine decreases as $m$ grows.

In the special case of two jobs, there are only two possible job sequences for each machine: $(1,2)$ or $(2,1)$. Let $x \in \{0,1,\ldots,m\}$ be the number of machines in which the jobs are scheduled in the order $(1,2)$, so that the number of machines with job sequence $(2,1)$ equals $m-x$. Observe that job $J_1$ is *not* carried out if and only if all $x$ machines scheduled according to the sequence $(1,2)$ fail on their first job and the other $m$ machines fail on either the first or second job. Hence, $J_1$ is carried out with probability

$$1-(1-p_1)^x(1-p_1p_2)^{m-x} \tag{3.16}$$

and, analogously, the probability that job $J_2$ is carried out equals

$$1-(1-p_2)^{m-x}(1-p_1p_2)^x. \tag{3.17}$$

The expected revenue of the resulting schedule therefore equals

$$r_1\left[1-(1-p_1)^x(1-p_1p_2)^{m-x}\right]+r_2\left[1-(1-p_2)^{m-x}(1-p_1p_2)^x\right]$$

$$= r_1 + r_2 - \left[ r_1(1-p_1)^x(1-p_1p_2)^{m-x} + r_2(1-p_2)^{m-x}(1-p_1p_2)^x \right].$$

Now define the function $f\colon \mathbb{R} \to \mathbb{R}$:

$$f(x) = r_1(1-p_1)^x(1-p_1p_2)^{m-x} + r_2(1-p_2)^{m-x}(1-p_1p_2)^x.$$

To maximize the expected revenue, we want to find a value $x \in \{0,1,\ldots,m\}$ for which $f(x)$ is minimal.

For every $x \in \mathbb{R}$, the derivative of $f$ in $x$ is given by

$$f'(x) = r_1 \ln\left(\frac{1-p_1}{1-p_1p_2}\right)\left(\frac{1-p_1}{1-p_1p_2}\right)^x (1-p_1p_2)^m + r_2 \ln\left(\frac{1-p_1p_2}{1-p_2}\right)\left(\frac{1-p_1p_2}{1-p_2}\right)^x (1-p_2)^m.$$

Denoting

$$\eta = \frac{r_2 \ln\left(\frac{1-p_1p_2}{1-p_2}\right)}{r_1 \ln\left(\frac{1-p_1p_2}{1-p_1}\right)},$$

we therefore obtain that $f'(x^\star) = 0$ for some $x^\star \in \mathbb{R}$ if and only if

$$\left(\frac{(1-p_1)(1-p_2)}{(1-p_1p_2)^2}\right)^{x^\star} = \eta \cdot \left(\frac{1-p_2}{1-p_1p_2}\right)^m,$$

which in turn is equivalent to

$$x^\star = \frac{m\left(\ln(1-p_2) - \ln(1-p_1p_2)\right) + \ln\eta}{\ln(1-p_1) + \ln(1-p_2) - 2\ln(1-p_1p_2)}. \tag{3.18}$$

Moreover, the second order derivative of $f$ is strictly positive since, for all $x \in \mathbb{R}$,

$$f''(x) = r_1 \left(\ln\left(\frac{1-p_1}{1-p_1p_2}\right)\right)^2 \left(\frac{1-p_1}{1-p_1p_2}\right)^x (1-p_1p_2)^m$$

$$+ r_2 \left( \ln \left( \frac{1-p_1p_2}{1-p_2} \right) \right)^2 \left( \frac{1-p_1p_2}{1-p_2} \right)^x (1-p_2)^m > 0.$$

The latter inequality shows that $f(x)$ is convex, and therefore, to determine a schedule that maximizes expected revenue, one has:

1. if $x^\star \geq m$, then it is optimal to schedule all $m$ machines in the order $(1,2)$;
2. if $x^\star \leq 0$, then it is optimal to schedule all $m$ machines in the order $(2,1)$;
3. if $x^\star \in (0,m)$, then it is optimal to either schedule $\lfloor x^\star \rfloor$ or $\lceil x^\star \rceil$ machines in the order $(1,2)$, and the remaining machines in the order $(2,1)$.

We now show that the marginal increase in the expected revenue thanks to an additional machine is decreasing in the number of machines. For every $m \geq 1$, let $x^\star(m)$ denote the value of $x^\star$ given in (3.18) and let $q(m)$ be the corresponding optimal number of machines scheduled in the order $(1,2)$ obtained by applying the method described above. Moreover, we denote by $Q_i(m)$ the resulting probability that job $J_i \in \{1,2\}$ is *not* carried out. Since $\ln(1-p_1) < \ln(1-p_1p_2)$ and $\ln(1-p_2) < \ln(1-p_1p_2)$, it follows from (3.18) that

$$x^\star(m+1) - x^\star(m) = \frac{\ln(1-p_2) - \ln(1-p_1p_2)}{\ln(1-p_1) + \ln(1-p_2) - 2\ln(1-p_1p_2)} \in (0,1)$$

for every $m$. As a consequence, $q(m+1)$ takes either value $q(m)$ or $q(m)+1$. Hence, $Q_1(m+1)$ takes value $(1-p_1)Q_1(m)$ if $q(m+1) = q(m)+1$ and value $(1-p_1p_2)Q_1(m)$ otherwise. Analogously, $Q_2(m+1)$ takes value $(1-p_1p_2)Q_2(m)$ if $q(m+1) = q(m)+1$ and value $(1-p_2)Q_2(m)$ otherwise. Now observe that the maximum expected revenue with $m$ machines equals $r_1(1-Q_1(m)) + r_2(1-Q_2(m))$. If we denote by $\Delta ER^\star(m)$ the marginal increase in the expected revenue thanks to an increase from $m$ to $m+1$ machines,

since $Q_i(m+1) < Q_i(m)$, we obtain that

$$\begin{aligned}
\Delta ER^\star(m) &= r_1(Q_1(m) - Q_1(m+1)) + r_2(Q_2(m) - Q_2(m+1)) \\
&= \max\{r_1 p_1 Q_1(m) + r_2 p_1 p_2 Q_2(m),\, r_1 p_1 p_2 Q_1(m) + r_2 p_2 Q_2(m)\} \\
&> \max\{r_1 p_1 Q_1(m+1) + r_2 p_1 p_2 Q_2(m+1),\, r_1 p_1 p_2 Q_1(m+1) + r_2 p_2 Q_2(m+1)\} \\
&= \Delta ER^\star(m+1).
\end{aligned}$$

Here, the inequality follows since $Q_1(m)$ and $Q_2(m)$ are decreasing in $m$. This establishes that the marginal increase in the expected revenue thanks to an additional machine is non-increasing as $m$ grows.

Although the analysis above focuses on the special case of only two jobs, we conjecture that also for a general number of jobs it holds that the marginal benefit of an additional machine is decreasing in the number of machines. If this conjecture is true, then it indicates that *ERMm* is mainly relevant for practical applications with a relatively small number of machines $m$. This motivates the relevance of *ERM2*, as studied in Section 3.2, and *ERM3*, for which results are presented in Section 3.3.7.

### 3.3.2 An upper bound for *ERMm*

Consider the relaxation of *ERMm* in which we are allowed to schedule the machines sequentially. More specifically, only when a certain machine gets blocked by a job failure, we need to decide how to schedule the jobs on the next machine. We call this variant the *sequential ERM*. A solution for the sequential *ERM* is represented by a policy that specifies, for every possible outcome of the jobs on the previous machines, which schedule to follow on the current machine. Observe that in case there is only a single machine, the sequential *ERM* is equivalent to the original *ERM*.

The optimal solution value for the sequential *ERM* provides an upper bound on the optimal value for the

original *ERM*. To see this, observe that, for every *ERM* solution, one possible policy for the sequential *ERM* would be to follow on each machine the corresponding schedule specified by the *ERM* solution, regardless of the outcome of the jobs on the previous machines. That is, even if a job is already carried out by a previous machine and therefore cannot generate any additional revenue, it is still included on the current machine according to the schedule described by the original *ERM* solution. This static policy leads to the same expected revenue for the sequential *ERM* as the original solution for the traditional *ERM*. By consequence, every schedule for the traditional *ERM* defines a policy for the sequential *ERM* with the same objective value, and therefore the maximum value for the sequential *ERM* is at least as large as the maximum value for the original *ERM*. We refer to the maximum objective value for the sequential *ERM* as the *sequential upper bound*.

In the following, we will show that for the sequential *ERM* it is an optimal policy to always schedule the jobs that have not yet been carried out in non-increasing order of their Z-ratio. Next, we will discuss a number of structural properties of the value of this upper bound that will be useful for the analysis in Section 3.3.3.

### Optimality of following the Z-ratio

Theorem 3.3.1 below states that for the sequential *ERM* it is optimal to follow the Z-ratio. More specifically, it is optimal to always schedule next the remaining jobs (i.e., the ones that have not yet been carried out on a previous machine) in non-increasing order of their Z-ratio. Here, we assume without loss of generality that the jobs are indexed in non-decreasing order of their Z-ratio. For every $i \in N$ and $k \in \{1,\dots,m\}$, the value $f(i,k)$ reflects the expected revenue when following the Z-ratio if the remaining jobs are $i,\dots,n$ and there are $k$ remaining machines. Equation (3.19) can then be interpreted as follows. Since job $i$ has the maximum Z-ratio among all remaining jobs $i,\dots,n$, it is scheduled next. If it succeeds, which occurs with probability $p_i$, then we obtain revenue $r_i$ and the value $f(i+1,k)$, reflecting that we still have $k$ machines

available to perform the remaining jobs $i+1,\ldots,n$. On the other hand, if job $i$ fails, which occurs with probability $(1-p_i)$, then we obtain value $f(i,k-1)$, reflecting that we lost one machine.

**Theorem 3.3.1** *For an arbitrary instance $(n,m,(r_i,p_i)_{i\in N})$ with $Z_1 \geq \ldots \geq Z_n$, the sequential upper bound equals $f(1,m)$ as determined by the recursion*

$$f(i,k) = p_i[r_i + f(i+1,k)] + (1-p_i)f(i,k-1) \tag{3.19}$$

*with boundary conditions $f(n+1,k) = 0$ and $f(i,0) = 0$ for all $i \in N$ and $k \in \{1,\ldots,m\}$.*

*Proof.* Define a value function $v\colon 2^N \times \{0,\ldots,m\} \to \mathbb{R}$ such that for every $S \subseteq N$ and $k \in \{0,\ldots,m\}$ the value $v(S,k)$ equals the maximum expected revenue that one can obtain from a set of remaining jobs $S$ and $k$ remaining machines. For every $S \subseteq N$ with $S \neq \emptyset$ and $k \in \{1,\ldots,m\}$, the value function satisfies the optimality equation

$$v(S,k) = \max_{j\in S} \left\{ p_j(r_j + v(S\setminus\{j\},k)) + (1-p_j)v(S,k-1) \right\}$$

with boundary conditions $v(S,0) = 0$ and $v(\emptyset,k) = 0$.

Now consider an arbitrary $S \subseteq N$ with $S \neq \emptyset$ and $k \in \{1,\ldots,m\}$. Moreover, let $i \in S$ be such that it has the maximum Z-ratio among all remaining jobs, i.e,

$$\frac{p_i r_i}{1-p_i} = \max_{j\in S} \frac{p_j r_j}{1-p_j}.$$

We then claim that

$$v(S,k) = p_i(r_i + v(S\setminus\{i\},k)) + (1-p_i)v(S,k-1), \tag{3.20}$$

such that following the Z-rule is indeed optimal. In particular, this would establish that $f(i,k) = v(\{i,\ldots,n\},k)$

for every $i \in N$ and $k \in \{1,\ldots,m\}$, and that the recursion as specified by Equation (3.19) is indeed correct. Hence, to complete the proof, it remains to be shown that Equation (3.20) holds.

