Doctoral Programme in Statistics, Optimization and Applied Mathematics

Project Scheduling: New Approaches and Solving Methods

Sofía Rodríguez Ballesteros

Supervisor

Javier Alcaraz Soria

Co-supervisor

Laura Antón Sánchez

Miguel Hernández University of Elche

-2025-

The doctoral thesis entitled *"Project Scheduling: New Approaches and Solving Methods"*, authored by **Sofía Rodríguez Ballesteros** under the supervision of Prof. **Javier Alcaraz Soria** and Prof. **Laura Antón Sánchez**, both affiliated with the Department of Statistics, Mathematics and Computer Science and the Center of Operations Research at Miguel Hernández University of Elche, is submitted in the form of a **conventional thesis**, and meets the following quality standards:

- Rodríguez-Ballesteros, S., Alcaraz, J., and Anton-Sanchez, L. (2024). Metaheuristics for the bi-objective resource-constrained project scheduling problem with time-dependent resource costs: An experimental comparison. *Computers & Operations Research*, **163**. https://doi.org/10.1016/j.cor.2023.106489.

Other publications under review

The following dissertation-related publications of the doctoral thesis entitled *"Project Scheduling: New Approaches and Solving Methods"* are being peer-reviewed:

- Rodríguez-Ballesteros, S., Alcaraz, J., and Anton-Sanchez, L. Multi-mode resource-constrained project scheduling problem with time-dependent resource costs and capacities: A bi-objective approach. *Expert Systems with Applications*.
- Rodríguez-Ballesteros, S., Goerigk, M., Alcaraz, J., and Anton-Sanchez, L. A robust optimization approach for scheduling with uncertain start-time dependent costs. *Computers & Operations Research*.

Prof. **Javier Alcaraz Soria** and Prof. **Laura Antón Sánchez**, both affiliated with the Department of Statistics, Mathematics and Computer Science and the Center of Operations Research at Miguel Hernández University of Elche, hereby certify that the doctoral thesis entitled *"Project Scheduling: New Approaches and Solving Methods"* has been conducted under their supervision by the doctoral candidate **Sofía Rodríguez Ballesteros**.

This work has been carried out in accordance with the terms and conditions established in the approved Research Plan and in compliance with the Code of Good Practice of Miguel Hernández University of Elche. The research has satisfactorily met the objectives and requirements to be defended publicly as a doctoral thesis.

In Elche, July, 2025.

The supervisor: The co-supervisor:

Prof. Javier Alcaraz Soria Prof. Laura Antón Sánchez

Prof. **Domingo Carlos Morales González**, Coordinator of the Doctoral Programme in Statistics, Optimization and Applied Mathematics at Miguel Hernández University of Elche, hereby certifies that the doctoral thesis entitled *"Project Scheduling: New Approaches and Solving Methods"* has been conducted under the supervision of the Doctoral Programme by the Doctoral Candidate **Sofía Rodríguez Ballesteros**.

This work has been carried out in accordance with the terms and conditions established in the approved Research Plan and in compliance with the Code of Good Practice of Miguel Hernández University of Elche. The research has satisfactorily met the objectives and requirements to be defended publicly as a doctoral thesis.

In Elche, July, 2025.

The coordinator:

Prof. Domingo Carlos Morales González

This doctoral thesis has been financially supported by:

- Grant PROMETEO-2021-063 from the Generalitat Valenciana, Spain.
- Contract funded by the Spanish National Statistics Institute (INE), within the project "Small area estimation of Spanish multivariable indicators" (reference 2021N1013002).

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento a mis directores, Javier Alcaraz y Laura Antón, por su guía, apoyo constante y confianza durante todos estos años. Del mismo modo, agradezco profundamente a Domingo Morales y a María Dolores Esteban por su cercanía y por brindarme la oportunidad de continuar mi trayectoria investigadora en la Universidad Miguel Hernández. No puedo dejar de mencionar a mis compañeros y compañeras del CIO, cuya amistad y apoyo diario han hecho que mi estancia en Elche sea una experiencia realmente positiva. Gracias a todos por hacerme sentir como en casa, incluso estando lejos.

También agradezco especialmente a Marc Goerigk por acogerme durante mi estancia de investigación en la Universidad de Passau. Esta experiencia me brindó la oportunidad de trabajar en uno de los estudios incluidos en esta tesis, y, sobre todo, de vivir una etapa enriquecedora que me ha hecho crecer tanto a nivel académico como personal. Su generosidad, cercanía y disposición han sido fundamentales durante todo el proceso.

Esta tesis ha sido posible gracias a la financiación recibida a través de un contrato predoctoral vinculado al proyecto PROMETEO/2021/063, concedido por la Generalitat Valenciana y gestionado por la Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital. Asimismo, quiero dar las gracias al Instituto Nacional de Estadística (INE) por la financiación otorgada mediante el contrato de investigación asociado al proyecto *"Small area estimation of Spanish multivariable indicators"* (referencia 2021N1013002), que me ha permitido mantener mi actividad investigadora durante la fase final de elaboración de esta tesis. Este trabajo también ha contado con el respaldo del proyecto PID2022-136383NB-I00, financiado por MICIU/AEI/10.13039/501100011033 y por el FEDER/Unión Europea. Finalmente agradezco al Vicerrectorado de Investigación de la Universidad Miguel Hernández de Elche por financiar mi estancia de investigación en el extranjero, la cual me ha permitido obtener la mención de Doctorado Internacional en el Título de Doctor.

Por último, pero de manera muy especial, quiero agradecer a mi familia, especialmente a mis padres y a mi hermana Diana, por su cariño incondicional durante todas las etapas de mi vida, y en particular durante la realización de esta tesis. También a mis amigos más cercanos, quienes siempre han estado ahí, escuchándome y apoyándome en los momentos difíciles. A mi querida amiga María, gracias por haber compartido esta etapa conmigo, por crecer juntas y por todo lo vivido a lo largo del camino. Y, finalmente, gracias de corazón a mi pareja, Víctor, por tu paciencia, tu comprensión y por estar siempre a mi lado a lo largo de este proceso.

Table of Contents

Abstract

This doctoral thesis delves into the project scheduling literature, a core field within operations research encompassing a variety of models and methods aimed at organizing interdependent activities over time—often subject to precedence and resource constraints—while optimizing specific objectives. As modern project environments become increasingly dynamic and uncertain, traditional models struggle to reflect key features such as multi-objective formulations, time-dependent costs, alternative execution modes, and incomplete information at the planning stage. This research addresses these limitations by proposing new mathematical formulations and tailored solution methods that improve the realism, flexibility, and effectiveness of scheduling models, ultimately contributing to more resilient and evidence-based decision-making.

Three interrelated research lines are developed, each addressing a critical aspect of project scheduling under realistic constraints. The first focuses on an extension of the classical resource-constrained project scheduling problem that incorporates time-varying resource costs, aiming to jointly minimize project duration and total cost. Several metaheuristic algorithms are adapted and rigorously evaluated through computational experiments, revealing performance trade-offs and confirming the effectiveness of evolutionary strategies in producing diverse, high-quality solutions. The second line expands the model by integrating time-varying resource capacities and multiple execution modes per activity, each reflecting different trade-offs between resource use and duration. A tailored metaheuristic with problem-specific operators is proposed, demonstrating robustness and scalability, consistently outperforming exact methods on medium- and large-scale instances while remaining competitive on smaller ones. The third line introduces a scheduling problem with starting-time-dependent costs, defined within a budgeted uncertainty set. Motivated by applications with fixed execution order and flexible starting times, a robust two-stage optimization framework is proposed. A compact mixed-integer formulation is introduced for the continuous case, while an iterative algorithm is developed for the discrete counterpart. Computational experiments confirm compact the model's efficiency, scalability, and ability to deliver high-quality solutions under uncertainty.

Overall, this thesis advances project scheduling by offering flexible modeling frameworks and efficient solution methods that bridge the gap between theory and practice. The proposed approaches support multi-objective decision-making, incorporate execution flexibility, and yield reliable schedules in uncertain environments. Together, these contributions provide valuable tools that enhance planning performance in complex, real-world scenarios.

Resumen

La planificación de proyectos es una disciplina fundamental en investigación operativa que abarca una amplia variedad de modelos y métodos para programar actividades interdependientes, sujetas habitualmente a restricciones de precedencia y recursos, buscando optimizar determinados objetivos. Conforme aumenta el dinamismo, la limitación y la incertidumbre de los problemas reales, los modelos tradicionales presentan dificultades para reflejar características clave como formulaciones multi-objetivo, costes dependientes del tiempo, modos alternativos de ejecución o información incompleta en la planificación. Esta tesis doctoral aborda estas limitaciones mediante nuevas formulaciones matemáticas y métodos de resolución que aumentan la flexibilidad y eficacia de los modelos, favoreciendo así una toma de decisiones más robusta y fundamentada.

La investigación desarrolla tres líneas interrelacionadas que extienden la planificación de proyectos hacia contextos más realistas. La primera se centra en una extensión del problema clásico de planificación de proyectos con recursos limitados, que incorpora costes de recursos dependientes del tiempo y busca minimizar conjuntamente la duración del proyecto y el coste total. Diversas metaheurísticas son adaptadas y evaluadas mediante experimentos computacionales, mostrando claramente sus ventajas e inconvenientes y confirmando su eficacia para obtener soluciones diversas y de alta calidad. La segunda línea amplía el modelo incorporando capacidades variables y múltiples modos de ejecución, cada uno con distintos equilibrios entre uso de recursos y duración. Se propone una metaheurística específica con operadores adaptados al problema, que demuestra ser robusta y escalable, superando consistentemente a métodos exactos en instancias de tamaño medio y grande, y obteniendo además resultados competitivos en problemas pequeños. La tercera línea se centra en un problema cuyos costes dependen del instante de inicio de las actividades y están sujetos a incertidumbre presupuestada. Motivado por aplicaciones donde el orden de ejecución está fijado, pero los tiempos de inicio son flexibles, se plantea un marco robusto de optimización en dos etapas, con una formulación compacta para el caso continuo y un algoritmo iterativo para el discreto. Los resultados computacionales confirman la eficiencia y escalabilidad del modelo compacto, proporcionando soluciones de alta calidad bajo incertidumbre.

En conjunto, esta tesis contribuye al campo de la planificación de proyectos mediante modelos flexibles y métodos de resolución eficientes, acortando la distancia entre teoría y práctica. Los enfoques propuestos permiten abordar problemas multi-objetivo, incorporar flexibilidad en la ejecución y generar calendarios fiables en contextos inciertos, proporcionando herramientas eficaces para afrontar escenarios reales y complejos.

Chapter 1
Introduction

Project scheduling is a central discipline within project management, responsible for organizing the execution of interdependent activities while adhering to constraints on time, cost, and resource availability, among others. Its relevance extends across a wide range of domains—including construction, software development, manufacturing, transportation, and service operations—where the timely and efficient delivery of projects is crucial for competitiveness, profitability, and overall operational success. As projects become increasingly complex and resource-constrained, scheduling emerges not only as a technical necessity but also as a strategic tool. Well-constructed schedules serve multiple purposes: they guide resource allocation, establish expectations for stakeholders, support procurement and logistics planning, and provide reference points for tracking progress. Conversely, scheduling delays or inefficiencies can propagate through the entire project lifecycle, leading to missed deadlines, budget overruns, and resource conflicts. From an academic standpoint, project scheduling is of great interest due to both its practical impact and its intrinsic difficulty. In the field of operations research, it is recognized as a rich class of combinatorial optimization problems, most of which are strongly NP-hard.

To address the challenges posed by real-world scenarios, a wide range of extensions and generalizations of the classical scheduling models have been proposed over the years. These include a variety of formulations, such as those that consider multiple execution modes, renewable and non-renewable resources, time-dependent costs, stochastic activity durations, and the simultaneous optimization of conflicting objectives. Such enhancements aim to reduce the gap between theoretical models and the complexities encountered in practice, enabling more realistic and robust decision-making in dynamic environments. At the same time, tackling these enriched models requires the development of advanced solution techniques. While classical exact algorithms remain useful for small instances, they become intractable as problem size and complexity grow. In response, a broad spectrum of heuristics, metaheuristics, and hybrid strategies—often combining elements of mathematical programming—has emerged as the predominant approach. Given the diversity of models and approaches, a structured understanding of the field is essential to properly position the contributions of this thesis. The following literature review explores key problem variants, the integration of uncertainty, and the development of multi-objective and robust optimization techniques, with particular attention to the lines of research on which this work builds.

1.1 Literature review

This section reviews the main research developments in project scheduling, with particular emphasis on the foundational models and extensions that relate to the contributions of this thesis. The starting point is the classical project scheduling problem (PSP), which involves determining the timing of a set of interdependent tasks such that the project's makespan—i.e., the total time required for completion—is minimized. As a core topic in operations research, the PSP has been widely studied, and numerous models and solution algorithms have been proposed over the years. A comprehensive overview is provided by Fahmy (2016). When all parameters are known with certainty and no resource constraints are present, the PSP can be efficiently solved using the critical path method, originally introduced by Kelley and Walker (1959). A major extension of the PSP is the resource-constrained project scheduling problem (RCPSP), which has received particular attention due to its ability to capture essential features of real-world projects. This problem involves scheduling a set of activities subject to precedence and resource constraints. As in the classical PSP, the typical objective is to minimize the makespan. An early discussion of the problem can be found in Blazewicz et al. (1983), where the RCPSP is shown to belong to the strongly NP-hard problems. A pioneering mathematical model is introduced by Pritsker et al. (1969).

While the RCPSP provides a detailed and representative framework, it does not account for all the complexities encountered in real-world project planning. As a result, many researchers have developed more general scheduling problems, often using the standard RCPSP as a foundation. Since the 1990s, numerous generalizations have been proposed to capture more realistic scenarios and support successful practical applications. From a deterministic perspective, a recent and comprehensive review is provided by Hartmann and Briskorn (2022). This work furnishes an up-to-date account on Hartmann and Briskorn (2010), offering an overview of the current state-of-the-art contributions to the RCPSP literature. Among the various extensions, activity modeling has received particular attention. For instance, preemptive scheduling allows activities to be interrupted and resumed, which is useful in domains with shifting resource availability or calendar constraints (see, e.g., Afshar-Nadjafi and Majlesi, 2014; Moukrim et al., 2015; Vanhoucke and Coelho, 2019).

Beyond activity-related generalizations, the RCPSP literature also includes several extensions related to temporal, resource, and objective dimensions. Under generalized temporal constraints, minimal and maximal time lags can be imposed between different combinations of activity start and finish times. This leads to the so-called RCPSP/max, extensively studied in several works (see, e.g., Ballestín et al., 2011; Bianco and Caramia, 2012; Schutt et al., 2013). These constraints allow for finer control over activity timing and enable the modeling of overlapping or delayed relations beyond strict finish-start dependencies. Another line of work introduces generalized resource constraints, such as non-renewable, partially renewable, and cumulative resources. For instance, cumulative resources can be both consumed and generated by activities, making the resource availability dependent on the schedule itself. Applications include energy generation or fluid processing systems, as seen in Sahli et al. (2016) and Carlier et al. (2018). Lastly, several studies have moved beyond makespan minimization to consider alternative objectives. These include minimizing total weighted tardiness (Wang

et al., 2020), total flow time (Hill et al., 2019), or promoting robustness by maximizing slack times to absorb delays (Palacio and Larrea, 2017).

These diverse extensions underline the versatility of the RCPSP framework and its continued evolution to address increasingly realistic project environments. Among these practical aspects is the existence of multiple execution modes per activity, which requires specific modeling approaches and leads to the development of the multi-mode RCPSP (MRCPSP). This prominent generalization allows each activity to be executed in one of several modes, each characterized by different durations and resource requirements. This flexibility enables the model to represent trade-offs between time and resource consumption, thereby broadening its applicability to a wide range of practical scenarios. In construction projects, for instance, assigning more labor or equipment may accelerate activity completion at the expense of greater resource usage, whereas limiting resources can prolong durations.

Over the years, the literature has introduced numerous MRCPSP variants that incorporate advanced modeling features. These include mode identity constraints, where groups of activities must be executed using the same mode (Rahimi et al., 2013); the inclusion of minimum and maximum time lags (Schnell and Hartl, 2016); multi-skilled resource assignments (Maghsoudlou et al., 2016); and rescheduling strategies to handle resource disruptions (Chakrabortty et al., 2016). Additionally, several studies have integrated financial considerations into the multi-mode setting, such as the maximization of net present value (Leyman and Vanhoucke, 2016) and activity payment-based objectives (Tirkolaee et al., 2019). The MRCPSP has also been extended to multi-project environments, where several concurrent projects compete for shared resources, increasing the complexity of coordination (Toffolo et al., 2016).

The mathematical formulation of the MRCPSP was first introduced by Talbot (1982), who also proposed an exact solution procedure based on enumeration. Since then, several other exact approaches have been developed (e.g., Patterson et al., 1989; Sprecher et al., 1997; Sprecher and Drexl, 1998; Demeulemeester et al., 2000). However, due to its combinatorial complexity, the MRCPSP and its generalizations are classified as strongly NP-hard problems, making exact methods impractical for medium and large-scale instances. Consequently, a rich body of heuristic and metaheuristic methods has been developed to solve the MRCPSP. Among these, evolutionary algorithms (EAs), such as genetic algorithms (GAs), and other bio-inspired techniques, such as particle swarm optimization (PSO), have shown promising results on large instances (e.g., Alcaraz et al., 2003; Jarboui et al., 2008; Lova et al., 2009; Shen and Li, 2013). Other approaches include ant colony optimization (ACO) (Li and Zhang, 2013), scatter search (Pourghaderi et al., 2008), and differential evolution (DE) (Damak et al., 2009). More recent studies have explored hybrid strategies—such as path-relinking techniques (Fernandes et al., 2018) and matheuristics based on local branching (Fernandes and de Souza, 2021)—which offer effective mechanisms for exploring complex solution spaces while maintaining computational efficiency.

Another major research stream in project scheduling focuses on multi-objective formulations, which reflect the reality that decision-makers seldom aim to optimize a single performance criterion. Instead, they often need to balance conflicting objectives such as minimizing the makespan, reducing project costs, ensuring robustness against delays, or leveling resource usage. The multi-objective RCPSP extends the classical single-objective problem by simultaneously considering two or more of these

goals. A number of literature reviews address this topic in depth (see, e.g., Ballestín and Blanco, 2015; Habibi et al., 2018).

In contrast to single-objective problems, multi-objective optimization gives rise to a set of non-dominated trade-off solutions, known as the Pareto front (PF). These solutions represent different compromises among the objectives and can assist decision-makers in choosing an appropriate plan according to their preferences. Common approaches include aggregating the objectives into a single weighted function (e.g., Zamani, 2013; Dai et al., 2018; Schnabel et al., 2018), using lexicographic orderings (Florez et al., 2013), or generating the full set of Pareto-optimal solutions (e.g., Ballestín et al., 2011). Several works also focus on resource leveling, where the aim is to reduce fluctuations in resource usage over time. For instance, Shahsavar et al. (2015) address an RCPSP with minimal and maximal time lags, optimizing three objectives: makespan, resource investment, and resource leveling. Other studies incorporate environmental or sustainability-related criteria, such as in Tabrizi (2018), who consider material procurement decisions under two objectives: minimizing the ecological impact of orders and minimizing the overall cost, which includes procurement, inventory, resource availability, and schedule-based penalties or rewards. Geiger (2017) consider objectives such as the minimization of total project delay and total makespan in multi-mode, multi-project settings.

To cope with the combinatorial complexity of generating high-quality approximations of the PF in large instances, researchers have developed a wide range of approximate solution techniques. Among them, multi-objective evolutionary algorithms (MOEAs) are widely applied due to their ability to explore diverse regions of the solution space and maintain a good spread of trade-off solutions (see, e.g., Fonseca and Fleming, 1993; Zitzler, 1999; Emmerich and Deutz, 2018). These include multi-objective adaptations of metaheuristics previously commented in the single-objective MRCPSP context—such as GAs, PSO, ACO, and DE—often enhanced with problem-specific operators or hybridized with mathematical programming. A comprehensive review of various approaches is provide by Durillo et al. (2010).

In a recent contribution, Alcaraz et al. (2022) introduce a bi-objective RCPSP that minimizes both makespan and total cost of resource usage, while introducing time-dependent resource costs, referred to as the bi-objective RCPSP_TDRC. This means that the costs associated with resources not only depend on the resource considered but also on the specific time period when the resource is utilized. This novel feature enables the modeling of more realistic scenarios, such as fluctuating energy prices or labor wages that increase on weekends or during holidays. Time-dependent resource costs are relevant in a wide range of practical contexts. In production environments, for instance, energy prices are often significantly lower during off-peak hours. Similarly, the cost of resources like labor, equipment, and raw materials may vary over time due to operational schedules or market conditions. In transportation logistics, resource costs—such as those associated with vehicles, fuel, or drivers—can fluctuate depending on the time of day and prevailing traffic conditions.

To tackle this problem, they propose a metaheuristic based on the non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002), and compare its performance with an exact method, the so-called augmented ϵ-constrained method (AUGMECON) (Mavrotas, 2009), demonstrating that metaheuristics are crucial for solving medium- and large-scale instances efficiently. Given the impor-

tance of this recent variant of the RCPSP—due to its bi-objective nature and time-dependent resource costs—this contribution serves as a natural starting point for part of the methodological developments presented in this thesis. Its practical modeling capabilities and algorithmic foundation provide a solid basis for the problems and solution methods explored in the following chapters.

Beyond resource-related extensions, time-dependency also arises in broader scheduling contexts. A well-known example is the PSP with start-time dependent costs, where the objective is to minimize the total cost of executing the schedule. In this setting, the cost of initiating an activity varies over time, capturing practical factors such as fluctuating prices, availability windows, or time-sensitive penalties. This variant has been extensively studied in the literature (see, e.g., Gröflin et al., 1982; Chang and Edmonds, 1985; Roundy et al., 1991; Sankaran et al., 1999). Notably, Möhring et al. (2003) show that it can be solved in strongly polynomial time by transforming it into a minimum cut problem in a directed graph. This model, although formulated without resource constraints, provides a valuable foundation for developing more complex and realistic scheduling frameworks. Its structural properties and algorithmic tractability make it a suitable starting point for the methodological developments introduced later in this thesis.

Real-world scheduling is rarely deterministic. In practice, projects are frequently affected by uncertain parameters—such as fluctuating resource availability, variable processing times, or unpredictable external disruptions—which make static planning approaches fragile. While traditional models often assume full knowledge of all relevant parameters at the planning stage, this assumption tends to break down in dynamic and volatile environments. For example, construction projects may be delayed by adverse weather, manufacturing may suffer from supply shortages, and service industries often face unanticipated shifts in demand. These sources of variability result in frequent deviations from the planned schedule, raising the need for more resilient decision-making frameworks.

As a result, recent research has focused on integrating uncertainty into scheduling models to better understand and mitigate its impact. Numerous approaches have been proposed, and several typologies are available in the literature (see, e.g., Suresh and Chaudhuri, 1993; Sabuncuoglu and Karabuk, 1999; Chaari et al., 2014). Notably, Herroelen and Leus (2005) classify scheduling under uncertainty into five main approaches, which are further elaborated in Section 2.3. Among these approaches, this thesis adopts a preventive perspective, where the aim is to anticipate and mitigate uncertainty prior to execution. Precisely, robust scheduling has emerged as a prominent line of research within this category. Rather than relying on precise probability distributions—often unavailable or unreliable in high-stakes or novel scenarios—robust optimization aims to construct baseline schedules that perform well across all scenarios within a predefined uncertainty set. This strategy is particularly valuable in contexts where delays entail high costs or critical consequences, such as infrastructure development or healthcare operations. We refer to Herroelen and Leus (2004) for a comprehensive understanding of the various procedures involved in generating robust schedules.

Robust optimization models have been widely applied to scheduling under uncertainty. Examples include risk-averse decision criteria (Kasperski and Zieliński, 2019), recoverable robustness (Bold and Goerigk, 2022b, 2024), and anchor-robustness (Bendotti et al., 2023). Building on this concept, Pass-Lanneau et al. (2024) introduce a robust optimization framework in which the goal is to construct

baseline schedules with a guaranteed subset of activities whose starting times remain fixed across all uncertainty realizations. The authors formalize the anchor-robust RCPSP and relate it to the existing adjustable-robust RCPSP, proposing exact mixed-integer programming formulations and tailored heuristics to balance robustness and flexibility under budgeted uncertainty. One influential framework is the so-called budgeted uncertainty model, or Γ-robustness, introduced by Bertsimas and Sim (2004), which limits the number of parameters that may simultaneously deviate from their nominal values, providing a controlled trade-off between conservatism and performance. This framework has inspired both single-stage and two-stage formulations. In this line, Bougeret et al. (2019) propose a min-max robust scheduling framework under budgeted uncertainty sets, focusing on the weighted completion time and makespan objectives, and develop several approximation algorithms with provable performance guarantees. Chapter 9 of the book by Goerigk and Hartisch (2024) presents a recent overview in robust approaches. For an in-depth look at robust optimization in general, we refer to the books by Ben-Tal et al. (2009); Bertsimas and den Hertog (2022).

