Normandie Université

UNIVERSITÉ LE HAVRE NORMANDIE

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité **INFORMATIQUE**

Préparée au sein de l'**Université Le Havre Normandie**

Problems, models and methods for risk reduction after industrial disasters involving dangerous substances

Présentée et soutenue par

THIAGO JOBSON BARBALHO

Thèse soutenue le 19/03/2024

devant le jury composé de :

M. NABIL ABSI	PROFESSEUR DES UNIVERSITES - ECOLE NATIONALE DES MINES DE ST ETIENNE	Rapporteur du jury
MME SAFIA KEDAD-SIDHOUM	PROFESSEUR DES UNIVERSITES - CNAM DE PARIS	Rapporteur du jury
MME NATHALIE BOSTEL	PROFESSEUR DES UNIVERSITES - UNIVERSITE NANTES	Membre du jury
M. GUILHERME DE CASTRO PENA	ASSOCIATE PROFESSOR - UNIVERSIDADE FEDERAL DE SAO JOAO DEL-REI	Membre du jury
M. CHRISTOPHE DUHAMEL	MAÎTRE DE CONFERENCES - Université Le Havre Normandie	Membre du jury
M. JUAN LUIS JIMENEZ LAREDO	ASSOCIATE PROFESSOR - UNIVERSIDAD DE GRANADA	Membre du jury
MME ANDREA CYNTHIA SANTOS	PROFESSEUR DES UNIVERSITES - Université Le Havre Normandie	Directeur de thèse

Thèse dirigée par **ANDREA CYNTHIA SANTOS** (LABORATOIRE D'INFORMATIQUE DE TRAITEMENT DE L'INFORMATION ET DES SYSTEMES)

MI/IΣ

litis

Mathématiques, Information, Ingénierie des Systèmes
Université Le Havre Normandie

Problems, models and methods for risk reduction after industrial disasters involving dangerous substances

Thiago Jobson Barbalho

Doctoral Thesis

2024

Abstract

In Europe, more than 250 major accidents involving industrial sites under the Seveso Directive were reported since 2010. Despite the regulations in place to prevent such accidents and minimize their impact, managing risk after such disasters remains a complex challenge. Once an industrial disaster occurs, preliminary on-the-ground information is collected to determine the extent of the accident, and operational decisions need to be made depending on the hazardous nature of the products involved and the extension of the affected area. In this thesis, we explore the realm of complex scheduling problems closely linked to risk factors stemming from the treatment (cleaning or neutralizing) of hazardous substances accidentally released by industrial sources. The primary objective is to develop effective optimization models and solutions that address the challenges faced by industries and logistical operations. In Chapter 3, we propose a new optimization problem as the Resource-Constrained Project Scheduling Problem with Risk and Priorities (RCPSP-RP), which combines Operations Research, Chemical Kinetics, and optimization methodologies. The RCPSP-RP integrates risk assessment and task prioritization, providing a framework for scheduling on-site operations to physically clean potential hazards. However, since the RCPSP-RP is a novel contribution, it employs a simplified approach that does not consider product transformation over time. That is, it is assumed that the state of the pollutants does not change with time. In Chapter 4, on the other hand, we introduce the Resource-Constrained Project Scheduling Problem with Risk and Product Transformation Dynamics (RCPSP-RTD) as a specialization of the RCPSP-RP, where the dynamic nature of product transformations are taken into consideration in the optimization process. Since the RCPSP-RTD accounts for time-dependent product evolution due to product transformation and degradation, the RCPSP-RTD offers a more realistic approach to optimizing risk mitigation. We provide mathematical formulations for both the RCPSP-RP and the RCPSP-RTD. In addition, an Iterated Local Search (ILS) metaheuristic is developed, and automatic calibrated by a tuning technique, for both problems. We propose several problem scenarios and investigate their numerical results obtained by the different optimization methods. The methods, in conjunction with the already existing risk management plans, can obtain interesting insights to be applied to realistic scenarios. The refinement of models to capture the complete dynamics of chemical transformations, alongside exploring the interplay between scheduling decisions and dynamic processes, holds promise. Expanding these models to embrace multi-objective optimization, encompassing objectives beyond risk, could render the framework more versatile and applicable. Leveraging advanced computational techniques, including machine learning and simulation, presents opportunities for enhancing the accuracy of risk mitigation strategy prediction.

Résumé

En Europe, plus de 250 accidents majeurs impliquant des sites industriels soumis à la Directive Seveso ont été signalés depuis 2010. Malgré les réglementations en place pour prévenir de tels accidents et minimiser leur impact, la gestion des risques après de telles catastrophes reste un défi complexe. Une fois qu'une catastrophe industrielle se produit, des informations préliminaires sur le terrain sont collectées pour déterminer l'étendue de l'accident, et des décisions opérationnelles doivent être prises en fonction de la nature dangereuse des produits impliqués et de l'extension de la zone affectée. Dans cette thèse, nous explorons le domaine des problèmes de planification complexes étroitement liés aux facteurs de risque découlant du traitement (nettoyage ou neutralisation) de substances dangereuses libérées accidentellement par des sources industrielles. L'objectif principal est de développer des modèles et des solutions d'optimisation efficaces qui répondent aux défis auxquels sont confrontées les industries et les opérations logistiques. Dans le Chapitre 3, nous proposons un nouveau problème d'optimisation sous la forme du Resource-Constrained Project Scheduling Problem with Risk and Priorities (RCPSP-RP), qui combine la recherche opérationnelle, la cinétique chimique et les méthodologies d'optimisation. Le RCPSP-RP intègre l'évaluation des risques et la priorisation des tâches, fournissant un cadre pour la planification des opérations sur site pour nettoyer physiquement les dangers potentiels. Cependant, puisque le RCPSP-RP est une contribution nouvelle, il emploie une approche simplifiée qui suppose que l'état des polluants ne change pas avec le temps. Dans le Chapitre 4, en revanche, nous introduisons le Resource-Constrained Project Scheduling Problem with Risk and Product Transformation Dynamics (RCPSP-RTD) comme une spécialisation du RCPSP-RP, où la nature dynamique des transformations de produits est prise en compte dans le processus d'optimisation. Puisque le RCPSP-RTD tient compte de l'évolution temporelle des produits due à la transformation et à la dégradation des produits, le RCPSP-RTD offre une approche plus réaliste pour optimiser l'atténuation des risques. Nous fournissons des formulations mathématiques pour le RCPSP-RP ainsi que pour le RCPSP-RTD. De plus, une métaheuristique Iterated Local Search (ILS) est développée et automatiquement calibrée par une technique de réglage de paramètres, pour les deux problèmes. Nous proposons plusieurs scénarios de problèmes et investiguons leurs résultats numériques obtenus par les différentes méthodes d'optimisation. Les méthodes, conjointement avec les plans de gestion des risques déjà existants, peuvent obtenir des aperçus intéressants à appliquer à des scénarios réalistes. Le raffinement des modèles pour capturer la dynamique complète des transformations chimiques, ainsi que l'exploration de l'interaction entre les décisions de planification et les processus dynamiques, est prometteur. L'expansion de ces modèles pour embrasser l'optimisation multi-objectifs, englobant des objectifs au-delà du risque, pourrait rendre le cadre plus polyvalent et applicable. L'exploitation de techniques de calcul avancées, y compris l'apprentissage automatique et la simulation, présente des opportunités pour améliorer la précision de la prédiction de la stratégie d'atténuation des risques.

Contents

List of figures

List of tables

List of algorithms

List of acronyms

ACS Ant Colony System. 38, 39

ALNS Adaptive Large Neighborhood Search. 41

CTSP-d Capacitated Traveling Salesman Problem with d–relaxed priority rule. 41

DAG Direct Acyclic Graph. 50

DCSP Debris Clearance Scheduling Problem. 42

DSUN Disruption Scheduling problem on Urban Networks. 42

EDD Earliest Due Date. 31

ESRP Early Stage Response Problem. 43

FIFO First In, First Out. 31

GA Genetic Algorithm. 36, 37

GRASP Greedy Randomized Adaptive Search Procedure. 41–43

HTSP Hierarchical Traveling Salesman Problem. 41

ILS Iterated Local Search. 5, 7, 13, 23, 24, 39–41, 43, 48, 55, 56, 59–64, 66–68, 70, 73, 86, 95, 104, 107, 108

IRACE Iterated Racing for Automatic Algorithm Configuration. 13, 48, 62

K-ARCP Multivehicle Synchronized Arc Routing Problem to Restore Network Connectivity. 42

LB Lower Bound. 62, 95, 96

LIFO Last In, First Out. 31

LITIS *Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes*. 60, 92

LNS Large Neighborhood Search. 43

LPT Longest Processing Time. 31

LST Latest Start Time. 39

MCMC Markov-Chain Monte Carlo. 76, 77

MILP Mixed Integer Linear Programming. 32, 35, 41, 43, 44, 75, 108

MRSU Multi-Resource Scheduling Model under Uncertainty. 44, 45

NRCSRP Network Repair Crew Scheduling and Routing Problem. 42

OR Operations Research. 22

PDDT Pulse Disaggregated Discrete Time. 33, 34

PSPLIB Project Scheduling Problem Library. 32, 36, 37, 39

RCESP Resource-Constrained Emergency Scheduling Problem for Forest Fires with Priority Areas. 44

RCPSP Resource-Constrained Project Scheduling Problem. 22, 23, 25, 28–30, 32, 35–40, 47, 50, 72

RCPSP-RP Resource-Constrained Project Scheduling Problem with Risk and Priorities. 5, 7, 13, 22, 23, 25, 41, 45–50, 60, 63, 71–73, 75, 107–109

RCPSP-RTD Resource-Constrained Project Scheduling Problem with Risk and Product Transformation Dynamics. 5, 7, 13, 22, 23, 25, 45, 46, 75, 83, 85, 91, 98, 103, 108, 109

RCPSP-TT Resource-Constrained Project Scheduling Problem with Transfer Time. 29

RUASP Rescue Unit Assignment and Scheduling Problem. 43

SA Simulated Annealing. 37, 38

SDCP Stochastic Debris Clearance Problem. 44

SDDT Step Disaggregated Discrete Time. 34

SDST Sequence-Dependent Setup Times. 23, 47

SPT Shortest Processing Time. 31

SRP-CD Scheduling vehicle Routing Problem to Clean Debris. 43

STT Shortest Setup Time. 31

TS Tabu Search. 37, 38

UB Upper Bound. 62, 95

VNS Variable Neighborhood Search. 39, 41, 42

VRP Vehicle Routing Problem. 31, 41, 42

VRP-RPR Vehicle Routing Problem with Relaxed Priority Rules. 41

Chapter 1

Introduction

The unintentional release of dangerous substances from industrial sources into the environment may occur as a result of fire, explosion or spills in the occurrence of a technological or a natural disaster (Keim, 2011). In despite of regulations in place to prevent such accidents and to minimize their impact (see, *e.g.*, the Seveso-III Directive (2018)), operations after industrial disasters remain a challenging task. This thesis aims to address these challenges by investigating and modeling potential problem scenarios and developing optimization models and techniques. The goal is to provide valuable insights and efficient solutions capable of handling the intricate interplay between scheduling, resource allocation, and risk mitigation. Specifically, the focus lies on scenarios where industrial accidents involving hazardous substances require timely and strategic resource allocation to minimize potential risks to the environment, human well-being, and the economy.

This study was initially motivated by the 2019 Lubrizol factory fire in Rouen (France), in which a chemical plant caught fire, generating a toxic cloud with toxic particles that reached the city center and the north of France (eMARS, 2019). According to Reuters (2019), *"people [were] told not to eat produce from their gardens,"* and farmers were *"not allowed to sell milk, vegetables and other products harvested in the area"* in the days following the accident. It was later confirmed that approximately 5,253 tons of multi-purpose additives, oil, detergent and other chemicals were burned in the fire, raising a generalized concern about the impact of the released substances on human health, the environment and the local economy (Seine-Maritime, 2019).

Unfortunately, the 2019 Lubrizol factory fire accident is not an isolated case. In Europe, on average, about 24 major industrial accidents involving dangerous substances are reported annually (European Commission, 2021). Once an industrial disaster occurs, preliminary on-the-ground information is collected to determine the extent of the accident, and operational decisions need to be made depending on the hazardous nature of the products involved and the extension of the affected area. In some cases, dangerous pollutants can be absorbed by plants and soil or infiltrate the ground, contaminating ground waters. Moreover, it is also possible that volatile products transform into other types of – potentially more harmful – byproducts (Görlitz et al., 2011).

The risk of environmental impacts over the population and the affected areas require the deployment of specialized teams and equipment to remove or neutralize the hazardous pollutants. In addition

to the environmental impact, the risk to human health increases in cases of proximity between industrial and urban areas. Thus, in order to take rational actions, good decision-making models are critical at this point.

Some studies in the literature have tackled related problems from the Operations Research (OR) perspective. For instance, scheduling or integrated scheduling-routing applications have been developed to provide decision-making support for emergency response to natural catastrophes (Wex et al., 2014; Tirkolaee et al., 2020), forest fires (Ren and Tian, 2016; Wu et al., 2019; Wang et al., 2020; Bodaghi et al., 2020), and provide strategical (Duque and Sörensen, 2011; Duque et al., 2016; Sakuraba et al., 2016; Barbalho et al., 2023) and operational (Pena et al., 2023) support after natural disasters. However, to the extent of our knowledge, there are a few works focusing on industrial disasters involving dangerous substances that can cause harmful effects to the environment or community at large from the OR point of view.

In this thesis, we investigate the post-catastrophe operations in the event of industrial disasters, where a group of specialized teams and equipment are deployed to clean up areas contaminated by dangerous substances, where the objective is to mitigate the impact of accidents as quickly and effectively as possible. We explore various optimization techniques commonly employed in the literature to solve complex scheduling problems. The investigation encompasses mathematical formulations, metaheuristics and extensive computational experiments, aiming to provide insights into the interplay between scheduling, resource allocation, and risk management. In this scenario, the whole contaminated area is mapped as a connected graph, where each contaminated site is represented as a node with an associated risk, which depends on the hazardous nature of the pollutants present on it. Teams can be deployed in the area covered by the graph by traversing secured paths to clean or to neutralize the dangerous products present on each node.

In this context, we provide an introduction to the core concepts and notations pertaining to scheduling problems, laying the foundation for subsequent discussions. In addition, we analyze the Resource-Constrained Project Scheduling Problem (RCPSP) as a baseline for scheduling tasks under resource constraints. Despite RCPSP's efficacy, it becomes evident that its generality might not encapsulate all the nuances of real-world scheduling challenges. Hence, common extensions and variations of the RCPSP in the literature are also explored.

In Chapter 3, we investigate the Resource-Constrained Project Scheduling Problem with Risk and Priorities (RCPSP-RP), a specialized model that adds a layer of complexity by considering risk factors associated with tasks. The objective is to develop an understanding of how risk influences scheduling decisions and to propose optimization approaches that efficiently balance scheduling objectives while mitigating potential risks. In this approach, it is assumed that product amounts remain constant over time, overlooking the dynamic nature of chemical transformations. This allows for solutions that can be computed efficiently while still accounting for essential risk factors. However, the complexities arising from the chemical kinetics involved in the natural transformations of products released into the environment introduce challenges that impact the feasibility of the RCPSP-RP model.

To extend the RCPSP-RP, in Chapter 4, we introduce the Resource-Constrained Project Scheduling Problem with Risk and Product Transformation Dynamics (RCPSP-RTD). The RCPSP-RTD is a specification of the RCPSP-RP that focuses on the intricate interplay between OR and Chemical Kinetics as a model for optimizing strategies and mitigating risks, considering both resource allocation and the dynamic transformations of products. By expanding the RCPSP-RP to encompass

the dynamics of product transformations, the RCPSP-RTD strives to enhance the applicability of risk management solutions in situations involving the release of hazardous substances.

1.1 Thesis structure

This thesis is composed of five chapters. In the first chapter, we provide the general context. The second chapter is dedicated to a bibliographical review of the scientific literature related to this thesis. In the third chapter, we introduce the RCPSP-RP, and in the fourth chapter, we investigate the RCPSP-RTD, an extension of the RCPSP-RP that models the dynamic transformation of products. In the final chapter, we present the general conclusions of this thesis by summarizing the contributions and suggesting directions for future research. The main contributions, classified by chapters, are presented as follows.

In Chapter 2, a general introduction to scheduling problems is provided with a review of the classic scheduling notation $\alpha|\beta|\gamma$ for the machine environment, characteristics of tasks and optimality criteria. A dedicated review to the RCPSP is also presented, with some extensions and variants commonly found in the literature, strong time-indexed formulations, exact and metaheuristics methods. Finally, some relevant scheduling and/or integrated scheduling routing studies in the post-disaster relief literature are reviewed.

In Chapter 3, the RCPSP-RP is introduced and analyzed. In this context, the polluted nodes in the graph are mapped as tasks with associated static risk and processing time attributes. Resources represent the specialized teams that are deployed from a specific node in the graph (called the depot) to perform the cleaning operation and mitigate risk by processing all mapped tasks. We consider that travel times when traversing the edges are implied based on their traveling distance. More specifically, these travel times are treated as Sequence-Dependent Setup Times (SDST). In addition, the RCPSP-RP performs task prioritization by employing priority constraints between tasks based on their risk attribute. As a consequence, tasks with higher risk take precedence over lower risk ones. In order to gain insights on the impact that task prioritization may have on the quality of solutions, we follow a relaxation schema inspired by Panchamgam et al. (2013) by introducing a relaxation threshold that partially or completely relaxes priority constraints according to the stated level of strictness. Thus, providing strict, moderate and no priority policies. We propose mathematical formulations inspired by an exact formulation for the classical RCPSP from Christofides et al. (1987). Finally, an Iterated Local Search (ILS) metaheuristic is also developed to cope with large problem instances, when optimal – or even integer – solutions cannot be computed by the exact methods in reasonable time. We perform experiments on a set of theoretical instances to investigate different problem scenarios and the three types of priority policies.

In Chapter 4, the RCPSP-RTD, an extension to the RCPSP-RP is investigated, where we consider that products may naturally transform into other products over time, at a specific rate; and the resulting product can be more or less risky than their parent product. The resulting products may also naturally degrade over time, which means their presence can vanish or disperse over time. The objective remains to schedule a sequence of on-site operations involving a set of heterogeneous teams to mitigate (by cleaning or neutralizing) the hazardous products present on contaminated sites. We consider that each contaminated site contains an initial presence of one or more dangerous products and, for every product, a numerical risk is associated corresponding to its hazardous nature. The more dangerous the product is, the higher the risk to to the environment and human health the

product will be. We propose a mathematical formulation and an ILS metaheuristic to cope with large problem instances. Several computational experiments are performed in order to analyses different problem scenarios and transformation rates parameters.

In Chapter 5, we provide some perspectives and future works. For instance, further exploration into realistic models that encompass the complete dynamics of chemical transformations is envisaged. This involves developing more accurate ways to account for the transformation dynamic processes of substances and their subsequent impact on risk levels. In addition, the interaction between scheduling decisions and these dynamic processes presents a rich area for investigation. Optimizing scheduling strategies while considering the time-dependent transformations of substances can lead to even more effective risk mitigation plans.

Chapter 2

Literature review

In this chapter, we provide an introduction to fundamental concepts and notations used in scheduling problems, including a dedicated section for the RCPSP, outlining extensions and commonly encountered applications in existing literature. In addition, we provide an overview that highlights the similarities and differences between the RCPSP-RP (introduced in Chapter 3), the RCPSP-RTD (presented in Chapter 4) and other relevant scheduling and integrated scheduling-routing problems and their applications documented in the literature.

2.1 Introduction to scheduling problems and notations

Scheduling problems are a family of optimization problems in which a set of tasks or activities need to be sorted for execution in a way that minimizes or maximizes a certain objective or cost function. It is assumed that each task can have its own requirements and constraints necessary to its execution. The objective function can be different depending on the context, such as minimizing the total time required to complete all tasks (makespan), minimizing the number of workforce required, maximizing the utilization of resources or minimizing the total cost. Some examples of scheduling problems may include:

- Workforce scheduling: determining the optimal schedule for a group of employees to complete a set of tasks while minimizing costs and meeting constraints such as availability, shift duration and skill levels.
- Production scheduling: determining the sequence of orders and production times to optimize production capacity and minimize costs while meeting delivery deadlines.
- Transportation scheduling: planning the optimal routes and schedules for a fleet of vehicles to deliver goods or services, while minimizing costs, reducing travel time, and meeting customer demand.
- Project scheduling: determining the optimal sequence and duration of tasks necessary to complete a project while minimizing costs, reducing time, and meeting resource constraints.

Many scheduling problems are known to be NP-complete, which usually limits the problem size that exact algorithms can handle in a practical time span. Often, heuristic and approximation algorithms are used to find good solutions to medium and large scale instances. In the scheduling literature, some largely used methods to solve scheduling problems include (see Section 2.2.2):

- Greedy algorithms are algorithms that perform locally optimal decisions at each step, but may not necessarily result in a globally optimal solution.
- Genetic algorithms uses principles of natural selection (genetic material, crossover, mutation, etc) to iteratively evolve (optimize) a population of solutions.
- Dynamic programming breaks down a complex problem into smaller sub-problems, solving each one and combining the solutions to either find optimal solutions or calculate lower bounds.
- Mathematical programming involves formulating and solving a mathematical optimization problem such as a linear programming or a integer programming model to either find optimal solutions or calculate lower bounds.

Scheduling problems are often described following the notation $\alpha|\beta|\gamma$ proposed in Graham et al. (1979). This notation refers to the set of symbols or representations used to express the machine environment (α), the characteristics of tasks (β) and the optimality criteria (γ) of scheduling problems.

2.1.1 Machine environment

The notation for the machine environment (α) describes the set of characteristics and constraints associated with the machines used for processing/performing tasks. This notation provides critical information for determining the feasibility of scheduling tasks to specific machines. According to Graham et al. (1979), the machine environment can be separated in two subfields $\alpha = \alpha_1\alpha_2$. Depending on the value of α_1, we may express the following machine characteristics.

If $\alpha_1 \in \{\circ, I, Q, R\}$, then each task J_j consists of a single operation that can be processed on any machine M_i and the processing time of task J_j on machine M_i is denoted by p_{ij}. The values of α_1 correspond to the following machine environments:

- *Single machine* ($\alpha_1 = \circ$): $p_{1j} = p_j$.
- *Identical parallel machines* ($\alpha_1 = I$): $p_{ij} = p_j$ for any machine M_i, i.e. the processing time of task j is the same for every machine.
- *Uniform parallel machines* ($\alpha_1 = Q$): $p_{ij} = p_j/e_i$ where e_i is the processing speed of machine M_i.
- *Unrelated parallel machines* ($\alpha_1 = R$): $p_{ij} = p_j/e_{ij}$, where e_{ij} is the processing speed of task J_j on machine M_i.

On the other hand, if $\alpha_1 \in \{F, J, O\}$, the machine environment corresponds to the class of multi-stage tasks. In this class of problem, each task J_j consists of a sequence of operations $(O_{1j}, ..., O_{mj})$ that should be executed on different machines M_i. Operations belonging to the same task cannot be executed simultaneously. There are three main types of shop scheduling:

- *Flow-shop* ($\alpha_1 = F$): each task J_j consists of m operations performed on different machines M_i during p_{ij} time units.
- *Job-shop* ($\alpha_1 = J$): similar to flow-shop, but the number of operations is not necessarily the same for every task, and each task may have its own order of execution on the machines.
- *Open-shop* ($\alpha_1 = O$): Similar to job-shop, however, the order of execution can vary freely.

The field α_2 expresses the number of machines or resources available. If α_2 is a positive integer, then $m = \alpha_2$; if $\alpha_2 = \circ$, then m is assumed to be any arbitrary number.

2.1.2 Characteristics of tasks

The field β expresses the characteristics of tasks, providing details about the features of individual tasks to be scheduled. According to Graham et al. (1979), the task characteristics can be separated into four subfields $\beta = \beta_1\beta_2\beta_3\beta_4$, which denote:

- *Preemption* ($\beta_1 \in \{pmnt, \circ\}$) states that the processing of any on-going operation may be interrupted and resumed at a later time unit, such that: $\beta_1 = pmnt$ if preemption is allowed; $\beta_1 = \circ$ otherwise.
- *Resource constraints* ($\beta_2 \in \{res1, res, \circ\}$) represents the number of resources present, such that:

  $\beta_2 = res1$: there is only a single resource.

  $\beta_2 = res$: there are y limited resources R_h, $(h = 1, ..., y)$, with the property that each task J_j requires the use of r_{hj} units of R_h to execute.

  $\beta_2 = \circ$: no resource constraints are specified.
- *Precedence relations* ($\beta_3 \in \{prec, tree, \circ\}$).

  $\beta_3 = prec$: a precedence relation $\prec$ between the tasks is specified. It is derived from a directed acyclic graph G with vertex set $\{1, ..., n\}$ for some integer n. If G contains a directed path from j to k, we write $J_j \prec J_k$ and require that J_i is completed before J_k start.

  $\beta_3 = tree$: G is a rooted tree with either outdegree at most one for each vertex or indegree at most one for each vertex.

  $\beta_2 = \circ$: no precedence relation is specified.
- *Release dates* ($\beta_4 \in \{u_j, \circ\}$).

  $\beta_4 = u_j$: release dates are specified per task, *i.e.* the earliest possible start time of task j (ES_j).

  $\beta_4 = \circ$: we assume that all tasks are available at time 0, *i.e.* $u_j = 0$.

2.1.3 Optimality criteria

Optimality criteria notation (γ) is used to define the objective function of a scheduling problem. This notation allows a clear definition of the goals of scheduling problems, making it easier to

evaluate the effectiveness of different scheduling algorithms and methods. For instance, given a schedule, we can compute for each task J_i:

- The completion time C_j.
- The lateness $L_j = C_j - p_j$ or tardiness $T_j = \max\{0, C_j - p_j\}$, where p_j is the processing time of task j.
- The unit penalty $U_i = 0$ if $C_j \leq p_j$, 1 otherwise.

The field $\gamma \in \{f_{\max}, \sum f_j\}$ defines the optimality criterion chosen by integrating the above variables in the form of a maximum function or a sum of functions, possibly weighted by some constant. For example:

- The completion time of the last task or makespan is the function $C_{\max} = \max_{1 \leq j \leq n}\{C_j\}$. The makespan can be also be interpreted as the duration of the scheduling.
- The weighted sum of completion times $\sum_{j=1}^{n} w_j C_j$ or the weighted sum of task lateness $\sum_{j=1}^{n} w_j L_j$, for some integer n representing the number of tasks.

2.2 The resource-constrained project scheduling problem

The RCPSP, described in Pritsker et al. (1969), considers a project with J tasks labeled $j = 1, ..., J$ that should be scheduled on limited renewable resources such that the makespan is minimized. Each task requires r_{jk} units of resource $k \in \mathcal{R}$ in every time instance of its duration p_j to complete. In the classical RCPSP, task preemption is not allowed and each resource $k \in \mathcal{R}$ has a limited capacity of R_k at any point in time. The parameters p_j, r_{jk}, and R_k are assumed to be deterministic non-negative integers. Due to technological requirements, there are acyclic precedence relations between some of the tasks given by sets of immediate predecessors P_j stating that a task j may only start after all its predecessors $i \in P_j$ are completed. Usually, two dummy tasks $j = 0$ and $j = J + 1$ with instant duration and no demand of resources represent the start and the completion of the project, respectively.

The RCPSP is a generalization of the Job Shop Scheduling Problem (Graham et al., 1979), which is a known NP-hard problem (Blazewicz et al., 1983). We review below some of the most commons RCPSP variants in the literature, as surveyed in Kolisch (1995), Demeulemeester and Herroelen (2002), Hartmann and Briskorn (2010) and more recently, Hartmann and Briskorn (2022).

2.2.1 Problem variants and extensions

While the RCPSP described earlier is a flexible model, it might not encompass all scheduling scenarios and constraints encountered in real-world applications. Therefore, many researchers have developed more general project scheduling problems or extensions that are suitable for specific applications. These works are often developed using the standard RCPSP as a starting point, with additional restrictions on the machine environment, task characteristics or optimality criteria.

The book Schwindt and Zimmermann (2015) covers many important models and methods for deterministic, stochastic and robust project scheduling. Since the 1990s, several survey papers on project scheduling have been published. The survey papers Brucker et al. (1999), Brucker (2002), Tavares (2002), Herroelen et al. (1998), Herroelen and Leus (2005), Kolisch and Padman (2001),

Hartmann and Briskorn (2010), Hartmann and Briskorn (2022) provide a broad overview over variants and extensions of the RCPSP that have been proposed in the literature.

Preemptive scheduling

The classical RCPSP requires tasks to be processed in a non-preemptively fashion. That is, once a task starts, it cannot be interrupted and must be processed continuously until it is completed (p_i is usually utilized to denote the processing time of task i). In this model, task completion times can be directly computed in function of their start times:

$$C_i = S_i + p_i \tag{2.1}$$

Some studies have extended the RCPSP to allow task preemption under certain circumstances. For example, Shou et al. (2015) presents an optimization method that considers at most one interruption of each task. Afshar-Nadjafi and Majlesi (2014), Vanhoucke and Coelho (2019) investigate multiple interruptions at integer points, but setup times (see below) are employed when resuming interrupted tasks. In Kreter et al. (2015, 2016) task interruptions are allowed due to calendar restriction on weekends and holidays.

Setup times

Setup times are defined as the time intervals required prior to the processing of tasks. In practice, a setup time can be seen as the time delay needed to adjust the settings and/or transportation of a unit of resource, so that it is ready to process a task (see, *e.g* Allahverdi et al. (2008) for a survey on scheduling problems with setup times).

Setup times can be resource dependent or sequence dependent. It is sequence-dependent if its duration depends on the last processed and the next to be processed task by the same unit of resource, and is sequence-independent if its duration depends solely on the specific type of resource and the current task to be processed.

An interesting use case of sequence-dependent setup times can be found in Hanzálek and Šůcha (2017), where authors assume that, if two tasks i and j are consecutively processed by the same unit of resource, then between the completion time of i (C_i) and the start time of j (S_j), this unit requires a setup of s_{ij} time units:

$$C_i + s_{ij} \leq S_j \tag{2.2}$$

A variant of this concept is the sequence-dependent transfer times s_{ijk}, which brings routing aspects into the RCPSP. The Resource-Constrained Project Scheduling Problem with Transfer Time (RCPSP-TT) studied in Quilliot and Toussaint (2012), Poppenborg and Knust (2016), and Kadri and Boctor (2018) is an extension that accounts that a resource k has to be transported from the location of task i to the one of task j, with the implication that resource k remains unavailable during its transport.

An extension similar to the RCPSP-TT can be found in Lacomme et al. (2019), which assumes that resources must be physically transferred from task to task in order to process them. To this end, a

limited fleet of vehicles with restricted capacity must be routed to pick up resource units and deliver them to their assigned tasks. Transportation times can be seen as sequence-dependent setup times that depend on the routing between two consecutive tasks processed by the same resource unit.

Precedence constraints

Precedence constraints are defined as a set of precedence relations between tasks stating that, if a relation (i, j) with $i \neq j$ exists, it is required that j only start if i has been completed (Lenstra and Rinnooy Kan, 1978; Möhring et al., 2004).

The ordered pairs (i, j) of precedence relations can be used to construct an acyclic digraph $D = (V, P)$, where each node corresponds to a task and each arc represents a precedence relation in the set of precedence constraints P(Möhring et al., 2004). Thus, the precedence constraints can be written as:

$$S_i + p_i \leq S_j \quad \forall (i, j) \in P \tag{2.3}$$

Where p_i is the processing time of task i, while S_i and S_j are the starting times of tasks i and j, respectively.

In the literature, some RCPSP applications have employed precedence constraints to model operational dependence between tasks (see, e.g., Sakuraba et al. (2016); Barbalho et al. (2023)); to perform task prioritization (see, e.g., Ren and Tian (2016); Wu et al. (2019); Wang et al. (2020)) and to model complex interactions between tasks and resources (see, *e.g.*, Hartmann and Briskorn (2022)).

Minimal and maximal time lags

Precedence constraints can be extended by allowing minimal or maximal time lags between any pair (i, j) of precedence-related tasks. These time lags can be employed in any of the following ways: between (i) the starting times of both tasks; (ii) the starting time of i and the completion time of j; (iii) the completion time of i and the starting time of j; or (iv) the completion times of both tasks. Each of these types can be transformed into each other type if processing times are known in advance.

For instance, time lag constraints between the starting times of consecutive tasks can be expressed in the form (Hurink and Keuchel (2001)):

$$S_i + l_{ij} \leq S_j \quad \forall (i, j) \in P \tag{2.4}$$

Where l_{ij} is the minimal or maximal time lag. If $l_{ij} \geq 0$, then task j cannot start earlier than l_{ij} time units after the starting time of task i (minimal time lag). On the other hand, if $l_{ij} < 0$, then task j cannot start earlier than $|l_{ij}|$ time units before the starting time of task i (maximal time lag). Expressing $l_{ij} = p_i$ is equivalent to the classical precedence constraints (2.3), while stating $l_{ij} = 0$, may allow i and j to start at the same time if there are enough available resources.

A large number of works in the literature have considered the RCPSP with minimal and/or maximal time lags, referred as RCPSP/max (see, *e.g.*, Hartmann and Briskorn (2022)).

Priority rules

In the scheduling literature, priority rules (or dispatching rules) are a technique by which an attribute π_i is assigned to each waiting task according to some method or metric. The scheduling of tasks is then performed in such a way that the waiting tasks with smaller attribute values are selected before the tasks with higher attribute values (Panwalkar and Iskander, 1977).