We prove Equation (3.20) using induction on $k$ and $|S|$. For $k = 1$, we are in the single-machine case, and it is optimal to follow the Z-rule by Theorem 3.1.1. For $|S| = 1$, the claim also holds trivially, since there is only a single job that can be performed. Hence, suppose that $k > 1$ and $|S| > 1$, and, as the induction hypothesis, that our claim holds for the case of $k-1$ remaining machines or $|S|-1$ remaining jobs. Consider an arbitrary job $j \in S \setminus \{i\}$ and observe that, by definition of the value function and the induction hypothesis,

$$\begin{aligned}
&p_i(r_i + v(S \setminus \{i\}, k)) + (1 - p_i)v(S, k-1) \\
&\quad \geq p_i\left[r_i + p_j(r_j + v(S \setminus \{i,j\}, k)) + (1-p_j)v(S \setminus \{i\}, k-1)\right] \\
&\qquad + (1-p_i)\left[p_j(r_j + v(S \setminus \{j\}, k-1)) + (1-p_j)v(S, k-2)\right] \\
&\quad = p_i r_i + p_j r_j + p_i p_j v(S \setminus \{i,j\}, k) + p_i(1-p_j)v(S \setminus \{i\}, k-1) \\
&\qquad + (1-p_i)p_j v(S \setminus \{j\}, k-1) + (1-p_i)(1-p_j)v(S, k-2)
\end{aligned}$$

and

$$\begin{aligned}
&p_j(r_j + v(S \setminus \{j\}, k)) + (1 - p_j)v(S, k-1) \\
&\quad = p_j\left[r_j + p_i(r_i + v(S \setminus \{i,j\}, k)) + (1-p_i)v(S \setminus \{j\}, k-1)\right] \\
&\qquad + (1-p_j)\left[p_i(r_i + v(S \setminus \{i\}, k-1)) + (1-p_i)v(S, k-2)\right] \\
&\quad = p_i r_i + p_j r_j + p_i p_j v(S \setminus \{i,j\}, k) + p_i(1-p_j)v(S \setminus \{i\}, k-1) \\
&\qquad + (1-p_i)p_j v(S \setminus \{j\}, k-1) + (1-p_i)(1-p_j)v(S, k-2).
\end{aligned}$$

From this, we obtain that

$$\begin{aligned} & p_i(r_i + v(S \setminus \{i\}, k)) + (1 - p_i)v(S, k-1) \\ & \geq p_j(r_j + v(S \setminus \{j\}, k)) + (1 - p_j)v(S, k-1), \end{aligned}$$

showing that Equation (3.20) holds, and therefore completing the proof. □

## Structural properties of the upper bound

The next three lemmas provide insight into the structural properties of the optimal objective value for the sequential *ERM*. Throughout, we consider an arbitrary instance $(n, m, (r_i, p_i)_{i \in N})$ with $Z_1 \geq \ldots \geq Z_n$. It then follows from Theorem 3.3.1 that $f(i,k)$ equals the maximum expected revenue for the instance defined on job set $\{i, \ldots, n\}$ and with $k \geq 1$ remaining machines. Lemma 3.3.2 states that for an instance with a single machine the maximum Z-ratio upper bounds the maximum expected revenue. Lemma 3.3.3, in turn, states that the average expected revenue per machine is non-increasing in the number of machines. Lemma 3.3.4, finally, implies that before performing a job with maximum Z-ratio the expected revenue is not smaller than when the job has been carried out.

**Lemma 3.3.2** *$f(i,1) \leq Z_i$ for all $i \in N$.*

*Proof.* We show the result by induction on $i$. For $i = n$, we have $f(n,1) = p_n r_n \leq Z_n$ since $0 < p_n < 1$ and $r_n > 0$. Now consider arbitrary $i < n$ and assume that $f(i+1,1) \leq Z_{i+1}$. Equation (3.19) and our assumption that $Z_i \geq Z_{i+1}$ then yield

$$f(i,1) = p_i\left[r_i + f(i+1,1)\right] \leq p_i\left(r_i + Z_{i+1}\right) \leq p_i\left(r_i + Z_i\right) = p_i\left(r_i + \frac{p_i r_i}{1 - p_i}\right) = \frac{p_i r_i}{1 - p_i} = Z_i,$$

which completes the proof. □

**Lemma 3.3.3** $\frac{f(i,k)}{k} \leq \frac{f(i,\ell)}{\ell}$ *for all* $i \in N$ *and* $k \geq \ell \geq 1$.

*Proof.* We show the result using induction on $k$ and $i$. For $k = 1$ the result trivially holds. Now suppose that $k \geq 2$ and, as the first induction hypothesis, assume that

$$\frac{f(i,k-1)}{k-1} \leq \frac{f(i,\ell)}{\ell} \tag{3.21}$$

for all $i \in N$ and $\ell = 1,\ldots,k-1$. Equation (3.19) then yields that

$$\begin{aligned}(k-1)f(n,k) &= (k-1)\sum_{\ell=0}^{k-1}(1-p_n)^{\ell}p_n r_n = (k-1)f(n,k-1) + (k-1)(1-p_n)^{k-1}p_n r_n \\ &\leq (k-1)f(n,k-1) + \sum_{\ell=0}^{k-2}(1-p_n)^{\ell}p_n r_n = kf(n,k-1).\end{aligned}$$

The first induction hypothesis, i.e., Inequality (3.21), then implies that

$$\frac{f(n,k)}{k} \leq \frac{f(n,k-1)}{k-1} \leq \frac{f(n,\ell)}{\ell}$$

for all $\ell = 1,\ldots,k-1$. Since we also trivially have $f(n,k)/k \leq f(n,\ell)/\ell$ for $\ell = k$, we obtain that the result holds for $i = n$.

Now consider arbitrary $i \in N \setminus \{n\}$ and, as the second induction hypothesis, assume that

$$\frac{f(i+1,k)}{k} \leq \frac{f(i+1,\ell)}{\ell} \tag{3.22}$$

for all $\ell = 1,\ldots,k$. Equation (3.19) and this second induction hypothesis applied to $\ell = k-1$ then imply that

$$(k-1)f(i,k) = (k-1)p_i r_i + (k-1)p_i f(i+1,k) + (k-1)(1-p_i)f(i,k-1)$$

$$
\begin{aligned}
&\leq (k-1)p_i r_i + kp_i f(i+1,k-1) + (k-1)(1-p_i)f(i,k-1) \\
&= k\left[p_i(r_i + f(i+1,k-1)) + (1-p_i)f(i,k-2)\right] \\
&\qquad - p_i r_i + (k-1)(1-p_i)f(i,k-1) - k(1-p_i)f(i,k-2) \\
&= kf(i,k-1) - p_i r_i + (1-p_i)\left[(k-1)f(i,k-1) - kf(i,k-2)\right]. \qquad (3.23)
\end{aligned}
$$

Now observe that it follows from the first induction hypothesis and Lemma 3.3.2 that

$$
\begin{aligned}
(k-1)f(i,k-1) - kf(i,k-2) &\leq (k-1)f(i,k-1) - \frac{k(k-2)}{k-1} f(i,k-1) \\
&= \frac{1}{k-1} f(i,k-1) \leq f(i,1) \leq Z_i.
\end{aligned}
$$

Here, we use the first induction hypothesis twice: once in the first inequality (by applying Inequality (3.21) to $\ell = k-2$) and again in the penultimate inequality (by applying Inequality (3.21) to $\ell = 1$). Substituting this into Inequality (3.23) then yields $(k-1)f(i,k) \leq kf(i,k-1)$. Hence, by Inequality (3.21),

$$
\frac{f(i,k)}{k} \leq \frac{f(i,k-1)}{k-1} \leq \frac{f(i,\ell)}{\ell}
$$

for all $\ell = 1,\ldots,k-1$. Since we also trivially have $f(i,k)/k \leq f(i,\ell)/\ell$ for $\ell = k$, this completes the induction, and therefore the proof. □

**Lemma 3.3.4** *$f(i,k) \geq f(i+1,k)$ for all $i \in N \setminus \{n\}$ and $k \geq 0$.*

*Proof.* Consider an arbitrary $i \in N \setminus \{n\}$. We show the result by induction $k$. First observe that $f(i,0) = f(i+1,0) = 0$, so for $k = 0$ the result trivially holds. Now take arbitrary $k > 0$ and suppose that $f(i,k-1) \geq$

$f(i+1,k-1)$. It then follows from Equation (3.19) and $Z_i \geq Z_{i+1}$ that

$$\frac{f(i,k)-f(i+1,k)}{1-p_i} = Z_i + f(i,k-1) - f(i+1,k) \geq Z_{i+1} + f(i+1,k-1) - f(i+1,k)$$

Hence, to establish $f(i,k) \geq f(i+1,k)$, it suffices to show that

$$Z_{i+1} + f(i+1,k-1) - f(i+1,k) \geq 0. \tag{3.24}$$

For $k = 1$ this immediately follows from Lemma 3.3.2. Hence, we can assume that $k \geq 2$. Lemma 3.3.2 and Lemma 3.3.3 (first applied with $\ell = 1$ and next with $\ell = k-1$) then imply that

$$\begin{aligned} f(i+1,k) &= \frac{f(i+1,k)}{k} + (k-1)\frac{f(i+1,k)}{k} \\ &\leq f(i+1,1) + f(i+1,k-1) \leq Z_{i+1} + f(i+1,k-1). \end{aligned}$$

Inequality (3.24) therefore holds for all $k \geq 1$, which completes the proof. □

### 3.3.3 The Z-rule heuristic

**Algorithm 3** Z-rule heuristic (ZRH).

**input:** An *ERM* instance $(n, m, (r_i, p_i)_{i \in N})$

1 for all $j$, compute $Z_j = p_j r_j/(1-p_j)$ let $\sigma$ be the sequence that schedules the jobs $j \in N$ in non-increasing order of $Z_j$ let $S \leftarrow \{\sigma, \sigma, \ldots, \sigma\}$ ($m$ times) **return** $S$

In this section we analyze the worst-case performance of the heuristic that schedules the jobs in non-increasing order of their Z-ratio on all machines (Algorithm 3), hence requiring $O(n \log n)$ time. We refer to this heuristic as the *Z-rule heuristic*, abbreviated as ZRH. More specifically, for an arbitrary instance, we compare the objective value obtained by this heuristic with the sequential upper bound. As such, our analysis

not only gives insight into the Z-rule heuristic's performance, but also into the quality of our upper bound.

For every $m \geq 1$, let $H_m = \sum_{k=1}^{m} 1/k$ be the $m^{\text{th}}$ harmonic number, and denote $\alpha_m = H_m/m$. That is, $\alpha_m$ is the inverse harmonic mean of the integers $1,\ldots,m$. Table 3.5 lists the value of $\alpha_m$ for $m \in \{1,\ldots,20\}$. In the following, we will show that, for every instance with $m$ machines, the Z-rule heuristic leads to an expected revenue that is at least $\alpha_m$ times the sequential upper bound. We will also provide a family of instances showing that the derived bound between the heuristic and this sequential upper bound is asymptotically tight.

Table 3.5: Approximation guarantee $\alpha_m$ for *ERMm* with $m \in \{1,\ldots,20\}$.

| $m$ | $\alpha_m$ | $m$ | $\alpha_m$ | $m$ | $\alpha_m$ | $m$ | $\alpha_m$ |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 6 | 0.4083 | 11 | 0.2745 | 16 | 0.2113 |
| 2 | 0.75 | 7 | 0.3704 | 12 | 0.2586 | 17 | 0.2023 |
| 3 | 0.611 | 8 | 0.3397 | 13 | 0.2446 | 18 | 0.1942 |
| 4 | 0.5208 | 9 | 0.3143 | 14 | 0.2323 | 19 | 0.1867 |
| 5 | 0.4567 | 10 | 0.2929 | 15 | 0.2212 | 20 | 0.1799 |

Since $\alpha_m = H_m/m$ decreases in $m$, the obtained approximation guarantee is only constant for a fixed number of machines. Observe in this respect that the approximation guarantee compares the objective value of the heuristic with the sequential upper bound, and not the true maximum expected revenue for the original *ERM*. As such, our analysis does not show to which extent the deteriorating performance is caused by an increasing gap between the heuristic and the optimal solution on the one hand, or between the optimal solution and the upper bound on the other hand. Intuitively, however, one would expect that the quality of both the heuristic and the upper bound deteriorate as the number of machines increases. Indeed, if there are more machines, then following the same sequence on every machine, as in the Z-rule heuristic, seems more harmful, and a sequential approach in which one can schedule the next machine after knowing the previous one's outcome seems more beneficial.

### Worst-case performance guarantee

Consider an arbitrary instance $(n, m, (r_i, p_i)_{i \in N})$ with $Z_1 \geq \ldots \geq Z_n$. The expected revenue of scheduling the jobs in non-increasing order of their Z-ratio on all machines then equals $h(1,m)$ as determined by the recursion

$$h(i,k) = \sum_{\ell=1}^{k} \binom{k}{\ell} p_i^{\ell} (1-p_i)^{k-\ell} \left[ r_i + h(i+1, \ell) \right] \tag{3.25}$$

with boundary condition $h(n+1,k) = 0$, for all $i \in N$ and $k \in \{1, \ldots, m\}$. The interpretation of Equation (3.25) is as follows. For arbitrary $i \in N$ and $k \in \{1, \ldots, m\}$, consider the instance defined on job set $\{i, \ldots, n\}$ with $k$ available machines. Since job $i$ has maximal Z-ratio among all jobs $i, \ldots, n$, it is scheduled first on all available machines, and only those machines on which job $i$ succeeds remain available afterwards. The number of available machines to perform jobs $i+1, \ldots, n$ therefore follows a binomial distribution with $k$ trials and success probability $p_i$. Moreover, we receive the reward $r_i$ if and only if job $i$ is successful on at least one machine. Hence, if job $i$ is successful on exactly $\ell \in \{1, \ldots, k\}$ machines, which occurs with probability $\binom{k}{\ell} p_i^{\ell} (1-p_i)^{k-\ell}$, then we receive the reward $r_i$ as well as the expected reward $h(i+1, \ell)$ from the instance defined on job set $\{i+1, \ldots, n\}$ with $\ell$ available machines. If job $i$ fails on all $k$ machines, on the other hand, then the reward equals zero. Equation (3.25) therefore indeed correctly specifies the expected reward associated with the Z-rule heuristic.