In the context of the project scheduling, two-stage robust optimization has received increasing attention. An early contribution is that of Artigues et al. (2013), who incorporate uncertain activity durations into the RCPSP and introduce an iterative scenario-relaxation algorithm to minimize worst-case absolute regret (Kouvelis and Yu, 1997). Later, Bruni et al. (2017) propose an adjustable two-stage robust model: the first stage determines partial ordering to ensure resource feasibility, and the second stage finalizes the schedule. For the case of budgeted uncertainty, Bruni et al. (2018) compare this approach to a Benders-style decomposition algorithm (Benders, 1962). More recently, Bold and Goerigk (2021) introduce a compact reformulation of the two-stage robust RCPSP, where the first stage allocates resources and the second constructs the schedule. This framework is extended to the multi-mode case in Bold and Goerigk (2022a), who provide a compact formulation for the two-stage robust MRCPSP. Further developments include compact exact formulations and efficient heuristics for large-scale instances (Koster et al., 2024). Additional approaches also propose proactive heuristics tailored to complex project structures (Zaman et al., 2021). Overall, we refer to Hazır and Ulusoy (2020) for a recent overview of two-stage project scheduling under uncertainty.

To conclude, the literature on project scheduling has evolved from classical deterministic models toward a rich ecosystem of formulations that account for real-world complexities such as multiple objectives, alternative execution modes, cost variability, and uncertainty. The field continues to advance through the integration of robust and stochastic models, hybrid algorithmic designs, and increasingly refined representations of project constraints. Building upon these foundational studies, this thesis develops advanced methods to address scheduling environments characterized by multiple conflicting objectives and uncertain cost structures, providing decision-makers with flexible and resilient tools for real-world project planning.

1.2 Objectives and document structure

The primary objective of this doctoral thesis is to develop new mathematical formulations and effective solution approaches within the field of project scheduling, targeting practical scenarios characterized by

complex constraints and uncertainty. Specifically, this research extends classical scheduling models to include and combine features such as time-dependent costs, time-varying resource availability, multiple execution modes, and cost uncertainty. These enhancements are essential to better capture the dynamic nature of real-life environments and to support more reliable and applicable decision-making processes.

To accomplish this overarching goal, the thesis addresses three specific research objectives, each corresponding to a distinct line of inquiry. The first focuses on a well-established extension of the classical RCPSP: a bi-objective formulation that incorporates time-dependent resource costs. This document not only outlines the mathematical formulation of the problem, but also adapts and implements several state-of-the-art MOEAs to effectively solve this problem variant. A rigorous comparative study is conducted to assess their performance, offering practical insights into their suitability for this enriched scheduling context. The second research line broadens the modeling framework by proposing a multi-mode, bi-objective extension of the RCPSP. In this version, both time-dependent resource costs and time-varying resource availabilities are considered, along with the possibility of executing each activity in different modes. To solve this more complex model, the thesis develops a tailored metaheuristic based on the NSGA-II scheme, integrating problem-specific operators and mechanisms to effectively navigate the search space. The third research objective departs from the classical RCPSP family and introduces a scheduling problem, where the cost is start-time dependent and affected by uncertainty. While most existing studies focus on uncertainty in activity durations, this work shifts the attention to uncertainty in time-dependent cost structures, a comparatively less explored yet highly relevant dimension in real-world project planning. To tackle this problem, the thesis adopts a budgeted uncertainty framework and proposes a robust two-stage optimization approach capable of addressing both continuous and discrete budgeted uncertainty scenarios. A compact formulation is developed for the continuous case, while an iterative solution algorithm is introduced for the discrete setting. The results demonstrate that the proposed robust methods yield reliable solutions that maintain performance under adverse conditions.

The document is structured into six chapters and several appendices, ensuring a coherent and self-contained presentation of the research. In this introductory chapter, we have presented the motivation, provided a comprehensive literature review, outlined the main objectives, and described the overall structure of the thesis. The remainder of the document is organized as follows. Chapter 2 presents the methodological framework of the thesis, covering the fundamentals of multi-objective optimization, the design of evolutionary algorithms, a classification of the main approaches to scheduling under uncertainty, and the materials and computational resources used. Chapters 3, 4, and 5 present the core research contributions, each corresponding to one of the three main objectives described earlier. Chapter 3 addresses the bi-objective RCPSP_TDRC, presenting its formal definition and providing a comprehensive comparative analysis of several MOEAs adapted to this problem variant. Chapter 4 extends the previous model by incorporating execution modes and variable resource capacities over time, leading to the multi-mode RCPSP with time-dependent resource costs and capacities (MRCPSP_TDRCC). A new metaheuristic based on NSGA-II is proposed and assessed. Chapter 5 is dedicated to a scheduling problem under cost uncertainty, where start-time dependent costs vary across the planning horizon. Several robust optimization approaches are developed and evaluated for this

purpose. Finally, Chapter 6 summarizes the findings and suggests avenues for future research. This document also includes several appendices that complement the main chapters. Appendix A presents the statistical analyses used for the calibration and comparison of the metaheuristics introduced in Chapters 3 and 4, while Appendix B presents the full set of computational results corresponding to the experiments described in Chapter 4.

Chapter 2
Proposed methodology

This chapter provides the relevant concepts and terminology related to the methodology employed in this thesis. It begins by introducing the main concepts related to multi-objective optimization and the theory of Pareto set approximations, concluding with a detailed discussion of performance indicators commonly used in the context of multiple objectives. The next section provides an overview of metaheuristic algorithms, including a general classification of these techniques. To apprehend the contents of this work, the main focus will be on evolutionary algorithms, which belong to the class of population based bio-inspired metaheuristics. Their general framework is introduced, followed by a concise description of their key features and procedures. The chapter then delves into the theory of scheduling under uncertainty, offering an overview of the various approaches discussed in the literature. Finally, the materials and resources used throughout this research are described. This includes details on the computational resources employed for the experiments, as well as a brief overview of the benchmark instances that are widely used in the project scheduling literature and have been adopted in this research.

2.1 Multi-objective optimization

Multi-objective optimization refers to an area of multi-criteria decision-making which deals with optimization problems having more than one objective function to be optimized at a time. It can be applied to many domains of science, where it is necessary to take optimal decisions with trade-offs between two or more conflicting objectives. Given the decision space $\mathbb{R}^n$, a continuous multi-objective optimization problem is defined as the process of finding a decision vector $\mathbf{x}^* \in \mathbb{R}^n$ which satisfies the set of constraints of the problem, as well as minimizing (or maximizing) the vector of objective functions. Assuming minimization, the problem can be defined as:

$$\underset{\mathbf{x}\in\Omega}{\text{minimize}} \quad \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T,$$

where $\Omega \subseteq \mathbb{R}^n \neq \emptyset$ represents the set of values satisfying the constraints of the problem, commonly referred to as the feasible set. The functions $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ are the m objective function , $i = 1, 2, \ldots, m$ and $m \geq 2$. The set $\mathbb{R}^m$ is known as the objective space, and the image of Ω under the

objective functions, denoted as, $\mathbf{f}(\Omega) \subseteq \mathbb{R}^m$, is called the (feasible) objective set. A point $\mathbf{x} \in \Omega$ is said to be a feasible solution.

In the following, we introduce key definitions in the context of multi-objective optimization. Notably, Emmerich and Deutz (2018) describe the Pareto dominance order as a special case of the cone order, offering a more geometric perspective on this relationship. Building on this foundation, we adopt the cone order framework and proceed to define the dominance relation between two objective vectors (Dächert et al., 2017). For the purpose of this research, we have ommited those definitions that are not used in the document. For an in depth comprehension on the dominance relation and other important definitions associated with it, we refer to Audet et al. (2021).

Definition 1. Given two objective vectors $\mathbf{y}^1$ and $\mathbf{y}^2$ in the objective space $\mathbb{R}^m$, we say:

- $\mathbf{y}^1$ weakly dominates $\mathbf{y}^2$ ($\mathbf{y}^1 \leq \mathbf{y}^2$) if and only if $y_i^1 \leq y_i^2$ for all $i \in \{1, 2, \ldots, m\}$.
- $\mathbf{y}^1$ dominates $\mathbf{y}^2$ ($\mathbf{y}^1 < \mathbf{y}^2$) if and only if $\mathbf{y}^1 \leq \mathbf{y}^2$ and $\mathbf{y}^1 \neq \mathbf{y}^2$ (strict inequality in at least one component).
- $\mathbf{y}^1$ and $\mathbf{y}^2$ are incomparable if neither $\mathbf{y}^1 \leq \mathbf{y}^2$ nor $\mathbf{y}^2 \leq \mathbf{y}^1$.

We continue with the concept of dominance relation between two decision vectors (Audet et al., 2008).

Definition 2. Given two decision vectors $\mathbf{x}^1$ and $\mathbf{x}^2$ in the feasible set $\Omega \subseteq \mathbb{R}^n$, we have that:

- $\mathbf{x}^1$ weakly dominates $\mathbf{x}^2$ ($\mathbf{x}^1 \preceq \mathbf{x}^2$) if and only if $\mathbf{f}(\mathbf{x}^1) \leq \mathbf{f}(\mathbf{x}^2)$.
- $\mathbf{x}^1$ dominates $\mathbf{x}^2$ ($\mathbf{x}^1 \prec \mathbf{x}^2$) if and only if $\mathbf{f}(\mathbf{x}^1) < \mathbf{f}(\mathbf{x}^2)$.
- $\mathbf{x}^1$ and $\mathbf{x}^2$ are incomparable if neither $\mathbf{x}^1 \preceq \mathbf{x}^2$ nor $\mathbf{x}^2 \preceq \mathbf{x}^1$.

Once we have introduced these relations, we can define what is meant by a Pareto-optimal solution in the context of multi-objective optimization (Ehrgott, 2005).

Definition 3. A decision vector $\mathbf{x}^* \in \Omega$ is said to be a Pareto-optimal solution if no other vector in Ω dominates it, i.e., no objective function can be improved without simultaneously deteriorating at least one of the others. The set of all non-dominated points is referred to as the Pareto-optimal set. The image of the Pareto-optimal set under the vector-valued function $\mathbf{f}$ is known as the Pareto front (PF).

In single-objective optimization problems, the set of Pareto-optimal solutions typically consists of a singleton. In contrast, considering multiple objectives often leads to a set of incomparable solutions, which constitutes the Pareto-optimal set. The goal is to identify this set of non-dominated solutions or, due to computational complexity, to obtain its best discrete representation. Without loss of generality, we denote by Y^P the PF or any finite representation of it. Note that if the objectives are not conflicting, this set may consist of a single point. However, in this work, we assume that the objectives are conflicting, resulting in a diverse set of solutions. In the following, we introduce the formal definition of a Pareto set approximation (Zitzler et al., 2008).

Definition 4. Let $X \subseteq \Omega$ be a set of decision vectors. X is said to be a Pareto set approximation if any element of X does not weakly dominate any other. Hence, X is a set of incomparable solutions. The image of such a set in the objective space is called a PF approximation. The set of all PF approximations is denoted by $\mathcal{Y}$.

Recalling the definition of incomparable decision vectors, we conclude that any PF approximation corresponding to a given Pareto set approximation contains only distinct elements. We justify this conclusion using the contrapositive. Suppose $\mathbf{x}^1, \mathbf{x}^2 \in X$ are two decision vectors. By definition, neither $\mathbf{x}^1 \preceq \mathbf{x}^2$ nor $\mathbf{x}^2 \preceq \mathbf{x}^1$. Let $\mathbf{y}^1 = \mathbf{f}(\mathbf{x}^1)$ and $\mathbf{y}^2 = \mathbf{f}(\mathbf{x}^2)$. Assume, for the sake of contradiction, that $\mathbf{y}^1 = \mathbf{y}^2$. In this case, we would have $\mathbf{y}^1 \preceq \mathbf{y}^2$, which contradicts the fact that $\mathbf{x}^1$ and $\mathbf{x}^2$ are incomparable. Hence, all elements in the PF approximation are distinct.

As previously discussed, obtaining the Pareto-optimal set for a given problem is often infeasible, and only approximation sets are produced. However, the definition of a Pareto set approximation does not inherently consider any notion of quality. In this context, it becomes necessary to evaluate when one PF approximation exhibits superior characteristics over another. Assessing the quality of these sets of non-dominated solutions is crucial, particularly for comparing the performance of different algorithms in solving a multi-objective optimization problem. In the following section we introduce the notion of performance indicator and discuss several metrics employed in this work.

2.1.1 Performance indicators

Comparing two or more PF approximations involves evaluating specific properties of interest and determining which sets exhibit superior characteristics according to defined criteria. The most straightforward approach to achieve this is to utilize the previously defined Pareto dominance order. In their work, Zitzler et al. (2003) extend this dominance relation for objective vectors to PF approximations.

Definition 5. Let $Y^1, Y^2 \in \mathcal{Y}$ be two PF approximations. We have the following relations:

- $Y^1 \prec Y^2$ (dominates): every $\mathbf{y}^2 \in Y^2$ is dominated by at least one $\mathbf{y}^1 \in Y^1$.
- $Y^1 \preceq Y^2$ (weakly dominates): every $\mathbf{y}^2 \in Y^2$ is weakly dominated by at least one $\mathbf{y}^1 \in Y^1$.
- $Y^1 \lhd Y^2$ (is better): every $\mathbf{y}^2 \in Y^2$ is weakly dominated by at least one $\mathbf{y}^1 \in Y^1$ and $Y^1 \neq Y^2$.
- Y^1 and Y^2 are incomparable if neither $Y^1 \preceq Y^2$ nor $Y^2 \preceq Y^1$.

While these relations enable a qualitative comparison of PF approximations, they lack precision in quantifying the extent to which one approximation set is superior to another. Moreover, they provide limited insight when approximation sets are incomparable. To address these limitations, it is necessary to rely on specific quality criteria that capture the key characteristics of a high-quality PF approximation. In particular, four properties are commonly considered: cardinality, quantifying the number of non-dominated solutions; convergence, measuring how close a PF approximation is to the exact PF (if known) or to a reference front; distribution, assessing the extent to which solutions evenly cover the objective space; and spread, focusing on the uniformity of the solution distribution—ensuring

that points are well-dispersed rather than clustered. Since these properties cannot be fully evaluated through Pareto dominance alone, dedicated quantitative performance indicators are required.

To this end, Zitzler et al. (2003) introduce the formal concept of performance indicators.

Definition 6. Let $Y^1, Y^2, \ldots, Y^k \subseteq \mathcal{Y}$ be a collection of k PF approximations. A k-ary performance indicator is a function $I : \mathcal{Y}^k \longrightarrow \mathbb{R}$ which assigns to the collection $Y^1, Y^2, \ldots, Y^k$ a real value $I(Y^1, Y^2, \ldots, Y^k)$.

Note that a performance indicator can take an arbitrary number of PF approximations as input. Typically, these functions operate on either one or two approximation sets, referred to as unary and binary performance indicators, respectively.

To conclude, we discuss the concept of a comparison method, which establishes a relationship between the values provided by one or more performance indicators and the notion of one PF approximation being superior to another. Specifically, Zitzler et al. (2003) define comparison methods for both unary and binary performance indicators.

Definition 7. Let $Y^1, Y^2 \in \mathcal{Y}$ be two PF approximations, and let $\mathbf{I} = (I_1, I_2, \ldots, I_k)$ represent a combination of performance indicators. We define a Boolean function $E : \mathbb{R}^k \times \mathbb{R}^k \to \{true, false\}$, which evaluates whether a given condition between two k-dimensional vectors holds. Assume that all I_i, $i = 1, 2, \ldots, k$, are unary indicators. In this case, the comparison method $C_{\mathbf{I},E}(Y^1, Y^2)$ is defined as:

$$C_{\mathbf{I},E}(Y^1, Y^2) = E(\mathbf{I}(Y^1), \mathbf{I}(Y^2)),$$

where $\mathbf{I}(Y) = (I_1(Y), I_2(Y), \ldots, I_k(Y)) \in \mathbb{R}^k$ for all $Y \in \mathcal{Y}$. Alternatively, if all I_i, $i = 1, 2, \ldots, k$, are binary indicators, the comparison method $C_{\mathbf{I},E}(Y^1, Y^2)$ is defined as:

$$C_{\mathbf{I},E}(Y^1, Y^2) = E(\mathbf{I}(Y^1, Y^2), \mathbf{I}(Y^1, Y^2)),$$

where $\mathbf{I}(Y^1, Y^2) = (I_1(Y^1, Y^2), I_2(Y^1, Y^2), \ldots, I_k(Y^1, Y^2)) \in \mathbb{R}^k$ for all $Y^1, Y^2 \in \mathcal{Y}$.

Note that a comparison method is defined as a Boolean function returning one of two possible values: *true* or *false*. The primary purpose of this function is to determine whether one PF approximation is superior to another, based on a single or multiple performance indicators. To illustrate, consider two PF approximations $Y^1, Y^2 \in \mathcal{Y}$ generated by Algorithms 1 and 2, respectively. Suppose we define a comparison method $C_{I,E}(Y^1, Y^2)$ as $(I(Y^1) > I(Y^2))$, where I is a single unary indicator for which larger values indicate better performance. Thus, if the comparison method returns *true*, Algorithm 1 is considered to outperform Algorithm 2 according to the indicator I.

As previously mentioned, the Pareto dominance order provides a fundamental criterion for comparing PF approximations in multi-objective optimization. Consequently, a robust comparison method should ideally reflect this dominance relationship in the objective space. Specifically, Zitzler et al. (2003) emphasize that a good comparison method should be compliant with the $\lhd$-relation—meaning that it must not assess Algorithm 1 as inferior to Algorithm 2 whenever approximation Y^1 is better than Y^2. Nevertheless, this desirable property is not satisfied by all performance indicators commonly used in practice. A comprehensive analysis of the theoretical properties of commonly used indicators

has been systematically conducted in the literature (see, for instance, Tables 3, 4, and 5 in the appendix of Audet et al. (2021)).

In the remainder of this section, we present the performance indicators used in this research. These are selected for their ability to assess one or more of the four properties described above—cardinality, convergence, distribution, and spread—commonly used to evaluate the quality of PF approximations (see, e.g., Fonseca and Fleming, 1996; Zitzler, 1999; Deb et al., 2002). We focus on seven indicators that are widely recognized in the literature for their effectiveness and interpretability. For a comprehensive overview of existing metrics, we refer to Audet et al. (2021).

- *Overall non-dominated vector generation (OVNG)* (Van Veldhuizen, 1999). This cardinality indicator measures the number of unique, non-dominated points in the PF approximation. Let $Y \in \mathcal{Y}$ be a PF approximation. The OVNG metric is then defined as:

$$OVNG(Y) := |Y|\,.$$

- *C-metric or coverage of two sets (C)* (Zitzler, 1999). Let $Y^1, Y^2 \in \mathcal{Y}$ be two PF approximations. Belonging to the cardinality indicators, the function C maps the ordered pair (Y^1, Y^2) to the interval $[0, 1]$, and returns the proportion of points in front Y^2 dominated by solutions in front Y^1. Formally, the C-metric is defined as:

$$C(Y^1, Y^2) := \frac{|\{\mathbf{y}^2 \in Y^2 \mid \exists \mathbf{y}^1 \in Y^1 : \mathbf{y}^1 < \mathbf{y}^2\}|}{|Y^2|},$$

  A value of $C(Y^1, Y^2) = 1$ implies that all solutions in Y^2 are dominated by Y^1. The opposite, $C(Y^1, Y^2) = 0$ means that none of the solutions in Y^2 are dominated by Y^1. Given the exact PF or a discrete representation of it, Y^P, it follows that $C(Y, Y^P) = 0$, since no solution in an approximation set Y can dominate a Pareto-optimal solution. Typically, the C-metric values are expressed as percentages.

  It is important to evaluate both directions as the metric is not symmetric, i.e., $C(Y^1, Y^2) \neq 1 - C(Y^2, Y^1)$. Finally, note that in the original definition by Zitzler (1999), the metric uses a weak dominance relationship, meaning that solutions equal to those in Y^2 are also included in the numerator of C.

- *μ-indicator (μ)* (Alcaraz, 2024). This distribution and spread indicator is defined as the ratio between two established performance metrics: the Γ indicator (Custódio et al., 2011) and Zitzler's $\mathcal{M}_3^*$ metric (Zitzler et al., 2000). The Γ indicator measures the maximum distance, using the infinity norm, between consecutive points in a given PF approximation. A lower value for this indicator is preferred as it reflects a more uniform distribution of points. Similarly, in the case of two objectives, the $\mathcal{M}_3^*$ metric corresponds to the distance between the two extreme solutions of the front, with a higher value indicating better spread. Let $Y \in \mathcal{Y}$ be a PF approximation. Formally, the μ-indicator is defined by:

$$\mu(Y) := \frac{\Gamma(Y)}{\mathcal{M}_3^*(Y)}.$$

By combining these two metrics, μ offers a more robust assessment of how well the solutions are distributed along the front. Lower values of the μ-indicator are desirable, as they indicate a better overall distribution of solutions.

- *Spread* (Δ) (Deb et al., 2002; Audet et al., 2021). The spread metric evaluates the dispersion of points along the PF, favoring solutions that are evenly distributed and well-spread. For bi-objective problems, this indicator is formally defined as:

$$\Delta(Y, Y^P) = \frac{\sum_{i=1}^{2} \min_{\mathbf{y} \in Y} \left\|\mathbf{y}^{i,*} - \mathbf{y}\right\| + \sum_{j=1}^{|Y|-1} \left|d(\mathbf{y}^j, Y \setminus \{\mathbf{y}^j\}) - \bar{d}\right|}{\sum_{i=1}^{2} \min_{\mathbf{y} \in Y} \|\mathbf{y}^{i,*} - \mathbf{y}\| + (|Y| - 1)\bar{d}},$$

where $d(\mathbf{y}^j, Y \setminus \{\mathbf{y}^j\})$ denotes the Euclidean distance between consecutive solutions in Y, $\bar{d}$ is the average of these distances, and $\min_{\mathbf{y} \in Y} \left\|\mathbf{y}^{i,*} - \mathbf{y}\right\|$ for $i = 1, 2$ are the Euclidean distances between the extreme solutions of the exact PF (or its discrete representation) and the boundary solutions of the approximation Y. Precisely, the extreme solutions are defined as $\mathbf{y}^{i,*} = \mathbf{f}(\mathbf{x}^{i,*})$, where $\mathbf{x}^{i,*}$ is solution to the i-th single-objective problem, $i = 1, 2$.

Lower values of Δ indicate better-quality PF approximations. A spread value of zero corresponds to a perfectly distributed and widely spread set of non-dominated solutions.

- *The additive ϵ-indicator (I_{ϵ^+})* (Zitzler et al., 2003). We continue with a convergence indicator. Let $Y^1, Y^2 \in \mathcal{Y}$ be two PF approximations. The function I_{ϵ^+} maps the pair (Y^1, Y^2) to the interval $[0, +\infty)$, and returns the minimum additive factor required to shift all solutions in Y^1 to weakly dominate Y^2. Formally, the additive ϵ-indicator is defined as:

$$I_{\epsilon^+}(Y^1, Y^2) := \min_{\epsilon} \left\{\forall \mathbf{y}^2 \in Y^2,\ \exists \mathbf{y}^1 \in Y^1 : y_i^1 - \epsilon \leq y_i^2, \text{for all } i \in \{1, 2, \ldots, m\}\right\}.$$

Lower values of I_{ϵ^+} indicate a closer alignment between the two fronts, with a value of 0 meaning that Y^1 already weakly dominates Y^2 without any shift. Thus, the ϵ-indicator is to be minimized. It follows that for all $Y \in \mathcal{Y}$, $I_{\epsilon^+}(Y^P, Y) = 0$.