Dispatching rules can be classified into static, dynamic, local or global. Static rules are the ones in which the task priority values do not change as a function of time, the opposite behavior is classified as dynamic rules. Local rules are the ones that require information only about those tasks that are waiting, while global rules require additional information about other tasks and resources (Panwalkar and Iskander, 1977; Kolisch, 1996). For example, the following rules are often employed as dispatching rules: Shortest or Longest Processing Time (respectively SPT or LPT); Earliest Due Date (EDD), Shortest Setup Time (STT); First In, First Out (FIFO); Last In, First Out (LIFO), etc. See, *e. g.,* Panwalkar and Iskander (1977), Kolisch (1996) and Sels et al. (2012) for extensive reviews on priority rules in the scheduling/sequencing literature.

An interesting example of task prioritization is the d-relaxed priority rule, introduced in Panchamgam et al. (2013) and also examined in Doan et al. (2021); Hà et al. (2022) within the context of the Vehicle Routing Problem (VRP) involving a single vehicle. In scenarios where certain locations require more immediate attention than others, this method establishes discrete priority groups $\Pi = 1, 2, ..., n$, for some integer $n \geq 1$, and assigns each demand node a priority group based on its urgency $\pi_i \in \Pi$. In this framework, node i is considered higher priority than node j if $\pi_i < \pi_j$. The common requirement is to visit all nodes within a given priority group prior to moving on to the subsequent one. Before departing from the depot, an integer threshold $d \geq 0$ can be set to define the degree of priority strictness the vehicle will adhere to. This means, depending on d, the vehicle may visit groups $\pi_i + 1, ..., \pi_i + d$ before completing its visits to all nodes in priority group π_i, if it results in a more efficient route. This innovative approach provides flexibility in adhering to the priority structure, allowing for adaptability based on the specified threshold d.

The aforementioned d–relaxed priority rule is an interesting prioritization technique because it provides flexible decision support according to the emergency level. Depending on the defined threshold, the vehicle will visit nodes in a a) strict order of priorities; or may b) ignore some or c) ignore all of them, if it leads to a shorter distance tour. The aforementioned concept can be applied to scheduling applications by employing precedence constraints with a relaxation scheme that, analogously, relax constraints according to the level of strictness. Thus, the precedence relations among demand nodes can be obtained as:

$$P = \{(i, j) \in V \times V \mid \pi_i < \pi_j - d\} \tag{2.5}$$

In Chapter 3, we will use Inequalities 2.4 and the concept of the d-relaxed priority rules to implement priority constraints, which are composed of precedence constraints with zero minimal time lag:

$$S_i \leq S_j \quad \forall (i, j) \in P \tag{2.6}$$

2.2.2 Optimization methods

Optimization methods encompass a range of mathematical and algorithmic strategies that are deployed to identify the most optimal solution among a set of potential solutions for a specific problem. These techniques are used to minimize or maximize an objective function subject to a set of constraints.

In general, methods for solving the RCPSP involve exact algorithms such as those proposed in Brucker et al. (1999), Demeulemeester and Herroelen (1992), Demeulemeester and Herroelen (1997), Mingozzi et al. (1998), Sprecher (2000), Demeulemeester and Herroelen (1997), Sprecher (2000); as well as heuristics and metaheuristics such as those in Kochetov and Stolyar (2003), Kolisch and Hartmann (2006), Mendes et al. (2009), Palpant et al. (2004), Tormos and Lova (2001), Kochetov and Stolyar (2003) and Mendes et al. (2009). We describe below some of the methods used in the aforementioned works.

- Mixed Integer Linear Programming (MILP): this method is used to solve combinatorial optimization problems by modeling problems as a minimization or maximization of a linear function subject to several constraints written as linear equations or inequalities. In MILP formulations, some or all the variables are constrained to be integers.
- Branch-and-bound algorithms: these methods involve partitioning the solution space into smaller subproblems and systematically exploring each subproblem until the optimal solution is found.
- Heuristics and metaheuristics: these methods are designed to tackle optimization problems that are hard or even impossible to solve with exact methods. They usually employ high level strategies to either approximate optimal values or explore the solution space in search for optimal solutions. Examples of metaheuristics include genetic algoritmhs, simulated annealing, ant colony systems, local search algorithms, tabu search, etc.

In addition, a benchmark test set of instances have been proposed by Kolisch and Sprecher (1997). This benchmark has been used by many researchers, and it is available from the Project Scheduling Problem Library (PSPLIB). See Kolisch and Hartmann (2006) for a comparative among heuristics and metaheuristics tested against the PSPLIB.

In the subsequent part of this section, we will present a selection of optimization methods that are frequently encountered in the literature. These methods are employed to effectively address the RCPSP.

Time-indexed mixed integer linear programming models

MILP models are often employed in project scheduling problems due to their high efficiency in solving small to medium instance problems. Typically, those formulations may be classified into three main categories: (i) time-indexed formulations; (ii) task sequencing or resource flow formulations and (iii) start and end formulations. Usually, the use of time-indexed variables leads to large models, but provides better lower bounds than other categories of MILP formulations (Sousa and Wolsey, 1992). Below, we present two families of strong time-indexed formulations for the non-preemptive project scheduling problem extracted from Artigues (2017).

Pulse formulations. The family of time-indexed formulations using pulse variables involves a

pseudo-polynomial number of variables and constraints. The concept of pulse variables is that they represent the exact time at which a task starts, *i.e.*, $S_i = t$ if task i starts at t. To this end, pulse binary variables $x_{it}, i \in V, t \in H$ are employed, such that $x_{it} = 1$ if, and only if, task i starts at period t. H is defined as the set of time units. We assume that, if a task i starts at t, it stay in process during the interval $[t, t + p_i]$. A strong formulation based on pulse variables based on a disaggregated way of modeling the precedence constraints was proposed by Christofides et al. (1987). This formulation, described below, is called the Pulse Disaggregated Discrete Time (PDDT) formulation. The PDDT uses the problem input data summarized in Table 2.1.

Input parameter	Description
V	Set of tasks
P	Set of precedence constraints
H	Set of time units
w_i	Weight of task i
B_k	Number of available resources of type k
b_{ik}	Number of resources of type k required to process task i
p_i	Processing time of task i
$\mathcal{R}$	Set of resources
ES_i	Release date or earliest possible start time of task i
LS_i	Latest possible start time of task i

Table 2.1: General problem input data and description.

$$\min \sum_{i \in V} w_i C_i \qquad \text{s.t.} \tag{2.7}$$

$$\sum_{\tau=0}^{t-p_i} x_{i\tau} - \sum_{\tau=0}^{t} x_{j\tau} \geq 0 \qquad \forall (i,t) \in P, t \in H \tag{2.8}$$

$$\sum_{i \in V} \sum_{\tau=t-p_i+1}^{t} b_{ik} x_{i\tau} \leq B_k \qquad \forall t \in H, k \in \mathcal{R} \tag{2.9}$$

$$\sum_{t \in H} x_{it} = 1 \qquad \forall i \in V \tag{2.10}$$

$$C_i = \sum_{t \in H} (t + p_i) x_{it} \qquad \forall i \in V \tag{2.11}$$

$$x_{it} = 0 \qquad \forall i \in V, t \in H \setminus [ES_i, LS_i] \tag{2.12}$$

$$x_{it} \in \{0, 1\} \qquad \forall i \in V, t \in H \tag{2.13}$$

$$C_i \geq 0 \qquad \forall i \in V \tag{2.14}$$

The objective function (2.7) minimizes the weighted sum of completion times, observing that, according to the definition of the pulse variables, $S_i = \sum_{t \in H} t x_{it}$. The inequalities (2.8) are the

precedence constraints and the inequalities (2.9) represent the resource constraints. The constraints (2.10) state that each activity has to be started exactly once in the scheduling horizon. Task completion times are calculated by the constraints (2.11). The constraints (2.12) forbid that tasks start outside the allowed time interval $[ES_i, LS_i]$. The decision variables are defined in (2.13) and (2.14).

Step formulations. Another family of strong time-indexed formulation is known as the Step Disaggregated Discrete Time (SDDT) formulation from Klein (1999), which is a variant of the PDDT. This formulation is based on binary step variables z_{it} such that $z_{it} = 1$ if, and only if, task i either starts at time t or has started before t. That is, for a given task, variables z_{it} with $t < S_i$ are all equal to 0, while the variables with $t \geq S_i$ are all equal to one. With these definitions, the start time can be expressed as:

$$S_i = \sum_{t \in H} t(z_{it} - z_{,t-1}) \qquad \forall i \in V \tag{2.15}$$

The SDDT formulation can be expressed as (see Table 2.1 for a summary of the problem input data):

$$\min \sum_{i \in V} w_i C_i \qquad \text{s.t.} \tag{2.16}$$

$$z_{i,t-p_i} - z_{jt} \geq 0 \qquad \forall (i,j) \in P, t \in H \tag{2.17}$$

$$\sum_{i \in V} b_{ik}(z_{it} - z_{i,t-p_i}) \leq B_k \qquad \forall t \in H, k \in \mathcal{R} \tag{2.18}$$

$$z_{i,LS_i} = 1 \qquad \forall i \in V \tag{2.19}$$

$$z_{it} - z_{i,t-1} \geq 0 \qquad \forall i \in V, t \in H \tag{2.20}$$

$$z_{it} = 0 \qquad \forall i \in V, t < ES_i \tag{2.21}$$

$$C_i = \sum_{t \in H} (t + p_i)(z_{it} - z_{i,t-1}) \qquad \forall i \in V \tag{2.22}$$

$$z_{it} \in \{0, 1\} \qquad \forall i \in V, t \in H \tag{2.23}$$

$$C_i \geq 0 \qquad \forall i \in V \tag{2.24}$$

The objective function (2.16) minimizes the weighted sum of completion times. The precedence constraints (2.17) state that if a task j has started at time t or before (i.e. $z_{jt} = 1$), then task i has started at time $t - p_i$ or before. The inequalities (2.18) are the resource constraints. It is assumed that a task is in process at time t if, and only if, $z_{it} - z_{i,t-pi} = 1$. The constraints (2.19) state that each task has to be started at or before its latest start time LS_i. The constraints (2.19) and (2.20) define the step function, *i. e.* once $z_{it} = 1$ for a given t, it cannot be zero for any other $t' > t$. Note that these constraints also force to 1 all variables z_{it} with $t \geq LS_i$. The constraints (2.21) forbid that tasks start before their earliest start times ES_i. Task completion times are computed by the constraints (2.22). The decision variables are defined in (2.23) and (2.24).

Branch-and-bound algorithms

Branch-and-bound algorithms are a class of algorithms used in optimization and search problems. They work by breaking a problem into smaller sub-problems and creating a tree-like structure to represent the sub-problems. The algorithm explores the nodes of the tree in a particular order, and at each node, it generates one or more child nodes that represent the sub-problems that can be created by further dividing the current problem. The algorithm keeps track of the best solution found so far and uses this information to prune the search tree. By eliminating branches that are guaranteed not to yield an optimal solution, the algorithm can greatly reduce the search space and find the optimal solution more quickly. Branch-and-bound algorithms are commonly used in combinatorial optimization problems, such as the traveling salesman problem, the knapsack problem, and the graph coloring problem. In the RCPSP literature, efficient branch-and-bounds designs can be found, for example:

- In Demeulemeester and Herroelen (1992), a branch-and-bound approach for scheduling PERT/CPM[1] projects is presented, considering precedence and resource constraints to minimize project duration. The depth-first solution strategy employs nodes representing feasible partial schedules, utilizing exhaustive combinations of activities to resolve resource conflicts.
- Furthermore, Demeulemeester and Herroelen (1997) explores insights gained from computational results using an enhanced version of Demeulemeester and Herroelen (1992) branch-and-bound procedure. The study investigates the impact of variables such as addressable memory, search strategy (depth-first, best-first, or hybrid), and a stronger lower bound on computation time.
- Another branch-and-bound algorithm is introduced in Sprecher (2000). The concept is adaptable to various settings such as multimode problems, time-varying resource availability, and diverse objectives. Despite its generality, the algorithm competes, in computational performance, with the best single-mode problem approaches in the previous literature.
- In Sousa and Wolsey (1992), a non-preemptive single machine scheduling problems using time-indexed variables is studied, resulting in extensive models but improved lower bounds compared to other MILP formulations. The study introduces valid inequalities, emphasizing the role of constraint aggregation with generalized upper bound constraints. Implementation of a branch-and-bound algorithm based on these insights is discussed, with computational results on small problems (20–30 tasks) and various constraints and objective functions.

Metaheuristics

Metaheuristics are largely used to solve optimization problems, since they can provide good solutions to problem instances that are difficult to solve analytically. In the context of scheduling problems, metaheuristics have been shown to be efficient for finding optimal or near-optimal solutions. For instance, the studies of Asif Raza and Mustafa Al-Turki (2007) and Das and Acharyya (2011) (see further below) compared the performance of different metaheuristics in the context of scheduling problems, showing that they were able to find solutions closer to the optimal ones than traditional heuristic methods.

[1] PERT/CPM (Program Evaluation and Review Technique/Critical Path Method) are project management methodologies that aid in planning, scheduling, and controlling projects by identifying critical paths and estimating activity durations.

Metaheuristics are one of the most popular families of optimization methods applied to solve the RCPSP and its extensions. We describe below some of the most common metaheuristics in the literature that were successfully tested against the PSPLIB.

Genetic algorithms. A Genetic Algorithm (GA) is a class of algorithm that tries to emulate the biological evolutionary process of natural selection and genetic crossover to find the best solution to a problem. GAs use a computational approach for solving optimization problems by iteratively optimizing a population of potential solutions through selection, crossover (recombination), and mutation. In each iteration, the fittest solutions are selected for reproduction and their genetic material (parameters) are combined to create new offspring solutions. This process continues until an optimal solution is found or a stopping criterion is met. GAs are one of the most employed metaheuristic for solving the classical RCPSP (Kolisch and Hartmann, 2006). Relevant works in GAs for the RCPSP can found in, for example:

- Alcaraz and Maroto (2001) addressed the scarcity of studies on GAs for resource allocation in project scheduling and presented a genetic algorithm for the single-mode RCPSP, introducing a new solution representation based on the standard activity list. The study developed effective crossover techniques with extensive computational experiments using standard project instances, the proposed GA demonstrated superior performance compared to existing algorithms in the literature.

- Mendes et al. (2009) introduced a GA utilizing a chromosome representation based on random keys. The schedule construction employs a heuristic priority rule, where activity priorities are determined by the GA, resulting in parameterized active schedules. The approach was tested on standard instances from the literature and compared with other methods, demonstrating the effectiveness of the proposed algorithm through favorable computational results.

- Hartmann (2002) addressed the classical RCPSP, aiming to minimize project makespan. A novel heuristic, the self-adapting GA, is proposed. Utilizing the activity list representation, the algorithm employs two decoding procedures, and an additional gene determines their usage. This self-adaptation allows the algorithm to learn and optimize its own decoding strategy based on the problem instance. Computational experiments demonstrated the effectiveness of this self-adaptive mechanism, showcasing the proposed heuristic as one of the top-performing approaches for the RCPSP.

- Hindi et al. (2002) introduced an algorithmic framework involving a Memetic Algorithm[2] with novel heuristics and multiple local search methods. The proposed methods were tested, demonstrating significant improvements over traditional GAs and state-of-the-art algorithms in terms of solution quality and computational efficiency, especially for large-scale problems.

- Toklu (2002) developed a GA applicable to both constrained and unconstrained projects. The algorithm employs chromosomes representing activity start days, introduces mathematical operators, and emphasizes a fine mutation operator. A generalized fitness function evaluation was conducted, and the algorithm was applied to a simplified bridge construction problem. The results and parameter effects were discussed.

[2]A Memetic Algorithm is a hybrid evolutionary optimization approach that combines genetic algorithms with local search heuristics.

- Valls et al. (2008) presented a hybrid GA, introducing specific changes in the genetic algorithm paradigm. These changes included a problem-specific crossover operator, a universal local improvement operator for all schedules, a novel parent selection method, and a two-phase strategy involving a restart from a neighbor's population in the second phase. Computational results demonstrated this approach surpassed previous known state-of-the-art algorithms for the RCPSP.

Tabu Search. Tabu Search (TS) is an optimization algorithm used to solve complex problems that explore the space of possible solutions by employing local search methods to move towards optimal solutions. TS algorithms are particularly useful when the solution space is large and complex, making it difficult to find the optimal solutions using traditional exact methods and algorithms. The algorithm operates based on the move generator, evaluator, and acceptance criterion. The algorithm works by maintaining a 'tabu list' of previously visited solutions and avoiding them in future searches. This helps the algorithm escape local maxima and explore a larger portion of the solution space. In the RCPSP literature, efficient works such as have been proposed with competitive results for the PSPLIB, such as:

- Artigues et al. (2003) introduced a flow network model for both static and dynamic RCPSP based on a polynomial insertion algorithm. The algorithm efficiently inserts a new activity into an existing solution represented by an activity-on-node-flow network. For the dynamic context, a rescheduling algorithm is developed to handle unexpected activity occurrences. Two heuristics for static RCPSP, incorporating the insertion algorithm, are presented: a constructive method and a TS approach. Computational results on benchmark instances demonstrated the efficiency of this approach compared to metaheuristics presented in previous studies. The work provided experimental results, marking a significant contribution to solving the static RCPSP.
- Ribeiro et al. (2002) extended the RCPSP formulation to handle various complex constraints and objective functions, such as selectable modes for activities, variable amounts of renewable resources, setup activities, and intricate objective functions. A TS heuristic algorithm was developed, featuring innovations in solution representation and neighborhood construction. Computational testing on RCPSP benchmarks and real-world instances demonstrated the effectiveness of the approach, producing improved solutions compared to existing methods and highlighting its practical utility.
- Thomas and Salhi (1998) presented a TS implementation that incorporates defined move strategies, a structured neighborhood, appropriate tabu status and tenure, and integrates objective function approximation for accelerated search. Informed by a solid understanding of the problem, the methodology employs diversification, intensification, and strategic oscillation to handle infeasibility. Computational tests were performed on existing literature instances and on generated instances. Optimal solutions and lower bounds were compared to existing results in the literature.

Simulated Annealing. Simulated Annealing (SA) is a type of metaheuristic algorithm inspired by the annealing process used in metallurgy to achieve optimal crystalline structures. SA algorithms use a randomized approach to find the global minimum or maximum of a given function by iteratively modifying a potential solution. These modifications are done by swapping or changing

the values of the variables in the solution. The new proposed solution is then evaluated using a cost function. If the new solution is better than the previous solution, it is accepted, and the process continues. However, if the new solution is worse, the algorithm will accept it with a certain probability based on temperature (initially high and gradually decreased) that allows it to escape local minima and explore the solution space. SA algorithms have been successfully used for solving the RCPSP, for example:

- Bouleimen and Lecocq (2003) introduced SA algorithms to both classical RCPSP and its multiple-mode version, with the objective of minimizing the makespan. The conventional SA search scheme is replaced by designs tailored to the specific solution space of project scheduling problems. For the RCPSP, an alternated activity and time incrementing process is used, with parameters set after preliminary statistical experiments. For the RCPSP with multiple modes, a unique approach employs two embedded search loops alternating activity and mode neighborhood exploration. Evaluation on benchmark instances demonstrates the efficiency of both adaptations, making them among the most competitive algorithms for these problems.
- Valls et al. (2003) (see also Valls et al. (2005)) presented a novel SA metaheuristic incorporating fundamental concepts without explicit memory structures within a population-based framework. The algorithm employs a search strategy for intensification and a strategic oscillation mechanism for diversification. Using the topological order representation of schedules, three types of moves are introduced, two based on relative criticality and a third on multi-pass sampling ideas. The strategic use of probabilities for move construction further distinguishes this approach. Extensive computational testing on over 2000 instances demonstrated the effectiveness of the proposed solution method.

In addition to the aforementioned works dedicated to TS and SA approaches, Asif Raza and Mustafa Al-Turki (2007) compared the effectiveness of TS and SA, hybridized using properties of an optimal schedule from existing literature, for scheduling maintenance operations and job processing on a single machine. The numerical experimentation, involving large-sized problems, demonstrated that the developed meta-heuristics outperform the previous best-known heuristic algorithm, showcasing greater robustness against problem-related parameters. The study suggested future work incorporating machine failure and preventive maintenance, exploring multi-criteria scheduling, and considering stochastic parameters for job processing and maintenance-related factors. Furthermore, Das and Acharyya (2011) explored hybrid TS ans SA approaches for solving the RCPSP. Building upon the success of SA algorithms in previous works, the study introduced three new SA variants. The results demonstrated that SA combined with a tabu list and greedy heuristic surpasses other known methods, achieving optimal results in benchmark instances of the RCPSP.

Ant Colony Systems. Ant Colony System (ACS) is a class of optimization algorithms inspired by the behavior of ant colonies. Ants use pheromone trails to communicate with each other and mark their paths. When an ant finds a source of food, it leaves a trail of pheromones to guide other ants to the food source. The pheromones act as a chemical message that the ant uses to communicate with other members of the colony. Over time, the pheromone trails grow stronger as more ants follow them, making the path to the food source more attractive. The basic idea behind ACS algorithms is to mimic the behavior of ants as they forage for food. The algorithm starts by simulating a number of ants moving through a network. As the ants move, they lay down pheromone trails,

and the strength of these trails is updated based on the quality of the path they took. Over time, the algorithm identifies the path with the strongest pheromone trail, which corresponds to the best solution found to the problem. An efficient design of the ACS algorithm to solve the RCPSP can be found in Merkle et al. (2002), which uses a weighted version of the Latest Start Time (LST) priority rule to prioritize tasks to be scheduled. Unique features included a combination of two pheromone evaluation methods, dynamic influence of heuristics on ant decisions, and the ability for an elitist ant to forget the best-found solution. The algorithm was tested on a set of large benchmark problems from the PSPLIB. Comparative evaluations against various heuristics revealed that the proposed ACS algorithm performs well, on average. The results also revealed new best solutions for nearly one-third of benchmark problems not previously known to be solved optimally.

Local search-oriented approaches. Local search algorithms are optimization methods used to find the best solution for a particular problem by exploring the immediate neighborhood of a starting point. These algorithms begin with an initial solution and then make incremental changes to the solution in order to improve its quality. The goal is to find a locally optimal solution that maximizes or minimizes a particular objective function. Below, we describe examples of local search-oriented metaheuristics known for their ability to find high-quality solutions to complex optimization problems.

The Variable Neighborhood Search (VNS) (Mladenović and Hansen, 1997) is a metaheuristic algorithm used to explore different neighborhoods of a given solution to find a better candidate solution. In a VNS algorithm, the neighborhood structure is varied dynamically during the search process, allowing the algorithm to escape local optima and explore different regions of the solution space. The basic steps of a VNS algorithm typically include: generating an initial solution, selecting a neighborhood structure, generating a new solution in the neighborhood, evaluating the objective function to determine if the new solution is better than the current solution, and repeating the process until a satisfactory solution is found or a termination criterion is met. These algorithms have been shown to be effective for a wide range of optimization problems, including traveling salesman problems, vehicle routing problems, scheduling problems, and many others.

The Iterated Local Search (ILS) (Lourenço et al., 2003) is another local search-based metaheuristic that combines two strategies to solve combinatorial optimization problems. The first strategy is the local search, which starts from an initial solution and iteratively improves it by making small modifications until a local minimum is reached. The second strategy is the perturbation, which allows the algorithm to escape from local minima by introducing randomness and exploring new regions of the search space. In the ILS algorithm, the local search procedure is repeated for a fixed number of iterations or until no further improvements can be made. Then, a perturbation is applied to the current solution to generate a new starting point for the local search. The perturbation can be either a random shake-up of the current solution or a more sophisticated mechanism that incorporates problem-specific knowledge. ILS is a flexible algorithm that can be adapted to various optimization problems. Its effectiveness depends on the choice of the local search operator, the perturbation operator, and the stopping criteria.

Efficient local search-oriented approaches designs to the RCPSP can be found in, for example:

- Fleszar and Hindi (2004) presented a solution scheme based on VNS. The solution is encoded using valid activity sequences with respect to precedence constraints, and these sequences are transformed into valid active schedules through a serial scheduler. The search in the solution

space is facilitated by generating valid sequences using two types of move strategies. The solution scheme's effectiveness is attributed to the repeated use of effective lower bounding and precedence augmentation, which help reduce the solution space. Extensive experimentation on a standard set of 2040 benchmark instances demonstrated the scheme's efficacy, improving upon the best-known solutions for 48 instances and best-known lower bounds for 148 instances.

- Palpant et al. (2004) introduced a large neighborhood search approach for the RCPSP. The method involves fixing a subpart of the current solution while solving the remaining subproblem externally using either a heuristic or an exact solution approach, making it a hybrid scheme. The key aspect of the method lies in choosing the subproblem to be optimized. The paper explored several strategies for generating the subproblem and conducted extensive numerical experiments to evaluate these strategies. The results demonstrated the efficiency of the proposed method, comparing favorably with previous state-of-the-art heuristics for the RCPSP.
- In Barbalho et al. (2023), an extension of the RCPSP focusing on road network accessibility for logistics operations in relief and supply distribution. The study introduced a dedicated local search and an ILS metaheuristic. The methods were tested on theoretical instances and applied to a realistic instance featuring a complex urban networks. The proposed approach outperformed existing literature results.

2.3 Related problems and applications in disaster relief

In the recent scientific literature, new scheduling and routing problems and applications have been used to model several aspects related to issues in post disaster relief, such as debris removal, humanitarian logistics, risk mitigation and emergency response to forest fires. In general, these scheduling or integrated scheduling-routing approaches aim at minimizing operational cost, or at maximizing some satisfaction measure such as road accessibility, number of lives saved, etc. Another characteristic is that decisions may be restricted by financial or time budgets, thus, not all demands may be fulfilled (Duque and Sörensen, 2011; Duque et al., 2016). Some relevant reviews can be found in Overstreet et al. (2011), Farahani et al. (2020), Santos (2019) and Hu et al. (2019). We describe below in detail some of these approaches proposed in the literature.

2.3.1 Humanitarian logistics

Humanitarian logistics is the process of planning, implementing and coordinating the flow of goods, services and information to support humanitarian efforts in response to crises and disaster situations. It involves the management of resources to ensure that necessary aid reaches those in need as efficiently as possible, with the aim of reducing the impact of disasters and their aftermath on affected populations (Overstreet et al., 2011).

Humanitarian logistics play a crucial role to overcome natural disasters such as earthquakes, hurricanes and floods that can cause immense destruction, leaving communities in urgent needs, such as food, water, shelter, and medical aid (Sakuraba et al., 2016). Works in humanitarian logistics have become an important component in developing efficient use of resources and effective disaster management planning. Below, we review some optimization problems with applications in humanitarian logistics that involve the coordination and management of work-forces, supplies, and

equipment in response to these emergencies.

The Hierarchical Traveling Salesman Problem (HTSP) presented in Panchamgam et al. (2013) consists of a routing problem for relief operations whose primary goal is to deliver relief supplies to demand points. Each location has a priority indicating the urgency of the demand and the deliver of supplies is performed by a single vehicle of unlimited capacity that takes into account the priority. Typically, the demands with the highest priorities need to be satisfied before those with a lower priority. The authors introduce the d–relaxed priority rule (see Section 2.2.1) to relax priorities of demand locations and perform a tradeoff between minimizing vehicle route total distance and satisfying location priorities.

The HTSP was later extended in Hà et al. (2022) as the Capacitated Traveling Salesman Problem with d–relaxed priority rule (CTSP-d), introducing a more efficient MILP formulation. The authors also develop a hybrid metaheuristic that mixes components of ILS, VNS and the Greedy Randomized Adaptive Search Procedure (GRASP). The methods were tested against a dataset from the TSPLIB[3], by randomly adding priorities $\{1, 3, 5\}$ and $d = \{0, 1, 3\}$ values. The computational experiments evaluated the performance of the MILP formulations and the mataheuristic approach on instances with 42–52 vertices. In addition, a set of instances with 100–200 vertices was also generated to analyze the effectiveness of the metaheuristic on larger instances.

Similarly to the HTSP and the CTSP-d, the Vehicle Routing Problem with Relaxed Priority Rules (VRP-RPR) presented in Doan et al. (2021) assigns costumers (demand locations) to several priority groups and the ones with the higher priorities typically need to be served before the lower ones. The authors also propose a relaxation scheme of priorities based on the d–relaxed priority rules from Panchamgam et al. (2013), where choosing the appropriate d value depends on the emergency level. The authors consider a fleet of heterogeneous vehicles and each route is constrained by the vehicle's capacity and autonomy with the objective of minimizing the weighted sum of the total demand lost and the total travel cost. The authors also propose a MILP formulation and an Adaptive Large Neighborhood Search (ALNS) to solve a benchmark of instances composed of theoretical instances from Hà et al. (2022). For the computational experiments, the authors consider three test scenarios with strict, partially and zero priority rules. Both optimization methods were applied to solve a set of 10 small-sized instances with 20 to 30 nodes and 3 priority groups; and the ALNS alone was tested against larger instances with 100 nodes and 3 priority groups.

Despite the aforementioned studies have focused on simple VRPs, one important factor presented in those problems is the d–relaxed priority rules, where the trade-off between the demand urgency and the solution quality may be adjusted by a value threshold. That is, less strict priority rules (higher values of the d threshold) generate better solutions than stricter ones, but with the possible negative effect that urgent locations may be neglected until late operation dates. In Chapter 3, we investigate the RCPSP-RP, modeled as an integrated scheduling-routing application with the d–relaxed priority rules.

2.3.2 Improving accessibility in post-disaster urban networks

During natural disasters such as earthquakes, hurricanes, floods, or wildfires, roads can become damaged or unsafe, making it challenging for rescue teams and aid workers to reach the affected

[3]TSPLIB is a library of sample instances for the Travelling Salesman Problem from various sources and of various types. See `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`

areas quickly. Some studies have focused on improving physical accessibility in urban networks damaged by natural disasters, with the objective of reducing the delay in response time and help mitigate the loss of life and property.

In Duque and Sörensen (2011), a GRASP metaheuristic coupled with a VNS was proposed to allocate scarce resources to restore road connectivity between towns and regional distribution centers. The authors consider financial and manpower-time budgets, which are associated with each road to be repaired. Another study by Duque et al. (2016) formulated the Network Repair Crew Scheduling and Routing Problem (NRCSRP)), which consists in defining optimal sequencing for repairing damaged nodes in a network. The problem assumes that a repairing crew is employed to optimize accessibility to the towns and villages that demand humanitarian relief by repairing roads. The goal is to minimize the date in which each demand node is available, weighted by the amount of demand. The authors proposed an exact dynamic programming algorithm and an iterated greedy-randomized constructive procedure. Computational experiments were performed on instances with nodes varying from 21 to 401 nodes, with percentage from 5% to 50% of the nodes damaged.

The study by Feng and Wang (2003) was dedicated to rehabilitation of intercity highways in emergencies, where a multi-objective scheduling problem is formulated with the following three objectives: maximizing the performance of emergency rehabilitation; minimizing the risk to rescuers; and maximizing the savings of lives. The computational experiments are performed on an instance based on the 1999 earthquake that struck Nantou county in Taiwan. This corresponding instance is composed of 52 nodes, 62 arcs, and 10 damages nodes. In Aksu and Ozdamar (2014), two mathematical formulations are proposed for the Debris Clearance Scheduling Problem (DCSP), where a set of work-troops is assigned to repair blocked paths, with the goal of maximizing the total earliness in repairing all paths. The methods were tested over two districts of Istanbul in Turkey. The first one contains 212 roads and 49 blocked roads, and the second instance has 386 roads and 79 blocked roads.

The Multivehicle Synchronized Arc Routing Problem to Restore Network Connectivity (K-ARCP) presented in Akbari and Salman (2017) (see also Akbari et al. (2021)) is an open vehicle routing problem to restore blocked arcs in an urban network. A mathematical formulation and a matheuristic based on the linear relaxation of the problem are proposed. Three data sets based on Istanbul city were generated including up to six vehicles: the first one corresponds to the highways network, with 74 nodes and 179 edges, the second data set has 389 nodes and 689 edges, and the third one is based on the southwestern region of Istanbul city with 250 nodes and 539 edges. Route damage is taken as a probability of being impassable after a earthquake, high, medium and low-risk edges, respectively 0.3, 0.2 and 0.1 of probability. In another study by Vodák et al. (2018), a VRP to minimize the time of repairing blocked road segments is solved with an ant colony algorithm. The authors have produced a realistic instance based on the road network of the Zlín region in the Czech Republic, composed of 723 nodes, 974 links and 46 blocked roads.

The Disruption Scheduling problem on Urban Networks (DSUN) was introduced in Coco et al. (2020) in the context of rapid and unplanned urban growth. The authors address critical issues that raise from various social, economic, and ecological challenges in urban mobility. The DSUN involves scheduling planned disruptions in an urban road network while ensuring strong connectivity. Disruptions can break network connections, requiring route direction modifications (arc reversals) and potentially disturbing users habits. The objective is to minimize the number of arc reversals and the sum of starting times for all disruptions simultaneously. The study formulates DSUN

mathematically and presents an exact algorithm based on the ϵ-constraint method to handle its bi-objective nature. Computational experiments were conducted on theoretical and realistic instances, demonstrating the algorithm's ability to prove optimality for instances with up to 100 vertices and 20 disruptions.

The Work-troops Scheduling Problem, described in Sakuraba et al. (2016), concerns the scheduling of a homogeneous fleet of work-troops to unblock blocked roads in urban networks affected by natural disasters, such as earthquakes. Implicit precedence relations are considered due to roads' physical accessibility in the network. The authors developed three greedy heuristics to solve a theoretical benchmark of small instances with up to 20 blocked roads and a large-scale realistic instance of the 2010 Haiti earthquake with 16,660 nodes, 19,866 roads and 536 blocked roads. An extension of this work is subsequently found in Barbalho et al. (2023), where the authors developed a GRASP and an ILS metaheuristic coupled with a dedicated local search to solve the set of theoretical and realistic instances from Sakuraba et al. (2016).