In order to derive the worst-case performance guarantee, it is convenient to rewrite Equation (3.25). Consider an arbitrary $i \in N$ and $k \in \{2, \ldots, m\}$. Using the fact that $\binom{k}{\ell} = \binom{k-1}{\ell-1} + \binom{k-1}{\ell}$ for all $\ell = 1, \ldots, k-1$, where $\binom{k-1}{0} = 1$, we can rewrite Equation (3.25) as follows:

$$\begin{aligned} h(i,k) &= p_i^k \left[ r_i + h(i+1,k) \right] + \sum_{\ell=1}^{k-1} \left[ \binom{k-1}{\ell-1} + \binom{k-1}{\ell} \right] p_i^{\ell} (1-p_i)^{k-\ell} \left[ r_i + h(i+1, \ell) \right] \\ &= p_i^k \left[ r_i + h(i+1,k) \right] + p_i \sum_{\ell=1}^{k-1} \binom{k-1}{\ell-1} p_i^{\ell-1} (1-p_i)^{k-\ell} \left[ r_i + h(i+1, \ell) \right] \end{aligned}$$

$$+(1-p_i)\sum_{\ell=1}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}\left[r_i+h(i+1,\ell)\right]$$
$$=p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}\left[r_i+h(i+1,\ell+1)\right]+(1-p_i)h(i,k-1)$$
$$=p_ir_i+p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}h(i+1,\ell+1)+(1-p_i)h(i,k-1). \tag{3.26}$$

Here, the third equality results from merging the first two terms of the second equality (where the indices of the summation have been shifted down one unit), and from applying Equation (3.25) for $k-1$. The fourth equality, in turn, follows since

$$\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}=1, \tag{3.27}$$

because it is the total probability mass of a binomial distribution with $k-1$ trials and success probability $p_i$.

Theorem 3.3.6, which forms the main result of this section, states that for every instance with $m$ machines, the Z-rule heuristic leads to an expected revenue that is at least $\alpha_m = H_m/m$ times the sequential upper bound, where $H_m=\sum_{k=1}^{m}1/k$ is the $m^{\text{th}}$ harmonic number. To show this result, we use the following lemma.

**Lemma 3.3.5** *For every $k\in\mathbb{N}_0$ and $p\in(0,1)$ it holds that*

$$(1-p)\left(1-\frac{1}{k}\right)+p\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p^{\ell}(1-p)^{k-1-\ell}H_{\ell+1}\geq pH_k. \tag{3.28}$$

*Proof.* Consider an arbitrary $p\in(0,1)$ and $k\in\mathbb{N}_0$. Since one can easily verify that the inequality holds for $k=1$, we assume that $k\geq 2$. Using Equation (3.27), we can rewrite Inequality (3.28) as

$$p\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p^{\ell}(1-p)^{k-1-\ell}\left(H_k-H_{\ell+1}\right)\leq(1-p)\left(1-\frac{1}{k}\right),$$

which in turn is equivalent to

$$\sum_{\ell=0}^{k-2}\binom{k-1}{\ell}p^{\ell+1}(1-p)^{k-2-\ell}(H_k-H_{\ell+1})\leq\frac{k-1}{k}. \tag{3.29}$$

Now observe that for every $\ell=0,\ldots,k-2$

$$H_k-H_{\ell+1}=\sum_{u=\ell+2}^{k}\frac{1}{u}\leq\frac{k-1-\ell}{\ell+2}$$

and

$$\binom{k-1}{\ell}\cdot\frac{k-1-\ell}{\ell+2}=\frac{(k-1)!}{(\ell+1)!(k-2-\ell)!}\cdot\frac{\ell+1}{\ell+2}=\binom{k-1}{\ell+1}\cdot\frac{\ell+1}{\ell+2}.$$

Hence,

$$\begin{aligned}\sum_{\ell=0}^{k-2}\binom{k-1}{\ell}p^{\ell+1}(1-p)^{k-2-\ell}(H_k-H_{\ell+1})&\leq\sum_{\ell=0}^{k-2}\binom{k-1}{\ell}p^{\ell+1}(1-p)^{k-2-\ell}\frac{k-1-\ell}{\ell+2}\\&=\sum_{\ell=0}^{k-2}\binom{k-1}{\ell+1}p^{\ell+1}(1-p)^{k-2-\ell}\frac{\ell+1}{\ell+2}\\&=\sum_{\ell=1}^{k-1}\binom{k-1}{\ell}p^{\ell}(1-p)^{k-1-\ell}\frac{\ell}{\ell+1}\\&\leq\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p^{\ell}(1-p)^{k-1-\ell}\frac{k-1}{k}=\frac{k-1}{k},\end{aligned}$$

where the final inequality follows since $\ell k\leq(k-1)(\ell+1)$ for all $\ell=1,\ldots,k-1$, and where the final equality follows from Equation (3.27). This establishes Inequality (3.29) and therefore completes the proof. □

**Theorem 3.3.6** *Consider an arbitrary instance $(n,m,(r_i,p_i)_{i\in N})$ with $Z_1\geq\ldots\geq Z_n$. For every $i\in N$ and $k\in\{1,\ldots,m\}$ it then holds that*

$$\frac{h(i,k)}{f(i,k)}\geq\alpha_k=\frac{H_k}{k}=\frac{1}{k}\sum_{\ell=1}^{k}\frac{1}{\ell}. \tag{3.30}$$

*In particular,* $h(1,m) \geq \alpha_m f(1,m)$.

*Proof.* We establish Inequality (3.30) using induction on $i$ and $k$. For $k = 1$, it follows from expanding the recursions defined by Equations (3.19) and (3.25) that, for every $i \in N$,

$$h(i,1) = p_i[r_i + h(i+1,1)] = \sum_{j=i}^{n} \left( \prod_{u=i}^{j} p_u \right) r_j = f(i,1).$$

Since $\alpha_1 = 1$, this shows that Inequality (3.30) indeed holds if $k = 1$. Analogously, for $i = n$, it follow from expanding Equations (3.19) and (3.25) that, for every $k = 1, \ldots, m$,

$$h(n,k) = \sum_{\ell=1}^{k} \binom{k}{\ell} p_n^{\ell} (1-p_n)^{k-\ell} r_n = (1-(1-p_n)^k) r_n = \sum_{\ell=1}^{k} (1-p_n)^{k-\ell} p_n r_n = f(n,k).$$

Since $\alpha_k \leq 1$ for all $k \geq 1$, this shows that Inequality (3.30) indeed holds if $i = n$.

Now consider arbitrary $i \in N \setminus \{n\}$ and $k \in \{2, \ldots, m\}$. As the induction hypothesis, assume that $h(i,k-1) \geq \alpha_{k-1} f(i,k-1)$ and $h(i+1,\ell) \geq \alpha_\ell f(i+1,\ell)$ for all $\ell = 1, \ldots, k$. Equation (3.26) then yields

$$\begin{aligned} h(i,k) &\geq p_i r_i + p_i \sum_{\ell=0}^{k-1} \binom{k-1}{\ell} p_i^{\ell} (1-p_i)^{k-1-\ell} \alpha_{\ell+1} f(i+1,\ell+1) + (1-p_i)\alpha_{k-1} f(i,k-1) \\ &= \alpha_k p_i r_i + (1-\alpha_k) p_i r_i + p_i \sum_{\ell=0}^{k-1} \binom{k-1}{\ell} p_i^{\ell} (1-p_i)^{k-1-\ell} \alpha_{\ell+1} f(i+1,\ell+1) \\ &\quad + \alpha_k (1-p_i) f(i,k-1) + (\alpha_{k-1} - \alpha_k)(1-p_i) f(i,k-1). \end{aligned} \tag{3.31}$$

We now use Lemmas 3.3.2-3.3.4 to bound the right-hand side of the above inequality in terms of $f(i+1,k)/k$. In particular, Lemmas 3.3.2 and 3.3.3 imply that

$$p_i r_i = (1-p_i) Z_i \geq (1-p_i) Z_{i+1} \geq (1-p_i) f(i+1,1) \geq (1-p_i) \frac{f(i+1,k)}{k}. \tag{3.32}$$

Moreover, for every $\ell = 0, \ldots, k-1$, Lemma 3.3.3 and the fact that $\alpha_{\ell+1} = H_{\ell+1}/(\ell+1)$ yield

$$\alpha_{\ell+1} f(i+1, \ell+1) = H_{\ell+1} \frac{f(i+1, \ell+1)}{\ell+1} \geq H_{\ell+1} \frac{f(i+1,k)}{k}. \tag{3.33}$$

Next, observe that

$$\alpha_{k-1} - \alpha_k = \frac{1}{k-1} \left( H_{k-1} - \frac{k-1}{k} H_k \right) = \frac{1}{k-1} \left( \alpha_k + H_{k-1} - H_k \right) = \frac{1}{k-1} \left( \alpha_k - \frac{1}{k} \right)$$

and, by Lemmas 3.3.3 and 3.3.4, that

$$f(i,k-1) \geq f(i+1,k-1) \geq (k-1) \frac{f(i+1,k)}{k}.$$

Hence, we also have that

$$(\alpha_{k-1} - \alpha_k)(1-p_i) f(i,k-1) \geq \left( \alpha_k - \frac{1}{k} \right) (1-p_i) \frac{f(i+1,k)}{k}. \tag{3.34}$$

Substituting Inequalities (3.32)-(3.34) into Inequality (3.31) and rearranging terms then yields

$$\begin{aligned} h(i,k) \geq\ & \alpha_k p_i r_i + \alpha_k (1-p_i) f(i,k-1) \\ & + \left[ \left( 1 - \alpha_k + \alpha_k - \frac{1}{k} \right) (1-p_i) + p_i \sum_{\ell=0}^{k-1} \binom{k-1}{\ell} p_i^{\ell} (1-p_i)^{k-1-\ell} H_{\ell+1} \right] \frac{f(i+1,k)}{k}. \end{aligned}$$

On the other hand, it follows from Equation (3.19) that

$$\begin{aligned} \alpha_k f(i,k) &= \alpha_k p_i r_i + \alpha_k p_i f(i+1,k) + \alpha_k (1-p_i) f(i,k-1) \\ &= \alpha_k p_i r_i + \alpha_k (1-p_i) f(i,k-1) + p_i H_k \frac{f(i+1,k)}{k}. \end{aligned}$$

Hence, to establish that $h(i,k) \geq \alpha_k f(i,k)$, it suffices to show that

$$(1-p_i)\left(1-\frac{1}{k}\right)+p_i\sum_{\ell=0}^{k-1}\binom{k-1}{\ell}p_i^{\ell}(1-p_i)^{k-1-\ell}H_{\ell+1} \geq p_i H_k.$$

Lemma 3.3.5 states that this inequality indeed holds, which completes the proof. □

### Asymptotic tightness of the performance guarantee

Consider the family of *ERM* instances in which all jobs have unit reward and identical success probability. Observe that an instance $(n,m,(1,p)_{i\in N})$ from this family is completely specified by the parameters $n,m \in \mathbb{N}_0$ and $p \in (0,1)$. Let $F(n,p,m)$ and $H(n,p,m)$ be the corresponding objective value of the sequential upper bound and Z-rule heuristic, respectively.

Theorem 3.3.8 below shows that, for every $m \in \mathbb{N}_0$, the ratio between $H(n,p,m)$ and $F(n,p,m)$ tends to $\alpha_m$ as $n$ grows to infinity and $p$ tends to one. As such, it establishes that the worst-case performance guarantee derived above is asymptotically tight. Before proving this result, we first characterize the values of $F(n,p,m)$ and $H(n,p,m)$ as $n$ tends to infinity in the next lemma.