- *Hypervolume (HV)* (Zitzler, 1999). Also referred to as the S-metric, the hypervolume is a convergence and distribution indicator which measures the volume of the objective space dominated by a PF approximation and bounded from below by $\mathbf{0} \in \mathbb{R}^m$. Let $\{\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^s}\} \subseteq \Omega$ be a set of s decision vectors, and let $Y = \{\mathbf{y^1}, \mathbf{y^2}, \ldots, \mathbf{y^s}\} \in \mathcal{Y}$ be the corresponding PF approximation, where $\mathbf{y^i} = (f_1(\mathbf{x^i}), f_2(\mathbf{x^i}))$, for all $i = 1, \ldots, s$. Formally, the hypervolume indicator is defined as:

$$HV(Y) := \lambda_m\left(\bigcup_{\mathbf{y}^i \in Y} [\mathbf{y}, \mathbf{0}]\right),$$

where λ_m is the m-dimensional Lebesgue measure. For $m = 2$, the problem is bi-objective, and the hypervolume can be computed as follows:

$$HV(Y) = \text{vol}\left(\bigcup_{i=1}^{s} R_i\right) = \int_{\mathbb{R}^2} \chi_{\bigcup_{i=1}^{s} R_i}(\mathbf{y}) d\mathbf{y},$$

where R_i represents the rectangle defined by the points:

$$R_i = \{(a, b) \in \mathbb{R}^2 \mid 0 \leq a \leq f_1(\mathbf{x^i}), 0 \leq b \leq f_2(\mathbf{x^i})\},$$

and $\chi_{\bigcup_{i=1}^{s} R_i}$ is the characteristic function of the union of the rectangles:

$$\chi_{\bigcup_{i=1}^{s} R_i}(\mathbf{y}) = \begin{cases} 1, & \text{if } \mathbf{y} \in \bigcup_{i=1}^{s} R_i, \\ 0, & \text{otherwise.} \end{cases}$$

Originally designed for maximization problems, the hypervolume metric is adapted for our minimization problem by inverting the objective vectors. A higher hypervolume value indicates that a greater portion of the objective space is covered by the PF approximation, signifying a more accurate and comprehensive approximation of the PF.

- *Modified Inverted Generational Distance* (IGD^+) (Ishibuchi et al., 2015). Overcoming the limitations of its predecessors, the GD and IGD measures (Audet et al., 2021), this convergence and distribution performance indicator computes the distance between two fronts while integrating the dominance relation. Let $Y \in \mathcal{Y}$ be a PF approximation and let Y^P denote the exact PF or a discrete representation of it. The IGD^+ metric is then defined by:

$$IGD^+(Y, Y^P) := \frac{1}{|Y^P|} \sum_{\mathbf{y}^2 \in Y^P} \min_{\mathbf{y}^1 \in Y^1} \left\| (\mathbf{y}^1 - \mathbf{y}^2)_+ \right\|,$$

where $(\mathbf{y}^1 - \mathbf{y}^2)_+ = (\max\{0, \mathbf{y}^1 - \mathbf{y}^2\})_{i=1,2,\ldots,m}$, and $\|.\|$ refers to the 2-norm.

Note that IGD^+ takes into account the dominance relation when computing the Euclidean distance within elements of both fronts. Precisely, it is weakly Pareto compliant. That is, for all $Y^1, Y^2 \in \mathcal{Y}$, it follows that:

$$Y^1 \preceq Y^2 \Rightarrow IGD^+(Y^1, Y^P) \leq IGD^+(Y^2, Y^P).$$

For this indicator, a lower value is desirable. Moreover, a reference set can be used instead of Y^P.

2.2 Evolutionary algorithms

Many real-world optimization problems are computationally intractable due to complex search spaces and expensive evaluations, making exact methods unsuitable for large-scale instances. In this context, metaheuristic algorithms emerge as a powerful alternative. Unlike exact techniques, metaheuristics efficiently explore the solution space to obtain high-quality approximate solutions within reasonable time limits. Rather than generating and evaluating all candidate solutions, they rely on iterative improvement strategies that balance refining current solutions with exploring new possibilities. This approach enables both intensification, by focusing the search on promising regions, and diversification, by exploring broader areas, therby reducing the risk of premature convergence to local optima. Their

flexibility and scalability have led to widespread adoption across various fields, including engineering, logistics, machine learning, and scheduling, establishing them as a fundamental class of modern optimization techniques. It is important to emphasize that metaheuristics provide approximate solutions, meaning there is no guarantee of finding global optima, or even bounded solutions.

The concept of "metaheuristic" was first introduced by Glover (1986). However, several metaheuristic algorithms were formally proposed and developed in the 1960s or even earlier. Drawing inspiration from Darwin's theory of evolution, computer scientists design a new class of algorithms known as evolutionary algorithms (EAs). EAs are part of the broader category of general stochastic, population-based metaheuristics, which operate through iterative improvements on a population of solutions. Various major branches of EAs were developed independently: genetic algorithms (GAs) (Holland, 1962; Holland, 1975), developed in Michigan, USA; evolution strategies (ES) (Rechenberg, 1965), developed in Berlin, Germany; evolutionary programming (EP) (Fogel et al., 1966), developed in San Diego, USA; and, towards the end of the 1980s, genetic programming (GP) (Koza, 1992). Despite differences in their implementation, all of these EAs share common principles derived from natural evolution. In addition to EAs, metaheuristics also include ant colony optimization (Colorni et al., 1992), tabu search (Glover, 1989; Glover, 1990), simulated annealing (Černý, 1985), differential evolution (Storn and Price, 1997), and iterated local search (Martin et al., 1991), among others. For a comprehensive discussion on metaheuristic algorithms, including their classifications and properties, we refer the reader to Blum and Roli (2003).

Although numerous metaheuristics have been proposed and extensively studied in the literature, this work focuses on EAs. As mentioned earlier, EAs belong to the class of population-based metaheuristics, which are iterative processes designed to improve a population of solutions. The procedure begins with the initialization of the population, followed by the generation of a new set of solutions. This new population is then integrated into the current one using selection procedures. The search continues until a predefined stopping criterion is satisfied. Other techniques following this approach include scatter search (Glover, 1977), estimation of distribution algorithms (Lozano et al., 2006), particle swarm optimization (Kennedy and Eberhart, 1995;Poli et al., 2007), and bee colony optimization (Seeley, 1995), among others. In particular, EAs are the most extensively studied population-based algorithms and are widely applied to solving both combinatorial and non-convex numerical optimization problems.

2.2.1 General framework for EAs

EAs are optimization techniques based on the principles of natural evolution. They involve mechanisms that simulate the competition for survival within a population, where individuals with better characteristics are more likely to survive and pass their traits to the next generation. This evolutionary process results in the refinement of these traits, leading to a population that is better adapted to the challenges of their environment. In essence, EAs leverage key evolutionary procedures, such as selection, recombination, and mutation, to iteratively guide a population of candidate solutions—decision vectors—toward optimal or near-optimal solutions (Bäck, 1996).

Algorithm 1 EA template

```
1: i ← 0;
2: P_i ← create_initial_population(N);
3: P_i ← evaluate_population(P_i);
4: while not stopping_criterion do
5:    R_i ← selection(P_i);
6:    R_i ← crossover(R_i);
7:    R_i ← mutation(R_i);
8:    R_i ← evaluate_population(R_i);
9:    P_{i+1} ← replacement(P_i, R_i);
10:   i ← i + 1;
11: end while
```

Algorithm 1 illustrates the general template of EAs, which is common to most algorithms of this type. Initially, the population is typically generated randomly, though heuristic-based initialization can enhance performance. Each individual in the population represents a candidate solution to the optimization problem, and it is evaluated by assigning a fitness value based on an objective function. The main loop represents the evolutionary process, which iterates until a stopping criterion is met. Common criteria include a maximum number of function evaluations or a CPU time limit. At each iteration, the current population undergoes three genetic operators—selection, crossover, and mutation—to generate an offspring population. Note that individuals with higher fitness values are more likely to be selected as parents, mirroring the process of natural evolution. After the offspring are evaluated, individuals from both the current population and the offspring participate in the replacement procedure, where the best solutions are selected to form the population for the next generation.

Following Talbi (2009), we discuss the main design features that define EAs, some of which are common to all metaheuristics.

- *Solution encoding*: This component is fundamental to all metaheuristics, as it defines how candidate solutions are represented within the algorithm. A well-designed encoding scheme is crucial for incorporating problem-specific knowledge and enhancing the efficiency of the search process. In EAs, the genotype represents the encoded solution, while the phenotype corresponds to the actual solution.

- *Fitness function*: The fitness function evaluates the quality of each solution in the search space by assigning it a numerical value. It plays a crucial role in guiding the search toward high-quality solutions. If poorly designed, it may mislead the search process, leading to suboptimal results regardless of the metaheuristic used. In EAs, the fitness function is often derived from a transformation of the objective function, where higher (or lower, depending on the problem) values indicate better solutions. In some cases, though, the objective function itself is used directly as the fitness function, without any modification.

- *Population initialization*: This design element is common to all population-based metaheuristics.

The initial population serves as the starting point of the evolutionary process and significantly influences the efficiency of the search. While random initialization is the most common approach, alternative strategies—such as heuristic-based initialization—can enhance solution quality and accelerate convergence. A well-designed initialization method ensures sufficient diversity in the initial population, preventing premature convergence and enabling broader exploration of the search space.

- *Selection mechanism*: The selection procedure determines which individuals will contribute to the next generation, directly influencing the algorithm's convergence behavior. The core principle of selection in EAs is that fitter individuals have a higher probability of being chosen as parents, promoting the propagation of favorable traits. However, maintaining a nonzero probability of selecting lower-fitness individuals is essential to preserving genetic diversity and preventing premature convergence. Various selection strategies exist, each with distinct properties. For instance, roulette-wheel selection assigns selection probabilities proportional to fitness values, favoring stronger individuals while still allowing weaker ones to contribute; rank-based selection reduces bias by determining selection probabilities based on relative ranking rather than absolute fitness; and tournament selection selects individuals through local competitions, providing better control over selection pressure. A comprehensive review of parent selection strategies, along with a detailed discussion of the selection mechanism, can be found in Talbi (2009).

- *Reproduction strategy*: The reproduction process in EAs consists of two fundamental variation operators: crossover and mutation, which work together to generate new solutions from selected parents. Crossover recombines genetic material from two or more parents to create offspring, promoting the inheritance of favorable traits. The effectiveness of crossover depends on the representation used. In binary-encoded solutions, n-point crossover selects n locations in the parent solutions where genetic material is exchanged, maintaining structural integrity while introducing diversity. Uniform crossover, instead, assigns each gene randomly from either parent with equal probability, allowing for greater genetic mixing. For real-valued representations, arithmetic recombination generates offspring by computing a weighted average of the parent values, ensuring smooth transitions in the search space (Talbi, 2009). In contrast, mutation introduces small random alterations in individuals, ensuring population diversity and preventing premature convergence. A common mutation approach is bit-flipping, particularly in binary-encoded solutions. Some algorithms, such as ES, employ self-adaptive mutation, where mutation rates dynamically adjust based on the problem landscape (Talbi, 2009). The probabilities for crossover (p_c) and mutation (p_m) are critical in controlling the balance between exploration and exploitation within the solution space. In particular, the mutation rate must be carefully managed, as excessive mutation can disrupt convergence, while too little may lead to stagnation. For a more comprehensive review of crossover and mutation operators, including their application-specific advantages and role in evolutionary search, see Siew Mooi et al. (2017); Singh and Gupta (2022).

- *Replacement Strategy*: Once offspring are created and evaluated, the next generation is formed by selecting survivors from the current and offspring populations. The choice of replacement strategy

influences the algorithm's convergence behavior and ability to maintain diversity. Following Talbi (2009), two extreme approaches exist: generational replacement, where the entire population is replaced in each iteration, and steady-state replacement, where only a few individuals are updated at a time. Between these extremes, hybrid strategies replace a subset of individuals, striking a balance between convergence speed and diversity preservation. Additionally, many EAs incorporate elitism, which ensures that the best solutions persist in future generations, reducing the risk of losing high-quality individuals. While elitism accelerates convergence, it may also lead to premature stagnation if diversity is not maintained.

To conclude, the stopping criterion must be defined in the evolutionary process. As mentioned earlier, a fixed number of function evaluations or a time limit are commonly used criteria in the literature. The need for a stopping condition is a common requirement across all metaheuristics. Nonetheless, some stopping criteria are specific to population-based metaheuristics and rely on statistical measures of the current population or its evolutionary progress. Most of these criteria assess population diversity and terminate the algorithm when diversity falls below a predefined threshold. These criteria are designed to prevent stagnation, as continuing the search once the population has converged becomes ineffective—further iterations would merely re-evaluate highly similar solutions without producing meaningful improvements (Talbi, 2009).

2.2.2 Taxonomy of EAs

In this section, we briefly compare the four major families of EAs, highlighting their key characteristics and differences. These families are categorized based on their representation schemes and the design of genetic operators:

- *Genetic algorithms*: GAs are one of the most well-known families of EAs, particularly due to their close resemblance to the natural evolutionary process in a computational context. Originally proposed by John Holland in the 1970s, GAs simulate adaptive systems by employing principles of natural selection and genetic recombination. Solutions in GAs are typically represented as strings of numbers, commonly in binary form, though alternative encodings exist. The search process relies on two recombination operators: crossover and mutation. The selection mechanism is probabilistic, ensuring that fitter individuals have a higher chance of survival, and the population typically undergoes generational replacement. Due to their versatility and effectiveness, GAs remain a widely studied optimization approach, with ongoing research focusing on their theoretical foundations, algorithmic improvements, and diverse applications (Katoch et al., 2021).

- *Evolution strategies*: ES, introduced by Rechenberg in 1973, are a class of EAs designed for continuous optimization. Typically, ES represent solutions as real-valued vectors, making them particularly effective for multimodal and non-differentiable functions. Due to their efficiency and adaptability, ES are widely applied in engineering, biochemistry, and complex system optimization (Coello Coello, 2005). The primary variation operator in ES is mutation, which follows a

self-adaptive mechanism. Crossover is rarely used, but when applied, it follows either discrete recombination, where each variable is randomly inherited from one of the parents (similar to uniform crossover in GAs), or intermediary (arithmetic) recombination. The selection operator in ES is deterministic, typically based on fitness ranking, ensuring that the best individuals are chosen as parents. The replacement strategy is more flexible and can incorporate elitist replacement. For a comprehensive review of ES, including recent advancements, existing techniques, key extensions, and their applications to various optimization problems, we refer to Li et al., 2020.

- *Evolutionary programming*: EP, introduced by Fogel in 1966, relies exclusively on mutation as its variation operator and does not incorporate recombination. Originally, EP was developed for evolving finite state machines in time series prediction problems and, more generally, for evolving learning machines (Fogel et al., 1966). Over time, its application expanded to continuous optimization problems, where solutions are represented using real-valued encodings rather than symbolic or binary representations. Similar to ES, EP employs normally distributed mutations with self-adaptive parameter control and a deterministic selection operator. In contrast, the replacement strategy is probabilistic, typically employing stochastic tournament selection or $(\mu+\mu)$ selection (Eiben and Smith, 2003). Due to its similarity with ES, EP has been less widely adopted, but remains relevant for applications such as forecasting, games, automatic control, and optimization of complex systems (Coello Coello, 2005), where its fixed program structure allows the evolution of numerical parameters.

- *Genetic programming*: GP, introduced by Koza in 1992, extends EAs to the space of computer programs (Koza, 1992). Instead of operating on fixed-length strings, GP evolves tree-structured representations, where each tree consists of terminals (input variables or constants) and operators (functions or commands). The structure of the tree varies depending on the specific problem being addressed. The fitness of a GP individual is evaluated based on its performance in a given task, such as a Boolean operation or a classification problem. To evolve better programs, GP employs specialized genetic operators, including subtree crossover, which exchanges tree segments between parents, and mutation, which randomly modifies nodes within the tree. A major challenge in GP is bloat—the uncontrolled growth of tree structures—which increases computational cost and reduces efficiency. Despite its computational demands, GP has been successfully applied in machine learning, data mining, and automated program synthesis (Coello Coello, 2005).

2.2.3 An extension to multi-objective optimization

In the context of multi-objective optimization, multi-objective evolutionary algorithms (MOEAs) extend traditional evolutionary approaches to handle problems with multiple conflicting objectives. Like standard EAs, MOEAs operate on a population of individuals, applying a variety of biologically inspired operators—such as selection, crossover, and mutation—to iteratively refine candidate solutions. However, unlike single-objective optimization, where the goal is to identify the best solution according to a single criterion, MOEAs aim to progressively approximate the Pareto-optimal set, ensuring both

diversity and convergence toward the PF. The development of MOEAs dates back to the 1990s, with early contributions from Kursawe (1990) and Fonseca and Fleming (1993). However, the field experienced significant growth following the seminal book by Deb (2001), which established the foundation for modern MOEAs. Since then, research in this area has expanded rapidly, leading to the development of numerous algorithms, each distinguished by the design of its specific components.

As mentioned earlier, several design elements are shared among single-objective metaheuristics and, in particular, among EAs. Beyond these common concepts, in the context of multi-objective optimization, the search process in MOEAs is structured around three fundamental components: fitness assignment, diversity preservation, and elitism. Fitness assignment plays a crucial role in ranking solutions based on their performance across multiple objectives, guiding the search toward Pareto-optimal solutions. Diversity preservation mechanisms ensure that solutions remain well-distributed along the PF, preventing convergence to a limited region of the objective space. Lastly, elitism guarantees that high-quality solutions are retained across generations, improving convergence while maintaining diversity. A common strategy for implementing elitism in MOEAs is the use of a secondary population, named archive, which stores the best solutions identified during the search. In this regard, Talbi (2009) proposes a unified classification framework for multi-objective metaheuristics, centered on the definition of these three elements. This classification provides a standardized terminology and a structured comparison mechanism, enabling the qualitative evaluation of different algorithms. Moreover, it serves as a foundation for designing new MOEAs, allowing researchers to leverage existing concepts and combine search strategies from various approaches.

In this research, we adopt the general classification proposed by Emmerich and Deutz (2018), which categorizes MOEAs into three main groups based on the paradigm used to design the selection operator: Pareto-based, indicator-based and decomposition-based. It is important to note that this categorization does not refer to specific selection mechanisms—such as roulette wheel or tournament selection (Talbi, 2009)—but rather to the broader framework that governs selection. Consequently, this classification is closely linked to the fitness assignment procedure, which assigns a scalar value to each solution in the multi-objective problem, quantifying its quality and guiding the search toward the Pareto-optimal set. Since this mechanism directly influences selection, it determines which individuals are more likely to be chosen for reproduction, thereby promoting high-quality solutions. Moreover, although fitness assignment is the primary criterion in this classification, some selection paradigms also integrate diversity preservation mechanisms. In the following, we discuss briefly the three main paradigms of MOEAs:

- *Pareto-based MOEAs*: Also referred to as dominance-based approaches, these methods use the Pareto dominance order (Section 2.1) for fitness assignment. Specifically, they employ a two-level ranking scheme that also incorporates diversity preservation. The first ranking step prioritizes non-dominated solutions—those that are not dominated by any other individual in the population—establishing an ordering among solutions. The second ranking step is applied to solutions that share the same position in the first ranking scheme. For these incomparable solutions, diversity is taken into account to ensure a well-distributed approximation of the PF. Since each solution in the population is assigned a single fitness value (or rank), standard design ele-

ments from single-objective EAs can be adapted for multi-objective optimization. For instance, selection mechanisms can be directly incorporated into MOEAs. Moreover, a key advantage of Pareto-based approaches is that they assess solution quality relative to the entire population, rather than assigning absolute values to individual solutions.

- *Indicator-based MOEAs*: These algorithms evaluate and compare approximation sets based on performance indicators (Section 2.1.1), such as the hypervolume, rather than relying on direct dominance relations. Instead of ranking solutions solely based on Pareto dominance, an indicator function assigns a fitness value that reflects the quality of an entire approximation set. This approach can be seen as an extension of the dominance concept, where the goal is to optimize an indicator function relative to a reference set—often the Pareto-optimal front. One advantage of this methodology is that it allows for a more refined comparison between solutions, particularly in cases where dominance-based methods struggle to differentiate individuals. Additionally, indicator-based approaches naturally integrate decision-maker preferences into the optimization process. However, unlike Pareto-based MOEAs, they do not explicitly maintain diversity, as this aspect is typically embedded within the indicator definition. These methods are also known for their robustness, as they exhibit low sensitivity to the characteristics of the PF and generally require fewer algorithmic parameters.

- *Decomposition-based MOEAs*: Instead of using dominance relations or performance indicators, these algorithms belong to the general class of scalar approaches, which convert the multi-objective problem into series of single-objective problems, each focusing on a specific region of the PF (Talbi, 2009). Different scalarization techniques exist, including aggregation (or weighted) methods, or weighted metric approaches. The aggregation methods linearly combine multiple objectives using predefined weights, while weighted metric methods minimize the distance between the feasible region of the objective space and a reference point. The selection process in decomposition-based approaches prioritizes solutions that perform well within their assigned subproblem, ensuring a structured exploration of the PF. However, the effectiveness of these methods is highly dependent on the chosen weight distribution and reference points, as poor parameter selection may lead to weakly Pareto-optimal solutions or an uneven spread of solutions along the PF.

Although this classification provides a general framework for MOEAs, their implementation requires additional considerations. While the selection mechanism determines which individuals are chosen based on criteria such as fitness assigment or diversity preservation, the variation operators—crossover and mutation—must be carefully adapted to the solution representations in the decision space. Hence, the success of a MOEA depends not only on the appropriate design of these components but also on incorporating problem-specific knowledge, as addressed in Chapters 3 and 4, to enhance both efficiency and solution quality.

2.3 Scheduling under uncertainty

Scheduling plays a fundamental role in manufacturing, project management, and service industries, where effective resource allocation and activity sequencing are essential for maintaining operational efficiency. Typical resources—such as equipment, labor, and materials—are often limited and must be carefully managed to meet objectives like minimizing completion time, optimizing resource use, or reducing operational costs. Traditional scheduling approaches usually assume a deterministic environment in which key parameters (e.g., activity durations or resource availability) are known in advance. In this context, project scheduling literature has led to the development of baseline schedules (Demeulemeester and Herroelen, 2002)—also known as predictive schedules or pre-schedules—which serve as reference plans that ensure precedence feasibility, resource allocation, and a structured execution framework while optimizing scheduling objectives. Beyond internal coordination, baseline schedules also support external planning activities such as procurement, subcontractor coordination, and delivery planning, while serving as benchmarks for tracking progress and implementing corrective actions when needed.

Nonetheless, real-world scheduling is inherently uncertain, often leading to deviations from the planned schedule. Sources of uncertainty include fluctuating processing times, unexpected resource shortages, supply chain disruptions, and evolving project constraints. Activities may take longer or shorter than expected, materials may arrive late, and due dates may require adjustments. In some cases, new tasks must be integrated, while others may need to be removed due to changing operational conditions. Additionally, external factors such as weather delays or last-minute customer changes further contribute to scheduling instability. Such disruptions can lead to cascading inefficiencies, including higher costs from missed deadlines, idle resources, increased work-in-progress inventory, and frequent rescheduling efforts that destabilize operations. Given these challenges, static deterministic approaches have been increasingly criticized for their inability to effectively handle uncertainty (Goldratt, 1997). In response, research in operations management, artificial intelligence, and optimization has given rise to methodologies that explicitly incorporate uncertainty into scheduling frameworks, helping systems remain robust, adaptive, and capable of sustaining performance under unpredictable conditions.

To address these challenges, several scheduling methodologies have emerged. Herroelen and Leus (2005) classify scheduling under uncertainty into five main approaches: reactive scheduling, stochastic scheduling, robust scheduling, fuzzy scheduling, and sensitivity analysis. These approaches fall under two broader paradigms: reactive and preventive scheduling. Reactive scheduling focuses on real-time adjustments, modifying an existing schedule during execution to accommodate unforeseen events such as last-minute order changes, cancellations, or equipment breakdowns. In contrast, preventive scheduling anticipates uncertainty by integrating historical data and forecasting techniques to model expected variations in processing times, demand, or resource availability. It provides a structured planning framework that guides resource commitments and timeline decisions, although modifications may still be needed as conditions evolve. Among preventive methods, we distinguish stochastic-based approaches, robust optimization methods, fuzzy programming, and sensitivity analysis. Other taxonomies in the literature align with this classification. For example, Sabuncuoglu and Karabuk (1999)

distinguish between offline and online scheduling. Offline scheduling generates a complete plan before execution begins, assuming full knowledge of the system, while online scheduling adapts dynamically to real-time events as they occur.

As previously mentioned, five different approaches to scheduling under uncertainty can be identified in the literature (Herroelen and Leus, 2005):

- *Reactive scheduling*: Reactive scheduling addresses uncertainty by modifying an initially deterministic schedule in real time when disruptions occur. Rather than anticipating uncertainty, it focuses on restoring feasibility once unexpected events—such as machine failures, processing time deviations, or last-minute changes—take place. Typical methods include local repair strategies that adjust only the affected tasks, such as the classical right-shift rule, or more comprehensive approaches like full rescheduling or match-up scheduling, which aim to reoptimize part or all of the schedule. These techniques prioritize responsiveness and feasibility, often at the expense of long-term optimality. Several reviews of reactive scheduling in the context of manufacturing environments can be found in the literature (see, e.g., Szelke and Kerr, 1994; Vieira et al., 2003). For production scheduling, we refer to Li and Ierapetritou (2008), who summarize the principal reactive scheduling techniques developed to address uncertainty.