A similar application is the Scheduling vehicle Routing Problem to Clean Debris (SRP-CD), presented in Pena et al. (2023) as an integrated scheduling-routing problem, concerning the strategic (scheduling) and the operational (routing) decisions to remove debris in urban areas damaged by major disasters. The objective is simultaneously to minimize the operation duration, at the strategic level, and the total costs of vehicles routes, at the operational one. The authors propose a new mathematical model based on a dynamic multi-flow formulation, several constructive heuristics and a Large Neighborhood Search (LNS)-based metaheuristics. These methods were applied to theoretical instances with 10 to 500 demand nodes and 2 to 7 homogeneous vehicles.

2.3.3 Scheduling problems in emergencies

Some works in the literature focus on providing emergency response models to minimize the impact of an incident or emergency, by enabling organizations and individuals to take measures to mitigate the risks and prevent further harm, ensuring the health and safety of people, property and the environment.

The Rescue Unit Assignment and Scheduling Problem (RUASP), presented in Wex et al. (2014), concerns the allocation and scheduling of heterogeneous rescue units located in operation centers, to emerging incidents with the objective of minimizing the sum of completion times of incidents weighted by their severity. The problem was modeled as a binary quadratic optimization model where each incident may require a specific type of rescue unit. The authors developed a GRASP metaheuristic, a Monte Carlo-based heuristic and a conjoint of 8 construction heuristics coupled with 5 improvement heuristics. The methods were tested against small and medium sized instances with up to 40 incidents and 40 rescue units, designed in contact with rescue teams active in the major 2011 Japan earthquake.

The Early Stage Response Problem (ESRP), presented in Kim et al. (2018), provides an emergency response composed of routing and scheduling of teams with the objective of maximizing risk prevention to buildings under imminent threat such as fire spread, gas leak and explosions, neutralizing further hazards to material and human lives. Thus, the model reflects dynamic changes in emergency situations, such as changes in hazard intensity and time required to neutralize dangerous targets. In addition, the authors developed a MILP and two greedy algorithms to solve theoretical instances based on data from the central city building in Seoul, Korea, with up to 5 dangerous

targets.

In addition to previous publications, in recent years, works in the actual context of emergency response to forest fires have been proposed as integrated scheduling-routing problems with static, dynamic or stochastic characteristics. The Resource-Constrained Emergency Scheduling Problem for Forest Fires with Priority Areas (RCESP) presented in Ren and Tian (2016) provides a disaster relief model to extinguish forest fires, where a homogeneous fleet of firefighting teams is dispatched to multiple areas according to their priorities following a strict policy, with the objective of minimizing teams' total travel distance constrained by a maximum operational autonomy. The authors developed a genetic algorithm coupled with a particle swarm metaheuristic to solve a case study based on realistic data from Heilongjiang, China, with 7 fire points and 3 teams. A dynamic version of the RCESP was later presented in Wu et al. (2019), where fire spreading and fire extinguishing speed of teams were considered. The problem was solved as a MILP with total rescue time minimization. Furthermore, Wang et al. (2020) extended the two previous works with a bi-objective MILP to simultaneously minimize the total travel distance and the total fire extinguishing rescue time. Fuzzy logic decision-making based on an ϵ-constraint method was also designed to obtain schedule schemes and produce Pareto fronts. The MILP and the ϵ-constraint method were tested against the benchmark instances of Ren and Tian (2016) and several randomly generated instances with up to 16 fire points and 6 rescue teams.

The RCESP was examined in Fang et al. (2019), where the concept of "weak" priority levels was introduced. This allows fire points to share the same priority level, enabling methods to optimize routes within that level while maintaining priorities between different priority levels. This can be seen as a weaker version of the d–relaxed priority rules.

2.3.4 Scheduling problems under uncertainties

Parameter uncertainty refers to the situation where the values of some or all problem parameters are unknown or uncertain (probabilistic). Solving optimization problems with parameter uncertainty requires the use of stochastic optimization techniques, which involves modeling the uncertain parameters as random variables and solving the problem using probabilistic methods. Another approach is to use robust optimization techniques, which involve optimizing the worst-case scenario under the assumed uncertainty. Ultimately, the choice of method for handling parameter uncertainty will depend on the specific problem and the degree of uncertainty in the problem's parameters. The book Billaut et al. (2013) covers many important models flexibility and robustness in scheduling problems.

In the Stochastic Debris Clearance Problem (SDCP), presented in Çelik et al. (2015), the main objective is to clear blocked roads during the response phase by moving debris to the sides of the roads in order to maximize benefit received from meeting relief demand. To find an optimal solution for small and large instances of this problem, a partially observable Markov decision process model was used. Tests were conducted using two benchmark sets of instances: one featuring structured grid and ring networks resembling urban city road networks with 25–225 nodes, and another based on a realistic earthquake scenario in Boston, simulated using a disaster simulation tool.

The Multi-Resource Scheduling Model under Uncertainty (MRSU), introduced in Bodaghi et al. (2020), presents a MILP framework for scheduling and sequencing heterogeneous resources under uncertain parameters, where the deterministic MILP is solved many times with multiple stochas-

tic scenarios to identify the most persistent optimal solution. The authors applied the MRSU framework to a 2009 bushfire case study in Australia with 25 bushfires with stochastic severity and propagation and 16 vehicles, where a total of 1000 scenarios were solved to identify the most probable one.

In Tirkolaee et al. (2020), an extension of Wex et al. (2014) was proposed by adding a second objective function and parameter uncertainty such as travel times, incident severity and processing times to minimize also the total delay time. The authors applied the robust optimization technique to solve the model using several theoretical instances with up to 20 incidents and 15 rescue units.

2.3.5 Position of this thesis

Based on our current understanding, none of the previous relevant contributions addresses the particularities of industrial catastrophe scenarios where the toxicity of the released substances must be considered. Knowledge about the long-term consequences of these substances to the environment and human health are often unknown or limited (Keim, 2011), which motivates further study of industrial disasters from the perspective of operational research so in order to minimize their impacts once they occur. The main contributions of this thesis are both proposing and modeling the RCPSP-RP and the RCPSP-RTD as optimization problems in the context of industrial disasters, as well as proposing methods that, in conjunction with existing risk management plans, can help support decision making in the occurrence of such situations.

In Table 2.2, we present an overview of the most related works to the RCPSP-RP regarding the characteristics of problems found in the literature and the approaches proposed. The works describing at least 4 features similar to RCPSP-RP and the RCPSP-RTD were included. The works are sorted by year, by classical problem or by application. A black dot expresses that the corresponding study includes the characteristic described on the top of the column, such that:

- The column **Problem class** indicates whether the work is an scheduling, routing or integrated scheduling and routing problem.
- The column **Precedence relations** specify if the problem includes implicit or explicit precedence relations between tasks. The relations may either refer to operational dependence (such as the classical precedence constraints) or priority constraints coupled with a relaxation threshold.
- The column **Task type** describes whether the tasks are considered to be static, dynamic or have uncertain parameters.
- The column **Depot/source** are related to routing or integrated scheduling-routing problems and state if vehicles/resources initially start on one or more specific nodes in the network.
- The column **Resources/fleet** state if single or multiple resources or vehicles are considered and whether they are homogeneous or heterogeneous.
- The column **Objective function** refers to the optimality criteria.
- The column **Approaches** state the optimization methods applied to solve the proposed problem.

Context	Works	Problem class: Scheduling	Problem class: Routing	Precedence relations: Non adjustable	Precedence relations: Adjustable	Task type: Static	Task type: Dynamic	Task type: Uncertain	Depot/source: Single	Depot/source: Multiple	Resources/fleet: Single	Resources/fleet: Multiple	Resources/fleet: Homogeneous	Resources/fleet: Heterogeneous	Objective function: Min. time	Objective function: Min. cost	Objective function: Max. satisf.	Approach: Mathematical model	Approach: Exact method	Approach: Heuristics	Approach: Simulation
Humanitarian logistics	Panchamgam et al. (2013)		•		•	•			•		•		•			•		•			
	Hà et al. (2022)		•		•	•			•		•		•			•		•		•	
	Doan et al. (2021)		•		•	•			•			•		•		•		•		•	
Post-disaster recovery	Duque and Sörensen (2011)	•				•						•		•			•	•		•	
	Duque et al. (2016)	•	•	•		•			•		•		•				•		•	•	
	Akbari and Salman (2017)		•			•				•		•	•		•			•		•	
	Pena et al. (2023)	•	•			•			•			•	•		•	•		•		•	
	Sakuraba et al. (2016)	•		•		•				•		•	•			•		•		•	
	Barbalho et al. (2023)	•		•		•				•		•	•			•				•	
Emergency response	Wex et al. (2014)	•	•			•						•		•		•		•		•	
	Kim et al. (2018)	•	•				•			•		•	•				•	•		•	
	Ren and Tian (2016)	•	•	•		•			•			•		•		•		•		•	
	Wu et al. (2019)	•	•	•			•		•			•		•		•		•			
	Wang et al. (2020)	•	•	•			•		•			•		•	•	•		•		•	
	Fang et al. (2019)	•	•	•		•			•			•		•		•		•			
Parameter uncertainty	Çelik et al. (2015)	•						•		•		•	•				•	•		•	
	Bodaghi et al. (2020)	•	•					•		•		•	•			•		•			•
	Tirkolaee et al. (2020)	•	•					•				•		•		•		•			
Thesis contributions	RCPSP-RP (Chapter 3)	•	•		•	•			•			•	•			•		•		•	
	RCPSP-RTD (Chapter 4)	•					•		•			•		•		•		•		•	

Table 2.2: Summary of the application works related to the RCPSP-RP and the RCPSP-RTD.

Chapter 3

The resource-constrained project scheduling problem with risk and priorities

In this chapter, we investigate the post-catastrophe operations in the event of industrial disasters, where a group of specialized teams and equipment are deployed to clean up areas contaminated by dangerous substances, with the objective of mitigating the impact of accidents as quickly as possible. In this scenario, the whole contaminated area is mapped as a connected graph, where each contaminated site is represented as a node with an associated risk attribute that depends on the hazardous nature of the pollutants present on it. Teams can be deployed in the area covered by the graph by traversing secured paths to perform the cleaning operation and neutralize the risk associated with each node. In order to model this scenario, we propose a new optimization problem named RCPSP-RP, which is composed of an integrated scheduling and routing model with different policies of task prioritization.

The proposed RCPSP-RP is an extension of the well-known RCPSP, were resources represent the specialized teams and equipment that are deployed from a specific node in the graph (called the depot) to perform the cleaning operation and mitigate risk by processing all mapped tasks. Travel times when traversing the edges are implied based on their traveling distance. More specifically, these travel times are treated as Sequence-Dependent Setup Times (SDST) (see, *e.g.*, Allahverdi et al. (2008)) by taking into account the shortest travel time between every pair of nodes in the graph. The RCPSP-RP performs task prioritization by employing precedence constraints between tasks based on their risk attribute. As a consequence, tasks with higher risk take precedence over lower risk ones. This behavior is defined as a strict priority policy. In order to gain insights on the impact that task prioritization may have on the quality of solutions, we follow a relaxation schema inspired by Panchamgam et al. (2013) by introducing a relaxation threshold that partially or completely relaxes precedence constraints according to the stated level of strictness. Thus, providing a moderate or no priority policy, respectively.

In order to conduct an experimental analysis on the RCPSP-RP, we propose mathematical for-

mulations inspired by an exact formulation for the classical RCPSP-RP from Christofides et al. (1987). In addition, an ILS metaheuristic is also developed to cope with large problem instances, when optimal – or even integer – solutions cannot be computed by the exact methods in reasonable time. Experiments are conducted on a set of theoretical instances to investigate different problem scenarios and the three types of priority schemes, namely strict, moderate and no priority policies. Overall, results show that less strict priority policies can improve the quality of the solutions, although longer running times are required due to the exploration of a larger solution space. Likewise, in certain instances, a full relaxation of travel times yields up to 100% improvement in solutions. This particular scenario can arise in the context of industrial disasters contained within enclosed areas, where traveling times are significantly smaller than processing times and, therefore, they can be neglected from a practical perspective.

The remainder of this chapter is organized as follows: Section 3.1 introduces the formal mathematical formulation of the RCPSP-RP, including the modeling of task prioritization in Section 3.1.2 and integer programming models in Section 3.1.3. Section 3.2 is dedicated to the proposed ILS metaheuristic. Afterwards, a description of the computational experiments including the experimental setup, data generation and automated calibration of ILS parameters using the Iterated Racing for Automatic Algorithm Configuration (IRACE) package are presented in Section 3.3. In this section, the analysis of results is presented, including a comparison between the results obtained using CPLEX and ILS for the different experimental scenarios. Finally, concluding remarks and perspectives for future works are discussed in Section 3.4.

3.1 Problem definition

Let $G = (V, E)$ be a connected spatial graph, where V is the set of nodes and E is the set of edges. Let $V_o \subset V$ be defined as the singleton set containing the depot node where K homogeneous resources are initially located. Let $V_d \subset V$ ($V_o \cap V_d = \emptyset$) be the set containing tasks that need to be performed by the resources, where each task $i \in V_d$ has risk $r_i \in (0, 1]$ and processing time $p_i \in \mathbb{N}^+$ attributes. The r_i is determined by the hazardous nature of the materials present at i in a scale of low- ($r_i \leq 0.3$), medium- ($0.3 < r_i \leq 0.5$), high- ($0.5 < r_i < 0.9$) or critical- ($r_i \geq 0.9$) risk; and p_i is the estimation of the time required to process it, *i.e.* completely remove the hazardous pollutants from the site. Every task should be processed by a single resource and after been started it may not be interrupted (preemption is not allowed).

The set E stands for the secured paths resources use to traverse from the depot to tasks, and from tasks to other tasks, with reduced exposition to dangerous substances. Each path has a $s_{ij} \in \mathbb{N}_0$, $(i, j) \in V \times V$, that represents the sequence-dependent setup time between i and j. Thus, the problem consists in defining a feasible scheduling over a time horizon $H = \{1, ..., T\}$, for some large enough integer T, that minimizes the sum of completion times weighted by task risk:

$$\mathcal{F} = \sum_{i \in V_d} r_i C_i \tag{3.1}$$

The objective function $\mathcal{F}$ represents the accumulated risk of unfinished tasks at each time unit $t \in H$, from which we can generate a graph that represents the risk mitigation process over the time horizon. For instance, Figure 3.1 presents the visualization of the example problem input and

solution in Table 3.1. The accumulated risk, in orange, starts at its maximum and decreases every time a task is completed. The graph total area is denoted as the solution *overall risk*. Naturally, graphs with the smallest possible areas in the interval $[1, T]$ are optimal solutions.

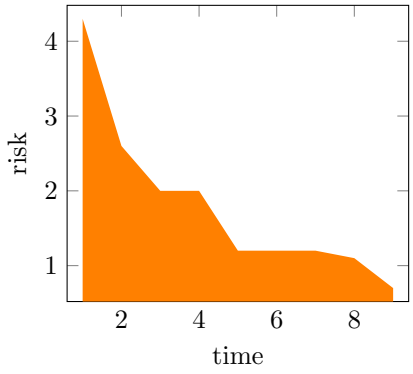


Figure 3.1: Example solution with a total area (overall risk) of 16.3.

Problem data			Completion time
i	p_i	r_i	C_i
1	9	0.7	9
2	7	0.3	8
3	6	0.1	8
4	3	0.1	7
5	4	0.8	4
6	1	0.9	1
7	1	0.6	2
8	1	0.8	1

Table 3.1: Example problem input and optimal solution.

3.1.1 Estimating an upper bound for the operation horizon

Estimating a good value for T is not a simple task, and actually determining its optimal value is as hard as solving the RCPSP-RP. Although methods such as the earliest start time are useful for $K = 1$, they do not work for $K \geq 2$ since tasks may overlap (Section 3.1.2 will review this with more detail). If sequence-dependent setup times are not considered, a good estimation for T can be obtained by Equation (3.2):

$$T = \max_{i \in V_d}\{p_i\} + \left\lceil \frac{\sum_{i \in V_d} p_i}{K} \right\rceil \tag{3.2}$$

Equation (3.2) is composed of the longest processing time added to the ceiling the of the sum of processing times divided by the number of resources to account for tasks being executed in parallel.

On the other side, if sequence-dependent setup times are required, Equation (3.3) can offer a decent estimation for T in most cases:

$$T = \left\lceil \frac{|V_d| \max_{i,j \in V \times V}\{s_{ij}\} + \sum_{i \in V_d} p_i}{K} \right\rceil \tag{3.3}$$

Equation (3.3) is composed of the ceiling of the number of tasks multiplied by the longest sequence-dependent setup time, added to the sum of processing times, divided by the number of resources to account for tasks that execute in parallel.

3.1.2 Priority constraints

The RCPSP-RP performs task prioritization in a similar fashion as the d–relaxed priority rule presented in Panchamgam et al. (2013) (see also Doan et al. (2021)). Priority constraints are represented by a Direct Acyclic Graph (DAG) defined as $D = (V_d, P)$, P is defined as the set of ordered pairs $P = \{(i, j) \in V_d^2 \mid r_i > r_j + d\}$ representing the priority relations. The relaxation threshold $d \in [0, 1]$ is used to adjust the priority policy strictness. We define the cases $d = 0$, $d = 0.5$ and $d = 1$ as, respectively, strict, moderate and no priority policy. The strict policy ($d = 0$) imposes tasks to be processed in a decreasing order of risk; the moderate policy ($d = 0.5$) provides relative task prioritization; and the no priority policy ($d = 1$) completely disables task prioritization.

As an illustration, Figure 3.2 shows the priority relations generated for the strict and moderate priority policies using the example problem data $r = [0.9, 0.7, 0.7, 0.7]$.

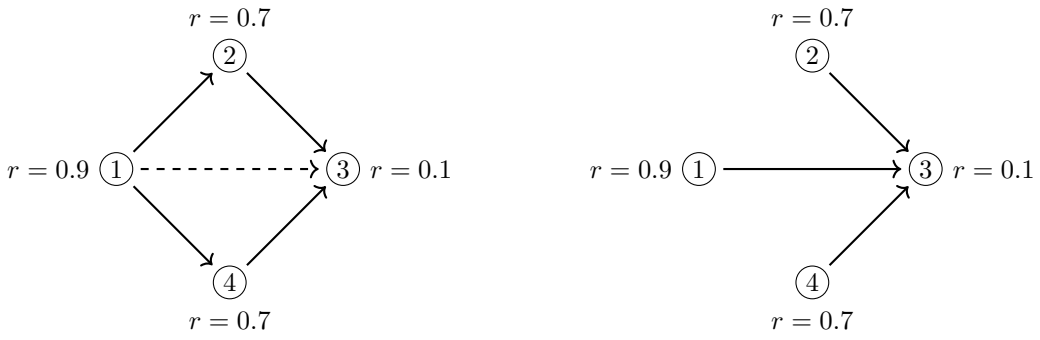
- Figure 3.2a displays the complete set of priority relations. It is worth noting that the priority relation $(1, 3)$ is included for illustrative purposes and do not need be a part of set P since it is redundant. Redundant arcs can be eliminated through a process known as transitive reduction of acyclic graphs (see, *e. g.*, Gries et al. (1989)).
- In Figure 3.2b, the moderate case, the priorities $(1, 2)$ and $(1, 4)$ have been relaxed because the conditions $r_1 > r_2 + d$ and $r_1 > r_4 + d$ **are no longer met**. Recall that an arc (i, j) is inserted whenever the condition $r_i > r_j + d$ is satisfied. For example, let $r_i = 0.9$ and $r_j = 0.5$, this condition is not met for $d = 0.5$, thus the arc (i, j) is not present. Moreover, the priority relation $(1, 3)$ emerges as it is no longer redundant.

It is clear that $P = \emptyset$ for the no priority policy.

Given that priority constraints aim at modeling task prioritization and not operational relations, tasks are allowed starting at the same time as their predecessors as long as there are enough available resources. For instance, $(i, j) \in P$ indicates that i has priority over j and j may start at the same time or after i, but never before.

3.1.3 Mathematical formulations

This section presents the mathematical models for the problem scenarios with and without sequence-dependent setup times. The models are designed as time-indexed integer formulations inspired by Christofides et al. (1987) formulation for the RCPSP (see Artigues (2017) for a review on time-indexed formulations for the RCPSP). For reference, Table 3.2 summarizes the notations for the problem input data.



(a) strict policy.

(b) moderate policy.

Figure 3.2: Priority relations for the (3.2a) strict and (3.2b) moderate priority policies.

Input parameter	Description
V	Set of nodes
E	Set of secured paths
V_d	Set of tasks
V_o	Singleton set containing the depot node
P	Set of precedence relations (see Section 3.1.2)
H	Set of time units
T	Operation duration upper bound
K	Number of available resources at the depot
$r_i \in (0, 1]$	Risk attribute of $i \in V_d$
$p_i \in \mathbb{N}^+$	Processing time attribute of task $i \in V_d$
$s_{ij} \in \mathbb{Z}_0^+$	Sequence-dependent setup times (travel times) between i and j, $(i, j) \in V \times V$

Table 3.2: General problem input data and description.

Scenario without sequence-dependent setup times

The model for the scenario without sequence-dependent setup times employs the following decision variables:

- The binary pulse variables x_i^t, $(i, t) \in V_d \times H$, are such that $x_i^t = 1$ if task i starts at period t, $x_i^t = 0$ otherwise;

- The integer variables S_i and C_i determine, respectively, the start and completion times of task $i \in V_d$ and are calculated in function of the x_i^t variables.

$$\text{Minimize} \sum_{i \in V_d} r_i C_i \qquad \text{s.t.} \tag{3.4}$$

$$\sum_{i \in V_d} \sum_{\tau=t-p_i+1}^{t} x_i^\tau \leq K \qquad \forall t \in H \tag{3.5}$$

$$\sum_{t \in H} x_i^t = 1 \qquad \forall i \in V_d \tag{3.6}$$

$$S_i = \sum_{t \in H} t x_i^t \qquad \forall i \in V_d \tag{3.7}$$

$$C_i = S_i + p_i \qquad \forall i \in V_d \tag{3.8}$$

$$S_i \leq S_j \qquad \forall (i,j) \in P \tag{3.9}$$

$$S_i \geq 0 \qquad \forall i \in V_d \tag{3.10}$$

$$C_i \geq 0 \qquad \forall i \in V_d \tag{3.11}$$

$$x_i^t \in \{0,1\} \qquad \forall (i,t) \in V_d \times H \tag{3.12}$$

The objective function (3.4) minimizes the sum of completion times weighted by task risk. The resource constraints are defined by inequalities (3.5). The constraints (3.6) state that every task starts exactly once in the scheduling horizon. The equations (3.7) and (3.8) compute, respectively, the start and completion times of tasks. The constraints (3.9) are the priority constraints generated by the the employed priority policy. The model's variables are defined in (3.10) – (3.12). This formulation has $\mathcal{O}(|V_d \times H|)$ binary variables, $\mathcal{O}(|V_d|)$ integer variables and $\mathcal{O}(|V_d| + |P| + |H|)$ constraints.

Scenario with sequence-dependent setup times

The model for the scenario with sequence-dependent setup times employs the following decision variables:

- The binary pulse variables y_{ij}^t represent the time-indexed flow between i and j at period t, $(i,j,t) \in V \times V \times H$, such that $y_{ij}^t = 1$ if task j starts after i at period t, $y_{ij}^t = 0$ otherwise;

- The integer variables S_i and C_i determine, respectively, the start and completion times of task $i \in V_d$ and are calculated in function of the y_{ij}^t variables.

$$\text{Minimize} \sum_{i \in V_d} r_i C_i \qquad \text{s.t.} \tag{3.13}$$

Constraints (3.9)

$$\sum_{t \in H} \sum_{j \in V_d} y_{ij}^t \leq K \qquad i \in V_o \tag{3.14}$$

$$\sum_{t \in H} \sum_{i \in V \setminus \{j\}} y_{ij}^t = 1 \qquad \forall j \in V_d \tag{3.15}$$

$$\sum_{t \in H} \sum_{i \in V \setminus \{j\}} y_{ji}^t = 1 \qquad \forall j \in V_d \tag{3.16}$$

$$\sum_{t \in H} \sum_{i \in V \setminus \{j\}} t y_{ji}^t \geq C_j \qquad \forall j \in V_d \tag{3.17}$$

$$S_j = \sum_{t \in H} \sum_{i \in V \setminus \{j\}} t y_{ij}^t \qquad \forall j \in V_d \tag{3.18}$$

$$C_j = \sum_{t \in H} \sum_{i \in V \setminus \{j\}} (t + s_{ij} + p_j) y_{ij}^t \qquad \forall j \in V_d \tag{3.19}$$

Variables (3.10) – (3.11)

$$y_{ij}^t \in \{0, 1\} \qquad \forall (i, j, t) \in V \times V \times H \tag{3.20}$$

The objective function (3.13) and the priority constraints (3.9) are previously defined. The constraints (3.14) represent the resources allocation from the depot node, and their flow conservation of allocations from task to task is ensured by the constraints (3.15) – (3.17). The equations (3.18) and (3.19) calculate, respectively, the start and completion times of tasks. The model's variables are defined by (3.10) – (3.11) and (3.20). This formulation has $\mathcal{O}(|V \times V \times H|)$ binary variables, $\mathcal{O}(|V_d|)$ integer variables and $\mathcal{O}(|V_d| + |P| + |H|)$ constraints. One may note that the model (3.4) - (3.12) can be simulated by switching off sequence-dependent setup times, *i.e.* setting $s_{ij} = 0 \ \forall \ (i, j) \in V^2$, but with a considerably less efficient formulation.

Waiting time constraints

The waiting time constraints (3.17) are required to avoid a very specific case when priority constraints together with sequence-dependent setup times and processing times can lead to a non-optimal greedy decision. Without those constraints, resources, after finishing a task, can prematurely start a next task that would be better serviced by another resource in a future time unit.

In order to illustrate that specific scenario, Figure 3.3 presents a graph with 4 task nodes that is part of an omitted larger graph. The tasks are numbered from 1 to for 4 and $r_1 = 0.9$, $r_2 = 0.9$, $r_3 = 0.7$ and $r_4 = 0.5$; and $p_1 = 2$, $p_2 = 1$, $p_3 = 1$ and $p_4 = 1$ are their respective attributes (see Table 3.3). The edges indicate the sequence-dependent setup times. It is assumed that M and m are two arbitrary integer values, with $M > m + 1$.

The Figure 3.4 illustrates the issue faced when the strict policy is applied but waiting times are not supported. A non-optimal solution is obtained after the following order of decisions:

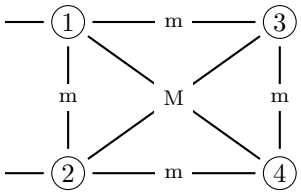
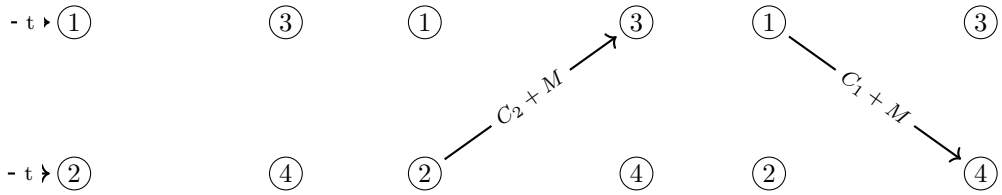


Figure 3.3: Special case: graph fragment.

1. Suppose that at time t, the resource k_1 starts task 1 and the resource k_2 starts task 2, since these are the highest priority tasks.
2. At time $t + p_2$, k_2 finishes processing task 2 ($C_2 = t + p_2$) and will immediately start task 3, since it is the next highest priority task.
3. At a later time $t + p_1$, k_1 finishes processing task 1 ($C_1 = t + p_1$) and will immediately start task 4, which is the last remaining task.
4. At time $t + p_2 + M + p_3$, task 3 is completed ($C_3 = C_2 + M + p_3$);
5. At time $t + p_1 + M + p_4$, task 4 is completed ($C_4 = C_1 + M + p_4$).



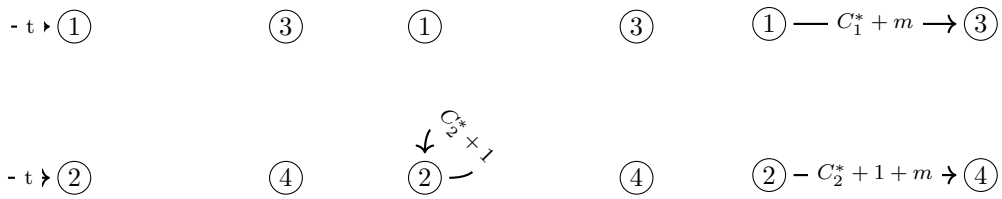
(a) t. (b) $C_2 + M$. (c) $C_1 + M$.

Figure 3.4: Special case: non-optimal solution without waiting time.

It is clear to see that the decision taken at step 2 is not optimal, since task 3 would be better serviced by the resource processing task 1. This issue is handled by the waiting time constraints which provide resources with the option to stay idle and wait for a better suitable task instead of immediately starting the next one. The correct solution is obtained by the following order of optimal decisions:

1. Suppose that at time t, the resource k_1 starts task 1 and the resource k_2 starts task 2, since these are the highest priority tasks.
2. At time $t + p_2$, k_2 finishes processing task 2 ($C_2^* = t + p_2$). Recall that starting task 3 now is not an optimal decision and starting task 4 is not possible due to the priority constraints. k_2 now stays idle for 1 time unit.
3. At a later time $t + p_1$, k_1 finishes processing task 1 ($C_1^* = t + p_1$). Now each remaining tasks can be optimally allocated to the best resource available: k_1 starts task 3 and k_2 starts task 4.
4. At time $t + p_1 + m + p_3$, task 3 is completed ($C_3^* = C_1^* + m + p_3$);

5. At time $t + p_2 + 1 + m + p_4$, task 4 is completed ($C_4^* = C_2^* + 1 + m + p_4$).



(a) t. (b) $C_2^* + 1$. (c) $C_1^* + m$.

Figure 3.5: Special case: optimal solution with waiting times.

Input parameter			Solution	
i	p_i	r_i	C_i	C_i^*
1	2	0.9	$t + p_1$	$t + p_1$
2	1	0.9	$t + p_2$	$t + p_2$
3	1	0.7	$C_2 + M + p_3$	$C_1^* + m + p_3$
4	1	0.5	$C_1 + M + p_4$	$C_2^* + 1 + m + p_4$

Table 3.3: Comparison of non-optimal (C_i) and optimal (C_i^*) solutions.

The optimal decisions at step 3 were possible due the idle time unit at step 2. The optimality is verified by $C_3 > C_3^*$ and $C_4 > C_4^*$, which holds for any $M > m + 1$ and $p_1 > p_2$:

$$C_3 > C_3^* \tag{3.21}$$
$$t + p_2 + M + p_3 > t + p_1 + m + p_3 \quad \text{(apply } p_1 = p_2 + 1\text{)} \tag{3.22}$$
$$t + p_2 + M + p_3 > t + p_2 + 1 + m + p_3 \tag{3.23}$$

$$C_4 > C_4^* \tag{3.24}$$
$$t + p_1 + M + p_4 > t + p_2 + 1 + m + p_4 \quad \text{(apply } p_2 = p_1 - 1\text{)} \tag{3.25}$$
$$t + p_1 + M + p_4 > t + p_1 + m + p_4 \tag{3.26}$$

3.2 Iterated local search for the RCPSP-RP

The ILS is a metaheuristic composed of an iterative single-chain of solutions generated by an embedded heuristic. From the stand point of metaheuristic classification, it opposes to population-based algorithms because it starts with one single initial solution and interactively improves that solution by applying consecutive perturbations and local search heuristics. Usually, ILS leads to better solutions than if one were to use repeated random trials of that same embedded heuristic (Lourenço et al., 2003).

The concept of ILS leaves many implementation choices to developers – thus, one can apply problem-specific knowledge to the optimization process – and is known to have achieved numerous state-of-the-art results in the past (Lourenço et al., 2003).

The objective of ILS, as described in Lourenço et al. (2003), is to explore the set of possible solutions $\mathcal{S}$ using a stochastic walk that interactively steps from one local optima to another "nearby" local optima in the smaller set $\mathcal{S}^* \subset \mathcal{S}$ of locally optimal solutions.

This stochastic walk is presented in Algorithm 1 and is explained as follows. The optimization process starts by generating an initial feasible solution $s_0 \in \mathcal{S}$ using a problem-specific constructive heuristic. Then, a local search heuristic is applied to s_0 to obtain a local optimal solution $s^* \in \mathcal{S}^*$. Subsequently, the optimization loop (lines 3 – 7) starts with s^* as the current state of the random walk in $\mathcal{S}^*$, from where consecutive calls to the perturbation procedure and the local search are sequentially performed on s^*. Given any current state s^*, a perturbation is applied on s^* that leads to an intermediate state s' (which belongs to $\mathcal{S}$). Then, local search is applied to s', reaching a local optima solution $s^{*\prime} \in \mathcal{S}^*$. If $s^{*\prime}$ passes an acceptance test, it becomes the next element of the walk in $\mathcal{S}^*$; otherwise, the walk returns to s^*. If the acceptance test depends exclusively on the current state of the walk, the walk has no memory. The random walk continues until the termination condition is fulfilled.

Algorithm 1: Pseudo-code for the Iterated local search

Data: Problem data $G = (V, E)$
Result: A feasible solution
1 $s_0 \leftarrow GenerateInitialSolution()$
2 $s^* \leftarrow LocalSearch(s_0)$
3 **repeat**
4 $\quad s' \leftarrow Perturbation(s^*)$
5 $\quad s^{*\prime} \leftarrow LocalSearch(s')$
6 $\quad s^* \leftarrow AcceptanceCriterion(s^*, s^{*\prime})$
7 **until** *termination condition met*;
8 **return** s^*

A detailed description of the components of the proposed ILS is provided in the following sections.