**Lemma 3.3.7** *For arbitrary $m \in \mathbb{N}_0$ and $p \in (0,1)$, define*

$$F(p,m) = m\frac{p}{1-p} \tag{3.35}$$

*and let $H(p,m)$ be as defined by the recursion, for $k = 1,\ldots,m$,*

$$H(p,k) = \frac{p^k}{1-p^k}+\sum_{\ell=1}^{k-1}\binom{k}{\ell}\frac{p^{\ell}(1-p)^{k-\ell}}{1-p^k}\left[1+H(p,\ell)\right]. \tag{3.36}$$

*It then holds that $\lim_{n\to\infty} F(n,p,m) = F(p,m)$ and $\lim_{n\to\infty} H(n,p,m) = H(p,m)$.*

*Proof.* Consider an arbitrary $p \in (0,1)$. We prove the result by induction on $m$. For $m = 1$, it immediately follows from Equations (3.19), (3.25), (3.35), and (3.36) that

$$\lim_{n\to\infty} F(n,p,1) = \lim_{n\to\infty} H(n,p,1) = \sum_{j=1}^{\infty} p^j = \frac{p}{1-p} = F(p,1) = H(p,1).$$

Hence, the result holds for $m = 1$. Now consider an arbitrary $m \geq 2$ and, as the induction hypothesis, assume that $\lim_{n\to\infty} F(n,p,k) = F(p,k)$ and $\lim_{n\to\infty} H(n,p,k) = H(p,k)$ for all $k = 1,\ldots,m-1$. For arbitrary $n \in \mathbb{N}_0$, let $P(n)$ and $Q(n)$ denote the probability that all $n$ jobs can be completed successfully in the solution obtained from the sequential upper bound and Z-rule heuristic, respectively. Since for given $m$ and $p$ the expected revenue for an instance with $n$ jobs only differs from the one with $n-1$ jobs if all $n$ jobs succeed, and since we consider unit revenues, it holds that $F(n,p,m) = F(n-1,p,m) + P(n)$ and $H(n,p,m) = H(n-1,p,m) + Q(n)$. Equation (3.19) then yields that

$$\begin{aligned} F(n,p,m) &= p\,[1 + F(n-1,p,m)] + (1-p)F(n,p,m-1) \\ &= p\,[1 + F(n,p,m) - P(n)] + (1-p)F(n,p,m-1) \end{aligned}$$

and therefore that

$$F(n,p,m) = \frac{p\,[1-P(n)]}{1-p} + F(n,p,m-1).$$

Since $p < 1$, we have $\lim_{n\to\infty} P(n) = 0$. Combined with the induction hypothesis, we obtain

$$\lim_{n\to\infty} F(n,p,m) = \frac{p\,[1-\lim_{n\to\infty} P(n)]}{1-p} + \lim_{n\to\infty} F(n,p,m-1) = m\frac{p}{1-p},$$

which establishes that $\lim_{n\to\infty} F(n,p,m) = F(p,m)$. Analogously, Equation (3.25) yields that

$$\begin{aligned} H(n,p,m) &= \sum_{\ell=1}^{m} \binom{m}{\ell} p^{\ell}(1-p)^{m-\ell}\,[1+H(n-1,p,\ell)] \\ &= p^m\,[1+H(n,p,m)-Q(n)] + \sum_{\ell=1}^{m-1} \binom{m}{\ell} p^{\ell}(1-p)^{m-\ell}\,[1+H(n-1,p,\ell)] \end{aligned}$$

and therefore that

$$H(n,p,m) = \frac{p^m\,[1-Q(n)]}{1-p^m} + \sum_{\ell=1}^{m-1} \binom{m}{\ell} \frac{p^{\ell}(1-p)^{m-\ell}}{1-p^m}\,[1+H(n-1,p,\ell)]\,.$$

Using $\lim_{n\to\infty} Q(n) = 0$ and the induction hypothesis, we thus obtain

$$\begin{aligned} \lim_{n\to\infty} H(n,p,m) &= \frac{p^m\,[1-\lim_{n\to\infty} Q(n)]}{1-p^m} + \sum_{\ell=1}^{m-1} \binom{m}{\ell} \frac{p^{\ell}(1-p)^{m-\ell}}{1-p^m} \Big[1+\lim_{n\to\infty} H(n-1,p,\ell)\Big] \\ &= \frac{p^m}{1-p^m} + \sum_{\ell=1}^{m-1} \binom{m}{\ell} \frac{p^{\ell}(1-p)^{m-\ell}}{1-p^m}\,[1+H(p,\ell)]\,. \end{aligned}$$

This establishes that also $\lim_{n\to\infty} H(n,p,m) = H(p,m)$, and therefore completes the proof. □

And now we are in the position to prove that the worst case result of Section 1 is asymptotically tight.

**Theorem 3.3.8** *For every $m \in \mathbb{N}_0$ it holds that*

$$\lim_{p\to 1} \lim_{n\to\infty} \frac{H(n,p,m)}{F(n,p,m)} = \alpha_m.$$

*Proof.* Given the result of Lemma 3.3.7, it suffices to show that

$$\lim_{p\to 1} \frac{H(p,m)}{F(p,m)} = \alpha_m \tag{3.37}$$

for every $m \in \mathbb{N}_0$. We show this using induction on $m$. Since $F(p,1) = H(p,1)$ for all $p \in (0,1)$, we have

$$\lim_{p\to 1} \frac{H(p,1)}{F(p,1)} = 1 = \alpha_1,$$

so Equation (3.37) holds for $m = 1$. Now consider arbitrary $m > 1$ and, as the induction hypothesis, assume that

$$\lim_{p\to 1} \frac{H(p,k)}{F(p,k)} = \alpha_k$$

for all $k = 1,\ldots,m-1$. Equations (3.35)-(3.36) imply that

$$\begin{aligned}\lim_{p\to 1} \frac{H(p,m)}{F(p,m)} &= \lim_{p\to 1} \left[ \frac{p^m}{(1-p^m)F(p,m)} + \sum_{\ell=1}^{m-1} \binom{m}{\ell} \frac{p^\ell (1-p)^{m-\ell}}{1-p^m} \left( \frac{1}{F(p,m)} + \frac{H(p,\ell)}{F(p,m)} \right) \right] \\ &= \lim_{p\to 1} \left[ \frac{p^m(1-p)}{mp(1-p^m)} + \frac{1}{m} \sum_{\ell=1}^{m-1} \binom{m}{\ell} \frac{p^\ell (1-p)^{m-\ell}}{1-p^m} \left( \frac{1-p}{p} + \ell \frac{H(p,\ell)}{F(p,\ell)} \right) \right], \end{aligned} \tag{3.38}$$

where in the second equality we use that $F(p,m) = mp/(1-p) = (m/\ell)F(p,\ell)$ for all $\ell = 1,\ldots,m$. Now observe that, using l'Hopital's rule,

$$\lim_{p\to 1} \frac{p^m(1-p)}{p(1-p^m)} = \lim_{p\to 1} \frac{mp^{m-1} - (m+1)p^m}{1-(m+1)p^m} = \frac{1}{m}$$

and, for every $\ell = 1,\ldots,m-1$,

$$\begin{aligned}\lim_{p\to 1} \frac{p^\ell (1-p)^{m-\ell}}{1-p^m} &= \lim_{p\to 1} \frac{\ell p^{\ell-1}(1-p)^{m-\ell} - (m-\ell)p^\ell (1-p)^{m-\ell-1}}{-mp^{m-1}} \\ &= \begin{cases} 0 & \text{for } \ell = 1,\ldots,m-2; \\ \frac{1}{m} & \text{for } \ell = m-1. \end{cases}\end{aligned}$$

Substituting this into Equation (3.38) and using the induction hypothesis, the definition of $\alpha_m$, and the fact

that $\lim_{p\to 1}(1-p)/p = 0$, we obtain that

$$\lim_{p\to 1}\frac{H(p,m)}{F(p,m)} = \frac{1}{m^2} + \frac{m-1}{m}\alpha_{m-1} = \alpha_m,$$

which completes the proof. □

### 3.3.4 Modified Z-rule heuristic

One potential weakness of the Z-rule heuristic is that it leads to a solution that follows the same sequence on each machine. Because of this, jobs occurring early on in the sequence have a relatively higher probability to be successful on multiple machines, whereas jobs occurring later on in the sequence have a relatively higher probability not to be successful on any machine. In this section, we consider a construction heuristic based on the so-called modified Z-ratio. This ratio was already introduced in Section 3.2.1 for the case of two machines and takes into account the probability that a job might already be successful on an other machine.

Given a multiset $S = \{\sigma_1, \ldots, \sigma_k\}$ of $k \in \{1, \ldots, m\}$ sequences and a job $j \in N$, we define the *modified Z-ratio* of $j$ given the partial solution $S$ as

$$Z_j(S) = Z_j \prod_{\sigma\in S}(1 - P_j(\sigma)) = \frac{p_j r_j}{1-p_j}\prod_{\sigma\in S}(1-P_j(\sigma)).$$

In words, the modified Z-ratio equals the expected revenue of performing job $j$ on a single machine times the probability that $j$ fails on all sequences in $S$, divided by job $j$'s failure probability. Observe that if $S = \emptyset$, then the modified Z-ratio reduces to the original Z-ratio. If $|S| = 1$, in turn, then it coincides with the modified Z-ratio as defined in [65] for *ERM2*, i.e., *ERM* constrained to two machines.

Before describing the heuristic, we first state the following result that motivates its description. It shows that if $m-1$ sequences are fixed, then for the remainder machine it is optimal to schedule the jobs in non-

increasing order of their modified Z-ratio. As such, it generalizes the corresponding result in [65] for *ERM2*. Here and below, we use the additive union, with symbol $\uplus$, to merge two multisets such that the multiplicities of their elements are summed up. For example, with $S = \{\sigma\}$ for some arbitrary sequence $\sigma$, we have that $S \uplus \{\sigma\} = \{\sigma, \sigma\}$.

**Theorem 3.3.9** *For a multiset $S = \{\sigma_1, \ldots, \sigma_{m-1}\}$ of $m-1$ sequences and a sequence $\sigma'$,*

$$ER(S \uplus \{\sigma'\}) = \max_{\sigma} ER(S \uplus \{\sigma\})$$

*if and only if $Z_i(S) > Z_j(S)$ implies $\sigma'(i) < \sigma'(j)$ for all jobs $i, j \in N$.*

*Proof.* Equation (3.1) yields that

$$\begin{aligned} ER(S \uplus \{\sigma'\}) &= \sum_{j \in N} \left[ 1 - \prod_{\sigma \in S \uplus \{\sigma'\}} (1 - P_j(\sigma)) \right] r_j \\ &= \sum_{j \in N} \left[ 1 \pm \prod_{\sigma \in S} (1 - P_j(\sigma)) - \left(1 - P_j(\sigma')\right) \prod_{\sigma \in S} (1 - P_j(\sigma)) \right] r_j \\ &= \sum_{j \in N} \left[ 1 - \prod_{\sigma \in S} (1 - P_j(\sigma)) \right] r_j + \sum_{j \in N} \left[ 1 - \left(1 - P_j(\sigma')\right) \right] r_j \prod_{\sigma \in S} (1 - P_j(\sigma)) \,. \end{aligned}$$

Denoting $C = \sum_{j \in N} \left[ 1 - \prod_{\sigma \in S} (1 - P_j(\sigma)) \right] r_j$ and $r'_j = r_j \prod_{\sigma \in S} (1 - P_j(\sigma))$ for each $j \in N$, we obtain that

$$ER(S \uplus \{\sigma'\}) = C + \sum_{j \in N} P_j(\sigma') r'_j.$$

Here, the quantity $\sum_{j \in N} P_j(\sigma') r'_j$ equals the expected revenue of following sequence $\sigma'$ for the corresponding *ERM* instance with a single machine and revenues $(r'_j)_{j \in N}$ instead of $(r_j)_{j \in N}$. Since the constant $C$ does not depend on $\sigma'$, Theorem 3.1.1 yields that, for fixed $S$, the expected revenue $ER(S \uplus \{\sigma'\})$ is maximized by $\sigma'$

if and only if the jobs $j \in N$ are scheduled in non-increasing order of the ratio

$$\frac{p_j r'_j}{1-p_j} = \frac{p_j r_j}{1-p_j} \prod_{\sigma \in S} (1 - P_j(\sigma)) = Z_j(S),$$

which establishes the result. □

**Algorithm 4** Modified Z-rule heuristic (MZRH).

**input:** An *ERM* instance $(n, m, (r_i, p_i)_{i \in N})$

initialize $S \leftarrow \emptyset$ **while** $|S| < m$, **do**

  let $\sigma$ be a sequence that schedules the jobs $j \in N$ in non-increasing order of $Z_j(S)$ let $S \leftarrow S \uplus \{\sigma\}$

**return** $S$

Based on the above result, we propose the *modified Z-rule heuristic* (MZRH), summarized in Algorithm 4. The modified Z-rule heuristic builds a feasible solution such that, in each iteration, the next machine schedules the jobs in non-increasing order of their modified Z-ratio, given the partial solution formed by the previous machines' sequences. Notice that computing $P_j(S)$, given the values $P_j(S - \{\sigma\})$ and $\sigma$ can be done in constant time (as $P_j(S) = P_j(S - \{\sigma\}) + P_j(\sigma) - P_j(S - \{\sigma\})P_j(\sigma)$). Since there are $m$ iterations and scheduling the jobs in non-increasing order of their modified Z-ratio takes time $O(n \log(n))$, the modified Z-rule heuristic runs in time $O(mn \log(n))$.