- *Stochastic scheduling*: Stochastic scheduling is one of the most widely used approaches for preventive scheduling, where uncertainties are explicitly incorporated into the deterministic model by means of stochastic variables represented by probability distributions. Rather than generating a fixed baseline schedule, it typically defines scheduling policies that adapt dynamically as uncertainty unfolds (see, e.g., Lgelmund and Radermacher, 1983a; Lgelmund and Radermacher, 1983b; Möhring and Stork, 2000). Common formulations include two-stage stochastic programming—where initial decisions are combined with scenario-dependent recourse actions—and chance-constrained programming, which ensures that constraints are satisfied with high probability. These models are particularly suited for problems involving uncertain durations or demand and aim to optimize expected performance metrics, such as makespan or cost. Comprehensive overviews can be found in works such as Chapter 9 in Demeulemeester and Herroelen (2002) and Möhring and Stork (1997).

- *Robust optimization*: Robust scheduling, also referred to as proactive scheduling, aims to construct baseline schedules that maintain feasibility and acceptable performance under a predefined range of uncertain conditions. Unlike reactive approaches, it anticipates disruptions by incorporating protective measures such as buffer times, redundant resources, or robustness constraints. A central methodology within this framework is robust optimization, which formulates the scheduling problem based on worst-case realizations of uncertainty, rather than relying on probabilistic models. The objective is to produce schedules that are both executable and resilient, particularly in environments where rescheduling is costly or impractical. This makes the approach especially suitable when uncertainty is difficult to quantify or historical data is scarce. Applications span production planning, logistics, and project scheduling. Recent developments include budgeted

uncertainty sets, risk-averse decision criteria, and concepts such as recoverable and anchor robustness, which allow limited adjustments during execution while preserving overall robustness.

- *Fuzzy programming*: Fuzzy scheduling provides an alternative to probabilistic models by representing uncertainty through fuzzy logic and possibility theory. Rather than assigning probability distributions to uncertain parameters, this approach models imprecise task durations and resource availabilities using fuzzy numbers and membership functions. It is particularly useful in environments where historical data is scarce or uncertainty is subjective, such as in human-dependent systems or expert-driven planning. In fuzzy scheduling, parameters are treated as fuzzy sets, and constraints are allowed to be partially satisfied, with their degree of satisfaction quantified via membership functions. This enables more flexible decision-making when precise values are difficult to estimate. The existing literature on fuzzy set theory and applications is abundant (see, e.g., Chanas and Kuchta, 1998; Guiffrida and Nagi, 1998; Slowiński and Hapke, 2000). While fuzzy scheduling offers a high degree of modeling flexibility, its effectiveness depends strongly on the quality of the membership functions used to characterize uncertainty. Poorly defined fuzzy parameters can undermine the model's reliability and lead to suboptimal schedules. Recent contributions combine fuzzy logic with metaheuristics or learning-based methods to improve scalability and solution quality in complex scheduling environments.

- *Sensitivity analysis*: Sensitivity analysis is a complementary tool used to assess how variations in input parameters affect the feasibility and performance of a schedule. Rather than generating new schedules, it perturbs specific inputs—such as activity durations or resource availabilities—and analyzes their impact on key outcomes. This helps identify critical tasks or constraints that are particularly sensitive to uncertainty and can inform robustness-enhancing strategies. Notable contributions include early work by Samikoglu et al. (1998), who analyzed the effect of perturbations on optimal solutions and identified parameter regions where feasibility and performance remain stable. Although traditionally used in process optimization, its application to discrete scheduling problems is more limited due to computational complexity and the absence of general frameworks. These issues have been highlighted by Hall and Posner (2004), who pointed to the need for more targeted methodologies. Nonetheless, sensitivity analysis remains valuable for identifying points of fragility and guiding decision-making under uncertainty.

Each of these methodologies presents trade-offs in terms of adaptability, computational complexity, and robustness. Their suitability largely depends on the specific characteristics of the scheduling environment. Reactive scheduling is widely used for its simplicity and real-time responsiveness but may lead to inefficiencies such as frequent rescheduling, increased operational costs, and reduced system stability. In contrast, stochastic and fuzzy approaches enhance flexibility by explicitly modeling uncertainty; robust scheduling emphasizes resilience under worst-case conditions; and sensitivity analysis serves as a diagnostic tool to assess the vulnerability of scheduling decisions. As real-world scheduling problems become more complex, hybrid methodologies that combine the strengths of multiple approaches are gaining traction. These integrated strategies aim to balance robustness and adaptability, offering more reliable performance in uncertain environments. For instance, Wu et al. (2009) propose

a hybrid approach that first generates a robust baseline schedule and subsequently applies reactive adjustments to handle unforeseen disruptions—illustrating how proactive and reactive strategies can be effectively integrated within a unified scheduling framework.

In this thesis, we adopt a robust optimization framework within the preventive scheduling paradigm, where the objective is to anticipate uncertainty and construct baseline schedules that remain feasible under worst-case realizations. Unlike most robust scheduling models in the literature, which primarily focus on uncertainty in activity durations, we address a different but equally significant source of uncertainty: time-dependent cost structures. This type of uncertainty naturally arises in settings such as energy pricing, labor costs, or tariffs that fluctuate over time, and introduces modeling challenges that are not captured by duration-based formulations. Chapter 5 develops a robust formulation tailored to this scenario, incorporating time-indexed cost uncertainty directly into the scheduling framework. By doing so, we contribute to the growing body of literature on robust project scheduling with a novel formulation that better reflects cost-related uncertainties in real-world applications.

2.4 Materials and resources

This section presents the datasets, computational tools, and infrastructure employed throughout the experimental studies conducted in this dissertation. These resources ensure the reproducibility and robustness of the proposed methodologies.

2.4.1 Benchmark datasets

In the context of project scheduling research, several benchmark libraries have been widely adopted by the scientific community to evaluate the performance of optimization algorithms. These libraries provide structured and standardized problem sets that reflect diverse project configurations and resource constraints. In this document, we build on the well-established benchmark instances from the PSPLIB (Kolisch and Sprecher, 1997) and MMLIB (Van Peteghem and Vanhoucke, 2014) libraries, which are employed as a basis for the computational experiments presented in Chapters 3 and 4. In contrast, the instances used in Chapter 5 are generated from scratch, due to the absence of benchmark datasets aligned with the requirements of the problem addressed.

The PSPLIB library is among the most recognized sources in the project scheduling literature. Originally introduced by Kolisch and Sprecher (1997), this library has become a standard benchmark for testing and validating both heuristic and exact methodologies tailored to single- and multi-mode RCPSP. The instances are generated using the project generator ProGen and are available in the PSPLIB library from the ftp server of the Technische Universität München (`http://www.om-db.wi.tum.de/psplib/`). Overall, this library contains several datasets, each labeled according to the number of activities in the project. These datasets are systematically generated considering critical project scheduling parameters of ProGen, which capture both the characteristics of the project network and resource scarcity. Examples of network topology measures include the network complexity (NC) and the order strength (OS), while resource scarcity is quantified using metrics such as the resource factor

(RF) and the resource strength (RS). Precisely, NC represents the average number of non-redundant arcs per node, including those associated with dummy activities. OS quantifies the strength of the partial ordering assumed in the mathematical formulation, measured as the total number of precedence relationships between activities. RF evaluates the proportion of resource requirements per activity, where values closer to 1 indicate higher resource demand. Finally, RS reflects the average tightness of the resource constraints. This value is normalized, with 0 representing the most restrictive scenario, in which resource limitations prevent activities from starting at their earliest possible time.

In the single-mode setting, we find the J30, J60, J90, and J120 datasets. The first three contain 480 project instances with 30, 60, and 90 activities, respectively. The J120 dataset has a total of 600 project instances, each with 120 activities. These datasets assume 4 renewable resources and do not include non-renewable resources. The J30, J60, and J90 datasets are generated using three levels for NC (1.5, 1.8, and 2.1), four levels for RF (0.25, 0.5, 0.75, and 1), and four levels for RS (0.2, 0.5, 0.7, and 1), yielding 48 parameter combinations, with 10 instances per combination. The instances are named as JXY_Z, where X denotes the dataset size (30, 60, and 90), Y ranges from 1 to 48, and Z ranges from 1 to 10. As for the J120 dataset, it considers three NC levels (1.5, 1.8, and 2.1), four levels for RF (0.25, 0.5, 0.75, and 1), and five levels for RS (0.1, 0.2, 0.3, 0.4, and 0.5). All combinations of the previous instance factors lead to a total of 60 possibilities. Lastly, 10 instance replicates are generated for each possible combination, yielding a total of 600 instances. We refer to these instances as J120Y_Z, where Y ranges from 1 to 60, and Z ranges from 1 to 10.

For the multi-mode problem, a wider variety of datasets is available. We focus on the J10, J12, J14, J16, J18, J20, and J30 datasets, which consist of 640 project instances with 10, 12, 14, 16, 18, 20, and 30 activities, respectively. These projects involve 2 renewable and 2 non-renewable resources, and 3 execution modes per activity. NC is set to 1.8 in all instances. Regarding resource scarcity, there are two levels for RF (0.5, and 1) and four levels for RS (0.2, 0.5, 0.7, and 1) for both renewable and non-renewable resources. The original datasets consist of 640 instances, generated using a full factorial design with 10 instances per problem class. However, due to conflicts between resource constraints and execution modes, not all instances are feasible. Infeasible ones are removed, resulting in smaller effective datasets. The instances are labeled as JXY_Z, where X refers to the number of activities (10 to 30), Y ranges from 1 to 64, and Z ranges from 1 to 10.

Despite their widespread use, multi-mode PSPLIB instances have several limitations, such as a narrow range of parameter values for network topology, the presence of infeasible instances, and a high proportion of inefficient execution modes. To overcome these issues, the MMLIB library was developed by Van Peteghem and Vanhoucke (2014). It provides new multi-mode instances, which retain essential structural features from the PSPLIB libray, but which guarantee feasibility, include only efficient modes, and offer a wider range of project topology parameters. The library includes three datasets: MMLIB50, MMLIB100, and MMLIB+. The first two datasets contain projects instances with 50 and 100 activities, respectively. Each project uses 2 renewable and 2 non-renewable resources, with 3 execution modes per activity. Diversity in project topology complexity is ensured by exploring a wide range of OS values (0.25, 0.5, and 0.75). To control renewable and non-renewable resource availability, two RF levels (0.5, and 1) and three RS levels (0.25, 0.5, and 0.75) are considered. Furthermore, all

instances guarantee at least one feasible solution, and only efficient modes are included. By considering a full factorial design and generating 5 instances for each problem class, two datasets of 540 instances each are created. These instances are labeled as JXY_Z, where X has two possible values (50, and 100), Y ranges from 1 to 108, and Z ranges from 1 to 5.

Finally, the MMLIB+ library includes 3240 instances with 50 or 100 activities. Projects in this dataset use either 2 or 4 renewable and non-renewable resources and include 3, 6, or 9 execution modes per activity. The instances are generated by considering three values for OS (0.25, 0.50, and 0.75), three values for RS (0.25, 0.50, and 0.75) for both renewable and non-renewable resources, and a fixed RF of 1 for both resource types. Taking into account all combinations of these factors and five replicates per problem class, a total of 3240 instances are obtained.

In summary, the selected datasets from the PSPLIB and MMLIB libraries provide a robust and diverse experimental foundation for constructing the instances used in this thesis. While these benchmark problems are adapted to address the specific requirements of each proposed methodology, their structured design, extensive parameter coverage, and widespread acceptance in the literature ensure a consistent and meaningful basis for performance evaluation. For a comprehensive overview of the original benchmark instances, the reader is referred to Kolisch and Sprecher (1997) and Van Peteghem and Vanhoucke (2014).

2.4.2 Computational infrastructure

The computational experiments presented in this doctoral thesis were conducted using high-performance computing resources from the Miguel Hernández University of Elche (UMH). Specifically, two different computing infrastructures were employed. For the experiments reported in Chapters 3 and 5, the Dantzig Cluster at UMH was utilized, which operates under Rocky Linux release 8.7. All executions were performed exclusively on the g-0 node, a Supermicro SYS-120GQ-TNRT equipped with two Intel(R) Xeon(R) Silver 4316 CPUs at 2.30 GHz (80 cores in total) and 768 GB of RAM.

For the experiments in Chapter 4, the Cluster of Scientific Computing at UMH (`https://ccc.umh.es/`) was employed. In particular, the PRMTO-CIO-0 node was used, based on the Supermicro SYS-1029GQ-TRT model with two Intel(R) Xeon(R) Gold 6242R CPUs running at 3.10 GHz (80 cores in total) and 768 GB of RAM.

The codes developed in this thesis are implemented in the programming languages Java (Arnold et al., 2005), `R` (R Development Core Team, 2024), and C++ (Stroustrup, 2013), and can be provided upon request. Additionally, all datasets used in the computational experiments are available through open-access repositories, which are referenced in the corresponding chapters.

Chapter 3

Metaheuristics for the bi-objective resource-constrained project scheduling problem with time-dependent resource costs

The integration of time-dependent resource costs into project scheduling introduces significant complexity, requiring advanced optimization techniques to achieve high-quality solutions. This chapter explores the development and comparison of metaheuristic approaches for addressing a challenging variant of the resource-constrained project scheduling problem in a multi-objective context. The aim is to evaluate different optimization paradigms and provide insights into their strengths and weaknesses when solving complex scheduling problems.

3.1 Motivation and structure

Numerous extensions of the resource-constrained project scheduling problem (RCPSP) have been developed to better reflect real-world complexities, including multi-objective formulations and time-dependent resource considerations. This chapter presents an extensive computational study that evaluates several metaheuristic paradigms for solving the bi-objective RCPSP with time-dependent resource costs (RCPSP_TDRC). Building upon the recent work of Alcaraz et al. (2022), who introduced this problem variant and proposed a solution based on the non-dominated sorting genetic algorithm II (NSGA-II; Deb et al., 2002), we extend their analysis by implementing six additional multi-objective evolutionary algorithms (MOEAs) representing diverse optimization paradigms: the strength Pareto evolutionary algorithm 2 (SPEA2; Zitzler et al., 2001), the multi-objective optimization cellular genetic algorithm (MOCell; Nebro et al., 2009), the Pareto envelope-based selection algorithm II (PESA-II; Corne et al., 2001), the indicator-based evolutionary algorithm (IBEA; Zitzler and Künzli, 2004), the $\mathcal{S}$-metric selection evolutionary multi-objective optimization algorithm (SMS-EMOA; Emmerich et al., 2005), and the multi-objective evolutionary algorithm based on decomposition (MOEA/D; Zhang and

Li, 2007). This selection encompasses a broad range of metaheuristic strategies, ensuring a comprehensive comparison across different approaches.

All algorithms share the same encoding and operator mechanisms originally developed for the NSGA-II approach, ensuring that the comparison focuses on the performance of the different paradigms themselves rather than implementation-specific differences. Conducting such comparative studies is essential for identifying the most effective algorithms for a given problem (see, e.g., Schlünz et al., 2016; Danloup et al., 2018; Govindan et al., 2019). As the RCPSP_TDRC is a bi-objective problem, each algorithm yields an approximation of the Pareto front (PF). Zitzler et al. (2003) emphasize the importance of quantitative comparisons between approximation sets, highlighting the challenges involved in multi-objective optimization.

The experimental study is based on problem instances derived from the classic PSPLIB library (Kolisch and Sprecher, 1997), modified to reflect the characteristics of the bi-objective RCPSP_TDRC. These instances cover both medium-sized and large-scale project scenarios. To evaluate performance, five well-established indicators are used to provide a comprehensive assessment of the PF approximations: the μ-indicator (μ), spread (Δ), additive ϵ-indicator (I_{ϵ^+}), hypervolume (HV), and modified inverted generational distance (IGD^+). In addition, statistical analyses are performed to ensure that the results can be compared with a certain level of confidence.

The contents of this chapter correspond to the contributions by Rodríguez-Ballesteros et al. (2024), which are twofold. First, we broaden the set of available solution methods for the bi-objective RCPSP_TDRC by implementing six additional MOEAs beyond the NSGA-II-based technique originally proposed by Alcaraz et al. (2022). Second, we conduct a rigorous comparative analysis of these paradigms across a large number of problem instances, supported by a calibration phase that ensures all algorithms are configured on an equal footing. The results are evaluated using well-established performance indicators and statistically validated to ensure reliable conclusions.

The chapter is organized as follows. Section 3.2 introduces the bi-criteria RCPSP_TDRC, presenting its mathematical formulation and key concepts in multi-objective optimization. The seven metaheuristics considered in the computational study are described in Section 3.3, including both their common components and specific characteristics. Section 3.4 presents and analyzes the experimental results. Finally, conclusions and future research directions are given in Section 3.5.

3.2 Problem description

As we have stated previously, this chapter aims at showing a comprehensive comparison of several metaheuristic algorithms in the context of multi-objective optimization. In this section, we present the problem on which we center our study.

The RCPSP is a well-known combinatorial optimization problem that involves scheduling a set of n activities, $V = \{1, \ldots, n\}$, subject to precedence and resource constraints over a finite time horizon $\mathcal{T} = \{0, \ldots, T-1\}$. Each activity $j \in V$ has an associated processing time d_j. Preemption is not allowed, meaning that once an activity starts, it must be executed continuously for d_j time periods. Precedence relations between activities are defined by the set P of immediate predecessor pairs, where

$(i, j) \in P$ indicates that activity j cannot start until activity i is completed. Renewable resources are represented by the set $\mathcal{R}$, where each activity $j \in V$ requires r_{jk} units of resource $k \in \mathcal{R}$ for each time unit of its execution. The constant availability of each resource k per time period is denoted by R_k. If necessary, two additional activities, 0 and $n+1$, are included to represent the start and completion of the project, respectively. These are "dummy" activities with zero processing time and no resource consumption.

Formally, a schedule $S = (S_1, \ldots, S_n)$ is an assignment of a non-negative starting time S_j to each activity $j \in V$. Traditionally, the objective of the RCPSP is to find a schedule which results in the earliest possible end of the project, i.e., the completion time of end activity (makespan), and both the precedence and the resource constraints are satisfied. In their work, Pritsker et al. (1969) propose a model for this problem based on the maximum time required to complete all the activities, the so-called planning horizon T, which should represent a sharp upper bound of the project optimal makespan. If not, this horizon is set to the sum of the duration of all activities.

At this point, we need to define some elements in order to present the mathematical formulation of the RCPSP. Moreover, given an activity $j \in V$, we define its earliest and latest starting times, denoted by E_j and L_j, respectively. We state the former as the minimum time required to execute all the direct and transitive predecessors of j, while the latter is the maximum time j can be held up without causing a delay in the project, i.e., the duration of the project surpasses the planning horizon, T. Note that L_{n+1} is set equal to T. Lastly, the binary variables of the model are defined as $x_{jt} = 1$ if activity $j \in V$ starts at time $t \in \{E_j, \ldots, L_j\}$; otherwise, $x_{jt} = 0$. Under these conditions, the RCPSP can be formulated as follows:

$$\text{minimize} \quad \sum_{t=E_{n+1}}^{L_{n+1}} t\, x_{(n+1),t}, \tag{3.1}$$

$$\text{subject to} \quad \sum_{t=E_j}^{L_j} x_{jt} = 1, \qquad \forall j \in V, \tag{3.2}$$

$$\sum_{t=E_j}^{L_j} t x_{jt} - \sum_{t=E_i}^{L_i} t x_{it} \geq d_i, \qquad \forall (i,j) \in P, \tag{3.3}$$

$$\sum_{j \in V} r_{jk} \cdot \sum_{q=\max\{t-d_j+1, E_j\}}^{\min\{t, L_j\}} x_{jq} \leq R_k, \quad \forall k \in \mathcal{R}, t \in \mathcal{T}, \tag{3.4}$$

$$x_{jt} \in \{0, 1\}, \qquad \forall j \in V, t \in \{E_j, \ldots, L_j\}. \tag{3.5}$$

First of all, the makespan is defined in (3.1). Constraints (3.2) guarantee the execution of every activity of the project, beginning at a particular time period between its earliest and latest starting times. Constraints (3.3) refer to the precedence relationship, ensuring no activity starts before its predecessors have been completed. The resource constraints are included in (3.4), making sure the availability of each resource is not exceeded during any time period. Finally, (3.5) restrict the decision variables to the subset $\{0, 1\}$.

Taking the RCPSP and the model by Pritsker et al. (1969) as a starting point, Alcaraz et al. (2022) introduce time-dependent resource costs, i.e., the cost depends on the resource being considered as well

as the time period in which it is being used. This type of costs are very common in projects when scarce resources are considered. In this context, they propose adding a second objective to be minimized in the problem, the total cost of the resource usage. Referring to (3.1) as $f_1(\mathbf{x})$, the second objective function is formally define as:

$$f_2(\mathbf{x}) = \sum_{j \in V} \sum_{t=\max\{0,E_j\}}^{\min\{T-1,L_j\}} \left[x_{jt} \sum_{q=t}^{t+d_j-1} \sum_{k \in \mathcal{R}} r_{jk} c_{kq} \right], \tag{3.6}$$

where c_{kt} is the cost of employing one unit of renewable resource k during the interval of time $[t, t+1)$, $\forall k \in \mathcal{R}$ and $t \in \mathcal{T}$.

In this way, the authors define a new variant of the RCPSP: a bi-objective problem that incorporates time-dependent resource costs, known as the bi-objective RCPSP_TDRC. This problem inherits the NP-hard complexity of the classical RCPSP, and can be formulated as follows:

$$\begin{aligned} &\text{minimize} && \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ &\text{subject to} && (3.2) - (3.5), \end{aligned} \tag{3.7}$$

where $\Omega = \{\mathbf{x} \in \{0,1\}^{J \times \mathcal{T}} : (3.2) - (3.5)\} \subseteq \mathbb{R}^{J \times \mathcal{T}} \neq \emptyset$ is the feasible set, and $f_i : \{0,1\}^{J \times \mathcal{T}} \to \mathbb{R}$ are the objective functions, for $i \in \{1,2\}$. The decision space and objective space are given by $\mathbb{R}^{J \times \mathcal{T}}$ and $\mathbb{R}^2$, respectively.

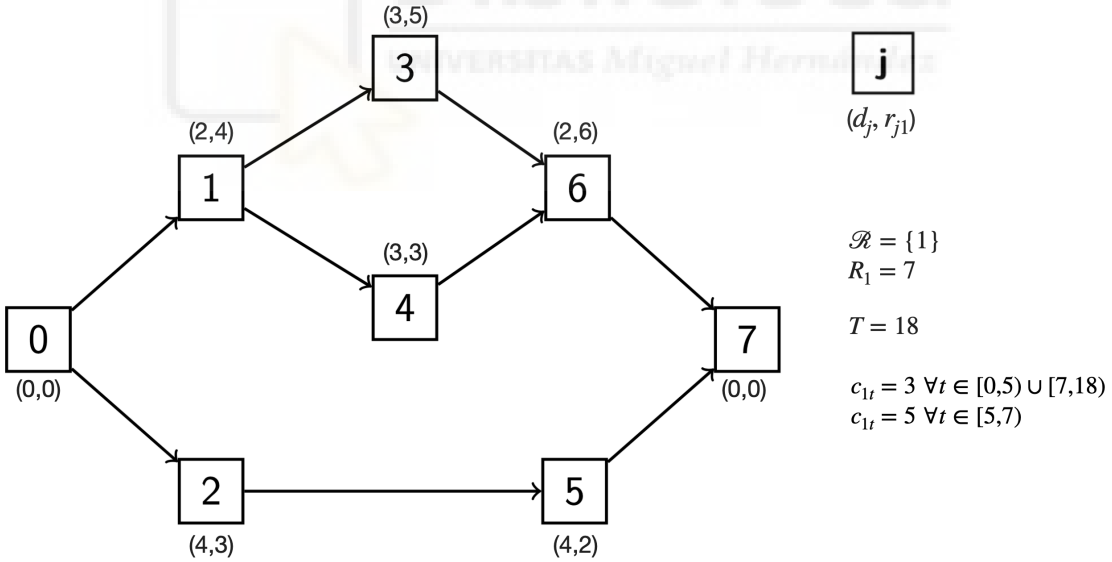


Figure 3.1: An example of the RCPSP with time-dependent resource costs.

An example of the above problem is given in Figure 3.1. The project comprises 6 activities that utilize a single renewable resource with an availability of 7 units per period. Each activity is paired with its duration and the resource requirement. Dummy activities 0 and 7 are included to represent project start and completion, and the planning horizon is determined by summing the processing times

of all activities in the project. Finally, the time-dependent resource costs are shown in the figure with two possible values.