3.2.1 The solution representation

A solution is represented by a tuple $s = (j_1, \ldots, j_{|V_d|})$ of length $|V_d|$ containing tasks $j_i \in V_d$ queued in an explicit scheduling order. A solution s is feasible if each task appears exactly once in s and the task scheduling does not violate the priority constraints. That is, if we define $R(j_i)$ as a function that returns the risk of task j_i, we have $\{j_i | 1 \leq i \leq V_d\} = V_d$ and $R(j_u) \geq R(j_v) - d$ holds for every u-th and v-th elements in s, with $u < v$.

3.2.2 The constructive heuristic

The construction of the initial solution is performed by a greedy heuristic that sequences tasks in a non-increasing order of risk. That is, the tasks with the highest risk attributes are queued before the

tasks with lower risk ones. This is a computationally efficient greedy method that always generates a feasible solution. The constructive heuristic is described in Algorithm 2 and can be performed by a sorting algorithm with a $\mathcal{O}(|V_d|log|V_d|)$ computational complexity.

The constructive phase is equivalent to assuming that the strict priority policy is always enforced, thus, one can provide a quick solution by simply ordering tasks by risk attribute. Nevertheless, the construction phase provides a good starting solution that will be iteractively improved in the optimization loop.

Algorithm 2: Pseudo-code for the constructive heuristic

Data: V_d
Result: a solution s
1 **return** $s = (j_1, \ldots, j_{|V_d|})$, such that $R(j_i) \geq R(j_{i+1})\ \forall 1 \leq i < |V_d|$

3.2.3 The evaluation procedure

The evaluation procedure is a deterministic function $\mathcal{E}(s)$ that evaluates a feasible solution s and calculates the tasks weighted sum of the completion times. It works by scanning the solution, from head to tail, and assigning resources to tasks in the order the tasks appear in s. If, at any given time, more than one resource can be assigned to the same task, the resource that provides the earliest completion time is chosen.

Algorithm 3 presents a pseudo-code for $\mathcal{E}(s)$ that works as following. Initially, a vector of size $|V_d|$ is allocated to store tasks' completion times. It then starts scheduling tasks to available resources, choosing the resource that provides the earliest completion time. The *expected completion time* on line 6 is computed by the sum of the completion time of the previous task processed by the resource and the processing time of the current task, added to the sequence-dependent setup time between the previous and the current task. For instance, suppose k_1 is a resource that just finish processing task i and is now available; if j is the current task to be scheduled, the expected completion time of j, if processed by k_1 is calculated as $C_i + s_{ij} + p_j$.

Algorithm 3: Pseudo-code for the evaluation procedure ($\mathcal{E}$)

Data: a solution s
Result: the weighted sum of completion times
1 $C \leftarrow [C_1, \ldots, C_{|V_d|}]$
2 **for** $l \leftarrow 1$ **to** $|V_d|$ **do**
3 $i \leftarrow j_l$
4 $C_i \leftarrow T$
5 **for** $k \leftarrow 1$ **to** K **do**
6 $C' \leftarrow$ the expected completion time of j_l, if processed by k
7 **if** $C_i > C'$ **then**
8 $C_i \leftarrow C'$
9 **return** $\sum_{i \in V_d} r_i C_i$

3.2.4 The local search and the $\mathcal{N}$ solution neighborhood

The local search, presented in Algorithm 5, is a first-improving heuristic (see, *e. g.*, Anderson (1996)) that searches for local optima in a solution's neighborhood $\mathcal{N}(s)$ defined by the *swap* move:

$$\mathcal{N}(s) = \{swap(s, j_u, j_v) \mid \forall (j_u, j_v) \in V_d^2\} \tag{3.27}$$

The *swap* move, presented in Algorithm 4, is a procedure that changes the scheduling order of two tasks in a solution s if the resulting solution remains feasible according to the priority constraints. The line (1) performs the swap, while in the lines (2) – (5) it is checked if the solution remains feasible. For instance, when the strict policy is considered, it is only feasible to apply *swap* to tasks that have the same risk attributes. On the other hand, when the no priority policy is applied, *swap* can always be employed since the set of priority constraints is empty. Thus, the neighborhood $\mathcal{N}(s)$ may contain up to $\mathcal{O}(|V_d|^2)$ solutions.

The local search begins with a given solution s and iteratively searches for a solution $s^* \in \mathcal{N}(s)$ that has a better objective function than s. If s^* exists, the local search will continue by assigning $s \leftarrow s^*$ and repeating the process until no neighbor s^* outperforms s, which means that a local optimum has been found.

Algorithm 4: Pseudo-code for the swap procedure

Data: a solution s and two candidate tasks j_u, j_v
Result: a solution s' if feasible, s otherwise

```
1 s' ← s, but swap indices of tasks j_u and j_v
2 for u ← 1 to |V_d| − 1 do
3     for v ← v + 1 to |V_d| do
4         if R(j_u) < R(j_v) − d then
5             return s
6 return s'
```

Algorithm 5: Pseudo-code for the local search heuristic

Data: a solution s'
Result: a solution s^*, such that $\mathcal{E}(s^*) \leq \min_{s \in \mathcal{N}(s^*)}\{\mathcal{E}(s)\}$

```
1 foreach s* ∈ N(s') do
2     if E(s*) < E(s') then
3         return LocalSearch(s*)
4 return s'
```

3.2.5 The perturbation phase

The perturbation phase is the process of performing a random step from a local optima solution in $\mathcal{S}^*$ to another solution in $\mathcal{S}$, which will be the new starting point for the local search. The

perturbation, together with the acceptance test will define the next state of the walk in $\mathcal{S}^*$. If the perturbation is too strong, ILS may behave like a random restart, since all or most of the characteristics of the current solution will be lost. On the other hand, if the perturbation is too weak, the search space will be very limited, since the local search will often fall back to a local optimum previously visited in the walk.

In Algorithm 6, the perturbation phase is defined as a procedure that receives a local optimal solution s^* and a strength parameter ($\rho \in [0, 1]$). This parameter is used to control the number of random moves that are introduced in s^*, calculated by the formula $\lfloor |V_d| \times \rho \div 2 \rfloor$. Thus, the higher the value of the strength parameter, the more diversified the resulting s' is. The perturbation procedure uses the same swap move defined in Algorithm 6 between two random indices drawn over the uniform distribution $U[1, |V_d|]$. A random swap is only introduced if the solution remains feasible. The perturbation phase results in an intermediate solution s' that usually is not locally optimal.

Algorithm 6: Pseudo-code for the perturbation procedure

Data: a solution s^*, ρ
Result: a solution s'
1 $s' \leftarrow s^*$
2 **repeat**
3 $\quad u \leftarrow$ a random integer in $U[1, |V_d|]$
4 $\quad v \leftarrow$ a random integer in $U[1, |V_d|]$
5 $\quad s' \leftarrow swap(s', j_u, j_v)$
6 **until** $\lfloor |V_d| \times \rho \div 2 \rfloor$ *swaps are performed*;
7 **return** s'

3.2.6 The acceptance criterion

The acceptance criterion is a memory less function that prioritizes solutions with better objective function values, defined as:

$$Better(s^*, s^{*\prime}) = \begin{cases} s^{*\prime} \text{ if } \mathcal{E}(s^{*\prime}) < \mathcal{E}(s^*) \\ s^* \text{ otherwise} \end{cases} \tag{3.28}$$

In particular, $s^{*\prime}$ is only accepted if it is better than s^*. This criterion performs a strong intensification over diversification, forcing the walk to move to a state with smaller objective function value, which corresponds to a first-improvement descent in $\mathcal{S}^*$.

3.2.7 The termination condition

The termination condition is defined as a maximum number of calls to the evaluation procedure. This number is named the *solution budget* and it is a configurable parameter that can vary from 1 to any large integer. Each time a solution is evaluated, the *solution budged* decreases by 1. Once the solution budget reaches 0, the optimization process will stop and return the current local optima solution s^*. This strategy is used to limit ILS complexity to a certain polynomial in the worst-case,

defined by the solution budget value. It is also particularly interesting for ensuring repeatability of results and fair comparison across methods of different nature (Wang et al., 2015).

3.3 Computational experiments

In order to further investigate the the RCPSP-RP, we conducted computational experiments to measure the impact of different experimental settings on the problem complexity, quality of solution and algorithmic performance with a diverse experimental instance benchmark. The benchmark tests aim precisely at (a) analyzing the scalability of the problem versus different problem instances of increasing order of complexity; (b) assessing the impact of sequence-dependent setup times that may or may not be considered depending on whether we assume large or confined areas, and (c) evaluating the effects of the priority constraints under the three different priority schemes in use, namely strict, moderate and no priority policies.

The remainder of this section is organized as follows: the experimental setup for the benchmark tests is detailed in Section 3.3.1, including the virtual environment where tests were performed, data generation, implementation details and tuning of parameters; and the analysis of results are presented in Section 3.3.2, including a comparison between results obtained using CPLEX and ILS for the two problem scenarios and the different priority schemes under consideration.

3.3.1 Experimental setup

All the benchmark tests were carried out in a Ubuntu (version 18.04.1) machine with a 26 core Intel Xeon Processor (Skylake) CPU @ 2.1Ghz and 288GB available RAM, at the computing facilities of the *Laboratoire d'Informatique, de Traitement de l'Information et des Systèmes* (LITIS).

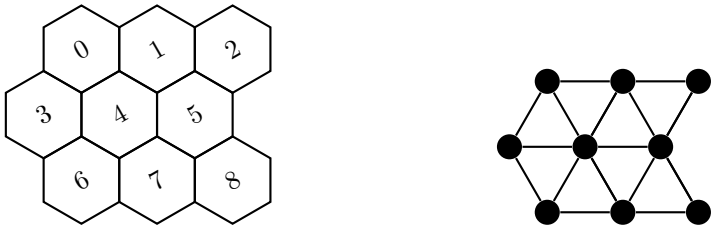
Below, the procedure for generating the benchmark instances, the algorithm implementation details and the parameter tuning are detailed. The source code for the all implementations has been made publicly available under MIT license at Barbalho (2022) with instructions on how to setup and reproduce the results obtained in the benchmark tests.

Data generation

We follow an analogous approach to the one presented in Pena et al. (2023) to propose a theoretical benchmark composed of regular hexagonal grid instances (see Figure 3.6). The objective of using this graph topology is that they provide more precise euclidean distance measures than other planar graph structures (Birch et al., 2007; Mora et al., 2007), which is useful for accurately representing planar terrains in the context of applications in industrial disaster management.

The instance benchmark is designed with the objective of investigating which features render the RCPSP-RP more or less intractable. To this end, we produce instances with an increasing number of tasks and with a variable number of resources. Each of the instances resembles the one illustrated in Figure 3.6. That is, Figure 3.6a represents a planar region mapped as a hexagonal lattice, and Figure 3.6b represents its respective graph topology, where adjacent hexagons are viewed as adjacent nodes.

The procedure to generate the instance benchmark is described as follows.



(a) Hexagonal lattice. (b) Generated graph.

Figure 3.6: Example of a hexagonal grid instance.

1. It starts by generating a hexagonal grid graph with $n = |V_d| + 1$ nodes; the top-left node is set to be the depot; for every other node, the attributes r_i and p_i are individually drawn over the discrete uniform distributions $U(0, 1]$ and $U[1, 10]$, respectively.
2. Subsequently, the sequence-dependent setup times are calculated as the shortest distances between each pair of nodes in the graph; where each edge is weighted by 1.
3. Afterwards, a copy of this instance is created for each value of $K = \{2^0, 2^1, 2^2, ..., 2^{log_2|V_d|}\}$, with each copy getting a different number of available resources at the the depot node.

Table 3.4 presents a summary of the different parameters and values that were used for the generation of such instances. Following this procedure for the values of $|V_d|$ and K present in Table 3.4, the instance benchmark contains 22 unique instances, although it could be easily parameterized to generate other instances at will.

Parameter	Value
graph topology	regular hexagonal grid
number of tasks ($\|V_d\|$)	8, 16, 32, 64
number of resources (K)	from 1 to $\|V_d\|$, in powers of 2
task risk (r_i)	randomly drawn over $U(0, 1]$
task duration (p_i)	randomly drawn over $U[1, 10]$

Table 3.4: Summary of the benchmark instance parameters.

Implementation details and tuning of parameters

The integer models for the first [(3.4) – (3.12)] and second [(3.13) – (3.20), (3.9) – (3.11)] problem scenarios were implemented in Python (version 3.8.10) using the PulP package (version 2.4) and linked to IBM ILOG CPLEX (version 22.1.0.0) via CPLEX's Python bindings. The experiments in CPLEX were carried out with the default parameters and time limit of 4h for every individual instance.

The ILS components were implemented in Python (version 3.8.10), except the local search and the evaluation procedure that were written in C for faster performance and compiled with GCC (version 9.4.0) as a shared library. The experiments with ILS were carried out with the following

parameter configuration: the *solution budget* was set by the formula $|V_d| \times 10,000$, which is large enough to evaluate an extensive number of solutions; since ILS is a stochastic method, a total of 35 repetitions with different random seeds were performed for every instance; and the perturbation strength parameter (ρ) was tuned by the IRACE package as described below.

The IRACE package (López-Ibáñez et al., 2016) is a software that implements the iterated F-race algorithm to automatically find the best parameter settings of an optimizer. The package performs an elitist racing procedure to ensure that the best configurations returned are evaluated in the highest number of training instances.

For tuning ILS perturbation parameter, 5 training instances in which optimal values were known were used as training samples for the two problem scenarios and the three priority policies. The elite configurations returned by IRACE are shown in Table 3.5, these values of ρ were then used to solve the instance benchmark.

Sequence-dependent setup times?	Priority policy		
	ρ (strict)	ρ (moderate)	ρ (none)
Scenario 1: no	0.61	0.19	0.86
Scenario 2: yes	0.50	0.56	0.24

Table 3.5: Best configuration returned by IRACE for ILS perturbation strength parameter according to problem scenario × priority policy.

3.3.2 Analysis of results

In this section we analyze the numerical results for the two problem scenarios obtained by CPLEX and ILS for the benchmark instances under study. From now on, for the sake of simplicity, each individual instance is identified by a tuple $(|V_d|, K)$ representing the number of tasks and the number of resources, respectively.

The complete experimental results are compiled in Tables 3.6 and 3.7 and will be analyzed in dedicated sections further below. These tables can be read as follows. In the case of CPLEX, results show both the best Lower Bound (LB) and, if available, the Upper Bound (UB). As for ILS, the best solution found out of all independent runs is given as well as the mean solution values in all the runs. Solutions known to be optimal are shown in **bold font** in both cases, CPLEX and ILS. A preliminary analysis of these results suggests that:

- In general terms, and not surprisingly, the bigger the instance size is, the more computation time is required by both methods to solve it. For example, in CPLEX, problem complexity can vary from solvable within fraction of seconds to completely untracktable within the time limit of 4 hours. On the other side, ILS presents a polynomial growth in complexity for all instance sizes, while still reliably providing good results.
- Another aspect that influences the problem's tractability is the type of problem scenario in consideration, *i.e.* by adding travel times, optimizing both sequence-dependent setup times and task sequencing adds additional complexity to the scheduling problem.
- In contrast to the previous points, the larger the number of resources (K) is, the easier the

problem instance gets. The specific case $K = |V_d|$ can be trivially solved in $\mathcal{O}(|V_d|)$ by allocating 1 resource to each task.

- Also, the three priority schemes can have different effects on the computational performance of methods:
  - The integer model for the first problem scenario gets easier to solve the less priority constraints there are.
  - In opposition, both the integer model for the second problem scenario and ILS complexity (in both scenarios) increase as the number of priority constraints decreases. A possible explanation is that relaxing these constraints implies exploring a larger solution space.
- Despite the fact that the problem complexity is hardened when relaxing priority constraints, it usually pays off in terms of objective function value, *i. e.* solving an instance with the no priority policy can yield up to -20.87%, in the first scenario, and up to -20.02%, in the second scenario, objective function value than the strict priority policy.
- In all of the instances under study in which CPLEX is able of computing the optimum, ILS is able to yield either the optimal or a near-optimal value with a small percentage deviation. This fact, together with CPLEX incapability to find solutions for the hardest instances, points to the proposed ILS implementation as a good alternative to solve the RCPSP-RP efficiently both in computational and algorithmic terms.

Given the impact that travel times and priority policies have on the numerical results, the following sections provide a more detailed analysis of such experimental scenarios.

To compute the percentage differences between solutions found by CPLEX and ILS, the following formula is used, where S_1 represents the UB found by CPLEX and S_2 represents the best solution found by ILS. Negative percentage gaps indicate cases where ILS provided a better solution than CPLEX.

$$\frac{(S_2 - S_1)}{S_1} \times 100 \tag{3.29}$$

Scenario without sequence-dependent setup times

The first experimental scenario aims at analyzing the results of the benchmark instances without sequence-dependent setup times, *i. e.* we set $s_{ij} = 0\ \forall (i,j) \in V_d^2$. In practice, this scenario can arise, for instance, in the context of industrial disasters contained in self-enclosed spaces, where travel times can be neglected from a practical perspective since processing times are considerable longer. The complete numerical results are compiled in Table 3.6.

In the experiments carried out using CPLEX, out of the 22 instances, the number of obtained optimal solutions was 20, 22 and 22 for the strict, moderate and no priority policy, respectively. For the strict policy, the integer solutions found for the instances $(64, 1)$ and $(64, 2)$ are the only ones not guaranteed to be optimal, with gaps of 0.42% and 0.26% to their best lower bounds, respectively.

The number of optimal solutions obtained by ILS was 20 out of 22 for every different priority policy under study:

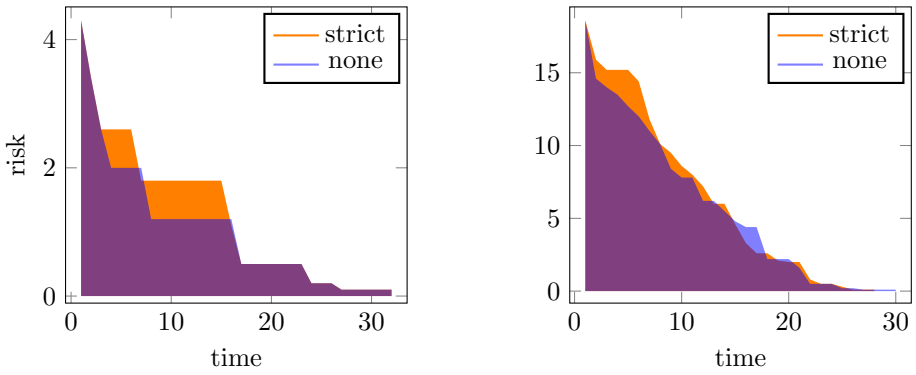
- For the strict priority policy, all the solutions are identical to the ones obtained by CPLEX, including the solutions for the instance $(64, 1)$ and $(64, 2)$, which are not guaranteed to be optimal.
- For the moderate priority policy, the best solutions found for the instances $(64, 1)$ and $(64, 2)$ are the only non-optimal, with percentage deviation of 0.011% and 0.017% to the optimal solutions, respectively.
- For the no priority policy, the best solutions found for the instances $(64, 2)$ and $(64, 4)$ are the only non-optimal, with percentage deviation of 0.021% and 0.005%, respectively.

Even though ILS failed to compute the optimal solutions in the last 4 cases, the obtained solutions are considered near-optimal, since the percentage deviations to the optimal solutions are in the order of hundredths or even thousandths. In absolute values, those errors present an increase of, at maximum, 0.6 in the objective function value.

Instance		Priority policy											
		strict				moderate				none			
		CPLEX		ILS		CPLEX		ILS		CPLEX		ILS	
$\|V_d\|$	K	UB	LB	best UB	mean UB	UB	LB	best UB	mean UB	UB	LB	best UB	mean UB
8	1	**40.1**	40.1	**40.1**	40.1	**33.8**	33.8	**33.8**	33.8	**33.8**	33.8	**33.8**	33.8
	2	**23.2**	23.2	**23.2**	23.2	**21.4**	21.4	**21.4**	21.4	**21.4**	21.4	**21.4**	21.4
	4	**16.3**	16.3	**16.3**	16.3	**16.3**	16.3	**16.3**	16.3	**16.3**	16.3	**16.3**	16.3
	8	**14.8**	14.8	**14.8**	14.8	**14.8**	14.8	**14.8**	14.8	**14.8**	14.8	**14.8**	14.8
16	1	**264.5**	264.5	**264.5**	264.5	**209.3**	209.3	**209.3**	209.3	**209.3**	209.3	**209.3**	209.3
	2	**141.2**	141.2	**141.2**	141.2	**117.3**	117.3	**117.3**	117.3	**117.3**	117.3	**117.3**	117.4
	4	**79.6**	79.6	**79.6**	79.6	**72.5**	72.5	**75.5**	72.5	**72.5**	72.5	**72.5**	72.5
	8	**52.3**	52.3	**52.3**	52.3	**52.0**	52.0	**52.0**	52.0	**52.0**	52.0	**52.0**	52.0
	16	**47.2**	47.2	**47.2**	47.2	**47.2**	47.2	**47.2**	47.2	**47.2**	47.2	**47.2**	47.2
32	1	**1167.1**	1167.1	**1167.1**	1167.1	**977.9**	977.9	**977.9**	977.9	**977.9**	977.9	**977.9**	977.9
	2	**601.8**	601.8	**601.8**	601.8	**516.0**	516.0	**516.0**	516.0	**516.0**	516.0	**516.0**	516.0
	4	**324.3**	324.3	**324.3**	324.3	**285.5**	285.5	**285.5**	285.5	**285.5**	285.5	**285.5**	285.6
	8	**183.3**	183.3	**183.3**	183.3	**172.5**	172.5	**172.5**	172.5	**172.5**	172.5	**172.5**	172.5
	16	**120.1**	120.1	**120.1**	120.1	**118.7**	118.7	**118.7**	118.7	**118.7**	118.7	**118.7**	118.7
	32	**104.1**	104.1	**104.1**	104.1	**104.1**	104.1	**104.1**	104.1	**104.1**	104.1	**104.1**	104.1
64	1	4258.2	4240.2	4258.2	4258.2	**3501.6**	3501.6	3502.2	3504.5	**3486.6**	3486.6	**3486.6**	3486.6
	2	2161.2	2155.5	2161.2	2161.2	**1797.5**	1797.5	1797.7	1798.4	**1791.4**	1791.4	1791.5	1791.5
	4	**1115.8**	1115.8	**1115.8**	1116.5	**947.2**	947.2	**947.2**	947.3	**944.6**	944.6	944.8	944.9
	8	**600.1**	600.1	**600.1**	600.2	**523.7**	523.7	**523.7**	523.8	**522.8**	522.8	**522.8**	522.9
	16	**345.0**	345.0	**345.0**	345.1	**315.7**	315.7	**315.7**	315.7	**315.3**	315.3	**315.3**	315.4
	32	**220.3**	220.3	**220.3**	220.3	**217.9**	217.9	**217.9**	217.9	**217.9**	217.9	**217.9**	217.9
	64	**189.4**	189.4	**189.4**	189.4	**189.4**	189.4	**189.4**	189.4	**189.4**	189.4	**189.4**	189.4

Table 3.6: Results for the problem scenario without sequence-dependent setup times.

Another aspect to analyze is the effect that the priority policy strictness has on the computed overall risk. The strict priority policy is, in general, the one that yields the worst objective function values. Clearly, results indicate that using such policy is not the best operational strategy if one wants to obtain the best solutions possible. Nonetheless, it can be argued that decision makers may impose task prioritization in certain emergency situations, or task priorities cannot easily be ignored. In such cases, the moderate policy seems to be a good alternative to the strict one.



(a) Instance (8, 1).

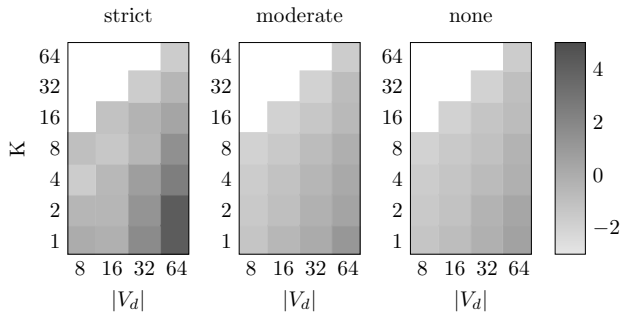
(b) Instance (32, 8).

Figure 3.7: Comparison of optimal solutions for the strict and no priority policies. Purple regions are the overlapping of the solutions in orange and blue. The respective objective functions are (3.7a) 40.1 and 33.8; and (3.7b) 183.3 and 172.5.

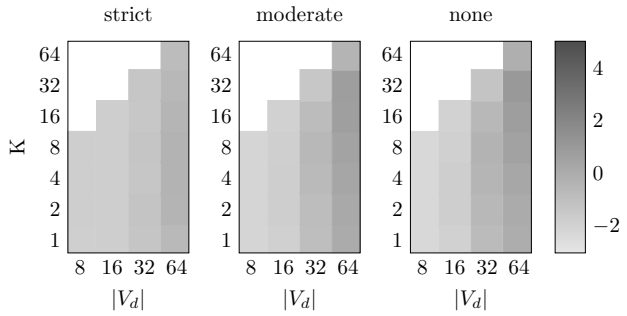
While imposing certain constraints in the order that tasks are processed, the moderate policy is able of generating solutions that are, on average 8.74% (and up to 20.87% in the best case) better than the strict policy. Although, if task prioritization is not required, the no priority policy can yield solutions that are, on average, 8.79% (and also up to 20.87% in the best case) better than the strict policy. The no priority policy is also slightly better than the moderate by 0.06%, on average, and up to 0.43% in the best case. For visualization purposes, Figure 3.7 shows the strict vs. no priority policies comparison for two representative instances. Naturally, the no priority policy yields smaller graph areas, mostly due to a stronger decrease of the overall risk at early periods.

Another important aspect to consider is the running times of the different algorithmic approaches. Typically, when comparing different optimization techniques, the preference is to select the one that, given the same results, takes the shorter time. Moreover, in many cases, a small loss in the quality of the solutions provided by a method can be justified if its running time is substantially shorter than that of another method. This may be the case, for instance, in disaster relief operations where a rapid post-disaster response is usually expected.

In order to compare CPLEX and ILS in terms of their computational performance, Figure 3.8 shows the running times of both algorithmic approaches for the benchmark instances. The plots were obtained by retrieving, for each instance, the time to solution for CPLEX and the average execution time of the 35 independent runs for ILS. The running times are presented in log scale for better visualization. In general, the instances towards the bottom right corner are the hardest to solve. These are the ones with the largest $|V_d| \div K$ ratio. The source data for these plots can be found in Appendix A, Table A.1.



(a) CPLEX's running times.



(b) ILS running times.

Figure 3.8: Color maps representing CPLEX and ILS running times for the scenario without sequence-dependent setup times. The gray scale, from light to dark, stands for the number of seconds (in log scale) employed to find the best solution. Each gray box represents an instance $(|V_d|, K)$.

For CPLEX, instances under the strict policy usually have longer running times than their moderate or none counterparts. In particular, the instances $(64, 1)$ and $(64, 2)$ are the hardest ones for CPLEX. Under strict priorities, they reached the 4 hours time limit while their moderate and none equivalent required respectively 12.45s and 3.36s to complete. These two specific instances required between 0.20s and 1.5s to be solved in ILS, with the shortest running time for the instance $(64, 1)$ under strict priorities and the longest running time for the instance $(64, 2)$ under moderate policy. This signals that the problem scenario becomes easier to solve in CPLEX as the number of priority constraints decreases. For ILS, contrary to CPLEX, the relaxation of priority constraints slightly increases the running times due to the exploration of a larger solution space, which consumes more times during local search.

In the experiments carried out with the moderate and no priority policies, CPLEX and ILS can be considered equivalent in terms of solution quality and computational complexity, since both methods can calculate optimal solutions with similar running times in the order of milliseconds or seconds, depending on the instance size and number of available resources. The only notable difference is for the strict policy, where ILS can provide the same optimal solutions as CPLEX in substantially less time. For more complex scenarios as the one presented in the next section,

CPLEX starts to fail to obtain optimal – or even integer – results while ILS continues to scale polynomially to the instance size, while still reliably providing optimal or near-optimal solutions, when optima are known.

Scenario with sequence-dependent setup times

This second experimental scenario aims at analyzing the results of the benchmark instances with enabled sequence-dependent setup times. In practice, this scenario can arise, for instance, in the context of industrial disasters affecting large areas, where travel times cannot be neglected and must be taken into consideration in the optimization process. The complete numerical results are compiled in Table 3.7.

In the experiments carried out using CPLEX, out of the 22 instances, the number of obtained optimal solutions was 14, 13 and 13 for the strict, moderate and no priority policies, respectively. CPLEX failed to compute 8, 9 and 7 integer solutions for each priority policy, which shows that this scenario is harder to solve than the one without sequence-dependent setup times. Additionally, for the no priority policy, the solutions found for the instances $(16, 1)$ and $(32, 4)$ are not guaranteed to be optimal, with gaps of 55.02% and 27.24% to their respective best lower bounds.

Overall, results show that ILS is able to solve all instances with enabled sequence-dependent setup times, while CPLEX does not. Since CPLEX failed to yield integer solutions for some of the instances and obtained non-optimal results in another two of them, it was not possible to determine optimality for the strict, moderate and no priority policies in 8, 9, and 9 of the instances, respectively. Clearly, by adding sequence-dependent setup times to the optimization process, a large portion of the instances becomes unsolvable by CPLEX within the given time limit of 4 hours and most of the solution from 32 tasks ahead are left without integer solutions. Meanwhile, ILS presents the same polynomial complexity growth shown in the first problem scenario.

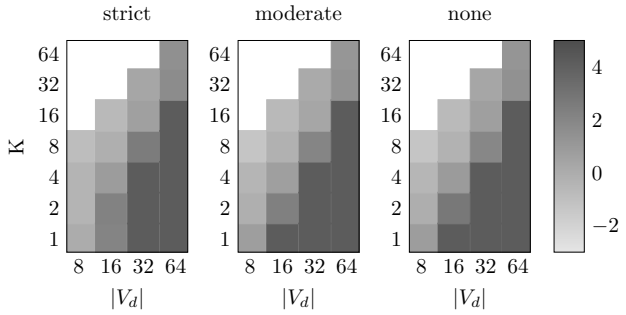
ILS obtains both optimal and near-optimal results in instances with known optimal values and is capable of obtaining competitive results in instances where CPLEX fails to achieve any integer solution. ILS was able to provide a solution in all of these test cases and to outperform the non-optimal integer solutions provided by CPLEX, such as the solutions obtained for the instances $(64, 2)$ and $(64, 4)$ that are, respectively, 3.19% and 10.81% better than those achieved by CPLEX. Additionally, ILS also obtained 11, 11 and 12 optimal solutions for the strict, moderate and no priority policies, respectively. Below we detail the cases in which ILS was not able to attain known optimal solutions:

- For the strict policy, the solutions computed for the instances $(16, 4)$, $(32, 16)$ and $(64, 32)$ are non-optimal with percentage deviation of 0.33%, 0.19% and 0.66% to the optimal values, respectively.
- For the moderate policy, the solutions calculated for the instances $(16, 2)$ and $(64, 32)$ are also non-optimal with percentage deviation of 0.06% and 0.16%, respectively.
- For the no policy, the solution for the instance $(64, 32)$ is non-optimal with a percentage deviation of 0.18%.

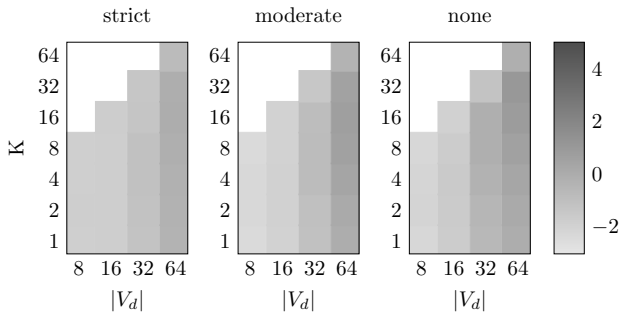
Instance		Priority policy											
		strict				moderate				none			
		CPLEX		ILS		CPLEX		ILS		CPLEX		ILS	
$\|V_d\|$	K	UB	LB	best UB	mean UB	UB	LB	best UB	mean UB	UB	LB	best UB	mean UB
8	1	**77.5**	77.5	**77.5**	77.5	**63.4**	63.4	**63.4**	63.4	**63.4**	63.4	**63.4**	63.4
	2	**45.1**	45.1	**45.1**	45.1	**43.4**	43.4	**43.4**	43.4	**43.4**	43.4	**43.4**	43.4
	4	**33.7**	33.7	**33.7**	33.7	**32.9**	32.9	**32.9**	32.9	**32.9**	32.9	**32.9**	32.9
	8	**28.6**	28.6	**28.6**	28.6	**28.6**	28.6	**28.6**	28.6	**28.6**	28.6	**28.6**	28.6
16	1	**376.6**	376.6	**376.6**	376.6	-	190.8	291.6	291.6	301.2	194.3	291.6	291.7
	2	**202.5**	202.5	**202.5**	202.5	**162.6**	162.6	162.7	165.53	**162.6**	162.6	**162.6**	163.1
	4	**121.2**	121.2	121.6	121.6	**106.6**	106.6	**106.6**	107.4	**106.6**	106.6	**106.6**	106.7
	8	**81.2**	81.2	**81.2**	82.2	**81.1**	81.1	**81.1**	81.1	**81.1**	81.1	**81.1**	81.1
	16	**75.3**	75.3	**75.3**	75.3	**75.3**	75.3	**75.3**	75.3	**75.3**	75.3	**75.3**	75.3
32	1	-	349.5	1667.0	1669.4	-	331.0	1402.0	1410.5	-	381.9	1402.3	1412.0
	2	-	430.7	900.6	903.5	-	364.4	735.5	739.7	-	297.9	735.4	739.4
	4	-	433.1	500.4	502.9	-	368.0	420.9	426.4	471.8	370.8	420.8	424.6
	8	**300.6**	300.6	**300.6**	301.4	**272.1**	272.1	**272.1**	274.4	**272.1**	272.1	**272.1**	273.0
	16	**214.7**	214.7	215.1	215.1	**205.9**	205.9	**205.9**	205.9	**205.9**	205.9	**205.9**	205.9
	32	**185.6**	185.6	**185.6**	185.6	**185.6**	185.6	**185.6**	185.6	**185.6**	185.6	**185.6**	185.6
64	1	-	433.3	6250.1	6333.9	-	421.1	4936.6	5117.1	-	420.0	4987.3	5118.3
	2	-	429.2	3164.6	3202.8	-	417.3	2578.0	2650.3	-	648.5	2634.8	2710.9
	4	-	533.9	1681.4	1698.2	-	417.3	1401.4	1429.3	-	493.0	1403.0	1450.7
	8	-	497.5	979.7	986.2	-	504.4	823.0	838.4	-	490.5	831.1	850.0
	16	-	591.8	615.1	616.6	-	489.6	547.2	553.2	-	542.2	551.4	559.1
	32	**439.3**	439.3	442.2	443.3	**431.0**	431.0	431.7	432.2	**430.5**	430.5	431.3	432.2
	64	**392.6**	392.6	**392.6**	392.6	**392.6**	392.6	**392.6**	392.6	**392.6**	392.6	**392.6**	392.6

Table 3.7: Results for the problem scenario with sequence-dependent setup times.