In the following, an approximation result for the Modified Z-rule heuristic is presented. More precisely, we show that this heuristic produces a solution whose value is at least $1 - (\frac{m-1}{m})^m$ times the optimal value. Such a result follows from an approximation study of [74] establishing an approximation bound of a "greedy" heuristic for the maximization of submodular set functions.

**Definition 3.3.1** *Given a finite set M, a set function* $f \colon 2^M \to \mathbb{R}$ *is submodular if for every* $\rho \in M$ *and* $S \subseteq T \subseteq M$ *it holds that* $f(S \cup \{\rho\}) - f(S) \geq f(T \cup \{\rho\}) - f(T)$.

In words, for a submodular function $f$, the marginal benefit of including an additional element $\rho$ is non-increasing in the set of already included elements.

Given a finite set $M$, a submodular function $f\colon 2^M \to \mathbb{R}$, and an integer $m \geq 1$, consider the problem $\max\{f(S)\colon |S| \leq m, S \subseteq M\}$ and the greedy algorithm that, starting from the empty set $S = \emptyset$, at each step adds to $S$ an element $\sigma \in M \setminus S$ maximizing $f(S \cup \{\sigma\}) - f(S)$. [74] proved that such a greedy algorithm is $1 - (\frac{m-1}{m})^m$ approximate.

Below, we use the result of [74] to argue that the modified Z-rule heuristic has the same approximation guarantee. To do so, we first show that the expected revenue is submodular in the included sequences and, next, argue that the modified Z-rule heuristic coincides with the greedy algorithm described by [74]. Since we represent a solution by a multiset instead of a set, however, we first need to introduce some additional notation.

Let $\mathscr{P}$ denote the set of all permutations (i.e., sequences) of the jobs in $N$, and let $M = \mathscr{P} \times \{1, \ldots, m\}$ be the set containing $m$ copies of each permutation. Hence, $M$ contains $n! \times m$ job permutations. For an arbitrary subset $S \subseteq M$, denote by $\bar{S}$ the multiset obtained from $S$ by including each permutation $\sigma \in \mathscr{P}$ with the number of times that it appears in $S$. Additionally, define the set function $f\colon 2^M \to \mathbb{R}$ with $f(S) = ER(\bar{S})$ for every $S \subseteq M$. It then follows that every subset $S \subseteq M$ with $|S| = m$ corresponds to a unique solution $\bar{S}$ of our *ERM* problem, and that $f(S)$ equals the expected revenue of this solution. The following lemma shows that $f$ is submodular, implying that the additional expected revenue thanks to scheduling an additional machine according to a given permutation is non-increasing in the multiset of sequences scheduled on other machines.

**Lemma 3.3.10** *The set function $f\colon 2^M \to \mathbb{R}$ with $f(S) = ER(\bar{S})$ for every $S \subseteq M$ is submodular.*

*Proof.* Consider arbitrary $S \subseteq T \subseteq M$ and $(\rho,k) \in M$. From Equation (3.1),

$$\begin{aligned} f(S \cup \{(\rho,k)\}) = ER(\bar{S} \uplus \{\rho\}) &= \sum_{j \in N} \left[ 1 - (1 - P_j(\rho)) \prod_{\sigma \in \bar{S}} (1 - P_j(\sigma)) \right] r_j \\ &= ER(\bar{S}) + P_j(\rho) \sum_{j \in N} \left[ \prod_{\sigma \in \bar{S}} (1 - P_j(\sigma)) \right] r_j \end{aligned}$$

and

$$f(T \cup \{(\rho,k)\}) = ER(\bar{T}) + P_j(\rho) \sum_{j \in N} \left[ \prod_{\sigma \in \bar{T}} (1 - P_j(\sigma)) \right] r_j.$$

Since $S \subseteq T$ and $0 \leq 1 - P_j(\sigma) \leq 1$ for every $\sigma \in M$ and $j \in N$, it follows that

$$\begin{aligned} f(S \cup \{(\rho,k)\}) - f(S) &= ER(\bar{S} \uplus \{\rho\}) - ER(\bar{S}) \\ &= P_j(\rho) \sum_{j \in N} \left[ \prod_{\sigma \in \bar{S}} (1 - P_j(\sigma)) \right] r_j \\ &\geq P_j(\rho) \sum_{j \in N} \left[ \prod_{\sigma \in \bar{T}} (1 - P_j(\sigma)) \right] r_j \\ &= ER(\bar{T} \uplus \{\rho\}) - ER(\bar{T}) = f(T \cup \{(\rho,k)\}) - f(T). \end{aligned}$$

By Definition 3.3.1, this shows that $f$ is submodular. □

Now observe that, with the definition of $M$ and $f$ as described above, our *ERM* problem corresponds exactly to the problem $\max\{f(S) \colon |S| \leq m, S \subseteq M\}$. Moreover, it follows from Theorem 3.3.9 that, starting from the empty set $S = \emptyset$, the modified Z-rule heuristic adds at each iteration $t = 1, \ldots, m$ a permutation $\sigma \in \mathscr{P}$ such that

$$f(S \cup \{(\sigma,t)\}) - f(S) = \max_{(\rho,k) \in M \setminus S} \{f(S \cup \{(\rho,k)\}) - f(S)\}.$$

As such, our modified Z-rule heuristic is a special case of the greedy algorithm described by [74]. Moreover,

by Lemma 3.3.10, the function $f$ is submodular. If we let $S_H$ be the solution produced by the Modified Z-rule heuristic, and $S^*$ an optimal solution, we thus obtain:

**Theorem 3.3.11 (Nemhauser et al., 1978)** *For every $m \in \mathbb{N}_0$ it holds that*

$$\frac{ER(S_H)}{ER(S^*)} \geq 1 - \left(\frac{m-1}{m}\right)^m \geq 1 - \frac{1}{e}.$$

### 3.3.5 Mutual-best-reply heuristic

In this section we introduce a third heuristic algorithm, called the *mutual-best-reply heuristic* (MBRH) and summarized in Algorithm 5. Here, for a solution $S$ and a sequence $\sigma \in S$, we denote by $S - \{\sigma\}$ the multiset obtained by decreasing the multiplicity of $\sigma$ in $S$ by one. We also define

$$W(S) = \{\sigma \in S \colon \exists i, j \in N \text{ with } \sigma(i) < \sigma(j) \text{ and } Z_i(S - \{\sigma\}) < Z_j(S - \{\sigma\})\}$$

as the set of sequences $\sigma \in S$ that do not schedule the jobs $j \in N$ in non-increasing order of their modified Z-ratios with the partial solution $S - \{\sigma\}$. The mutual-best-reply heuristic then iteratively checks whether there is a machine whose sequence disagrees with the modified Z-rule given the partial solution formed by all other machines. If so, then the machine is rescheduled according to the modified Z-rule and we proceed to the next iteration. Hence, when the procedure terminates, it follows from Theorem 3.3.9 that every machine sequences the jobs optimally if we consider the other machines' sequences to be fixed. That is, the sequences form a mutual best reply to each other.

It follows from Theorem 3.3.9 that whenever a sequence disagrees with the modified Z-rule, then rescheduling leads to a strict increase in the expected revenue. Equation (3.3) in the proof of Theorem 3.1.1

**Algorithm 5** Mutual-best-reply heuristic (MBRH).

**input:** An *ERM* instance $(n, m, (r_i, p_i)_{i \in N})$ and an initial solution $S = \{\sigma_1, \ldots, \sigma_m\}$

**while** there exists a $\sigma \in W(S)$, **do**

  let $S \leftarrow S - \{\sigma\}$ let $\sigma'$ be a sequence that schedules the jobs $j \in N$ in non-increasing order of $Z_j(S)$ let $S \leftarrow S \uplus \{\sigma'\}$

**return** $S$

implies that this increase is lower bounded by

$$\gamma = \min_{i,j \in N:\, Z_i > Z_j} \left\{ (1-p_i)(1-p_j)(Z_i - Z_j) \prod_{k \in N \setminus \{i,j\}} p_k \right\}.$$

Since the expected revenue of an arbitrary solution lies in the interval $\left[0, \sum_{j \in N} r_j\right]$, the mutual-best-reply heuristic thus terminates in at most $\sum_{j \in N} r_j / \gamma$ iterations. This is finite, but not polynomial (or even pseudo-polynomial) in the input size. Establishing the exact computational complexity of the mutual-best-reply heuristic still forms an open problem.

### 3.3.6 A tabu search heuristic

In this section, a Tabu Search (TS) algorithm for *ERM* is presented. The overall structure is summarized in Algorithm 6. Starting from an initial solution $\{\sigma_1, \ldots, \sigma_m\}$, in each iteration $t$ of the TS scheme we select the best non-tabu neighbor solution $\{\hat{\sigma}_1, \ldots, \hat{\sigma}_m\}$. The solution neighborhood is constructed according to the method explained below. If this best neighbor improves the best known solution $S_{\text{best}}$, we update this latter solution. If the best neighbor is worse than the solution with which we started the iteration, i.e., no improving solution is found in the neighborhood, then we make the move that led to our neighbor solution tabu for the next $\bar{\tau}$ iterations in order to avoid cycling. We refer to the parameter $\bar{\tau}$ as the *tabu tenure*. Finally, we proceed to the next iteration with $\{\hat{\sigma}_1, \ldots, \hat{\sigma}_m\}$ as our new current solution. The algorithm terminates when the maximum number of iterations $T$ is reached.

**Algorithm 6** Tabu Search for *ERM*.

**input:** an *ERM* instance $(n, m, (r_i, p_i)_{i \in N})$ and an initial solution $\{\sigma_1, \ldots, \sigma_m\}$

initialize $S_{\text{best}} \leftarrow \{\sigma_1, \ldots, \sigma_m\}$ and $t \leftarrow 0$ **for** $k = 1, \ldots, m$ **do**
    **for** $i, j \in N$ with $i < j$ **do**
        $\tau(i, j, k) \leftarrow 0$ ; `// Initialize tabu list`
**while** $t < T$, **do**
    let $B \leftarrow 0$ **for** $k = 1, \ldots, m$ **do** `// Loop over machines`
        **for** $i, j \in N$ with $i < j$ and $\tau(i, j, k) \geq t$ **do** `// Loop over non-tabu swaps`
            swap $i$ and $j$ in $\sigma_k$ to obtain a new sequence $\sigma'_k$ initialize $S' \leftarrow \{\sigma'\}$ and $R \leftarrow \{1, \ldots, m\} \setminus \{k\}$ **while** $R \neq \emptyset$ **do** `// Schedule remaining machines`
                pick arbitrary $\ell \in R$ and let $R \leftarrow R \setminus \{\ell\}$ let $\sigma'_\ell$ be a sequence with jobs $u \in N$ in non-increasing order of $Z_u(S')$ let $S' \leftarrow S' \uplus \{\sigma'_\ell\}$
            **if** $ER(S') > B$ **then** `// Store best non-tabu neighbor`
                $B \leftarrow ER(S')$ $(\hat{i}, \hat{j}, \hat{k}) \leftarrow (i, j, k)$ $(\hat{\sigma}_1, \ldots, \hat{\sigma}_m) \leftarrow (\sigma'_1, \ldots, \sigma'_m)$
    **if** $B > ER(S_{best})$ **then** `// Store overall best solution`
        $S_{\text{best}} \leftarrow \{\hat{\sigma}_1, \ldots, \hat{\sigma}_m\}$
    **else if** $B < ER(\{\sigma_1, \ldots, \sigma_m\})$ **then** `// Make non-improving move tabu`
        $\tau(\hat{i}, \hat{j}, \hat{k}) \leftarrow t + \bar{\tau}$
    let $(\sigma_1, \ldots, \sigma_m) \leftarrow (\hat{\sigma}_1, \ldots, \hat{\sigma}_m)$ let $t \leftarrow t + 1$
**return** $S_{\text{best}}$

Given a tuple $(\sigma_1, \ldots, \sigma_m)$, we consider for every two jobs $i, j \in N$ with $i < j$ and for every machine $k \in \{1, \ldots, m\}$ a *move* $(i, j, k)$ that generates a *neighbor* $(\sigma'_1, \ldots, \sigma'_m)$ as follows. First, we swap jobs $i$ and $j$ in sequence $\sigma_k$, i.e., the sequence followed on machine $k$, to obtain a new sequence $\sigma'_k$. Next, we run through the remaining machines in an arbitrary order and, at each step, the next machine $\ell$ is scheduled according to a sequence $\sigma'_\ell$ as specified by the modified Z-rule. The definition of a move $(i, j, k)$ and the associated *tabu list* $\tau(i, j, k)$ not only keeps track of which two jobs are interchanged, but also of the machine on which this is done. Indeed, to prevent cycling, we only need to prevent undoing a non-improving move on the relevant machine. As such, for the purpose of defining the neighborhood and the tabu list, the order of the schedules on the different machines matters. We therefore represent a solution as an ordered tuple $(\sigma_1, \ldots, \sigma_m)$ rather than as an unordered multiset $\{\sigma_1, \ldots, \sigma_m\}$.