In the presence of two objectives, makespan and total cost of resource usage, we can obtain different solutions to the problem, depending on the objective we are prioritizing. In Figure 3.2, we present two feasible solutions and calculate their respective objective values. The schedule shown on the left achieves a lower makespan value compared to the one on the right, while the cost of resource usage exhibits the opposite trend. Hence, both solutions are incomparable, in the sense that no objective can be improved without deteriorating the other (see Section 2.1).

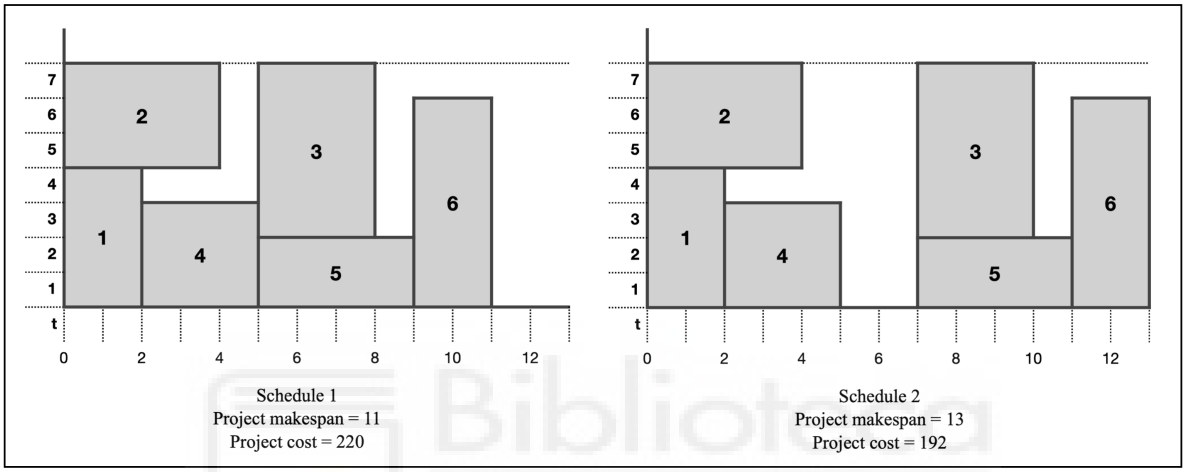


Figure 3.2: Different solutions to the problem in Figure 3.1.

Alcaraz et al. (2022) demonstrate that exact techniques fail to solve medium- and large-sized instances of this problem and propose a metaheuristic based on NSGA-II as an alternative. Specifically, the authors compare the PF approximations obtained by the metaheuristic with those provided by an exact method, namely the AUGMECON algorithm (Mavrotas, 2009). This exact approach is described in detail in Section 4.3. For small and medium-sized instances (with 30 and 60 activities, respectively), where exact techniques can solve the optimization models to proven optimality, the metaheuristic demonstrates its efficiency by achieving comparable results with significantly lower computation times. For instances where the exact method fails to find the PF, the metaheuristic provides approximations with very good characteristics.

Building on these experiments and aiming to compare several metaheuristic paradigms, this chapter focuses primarily on large-sized projects with 120 activities. However, to support the conclusions more comprehensively, experiments on medium-sized projects with 60 activities are also included. All metaheuristics implemented in this study follow the general template introduced in Section 2.2.1, which provides a unified framework for evolutionary algorithms (EAs). Importantly, we have selected seminal variants of each metaheuristic, allowing the comparison to focus on the effectiveness of the underlying paradigms rather than on specific algorithmic refinements. Based on the common evolutionary framework and the problem-specific requirements discussed above, the following section presents the adaptation of these paradigms to this specific scheduling problem.

3.3 Multi-objective evolutionary algorithms

As discussed in Section 2.2, EAs provide a robust framework for solving complex optimization problems. In the context of multiple conflicting objectives, MOEAs have emerged as a natural extension of these techniques, forming a powerful and flexible family of metaheuristics widely adopted in the literature for approximating the PF. This section focuses on the design and implementation of MOEAs tailored to solve the bi-objective RCPSP_TDRC.

While MOEAs share a common structure—including population-based search, fitness assignment, and diversity preservation—their effective application to a specific problem requires adapting certain algorithmic components. In particular, the solution encoding and the variation operators, crossover and mutation, must be carefully designed to incorporate problem-specific knowledge. These components, originally proposed by Alcaraz et al. (2022) for the NSGA-II approach, are shared across all algorithms considered in this study, ensuring a fair and consistent experimental comparison.

We first describe the problem-specific components shared by all algorithms. Subsequently, we present the seven MOEAs considered for experimental evaluation, following the classification outlined in Section 2.2.3: Pareto-based, indicator-based, and decomposition-based approaches. Among these three paradigms, most MOEAs proposed in the literature to solve a wide range of optimization problems belong to the Pareto-based category. In particular, the most widely used and well-established algorithms fall into this group. Accordingly, the majority of the MOEAs included in our study—NSGA-II, SPEA2, MOCell, and PESA-II—are Pareto-based. To broaden the comparison across all three categories, we also consider two indicator-based algorithms, IBEA and SMS-EMOA, as well as one decomposition-based algorithm, MOEA/D. This selection enables a comprehensive comparison of the main design paradigms in multi-objective evolutionary optimization.

3.3.1 Specific components for the RCPSP_TDRC

As stated above, Alcaraz et al. (2022) propose a metaheuristic to solve the bi-criteria RCPSP_TDRC. Specifically, the authors design a solution encoding and genetic operators to implement the algorithm and solve the problem being considered. These features, which are described in detail by Alcaraz et al. (2022), are summarized below:

- **Activity list with scheduling objective.** The solutions are encoded by a double-list, an activity list (AL) (Hartmann, 1998) where each activity must appear after all its predecessors and a binary list representing, for each activity, one of the two objectives considered: the makespan or the total cost of resource usage. This encoding is an extension of the one proposed by Alcaraz and Maroto (2006) for dealing with the RCPSP with only one objective. The order given by the AL is followed when scheduling the activities in order to build the schedule. Nevertheless, the binary list determines which scheduling objective we prioritize for each activity. Thus, when prioritizing the makespan, activities must be executed the moment there is availability of resources, whereas when considering the second scheduling objective, i.e., the cost, the starting time of an activity can be delayed a maximum number of time periods until its processing cost is cheaper, provided

that there are sufficient resources to execute it.

1	2	4	3	5	6
0	0	0	1	1	0

Figure 3.3: Example of the AL with scheduling objective representation.

The design of the solution encoding is crucial, since the good performance of the algorithm depends, to a great extent, on it. Figure 3.3 shows a solution encoded with the AL with scheduling objective representation for the project example presented in Figure 3.1. We observe that the first list orders activities according to their precedence relationships in the project, i.e., an activity will always appear after all its predecessors. In the second list, activities 1, 2, 4, and 6 prioritize minimizing the makespan when they are scheduled, as their scheduling objective is 0. Meanwhile, activities 3 and 5 have a scheduling objective of 1, indicating a priority on the cost when they are going to be placed in the schedule. More examples of this double-list encoding and its subsequent transformation into a schedule are available in the work by Alcaraz et al. (2022).

- **Double-list crossover.** The crossover operator is applied to a pair of solutions, where information from each double-list is combined to create an offspring. To achieve this goal, a two-step procedure is carried out. In a first stage, the AL of the parents undergo the two-point crossover (Hartmann, 1998). Next, the activities conforming the offsprings inherit the scheduling objectives from their parents, taking into account the provenance of each activity.

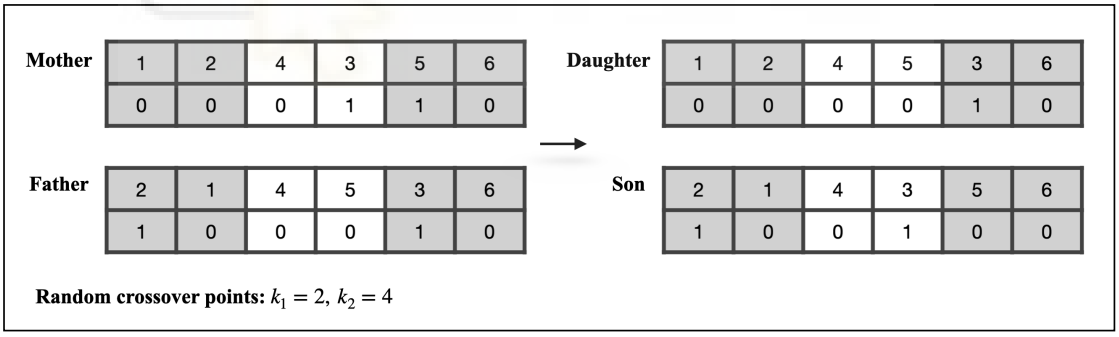


Figure 3.4: Example of the double-list crossover.

Figure 3.4 illustrates an example of the double-list crossover procedure. In this example, we have two solutions derived from the project shown in Figure 3.1: the mother and the father. The crossover operation begins by choosing two random crossover points k_1 and k_2. Let us suppose that $k_1 = 2$ and $k_2 = 4$ are selected. The double lists of the parents are divided into three sections, resulting in the daughter inheriting the first two positions in the AL from the mother. The next two positions are inherited from the father, following a specific rule: we identify the first two activities in the father that are not already present in the daughter, maintaining their relative order in the father's list. Finally, the last two activities are taken from the mother,

ensuring that their relative order on the mother is maintained. The procedure for generating the son's AL is analogous to that of the daughter but interchanging the role of the parents. After the crossover of the ALs of the parents, the activities in the offspring inherit the scheduling objective from the respective parent they were copied from.

- **Double-list mutation.** As for the crossover operator, the mutation mechanism is performed in two steps. First, each of the activities in the sequence is moved to a random position—between the last of its predecessors and the first of its successors, generating a feasible solution—with a determined mutation probability (see, e.g., Boctor, 1996; Alcaraz and Maroto, 2001). The scheduling objective list of all the activities is preserved after this procedure. Afterwards, the binary list is altered with certain probability, changing from 0 to 1 or vice versa.

The features described above make the algorithm unique, and its performance strongly depends on them. These features will be employed as a common basis in all the metaheuristics compared in this chapter. Furthermore, in all the metaheuristics, the procedure begins by creating an initial population through the same random mechanism. First, an AL is generated by randomly selecting activities from an eligible list while maintaining the precedence relationships. After selecting an activity, the eligible list is updated to ensure that no activity is chosen more than once. Second, the scheduling objective for each activity is randomly determined from the two possible values with a 50% probability for each. Once the initial population is created, we evaluate each solution, i.e., we obtain the values of makespan and cost for every individual.

With this setup, any differences in performance among the metaheuristics cannot be attributed to the solution encoding, the specific genetic operators, or the initialization procedure, but rather to the general scheme of each algorithm. The specific characteristics of the metaheuristics considered in this chapter are presented in the following sections.

3.3.2 NSGA-II

Within the Pareto-based MOEAs, we begin by describing the non-dominated sorting genetic algorithm II (NSGA-II) introduced by Deb et al. (2002), which has been successfully applied to a wide range of problems across diverse contexts. First, the random initialization mechanism described earlier is used to generate the initial population, that is, the set of candidate solutions. Each individual is then decoded into a schedule, and its objective values—makespan and total cost—are computed.

At this point, the two-level ranking scheme, known as the crowded-comparison operator (Deb et al., 2002), is applied to guide the selection process toward a uniformly spread-out PF. Without delving too deeply, this operator assigns each individual two attributes: the non-domination rank and the crowding distance. The non-domination rank involves assigning a fitness (or rank) to each solution based on the non-domination relationship. Specifically, the population is sorted into different fronts: individuals in the first front are not dominated by any other solution in the population and are assigned rank 1; individuals in the second front are only dominated by members of the first front and are assigned rank 2; and so on. In this way, the rank of each solution corresponds directly to the front it belongs to, with lower ranks indicating better fitness. The crowding distance, in turn, represents a density

estimation, calculating the distance between a given solution and the rest of the solutions in the same front. A higher crowding distance value is desirable. That is, between two solutions with different non-domination ranks, we prefer the solution with the lower rank. However, if both solutions belong to the same front, we prefer the solution located in a less crowded area of the objective space.

The evolutionary process then proceeds by applying typical genetic operators—binary tournament selection, crossover, and mutation—to produce an offspring population of the same size as the initial one. Both populations are combined resulting in a double-sized population, on which the non-domination ranking process is again performed. This procedure yields several layers, so that the best solutions in the combined population are the ones belonging to the first front. In order to build a new population with the original size, individuals are selected front by front, starting with the best-ranked (first) front. If the number of solutions in the first front is less than the original population size, all the individuals are chosen for the new population. As for the remaining members, they are selected from the subsequent non-dominated fronts, until no more layers can be fully accommodated. When all the individuals of a front cannot be placed in the new population, the crowding distances of the individuals in that front are computed, and the solutions are sorted in descending order. The top individuals are then selected until the new population is fulfilled.

In summary, the first-level ranking identifies preferable solutions based on Pareto dominance, while the second-level ranking, based on crowding distance, ensures diversity among individuals within the same front. For further details on this metaheuristic, we refer to Deb et al. (2002).

3.3.3 SPEA2

We shall now present the improved version of the strength Pareto evolutionary algorithm (SPEA) (Zitzler and Thiele, 1999), namely SPEA2, which is a Pareto-based MOEA for finding or approximating the Pareto-optimal set of solutions. Attempting to overcome the shortcomings of its predecessor, SPEA2 incorporates a fitness assignment procedure, which considers for each solution the individuals dominated by the solution as well as the ones that dominate it. Moreover, SPEA2 employs a regular population and an archive. The archive is an external population which stores the non-dominated solutions attained by the algorithm during the optimization process. Precisely, it serves as a representation of the non-dominated front among all the evaluated solutions considered so far. Following the ideas presented in Zitzler et al. (2001), let $\mathbf{x}$ be an individual belonging to either the archive, $\overline{P}$, or the population, P. We denote by $S(\mathbf{x})$ the strength value of $\mathbf{x}$, and it can be calculated as follows:

$$S(\mathbf{x}) = |\{\mathbf{y} : \mathbf{y} \in P + \overline{P} \wedge \mathbf{x} \prec \mathbf{y}\}|,$$

where $|\cdot|$ denotes the cardinality of a set, $+$ represents the multiset union and $\prec$ stands for the Pareto dominance relation (see Section 2.1). Thus, the strength value of an individual represents the number of population members it dominates. Subsequent to the above, the raw fitness $R(\mathbf{x})$ of an individual $\mathbf{x}$ is obtained:

$$R(\mathbf{x}) = \sum_{\mathbf{y} \in P + \overline{P}, \mathbf{y} \prec \mathbf{x}} S(\mathbf{y}).$$

Hence, the raw fitness of a solution $\mathbf{x}$ is determined by the strength values of the individuals that dominate it. Although the raw fitness reflects the extent to which an individual is dominated, it may fail when most of the individuals in a population do not dominate each other. For that reason, a density estimation based on the nearest neighbor technique is combined with the raw fitness in order to discriminate between population members with the same raw fitness. As a result, the selection process is biased towards minimizing the fitness values, thus preferring the exploration of less populated regions of the objective space. A detailed description on how to compute the fitness of an individual $\mathbf{x}$—by summing the two components described above—can be found in Zitzler et al. (2001).

Starting with an initial population and an empty archive, we now describe the main loop of this algorithm. First of all, the fitness values of the individuals in both, the original population and the archive, are calculated. We refer to this step as the fitness assignment. Then, the non-dominated individuals are copied into the archive of the next generation. At this point, the environmental selection procedure takes place. If the number of non-dominated solutions is greater than the archive size, a truncation operator based on calculating the distances to the k-th nearest neighbor is employed. Otherwise, the archive of the next generation is filled with dominated solutions. Finally, similarly to other MOEAs, the algorithm continues to the mating selection phase, where the mating pool is created by selecting individuals belonging to the previous archive. Afterwards, the variation procedure takes place, and the crossover and mutation operators are applied to the mating pool in order to obtain the resulting population. If a stopping criterion is satisfied, the process stops and the obtained archive which contains the non-dominated individuals will represent the approximation of the Pareto-optimal set, that is, the solution of our problem. Again, we refer to Zitzler et al. (2001) for a more detailed explanation of this methodology.

3.3.4 MOCell

Belonging to the Pareto-based MOEAs class, we now discuss a cellular genetic algorithm (cGA) called MOCell (Nebro et al., 2009), which is based on two chief components: a bounded external archive to store the non-dominated solutions and the small neighborhoods. Here, we concentrate on the concept of small neighborhood, as it constitutes a fundamental element of this technique. When we refer to a neighborhood, we address how individuals interact with each other and, in particular, which kind of cooperation is allowed and which is not. Hence, genetic operators may only be applied to an individual and its immediate neighbors, i.e., within the neighborhood. A neighborhood, in this sense, is a subset of solutions or individuals within the overall population that are considered "close" to each other in terms of their objective values. In this context, we can assume population size is greater or even much greater than the size of a neighborhood. Moreover, this algorithm works with overlapping small neighborhoods within which solutions are shared, inducing a gradual exploration of the search space that guarantees diversification. Besides, exploitation (intensification) occurs inside each neighborhood by means of genetic operators (Alba and Dorronsoro, 2008).

Regarding the breeding loop of MOCell, it starts by considering a bounded and empty external archive and by arranging the population members in a two-dimensional toroidal grid. Next, for each

individual, two parents are selected from its neighborhood and undergo the crossover and mutation operators to obtain an offspring. At this stage, the resulting individual is compared to the current one, making use of the crowded-comparison operator introduced in NSGA-II. The current individual is replaced if it is dominated by the offspring. On the other hand, if both individuals are not comparable according to the Pareto dominance relation, a population made up of nine neighbors, using the *Compact*9 criterion (see, e.g., Whitley, 1993; Alba and Dorronsoro, 2008; Jie et al., 2017) is considered. Therefore, the crowding distance of both individuals in the reduced population is computed, whereby the individual with the worst value is rejected. Lastly, the altered population replaces the old one after undergoing a feedback process where several individuals are selected from the archive with the purpose of replacing the corresponding randomly selected ones in the new population.

With regard to the external archive, a ranking based on the crowding distance is performed to decide whether a non-dominated solution is inserted into the archive once it is already full. Specifically, the individual with the worst crowding distance value is discarded, ensuring that the diversity of solutions on the PF is preserved.

3.3.5 PESA-II

To conclude with the Pareto-based MOEAs we present PESA-II (Corne et al., 2001), a version of the Pareto envelope-based selection algorithm (PESA) (Knowles and Corne, 2000), which outperforms its predecessor providing significantly superior results. Both algorithms share the habitual structure of MOEAs, already presented in previous techniques, in which genetic operators act to gradually approximate the PF. Nevertheless, differences arise when it comes to the selection procedure followed by each of them. In this section, we firstly provide an overview of PESA as it represents the basis of PESA-II, and subsequently describe the latter.

Individual-based selection employed in PESA relies on the adaptive hyper-grid division adopted in Knowles and Corne (1999), i.e., a subdivision of the objective space into hyperboxes within which the number of individuals is computed. Notice that the adaptive grid depends on the number of bi-sections considered, S, which conforms a configurable parameter of this algorithm. For each individual, this method keeps a track of the number of other solutions inside the same hyperbox, the so-called selective fitness or squeeze factor. After this fitness assignment procedure, a general selection scheme is used, through which the parents for genetic operators are chosen among the individuals with small squeeze factors. Furthermore, an external archive stores non-dominated solutions and utilizes the adaptive grid discussed earlier as a density estimator for the selection process.

In contrast, PESA-II employs an internal population from which parents are selected to generate offspring, and an external population to store non-dominated solutions. The external population also relies on the adaptive hyper-grid division of the objective space adopted by PESA. As for the internal population, a region-based selection scheme is applied. In region-based selection, the unit of selection is a hyperbox rather than an individual. Therefore, the procedure consists of selecting a hyperbox by any traditional selection method, within which an individual is randomly chosen. This alteration in the selection procedure leads the algorithm to choose isolated individuals instead of non-isolated ones

with higher probability than in PESA (Corne et al., 2001.) The resulting individuals are subjected to the crossover and mutation operators. Finally, in the environmental selection process, several non-dominated individuals from the current population are asked to enter in the archive one by one. If an individual is not dominated by any solution in the archive, then it is included in the external set and all the archive members dominated by the new one are removed from the set. Finally, if the archive is full, the density of the hyperboxes—measured by the number of solutions they contain—is used to remove the individual located in the most crowded hyperbox.

3.3.6 IBEA

The indicator-based evolutionary algorithm (IBEA) (Zitzler and Künzli, 2004) is an indicator-based MOEA which employs binary performance indicators adaptable to arbitrary decision maker's preferences. A binary performance indicator is defined as a function that maps two PF approximations to a real number. These indicator functions serve as performance measures that enable comparison between the quality of two approximation sets. One well-known example is the binary additive ϵ-indicator, I_{ϵ^+} (Zitzler et al., 2003), which returns the minimum distance by which a PF approximation must be translated in the objective space in order to weakly dominate another approximation set (see Section 2.1.1). In particular, binary performance indicators can be used for the fitness assignment procedure directly as they constitute a natural extension of the Pareto dominance relation. In IBEA, the fitness assignment of an individual $\mathbf{x}$ is performed as follows:

$$F(\mathbf{x}) = \sum_{\mathbf{y} \in P \setminus \{\mathbf{x}\}} -e^{-I(\{\mathbf{y}\},\{\mathbf{x}\})/(c \cdot \kappa)},$$

where I represents the binary quality indicator, P refers to the population, the parameter κ is a scaling factor and c represents the maximum absolute indicator value. Note that in this chapter we focus on the adaptive IBEA template, where a binary quality indicator based on the hypervolume is used for fitness assignment. This type of measure quantifies the portion of the objective space that is dominated by one solution set but not by another, relative to a predefined reference point. We refer the reader to Zitzler and Künzli (2004) for a detailed explanation of this approach and a formal definition of the indicator. Moreover, the authors demonstrate that this fitness assignment scheme satisfies the property of dominance preservation, i.e., it is compliant with the Pareto dominance relation.

Next, we briefly describe the main loop of the IBEA algorithm. Firstly, the initialization phase generates an initial population, followed by the fitness assignment scheme procedure, where the objective and indicator values are scaled. Hence, a fitness value is assigned to each population member and the individual with the worst value is removed from the population. Afterwards, the fitness values of the remaining individuals are updated. The former environmental selection step is iterated until a fixed population size is reached. At this stage, the algorithm must decide whether the procedure ends or whether it continues with the mating selection and variation steps. If no stopping criterion satisfied, the selection operator is applied to the population in order to fill the mating pool. Hereafter, this mating pool undergoes the crossover and mutation operators in order to obtain an offspring population. Again, we refer to Zitzler and Künzli (2004) for further details of the algorithm.

3.3.7 SMS-EMOA

Regarding the indicator-based MOEAs, we conclude with the $\mathcal{S}$-metric selection evolutionary multi-objective optimization algorithm (SMS-EMOA) (Emmerich et al., 2005). As previously stated, binary performance indicators allow us to compare the quality of two approximation sets. Nevertheless, there are performance indicators which compute the quality of a PF approximation, providing an absolute measure of its quality. We refer to the latter as unary performance indicators or just unary indicators. In particular, the hypervolume measure or $\mathcal{S}$-metric (Zitzler, 1999) is a function of such type. SMS-EMOA evolves around this unary indicator, which determines the selection procedure biasing the search towards the set of non-dominated solutions or Pareto-optimal solutions. Precisely, the hypervolume indicator calculates the dimensions of the dominated space regarding a reference point, which bounds it and must be fixed in advance. See Section 2.1.1 for a formal definition of this performance indicator. Obviously, a set of non-dominated solutions with a large number of points that are well-distributed along the PF would attain a vaster dominated area.

SMS-EMOA aims at maximizing the hypervolume indicator of a population of size N, in which the employment of an external archive is not necessary. As we contend with a multi-objective scenario, the proposed algorithm adopts a ranking criterion based on the non-domination relation, as in NSGA-II. Therefore, the selection procedure, which is governed by the $\mathcal{S}$-metric, takes place at the worst-ranked layer derived from the ranking scheme. Moreover, it must be emphasized that we are facing a steady-state $(N+1)$-EMOA, meaning only one offspring results from the genetic operators per iteration. Hence, population size is increased to $N+1$, so one individual must be removed from the set by the end of the loop. In a first scenario, if all the population members are non-dominated, the individual with the smallest hypervolume contribution is removed from the set of solutions. Nonetheless, in the presence of dominated solutions, SMS-EMOA employs the aforementioned ranking scheme and selects the solution with the smallest hypervolume contribution from the worst-ranked layer. We refer to Emmerich et al. (2005) for a detailed explanation of this metaheuristic, including its pseudocode.