In order to compare CPLEX and ILS in terms of their computational performance, Figure 3.9 shows the running times of both algorithmic approaches for the different benchmark instances. Instances towards the right and bottom-right corner are hardest ones to solve. These instances present the largest $|V_d| \div K$ ratio. This comparison reinforces the previous conclusion that by enabling sequence-dependent setup times in the optimization process, the problem becomes considerably less tractable for exact solvers such as CPLEX, while the computational performance of ILS scales polynomially. The source data for these plots can be found in Appendix A, Table A.2.



(a) CPLEX's running times.



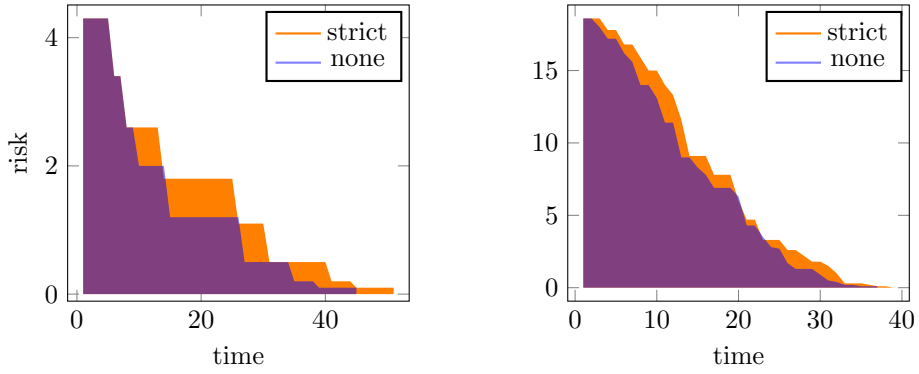
(b) ILS running times.

Figure 3.9: Color maps representing CPLEX and ILS running times for the scenario with sequence-dependent setup times. The gray scale, from light to dark, stands for the number of seconds (in log scale) employed to find the best solution. Each gray box represents an instance $(|V_d|, K)$.

Finally and similarly to the first experimental scenario, the regime of task prioritization has a great effect in the overall risk as illustrated in Figure 3.10. The no priority policy, in general, yields the smallest objective function values, although it can be considered virtually equivalent in terms of solution quality and running times to the moderate priority policy:

- The no policy can yield solutions that are, on average, 6.56% (and up to 20.02% in the best case) better than the strict policy.

- The moderate policy is able of generating solutions that are, on average 5.51% (and up to 19.70% in the best case) better than the strict policy.

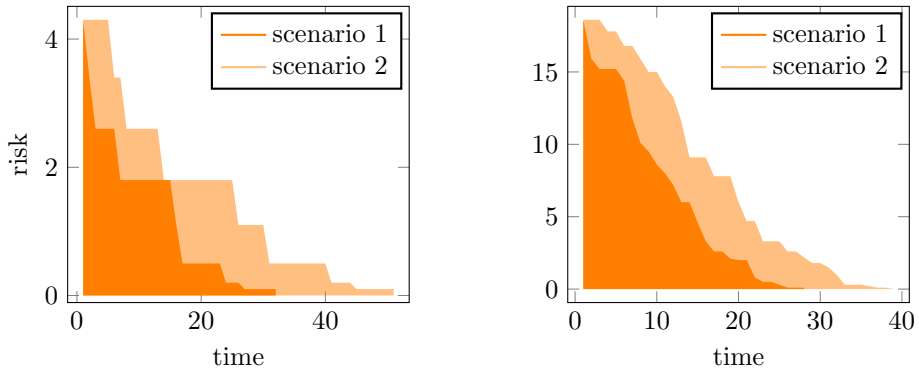


(a) Instance (8, 1). (b) Instance (32, 8).

Figure 3.10: Comparison of optimal solutions for the strict and no priority policies. Purple regions are the overlapping of the solutions in orange and blue. The respective objective functions are (3.10a) 77.5 and 63.4; and (3.10b) 300.6 and 272.1.

Comparison between RCPSP-RP scenarios

One last point worth analyzing is the impact of sequence-dependent setup times in the objective function values. For example, in the context of industrial disasters, how much does the overall risk increase when considering travel times? We may answer this question by looking at the differences in objective function values between the two experimental scenarios in Tables 3.6 and 3.7. These relative differences are presented in Table 3.8 for better visualization. See also Figure 3.11) for a visual comparison of instances (8, 1) and (32, 8) regarding the two problem scenarios, which is representative of the overall instances.



(a) Instance (8, 1). (b) Instance (32, 8).

Figure 3.11: Comparison of optimal solutions between the first (dark orange) and the second (light orange) problem scenarios. The respective objective functions are (3.11a) 40.1 and 77.5; and (3.11b) 183.3 and 300.6.

By analyzing Table 3.8, one may note that sequence-dependent setup times may add up to 100% to

Instance		Priority policy		
$\|V_d\|$	K	strict	moderate	none
8	1	93.27%	87.57%	87.57%
	2	94.40%	102.80%	102.80%
	4	106.75%	101.84%	101.84%
	8	93.24%	93.24%	93.24%
16	1	42.38%	-	43.91%
	2	43.41%	38.62%	38.62%
	4	52.26%	47.03%	47.03%
	8	55.26%	55.96%	55.96%
	16	59.53%	59.53%	59.53%
32	1	-	-	-
	2	-	-	-
	4	-	-	65.25%
	8	63.99%	57.74%	57.74%
	16	78.77%	73.46%	73.46%
	32	78.29%	78.29%	78.29%
64	1	-	-	-
	2	-	-	-
	4	-	-	-
	8	-	-	-
	16	-	-	73.04%
	32	99.41%	97.80%	97.57%
	64	107.29%	107.29%	107.29%

Table 3.8: Comparison between first and second problem scenario: relative increase in objective function values resulted by sequence-dependent setup times. Only optimal solutions were considered.

the overall risk, which is an interesting insight when assessing the extent of an industrial incident. We may know, *a priori*, if an incident can be more or less disastrous based on the extent of the contaminated area. Not only since larger areas may be affected, but also because time to move teams and equipment from one contaminated site to another may account for a valuable operational time.

3.4 Concluding remarks

This chapter presented an extended version of the RCPSP which allows the problem to be adapted to new application domaines where tasks have priorities associated with their risk attribute. The proposed RCPSP-RP is an integrated scheduling and routing optimization model that aims at minimizing the sum of completion times of tasks weighted by their risk factor, and that allows task prioritization employing priority constraints. These constraints are represented as precedence relations disposed as a directed acyclic graph with two particularities: i) they can be relaxed by means of a relaxation threshold parameter and ii) tasks are allowed starting at the same time as their predecessors as long as there are enough available resources. In addition, two versions of the

problem considering or not sequence-dependent setup times are investigated.

In order to conduct a study on the problem's tractability, we performed computational experiments on a theoretical instance benchmark designed on the context of the post-catastrophe operations in the event of industrial disasters application, where a group of specialized teams are deployed to clean up areas contaminated by dangerous substances after an industrial disaster. The contaminated area is mapped as a hexagonal grid graph, where each contaminated node is treated as a task with a associated risk that depends on the hazardous nature of pollutants present on it.

We developed two integer models and an ILS metaheuristic as the optimization methods. These methods were tested on a theoretical benchmark, where two problem scenarios were considered: with or without travel times, which implies the usage or not of sequence-dependent setup times.

A first conclusion that can be drawn from this study is that travel times make the problem far less tractable for the integer models. Particularly in CPLEX, instances with 16 tasks or more start to become unsolvable within the prefixed computing time limit of 4 hours. On the other hand, ILS shows polynomial scalability of the running times, obtaining solutions for every instance in the benchmark in the order of seconds. Furthermore, ILS is able to find optimal or near-optimal solutions in instances with known optimal values, and, in some specific case, better solutions than the ones achieved by CPLEX. Therefore, it is advisable to use a metaheuristic approach such as ILS, specially when tackling large or demanding instances.

A second conclusion concerning the RCPSP-RP is that relaxing the priority constraints, by increasing the value of the relaxation threshold, leads to a much stronger minimization of the overall risk. In particular, imposing priority constraints significantly impairs the results obtained. The more constraints, the smaller the search space and the worse the optimal value. Meanwhile, results show that, partially or completely disabling those constraints is a good strategy to improve the attainable optimal values, with the downside of longer running times.

Overall, this study on the RCPSP-RP shows the capabilities of the problem to address application domaines that take into account risks, priorities, and sequence-dependent context changes or costs. In addition to industrial post-disaster situations, the RCPSP-RP may be applied to other application domaines that involve risk or priority components such as preventive maintenance scheduling, evacuation planning with priorities or humanitarian logistics, etc.

Finally, this study opens several avenues of research and future work in operational research in the field of industrial catastrophes. In the next chapter, we will study a dynamic version of the RCPSP-RP, where task state may change through time. As a case study, we will investigate the application with dynamic transformation of chemicals released into the environment, where site contamination may evolve over time.

Chapter 4

The resource-constrained project scheduling problem with risk and product transformation dynamics

In this chapter, we present the RCPSP-RTD, which is an extended version of the RCPSP-RP model that incorporates product transformation dynamics, chemical transformations, and product degradation. Our primary focus is to examine the dynamics of these transformations. For this purpose, we propose an extension of a specific case of the RCPSP-RP, excluding sequence-dependent setup times and strict/moderate priority rules (see Table 4.1). By doing so, we investigate intricate dynamics of product transformations and degradation within the context of the problem. We consider the potential support for sequence-dependent setup times and alternative priority rules in the concluding remarks. These considerations expand the scope of our research and provide avenues for future studies to explore different aspects of the extended RCPSP-RTD model.

The objective of the RCPSP-RTD remains focused on reducing contamination risk. The model incorporates two types of on-site operations. While inspired by existing scientific literature, the extended model is kept generic, allowing it to handle a wide range of scenarios without specific product parameters. Instead of focusing on individual products, we explore a diverse set of generic scenarios to demonstrate its versatility in addressing real-world scenarios.

This chapter explores the concept of product transformation dynamics and operation speed, where task durations vary based on products' current amount on sites. This dynamic approach to task duration calculations ensures accurate and context-specific scheduling of on-site operations.

In summary, this chapter presents a comprehensive mathematical formulation for optimizing hazardous substance management in industrial disaster scenarios. By utilizing the time-indexed MILP model and considering the dynamic nature of product, the proposed approach enhances the effectiveness and safety of on-site operations. The integration of the optimization framework and heuristic for estimating the operation horizon provides a solution for mitigating risks and optimizing the use of limited resources during hazardous substance management in industrial disaster

	Products				Operations		Constraints	
Problem	Risk	Amount	Transformation	Degradation	Cleaning	Neutralizing	Task priority	Setup times
RCPSP-RP (Chapter 3)	•				•		•	•
RCPSP-RTD	•	•	•	•	•	•		

Table 4.1: Summary of RCPSP-RP and RCPSP-RTD problem features.

scenarios.

4.1 Introduction to risk assessment and parameter estimation

Understanding the evolution and dynamics of chemicals released into the environment is relevant for assessing risks, such as determining transformation and degradation rates of substances. Parameters, like the degradation and transformation rates of products, are often indirectly inferred from experimental data collected in controlled environments. Generally, nonlinear mathematical models are used to fit empirical data and estimate real parameter values, with statistical analysis employed to assess uncertainty. The objective of such models is to ascertain whether degradation rates significantly differ from zero. Although standard nonlinear least-squares methods are commonly employed, they can yield counterintuitive results due to wide or misshapen estimated probability distributions (Görlitz et al., 2011). Below, we review some materials and studies in the literature related to the risk assessment of chemicals, which provide a useful introduction to the field.

John W. Green and Holbech (2018) provide an introduction to fundamental concepts present in toxicology studies, particularly in the context of regulatory risk assessment. The authors covers among others subjects such as randomization, selecting test concentrations or doses, the choice of experimental species, and the challenges of extrapolating results to human toxicity. In addition to design issues, the authors detail the sources of variability and uncertainty in toxicity experiments. Various types of responses observed in toxicity experiments are introduced, with references to specific chapters where statistical analysis methods for each type are developed. The use of historical controls and their relevance to regulatory risk assessment of environmental chemicals is also discussed. A hierarchy of statistical models is presented, providing an overview of the statistical techniques employed in this field of study and throughout the text.

In addition to traditional nonlinear regression, some studies introduced alternative models for estimating transformation parameters from empirical data. In Görlitz et al. (2011), the authors proposed an approach based on Markov-Chain Monte Carlo (MCMC) sampling to estimate the real

probability distribution of model parameters. The effectiveness of this method is demonstrated using three data sets, specifically evaluating degradation of chemicals in soil. The method was applied to several complex scenarios, including kinetic data from compounds with one and five metabolites. Simulated data further confirms the MCMC method's ability to estimate real probability distributions of parameters more accurately than the standard nonlinear least square method. The efficacy of the approach is demonstrated through artificial and real data sets. The MCMC model correctly identifies situations with zero and nonzero degradation rates, providing more reliable and plausible characterizations compared to traditional methods.

In Figure 4.1, we present a schematic representation of the models utilized in Görlitz et al. (2011). The model is designed for scenarios in which a parent substance (the initial substance initiating chemical transformation) can undergo either transformation (metabolization) into another substance or degradation. The resulting product (metabolite) of the parent substance can only undergo degradation. The authors employed such model to study two datasets:

- Two species model with nonzero degradation rate: the first dataset correspond to a parent product p and a resulting product (metabolite) m, where $K_{pm} > 0$, $K_{p*} > 0$ and $K_{m*} > 0$. That is, product p transforms (metabolizes) into product m at a rate K_{pm}. In addition, p and m degrade at rates K_{p*} and K_{m*}, respectively.
- Two species model with zero degradation rate: the second dataset correspond to a parent product p and a resulting product (metabolite) m, such that $K_{pm} > 0$, $K_{p*} > 0$ and $K_{m*} = 0$. That is, product p transforms (metabolizes) into product m at a rate K_{pm}. In addition, p degrades at a rate K_{p*}, but m has a zero degradation rate.

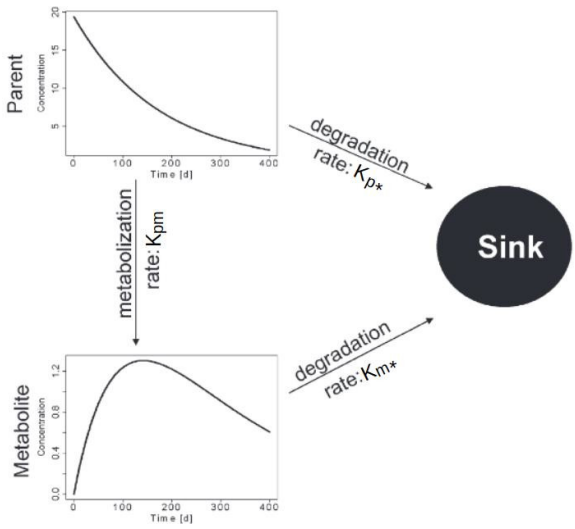


Figure 4.1: Schematic representation of a chemical transformation model. Adapted from Görlitz et al. (2011).

In Murzin et al. (2021), a MCMC model was also proposed for kinectic analysis and parameter estimation of the hydrogenation of toluene to methylcyclohexane in both gas and liquid phases. The study showed that, when dealing with complex reaction mechanisms involving multiple kinetically

significant steps, traditional nonlinear regression provided resulted in poorly defined kinetic parameters, which was also noticed by Görlitz et al. (2011). In contrast, the Bayesian statistics approach allowed for the identification of the most statistically probable values for those parameters.

Overall, the majority of studies in this field deal with the estimation of parameters from empirical data to determine transformation and degradation rates. However, providing an estimation model for transformation and degradation parameters is outside the scope of this study. Our focus is centered on delivering an optimization model able to use these estimated parameters to take into account the transformation dynamics of products, while minimizing risks.

4.2 Problem definition

In the following problem definition, we acknowledge that products may undergo chemical reactions, transforming into other products at specific rates, and that the byproducts can possess varying levels of risk compared to their parent products. In addition, products may experience natural degradation over time, causing their presence to diminish or disperse gradually. For instance, Figure 4.1 illustrates a dynamic transformation scheme involving a parent product A, which simultaneously degrades and undergoes transformation into product B at the rates K_{pm} and K_{p*}, respectively. The product B, on the other side, has no transformation rates but undergoes degradation at a rate K_{m*}. The process of transformation a product A into product B at a rate K_{pm} means that, for each time unit, K_{pm} percent of A's current amount is converted into product B. Similarly, product degradation at a rate K_{p*} indicates that its current amount will diminish by a factor of K_{p*} for each time unit. In general, if $K_{pm} > 0$, the product p will eventually be fully transformed into another substance m; if $K_{p*} > 0$, p's amount will eventually diminish completely, leaving no further traces. In cases where both $K_{pm} = 0$ and $K_{p*} = 0$, the product has zero transformation rate and zero degradation rate, representing the problem presented in the Chapter 3.

Moreover, we have developed an extensive benchmark of data, inspired by the work of Görlitz et al. (2011). The designed instance benchmarks serve two purposes. First, they aim to investigate the hypothesis of risk escalation over time, which entails the transformation of a less risky product into a more hazardous one. Second, they delve into the notion of risk reduction over time, where the transformation of a higher risk product leads to a less risky product. Alongside these hypotheses, we further examine four distinct scenarios concerning product degradation. These scenarios encompass instances where:

1. products undergo no degradation;
2. the parent product features a non-zero degradation rate;
3. the resulting product exhibits a non-zero degradation rate; and finally
4. both the parent and the resulting product possess non-zero degradation rates.

The new formulation detailed further below, introduces a set of decision variables that represent key aspects of the problem. At a glance, continuous variables w_{ip}^t signify the amount of product p on site i at time t, allowing for a dynamic representation of product amount. Pulse binary variables x_{ip}^t and y_i^t capture the occurrence and timings of neutralizing and cleaning operations, respectively. The continuous variables r_{ip}^t, d_{ip}^t, and q_{ipm}^t model the removal, degradation, and transformation of hazardous substances during on-site operations.

Let $P = \{0, 1, ..., N\}$ be the set of products present in the contaminated sites. Each product $p \in P$ is associated with a risk value denoted as $R_p \in [0, 1]$, which represents the hazardous nature of the material. The risk values are categorized based on a scale of low risk ($R_p \leq 0.3$), medium risk ($0.3 < R_p \leq 0.5$), high risk ($0.5 < R_p < 0.9$), or critical risk ($R_p \geq 0.9$). Furthermore, let V be the set comprising the contaminated sites. Each site $i \in V$ is characterized by an initial amount of product $p \in P$, denoted as $W_{ip} \geq 0$. In addition, we introduce the transformation rate of product p into product m, denoted as $K_{pm} \geq 0$, for all $(p, m) \in P \times P$. Finally, we define the degradation rate of each product $p \in P$ as $\bar{K}_{p*} \geq 0$.

The objective is to devise an optimal schedule of on-site operations using a set of heterogeneous resources or teams to either (i) *clean* or (ii) *neutralize* hazardous products at contaminated sites resulting from industrial disasters.

For each contaminated site, the following decisions must be made:

1. Determine the nature of the operation, either (i) fully cleaning the site, which involves physically removing all products present on it, or (ii) neutralizing a specific target product p on the site. The duration of each operation is determined based on the type of operation and its starting time (see Section 4.2.1 for more details).

    - Cleaning a site involves eliminating all hazardous products, resulting in a final concentration of 0 for all products on the target site, as shown in Figure 4.2. This ensures the site is safe and free from hazardous substances. This operation is analogous to the one presented in Chapter 3.

    - A neutralizing operation relies on applying a reactive substance s to neutralize a specific target product p. The objective is to fully neutralize the product p into a safe product, as shown in Figure 4.3, where an example of neutralizing product A is depicted. The result of this operation is the transformation of product A into a product C, which represents a generic safe product. It is essential to note that the reactive substance s only reacts with the target product p. Although the product p is neutralized on the site, there may still be other hazardous products present, which means the site is not guaranteed to be entirely safe.

2. Decide the date in time unit at which the operation will start. This decision is constrained by the availability of resources for each type of on-site operation.

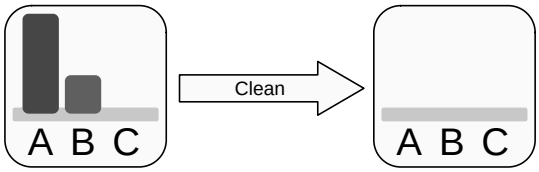


Figure 4.2: Cleaning operation scheme. The rectangle represents a site with products A, B and C before and after a cleaning operation.

The scheduling problem involves determining optimal timings and sequences for conducting on-site operations, aiming to minimize the risks associated with hazardous products as effectively as possible. Essential input data, as presented in Table 4.2, plays a crucial role in addressing this

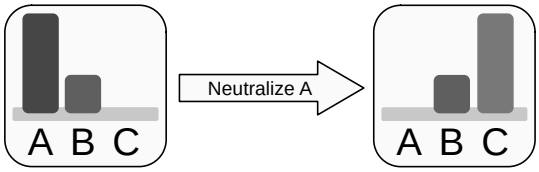


Figure 4.3: Neutralizing operation scheme. The rectangle represents a site with products A, B and C before and after a neutralizing operation. The target product A is transformed into C.

challenging task. The data include the contaminated sites, the types and quantities of dangerous substances involved, the availability of resources, and relevant temporal limitations (see Section 4.3 for details).

The constants $Q \geq 0$ and $\bar{Q} \geq 0$ indicate the number of available resources for conducting cleaning and neutralizing operations, respectively. Each task must be assigned to a single resource, and once it has commenced, interruption or preemption is not permitted. The task must be carried out by the designated resource until its completion.

The task duration attributes are computed using the functions $D(i, p)$ and $\bar{D}(i, p, t)$, which provide estimations of the time required to perform cleaning or neutralizing operations, respectively, based on their starting times. These functions take into account the cleaning or neutralizing speed (Z or $\bar{Z}_p$, respectively) and the expected product amount at a specific time t (see Section 4.2.1 for details).

The constant $T \geq 0$ represents an estimation of the total time required to fully mitigate the risk across all contaminated sites. It is considered as an upper bound since determining its optimal value is computationally as challenging as solving the problem itself (see Section 4.2.2 for details).

The objective function is defined as the minimization of risk across the entire operation horizon. Since each contaminated site $i \in V$ may contain one or more hazardous products, the risk at site i during any time $t \in H$ is computed as follows, with w_{ip}^t representing the amount of product p at time t:

$$\sum_{p \in P} R_p w_{ip}^t \tag{4.1}$$

To encompass all sites $i \in V$ and all time units $t \in H$, we expand Equation 4.1 to define the objective function to minimize the global risk across all contaminated sites and throughout the entire operation horizon.

$$\text{Minimize} \sum_{i \in V} \sum_{t \in H} \sum_{p \in P} R_p w_{ip}^t \tag{4.2}$$

By optimizing the scheduling of cleaning and neutralizing operations, the goal is to effectively mitigate the risk of dangerous products and ensure the safety and well-being of the affected areas.

Input parameter	Description
P	Set of products, representing the hazardous substances present in the contaminated sites
R_p	Risk attribute of product p, indicating the level of risk associated with each hazardous substance
V	Set of sites, representing the contaminated sites that require cleaning or neutralizing operations
W_{ip}	Initial amount of product p on site i, providing the starting level of contamination for each site
T	Operation horizon upper bound, an estimation of the maximum time required to complete all on-site operations effectively
H	Set of time units $t = 0, ..., T$, representing the time periods during the operation horizon
Q	Number of available resources to perform on-site cleaning operations
$\bar{Q}$	Number of available resources to perform on-site neutralizing operations
$D(i,t)$	Time required to clean site i at time t, accounting for the dynamics of products and cleaning speed (see Section 4.2.1)
$\bar{D}(i,p,t)$	Time required to neutralize product p on site i at time t, accounting for the dynamics of products and neutralizing speed (see Section 4.2.1)
K_{pm}	Transformation rate of product p into product m, indicating the rate at which one product is transformed into another
$\bar{K}_{p*}$	Degradation rate of product p, indicating the rate at which the product naturally degrades over time
Z	Cleaning speed, representing the pace at which hazardous substances can be removed from a contaminated site during cleaning operations
$\bar{Z}_p$	Neutralizing speed for product p, representing the rate at which a hazardous substance can be neutralized during neutralizing operations

Table 4.2: Table of problem input data.

4.2.1 Task duration and operation speed

Task duration is intrinsically linked to the current amount of products at a specific time t due to degradation dynamics. If the products' degradation rate is not zero, task durations tend to decrease with time. For example, $D(i,t) \geq D(i,t+1)$, with the rate of decrease determined by the value of the degradation parameters. Similarly, for neutralizing operations, task durations also tend to decrease with time, such that $\bar{D}(i,p,t) \geq \bar{D}(i,p,t+1)$, if product p has a non-zero degradation rate. In other words, operations take less time at later stages of the time horizon if product degradation is

positive. This dynamic nature of task durations reflects the impact of degradation on the hazardous products over time. As the products degrade or are neutralized, the required time for completing on-site operations diminishes, thereby influencing the scheduling decisions.

In Algorithms 7 and 8, we provide pseudo-codes for computing the task duration for cleaning ($D(i,t)$) and neutralizing ($\bar{D}(i,p,t)$) operations for a given site i and product p at time t. Both algorithms simulate the natural dynamics of products to calculate their expected amount at time t, and subsequently determine the time required to fully complete a cleaning or neutralizing task. The computation proceeds through the following steps:

1. Set the initial amount of products (lines 1 – 2, in both Algorithms 7 and 8).
2. Begin simulating the natural evolution of products based on their transformation and degradation rate parameters, fast-forwarding to time unit $t-1$ (lines 3 – 9, in both Algorithms 7 and 8).
3. Compute the required duration to fully complete a cleaning operation, considering the current amount (w_{ip}^t) and operation speed (lines 10 – 22, in both Algorithms 7 and 8).

The main differences between $\bar{D}(i,p,t)$ and $D(i,t)$ are the extra parameter p and the loop termination condition in line 21, which takes in consideration only the target product p. These algorithms provide a direct answer to the question: "How long does it take to complete this task if it starts at time t?" The answer is acquired through a comprehensive simulation of the products' natural behavior, taking into account their degradation and transformation rate parameters, as well as their initial amount and operation speed. This dynamic approach enables context-specific task duration calculations according to products on sites, which are essential for optimizing the scheduling and planning of on-site operations.

4.2.2 Estimating an upper bound for the time horizon

The MILP model's optimization framework leverages the proposed decision variables and constraints to create an effective schedule for on-site operations. However, selecting a suitable value for the operation horizon upper bound (T) plays a vital role in the model's complexity and solvability. To address this, the chapter introduces a heuristic approach for estimating an appropriate T value, considering the longest task duration and the available resources for cleaning operations.

Estimating an appropriate value for the operation horizon (T) significantly impacts the complexity of the optimization methods. In the MILP formulation (see Section 4.3), the number of decision variables directly depends on the constant T. Choosing an inadequate value for T may either render the model unsolvable due to infeasible constraints or an excessively increase the number of decision variables. To address this, we propose the following equation for estimating a suitable upper bound for the operation horizon:

$$T = \max_{i \in V}\{D(i,0)\} + \left\lceil \frac{\sum_{i \in V} D(i,0)}{Q} \right\rceil \tag{4.3}$$

Equation (4.3) consists of the longest task duration added to the ceiling of the ratio between the total processing times and the number of available resources (Q). This estimation approach provides an upper bound for the time horizon, considering the longest task duration and the available resources.

Algorithm 7: Pseudo-code for $D(i,t)$

Data: a site i and a time unit t
Result: the time required to clean i, if task starts at time t

foreach $p \in P$ **do**
  $w_{ip}^0 \leftarrow W_{ip}$
for $\tau \leftarrow 1$ **to** $t-1$ **do**
  foreach $p \in P$ **do**
    $d_{ip}^\tau \leftarrow \bar{K}_{p*} w_p^{\tau-1}$
  foreach $(p,m) \in P \times P$ **do**
    $q_{ipm}^\tau \leftarrow K_{pm}(w_{ip}^{\tau-1} - d_{ip}^\tau)$
    $w_{ip}^\tau \leftarrow w_{ip}^{\tau-1} - q_{ipm}^\tau$
    $w_{im}^\tau \leftarrow w_{im}^{\tau-1} + q_{ipm}^\tau$
repeat
  $\tau \leftarrow t$
  foreach $p \in P$ **do**
    $d_{ip}^\tau \leftarrow \bar{K}_{p*} w_p^{\tau-1}$
  foreach $(p,m) \in P \times P$ **do**
    $q_{ipm}^\tau \leftarrow K_{pm}(w_{ip}^{\tau-1} - d_{ip}^\tau)$
    $w_{ip}^\tau \leftarrow w_{ip}^{\tau-1} - q_{ipm}^\tau$
    $w_{im}^\tau \leftarrow w_{im}^{\tau-1} + q_{ipm}^\tau$
  foreach $p \in P$ **do**
    $w_{ip}^\tau \leftarrow \max\{w_{ip}^{\tau-1} - Z, 0\}$
  $\tau \leftarrow \tau + 1$
until $\exists w_{ip}^\tau \geq 0$ *for any* $p \in P$;
return $\tau - t$

By incorporating this estimated value for T in the MILP formulation, we can manage the model's complexity and improve the solvability of the optimization problem.

4.3 Mathematical formulation

We propose a time-indexed MILP formulation for the RCPSP-RTD model, utilizing the problem data described in Table 4.2 and the following set of decision variables:

- The continuous variables $w_{ip}^t \geq 0$, where $(i,p,t) \in V \times P \times H$, indicate the amount of product p on site i at time t. These variables represent the dynamic evolution of products over the time horizon.
- The pulse binary variables x_{ip}^t, where $(i,p,t) \in V \times P \times H$, are defined such that $x_{ip}^t = 1$ if a neutralizing operation targeting product p starts on site i at time t, and $x_{ip}^t = 0$ otherwise. These variables enable the modeling of neutralizing operations and their starting times.

Algorithm 8: Pseudo-code for $\bar{D}(i,p,t)$

Data: a site i, a target product p and a time unit t
Result: the time required to neutralize product p on site i, if task starts at time t

```
foreach p ∈ P do
    w_ip^0 ← W_ip
for τ ← 1 to t − 1 do
    foreach p ∈ P do
        d_ip^τ ← K̄_{p*} w_p^{τ−1}
    foreach (p, m) ∈ P × P do
        q_ipm^τ ← K_pm (w_ip^{τ−1} − d_ip^τ)
        w_ip^τ ← w_ip^{τ−1} − q_ipm^τ
        w_im^τ ← w_im^{τ−1} + q_ipm^τ
repeat
    τ ← t
    foreach p ∈ P do
        d_ip^τ ← K̄_{p*} w_p^{τ−1}
    foreach (p, m) ∈ P × P do
        q_ipm^τ ← K_pm (w_ip^{t−τ} − d_ip^τ)
        w_ip^τ ← w_ip^{τ−1} − q_ipm^τ
        w_im^τ ← w_im^{τ−1} + q_ipm^τ
    foreach p ∈ P do
        w_ip^τ ← max{w_ip^{τ−1} − Z̄_p, 0}
    τ ← τ + 1
until w_ip^τ ≥ 0;
return τ − t
```

- The pulse binary variables y_i^t, where $(i,t) \in V \times H$, are such that $y_i^t = 1$ if a cleaning operation starts on site i at time t, and $y_i^t = 0$ otherwise. These variables allow the representation of cleaning operations and their start times.
- The continuous variables $r_{ip}^t \geq 0$, where $(i,p,t) \in V \times P \times H$, represent the amount of product p removed from site i at time t. These variables account for the removal of hazardous substances during the cleaning operations.
- The continuous variables $d_{ip}^t \geq 0$, where $(i,p,t) \in V \times P \times H$, denote the amount of product p degraded on site i at time t. These variables model the degradation of hazardous substances over time, similar to a flow of product $p \rightarrow$ sink.
- Continuous variables $q_{ipm}^t \geq 0$, where $(i,p,m,t) \in V \times P \times P \times H$, represent the amount of product p transformed into product m at time t. These variables capture the transformation of products, resembling a flow of product $p \rightarrow m$.