Observe that rather than running through the remaining machines in an arbitrary order after having performed a swap on a given machine, we could also make this order a part of the move. This order is relevant for future iterations because the machines differ with respect to which pairwise interchanges are allowed by the tabu list. By considering all possible orders, however, the size of the neighborhood increases substantially. Preliminary computational experiments have shown that the increase in solution quality thanks to this larger neighborhood does not outweigh the additional time needed to explore it. Therefore, we chose to rely on a random order as described in Algorithm 6.

For the specific case of two machines, the tabu search described in Algorithm 6 is equivalent to the one developed by [65]. Our current approach therefore directly generalizes the one of [65] to an arbitrary number of machines.

### 3.3.7 Computational experiments

In this section, we assess the quality of the new upper bound and heuristics by means of computational experiments. To do so, we use the same instances as in Section 3.2.5. In the following, first, we briefly recall how the instances are generated. Then, we focus on the special case with two machines, i.e., *ERM2*, so that we can directly compare the performance of the sequential upper bound and the new heuristics with those of the 3AP-based upper bound and the tabu search for two machines proposed in Section 3.2. Next, we focus on *ERM3* to assess the quality of the new generalized tabu search algorithm and compare it with the other heuristics and the sequential upper bound. Finally, we show how the gap between our upper bound and heuristics scales in the number of machines.

#### Instance description and implementation details

As introduced in Section 3.2.5, 20 instances have been generated for each combination of the number of jobs $n \in \{10, 20, 30, 40, 50\}$ and intervals $I_p \in \{[0.9, 1], [0.5, 1], [0.1, 1]\}$ for the success probability. For

each job $i = 1, \ldots, n$ in a given instance, a success probability $p_i$ was drawn uniformly at random from the corresponding interval $I_p$, while the revenue $r_i$ was drawn from a integer uniform distribution on the interval $[10, 100]$. This then lead to $20 \times 5 \times 3 = 300$ instances in total.

The tabu search has been run as a Python 3.7 program on a single 1.7 GHz CPU core with 8 GB RAM. All heuristics and the upper bound introduced in this chapter have been run on a single 1.7 GHz CPU core and with 8GB of RAM. The tabu search was implemented as a Python 3.7 program, whereas the other methods were implemented using the C++ programming language and compiled with Microsoft Visual C++ 14.0.

As a starting solution for MBRH and tabu search, we used the solution obtained by MZRH and MBRH, respectively.

### Performance for *ERM2*

For the specific case of *ERM2*, in Section 3.2 we proposed a quadratic integer program, a tabu search heuristic, and an upper bound based on a three-dimensional assignment problem (3AP). Although the quadratic integer program cannot be solved exactly within a reasonable computation time, it does provide a feasible solution and a valid upper bound. As such, it can be used as a heuristic whose performance can be evaluated by means of the obtained upper bound. Among these methods, the tabu search and the 3AP-based upper bound turned out to perform best. Therefore, we only compare the new upper bound and heuristics to these latter two methods.

Table 3.6 displays the result of this experiment on *ERM2*-instances. For a given value V (be it a lower bound obtained through a heuristic or the upper bound provided by 3AP), the percentage gap is computed relative to the sequential upper bound $\mathrm{UB_s}$, presented in Section 3.3.2, as:

$$\frac{\mathrm{UB_s} - \mathrm{V}}{\mathrm{UB_s}} \times 100\%.$$

Table 3.6: Average percentage gap and cpu time in seconds for *ERM2* instances.

| | | ZRH | MZRH | MBRH | Tabu Search | | 3AP bound | |
|---|---|---|---|---|---|---|---|---|
| $n$ | $I_p$ | % gap | % gap | % gap | % gap | cpu | % gap | cpu |
| 10 | $[0.1,1]$ | 8.21 | 6.96 | 6.53 | 6.27 | 0.24 | −8.10 | 0.22 |
| 10 | $[0.5,1]$ | 8.66 | 6.94 | 5.89 | 5.52 | 0.25 | −7.52 | 0.94 |
| 10 | $[0.9,1]$ | 1.38 | 0.88 | 0.73 | 0.65 | 0.24 | −0.78 | 0.97 |
| 20 | $[0.1,1]$ | 10.52 | 8.41 | 7.29 | 6.72 | 3.76 | −12.72 | 1.05 |
| 20 | $[0.5,1]$ | 10.85 | 8.60 | 7.08 | 6.37 | 3.31 | −19.83 | 1.02 |
| 20 | $[0.9,1]$ | 3.37 | 2.35 | 1.88 | 1.60 | 3.64 | −2.64 | 1.07 |
| 30 | $[0.1,1]$ | 10.90 | 8.51 | 7.40 | 6.54 | 11.80 | −20.25 | 1.60 |
| 30 | $[0.5,1]$ | 11.96 | 9.40 | 7.56 | 6.77 | 11.47 | −27.10 | 1.52 |
| 30 | $[0.9,1]$ | 5.82 | 3.81 | 2.98 | 2.47 | 10.84 | −5.84 | 1.51 |
| 40 | $[0.1,1]$ | 10.51 | 8.65 | 7.11 | 6.48 | 27.00 | −28.06 | 4.44 |
| 40 | $[0.5,1]$ | 11.63 | 9.31 | 7.39 | 6.46 | 26.46 | −31.26 | 3.98 |
| 40 | $[0.9,1]$ | 7.53 | 5.40 | 4.07 | 3.43 | 25.81 | −7.95 | 4.81 |
| 50 | $[0.1,1]$ | 11.57 | 9.03 | 7.29 | 6.46 | 55.92 | −27.60 | 13.95 |
| 50 | $[0.5,1]$ | 11.68 | 9.18 | 7.05 | 6.27 | 55.08 | −36.39 | 9.78 |
| 50 | $[0.9,1]$ | 9.10 | 6.08 | 4.75 | 3.84 | 54.68 | −12.56 | 10.66 |

In addition to the percentage gap, the table also reports on the average cpu time in seconds for the tabu search and the 3AP bound. All other heuristics needed less than 0.001 seconds to solve any of the instances. The averages are taken over all twenty instances for the corresponding value of $n$ and interval $I_p$ as indicated by the table's rows.

Table 3.6 clearly shows that the more advanced heuristics also result in better solutions. When comparing the corresponding cpu-times, however, we observe that for larger instances the tabu search needs a relatively large amount of time to improve upon the solution obtained by MBRH.

The performance of all heuristics follows the same overall trend: with $I_p = [0.9,1]$, the gap clearly increases in the number of jobs $n$, whereas it is much more stable for $I_p = [0.5,1]$ or $I_p = [0.1,1]$. Moreover, when comparing the different intervals of the success probability, we see that the gap increases between $I_p = [0.9,1]$ and $I_p = [0.5,1]$ and, perhaps somewhat surprisingly, the gap stabilizes between $I_p = [0.5,1]$ and $I_p =$

$[0.1, 1]$. Since we do not know the optimal solution value, however, it is not clear which part of the gap is caused by the heuristic and the sequential upper bound, respectively.

With respect to the quality of the bound, we find that the sequential upper bound clearly dominates the one provided by 3AP, both in the quality of the bound as in the computation time. Indeed, for all considered instances, it took less than 0.001 seconds to compute the sequential upper bound, and its value was strictly smaller than the one provided by 3AP for all but one of the instances. As Table 3.6 shows, the average percentage gap of 3AP is always negative, and the sequential upper bound outperforms the 3AP bound especially when $n$ is larger and $I_p = [0.5, 1]$.

### Performance for *ERM3*

Table 3.7 provides the results for three-machine problems. The 3AP-based upper bound does not appear in the table since it only applies to *ERM2*. Moreover, we also need to use tabu search described in Section 3.3.6 that generalizes the method of Section 3.2.3 to more than two machines.

The computational results for *ERM3* display a similar pattern as for *ERM2*. When comparing Tables 3.6 and 3.7, however, we see that the average gaps decrease for $I_p = [0.9, 1]$ and $n \leq 30$, while they increase for most other settings. One possible explanation for this is that when $I_p = [0.9, 1]$ and $n$ is small, then it is easier to find a schedule in which most jobs are likely to be carried out with $m = 3$ than with $m = 2$. As the number of jobs increases or the range of success probabilities becomes wider, this effect dissipates and the gap increases, instead of decreasing, with the number of machines. In the next section we will demonstrate this effect even more clearly.

Especially for larger instances, we observe that the tabu search needs a relatively large amount of time to yield only a limited improvement over MBRH. It seems that, compared to the case of *ERM2*, the enlarged neighborhood harms the tabu search's performance. Since this effect would only get worse as the number of machines increases, it seems that the tabu search is mainly relevant for a relatively small number of machines.

Table 3.7: Average percentage gap and cpu time in seconds for *ERM3* instances.

| | | ZRH | MZRH | MBRH | Tabu Search | |
|---|---|---|---|---|---|---|
| $n$ | $I_p$ | % gap | % gap | % gap | % gap | cpu |
| 10 | $[0.1,1]$ | 13.52 | 10.12 | 9.43 | 9.24 | 1.40 |
| 10 | $[0.5,1]$ | 12.68 | 8.29 | 7.51 | 7.26 | 1.49 |
| 10 | $[0.9,1]$ | 0.80 | 0.37 | 0.33 | 0.32 | 1.58 |
| 20 | $[0.1,1]$ | 17.00 | 11.67 | 10.53 | 10.38 | 12.88 |
| 20 | $[0.5,1]$ | 17.53 | 11.60 | 9.93 | 9.70 | 13.98 |
| 20 | $[0.9,1]$ | 2.81 | 1.43 | 1.19 | 1.18 | 15.62 |
| 30 | $[0.1,1]$ | 17.61 | 11.90 | 10.43 | 10.26 | 48.99 |
| 30 | $[0.5,1]$ | 19.29 | 12.89 | 10.86 | 10.61 | 55.83 |
| 30 | $[0.9,1]$ | 6.17 | 3.09 | 2.50 | 2.49 | 65.93 |
| 40 | $[0.1,1]$ | 17.12 | 11.82 | 10.21 | 10.04 | 128.16 |
| 40 | $[0.5,1]$ | 18.80 | 12.47 | 10.30 | 10.08 | 139.66 |
| 40 | $[0.9,1]$ | 9.03 | 4.92 | 3.94 | 3.94 | 174.16 |
| 50 | $[0.1,1]$ | 18.68 | 12.31 | 10.55 | 10.30 | 263.77 |
| 50 | $[0.5,1]$ | 19.06 | 12.49 | 10.26 | 10.05 | 301.88 |
| 50 | $[0.9,1]$ | 11.84 | 6.35 | 5.04 | 5.02 | 394.70 |