3.3.8 MOEA/D

So far, we have described several MOEAs, including Pareto and indicator-based ones. Henceforth, we comment on an algorithm of the remaining category, that is, a multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li, 2007). This metaheuristic consists in decomposing a multi-objective optimization problem into N scalar optimization subproblems to be optimized simultaneously. Ideally, each of these subproblems attaches a specific region of the objective space and, all at once, aim at approximating the PF. In particular, this procedure reports less computational issues compared to other methods, such as NSGA-II. Among the different decomposition approaches, the Tchebycheff approach is the one adopted by the current algorithm. Following the ideas introduced in Miettinen (1998), the j-th subproblem of this kind can be expressed as follows:

$$\text{minimize } g^{te}(\mathbf{x} : \lambda^j, \mathbf{p}^*) = \max_{1 \leq i \leq n} \{\lambda_i^j : f_i(\mathbf{x}) - p_i^*\}$$

$$\text{subject to } \mathbf{x} \in \Omega$$

where $\mathbf{p}^* = (p_1^*, \ldots, p_m^*)^T$ is the reference point defined as $p_i^* = \max\{f_i(\mathbf{x}) : \mathbf{x} \in \Omega\}$ for each $i = 1, \ldots, m$ and $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$ is named the j-th weight vector. As usual, Ω refers to the feasible region, i.e., the set of values satisfying the constraints of the problem. In this context, given a multi-objective optimization problem and a non-dominated solution $\mathbf{x}^*$, there exists an appropriate λ such that $\mathbf{x}^*$ is also an optimal solution of $g^{te}(\mathbf{x} : \lambda, \mathbf{p}^*)$. Likewise, each optimal solution of this Tchebycheff scalarization subproblem provides a point in the optimal PF of the original problem. Hence, varying the weight vector allows for obtaining different non-dominated solutions. In particular, the continuity of g^{te} at λ ensures the closeness of the optimal solutions of two subproblems whose weight vectors are close. Therefore, given the j-th subproblem, information from other subproblems whose weight vectors are close to λ^j becomes relevant. At this point, the concept of the neighborhood of the j-th subproblem arises, defined as the set of subproblems whose weight vectors are closest to λ^j in terms of Euclidean distance. Clearly, the number of weight vectors in the neighborhood must be fixed in advance.

As for the main loop of MOEA/D, a population of N solutions $\mathbf{x}^1, \ldots, \mathbf{x}^N$ is considered, where each of the individuals corresponds to the best solution found so far by the scalar subproblems obtained by decomposition. Moreover, an external population or archive is maintained to store non-dominated solutions. In the first place, Euclidean distances between pairs of weight vectors are computed, and so neighborhoods are established. Next, for each individual $\mathbf{x}^j$, we consider the neighborhood of the j-th subproblem and randomly select two other solutions within it. The chosen parents undergo genetic operators to obtain an offspring $\mathbf{y}$. At this stage, a problem-specific improvement procedure is applied resulting in a new individual $\mathbf{y}'$, which is compared to the current solution and replaces it in case of obtaining an improvement in the value of the objective function of the j-th subproblem. As an improvement procedure, we have employed a basic local search based on the double-list mutation described in Section 3.3.1. Lastly, neighborhoods are conveniently updated, as well as the external archive, regarding the dominance relation. As usual, further details on this algorithm can be found in Zhang and Li (2007).

3.4 Computational results

This section presents the computational study conducted to evaluate the performance of the metaheuristics proposed for the bi-objective RCPSP_TDRC. It is organized as follows. First, we introduce the benchmark instances and describe the experimental setup. Then, we explain the evaluation criteria used to assess the quality of the PF approximations. Lastly, we report and analyze the results obtained in two stages: a calibration phase to tune each metaheuristic, followed by a comparison phase in which the most promising configurations are benchmarked.

3.4.1 Experimental setup

The instances used to evaluate the performance of the proposed methodologies are based on the benchmark datasets available in the PSPLIB library. In the single-mode setting, this library includes

the J30, J60, J90, and J120 datasets, which contain project instances with 30, 60, 90, and 120 activities, respectively. These instances assume 4 renewable resources available in limited quantities at each time period, and no non-renewable resources are considered. For detailed information regarding the instance generation procedure and the parameters involved in their construction, we refer to Section 2.4.

This chapter focuses on the J60 and J120 datasets, which correspond to medium-sized and large-scale projects, respectively. The J60 set comprises 480 instances, while the J120 set consists of 600 instances. Recall that instances in the J60 dataset are labeled as J60Y_Z, where Y ranges from 1 to 48 and Z from 1 to 10. Similarly, J120 instances are denoted as J120Y_Z, where Y ranges from 1 to 60 and Z from 1 to 10. Here, Y identifies the parameter configuration derived from a full factorial design, and Z corresponds to the instance replicate associated with each configuration. The planning horizon, T, is set to the sum of the processing times of all activities.

While the J60 and J120 datasets provide a well-established foundation for project scheduling experimentation, none of the original PSPLIB instances do include time-dependent resource costs, which are essential for modeling the problem considered in this study. Therefore, to adapt these benchmark instances to the bi-criteria RCPSP_TDRC, we incorporate time-dependent costs generated following the procedure proposed by Alcaraz et al. (2022). This approach defines a set of cost generation patterns based on trend, seasonality, and random fluctuations. Since these patterns are essential for constructing the instances used in this chapter, we briefly summarize their generation process below.

Resource costs are assumed to evolve over time according to a temporal model that incorporates a base level, a linear trend, a seasonal component, and a stochastic fluctuation. Specifically, the cost c_t of a resource at time interval $[t, t+1)$ is defined as:

$$c_t = \alpha + \beta t + \gamma_t + \omega_t, \quad t = 0, \ldots, T-1,$$

where α denotes a constant that sets the base level of the cost series, β defines a linear cost trend over time, γ_t represents the seasonal component associated with time period t, and ω_t is a random variable such that $\mathbb{E}[\omega_t] = 0$ and $\mathrm{Var}[\omega_t] = \sigma_\omega^2$.

The seasonal component γ_t accounts for recurring fluctuations in cost over time. These variations follow a periodic structure determined by the season length L, defined as the number of time units in one full cycle. In their study, $L = 12$ is used, corresponding to a 12-period horizon (e.g., weeks or months). The seasonality terms $\gamma_0, \gamma_1, \ldots, \gamma_{L-1}$ are assumed to repeat cyclically over the planning horizon. To ensure symmetry around a long-term average, the following constraint is imposed:

$$\sum_{\ell=0}^{L-1} \gamma_\ell = \Gamma \cdot L,$$

so that the average over any complete cycle equals a fixed value Γ. The value of Γ is sampled from a uniform distribution $U[20, 30]$. The specific seasonal sequence adopted, designed to satisfy the imposed symmetry and maintain a zero average slope, is given by the following pattern:

$$0,\ \Gamma,\ 2\Gamma,\ 3\Gamma,\ 2\Gamma,\ \Gamma,\ 0,\ -\Gamma,\ -2\Gamma,\ -3\Gamma,\ -2\Gamma,\ -\Gamma.$$

As for the trend component β, it is defined to produce a cost value near $\frac{\alpha}{2}$ at the end of the planning horizon (i.e., at time $T-1$), in the absence of seasonal and stochastic effects. A minimum slope of 0.1

(or maximum of -0.1) is enforced to ensure a noticeable trend. When $\frac{\alpha}{2T} > 0.1$, β is sampled from $U(0.1, \frac{\alpha}{2T})$ for a positive slope, or from $U(-\frac{\alpha}{2T}, -0.1)$ for a negative slope. Otherwise, β is fixed to 0.1 or -0.1. The noise term ω_t introduces random perturbations to the cost series and is generated independently at each time t from a normal distribution $N(0, 5)$.

Considering all these aspects, four patterns are constructed to represent different temporal cost behaviors. These patterns combine the direction of the trend—positive or negative—with the presence or absence of seasonality. Since each PSPLIB instance includes four renewable resources, a specific pattern is assigned to each one: resource 1 is associated with a positive trend and no seasonality (pattern 1); resource 2 with a negative trend and no seasonality (pattern 2); resource 3 with a positive trend and seasonal variation (pattern 3); and resource 4 with a negative trend and seasonal variation (pattern 4). The GitHub repository https://github.com/jalcarazs/Metaheuristics-for-Bi-objective-RCPSP_TDRC contains all instances used in this chapter.

To evaluate the seven algorithms, we adopt the standard experimental procedure, which involves two sequential phases. First, a calibration phase is conducted to determine the best configuration for each metaheuristic. Then, using the configurations selected in the first phase, all algorithms are compared on an evaluation set of instances to assess whether one or more of them outperform the others. The benchmark consists of the previously presented J60 and J120 datasets, and the procedure is applied independently to both sets of instances. For the J60 dataset, we consider 2 instance replicates, J60Y_1 and J60Y_2, for each of the 48 parameter configurations, yielding a total of 96 instances for the calibration set. The remaining 384 instances, corresponding to 8 replicates per problem class, form the evaluation set. Similarly, for the J120 dataset, we use 2 instance replicates, J120Y_1 and J120Y_2, from each of the 60 possibilities, considering a total of 120 instances for the calibration set. Hence, 8 instance replicates of each problem class remain, yielding a total of 480 instances for the evaluation set. Since the exact PF for the considered instances is not available, a reference front must be constructed for each instance. In both phases, the reference front of an instance is built by aggregating all the non-dominated solutions obtained across all runs performed on that instance.

3.4.2 Evaluation criteria

Before discussing the calibration and comparison phases in detail, we must first establish how the results obtained by the seven metaheuristics will be evaluated. As introduced in Chapter 2, several performance indicators can be used to assess the quality of PF approximations. In this chapter, we consider five metrics: the μ-indicator (μ), spread (Δ), additive ϵ-indicator (I_{ϵ^+}), hypervolume (HV), and modified inverted generational distance (IGD^+). Each performance indicator evaluates different aspects of solution quality, such as convergence, spread, and diversity, allowing us to identify when one approximation set outperforms another (see Section 2.1.1). Following Alcaraz et al. (2022), the objective values of each function are normalized before calculating any metric.

On the other hand, since the algorithms under study are stochastic, their results may vary across runs due to the use of random components. To draw reliable conclusions, performance assessment must be accompanied by appropriate statistical tests, which allow for comparing the obtained results with a

certain level of confidence. An analysis of variance (ANOVA) is applied to detect significant differences across algorithms (comparison phase) or configurations (calibration phase). When differences are found, Tukey's Honest Significant Difference (HSD) test is used for pairwise comparisons. A confidence level of 99% (i.e., $p < 0.01$) is adopted in both tests.

An important consideration is that the five indicators used are not always aligned, as they capture different properties of the approximation sets. For instance, an algorithm might excel in convergence but perform poorly in terms of spread, while another shows the opposite behavior. In such cases, it may be unclear which algorithm performs better overall. To address this, we adopt a discarded criterion, which establishes a preference order among the performance indicators. Suppose we are in the comparison phase. We begin by applying Tukey's HSD test to identify the best-performing algorithm with respect to the first indicator in the list (e.g., HV). If a single algorithm is statistically superior, it is selected. If several algorithms are tied, we move to the next indicator, applying the same process only among the previously selected ones. This continues until a unique best algorithm is identified or all performance indicators in the list have been considered.

Following the discarded criterion, the preference order adopted is: HV, Δ, I_{ϵ^+}, IGD^+, and μ. We prioritize the first three metrics as they are widely used in the literature (see, e.g., Deb et al., 2002; Durillo et al., 2010; Yepes-Borrero et al., 2021). HV measures both convergence to the PF and diversity of the obtained solutions. Similarly, the additive ϵ-indicator also measures convergence to the optimal front. In contrast to the latter two metrics, and providing a wider range of information, Δ aims at measuring the extents of the spread achieved among the obtained solutions. As will be seen in the comparison phase (Section 3.4.4), the three previous metrics are insufficient to determine the best metaheuristic, so we also consider the IGD^+ measure, which is also a widely used convergence and distribution performance indicator. On the other hand, the μ-indicator combines two well-known performance indicators, providing a distribution and spread measure that summarizes the information from both. Zitzler et al. (2003) prove that the number of criteria that determine what a good approximation set is, is infinite. Thus, an optimal order cannot be found and we can only achieve a convenient one, summarizing various aspects of the obtained approximation sets.

3.4.3 Calibration of the algorithms

The seven metaheuristics considered in this chapter, introduced in Section 3.3, are implemented in Java using the open-source jMetal framework (Durillo and Nebro, 2011; Nebro et al., 2015). This general framework has been adapted to incorporate the specific design features of our algorithms, such as the AL with scheduling objectives, as well as the double-list crossover and mutation operators. Moreover, we deal with algorithms that depend on different sets of parameters. To compare them, an extensive computational and statistical study is carried out. Nonetheless, a first stage, where those methodologies are calibrated, must be performed. As mentioned above, this first stage is referred to as the calibration phase where each algorithm is calibrated separately, considering a full factorial design of experiments (DOE), with all combinations of factors and levels. In particular, we have chosen a set of parameter values in order to allow a fair comparison between all the metaheuristics.

All the MOEAs considered share these three parameters: population size (N), crossover probability (p_c), and mutation probability (p_m). Note that the archive size is not included in the DOE since it has been set to match the population size. After some preliminary experiments, we select three values for N: 50, 100 and 200 individuals. On the other hand, two levels are defined for the crossover probability (0.7 and 0.9) and for the mutation probability ($\frac{1}{n}$ and $\frac{2}{n}$), where n denotes the number of activities in the project. Regarding the last parameter, mutation probabilities that depend on n are commonly used in the literature (see, e.g., Durillo et al., 2010, Deb, 2011, Salto and Alba, 2019). In PESA-II, we consider two possible number of bi-sections, S: 5 and 10. Likewise, in IBEA, we use three possible values for κ: $0.025, 0.05, 0.1$. Finally, the neighborhood selection probability, p_n, in MOEA/D is set at two levels: $0.7, 0.9$. An overview of the number of tuning parameter values involved in the calibration phase is included in Table 3.1, along with the number of possible configurations for each technique.

	Possible values						**Number of Configurations**
	N	p_c	p_m	S	κ	p_n	
NSGA-II	3	2	2	-	-	-	12
SPEA2	3	2	2	-	-	-	12
MOCell	3	2	2	-	-	-	12
PESA-II	3	2	2	2	-	-	24
IBEA	3	2	2	-	3	-	36
SMS-EMOA	3	2	2	-	-	-	12
MOEA/D	3	2	2	-	-	2	24

Table 3.1: Number of different parameter values and possible configurations.

Since all the metaheuristics employ the same solution encoding and also the same operators, crossover and mutation, to manage this type of encoding (see Section 3.3.1), differences in behavior of the algorithms cannot be due to the use of a better or worse encoding or the use of specific procedures that could be more efficient. Differences between two metaheuristics should indicate that the underlying scheme of one outperforms the other in solving the problem under consideration.

Once the parametrization details of the metaheuristics have been introduced, we focus on the calibration procedure. To illustrate the process, we consider the SPEA2 algorithm and describe the DOE conducted for this particular technique. We do not repeat the process for the remaining metaheuristics, as the steps followed and the statistical models applied are analogous. According to Table 3.1, the factorial design for SPEA2 yields a total of $3 \times 2 \times 2 = 12$ configurations. Each configuration is tested in both calibration sets—J60 and J120—performing 3 independent runs by instance. As described in Section 3.4.1, the calibration phase is conducted separately for each dataset, using 96 instances from the J60 set and 120 instances from the J120 set. Hence, the calibration of SPEA2 results in $12 \times 96 \times 3 = 3,456$ PF approximations for J60, and $12 \times 120 \times 3 = 4,320$ for J120, yielding a total of $7,776$ runs. Regarding the stopping criterion, each run is executed for a maximum of 2 minutes. Consequently, the total CPU time dedicated to calibrating SPEA2 amounts to $7,776 \times 2 = 15,552$ minutes, or approximately 10.8 days. This process is repeated independently for each metaheuristic,

resulting in one best configuration per algorithm and dataset.

As stated in Section 3.4.2, to statistically compare the performance of the different configurations tested during the calibration phase, a two-way ANOVA is applied for each performance indicator. As an example, let us consider I_{ϵ^+}. In this case, the two factors under analysis are the set of problem instances (factor A) and the set of parameter configurations (factor B). For the J120 dataset, we consider 120 problem instances and 12 configurations, with 3 independent runs per combination, yielding 3 observations for each of the 120×12 combinations. The corresponding statistical model is:

$$(I_{\epsilon^+})_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk}, \tag{3.8}$$

where $i \in \{1, \ldots, 120\}$, $j \in \{1, \ldots, 12\}$, and $k \in \{1, \ldots, 3\}$, $\epsilon_{ijk} \sim N(0, \sigma^2)$ and are independent. Parameter μ represents the overall mean, parameters α_i represent the main effect of factor A, parameters β_j represent the main effect of factor B, and parameters γ_{ij} represent the interaction between the two factors. All standard ANOVA assumptions are checked, with no problems in the residuals found. A similar model is used for the J60 dataset, adjusting the number of problem instances accordingly. Note that the number of configurations varies depending on the metaheuristic. Once significant differences are detected, Tukey's HSD test is used to identify the best-performing configuration(s).

We now present the computational results derived from this study. First, regarding the analysis of variance, we find that both main factors and their interaction are statistically significant for all metaheuristics and response variables, i.e., performance indicators. These results indicate that the observed differences are unlikely to have occurred by chance, with a confidence level of 99%. Subsequently, Tukey's HSD test is applied to identify the best-performing configurations. To select a single configuration per metaheuristic and dataset, the test is applied sequentially following the discarded criterion described in Section 3.4.2: starting with the highest-priority indicator, ties are resolved by progressively considering the next indicators in the list until a unique best configuration is identified or all metrics have been examined.

	J60						**J120**					
Algorithm	N	p_c	p_m	S	κ	p_n	N	p_c	p_m	S	κ	p_n
NSGA-II	100	0.9	$1.0/n$	-	-	-	200	0.9	$2.0/n$	-	-	-
SPEA2	100	0.7	$1.0/n$	-	-	-	100	0.7	$2.0/n$	-	-	-
MOCell	50	0.9	$1.0/n$	-	-	-	50	0.9	$1.0/n$	-	-	-
PESA-II	200	0.9	$2.0/n$	5	-	-	200	0.7	$2.0/n$	5	-	-
IBEA	200	0.7	$2.0/n$	-	0.05	-	200	0.7	$2.0/n$	-	0.05	-
SMS-EMOA	50	0.9	$2.0/n$	-	-	-	50	0.9	$2.0/n$	-	-	-
MOEA/D	200	0.7	$2.0/n$	-	-	0.7	200	0.7	$1.0/n$	-	-	0.7

Table 3.2: Best configurations obtained for each metaheuristic on the J60 and J120 calibration sets (n = number of activities).

Table 3.2 summarizes the best configurations selected during the calibration phase. Each row corresponds to one metaheuristic, and each column indicates the selected value for a tuning parameter.

Parameters not used by all metaheuristics—for example, the number of bi-sections—are marked with a "–" symbol. As shown in the table, the calibration procedure has been conducted independently for the J60 and J120 datasets, resulting in two best-performing configurations per metaheuristic. These configurations will be used in the subsequent comparison phase. A detailed example of how ANOVA and Tukey's HSD test are used to identify the best-performing configuration for a specific algorithm and dataset is provided in Appendix A.1.

To conclude, following the same procedure described earlier, the total CPU time required to complete the calibration phase across all seven metaheuristics can be estimated. For the J60 dataset, this amounts to approximately 53 days. Similarly, the calibration process for the J120 dataset requires around 66 days. Therefore, completing the full calibration phase across both datasets demands a total of nearly 119 days of CPU time.

3.4.4 Computational comparison among the best configurations

This section presents the numerical results used to evaluate the performance of the seven metaheuristics considered in this chapter. The objective is to compare the best configurations obtained during the calibration phase. The comparison is conducted independently on two evaluation datasets derived from the J60 and J120 sets (see Section 3.4.1), allowing us to assess the robustness and effectiveness of each algorithm across instances of different sizes. For both datasets, performance is evaluated using the five indicators listed in Section 3.4.2, which measure different properties of the obtained PF approximations: HV, Δ, I_{ϵ^+}, IGD^+, and μ. Additionally, the statistical methodology follows the procedure introduced earlier, combining ANOVA and Tukey's HSD test to identify significant differences among the algorithms. The results are presented separately for each dataset in the following subsections.

Results for J60 dataset

We begin with the experimental results obtained by evaluating the seven metaheuristics on the J60 evaluation set, which includes 384 medium-sized project instances. As in the calibration phase, each metaheuristic is tested with a CPU time limit of 2 minutes per run, performing three independent runs for each instance. Consequently, a total of 8,064 approximated PF are obtained, corresponding to 1,152 fronts per metaheuristic. The total computational time required to complete all these executions amounts to approximately 16,128 minutes, which is equivalent to about 11 days of CPU time.

NSGA-II	SPEA2	MOCell	PESA-II	IBEA	SMS-EMOA	MOEA/D
38.308	39.347	34.480	37.553	7.945	35.555	9.464

Table 3.3: Average number of non-dominated solutions for each metaheuristic.

Table 3.3 summarizes the average number of non-dominated solutions obtained by each metaheuristic. From these results, we observe notable differences among algorithms. Specifically, the IBEA and MOEA/D algorithms identify substantially fewer non-dominated solutions, demonstrating limitations

in exploring and capturing a diverse set of efficient solutions. Conversely, SPEA2 achieves the highest number of non-dominated solutions, closely followed by NSGA-II, indicating a better exploration capability in comparison with other techniques.

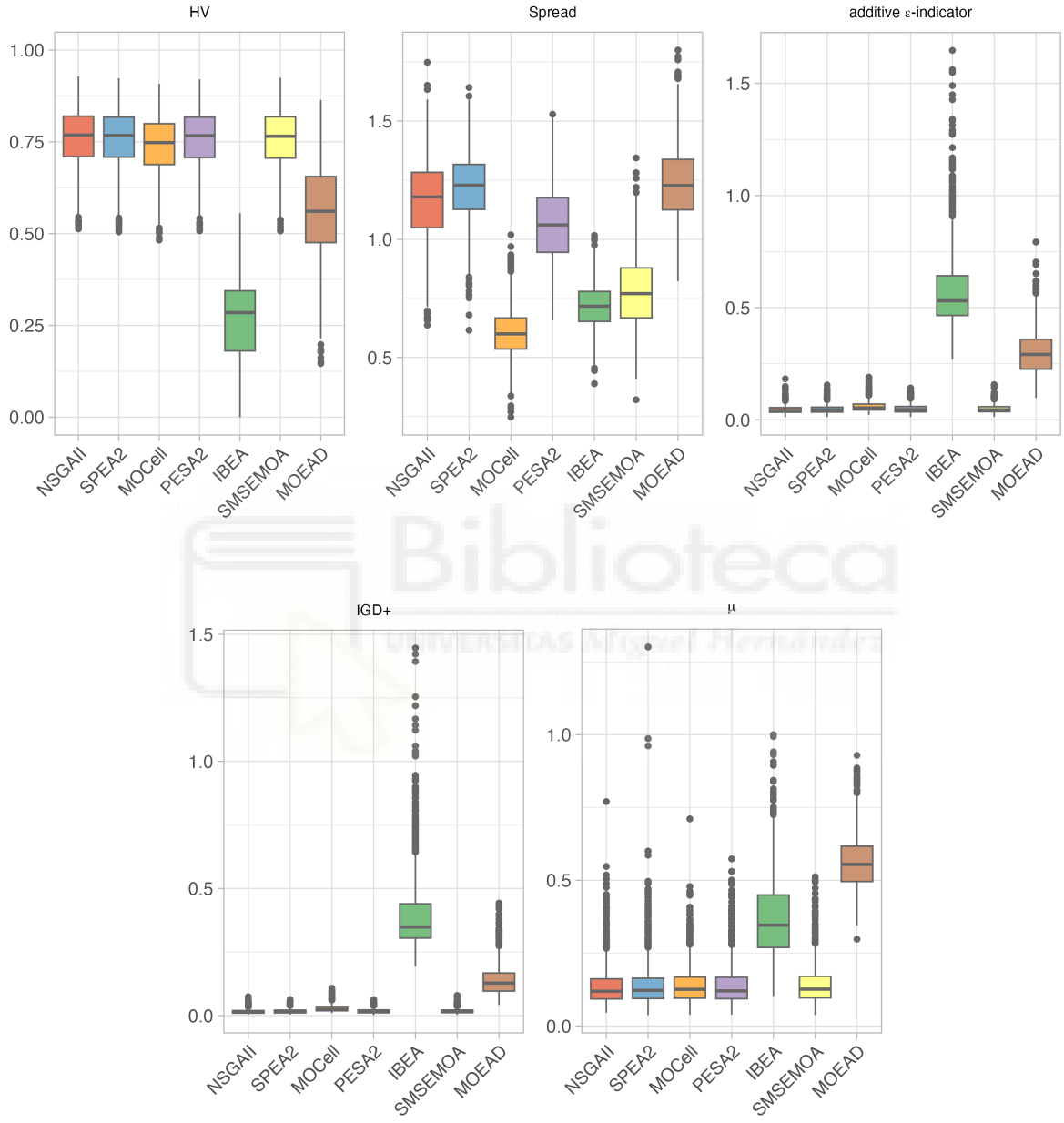


Figure 3.5: Boxplots of the performance indicators corresponding to the best configuration of each metaheuristic.