In the subsequent formulation, we designate $p = 0$ to represent the product associated with neutral-

izing operations. This implies that product 0 does not initially exist on any sites, and its appears only upon the execution of a neutralizing operation. Additionally, we assume that product 0 is entirely safe, devoid of any degradation or transformation processes ($R_0 = 0$, $\bar{K}_{0*} = 0$ and $K_{0m} = 0$ for all $m \in P$). Thus, the RCPSP-RTD can be formulated as:

$$\text{Minimize} \quad \sum_{i\in V}\sum_{t\in H}\sum_{p\in P} R_p w_{ip}^t \qquad \text{s.t.} \tag{4.4}$$

$$w_{ip}^0 = W_{ip} \qquad \forall i \in V, \forall p \in P \tag{4.5}$$

$$d_{ip}^t = \bar{K}_{p*} w_{ip}^{t-1} \qquad \forall i \in V, \forall p \in P, \forall t \in H\backslash\{0\} \tag{4.6}$$

$$q_{ipm}^t = K_{pm}(w_{ip}^{t-1} - d_{ip}^t) \qquad \forall i \in V, \forall p \in P, \forall m \in P\backslash\{0\}, \forall t \in H\backslash\{0\} \tag{4.7}$$

$$w_{ip}^t = w_{ip}^{t-1} - d_{ip}^t + \sum_{m\in P} q_{imp}^t - \sum_{m\in P} q_{ipm}^t - r_{ip}^t \qquad \forall i \in V, \forall p \in P, \forall t \in H\backslash\{0\} \tag{4.8}$$

$$r_{ip}^t \leq Z\left(\sum_{\tau=\max\{t-D(i,t)+1,0\}}^{t} y_i^\tau\right) \qquad \forall i \in V, \forall p \in P, \forall t \in H \tag{4.9}$$

$$q_{ip0}^t = \bar{Z}_p w_{ip}^{t-1}\left(\sum_{\tau=\max\{t-\bar{D}(i,p,t)+1,0\}}^{t} x_{ip}^\tau\right) - d_{ip}^t \qquad \forall i \in V, \forall p \in P, \forall t \in H\backslash\{0\} \tag{4.10}$$

$$\sum_{\tau=\max\{t-D(i,t)+1,0\}}^{t} y_i^\tau + \sum_{p\in P}\sum_{\tau=\max\{t-\bar{D}(i,p,t)+1,0\}}^{t} x_{ip}^\tau \leq 1 \qquad \forall i \in V, \forall t \in H \tag{4.11}$$

$$\sum_{i\in V}\sum_{\tau=\max\{t-D(i,t)+1,0\}}^{t} y_i^\tau \leq Q \qquad \forall t \in H \tag{4.12}$$

$$\sum_{i\in V}\sum_{p\in P}\sum_{\tau=\max\{t-\bar{D}(i,p,t)+1,0\}}^{t} x_{ip}^\tau \leq \bar{Q} \qquad \forall t \in H \tag{4.13}$$

$$\sum_{t\in H} y_i^t \leq 1 \qquad \forall i \in V \tag{4.14}$$

$$\sum_{t\in H}\sum_{p\in P} x_{ip}^t \leq 1 \qquad \forall i \in V \tag{4.15}$$

$$x_{ip}^t \in \{0,1\} \qquad \forall i \in V, \forall p \in P, \forall t \in H \tag{4.16}$$

$$y_i^t \in \{0,1\} \qquad \forall i \in V, \forall t \in H \tag{4.17}$$

$$w_{ip}^t \geq 0 \qquad \forall i \in V, \forall p \in P, \forall t \in H \tag{4.18}$$

$$d_{ip}^t \geq 0 \qquad \forall i \in V, \forall p \in P, \forall t \in H \tag{4.19}$$

$$r_{ip}^t \geq 0 \qquad \forall i \in V, \forall p \in P, \forall t \in H \tag{4.20}$$

$$q_{ipm}^t \geq 0 \qquad \forall i \in V, \forall p, m \in P, \forall t \in H \tag{4.21}$$

The objective function (4.4) minimizes the overall risk. Constraints (4.5) quantifies the initial

amount of each product on each site. Constraints (4.6) and (4.7) compute, respectively, the product degradation and transformation. Constraints (4.8) update the products amount based on their dynamic transformations (degradation or transformation) and on-site operations (cleaning or neutralizing). Constraints (4.9) state the amount of product (based on the cleaning speed) that is removed from i at time t if a cleaning operation is active (a cleaning operation is active on site i at time t if one variable $y_i^\tau = 1$ for $\tau \in [\max\{t - D(i,t) + 1, 0\}, t]$). Constraints (4.10) state the amount of p on i that is neutralized into product 0 (the neutral product) at time t if a neutralizing operation targeting p is active (similarly, a neutralizing operation is active on site i at time t if one variable $x_{ip}^\tau = 1$ for $\tau \in [\max\{t - \bar{D}(i,p,t) + 1, 0\}, t]$ for some $p \in P$). Constraints (4.11) determine that two on-site operations cannot overlap on the same site at the same time, while the constraints (4.12) and (4.13) limit the number of active operations to the number of available resources. Constraints (4.14) and (4.15) limit the number of operations at each site. The decision variables are defined from (4.16) to (4.21). This formulation contains $\mathcal{O}(|V \times P \times H|)$ binary variables, $\mathcal{O}(|V \times P^2 \times H|)$ continuous variables and $\mathcal{O}(|V \times P^2 \times H|)$ constraints.

Note that constraints (4.10) are non-linear and can be linearized, with a large constant M, as follows:

$$q_{ip0}^t \geq 0 \qquad \forall i \in V, \forall p \in P, \forall t \in H \quad (4.22)$$

$$q_{ip0}^t \leq \bar{Z}_p w_{ip}^{t-1} - d_{ip}^t \qquad \forall i \in V, \forall p \in P, \forall t \in H \backslash \{0\} \quad (4.23)$$

$$q_{ip0}^t \leq M \left(\sum_{\tau = \max\{t - \bar{D}(i,p,t) + 1, 0\}}^{t} x_{ip}^\tau \right) \qquad \forall i \in V, \forall p \in P, \forall t \in H \quad (4.24)$$

$$q_{ip0}^t \geq \bar{Z}_p w_{ip}^{t-1} - d_{ip}^t + M \left(\sum_{\tau = \max\{t - \bar{D}(i,p,t) + 1, 0\}}^{t} x_{ip}^\tau \right) - M \qquad \forall i \in V, \forall p \in P, \forall t \in H \backslash \{0\} \quad (4.25)$$

4.4 Iterated local search for the RCPSP-RTD

In Chapter 3, we introduced an ILS algorithm that employs a stochastic walk that iteratively moves from one local optima to another "nearby" local optima within the smaller set $\mathcal{S}^* \subset \mathcal{S}$ containing locally optimal solutions.

We recall this stochastic walk a process that begins by generating an initial feasible solution $s_0 \in \mathcal{S}$ using a problem-specific constructive heuristic. Then, a local search heuristic is applied to s_0 to obtain a local optimal solution $s^* \in \mathcal{S}^*$. Subsequently, the optimization loop starts with s^* as the current state of the random walk in $\mathcal{S}^*$, from where consecutive calls to the perturbation procedure and the local search are sequentially performed on s^*. Given any current state s^*, a perturbation is applied on s^* that leads to an intermediate state s' (which belongs to $\mathcal{S}$). Then, local search is applied to s', reaching a local optima solution $s^{*\prime} \in \mathcal{S}^*$. If $s^{*\prime}$ passes an acceptance test, it becomes the next element of the walk in $\mathcal{S}^*$; otherwise, the walk returns to s^*. If the acceptance test solely depends on the current state of the walk, the walk has no memory. The random walk continues until the termination condition is met.

In this section, we present an extended version of the ILS developed in Chapter 3, where some of the original components are adapted to support the novel aspects of the problem, such as: heterogeneous

resources, different types of task, non overlapping tasks and dynamic product transformations. The acceptance criterion and the termination condition are the same as the ones presented in Chapter 3.

4.4.1 The solution representation

A solution is represented by a variable-length list s that contains two types of tasks. We use the notation $J = i$, where $i \in V$, to identify a task representing a cleaning operation on site i, and $\bar{J}_p = i$, where $(p, i) \in P \times V$, to identify a task representing a neutralizing operation on site i, targeting product p. Specifically, a task J directly corresponds to a site i, while a task $\bar{J}_p$ targets product p on site i.

In other words, if we define j_i as the i-th generic task in s (representing either a J or a $\bar{J}_p$) and $|s|$ as the length of s, then $s = \{j_1, j_2, ..., j_{|s|}\}$. A solution s is considered feasible if s does not have duplicated tasks (subject to constraints 4.14 and 4.15).

4.4.2 The constructive heuristic

The construction of the initial solution begins by generating a cleaning task for each site $i \in V$. These tasks are then sorted based on the expected initial risk of the site at $t = 0$. To be precise, we define $\mathcal{R}(i)$ as the function that returns the initial risk present at site i:

$$\mathcal{R}(i) = \sum_{p \in P} R_p W_{ip} \tag{4.26}$$

The sites with the highest initial risk are prioritized and placed ahead of sites with lower initial risks in the list. This greedy approach ensures always ensures solution feasibility.

The constructive heuristic is described in Algorithm 9 and can be implemented using a sorting algorithm with a computational complexity of $\mathcal{O}(|V| \log |V|)$. Despite its simplicity, the construction phase provides a solid starting solution, which will be further improved iteratively in the optimization loop.

Algorithm 9: Pseudo-code for the constructive heuristic

Data: V
Result: a solution s
1 create a list s containing a cleaning task for each site $i \in V$
2 **return** $s = [j_1, \ldots, j_{|s|}]$, such that $\mathcal{R}(j_i) \geq \mathcal{R}(j_{i+1}),\ \forall 1 \leq i < |s|$

4.4.3 The product degradation and transformation schemes

The products' degradation and transformation schemes are modeled using the $\mathcal{D}(t, w)$ and $\mathcal{M}(t, w)$ functions, respectively. Both functions take a time unit t and the current values $w = \{w_{ip}^{\tau} \mid \forall (i, p, \tau) \in V \times P \times H\}$ as parameters and return the updated w_{ip}^{t} values for all $(i, p) \in V \times P$ at time unit t.

The function $\mathcal{D}(t, w)$ is represented in Algorithm 10, and the function $\mathcal{M}(t, w)$ is described in Algorithm 11. Both functions work by applying the degradation and transformation schemes,

respectively, in a manner similar to the constraints (4.6) – (4.8) in Section 4.3. These functions are largely used in the evaluation procedure $\mathcal{E}(s)$ (see Section 4.4.5). The computational complexities of $\mathcal{D}(t,w)$ and $\mathcal{M}(t,w)$ are $\mathcal{O}(|V||P|)$ and $\mathcal{O}(|V||P^2|)$, respectively.

Algorithm 10: Pseudo-code for the degradation scheme ($\mathcal{D}$)

Data: the current time unit t; the values $w_{ip}^{\tau},\ \forall(i,p,\tau) \in V \times P \times H$
Result: the updated values $w_{ip}^{\tau},\ \forall(i,p,\tau) \in V \times P \times H$

```
foreach i ∈ V do
    foreach p ∈ P do
        w_ip^t ← w_ip^{t-1} − K̄_{p*} w_ip^{t-1}
return w_ip^τ, ∀(i,p,τ) ∈ V × P × H
```

Algorithm 11: Pseudo-code for the transformation scheme ($\mathcal{M}$)

Data: the current time unit t; the values $w_{ip}^{\tau},\ \forall(i,p,\tau) \in V \times P \times H$
Result: the updated values $w_{ip}^{\tau},\ \forall(i,p,\tau) \in V \times P \times H$

```
foreach i ∈ V do
    foreach (p,q) ∈ P × P do
        q_ipq^t ← K_pq (w_ip^{t-1} − K̄_{p*} w_ip^{t-1})
        w_ip^t ← w_ip^{t-1} − q_ipq^t
        w_iq^t ← w_iq^{t-1} + q_ipq^t
return w_ip^τ, ∀(i,p,τ) ∈ V × P × H
```

4.4.4 The cleaning and the neutralizing schemes

The cleaning and the neutralizing schemes are modeled using the $\mathcal{Q}(i,t,w)$ and $\bar{\mathcal{Q}}(i,p,t,w)$ functions, respectively. The $\mathcal{Q}(i,t,w)$ function, described in Algorithm 12, takes a site i, the current time unit t, and the variables $w = \{w_{jm}^{\tau} \mid \forall(j,m,\tau) \in V \times P \times H\}$ as parameters. It returns the updated w_{ip}^{t} values for all $p \in P$ at site i for time unit t. On the other hand, the function $\bar{\mathcal{Q}}(i,p,t,w)$ is represented in Algorithm 13. It considers a site i, a target product p, the current time unit t, and the variables $w = \{w_{jp}^{\tau} \mid \forall(j,p,\tau) \in V \times P \times H\}$ as parameters. The function returns the updated w_{ip}^{t} value for site i, product p, at time unit t.

Both aforementioned functions operate similarly to the constraints (4.8), (4.9), and (4.10). The computation complexities of functions $\mathcal{Q}(i,t,w)$ and $\bar{\mathcal{Q}}(i,p,t,w)$ are $\mathcal{O}(|P|)$ and $\mathcal{O}(1)$, respectively. These functions are largely used in the evaluation procedure $\mathcal{E}(s)$ (see Section 4.4.5).

4.4.5 The evaluation procedure

The evaluation procedure is a deterministic function $\mathcal{E}(s)$ that evaluates a feasible solution s and calculates it corresponding overall risk. It works by scanning the solution, from head to tail, and assigning resources to tasks in the order the tasks appear in s.

Algorithm 12: Pseudo-code for the cleaning scheme ($\mathcal{Q}$)

Data: a site i; the current time unit t; the variables $w_{jp}^{\tau},\ \forall(j,p,\tau) \in V \times P \times H$
Result: the updated variables $w_{jp}^{\tau},\ \forall(j,p,\tau) \in V \times P \times H$

```
foreach p ∈ P do
    w_ip^t ← max{0, w_ip^t − Z_p}
return w_jp^τ, ∀(j,p,τ) ∈ V × P × H
```

Algorithm 13: Pseudo-code for the neutralizing scheme ($\bar{\mathcal{Q}}$)

Data: a site i; a product p; the current time unit t; the variables $w_{jm}^{\tau},\ \forall(j,m,\tau) \in V \times P \times H$
Result: the updated variables $w_{jm}^{\tau},\ \forall(j,m,\tau) \in V \times P \times H$

```
q_ip0^t ← Z̄_p w_ip^{t−1} − K̄_{p*} w_ip^{t−1}
w_ip^t ← w_ip^t − q_ip0^t
w_i0^t ← w_i0^t + q_ip0^t
return w_jm^τ, ∀(j,m,τ) ∈ V × P × H
```

Algorithm 14 presents a pseudo-code for $\mathcal{E}(s)$ that works as following. Initially, a vector of size $|V_d|$ is allocated to store tasks' completion times. It then starts scheduling tasks to available resources. Neutralizing operations after cleaning operations are ignored. The computational complexity of $\mathcal{E}(s)$ is $\mathcal{O}(|H||V^2||P^2|)$.

4.4.6 The local search and the $\mathcal{N}$ solution neighborhood

The local search, presented in Algorithm 16, is a first-improving heuristic (see, *e. g.*, Anderson (1996)) that searches for local optima in a solution neighborhood $\mathcal{N}(s)$ defined by the *swap* move:

$$\mathcal{N}(s) = \{swap(s, j_u, j_v) \mid \forall(j_u, j_v) \in V^2\} \tag{4.27}$$

The *swap* move, presented in Algorithm 15, is a procedure that changes the scheduling order of two tasks in a solution s. Thus, the neighborhood $\mathcal{N}(s)$ may contain up to $\mathcal{O}(|V|^2)$ solutions.

The local search begins with a given solution s and iteratively searches for a solution $s^* \in \mathcal{N}(s)$ that has a better objective function than s. If s^* exists, the local search will continue by assigning $s \leftarrow s^*$ and repeating the process until no neighbor s^* outperforms s, which means that a local optimum has been found.

4.4.7 The perturbation phases

As mentioned before in Chapter 3, the perturbation phase is the process of performing a random step from a local optima solution in $\mathcal{S}^*$ to another solution in $\mathcal{S}$, which will be the new starting point for the local search. The perturbation, together with the acceptance test will define the next state of the walk in $\mathcal{S}^*$. If the perturbation is too strong, ILS may behave like a random restart, since all or most of the characteristics of the current solution will be lost. On the other hand, if the

Algorithm 14: Pseudo-code for the evaluation procedure ($\mathcal{E}$)

Data: a solution s
Result: the solution's overall risk

```
w ← {w_ip^t | (i,p,t) ∈ V × P × H}
foreach (i,p) ∈ V × P do
    w_ip^0 ← W_ip
for t ← 0 to T do
    apply degradation scheme D(t,w)
    apply transformation scheme M(t,w)
    j ← next pending task
    if j is a neutralizing task then
        for q̄ ← 1 to Q̄ do
            if resource q̄ is allocated then
                apply neutralizing scheme Q̄(i,p,t,w_ip^τ), with (i,p) being the current task of q̄
            else if j is still pending and j is a neutralizing task and no active operation on site i then
                allocate resource to task j
                apply neutralizing scheme Q̄(j,t,w_ip^τ)
    else
        for q ← 1 to Q do
            if resource q is allocated then
                apply cleaning scheme Q(i,t,w_ip^τ), with i being the current task of q
            else if j is still pending and j is a cleaning task and no active operation on site i then
                allocate resource to task j
                apply neutralizing scheme Q(j,t,w_ip^τ)
return Σ_{t∈H} Σ_{i∈V} Σ_{p∈P} R_p w_ip^t
```

perturbation is too weak, the search space will be very limited, since the local search will often fall back to a local optimum previously visited in the walk.

In this section, we define the $\mathcal{P}_1(s)$ and $\mathcal{P}_2(s)$ perturbation phases as follows:

- $\mathcal{P}_1(s)$, described in Algorithm 17, is a procedure that receives a local optimal solution s and applies multiple swap moves between random indices drawn over the uniform distribution $U[1, |s|]$. The number of randomized swaps is controlled by the strength parameter ($\rho_1 \in [0, 1]$) by the formula $\lfloor |s| \times \rho_1 \div 2 \rfloor$.
- $\mathcal{P}_2(s)$, described in Algorithm 18, is a procedure that receives a local optimal solution s and inserts neutralizing tasks at the head of s. The number of new inserted tasks is controlled by the parameter ($\rho_2 \in [0, 3]$) by the formula $\lfloor \bar{Q} \times \rho_2 \rfloor$. This is the only way that neutralizing operations are inserted in the solution.

Algorithm 15: Pseudo-code for the swap procedure

Data: a solution s and two candidate tasks j_u, j_v
Result: a solution s'
return s, but swap indices of tasks j_u and j_v

Algorithm 16: Pseudo-code for the local search heuristic ($\mathcal{L}$)

Data: a solution s'
Result: a solution s^*, such that $\mathcal{E}(s^*) \leq \min_{s \in \mathcal{N}(s^*)}\{\mathcal{E}(s)\}$
foreach $s^* \in \mathcal{N}(s')$ **do**
  if $\mathcal{E}(s^*) < \mathcal{E}(s')$ **then**
    return $\mathcal{L}(s^*)$
return s'

In both perturbation phases, the higher the value of the ρ_1 and ρ_2 parameters, the more diversified the resulting solutions are. Those perturbation phases result in intermediate solutions that usually are not locally optima.

Algorithm 17: Pseudo-code for the perturbation procedure ($\mathcal{P}_1$)

Data: a solution s^*
Result: a solution s'
$s' \leftarrow s^*$
repeat
  $u \leftarrow$ a random integer in $U[1, |s|]$
  $v \leftarrow$ a random integer in $U[1, |s|]$
  $s' \leftarrow swap(s', j_u, j_v)$
until $\lfloor |s^*| \times \rho_1 \div 2 \rfloor$ *swaps are performed*;
return s'

4.5 Computational experiments

We conducted computational experiments for the RCPSP-RTD in order to measure how different experimental settings affect the solution quality and algorithmic performance using a diverse benchmark of instances based on Görlitz et al. (2011) composed of a two chemical species model generically named p_1 and p_2.

This section is organized as follows: Section 4.5.1 outlines the specifics of the experimental setup for the benchmark tests, encompassing details about the virtual environment utilized for conducting tests, data generation methods, implementation specifics, and parameter fine-tuning. Section 4.5.2 focuses on the analysis of results, providing a comparison between the outcomes obtained using CPLEX and ILS for the two distinct problem scenarios, alongside an evaluation of various priority schemes under consideration

Algorithm 18: Pseudo-code for the perturbation procedure ($\mathcal{P}_2$)

Data: a solution s^*
Result: a solution s'
$s' \leftarrow s^*$
repeat
  create a task $\bar{J}_p = i$, where i and p are randomly chosen
  if $\bar{J}_p$ *not in* s **then**
    insert $\bar{J}_p$ at the beginning of s
until $\lfloor \bar{Q} \times \rho_2 \rfloor$ *neutralizing tasks are inserted*;
return s'

4.5.1 Experimental setup

All the benchmark tests were conducted on a machine running Ubuntu (version 18.04.1), equipped with a 26-core Intel Xeon Processor (Skylake) CPU @ 2.1Ghz and 288GB of available RAM. The computations were carried out at the computing facilities of the LITIS.

Below, we provide a detailed explanation of the procedure used to generate the benchmark instances, the implementation specifics of the algorithm, and the parameter tuning process.

The source code for all implementations has been made publicly available under the MIT license at Barbalho (2022). This repository contains instructions on how to set up the environment and reproduce the results obtained during the benchmark tests.

Data generation

We adopt a similar approach to the one presented in Görlitz et al. (2011) to introduce a diverse benchmark of instances. This benchmark consists of two problem scenarios involving two species models p_1 and p_2, as illustrated in Figure 4.1. Initially, Görlitz et al. (2011) presented two datasets with the following characteristics:

1. **Two-species model with nonzero degradation rate**: In this model, there is a product parent p_1 that undergoes transformation into product p_2 at a rate $K_{12} > 0$. Both products have nonzero degradation rates ($K_{1*} > 0$ and $K_{2*} > 0$).

2. **Two-species model with zero degradation rate**: This model involves a product parent p_1 that undergoes transformation into product p_2 at a rate $K_{12} > 0$. The parent has a nonzero degradation rate ($K_{1*} > 0$), while the resulting product does not degrade ($K_{2*} = 0$).

We extend the existing dataset presented in Görlitz et al. (2011) by incorporating a range of transformation and degradation rate combinations. To this end, we introduce two generic products, denoted as p_1 and p_2, each with specified risk and degradation rates. It is noteworthy that the inherent risk associated with p_1 is greater than that of p_2. Subsequently, two distinct scenarios are investigated: (1) the transformation of product p_2 into product p_1 ($p_2 \rightarrow p_1$) , and (2) the transformation of product p_1 into product p_2 ($p_1 \rightarrow p_2$). Clearly, these scenarios correspond to situations where the risk naturally increases or decreases over time. In addition to investigating two distinct product transformation pathways, we also examine four different product degradation

schemes, specifically:

1. No degradation: the degradation rates of both products are 0.
2. Degradation of the parent product: only the parent product has a nonzero degradation rate.
3. Degradation of the resulting product: only the resulting product has a nonzero degradation rate.
4. Degradation of the parent and resulting products: both products have a nonzero degradation rate.

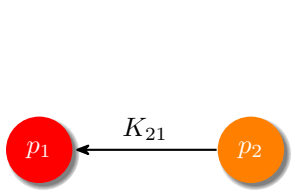
An elaboration of these testing scenarios follows below.

1. **Problem scenario 1 – two-species model with natural increase of risk**: This scenario demonstrates a natural *increase* in overall risk over time since parent p_2 is inherently less risky than product p_1. The test set consists of a four distinct sub-scenarios, illustrated in Figure 4.4. For illustrative purposes, the natural progressions of the overall risk for each degradation scheme are shown in Figure 4.5. These progressions are generated through simulation in the absence of any on-site operations. Notably, the observed trend showcases a tendency for risk to escalate over time due to the dynamics of product p_2 into p_1. Furthermore, the influence of each degradation scheme is evident, as they tend to mitigate the risk through the process of product degradation, thereby contributing to a natural reduction in overall risk.
2. **Problem scenario 2 – two-species model with natural decrease of risk**: This scenario shows a natural *decrease* in the overall risk over time because parent p_1 is inherently riskier than resulting product p_2. The test set comprises a two-species model with four distinct sub-scenarios, illustrated in Figure 4.6, for each degradation scheme. We present the natural progressions of the overall risk for each degradation scheme in Figure 4.7. These progressions are generated through simulation in the absence of any on-site operations. It is important to highlight that the observed trend prominently illustrates a pattern in which the risk diminishes over time due to the dynamics of product p_1 into p_2. Furthermore, the influence of the different degradation schemes becomes evident as they consistently mitigate risk through the degradation process, resulting in a more pronounced reduction in overall risk levels.

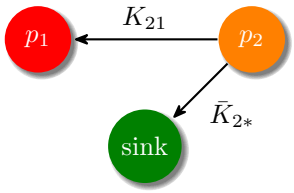
In addition, for each problem scenario, we consider the parameters in Tables 4.3 and 4.4 to be applicable. The number of sites and the number of resources align with the strategic approach employed in Chapter 3. The transformation and degrading rates closely resemble the data sets presented in Görlitz et al. (2011). To create two problem scenarios, we assigned the risk attributes to each product. It is important to note that p_1 and p_2 represent generic products, and their specific risk values may not significantly alter the shape of the risk evolution depicted in Figures 4.5 and 4.7 for each benchmark test case.

Finally, the procedure to generate the instance benchmarks is described as follows.

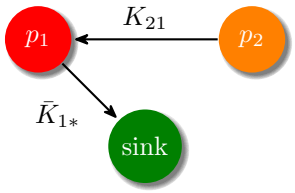
1. Begin by generating an instance in the form of a hexagonal grid graph with $n = |V| + 1$ nodes. The top-left node is designated as the depot.
2. Subsequently, independently draw attributes $W_{ip}, \forall p \in P$, from the discrete uniform distribution $U[0, 1]$, for each node i that is not the depot.



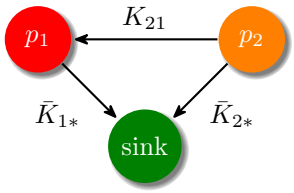
(a) No degradation.



(b) p_2 degradation.



(c) p_1 degradation.



(d) p_1 and p_2 degradation.

Figure 4.4: Product degradation schemes for the problem scenario 1.

3. To conclude, create a copy of this instance for every value of Q within the set $\{2^0, 2^1, 2^2, ..., 2^{\log_2 |V|} - 1\}$. Assign $\bar{Q} = Q$ for each copy and allocate the resources Q and $\bar{Q}$ at the depot node.

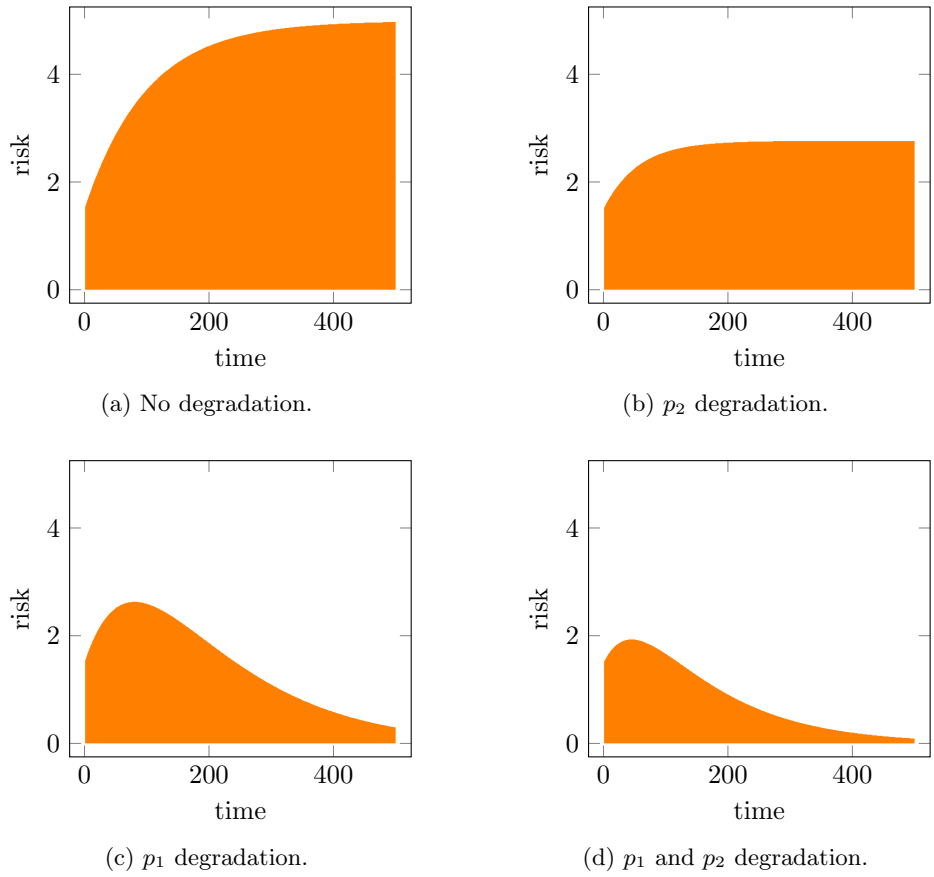
Parameter	Value
Number of sites ($\|V\|$)	8, 16, 32
Number of resources (Q and $\bar{Q}$)	from 1 to $\|V\| - 1$, in powers of 2
Distribution of product initial amount (W_{ip})	$N[0, 1]$
Product risk (R_p)	$R_1 = 1$, $R_2 = 0.5$
Transformation rates (K_{pm})	$K_{21} = 0.05$, $K_{12} = 0$
Degradation rates ($\bar{K}_{p*}$)	$\bar{K}_{1*} = 0.05$ (when applied) and $\bar{K}_{2*} = 0.05$ (when applied)

Table 4.3: Summary of the parameter values for the problem scenario 1.

Implementation details and tuning of parameters

The MILP model [(4.4) – (4.21)] was implemented using Python (version 3.8.10) and leveraged the PulP package (version 2.4). In addition, the model was integrated with IBM ILOG CPLEX (version 22.1.0.0) via CPLEX's Python bindings. For the CPLEX experiments, default parameters were employed, with a time limit of 2 hours assigned for each individual instance.

The ILS was implemented in Python (version 3.8.10), except for the local search and evaluation procedures which were written in C to attain enhanced computational speed. The C components were compiled using GCC (version 9.4.0). For the ILS experiments, a specific parameter configuration was adopted: the *solution budget* was determined using the formula $|V| \times 10,000$, ensuring a substantial evaluation of solutions. A total of 35 repetitions with distinct random seeds were



(a) No degradation.

(b) p_2 degradation.

(c) p_1 degradation.

(d) p_1 and p_2 degradation.

Figure 4.5: Evolution of the overall risk for each product degradation scheme (problem scenario 1).

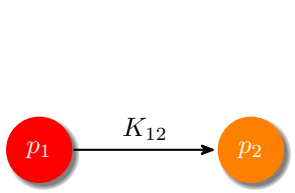
performed for each instance. Furthermore, the perturbation strength parameters (ρ_1 and ρ_2) underwent fine-tuning through the utilization of the IRACE package, as elaborated below.

To fine-tune the perturbation parameters of the ILS method, each problem scenario was partitioned into nine distinct training scenarios. The outcomes of this process, along with the elite configurations yielded by IRACE, are detailed in Table 4.5. Subsequently, the derived values of ρ_1 and ρ_2 were applied to address the instance benchmark.

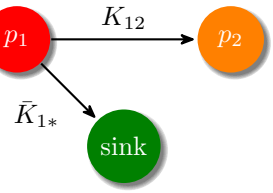
4.5.2 Analysis of results

In this section, we analyze the numerical results obtained by CPLEX and ILS for the two scenarios under study. Initially, we present the results achieved for the **scenario 1 – increasing risk** benchmark in Section 4.5.2, followed by the results for the **scenario 2 – decreasing risk** benchmark in Section 4.5.2. The complete experimental results are compiled in Tables (4.6) – (4.7) and will be analyzed in dedicated sections further below. These tables can be read as follows.

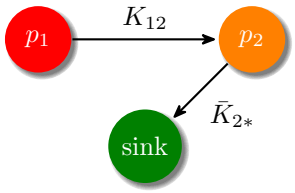
In the case of CPLEX, results show both the Upper Bound (UB) and the gap to the best Lower



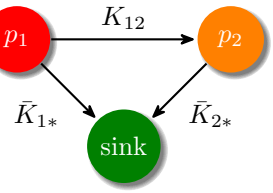
(a) No degradation.



(b) p_1 degradation.



(c) p_2 degradation.



(d) p_1 and p_2 degradation.

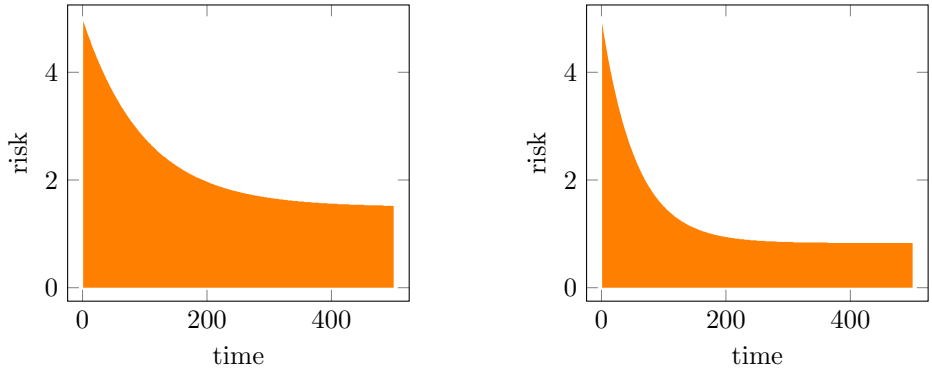
Figure 4.6: Product degradation schemes for the problem scenario 2.