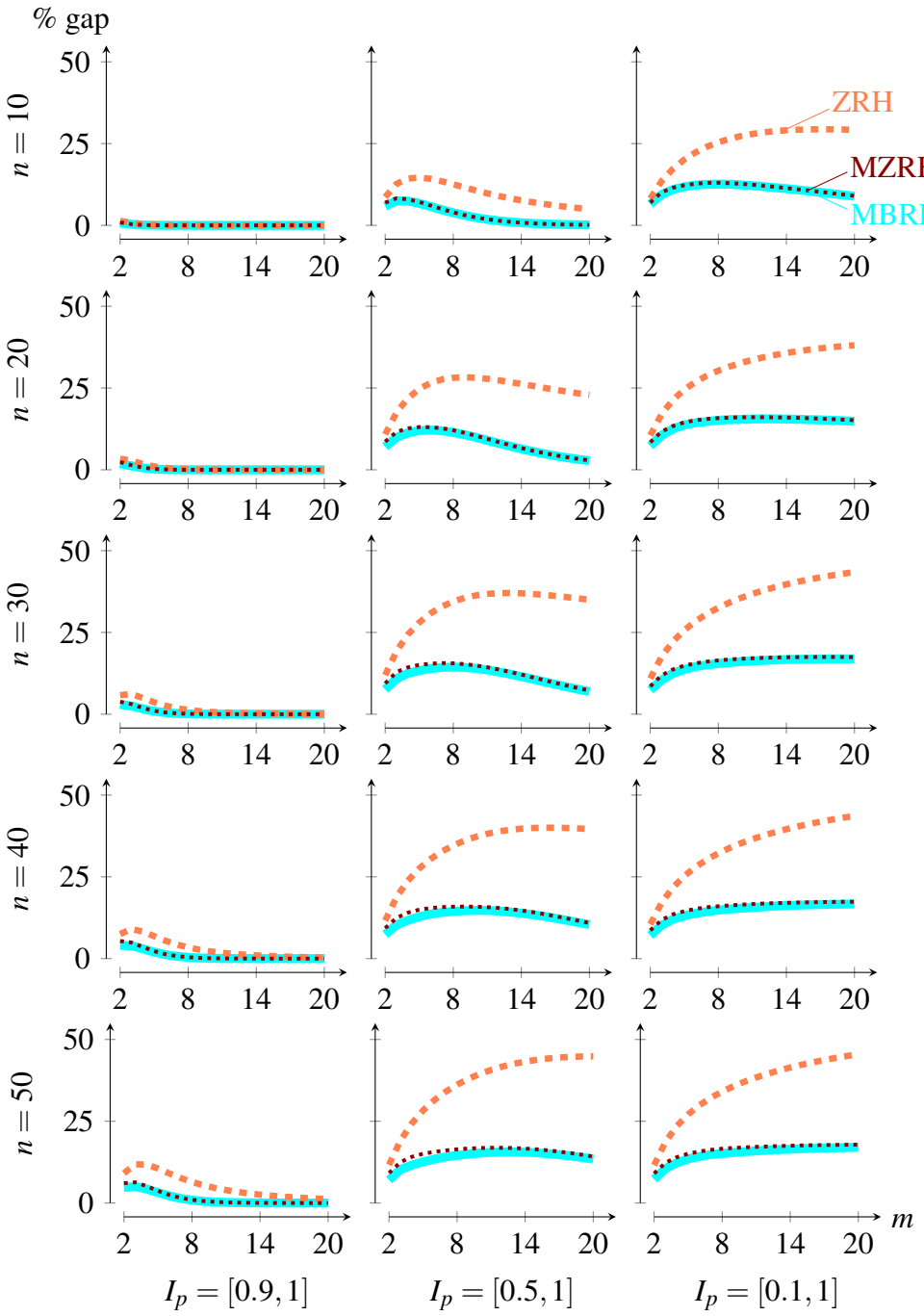
For this reason, we did not test the tabu search for more than three machines.

### Performance for *ERM*$m$

Figure 3.1 provides insight into how the gap between our heuristics (other than tabu search) and the sequential upper bound scales in $m$. The figure displays a similar pattern to what we observed when comparing Tables 3.6 and 3.7. The gaps initially tend to increase in $m$, but, as $m$ becomes larger, this growth slows down and, for most settings, eventually becomes negative. In fact, as the number of machines becomes sufficiently large, most jobs are likely to be carried out, leading to a smaller gap.

This empirical finding shows that the average percentage gap does not increase monotonically in the number machines. This seems to conflict with the fact that, as derived in Section 3.3.3, the worst-case performance does deteriorate monotonically in $m$. However, in establishing the asymptotic tightness of

Figure 3.1: Average percentage gap for ZRH, MZRH, and MBRH as a function of the number of machines $m$. The graph is repeated for different values of $n \in \{10, 20, \ldots, 50\}$ and intervals for the success probabilities $I_p \in \{[0.9, 1], [0.5, 1], [0.1, 1]\}$. Gaps are computed relative to the sequential upper bound $\mathrm{UB_s}$ as $(\mathrm{UB_s} - \mathrm{V})/\mathrm{UB_s} \times 100\%$ for the corresponding value V.



this worst-case guarantee we considered a family of instances in which $n$ tends to infinity. As such, our computational experiment suggests that one could perhaps derive a better performance guarantee if one parameterizes on the number of jobs $n$.

Comparing the average percentage gaps for ZRH and MZRH with the related worst-case performance guarantees (i.e., $H_m/m$ and $1-(\frac{m-1}{m})^m$, respectively), our computational results suggest that the average performance of these heuristics is considerably better than the worst-case performance. Moreover, the empirical performance can be further slightly improved by using the MBRH heuristic, even if it is still an open question whether MBRH yields a better worst-case guarantee than MZRH.

## 3.4 Conclusions

In this chapter we addressed the problem of sequencing unreliable jobs on parallel machines when job replication is allowed. In particular, we defined the Expected Revenue Maximization problem (*ERM*), where a set of jobs with failure probabilities are replicated and each copy is scheduled on a different machine. A revenue $r_j$ is attained when a job $J_j$ is carried out, and the problem is that of maximizing the total expected revenue. We first addressed the problem with $m = 2$ machines, showing that it is NP-Complete and providing a quadratic integer programming formulation, an upper bound based on the Three Dimensional Assignment problem (3AP) and a tabu search approach. We showed that the latter finds good solutions when job success probabilities are large. Then, we focused to the general case of m machines, proposing a novel upper bound (the sequential upper bound) and four heuristic approaches, namely the Z-rule heuristic, the modified Z-rule heuristic, the mutual-best-reply heuristic, and a new generalized tabu search. Our main theoretical contribution is to show that the Z-rule heuristic yields an expected revenue of at least $H_m/m$ times the sequential upper bound, where $H_m = \sum_{k=1}^{m} 1/k$ is the $m^{\text{th}}$ harmonic number. Moreover, this bound is asymptotically tight. This result simultaneously yields a worst-case guarantee for the Z-rule heuristic's performance as well

as for the gap between the sequential upper bound and the maximum expected revenue. Furthermore, we prove that the slightly more complex modified Z-rule heuristic is $1 - (\frac{m-1}{m})^m$ approximate.

Based on extensive computational experiments, we also find that our proposed methods display a good empirical performance. In particular, the sequential upper bound performs significantly better than the bound based on the three-dimensional assignment problem proposed for *ERM2*. With respect to the heuristics, especially the modified Z-rule heuristic and the mutual-best-reply heuristic seem to result in a high-quality solution in very limited computation time. An interesting open question is whether also the mutual-best-reply heuristic leads to an improved approximation guarantee.

The tabu search, however, only seems practical for instances with two or three machines. For larger values of $m$ the neighborhood becomes too large to be effectively explored.

Several possible venues for future research can be considered, including the following.

($i$) The complexity status of relevant special cases (e.g., those introduced in Section 3.2.5) is still unknown and hence requires further analysis.

($ii$) The mathematical programming formulation proposed in Section 3.2.2 solves to optimality only relatively small instances of *ERM2*. A promising direction for further research consists in developing exact solution methods. Indeed, since there are currently no such methods available, it is unclear what part of the gap between our sequential upper bound and the heuristics is due to performance of the upper bound and the heuristics, respectively. One main challenge in developing such exact solution methods is the highly non-linear objective function due to the product of probabilities.

($iii$) Further improvements might be obtained in the tabu search performances by considering other neighborhood definitions that scale better in the number of machines.

# Conclusions

In this dissertation, we focused on three different problems. The first problem (Chapter 1) arises from a real-world application in healthcare and concerns the transportation of perishable biological samples from draw centers to a main laboratory for analysis. The samples must be delivered within their lifetime but dedicated facilities called spoke centers are available to extend such lifetime. At spoke centers the transfer of samples between vehicles is allowed. To solve the problem, we developed three different mixed integer linear programming models (MILP) and different hybrid metaheuristic algorithms based on the Adaptive Large Neighborhood Search framework (H-ALNS), for which several ad-hoc procedures have been devised. Computational experiments on different sets of instances, based on real-life data provided by the Local Healthcare Authority of Bologna, Italy, have been also presented. A comparison between the solutions obtained by MILP formulations and the H-ALNS algorithms on small instances shows the effectiveness of the proposed metaheuristics and assesses the superiority of the two time-indexed MILP models with respect to the simple MILP one. On real-life instances, due to the big number of samples to deliver each day, we proposed an experimental study to evaluate different grouping policies. The computational results show that the H-ALNS algorithms are able to find solutions in which all samples are delivered on time while, in the real case, the lifetime requirements of samples are not currently satisfied. This means that the H-ALNS algorithms together with the devised grouping policies can improve significantly the decision-making process in this kind of problems.

In Chapter 2, we focused on the decision problem of crop planning in sustainable agriculture taking into account crop rotation benefits across growing seasons is considered. A formal characterization for the problem has been given. We performed a complexity study showing that the problem is NP-hard with rotation schemes of three or more crops. A special case of the problem that can be solved through a polynomial network-flow approach has been also presented. Furthermore, we formulated three different variants of the problem, accounting for different sustainable scenarios, in order to evaluate the effectiveness of existing public (Common Agricultural Policy) and private (Carta del Mulino) initiatives on the use of the soil and on the profits of the farmers. Different integer linear programs have been developed for each variant and tested through an extensive computational campaign. The data were provided by 23 Italian farms and the experiments have been performed on the problem considering rotation schemes of three crops (CRP-3), which is common practice in the Mediterranean pedo-climatic context. The results show that the proposed approaches could be embedded in a decision support tool that can be practically used by farmers to plan their production and to evaluate the convenience of joining sustainability initiatives. On the other hand, the models can also be employed by regulatory bodies to tune the incentives of such initiatives. The policy makers will be able to use the results to assess the suitability of rules for farm sustainability transition.

Finally, in Chapter 3 we addressed the problem of replicating and sequencing unreliable jobs on parallel machines. We first addressed the problem with $m = 2$ machines, showing that it is NP-Complete and providing a quadratic integer programming formulation, an upper bound based on the Three-Dimensional Assignment problem (3AP) and a tabu search approach. We showed that the latter finds good solutions when job success probabilities are large. Then, we focused to the general case of m machines, proposing a novel upper bound (the sequential upper bound) and four heuristic approaches, namely the Z-rule heuristic, the modified Z-rule heuristic, the mutual-best-reply heuristic, and a new generalized tabu search. Our main theoretical contribution is to prove the worst case performance guarantee of the Z-rule heuristic and the modified Z-rule heuristic. We also derived a performance guarantee of the gap between the sequential upper

bound and the maximum expected revenue. Based on extensive computational experiments, we also find that our proposed methods display a good empirical performance. In particular, the sequential upper bound performs significantly better than the bound based on the three-dimensional assignment problem proposed for *ERM2*. With respect to the heuristics, especially the modified Z-rule heuristic and the mutual-best-reply heuristic seem to result in a high-quality solution in very limited computation time.

The problems addressed in this dissertation come from totally different fields, but can be modeled and solved in a rigorous and quantitative way by applying combinatorial optimization techniques. The results reported in this dissertation show the effectiveness of the proposed methods and algorithms that can be helpful tools for managers to improve decision-making in real-world problems.

# Bibliography

## Bigliography of Chapter 1


[1] Anaya-Arenas A. M., Prodhon C., Renaud J., Ruiz A., An iterated local search for the biomedical sample transportation problem with multiple and interdependent pickups, *Journal of the Operational Research Society*, 72(2), 367-382, 2021.

[2] Azezan N.A., Ramli M.F., Masran H., A review on the modelling of collection and distribution of blood donation based on vehicle routing problem, *In AIP Conference Proceedings*, 1905, 1, 2017.

[3] Benini M., Detti P., Zabalo Manrique de Lara G., Mathematical programming formulations and metaheuristics for biological sample transportation problems in healthcare, Computers & Operations Research, Volume 146, 2022.

[4] Benini M., Detti P., Zabalo Manrique de Lara G., A milp model for biological sample transportation in healthcare, *In: M. Paolucci et al. (eds), Advances in Optimization and Decision Science for Society, AIRO Springer Series*, 3:81–94, 2019.

[5] Bonadies A., Mancini R., Maci M., Gibertoni C., Petrini A., Laboratorio unico metropolitano: innovazione e alta tecnologia per un nuovo paradigma di medicina di laboratorio, *MECOSAN*, XXIX(115):79–94, 2020.

[6] Cortés C.E., Matamala M., Contardo C., The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method, *European Journal of Operational Research*, 200(3):711–724, 2010.

[7] Detti P., Papalini F., Zabalo Manrique de Lara G., A multi-depot dial-a-ride problem with heterogeneous vehicles and compatibility constraints in healthcare, *Omega*, 70:1–14, 2017.

[8] Detti P., Zabalo Manrique de Lara G., and Benini M., A metaheuristic apporach for biological sample transportation in healthcare, *In: G. Gentile et al. (eds), Graphs and Combinatorial Optimization: from Theory to Applications, AIRO Springer Series*, 5:81–94, 2021.

[9] Doerner K., Gronalt M., Hartl R., Kiechle G., Reimann M., Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows, *Computers & Operations Research*, 9(35):3034 – 3048, 2008.

[10] Gendreau M., Hertz A., Laporte G., A tabu search heuristic for the vehicle routing problem, *Management Science*, 40(10):1276–1290, 1994.