To continue the analysis, we employ boxplots to illustrate the distributions of the five performance indicators considered in this study, obtained by each algorithm when solving the 384-instance evaluation set. The results are shown in Figure 3.5. We start with the performance indicators that

measure convergence and distribution—HV, I_{ϵ^+}, and IGD^+—which correspond to the first, third, and fourth boxplots, respectively. All metaheuristics, except IBEA and MOEA/D, achieve comparably high performance in terms of HV, with median values around 0.75. In contrast, IBEA shows notably lower performance, with a median HV below 0.3, while MOEA/D yields intermediate but clearly inferior results compared to the top-performing algorithms. It is important to note that higher HV values indicate better quality, whereas for the remaining indicators, lower values are preferred (see Section 2.1.1). Regarding I_{ϵ^+} and IGD^+, all algorithms outperform IBEA and MOEA/D, exhibiting similar performance among themselves, with median values close to 0. In particular, IBEA reports the most unsatisfactory results, with median values around 0.5 for both indicators.

Finally, we examine Δ and μ—the second and last boxplots—which focus on the spread and distribution of the approximation sets. In this case, we observe slightly different patterns. IBEA demonstrates moderate performance, with values relatively close to those of the other algorithms. MOEA/D, however, clearly underperforms, with higher median values that reflect lower diversity in the PF approximations. Interestingly, in the case of Δ, the best results are obtained by MOCell, with a median value close to 0.5, followed by IBEA and SMS-EMOA. A notable change in behavior is observed for NSGA-II, SPEA2, and PESA-II, which now rank lowest in terms of performance, reporting median values greater than 1.

Overall, the boxplots reveal a consistent trend: IBEA and MOEA/D tend to underperform across most performance indicators, highlighting their limitations in terms of convergence to the PF, solution diversity, and spread. In contrast, the remaining metaheuristics—NSGA-II, SPEA2, MOCell, PESA-II, and SMS-EMOA—exhibit consistently high performance.

We now proceed with the statistical tests. For the two-way ANOVA, we define five statistical models—one for each performance indicator. As in the calibration phase, we take I_{ϵ^+} as the response variable to illustrate the structure of the model. The two independent variables are: the evaluation set of instances (factor A), and the seven metaheuristics with their best configurations resulting from the calibration phase (factor B). For the J60 dataset, the evaluation set comprises 384 instances. For each of the 384×7 combinations, we perform 3 independent runs, yielding 3 observations for each combination. The corresponding statistical model is the following:

$$(I_{\epsilon^+})_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk}, \tag{3.9}$$

where $i \in \{1, \ldots, 384\}$, $j \in \{1, \ldots, 7\}$, and $k \in \{1, \ldots, 3\}$, with $\epsilon_{ijk} \sim N(0, \sigma^2)$ and assumed to be independent. The meaning of each model parameter is described in Model (3.8), and is omitted here for brevity. Regarding the results, the main effects and their interaction are statistically significant for all performance indicators considered in the study. This confirms that the metric values are influenced not only by the metaheuristic employed but also by the specific instance being solved, as well as by the interaction between these two factors. The detailed results of the ANOVA models fitted for each performance indicator are provided in Appendix A.2, under the subsection *J60 dataset*.

Since the effect of the algorithm (factor B) is statistically significant across all models, we proceed to identify which of the seven metaheuristics performs best, which is the final objective of the chapter. As in the previous section, Tukey's HSD test is applied to perform a single-step multiple comparison

	HV		Δ		$I_{\epsilon+}$		IGD^+		μ	
	Mean	Group	Mean	Group	Mean	Group	Mean	Group	Mean	Group
NSGA-II	**0.761**	a	***1.167***	e	***0.046***	a	***0.016***	a	***0.140***	a
SPEA2	**0.759**	ab	*1.218*	f	0.048	a	0.017	a	0.144	a
PESA-II	0.758	b	1.066	d	0.050	a	0.018	a	0.141	a
SMS-EMOA	0.758	b	0.778	c	0.049	a	0.018	a	0.144	a
MOCell	0.739	c	0.606	a	0.060	b	0.029	b	0.144	a
MOEA/D	0.560	d	1.241	g	0.301	c	0.139	c	0.565	c
IBEA	0.255	e	0.718	b	0.584	d	0.400	d	0.374	b
BEST	NSGA-II SPEA2		NSGA-II		NSGA-II		NSGA-II		NSGA-II	

Table 3.4: Computational results for the best configurations of the algorithms. Mean values (3 decimals) and Tukey groupings for each performance indicator.

procedure aimed at determining the best algorithm among the selected configurations. Table 3.4 presents the average values of the five performance indicators, ordered according to the discarded criterion (see Section 3.4.2). Alongside each metric, the corresponding grouping derived from Tukey's HSD test is shown. Algorithms belonging to different groups report significantly different mean values. In the table, the best-performing algorithms are marked in bold, while italics are used to indicate those that were previously identified as best according to the discarded criterion. For clarity, the last row of Table 3.4 summarizes the best results identified so far.

Following the proposed ordering of the performance indicators, we begin with the hypervolume. According to Tukey's HSD test, NSGA-II and SPEA2 exhibit statistically superior performance for this metric, both being placed in the top-performing groups. It is important to note that when two treatment groups share the same letter, it indicates no statistically significant difference between their mean values. Therefore, we conclude that there is no significant difference between NSGA-II and SPEA2, nor between SPEA2 and PESA-II. As no unique best algorithm can be identified based solely on HV, we move to the second indicator in the list: Δ. In this case, neither NSGA-II nor SPEA2 belong to the top-performing group. Nonetheless, NSGA-II achieves significantly better results than SPEA2, which falls into a clearly inferior group. According to the discarded criterion, this outcome allows us to conclude that NSGA-II is the best-performing algorithm for the J60 dataset.

Although the remaining performance indicators are not required to establish a conclusion under the criterion adopted, they offer further insights into the relative behavior of the metaheuristics. NSGA-II consistently ranks at the top across all performance indicators. In particular, it belongs to the first Tukey group in all tests except for Δ, where it ranks fifth. Similarly, SPEA2 belongs to the first treatment group in four out of the five tests applied, showing weaker performance only in the spread measure. Competitive results are also observed for PESA-II, SMS-EMOA, and MOCell, as reflected in the table. In contrast, MOEA/D and especially IBEA consistently report poor results, appearing in the lowest-performing groups for most indicators. For a more detailed view of the statistical comparisons,

Appendix A.2, subsection *J60 dataset*, includes the mean plots for the five performance indicators with Tukey's HSD 99% confidence intervals. These visualizations further support the ranking of the metaheuristics discussed above. It is important to emphasize that in the subsequent analysis of the J120 dataset, differences among algorithms become more subtle, thus requiring the consideration of all five indicators to draw reliable conclusions.

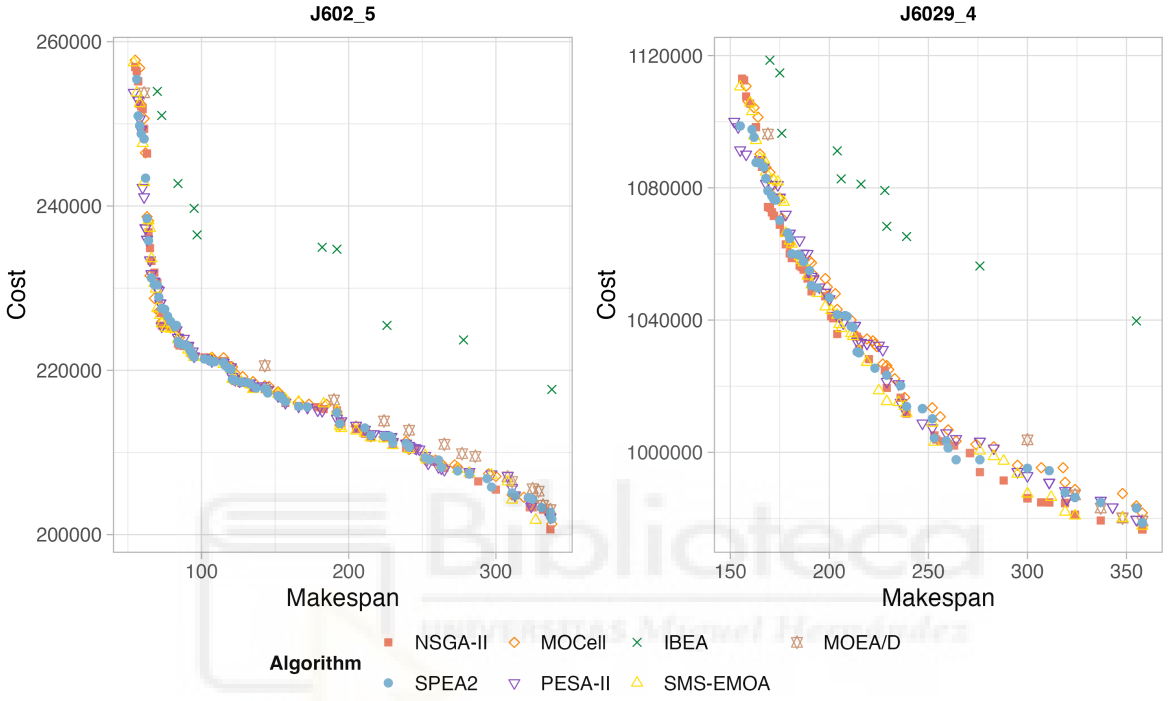


Figure 3.6: PF approximations obtained by the seven metaheuristics for two selected instances from the J60 evaluation set.

To conclude the analysis, Figure 3.6 illustrates two examples of the PF approximations obtained by the seven metaheuristics for specific instances from the J60 evaluation set. In particular, the results correspond to the fifth replicate of instance 2 (J602_5) and the fourth replicate of instance 29 (J6029_4). These visualizations reinforce the previously discussed findings: NSGA-II, SPEA2, SMS-EMOA, MOCell, and PESA-II effectively approximate the PF, providing solutions that adequately cover the objective space, and thus exhibit good performance in terms of convergence, solution diversity, and spread. In contrast, IBEA and MOEA/D present clear shortcomings in both diversity and convergence, further supporting the numerical results presented earlier.

Results for J120 dataset

To conclude the experimental analysis, we compare the best configurations obtained during the calibration phase on the J120 evaluation set, which consists of 480 instances—the largest projects from the PSPLIB library. Recall that these configurations were calibrated specifically on the J120 calibration set, independently from the J60 calibration process. As in the previous case, the same stopping

criterion is adopted for every algorithm, with a CPU time limit of 2 minutes per run. Likewise, 3 independent runs are performed per instance, attaining 1,440 runs per algorithm. A total of 10,080 fronts are obtained as a result. Thus, each of the seven metaheuristics needs 2,880 minutes to finish all the executions, for a total of 20,160 minutes, or 14 days of CPU time to complete the comparison phase.

NSGA-II	SPEA2	MOCell	PESA-II	IBEA	SMS-EMOA	MOEA/D
51.752	65.147	45.441	50.083	8.990	45.201	9.963

Table 3.5: Average number of non-dominated solutions of each metaheuristic.

Table 3.5 contains the average number of non-dominated solutions obtained by each of the seven algorithms. As observed for the J60 dataset, IBEA and MOEA/D achieve the lowest number of optimal solutions, showing the challenge these metaheuristics face in discovering new non-dominated solutions in the search space. To provide further insights for this large-sized instances, Table 3.6 shows the average, median and interquartile range of the number of solutions evaluated by the different algorithms within the time limit of 2 minutes. As one can notice, PESA-II obtains the greater value, resulting in an average of almost 400,000 solutions computed when solving the evaluation set instances. Conversely, the MOEA/D algorithm evaluates near 90,000 in the same time. Nonetheless, performing a larger number of evaluations does not necessarily mean getting better results, as we will observe later.

	NSGA-II	SPEA2	MOCell	PESA-II	IBEA	SMS-EMOA	MOEA/D
Mean	184205	153720	170487	383550	182856	166986	90139
Median	178400	148850	163328	368000	182800	162306	85100
IQR	55450	47225	55900	121650	46050	47745	31850

Table 3.6: Solutions evaluated by each metaheuristic in two minutes of computation time.

The boxplots in Figure 3.7 show, for each performance indicator, the observed values attached by every considered metaheuristic when solving the 480-instance evaluation set. The results reveal distinct patterns consistent with those reported for the J60 evaluation dataset. Focusing on one of the graphs, e.g. the boxplot with HV as the response variable, one can observe again the poor behavior of IBEA and MOEA/D compared to the five remaining algorithms. In particular, IBEA shows a median close to 0.3, which is almost half the value achieved by the other metaheuristics, excluding MOEA/D. As for the latter, its median exceeds 0.5, making it closer to the value of 0.75 reached by the five best-performing algorithms. Nevertheless, some outliers can be observed for MOEA/D, with the minimum value being similar to that of IBEA. Regarding Δ and the μ-indicator—second and last boxplots—a slightly different behavior can be observed, with MOEA/D reporting the worst results. In detail, IBEA behaves decently when considering the spread measure. Still, this does not change the fact that IBEA reports unsatisfactory results in terms of convergence and distribution.

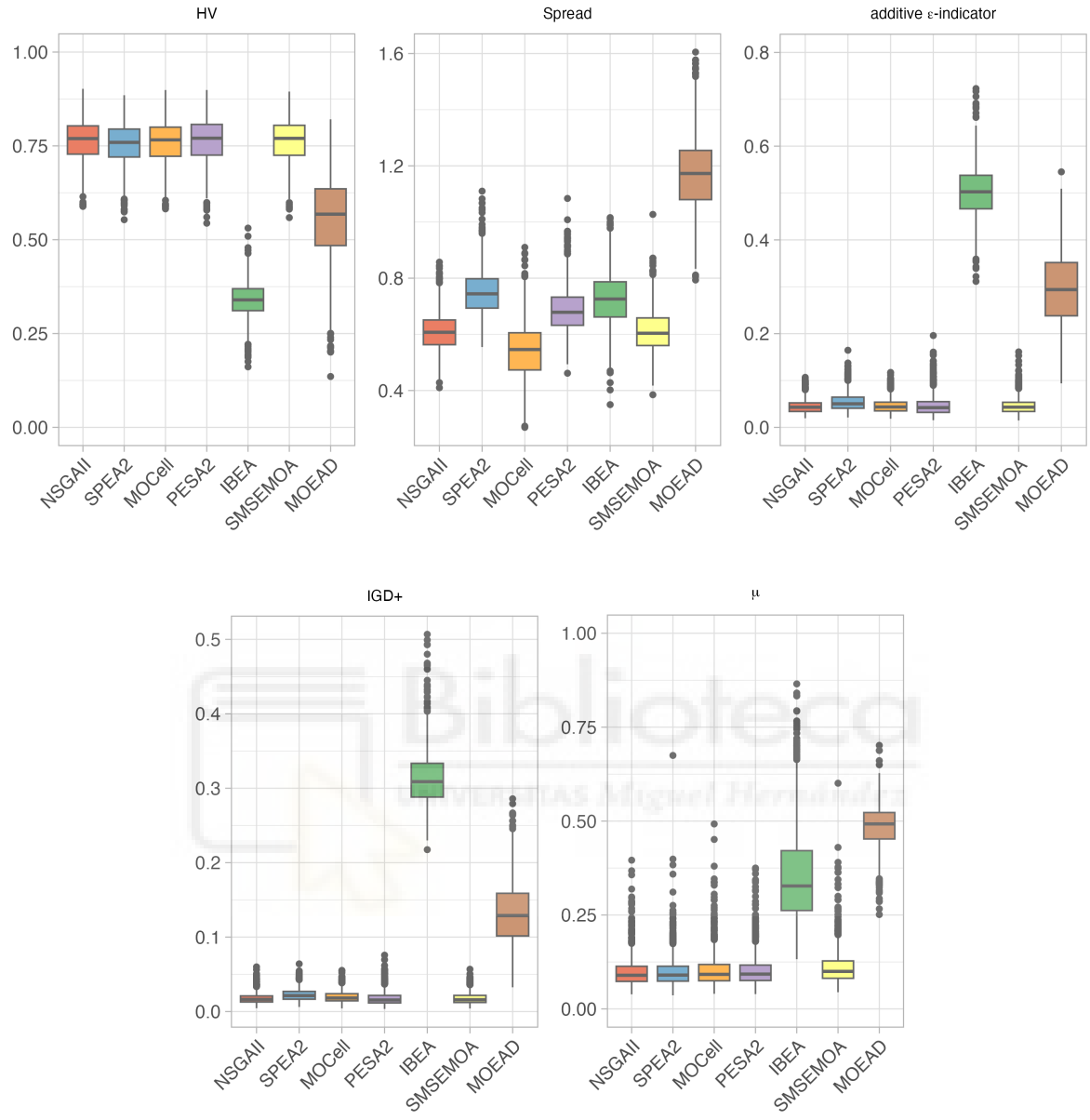


Figure 3.7: Boxplots of the performance indicators corresponding to the best configuration of each metaheuristic.

To summarize, the graphs show that five of the metaheuristics—NSGA-II, SPEA2, MOCell, PESA-II, and SMS-EMOA—achieve high-quality results in terms of convergence to the PF, solution diversity, and the extent of spread within the set of non-dominated solutions. At the same time, we observe that IBEA and MOEA/D exhibit the poorest performance. These findings are consistent with those reported in the previous analysis and reinforce the overall tendencies already established.

Regarding the two-way ANOVA, we again consider five statistical models, one for each performance indicator. As in the previous analysis, we illustrate the structure of the model for $I_{\epsilon+}$. In particular,

factor A is represented by the evaluation set—now composed of 480 instances—and factor B corresponds to the seven best metaheuristic configurations derived from the calibration phase. For each of the resulting 480 × 7 combinations, three independent runs are selected, leading to the same model structure already defined in Model (3.9). The model parameters and assumptions are identical to those previously described and are therefore omitted for brevity. As with the J60 dataset, the results reveal that the main effects and their interaction are statistically significant for all performance indicators. This confirms that the metric values are influenced not only by the algorithm used but also by the specific instance being solved and the interaction between both factors. The detailed ANOVA results for the J120 dataset are provided in Appendix A.2, under the subsection *J120 dataset*.

	HV		Δ		$I_{\epsilon+}$		IGD^+		μ	
	Mean	Group	Mean	Group	Mean	Group	Mean	Group	Mean	Group
NSGA-II	**0.765**	a	***0.610***	b	***0.045***	a	***0.018***	a	***0.099***	a
SPEA2	0.756	c	0.749	e	0.055	b	0.023	c	0.099	a
MOCell	0.761	b	0.540	a	0.046	a	0.020	b	0.102	a
PESA-II	**0.765**	a	*0.686*	c	0.047	a	0.018	a	0.103	a
IBEA	0.339	e	0.725	d	0.502	d	0.313	e	0.353	c
SMS-EMOA	**0.765**	a	***0.611***	b	***0.046***	a	***0.018***	a	*0.112*	b
MOEA/D	0.558	d	1.170	f	0.293	c	0.132	d	0.486	d
BEST	NSGA-II PESA-II SMS-EMOA		NSGA-II SMS-EMOA		NSGA-II SMS-EMOA		NSGA-II SMS-EMOA		NSGA-II	

Table 3.7: Computational results for the best configurations of the algorithms.

Given the results discussed so far, we can confirm that the algorithmic effect is statistically significant across all performance indicators. Subsequently, Tukey's HSD test is applied to identify the best-performing algorithm among the seven metaheuristics evaluated. Table 3.7 presents the post-hoc comparison results for the five performance indicators, ordered according to the discarded criterion (see Section 3.4.2). As in the previous table, the best-performing algorithms are marked in bold, and those previously selected according to the discarded criterion are shown in italics. The last row summarizes the best results identified so far.

We now proceed with the application of this criterion. In the first step, NSGA-II, PESA-II, and SMS-EMOA are selected, as they belong to the first treatment group according to the hypervolume indicator. The second metric, Δ is then used to discriminate among these algorithms. As a result, PESA-II is discarded, as it falls into a lower-performing group—the third treatment group—compared to the other two metaheuristics. The comparison continues with the ϵ-indicator and IGD^+, but a conclusive distinction still cannot be established, as both NSGA-II and SMS-EMOA remain in the top group. We then consider the final performance indicator in the sequencee, μ, for which NSGA-II achieves significantly better results than SMS-EMOA: it belongs to the first group, whereas the latter falls into the second. Therefore, the μ-indicator proves essential for establishing a final conclusion in

this experimental study, making it necessary to consult all metrics in the list.

In general, NSGA-II and PESA-II consistently perform well across all five indicators. Both algorithms belong to the first treatment group for every metric, except for spread, where NSGA-II is placed in the second group and PESA-II in the third. SMS-EMOA and MOCell also deliver competitive results, appearing in the second treatment group for only two of the five indicators and in the first group for the remaining ones. In contrast, IBEA and MOEA/D confirm the poor performance already discussed, consistently ranking among the lowest-performing groups. For further details, Appendix A.2, subsection *J120 dataset*, includes the mean plots with Tukey's HSD 99% confidence intervals for each performance indicator and algorithm, supporting the conclusions drawn from this analysis.

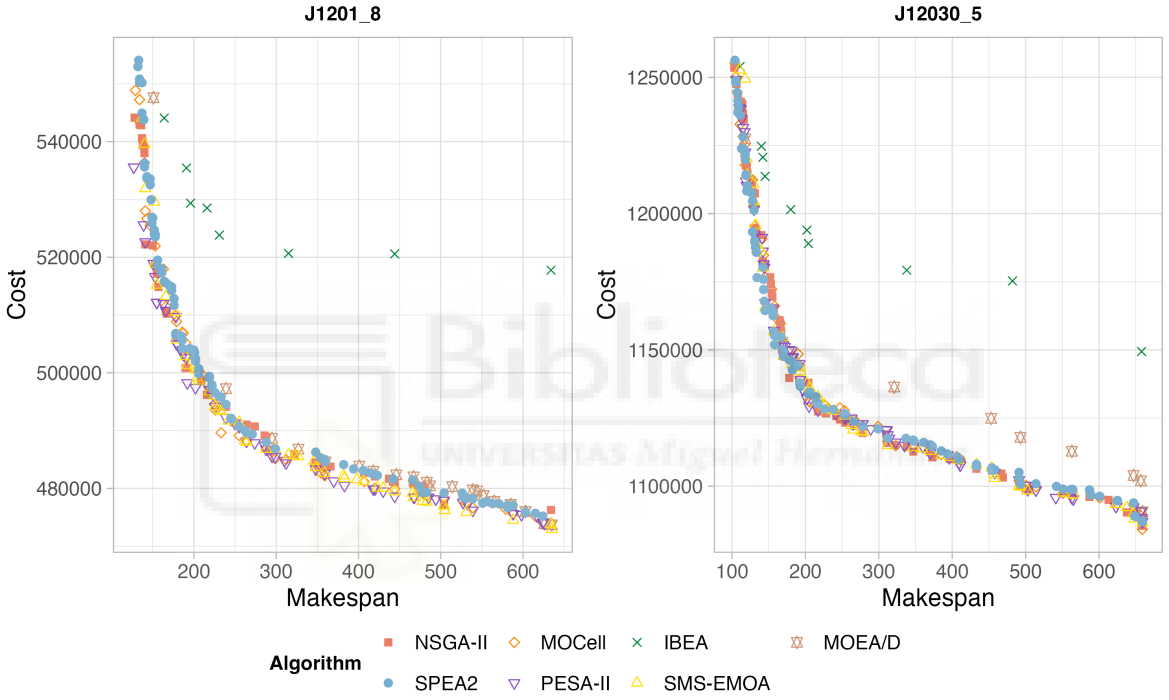


Figure 3.8: PF approximations obtained by the seven metaheuristics for two selected instances from the J120 evaluation set.

Finally, Figure 3.8 presents two illustrative examples of the PF approximations produced by the seven metaheuristics for selected instances from the J120 evaluation set. Specifically, we display the results for the eighth replicate of instance 1 (J1201_8) and the fifth replicate of instance 30 (J12030_5). In the first instance (left), MOEA/D achieves relatively better convergence compared to its usual performance, while in the second (right), its PF approximation deviates significantly from the rest. It also shows the poorest performance in terms of spread. In both cases, MOEA/D and IBEA produce a notably reduced number of non-dominated solutions in the final population. Moreover, IBEA displays clear deficiencies in convergence and distribution, highlighting important limitations in solution quality. In contrast, the remaining five metaheuristics provide more robust approximations in both instances, yielding well-distributed and convergent sets of non-dominated solutions.