Parameter	Value
Number of sites ($\|V\|$)	8, 16, 32
Number of resources (Q and $\bar{Q}$)	from 1 to $\|V\|-1$, in powers of 2
Distribution of product initial amount (W_{ip})	$N[0,1]$
Product risk (R_p)	$R_1 = 1$, $R_2 = 0.5$
Transformation rates (K_{pm})	$K_{12} = 0.05$, $K_{21} = 0$
Degradation rates ($\bar{K}_{p*}$)	$\bar{K}_{1*} = 0.05$ (when applied) and $\bar{K}_{2*} = 0.05$ (when applied)

Table 4.4: Summary of the parameter values for the problem scenario 2.

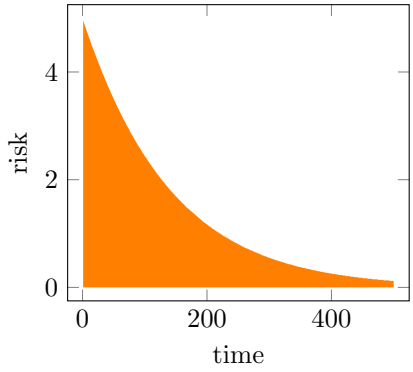
Bound (LB). As for ILS, the best solution found out of all independent runs is given as well as the mean solution values in all the runs. Solutions known to be optimal are shown in **bold font** in both cases, CPLEX and ILS. We consider solutions with a gap of less than 1% relative to their best lower bounds as near-optimal. A preliminary analysis of these results suggests that:

- In a broad sense, and in line with expectations, larger instance sizes invariably demand more computational time from both methods for their resolution. For CPLEX, the instance difficulty is reflected in the observed gaps in relation to the best lower bounds. Instances of smaller dimensions tend to be solved with minimal gaps, whereas larger instances might exhibit gaps reaching as high as 95% when measured against their best lower bounds. CPLEX's capacity to consistently provide solutions stems from the fact that any solution where $y_i^t = 0\ \forall(i,t) \in V \times P$ and $x_{ip}^t = 0\ \forall(i,p,t) \in V \times P \times T$ remains a valid one. However, this trivial solution is expected to deviate substantially from the optimal value. In contrast, ILS demonstrates a polynomial increase in complexity across all instance sizes, while

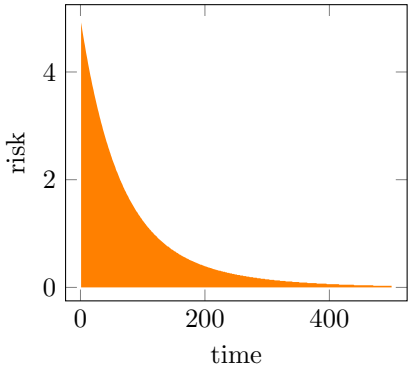


(a) No degradation.

(b) p_1 degradation.



(c) p_2 degradation.



(d) p_1 and p_2 degradation.

Figure 4.7: Evolution of the overall risk for each product degradation scheme (problem scenario 2).

consistently delivering good results.

- The computational complexity of the optimization models does not appear to be significantly influenced by the type of instance or the problem scenario.
- It is evident that as the number of resources (K) increases, the problem instances become relatively easier to solve. This alignment with the observation made in Chapter 3.
- The benchmarks of instances are designed to investigate two primary aspects. First, they seek to examine the hypothesis of risk increasing over time. Second, they intend to explore the hypothesis of risk decreasing with time. In addition to these hypotheses, we also explore four distinct scenarios involving product degradation: instances where (i) products have no degradation, (ii) the parent product possesses a non-zero degradation rate; (ii) the resulting product demonstrates a non-zero degradation rate; and finally, (iii) both the parent and the resulting product have non-zero degradation rate.
- In all the instances under investigation in which CPLEX is able to compute the optimal

Instance size	Problem scenario 1			
	No degradation	p_2 degradation	p_1 degradation	p_1 and p_2 degradation
Small and medium (8 – 16 sites)	$(0.48, 0.46)$	$(0.41, 2.70)$	$(0.36, 1.07)$	$(0.82, 1.71)$
Large (32 sites)	$(0.34, 2.64)$	$(0.32, 1.63)$	$(0.82, 2.66)$	$(0.28, 2.22)$

Instance size	Problem scenario 2			
	No degradation	p_1 degradation	p_2 degradation	p_1 and p_2 degradation
Small and medium (8 – 16 sites)	$(0.25, 2.26)$	$(0.70, 1.31)$	$(0.90, 0.58)$	$(0.38, 2.75)$
Large (32 sites)	$(0.69, 2.01)$	$(0.39, 1.96)$	$(0.71, 2.03)$	$(0.54, 1.86)$

Table 4.5: Best configuration returned by IRACE for ILS' perturbation strength parameters for the two problem scenarios. Tuples correspond to the values of (ρ_1, ρ_2).

solution, ILS produces either the optimal or a closely approximate value, with a minimal percentage deviation. This highlights the effectiveness of the proposed ILS implementation as a viable alternative for efficiently solving the RCPSP-RTD.

- To conduct a comparative analysis of the computational performance of CPLEX and ILS, the Figure 4.8 and 4.9 illustrate the running times of both algorithms across the problem scenarios and degradation schemes. The plots were built by collecting the time to solution for CPLEX and the average execution time of the 35 independent runs for ILS, for each respective instance. The presentation of running times is shown on a logarithmic scale to enhance visual clarity. In a general context, instances positioned in the lower right corner of the plots are the most challenging to solve. These instances exhibit a higher ratio of the number of sites divided by the number of resources, meaning a larger number of sites relative to the available teams.

In the following sections, for the sake of simplicity, each individual instance is identified by a tuple $(|V|, Q)$ representing the number of sites and the number of resources, respectively. It is essential to recall that $Q = \bar{Q}$.

To compute the percentage differences between solutions found by CPLEX and ILS, the following formula is used, where S_1 represents the UB found by CPLEX and S_2 represents the best solution found by ILS. Negative percentage gaps indicate cases where ILS provided a better solution than CPLEX.

$$\frac{(S_2 - S_1)}{S_1} \times 100 \tag{4.28}$$

Results for the Scenario 1 – natural increase of risk

The initial experimental benchmark is formulated to examine the hypothesis of risk escalating as time progresses. This situation can occur in practical scenarios where the resulting product is more hazardous than the parent product. In such cases, the chemical transformations alone can lead to an elevation in risk if timely operations are not executed following an incident. We aim to investigate the influence of various degradation schemes on solution quality, organizing them from the least favorable to the most favorable scenario. In addition, we intend to evaluate whether the degradation scheme can potentially stop the increase in the overall risk. We provide the complete numerical results in Table 4.6.

The examination of the numerical experiments yields the following conclusions, regarding the impact of the degradation schemes. When considering only the solutions with 20% or less gap to the best lower bound, the results indicate that the four degradation schemes play an import role in overall risk. Looking further into the numerical results, one can see that the "no degradation" scheme is the one that generated the higher overall risk, followed by, in descending order, by the "p_2 degradation", "p_1 degradation" and "p_1 and p_2 degradation." This trend becomes particularly evident when observing the area, as depicted in Figures 4.4. The visual analysis of these graphics allows for a direct assessment of the relative risk levels associated with each degradation scheme. In a broad sense, the "no degradation" scheme is the most unfavorable scenario, resulting in an anticipated increase in risk of up to 5% compared to the "p_2 degradation" scheme—the second least favorable scenario. Moreover, this "no degradation" scenario is expected to yield risk levels as much as 15% higher than the best possible scenario represented by the "p_2 and p_1 degradation" scheme.

In the experiments carried out using CPLEX, out of the 12 instances, the number of obtained optimal solutions was 7, 7, 6 and 6 for the "no degradation," "p_2 degradation," "p_1 degradation" and "p_2 and p_1 degradation," respectively.

The number of optimal solutions obtained by ILS was 6, 5, 5 and 5 out of the 12 under the "no degradation," "p_2 degradation," "p_1 degradation" and "p_2 and p_1 degradation," respectively. Under the "no degradation" benchmark, ILS outperformed CPLEX for the instances $(16, 1)$, $(32, 1)$, and $(32, 2)$, yielding deviations of -0.62%, -21.45%, and -8.87%, respectively. It is important to note that the optimality of these solutions remains uncertain. A similar trend is observed in these instances across the other degradation schemes. This phenomenon is consistent with observations made in Chapter 3, where ILS demonstrated its ability to provide better solutions for instances that are particularly challenging for exact methods.

- For the "p_2 degradation" scheme, the deviations for the mentioned instances are of -0.17%, -36.35% and -8.60%, respectively.
- For the "p_1 degradation" scheme, the deviations for the mentioned instances are of 1.29%, -25.41% and -7.72%, respectively.
- For the "p_2 and p_1 degradation" scheme, the deviations for the mentioned instances are of -0.76%, -17.13% and -14.42%, respectively.

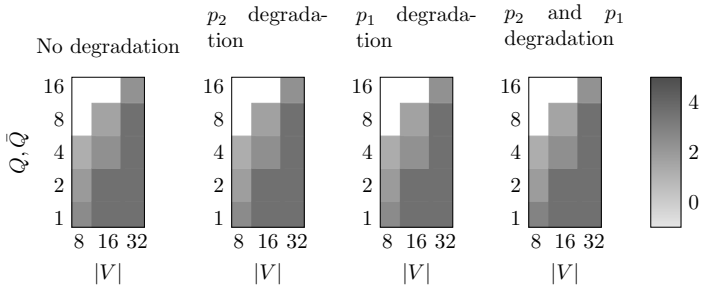
When comparing the various degradation schemes with respect to their computational running times, we observed no discernible impact of the degradation scheme on the overall instance complexity. No particular scheme proved to be more challenging to solve than the others. Instead, the

Instance		Benchmark							
		No degradation				p_2 degradation			
		CPLEX		ILS		CPLEX		ILS	
$\|V\|$	$Q, \bar{Q}$	UB	%gap	best UB	avg UB	UB	%gap	best UB	avg UB
8	1, 1	**90.06**	-	**90.06**	96.41	**83.86**	-	**83.86**	86.27
	2, 2	**39.68**	-	**39.68**	41.80	**38.25**	-	**38.25**	38.96
	4, 4	**19.77**	-	**19.77**	20.68	**19.40**	-	**19.40**	20.26
16	1, 1	479.86	19.21%	476.89	481.89	431.00	13.91%	430.25	474.70
	2, 2	217.86	5.86%	227.37	239.50	204.75	4.05%	211.71	227.96
	4, 4	**98.96**	-	**98.96**	112.24	**96.57**	-	96.89	108.52
	8, 8	**59.08**	-	**59.08**	63.46	**57.50**	-	**57.50**	61.63
32	1, 1	3,479.82	66.25%	2,733.37	3,082.93	3,198.99	66.65%	2,030.26	2,399.40
	2, 2	1,395.55	50.98%	1,271.70	1,478.39	1,269.45	49.71%	1,160.25	1,288.03
	4, 4	572.73	6.87%	588.01	657.35	537.63	5.85%	575.50	667.34
	8, 8	**249.49**	0.98%	259.22	304.37	**241.66**	0.51%	248.09	258.54
	16, 16	**136.18**	-	**136.18**	166.09	**133.48**	-	**133.48**	144.77

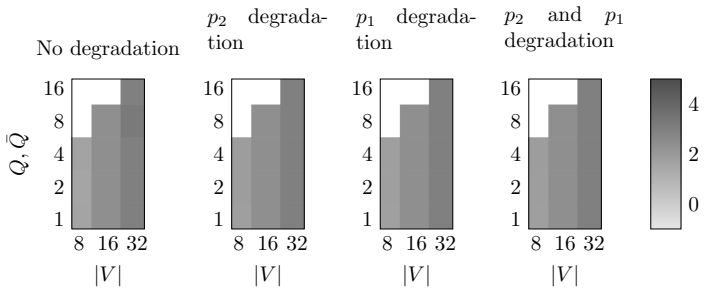
Instance		Benchmark							
		p_1 degradation				p_2 and p_1 degradation			
		CPLEX		ILS		CPLEX		ILS	
$\|V\|$	$Q, \bar{Q}$	UB	%gap	best UB	avg UB	UB	%gap	best UB	avg UB
8	1, 1	**85.08**	-	**85.08**	87.40	**79.63**	-	**79.63**	81.30
	2, 2	**38.65**	-	**38.65**	39.33	**37.24**	-	**37.24**	37.87
	4, 4	**19.43**	-	**19.43**	20.19	**19.06**	-	**19.06**	19.72
16	1, 1	409.66	13.58%	414.94	431.76	374.33	8.71%	371.49	394.09
	2, 2	202.97	4.40%	208.77	221.74	191.75	3.29%	199.51	210.57
	4, 4	**96.30**	-	96.95	108.38	**93.49**	-	93.52	104.61
	8, 8	**57.59**	-	**57.59**	62.87	**56.52**	-	**56.52**	60.65
32	1, 1	2,187.91	53.58%	1,635.64	1,720.44	1,732.08	46.59%	1,435.35	1,507.81
	2, 2	1,131.75	35.60%	1,043.49	1,124.92	1,016.26	25.92%	869.65	984.69
	4, 4	531.25	6.84%	565.74	633.96	503.42	6.09%	507.16	556.10
	8, 8	242.40	1.51%	272.60	321.56	234.06	1.19%	245.20	275.40
	16, 16	**134.00**	-	**134.00**	151.25	**131.32**	-	**131.32**	152.56

Table 4.6: Results for the problem scenario 1 (no degradation, p_2 degradation, p_1 degradation, p_2 and p_1 degradation benchmarks).

complexity of the problem appears to be primarily contingent upon the instance's size, specifically the number of sites and the number of available teams. In this context, it becomes evident that as the number of sites increases, the problem instance becomes more intricate to solve. Conversely, as the number of available teams increases, the instance's complexity diminishes. These findings align with the outcomes of the experiments conducted in Chapter 3.



(a) CPLEX's running times.



(b) ILS' running times.

Figure 4.8: Color maps representing CPLEX and ILS running times for the problem scenario 1. The gray scale, from light to dark, stands for the number of seconds (in log scale) employed to find the best solution. Each gray box represents an instance.

Results for the Scenario 2 - natural decrease of risk

The second experimental benchmark is formulated to examine the hypothesis of risk deescalating as time progresses. This situation can occur in practical scenarios where the resulting product is less hazardous than the parent product. In such cases, the chemical transformations alone can lead to a natural decrease in the overall risk even when on-site operations are not executed following an incident. We aim to investigate the influence of various degradation schemes on solution quality, organizing them from the least favorable to the most favorable scenario. In addition, we intend to evaluate whether the degradation scheme can potentially accelerate the decrease in the overall risk. We provide the complete numerical results in Table 4.7.

The numerical experiments suggest the following conclusions, regarding the impact of the degradation schemes. When considering only the solutions with 20% gap or less to the best lower bound, the degradation schemes seem to play an important role in the overall risk of solutions. As a clear also observed in the first problem scenario, the "no degradation" scheme is the one that generates the largest overall risk, followed by, in descending order, the "p_1 degradation", "p_2 degradation" and "p_1 and p_2 degradation." This trend becomes particularly evident when observing the area under the generated graphics, as depicted in Figures 4.6. The visual analysis of these graphics allows for an assessment of the risk levels associated with each degradation scheme. In a broad sense, the "no degradation" scheme is consistently regarded as the most unfavorable scenario, resulting in an

Instance		Benchmark							
		No degradation				p_1 degradation			
		CPLEX		ILS		CPLEX		ILS	
$\|V\|$	$Q, \bar{Q}$	UB	%gap	best UB	avg UB	UB	%gap	best UB	avg UB
8	1, 1	**103.22**	-	**103.22**	105.43	**96.00**	-	**96.00**	98.20
	2, 2	**47.72**	-	**47.72**	49.09	**45.93**	-	**45.93**	47.36
	4, 4	**23.53**	-	**23.53**	23.89	**23.20**	-	**23.20**	23.68
16	1, 1	381.53	10.77%	381.07	408.82	344.84	8.40%	353.41	375.25
	2, 2	184.69	3.05%	187.69	202.08	**175.39**	0.84%	184.75	195.68
	4, 4	**87.41**	-	89.92	98.67	**85.81**	-	88.77	96.98
	8, 8	**55.35**	-	56.81	59.39	**54.31**	-	**54.31**	58.01
32	1, 1	3,762.53	73.99%	2606.92	2,707.67	1,977.77	56.22%	1,939.78	2275.12
	2, 2	1,064.28	35.29%	989.56	1,139.99	974.00	29.74%	922.99	1043.87
	4, 4	511.38	10.64%	514.98	521.01	475.92	7.81%	484.80	587.02
	8, 8	**229.74**	0.79%	233.17	246.68	**223.95**	0.68%	229.43	241.01
	16, 16	**127.15**	-	129.63	131.20	**124.79**	-	128.62	130.99

Instance		Benchmark							
		p_2 degradation				p_1 and p_2 degradation			
		CPLEX		ILS		CPLEX		ILS	
$\|V\|$	$Q, \bar{Q}$	UB	%gap	best UB	avg UB	UB	%gap	best UB	avg UB
8	1, 1	**98.84**	-	**98.84**	102.22	**92.28**	-	**92.28**	93.85
	2, 2	**47.36**	-	**47.36**	47.44	**45.10**	-	**45.10**	46.23
	4, 4	**23.21**	-	**23.21**	23.64	**22.93**	-	**22.93**	23.49
16	1, 1	330.97	8.48%	329.38	355.79	301.52	7.83%	302.67	325.11
	2, 2	171.96	1.16%	172.68	189.12	164.50	1.46%	164.93	175.00
	4, 4	**84.06**	-	**84.06**	96.34	**82.55**	-	84.43	91.47
	8, 8	**54.04**	-	**54.04**	58.43	**53.01**	-	54.45	56.28
32	1, 1	1,563.63	45.82%	1,294.96	1428.62	1,316.12	42.26%	1,195.77	1377.45
	2, 2	898.45	24.25%	816.42	1056.75	835.45	21.19%	764.37	829.65
	4, 4	459.86	7.49%	475.87	525.19	434.64	6.39%	461.49	511.23
	8, 8	**220.17**	0.73%	225.80	237.51	**215.14**	0.53%	225.35	233.98
	16, 16	**123.74**	-	139.01	129.32	**121.74**	-	126.03	129.37

Table 4.7: Results for the problem scenario 2 (no degradation, p_1 degradation, p_2 degradation, p_1 and p_2 degradation benchmarks).

anticipated increase in risk of up to 10% compared to the "p_1 degradation" scheme – the second least favorable scenario. Moreover, this "no degradation" scenario is expected to yield risk levels as much as 14% higher than the best possible scenario represented by the "p_1 and p_2 degradation" scheme.

In the experiments carried out using CPLEX, out of the 12 instances, the number of obtained optimal solutions was 7, 8, 7 and 7 for the "no degradation," "p_1 degradation," "p_2 degradation" and "p_1 and p_2 degradation," respectively. This suggests that the "decreasing risk" scenario is

comparatively more tractable to solve than the initial problem scenario. This conclusion is drawn from the fact that the solution gaps are narrower and the count of attained optimal solutions is higher than in the former scenario.

The number of optimal solutions obtained by ILS was 6, 5, 5 and 5 out of the 12 under the "no degradation," "p_1 degradation," "p_2 degradation" and "p_1 and p_2 degradation," respectively. Under the "no degradation" benchmark of instances, ILS outperformed CPLEX for the instances $(16, 1)$, $(32, 1)$, and $(32, 2)$, yielding deviations of -0.09%, -30.71% and -7.30%, respectively. It is important to note that the optimality of these solutions remains uncertain. A similar trend is observed in these instances across the other degradation schemes. This phenomenon is consistent with observations made in Chapter 3, where ILS demonstrated its ability to provide better solutions for instances that are particularly challenging for exact methods.

- For the "p_1 degradation" scheme, the deviations for the mentioned instances are of 2.48%, -7.05% and -5.24%, respectively.
- For the "p_2 degradation" scheme, the deviations for the mentioned instances are of -0.48%, -17.18% and -9.13%, respectively.
- For the "p_1 and p_2 degradation" scheme, the deviations for the mentioned instances are of 0.38%, -9.14% and -8.50%, respectively.

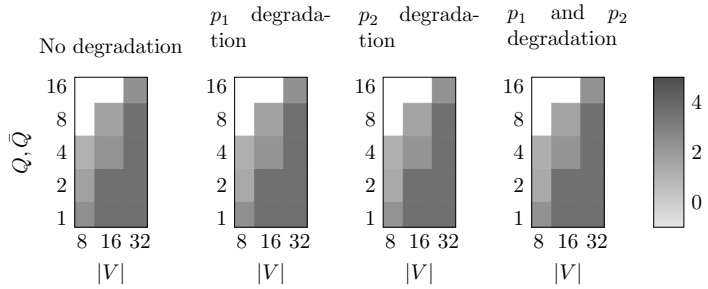
When comparing the various degradation schemes with respect to their computational running times, we observed no discernible impact of the degradation scheme on the overall problem complexity. No particular scheme proved to be more challenging to solve than the others. Instead, the complexity of the problem appears to be primarily contingent upon the instance's size, specifically the number of sites and the number of available teams. In this context, it becomes evident that as the number of sites increases, the problem instance becomes more intricate to solve. Conversely, as the number of available teams increases, the instance's complexity diminishes. These findings align with the outcomes of the experiments conducted in Chapter 3.

4.6 Concluding remarks

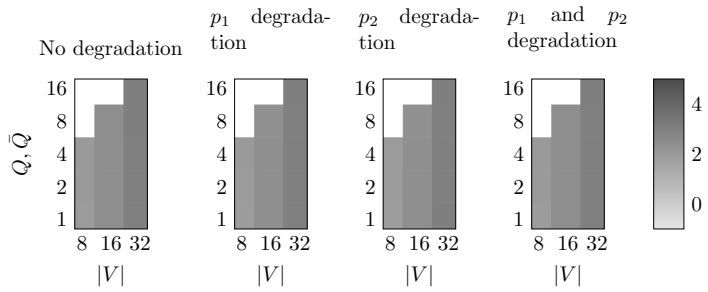
In this chapter, the new RCPSP-RTD is proposed as novel contribution at the intersection of Operations Research and Chemical Kinetics, serving as a framework for optimizing risk mitigation strategies while considering resource allocation and product transformation dynamics.

Through an extensive numerical analysis, we investigated the impact of various degradation schemes on the overall risk levels in two scenarios: one where the risk increases with time and the other where the risk decreases naturally with time. The results show a significant influence of degradation schemes on risk levels. Notably, the no degradation scheme resulted in the highest overall risk, followed by the degradation of the parent product, the degradation of the resulting product and, finally, by the degradation of the parent and resulting products. The visualization of the risk graphics further confirmed these trends, demonstrating that the "no degradation" scenario can lead to risk levels up to 10% higher than the degradation of the parent product and as much as 14% higher than the optimal parent and resulting products degradation schemes.

The findings of this study also suggest that the type of degradation scheme does not significantly contribute to the computational complexity of the instances. Instead, the complexity of the in-



(a) CPLEX's running times.



(b) ILS' running times.

Figure 4.9: Color maps representing CPLEX and ILS running times for the problem scenario 2. The gray scale, from light to dark, stands for the number of seconds (in log scale) employed to find the best solution. Each gray box represents an instance.

stances seems to be predominantly influenced by the instance's size, specifically the number of sites and the available teams. In particular, when employing CPLEX, instances with 16 sites or more tend to exhibit substantial gaps to the best lower bounds within the 4-hour time limit, rendering the determination of optimal solutions challenging.

In contrast, ILS demonstrates a polynomial scalability of running times, successfully providing solutions for all instances in the benchmark. ILS often achieves optimal or near-optimal solutions. In some specific cases, ILS even surpasses the solutions obtained by CPLEX. This happens since the heuristics embedded into CPLEX are not dedicated and cannot use specific information of the problem, particularly when dealing with larger or more complex instances. In light of these findings, it is recommended to consider the use of ILS or similar metaheuristic for instances that pose greater computational demands.

In summary, this study contributes a novel approach to optimizing risk mitigation strategies in the context of industrial accidents involving hazardous substances. While the model and the experiments were centered around simulated data, it offers an efficient and generic computational framework for addressing complex risk factors in resource-constrained scheduling problems. The study of various degradation schemes highlights the importance of considering product transformations when assessing risk levels. Further research avenues include investigating more detailed kinetic models and refining the algorithmics to enhance the accuracy and efficiency of risk mitigation

solutions.

Chapter 5

Conclusions and future research directions

In this thesis, we have explored complex scheduling problems involved with risk factors arising from the release of hazardous substances. The primary goal of this research has been to provide effective algorithms and solutions that address the intricate challenges faced in various industries and logistical operations after disasters involving the release of dangerous substances into the environment.

The RCPSP-RP, presented in Chapter 3, stands as a significant step forward in scheduling optimization for risk reduction after disasters involving dangerous substances. The proposed model integrates risk assessment and different levels of task prioritization, offering an approach to scheduling cleaning tasks in the presence of risk hazards.

In order to conduct a study on the model and method performances, computational experiments on a benchmark of instances were conducted. The instances were designed on the context of the post-catastrophe operations in the event of industrial disasters application, where a group of specialized teams are deployed to clean up areas contaminated by dangerous substances after an industrial disaster. The contaminated area is mapped as a hexagonal grid graph, where each contaminated node is treated as a task with a associated risk that depends on the hazardous nature of pollutants present on it. We developed two integer models and an ILS metaheuristic as the optimization methods. These methods were tested on the benchmark of instances, where two scenarios were considered: with or without travel times, which implies the usage or not of sequence-dependent setup times.

This study yields several conclusions. First, travel times significantly increase the computational complexity of the integer models, particularly noticeable in CPLEX. Instances containing 16 tasks or more start to become computationally infeasible within the 4-hour time limit set for CPLEX. On the other hand, the ILS approach demonstrates polynomial scalability in running times, providing solutions within seconds for every instance. Moreover, ILS was able to find either optimal or near-optimal solutions, occasionally outperforming the upper bound produced by CPLEX in some specific instances. Thus, for handling extensive or demanding instances, the use of a metaheuristic approach like ILS is a good option.

Second, with respect to the RCPSP-RP, it is apparent that relaxing priority constraints, by increasing the relaxation threshold, leads to a substantial reduction in the overall risk. The use of strict priority constraints significantly hinders the quality of results, as more constraints result in a diminished search space and inferior optimal values. Results indicate that partially or completely disabling these constraints is an effective strategy for improving the attainable optimal values, albeit with the trade-off of longer computational times.

In Chapter 4, we extended the RCPSP-RP to integrate the dynamics of products over time. The RCPSP-RTD is a specialization of the RCPSP-RP that focuses on incorporating the dynamic nature of product transformations over time. By considering the time-dependent evolution of substances released into the environment, the RCPSP-RTD model aims to offer a more realistic approach to risk mitigation optimization.

In order to conduct a study on the model and method performances, we performed computational experiments on a benchmark of instances containing generic product data, where a MILP model and an ILS metaheuristic as the optimization methods. These methods were tested on two benchmark of instances, (i) where the risk increases over time and the (ii) where the risk naturally decreases with time. We explored the impact of various degradation schemes on overall risk levels in these two scenarios. The results reveal a significant influence of degradation schemes on risk levels. Notably, the no degradation scheme resulted in the highest overall risk, followed by the degradation of the parent product, the degradation of the resulting product and the degradation of the parent and resulting products. The visualization of the risk graphics further confirmed these trends, demonstrating that the "no degradation" scenario can lead to risk levels up to 10% higher than the "parent degradation" scheme and as much as 14% higher than the optimal "parent and resulting product degradation" scenario.

Specifically, when utilizing CPLEX, instances containing 16 sites or more often display significant gaps to the best lower bounds within the 4-hour time limit, making the determination of optimal solutions challenging. In contrast, ILS demonstrates a polynomial scalability of running times, providing solutions for all instances in the benchmark. ILS often achieves optimal or near-optimal solutions.

This thesis opens several avenues of research. First, further exploration into refined and realistic models that encompass the complete dynamics of chemical transformations is warranted. This involves developing more accurate ways to account for the transformation processes of substances and their subsequent impact on risk levels. Additionally, the interaction between scheduling decisions and these dynamic processes presents a rich area for investigation. Optimizing scheduling strategies while considering the time-dependent transformations of substances can lead to more robust and effective risk mitigation plans.

Furthermore, extending these models to consider multi-objective optimization could provide a broader perspective. Integrating objectives such as cost, time efficiency, and environmental impact into the scheduling framework would make it more versatile and applicable to a wider range of real-world scenarios. Finally, as computational techniques advance, leveraging machine learning and simulation approaches could enhance the accuracy of predicting the outcomes of various risk mitigation strategies, leading to more informed decision-making. Last but not least, the models and methods could be applied using specific products parameters.

In conclusion, this thesis addressed the complexities of scheduling in the presence of risk factors,

particularly those arising from the release of hazardous substances. The proposed models, the RCPSP-RP and the RCPSP-RTD, offer innovative approaches that pave the way for more effective solutions in industries and logistics.

Bibliography

B. Afshar-Nadjafi and M. Majlesi. Resource constrained project scheduling problem with setup times after preemptive processes. 69:16–25, 2014.

V. Akbari and F. S. Salman. Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. *European Journal of Operational Research*, 257(2):625 – 640, 2017.

V. Akbari, M. E. H. Sadati, and R. Kian. A decomposition-based heuristic for a multicrew coordinated road restoration problem. *Transportation Research Part D: Transport and Environment*, 95:102854, 2021.

D. T. Aksu and L. Ozdamar. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, 61:56 – 67, 2014.

J. Alcaraz and C. Maroto. A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102(1):83, 2001.

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

E. Anderson. Mechanisms for local search. *European Journal of Operational Research*, 88(1): 139–151, 1996.

C. Artigues. On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters*, 45(2):154–159, Mar. 2017.

C. Artigues, P. Michelon, and S. Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267, 2003.

S. Asif Raza and U. Mustafa Al-Turki. A comparative study of heuristic algorithms to solve maintenance scheduling problem. *Journal of Quality in Maintenance Engineering*, 13(4):398–410, 2007.

T. J. Barbalho. The optlis package: v0.1.0, Dec. 2022.

T. J. Barbalho, A. C. Santos, and D. J. Aloise. Metaheuristics for the work-troops scheduling problem. *International Transactions in Operational Research*, 30(2):892–914, 2023.

J.-C. Billaut, A. Moukrim, and E. Sanlaville. *Flexibility and robustness in scheduling*. John Wiley & Sons, 2013.

C. P. D. Birch, S. P. Oom, and J. A. Beecham. Rectangular and hexagonal grids used for observation, experiment and simulation in ecology. *Ecological Modelling*, 206(3):347–359, Aug. 2007.

J. Blazewicz, J. Lenstra, and A. Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

B. Bodaghi, E. Palaneeswaran, S. Shahparvari, and M. Mohammadi. Probabilistic allocation and scheduling of multiple resources for emergency operations; a Victorian bushfire case study. *Computers, Environment and Urban Systems*, 81:101479, May 2020.

K. Bouleimen and H. Lecocq. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, 149(2):268–281, 2003. Sequencing and Scheduling.

P. Brucker. Scheduling and constraint propagation. *Discrete Applied Mathematics*, 123(1):227–256, 2002.

P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, 1999.

M. Çelik, Ergun, and P. Keskinocak. The post-disaster debris clearance problem under incomplete information. *Operations Research*, 63(1):65–85, Feb. 2015.

N. Christofides, R. Alvarez-Valdes, and J. Tamarit. Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, 29(3):262–273, 1987.

A. A. Coco, C. Duhamel, and A. C. Santos. Modeling and solving the multi-period disruptions scheduling problem on urban networks. *Annals of Operations Research*, 285(1):427–443, Feb. 2020.

P. P. Das and S. Acharyya. Simulated annealing variants for solving resource constrained project scheduling problem: A comparative study. In *14th International Conference on Computer and Information Technology (ICCIT 2011)*, pages 469–474. IEEE, 2011.

E. Demeulemeester and W. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38(12):1803–1818, 1992.

E. Demeulemeester and W. Herroelen. New benchmark results for the resource-constrained project scheduling problem. *Management Science*, 43(11):1485–1492, 1997.

E. Demeulemeester and W. Herroelen. *Project Scheduling: A research handbook*, volume 49. Springer New York, 2002.

T. T. Doan, N. Bostel, and M. H. Hà. The vehicle routing problem with relaxed priority rules. *EURO Journal on Transportation and Logistics*, 10:100039, 2021.

P. M. Duque and K. Sörensen. A GRASP metaheuristic to improve accessibility after a disaster. *OR Spectrum*, 33(3):525–542, July 2011.

P. M. Duque, I. S. Dolinskaya, and K. Sörensen. Network repair crew scheduling and routing for emergency relief distribution problem. *European Journal of Operational Research*, 248(1): 272–285, Jan. 2016.

eMARS. Fire in a plant manufacturing additives for oils and lubricants. 2019. Oct., 2019. Available in: `https://emars.jrc.ec.europa.eu/en/emars/accident/view/923dc1f3-f4a6-11e9-9637-005056ad0167`. Accessed on Oct 6th, 2022.

European Commission. On the implementation and efficient functioning of directive 2012/18/eu on the control of major-accident hazards involving dangerous substances for the period 2015-2018. 2021. URL `https://ec.europa.eu/environment/pdf/industrial-accidents/seveso_implementation_report.pdf`.

Y. Fang, Y. Deng, and Z. Zhou. Formulation and solution of an emergency routing problem for forest fire with priority areas. In *2019 International Conference on Industrial Engineering and Systems Management (IESM)*, pages 1–5, 2019.