[11] Grasas A., Ramalhinho H, Pessoa L.S., Resende M.G., Caballé I., Barba N., On the improvement of blood sample collection at clinical laboratories, *BMC health services research*, 14(1):12, 2014.

[12] Karakoc M., Gunay M., Priority based vehicle routing for agile blood transportation between donor/client sites, In *Computer Science and Engineering (UBMK), 2017 International Conference on*, pages 795–799. IEEE, 2017.

[13] Heidari-Fathian H., Reza Pasandideh S. H., Green-blood supply chain network design: Robust optimization, bounded objective function & Lagrangian relaxation, *Computers & Industrial Engineering*, 2018.

[14] Karakostas P., Panoskaltsis N., Mantalaris A., Georgiadis M.C., Optimization of CAR T-cell therapies supply chains, *Computers & Chemical Engineering*, 139, 106913, 2020.

[15] Kramer R., Cordeau J., Iori M., Rich vehicle routing with auxiliary depots and anticipated deliveries: an application to pharmaceutical distribution, *Transp. Res Part E*, 129, 162–174, 2019.

[16] Liu R., Xie X., Garaix T., Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics, *Omega*, 47:17–32, 2014.

[17] Liu R., Xie X., Augusto V., RodriguezC., Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care, *European Journal of Operational Research*, 230(3), 475–486, 2013.

[18] Masson R., Lehuédé F., Péton O., An adaptive large neighborhood search for the pickup and delivery problem with transfers, *Transportation Science*, 47(3):344–355, 2013.

[19] Mitrović-Minić S. Laporte G., The pickup and delivery problem with time windows and transshipment, *INFOR: Information Systems and Operational Research*, 44(3):217–227, 2006.

[20] Moussavi S., Mahdjoub M., Grunder O., A matheuristic approach to the integration of worker assignment and vehicle routing problems: application to home healthcare scheduling, *Expert Syst. Appl.*, 125, 317–332, 2019.

[21] Parragh S.N., Doerner K.F., Hartl R.F., Variable neighborhood search for the dial-a-ride problem, *Computers & Operations Research*, 37(6):1129–1138, 2010.

[22] Pirabán A., Guerrero W.J., Labadie N., Survey on blood supply chain management: Models and methods, *Computers & Operations Research*, Volume 112, 2019.

[23] Pisinger D., Ropke S., A general heuristic for vehicle routing problems, *Computers & operations research*, 34(8):2403–2435, 2007.

[24] Potvin J.Y., Rousseau J.M., A parallel route building algorithm for the vehicle routing and scheduling problem with time windows, *European Journal of Operational Research*, 66(3):331–340, 1993.

[25] Rais A., Alvelos F., Carvalho M., New mixed integer-programming model for the pickup-and-delivery problem with transshipment, *European Journal of Operational Research*, 3(235):530–539, 2014.

[26] Ropke S., Pisinger D., An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows, *Transportation science*, 40(4):455–472, 2006.

[27] Şahinyazan F.G., Kara B.Y., Taner M.R., Selective vehicle routing for a mobile blood donation system, *European Journal of Operational Research*, 245(1):22–34, 2015.

[28] Sapountzis C., Allocating blood to hospitals as a multiobjective transportation problem. In *Medical Informatics Europe'90*, pages 733–739. Springer, 1990.

[29] Shaw P., Using constraint programming and local search methods to solve vehicle routing problems, *In: Maher M, Puget JF, eds. Proc. 4th Internat. Conf. Principles and Practice of Constraint Programming*, 3:417–431, 1998.

[30] Shi, Y. et al., A hybrid genetic algorithm for a home health care routing problem with time window and fuzzy demand, *Expert Systems with Applications*, 2017

[31] Zhang Z., Liu M., Lim A., A memetic algorithm for the patient transportation problem, *Omega*, 54:60–71, 2015.

# Bigliography of Chapter 2


[32] AGER – Bologna. Accessed July 21, 2022. `https://agerborsamerci.it/listino/listino.html`.

[33] Ahumada O., Villalobos J.R., Application of planning models in the agrifood supply chain: A review, *European Journal of Operational Research*, 196(1), 1–20, 2009

[34] Alfandari L., Plateau A., Schepler X., A branch-and-price-and-cut approach for sustainable crop rotation planning, *European Journal of Operational Research*, Volume 241, Issue 3, 872-879, 2015

[35] Bachinger J. and Zander P., ROTOR, a tool for generating and evaluating crop rotations for organic farming systems, *European Journal of Agronomy*, 26(2), pp. 130–143, 2007

[36] Baldoni R. and Giardini L., Coltivazioni Erbacee. Pátron, Bologna ('Herbaceous Cultivations'), 2002

[37] Bonciarelli U., Onofri A., et al., Long-term evaluation of productivity, stability and sustainability for cropping systems in Mediterranean rainfed conditions, *European Journal of Agronomy*, 77, 146–155, 2016

[38] Boyabatli O., Nasiry J., and Zhou, Y., Crop Planning in Sustainable Agriculture: Dynamic Farmland Allocation in the Presence of Crop Rotation Benefits, *Management Science*, 65, (5), 2060-2076, 2019

[39] Britz W., van Ittersum M., Lansink A.O., Heckelei T., Tools for integrated assessment in agriculture State of the Art and Challenges, *Bio-Based Appl. Econ.*, 1 (2), 125–150, 2012

[40] Butkevicien L. M. et al., The Influence of Long-Term Different Crop Rotations and Monoculture on Weed Prevalence and Weed Seed Content in the Soil, *Agronomy*, 11, 2021

[41] de Frahan B.H., Buysse J., et al., Positive mathematical programming for agricultural and environmental policy analysis: review and practice, *Int. Ser. Oper. Res. Manag. Sci.*, 99, 129–154, 2016

[42] Detlefsen N., K. and Jensen A.L., Modelling optimal crop sequences using network flows, *Agricultural Systems*, 94(2), 566–572, 2007

[43] Di Bene C., Dolores Gómez-López M., Francaviglia R., Farina R., Blasi E., Martínez-Granados D. and Calatrava J., Barriers and Opportunities for Sustainable Farming Practices and Crop Diversification Strategies in Mediterranean Cereal-Based Systems, *Front. Environ. Sc.*, 10, 2022

[44] Decreto Minesteriale 18 luglio 2018, n. 6793 modificato da DM 9 aprile 2020, n. 3775. (Ministerial Decree n. 6793, 18 July 2018, modified by Ministerial Decree n. 3775, 9 April 2020). Accessed July 21, 2022. `https://www.gazzettaufficiale.it/eli/id/2020/06/22/20A03222/sg`.

[45] Dury J. et al., Models to support cropping plan and crop rotation decisions. A review. Agronomy for Sustainable Development, 32(2), 567–580, 2012

[46] Fikry I., Gheith M., Eltawil A., A New Mathematical Model for the Deterministic Crop Rotation Planning Problem, *IEEE Int. Conf. on Ind. Eng. and Eng. Man. (IEEM)*, 314-318, 2019

[47] Filippi C., Mansini R., Stevanato E., Mixed integer linear programming models for optimal crop selection, *Computers & Operations Research*, Volume 81, 26-39, 2017

[48] Garey M.R. and Johnson D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, *Freeman*, 1979

[49] Grunert K.G., Hieke S., Wills J., Sustainability labels on food products: Consumer motivation, understanding and use, *Food Policy*, 44, 177-189, 2014

[50] Haneveld W.K.K. and Stegeman A.W. Crop succession requirements in agricultural production planning, *European Journal of Operational Research*, 166(2), 406–429, 2005

[51] Hennessy D.A., On monoculture and the structure of crop rotations, *Am. J. Agric. Econ.*, 88, 900–914, 2006

[52] 'La Carta del Mulino': the Mulino Bianco charter for sustainable soft wheat. Accessed July 21, 2022. https://www.barillagroup.com/en/stories/stories-list/carta-del-mulino-for-sustainable-agriculture.

[53] Munari G.R., Improved mathematical model and bounds for the crop rotation scheduling problem with adjacency constraints, *European Journal of Operational Research*, Volume 278, Issue 1, 120-135, 2019

[54] National Agricultural Information System. Accessed September 22, 2022. https://www.sian.it/consRese/paiRicerca.do.

[55] van der Ploeg J.D., Barjolle D., Bruil J., et al., The economic potential of agroecology: Empirical evidence from Europe, *Journal of Rural Studies*, 71, 46-61, 2019

[56] Santos L.M.R., Costa A.M., Arenales M., Santos R.H.S., Sustainable vegetable crop supply problem, *European Journal of Operational Research*, Volume 204, Issue 3, 639-647, 2010

[57] Santos L.M.R., Michelon P., Arenales M., Santos, R.H.S., Crop rotation scheduling with adjacency constraints, *Annals OR*, 190, 165-180, 2011

[58] Santos L.M.R., Munari P., Costa A.M., Santos R.H.S., A branch-price-and-cut method for the vegetable crop rotation scheduling problem with minimal plot sizes, *European Journal of Operational Research*, Volume 245, Issue 2, 581-590, 2015

[59] Sharma R., Kamble S.S., Gunasekaran A., Kumar V., Kumar A., A systematic literature review on machine learning applications for sustainable agriculture supply chain performance, *Computers & Operations Research*, Volume 119, 2020

[60] Volte G., Bourreau E., Giroudeau R., Naud O., Using VRPSolver to efficiently solve the Differential Harvest Problem, *Computers & Operations Research*, Volume 149, 2023

[61] Wezel A., Casagrande M., Celette F. et al., Agroecological practices for sustainable agriculture: A review, *Agron. Sustain. Dev.*, 34, 1–20, 2014

[62] Yan B., Chen X., Cai C., Guan S., Supply chain coordination of fresh agricultural products based on consumer behavior, *Computers & Operations Research*, Volume 123, 2020

[63] Yigezu Y.A., El-Shater T., Boughlala M. et al., Legume-based rotations have clear economic advantages over cereal monocropping in dry areas, *Agron. Sustain. Dev.*, 39, 58, 2019

# Bigliography of Chapter 3

[64] Adiri I., Bruno J., Frostig E., Rinnooy Kan A.H.G., Single machine flow-time scheduling with a single breakdown, *Acta Informatica*, 26, 679–696, 1989.

[65] Agnetis A., Benini M., Detti P., Hermans B., Pranzo M.. Replication and sequencing of unreliable jobs on parallel machines. Computers & Operations Research 139, 105634, 2021

[66] Agnetis A., Detti P. Pranzo M., Sodhi M. S., Sequencing unreliable jobs on parallel machines, *Journal of Scheduling*, 12, 45–54, 2009

[67] Agnetis A., Detti P., Martineau P., Scheduling nonpreemptive jobs on parallel machines subject to exponential unrecoverable interruptions, *Computers and Operations Research*, 79, 109–118, 2017.

[68] Agnetis A., Detti P., Pranzo M., The list scheduling algorithm for scheduling unreliable jobs on two parallel machines, *Discrete Applied Mathematics*, 165, 2–11, 2014.

[69] Agnetis A., Lidbetter T., List scheduling is 0.8531-approximate for scheduling unreliable jobs on *m* parallel machines, *Operations Research Letters*, 48, 405–409, 2020.

[70] Benoit A., Robert Y., Rosenberg A.L., Vivien F., Static Strategies for Worksharing with Unrecoverable Interruptions, *Theory of Computing Systems*, 53, 386–423, 2013.

[71] Burkard R.E., Rudolf R., Woeginger G.J., Three-dimensional axial assignment problems with decomposable cost coefficients. *Discrete Applied Mathematics*, 65(1-3), 123-139, 1996.

[72] Frostig E., A Note on Stochastic Scheduling on a Single Machine Subject to Breakdown–The Preemptive Repeat Model, *Probability in the Engineering and Informational Sciences*, 5, 349–354, 1991.

[73] Garey M.R., Johnson D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., 1979.

[74] Nemhauser G.L., Wolsey L.A., Fisher M.L., An analysis of approximations for maximizing submodular set functions - I, *Mathematical Programming*, 14, 265–294, 1978

[75] Ngetal C.T., Barketau M.S., Cheng T.C.E., Kovalyov M.Y., "Product partition" and related problems of scheduling and systems reliability: Computational complexity and approximation, *European Journal of Operational Research*, 207, 601–604, 2010.

[76] Schmidt G., Scheduling with limited machine availability, *European Journal of Operational Research*, 121, 1–15, 2000.

[77] Shabtay D., Gaspar N., Kaspi M., A survey on offline scheduling with rejection, *Journal of Scheduling*, 16, 3-28, 2013.

[78] Qi X., Bard J.F., Yu G., Disruption management for machine scheduling: The case of SPT schedules, *International Journal of Production Economics*, 103, 166–184, 2006.

[79] Stadje W., Selecting jobs for scheduling on a machine subject to failure, *Discrete Applied Mathematics*, 63, 257–265, 1995.