R. Z. Farahani, M. M. Lotfi, A. Baghaian, R. Ruiz, and S. Rezapour. Mass casualty management in disaster scene: A systematic review of OR&MS research in humanitarian operations. *European Journal of Operational Research*, 287(3):787–819, Dec. 2020.

C.-M. Feng and T.-C. Wang. Highway emergency rehabilitation scheduling in post-earthquake 72 hours. *Journal of the Eastern Asia Society for Transportation Studies*, 5(1), 2003.

K. Fleszar and K. S. Hindi. Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155(2):402–413, 2004.

R. Graham, E. Lawler, J. Lenstra, and A. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. Hammer, E. Johnson, and B. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287–326. Elsevier, 1979.

D. Gries, A. J. Martin, J. L. Van De Snepscheut, and J. T. Udding. An algorithm for transitive reduction of an acyclic graph. *Science of Computer Programming*, 12(2):151–155, July 1989.

L. Görlitz, Z. Gao, and W. Schmitt. Statistical Analysis of Chemical Transformation Kinetics Using Markov-Chain Monte Carlo Methods. *Environmental Science & Technology*, 45(10):4429–4437, May 2011.

Z. Hanzálek and P. Šůcha. Time symmetry of resource constrained project scheduling with general temporal constraints and take-give resources. *Annals of Operations Research*, 248(1-2):209–237, Jan. 2017.

S. Hartmann. A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics (NRL)*, 49(5):433–448, 2002.

S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, Nov. 2010.

S. Hartmann and D. Briskorn. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 297(1):1–14, 2022.

W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306, Sept. 2005.

W. Herroelen, B. De Reyck, and E. Demeulemeester. Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, 25(4):279–302, 1998.

K. S. Hindi, H. Yang, and K. Fleszar. An evolutionary algorithm for resource-constrained project scheduling. *IEEE Transactions on evolutionary computation*, 6(5):512–518, 2002.

H. Hu, J. He, X. He, W. Yang, J. Nie, and B. Ran. Emergency material scheduling optimization model and algorithms: A review. *Journal of Traffic and Transportation Engineering (English Edition)*, 6(5):441–454, Oct. 2019.

J. Hurink and J. Keuchel. Local search algorithms for a single-machine scheduling problem with positive and negative time-lags. *Discrete Applied Mathematics*, 112(1-3):179–197, Sept. 2001.

M. H. Hà, H. Nguyen Phuong, H. Tran Ngoc Nhat, A. Langevin, and M. Trépanier. Solving the clustered traveling salesman problem with d-relaxed priority rule. *International Transactions in Operational Research*, 29(2):837–853, 2022.

T. A. S. John W. Green and H. Holbech. *An introduction to toxicity experiments*, chapter 1, pages 1–17. John Wiley & Sons, Ltd, 2018.

R. L. Kadri and F. F. Boctor. An efficient genetic algorithm to solve the resource-constrained project scheduling problem with transfer times: The single mode case. *European Journal of Operational Research*, 265(2):454–462, 2018.

M. E. Keim, MD. The public health impact of industrial disasters. *American Journal of Disaster Medicine*, 6(5):265–272, Sept. 2011.

S. Kim, Y. Shin, G. M. Lee, and I. Moon. Early stage response problem for post-disaster incidents. *Engineering Optimization*, 50(7):1198–1211, July 2018.

R. Klein. *Scheduling of Resource-Constrained Projects*. Springer Science & Business Media, Nov. 1999.

Y. A. Kochetov and A. A. Stolyar. Evolutionary local search with variable neighborhood for the resource constrained project scheduling problem. In *in: Proceedings of the 3rd International Workshop of Computer Science and Information Technologies*, 2003.

R. Kolisch. *Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes*. Physica Heidelberg, 1995.

R. Kolisch. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, 14(3):179–192, Sept. 1996.

R. Kolisch and S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174(1):23–37, 2006.

R. Kolisch and R. Padman. An integrated survey of deterministic project scheduling. *Omega*, 29 (3):249–272, 2001.

R. Kolisch and A. Sprecher. PSPLIB – a project scheduling problem library. *European journal of operational research*, 96(1):205–216, 1997.

S. Kreter, A. Schutt, and P. J. Stuckey. Modeling and Solving Project Scheduling with Calendars. In G. Pesant, editor, *Principles and Practice of Constraint Programming*, pages 262–278, Cham, 2015. Springer International Publishing.

S. Kreter, J. Rieck, and J. Zimmermann. Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars. *European Journal of Operational Research*, 251(2):387–403, 2016.

P. Lacomme, A. Moukrim, A. Quilliot, and M. Vinot. Integration of routing into a resource-constrained project scheduling problem. *EURO Journal on Computational Optimization*, 7(4): 421–464, 2019.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of scheduling under precedence constraints. *Operations Research*, 26(1):22–35, 1978.

H. R. Lourenço, O. C. Martin, and T. Stützle. *Iterated Local Search*, pages 320–353. Springer US, Boston, MA, 2003.

M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, Jan. 2016.

J. Mendes, J. Gonçalves, and M. Resende. A random key based genetic algorithm for the resource constrained project scheduling problem. *Computers & Operations Research*, 36(1):92–109, 2009.

D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *IEEE transactions on evolutionary computation*, 6(4):333–346, 2002.

A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44(5):714–729, 1998.

N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & operations research*, 24 (11):1097–1100, 1997.

R. H. Möhring, M. Skutella, and F. Stork. Scheduling with and/or precedence constraints. *SIAM Journal on Computing*, 33(2):393–415, 2004.

A. M. Mora, J. J. Merelo, C. Millan, J. Torrecillas, J. L. J. Laredo, and P. A. Castillo. Enhancing a MOACO for Solving the Bi-criteria Pathfinding Problem for a Military Unit in a Realistic Battlefield. In M. Giacobini, editor, *Applications of Evolutionary Computing*, Lecture Notes in Computer Science, pages 712–721, Berlin, Heidelberg, 2007. Springer.

D. Y. Murzin, J. Wärnå, H. Haario, and T. Salmi. Parameter estimation in kinetic models of complex heterogeneous catalytic reactions using Bayesian statistics. *Reaction Kinetics, Mechanisms and Catalysis*, 133(1):1–15, June 2021.

R. E. Overstreet, D. Hall, J. B. Hanna, and R. Kelly Rainer Jr. Research in humanitarian logistics. *Journal of humanitarian logistics and supply chain management*, 1(2):114–131, 2011.

M. Palpant, C. Artigues, and P. Michelon. LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research*, 131(1): 237–257, 2004.

K. Panchamgam, Y. Xiong, B. Golden, B. Dussault, and E. Wasil. The hierarchical traveling salesman problem. *Optimization Letters*, 7(7):1517–1524, Oct. 2013.

S. S. Panwalkar and W. Iskander. A Survey of Scheduling Rules. *Operations Research*, 25(1):45–61, Feb. 1977.

G. C. Pena, A. C. Santos, and C. Prins. Solving the integrated multi-period scheduling routing problem for cleaning debris in the aftermath of disasters. *European Journal of Operational Research*, 306(1):156–172, 2023.

J. Poppenborg and S. Knust. A flow-based tabu search algorithm for the RCPSP with transfer times. *OR Spectrum*, 38(2):305–334, Mar. 2016.

A. A. B. Pritsker, L. J. Waiters, and P. M. Wolfe. Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach. *Management Science*, 16(1):93–108, Sept. 1969.

A. Quilliot and H. Toussaint. Resource constrained project scheduling with transportation delays. *IFAC Proceedings Volumes*, 45(6):1481–1486, 2012. 14th IFAC Symposium on Information Control Problems in Manufacturing.

Y. Ren and G. Tian. Emergency scheduling for forest fires subject to limited rescue team resources and priority disaster areas. *IEEJ Transactions on Electrical and Electronic Engineering*, 11(6): 753–759, Nov. 2016.

Reuters. Over 5,250 tonnes of chemicals burned in rouen, france, industrial fire. *Reuters*, 2019. Oct., 2019. Available in: `https://www.reuters.com/article/france-fire-rouen-idUSL5N26M68E`. Accessed on Sep 12th, 2022.

C. C. Ribeiro, P. Hansen, K. Nonobe, and T. Ibaraki. Formulation and tabu search algorithm for the resource constrained project scheduling problem. *Essays and surveys in metaheuristics*, pages 557–588, 2002.

C. S. Sakuraba, A. C. Santos, C. Prins, L. Bouillot, A. Durand, and B. Allenbach. Road network emergency accessibility planning after a major earthquake. *EURO Journal on Computational Optimization*, 4(3):381–402, 2016.

A. C. Santos. New trends and opportunities in post-disaster relief optimization problems. *Brazilian Journal of Operations and Production Management*, 16(3):528, Sept. 2019.

C. Schwindt and J. Zimmermann, editors. *Handbook on Project Management and Scheduling Vol. 2*. Springer International Publishing, Cham, 2015.

Seine-Maritime. Incendie Lubrizol et NL Logistique du 26 septembre 2019. 2019. Dec., 2019. Available in: `https://www.seine-maritime.gouv.fr/Actualites/Incendie-Lubrizol-et-NL-Logistique-du-26-septembre-2019/Incendie-Lubrizol-et-NL-Logistique-du-26-septembre-2019`. Accessed on Oct 6th, 2022.

V. Sels, N. Gheysen, and M. Vanhoucke. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. *International Journal of Production Research*, 50(15):4255–4270, Aug. 2012.

Seveso-III Directive. The seveso-iii directive, 2018. Available in: `https://ec.europa.eu/environment/seveso/`. Accessed on Oct 29th, 2021.

Y. Shou, Y. Li, and C. Lai. Hybrid particle swarm optimization for preemptive resource-constrained project scheduling. *Neurocomputing*, 148:122–128, 2015.

J. P. Sousa and L. A. Wolsey. A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, 54(1):353–367, Feb. 1992.

A. Sprecher. Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science*, 46(5):710–723, 2000.

L. Tavares. A review of the contribution of operational research to project management. *European Journal of Operational Research*, 136(1):1–18, 2002.

P. R. Thomas and S. Salhi. A tabu search approach for the resource constrained project scheduling problem. *Journal of heuristics*, 4:123–139, 1998.

E. B. Tirkolaee, N. S. Aydın, M. Ranjbar-Bourani, and G.-W. Weber. A robust bi-objective mathematical model for disaster rescue units allocation and scheduling with learning effect. *Computers & Industrial Engineering*, 149:106790, Nov. 2020.

Y. C. Toklu. Application of genetic algorithms to construction scheduling with or without resource constraints. *Canadian Journal of Civil Engineering*, 29(3):421–429, 2002.

P. Tormos and A. Lova. A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research*, 102(1):65–81, 2001.

V. Valls, S. Quintanilla, and F. Ballestın. Resource-constrained project scheduling: a critical activity reordering heuristic. *European journal of operational research*, 149(2):282–301, 2003.

V. Valls, S. Quintanilla, and F. Ballestın. Justification and rcpsp: A technique that pays. *European Journal of Operational Research*, 165(2):375–386, 2005.

V. Valls, F. Ballestin, and S. Quintanilla. A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European journal of operational research*, 185(2):495–508, 2008.

M. Vanhoucke and J. Coelho. Resource-constrained project scheduling with activity splitting and setup times. *Computers & Operations Research*, 109:230–249, 2019.

R. Vodák, M. Bíl, and Z. Křivánková. A modified ant colony optimization algorithm to increase the speed of the road network recovery process after disasters. *International Journal of Disaster Risk Reduction*, 31:1092 – 1106, 2018.

L. Wang, P. Wu, and F. Chu. A Multi-objective Emergency Scheduling Model for Forest Fires with Priority Areas. In *2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 610–614, Dec. 2020.

Q. Wang, M. Guidolin, D. Savic, and Z. Kapelan. Two-objective design of benchmark problems of a water distribution system via moeas: Towards the best-known approximation of the true pareto front. *Journal of Water Resources Planning and Management*, 141(3):04014060, 2015.

F. Wex, G. Schryen, S. Feuerriegel, and D. Neumann. Emergency response in natural disaster management: Allocation and scheduling of rescue units. *European Journal of Operational Research*, 235(3):697–708, June 2014.

P. Wu, J. Cheng, and C. Feng. Resource-Constrained Emergency Scheduling for Forest Fires with Priority Areas: An Efficient Integer-Programming Approach. *IEEJ Transactions on Electrical and Electronic Engineering*, 14(2):261–270, Feb. 2019. Publisher: John Wiley & Sons, Ltd.

Appendix A

Source data and plots

This appendix provides the source data and complementary plots for the results presented in Chapter 3.

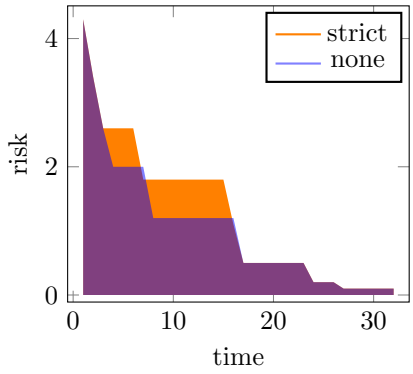
Instance		Priority policy					
		strict		moderate		none	
$\|V_d\|$	K	CPLEX time	ILS avg. time	CPLEX time	ILS avg. time	CPLEX time	ILS avg. time
8	1	0.99	0.01	0.05	0.01	0.05	0.01
	2	0.31	0.02	0.03	0.01	0.03	0.01
	4	0.02	0.02	0.02	0.01	0.02	0.01
	8	0.10	0.02	0.01	0.01	0.01	0.01
16	1	0.62	0.02	0.25	0.01	0.13	0.01
	2	0.29	0.02	0.10	0.02	0.07	0.02
	4	0.22	0.02	0.07	0.02	0.05	0.02
	8	0.04	0.02	0.03	0.01	0.03	0.01
	16	0.07	0.02	0.01	0.01	0.01	0.01
32	1	47.74	0.05	1.09	0.11	0.63	0.16
	2	16.94	0.05	0.53	0.13	0.41	0.24
	4	5.39	0.04	0.28	0.19	0.16	0.32
	8	0.31	0.05	0.14	0.23	0.08	0.40
	16	0.41	0.04	0.04	0.14	0.05	0.24
	32	0.02	0.04	0.01	0.04	0.01	0.05
64	1	14400	0.20	12.45	1.11	3.36	0.81
	2	14400	0.35	2.46	1.44	2.10	1.14
	4	234.69	0.43	1.34	2.07	0.61	1.78
	8	27.37	0.43	0.69	2.93	0.43	3.03
	16	2.21	0.32	0.20	4.34	0.14	5.38
	32	0.28	0.24	0.14	4.97	0.10	9.45
	64	0.02	0.14	0.02	0.37	0.02	0.64

Table A.1: Running times (in seconds) for CPLEX and ILS for the RCPSP-RP without sequence-dependent setup times.

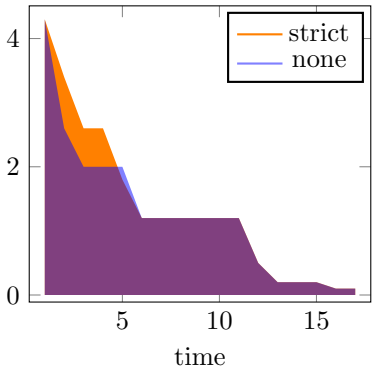
Instance		Priority policy					
		strict		moderate		none	
$\|V_d\|$	K	CPLEX time	ILS avg. time	CPLEX time	ILS avg. time	CPLEX time	ILS avg. time
8	1	0.92	0.01	5.00	0.01	5.48	0.01
	2	0.35	0.02	0.65	0.01	0.75	0.01
	4	0.35	0.01	0.34	0.01	0.29	0.01
	8	0.12	0.01	0.05	0.01	0.05	0.01
16	1	114.20	0.02	14400	0.01	14400	0.02
	2	144.93	0.02	180.00	0.02	511.28	0.03
	4	6.29	0.02	4.85	0.02	7.26	0.03
	8	0.64	0.02	0.52	0.01	0.46	0.02
	16	0.21	0.02	0.20	0.01	0.18	0.01
32	1	14400	0.06	14400	0.11	14400	0.22
	2	14400	0.07	14400	0.13	14400	0.28
	4	14400	0.08	14400	0.19	14400	0.47
	8	297.80	0.08	101.42	0.23	85.28	0.67
	16	3.86	0.05	2.03	0.14	4.43	0.64
	32	2.03	0.04	1.16	0.04	2.00	0.06
64	1	14400	0.42	14400	1.11	14400	0.81
	2	14400	0.45	14400	1.44	14400	1.16
	4	14400	0.55	14400	2.07	14400	2.09
	8	14400	0.64	14400	2.93	14400	3.81
	16	14400	0.88	14400	4.34	14400	6.36
	32	42.51	0.73	21.38	4.97	25.19	10.27
	64	28.38	0.15	14.22	0.37	14.66	0.65

Table A.2: Running times (in seconds) for CPLEX and ILS for the RCPSP-RP with sequence-dependent setup times.

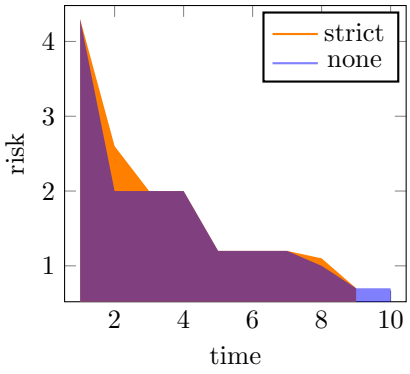
A.1 List of optimal solution plots for the scenario without sequence-dependent setup times



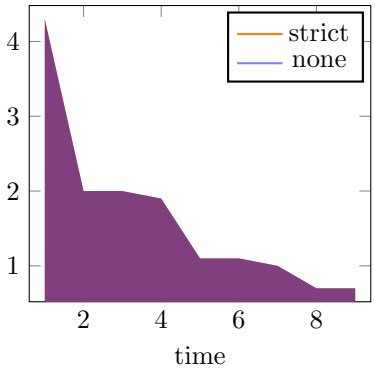
Instance (8, 1).



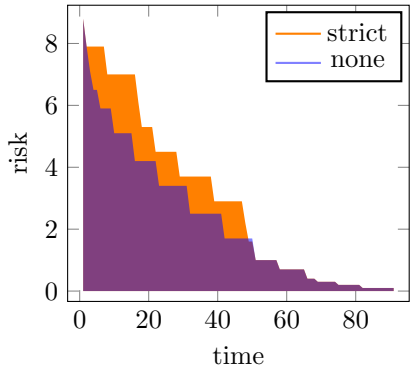
Instance (8, 2).



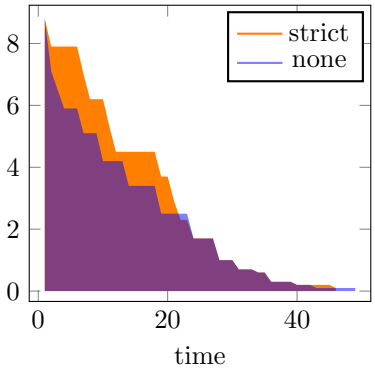
Instance (8, 4).



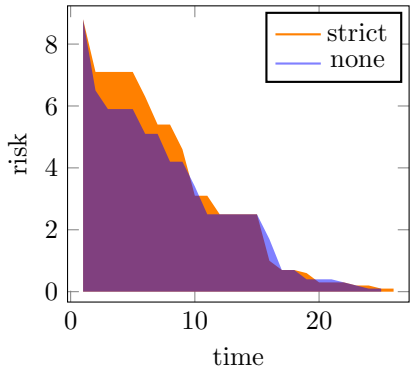
Instance (8, 8).



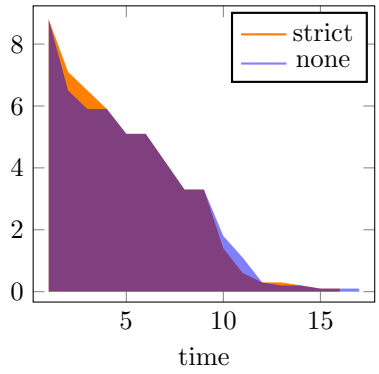
Instance (16, 1).



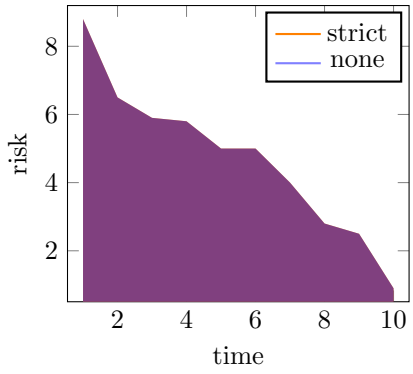
Instance (16, 2).



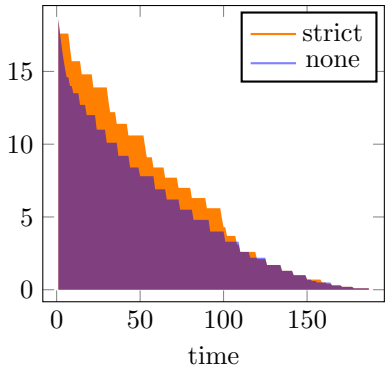
Instance (16, 4).



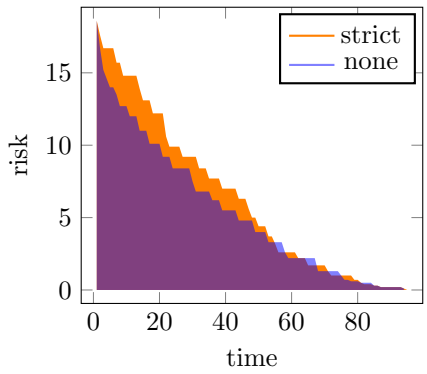
Instance (16, 8).



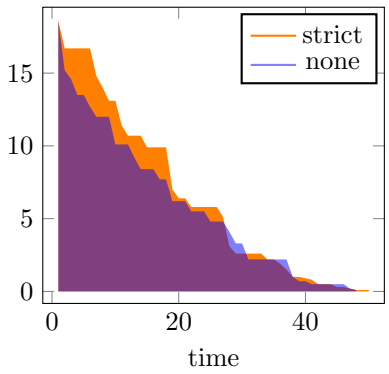
Instance (16, 16).



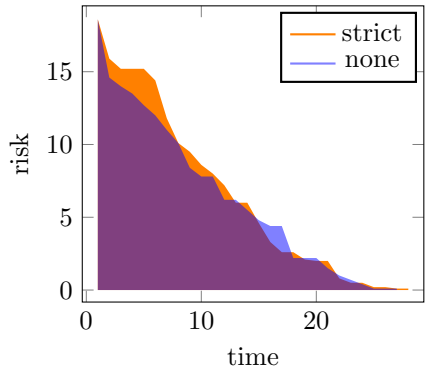
Instance (32, 1).



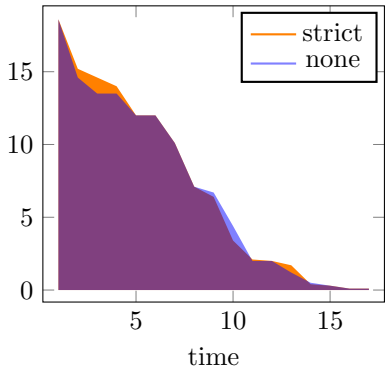
Instance (32, 2).



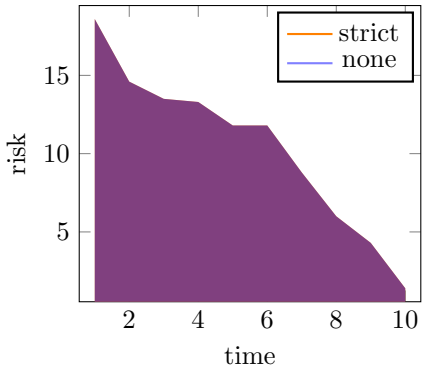
Instance (32, 4).



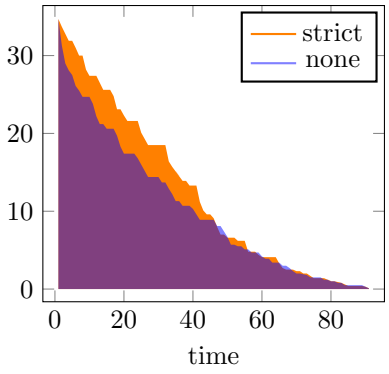
Instance (32, 8).



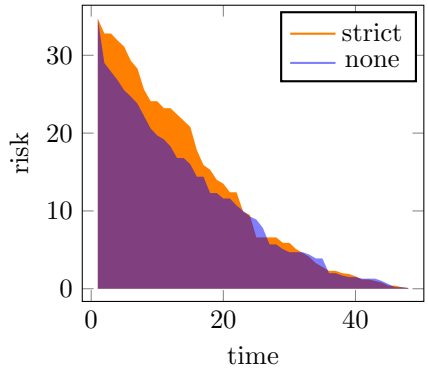
Instance (32, 16).



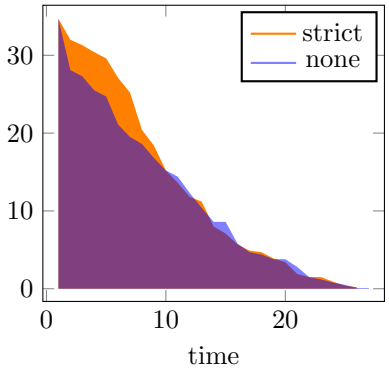
Instance (32, 32).



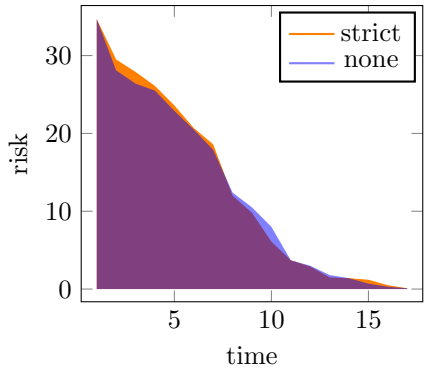
Instance (64, 4).



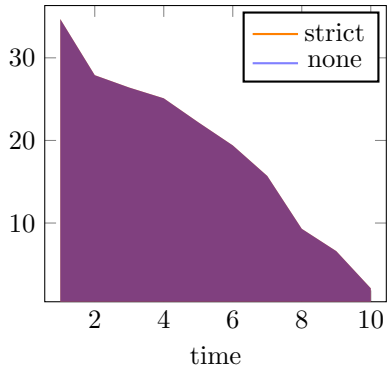
Instance (64, 8).



Instance (64, 16).

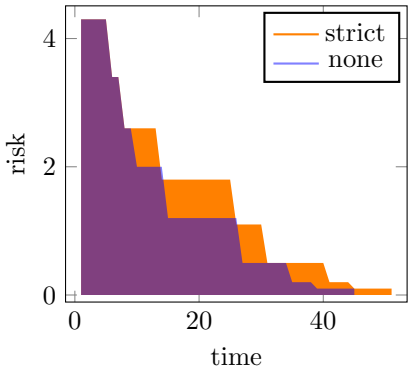


Instance (64, 32).

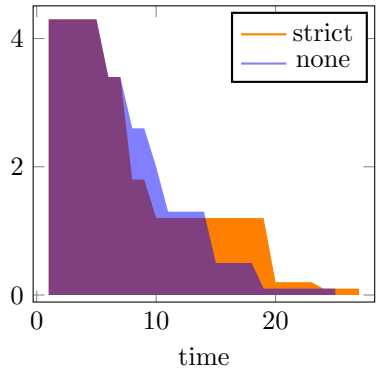


Instance (64, 64).

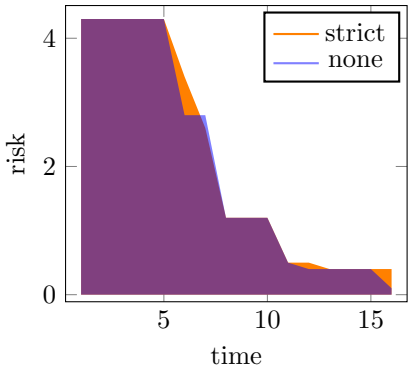
A.2 List of optimal solution plots for the scenario with sequence-dependent setup times



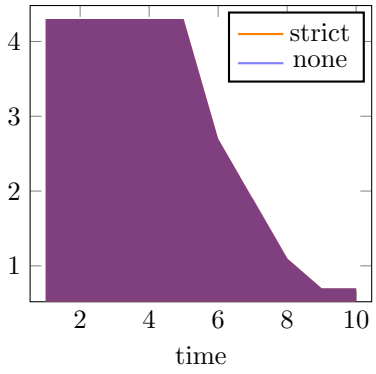
Instance (8, 1).



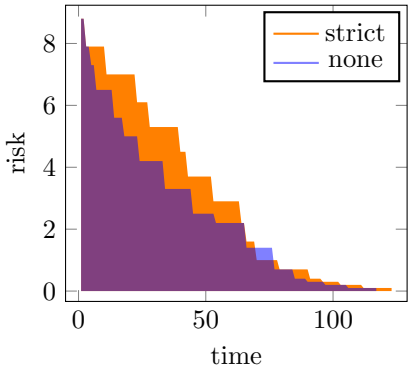
Instance (8, 2).



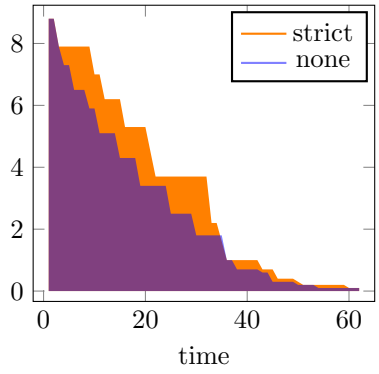
Instance (8, 4).



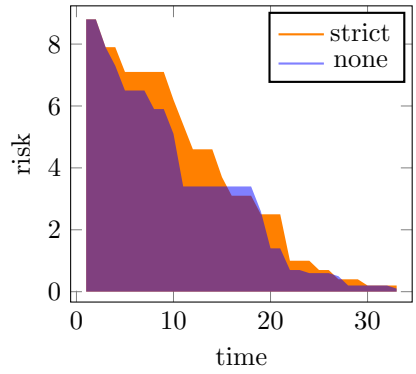
Instance (8, 8).



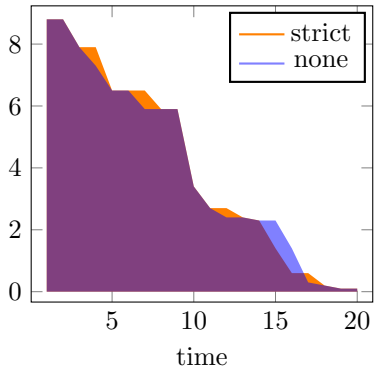
Instance (16, 1).



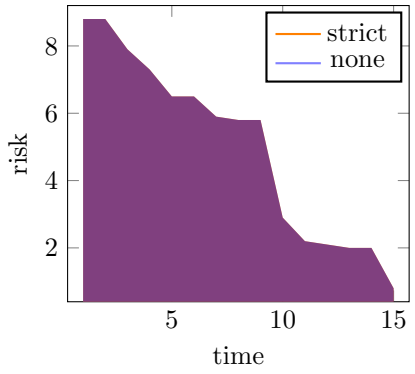
Instance (16, 2).



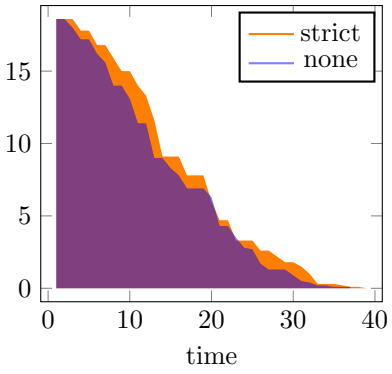
Instance (16, 4).



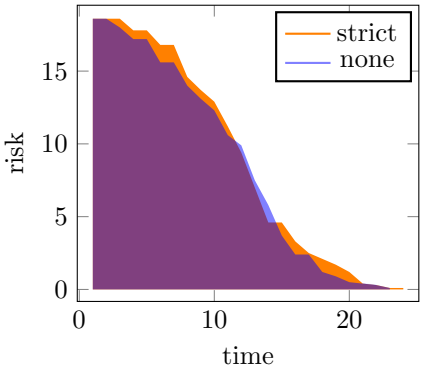
Instance (16, 8).



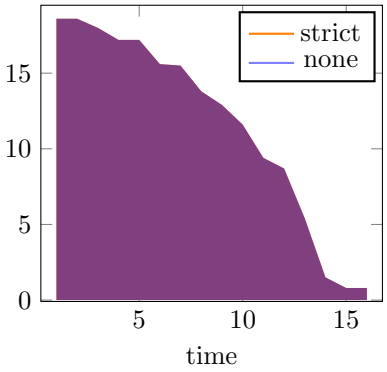
Instance (16, 16).



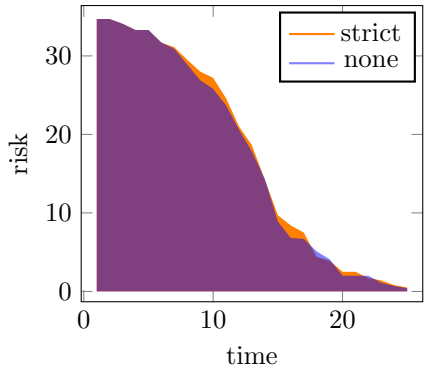
Instance (32, 8).



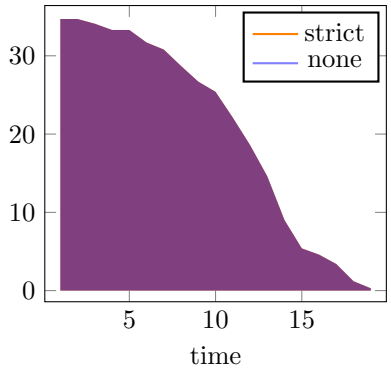
Instance (32, 16).



Instance (32, 32).



Instance (64, 32).



Instance (64, 64).