Overall, these graphical representations are consistent with the trends observed throughout the numerical analysis, further confirming the superior performance of the five top-ranked metaheuristics—NSGA-II, SPEA2, MOCell, PESA-II, and SMS-EMOA—in terms of convergence, diversity, and spread. Notably, NSGA-II stands out as the most effective algorithm across both evaluation sets, consistently achieving significantly better results according to the discarded criterion. The remaining four metaheuristics also demonstrate competitive performance, frequently ranking among the top groups in the statistical tests. These consistent findings across datasets of different sizes reinforce the robustness of the conclusions drawn and highlight the reliability of the proposed comparison framework.

3.5 Final remarks and future work

This chapter focuses on the bi-criteria RCPSP_TDRC, a recent extension of the classical RCPSP, in which the objectives are to minimize the project makespan and the total cost of resource usage. To tackle this problem, seven state-of-the-art MOEAs are implemented and compared under a rigorous benchmarking framework. The computational experiments are conducted on medium- and large-scale instances adapted from the PSPLIB library, involving projects with 60 and 120 activities. The experimental study is structured into two well-differentiated stages. The first stage is a calibration phase, where each algorithm is properly tuned using a specific calibration set of instances. The second stage involves the comparison of the best configurations of each metaheuristic on a separate evaluation set. Each stage is analyzed independently for the two benchmark sets, thus providing a robust and unbiased assessment of the results. Importantly, all metaheuristics share the same encoding and genetic operators, ensuring that performance differences are attributable solely to the core algorithmic design—that is, to their underlying search strategies. These considerations are essential for a fair comparison of the algorithmic performance on this variant of the RCPSP.

The experimental study shows that the majority of the algorithms provide sharp approximations of the reference PF. Moreover, high-quality approximation sets are found within a CPU time limit of 2 minutes per execution, which is a considerably small amount of time. Results are consistent across five established performance indicators, and both benchmark sets. This research demonstrates the strong performance of several MOEAs, with NSGA-II emerging as one of the most robust and effective approaches to deal with the bi-criteria RCPSP_TDRC, delivering superior results in terms of convergence and distribution. In contrast, other paradigms such as MOEA/D and IBEA consistently yield inferior results. Overall, the study highlights the potential of evolutionary computation for solving rich, bi-objective variants of classical scheduling problems, providing decision-makers with a diverse set of high-quality algorithms that can generate alternative solutions and enhance informed decision-making.

Looking ahead, a natural extension of this work is to incorporate uncertainty into the resource cost profiles, aligning the model more closely with real-world planning environments. Robust or stochastic formulations could be developed to address temporal variability and risk in resource pricing. Furthermore, given the promising performance of NSGA-II, future research may explore enhanced variants—such as r-NSGA-II (Said et al., 2010), U-NSGA-III (Seada and Deb, 2015), or CHIM-NSGA (Fi-

latovas et al., 2020)—to assess whether further improvements can be achieved in terms of convergence and solution diversity. These directions open promising opportunities to refine the optimization process and expand its applicability to increasingly realistic and dynamic scheduling scenarios.

Chapter 4

Multi-mode resource-constrained project scheduling problem with time-dependent resource costs and capacities: a bi-objective approach

The increasing complexity of modern project environments demands scheduling models that account for real-world scenarios such as fluctuating resource availability and varying resource costs. This chapter advances both the modeling and solution approaches for project scheduling under such conditions, introducing a novel problem variant that integrates time-dependent resource costs and capacities within a multi-mode, bi-objective framework.

4.1 Motivation and structure

Despite significant advancements in multi-mode project scheduling and multi-objective optimization, the integration of multiple execution modes into the bi-objective resource-constrained project scheduling problem with time-dependent resource costs (RCPSP_TDRC) remains an unexplored area in the literature. Addressing this gap, this chapter presents an extension of the multi-mode resource-constrained project scheduling problem (MRCPSP), which incorporates time-dependent resource costs as well as a further generalization of resource constraints: time-varying resource capacities (Hartmann, 2015). Typically, resource availability is assumed to remain constant over time. However, this assumption can be overly restrictive for practical applications. For instance, factors such as staff vacations or planned machine maintenance can lead to fluctuations in resource capacities. To better capture a broader range of real-world scenarios, resource availability can vary across time periods within the planning horizon.

To tackle this novel problem, we build upon the classical MRCPSP formulation proposed by Talbot (1982), which accounts for time-varying resource capacities. Afterward, we attempt to solve the problem using the AUGMECON method (Mavrotas, 2009), which is an improved version of the ϵ-

constrained method. However, due to the inherent complexity of the problem, the exact algorithm fails to solve instances with more than 30 activities, leaving approximate methods as the only viable alternative. Building upon the algorithm proposed by Alcaraz et al. (2022) and drawing insights from the work of Rodríguez-Ballesteros et al. (2024) on the single-mode version of the problem, we develop a metaheuristic based on the non-dominated sorting genetic algorithm II (NSGA-II) framework to approximate the Pareto front (PF). Our algorithm incorporates a novel solution encoding scheme designed to handle both the multi-mode and multi-objective aspects of the problem. Additionally, we tailor key components of the metaheuristic, such as genetic operators, to effectively address this new variant. To evaluate the performance of the proposed algorithm, we conduct extensive experimental studies on a benchmark set of instances adapted from the well-known PSPLIB (Kolisch and Sprecher, 1997) and MMLIB (Van Peteghem and Vanhoucke, 2014) libraries. A rigorous assessment framework based on established quality indicators (Audet et al., 2021) is used to compare the PF approximations generated by the metaheuristic and the exact method.

In short, the contents of this chapter, which correspond to the contributions by Rodríguez-Ballesteros et al. (2025a), are twofold. First, it introduces a novel problem variant that incorporates multiple modes in multi-objective optimization, together with time-dependent resource costs and time-varying resource capacities. Second, it proposes a newly designed genetic algorithm tailored to this problem variant, along with a comprehensive evaluation framework that sets a benchmark for future research.

The remainder of this chapter is organized as follows. Section 4.2 provides a detailed discussion of the problem under study and introduces the corresponding mathematical model. An overview of the exact algorithm is presented in Section 4.3, followed by an extensive description of the proposed metaheuristic and its key features in Section 4.4. Section 4.5 outlines the experimental setup and presents a comprehensive analysis of the results obtained from the conducted experiments. Finally, Section 4.6 concludes the chapter by summarizing the main findings and discussing directions for future research.

4.2 Problem description

The MRCPSP is presented in this section, following the same notation conventions as in Chapter 3. This problem consists of scheduling a set of n activities $V = \{1, \ldots, n\}$, subject to precedence and resource constraints, within a finite time horizon $\mathcal{T} = \{0, \ldots, T-1\}$. Associated with each activity j we define the set of its m_j possible execution modes, $M_j = \{1, \ldots, m_j\}$. Each mode represents an alternative combination of activity duration and resource requirements. Let $\mathcal{M} = \{M_1, \ldots, M_n\}$ be the set of all possible execution modes for every activity in the problem. The duration or processing time of activity j operating in mode m is denoted by d_{jm}. The execution of an activity cannot be interrupted. For the sake of simplicity, "dummy" activities, 0 (start) and $n+1$ (end), are added to the project if necessary. Both activities have one mode with zero duration and zero resource requirements. Let P be the set of all pairs of immediate predecessors of activities, where single precedence constraints are given by (i, j), meaning that activity i is a direct predecessor of j.

We consider two resource categories: renewable and non-renewable. The set $\mathcal{R}$ of renewable re-

sources is composed by those resources available in limited quantities each time period. If the total consumption of a resource over the entire project is constrained, it is called non-renewable. We denote by $\mathcal{W}$ the set of non-renewable resources. The per period demand of activity j, performed in mode m, of renewable resource k is denoted by r_{jkm}, while R_{kt} represents the number of units of resource k available in time period t, i.e., resource capacities fluctuate over the time horizon. The amount of non-renewable resource l consumed by activity j, performed in mode m, is given by w_{jlm}, while W_l corresponds to the number of units of non-renewable resource l available for the entire life of the project.

We rely on the formal definition of the problem introduced by Talbot (1982), in which the objective is to find a schedule so that the project makespan or total project duration is minimized, and both the precedence and the resource constraints are satisfied. As in Chapter 3, a schedule $S = (S_1, \ldots, S_n)$ assigns a non-negative starting time S_j to every activity j. We define the earliest and latest starting times of each activity j—E_j and L_j, respectively—in the usual way, but using the set of minimum-duration modes for all activities. The former corresponds to the minimum time required to execute all the direct and transitive predecessors of j, while the latter is the maximum time j can be held up without causing a delay in the project. Typically, L_{n+1} is set equal to a known heuristic project completion time T, or, if T is not known, L_{n+1} is set equal to the sum of all maximum activity durations. Lastly, the binary variables of the model are defined as $x_{jtm} = 1$ if activity $j \in V$ operating in mode $m \in M_j$ starts at time $t \in \{E_j, \ldots, L_j\}$; otherwise, $x_{jtm} = 0$. Under these conditions, the MRCPSP can be formulated as follows:

$$\text{minimize} \quad \sum_{t=E_{n+1}}^{L_{n+1}} t\, x_{(n+1)t1} \tag{4.1}$$

$$\text{subject to} \quad \sum_{m\in M_j} \sum_{t=E_j}^{L_j} x_{jtm} = 1, \qquad \forall j \in V, \tag{4.2}$$

$$\sum_{m\in M_j} \sum_{t=E_j}^{L_j} t x_{jtm} - \sum_{m\in M_i} \sum_{t=E_i}^{L_i} (t + d_{im}) x_{itm} \geq 0, \quad \forall (i,j) \in P, \tag{4.3}$$

$$\sum_{j\in V} \sum_{m\in M_j} r_{jkm} \cdot \sum_{q=\max\{t-d_{jm}+1, E_j\}}^{\min\{t, L_j\}} x_{jqm} \leq R_{kt}, \quad \forall k \in \mathcal{R}, t \in \mathcal{T}, \tag{4.4}$$

$$\sum_{j\in V} \sum_{m\in M_j} \sum_{t=E_j}^{L_j} w_{jlm} x_{jtm} \leq W_l, \qquad \forall l \in \mathcal{W}, \tag{4.5}$$

$$x_{jtm} \in \{0,1\}, \qquad \forall j \in V, t \in \{E_j, \ldots, L_j\}, m \in M_j. \tag{4.6}$$

First of all, the makespan is defined in (4.1) and constitutes the objective function of the problem. Constraints (4.2) guarantee the execution of every activity of the project, beginning at a particular time period between its earliest and latest starting times. Constraints (4.3) refer to the precedence relationships, ensuring no activity starts before its predecessors have been completed. The renewable

resource constraints are included in (4.4), making sure the availability of each renewable resource is not exceeded during any time period. On the other hand, (4.5) refer to the non-renewable resource constraints, ensuring the number of units of the resource is not exceed withing the time horizon. Finally, (4.6) define the domain of the decision variables.

On the other hand, Alcaraz et al. (2022) introduce the RCPSP_TDRC, i.e., the cost depends on the resource being considered as well as the time period in which it is being used. In this context, they propose adding a second objective to be minimized in the problem, the total cost of the resource usage. Here, we incorporate the multiple modes of the activities and non-renewable resources, modeling the total cost as follows:

$$\sum_{j\in V}\sum_{m\in M_j}\sum_{t=\max\{0,E_j\}}^{\min\{T-1,L_j\}}\left[x_{jtm}\left(\sum_{q=t}^{t+d_{jm}-1}\sum_{k\in\mathcal{R}} r_{jkm}c^{\mathcal{R}}_{kq}+\sum_{l\in\mathcal{W}} w_{jlm}c^{\mathcal{W}}_{lt}\right)\right], \tag{4.7}$$

where $c^{\mathcal{R}}_{kt}$ ($c^{\mathcal{W}}_{lt}$) denotes the cost of employing one unit of renewable (non-renewable) resource k (l) during the interval of time $[t, t+1)$, $\forall k \in \mathcal{R}$ ($\forall l \in \mathcal{W}$) and $t \in \mathcal{T}$.

In this way, we define a new variant of the MRCPSP, a multi-objective problem with two objectives to minimize, the so-called bi-objective multi-mode resource-constrained project scheduling problem with time-dependent resource costs and capacities (MRCPSP_TDRCC), which can be formulated as follows:

$$\begin{aligned} &\text{minimize} && \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x})) \\ &\text{subject to} && (4.2)-(4.6), \end{aligned} \tag{4.8}$$

where $\Omega = \{\mathbf{x} \in \{0,1\}^{J\times\mathcal{T}\times\mathcal{M}} : (4.2)-(4.6)\} \subseteq \mathbb{R}^{J\times\mathcal{T}\times\mathcal{M}} \neq \emptyset$ is the feasible set, and $f_i : \{0,1\}^{J\times\mathcal{T}\times\mathcal{M}} \to \mathbb{R}$ are the objective functions, for $i \in \{1,2\}$. In particular, $f_1(\mathbf{y})$ corresponds to (4.1) and $f_2(\mathbf{y})$ corresponds to (4.7). The decision space and objective space are given by $\mathbb{R}^{J\times\mathcal{T}\times\mathcal{M}}$ and $\mathbb{R}^2$, respectively.

An illustration of the above problem is presented in Figure 4.1. The project comprises 7 activities that utilize both renewable and non-renewable resources. Only one resource of each type is employed. The availability of the renewable resource fluctuates within the time horizon, with 6 units during even time slots and 5 units during the odd ones. As for the non-renewable resource, there are 14 units available for the entire life of the project. Additionally, each activity can operate in one of two distinct execution modes. In particular, the time horizon is set to the sum of all maximum activity durations. Dummy activities 0 and 8 are also depicted in the figure, representing the project start and completion, respectively. Note that these dummy activities have only one execution mode, with no duration or resource consumption. Lastly, the time-dependent resource costs are included in the figure, varying from 2 to 4 depending on the time interval considered. Notice that in this example, the cost depends only on the time slot in which the resource is used, and not on the type of resource utilized.

As discussed in Chapter 3, working in a multi-objective context poses a challenge. Unlike single-objective optimization, where the concept of an optimal solution is straightforward and well-defined, multi-objective problems require balancing trade-offs among conflicting objectives. In this context, a good solution is no longer a single point but rather a set of solutions representing optimal compromises

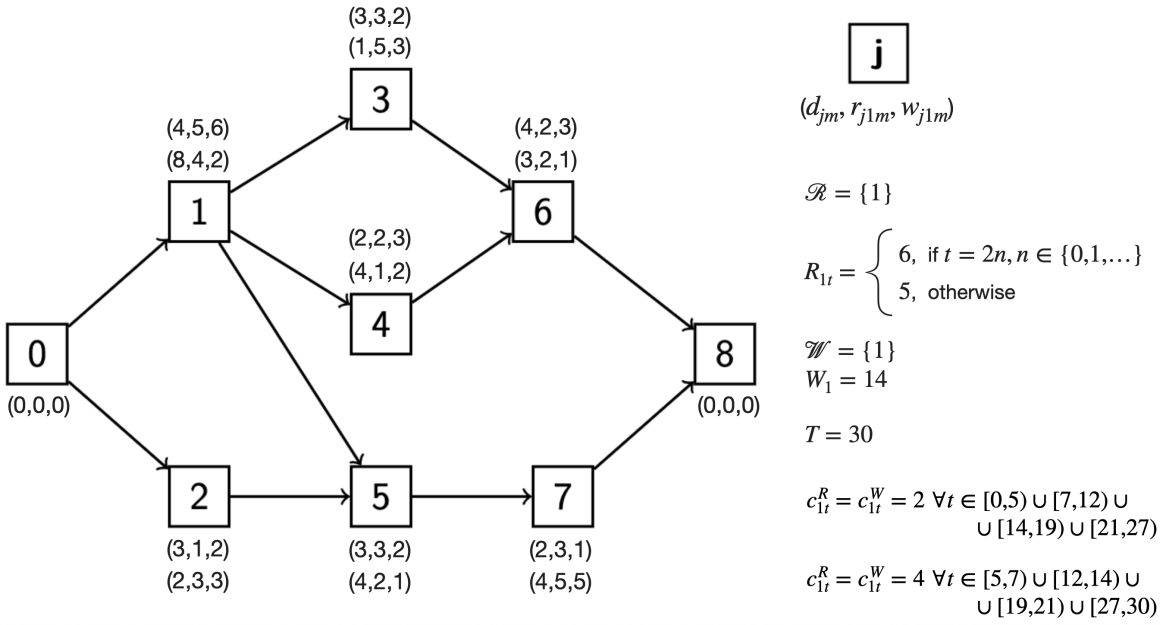


Figure 4.1: An example of the MRCPSP_TDRCC.

across all objectives. Figure 4.2 illustrates two feasible solutions to the previous project example. Activities are depicted by boxes, where the length of the box denotes the duration of the activity and the height indicates the units of the renewable resource demanded by the activity. The time-varying resource capacities are represented by the dashed lines. We observe that Schedule 1 has a greater makespan value than Schedule 2, whereas the total cost of resource usage is smaller. Hence, none of these solutions dominates the other, meaning that we cannot improve one objective without deteriorating the other. To address this, we seek the set of non-dominated solutions, commonly referred to as the PF, which provides decision-makers with alternative solutions that are incomparable with respect to the Pareto dominance order (see Section 2.1).

4.2.1 Data reduction

In this section, we present a preprocessing procedure designed to reduce the solution space by removing certain modes that do not affect the feasibility of the problem. This step is particularly valuable for enhancing computational performance without compromising solution quality. Specifically, we adopt the bounding rule proposed by Sprecher et al. (1997), which serves to adapt the input data before solving the problem.

We commence by introducing some notation related to the specific issue we are working with. First, we refer to the minimal request of activity j for non-renewable resource l by $wmin_{jl} := \min\{w_{jlm} \mid m = 1, \dots, M_j\}$. Analogously, the maximal request of activity j for non-renewable resource l is denoted by $wmax_{jl} := \max\{w_{jlm} \mid m = 1, \dots, M_j\}$.

Let us consider a mode m of an activity j. A mode m is said to be non-executable with respect to

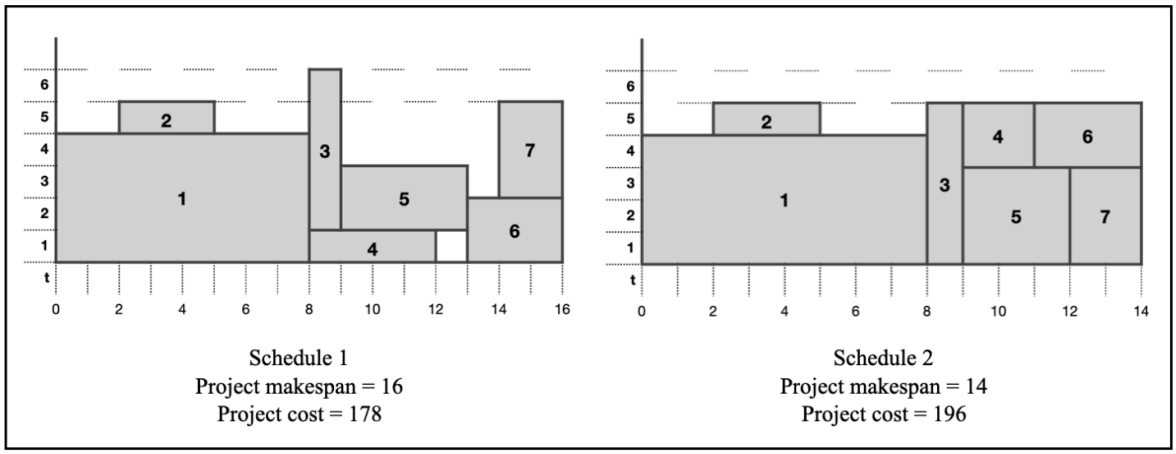


Figure 4.2: Different solutions to the problem in Figure 4.1.

a non-renewable resource $l \in \mathcal{W}$ if the following condition holds:

$$\sum_{\substack{i=1 \\ i \neq j}}^{n} wmin_{il} + w_{jlm} > W_l,$$

in other words, even if all other activities consume their minimum required amount of resource l, the total demand exceeds the available capacity. Regarding the renewable resources, we need to be more careful with the previous definition, as we are working with time-varying resource capacities, i.e., the per period availability of a renewable resource k varies within the entire life of the project. In this context, we say that a mode m is non-executable with respect to a renewable resource $k \in \mathcal{R}$ if

$$\nexists t \in \mathcal{T} \mid r_{jkm} \leq R_{kq}, \forall q \in \{t, \ldots, t + d_{jm} - 1\},$$

that is, the per period availability of the renewable resource k must be greater than the demand of activity j when executed in mode m until it is completed. Finally, mode m is called inefficient, if there exists another mode m' of activity j with $d_{jm} \geq d_{jm'}$ and $r_{jkm} \geq r_{jkm'}$ for each renewable resource $k \in \mathcal{R}$ and $w_{jlm} \geq w_{jlm'}$ for each non-renewable resource $l \in \mathcal{W}$.

In their work, Sprecher et al. (1997) also consider non-renewable resource $l \in \mathcal{W}$ to be redundant if its total maximum demand across all activities does not exceed its availability:

$$\sum_{j=1}^{n} wmax_{jl} \leq W_l.$$

Nonetheless, we cannot exclude redundant non-renewable resources from the input data, as we are addressing a bi-objective problem where the second objective function calculates the total cost of resource usage. While redundant resources may not impact the set of feasible schedules when minimizing the makespan alone, they are relevant in our context due to the associated costs.

As a result, the bounding rule has been simplified to a two-step procedure. It begins by removing all non-executable modes from the project data, followed by eliminating inefficient modes. Note that,

after this preprocessing procedure, an activity might not have a single executable mode, resulting in no feasible solution to the problem. Infeasible instances detected so far have been removed from the input data.

4.3 An exact solution method

After the preprocessing procedure is applied, the initial approach for solving this new variant of the MRCPSP involves employing an exact technique to obtain the set of Pareto-optimal solutions. Following Mavrotas (2009), we utilize the improved ϵ-constrained algorithm, known as the AUGMECON method. This procedure is designed to enhance the efficiency and effectiveness of its predecessor in finding Pareto-optimal solutions in multi-objective optimization problems. This method is selected due to its proven efficiency in solving multi-objective combinatorial problems with discrete decision variables. In particular, it has been successfully applied to the bi-criteria RCPSP_TDRC (Alcaraz et al., 2022), demonstrating strong performance in generating exact PF with reduced computational effort. First, we briefly describe the ϵ-constrained method, and then we present the mathematical formulation of the augmented problem.

The ϵ-constrained method is a well-known technique in vector optimization used to convert a multi-objective problem into a single-objective problem by optimizing one objective while setting bounds on the others through additional constraints. Given the multi-objective optimization problem in (4.8), the ϵ-constrained method transforms it into:

$$\begin{aligned} &\text{minimize} && f_1(\mathbf{x}), \\ &\text{subject to} && (4.2) - (4.6), \\ & && f_2(\mathbf{x}) \leq \epsilon. \end{aligned} \tag{4.9}$$

Here, ϵ is a predefined threshold value for the second objective. By varying the ϵ value, a set of Pareto-optimal solutions can be obtained. An optimal solution to the above problem is guaranteed to be a Pareto-optimal solution to (4.8) only if the ϵ-constraint is binding (Mavrotas, 2009). With this in mind, the augmented problem is defined as follows:

$$\text{minimize} \quad \sum_{t=E_{n+1}}^{L_{n+1}} t\, x_{(n+1)t1} - \gamma s, \tag{4.10}$$

$$\text{subject to} \quad (4.2) - (4.6),$$

$$\sum_{j\in V}\sum_{m\in M_j}\sum_{t=\max\{0,E_j\}}^{\min\{T-1,L_j\}} \left[x_{jtm} \left(\sum_{q=t}^{t+d_{jm}-1} \sum_{k\in\mathcal{R}} r_{jkm} c^{\mathcal{R}}_{kq} + \sum_{l\in\mathcal{W}} w_{jlm} c^{\mathcal{W}}_{lt} \right) \right] + s = \epsilon, \tag{4.11}$$

$$s \geq 0. \tag{4.12}$$

where s is the slack variable of the ϵ-constraint and γ is a small factor ensuring that the slack of the ϵ-constraint is as high as possible. The AUGMECON method iteratively replaces ϵ with different values

within the range of interest for the second objective function, $f_2(\mathbf{x})$, and solves the corresponding mixed-integer linear programming (MILP) problem for each value to obtain a set of Pareto-optimal solutions. The number of breakpoints assumed for the range of this function needs to be specified, which will be discussed in the computational results section. For a formal definition of this procedure, we refer to Alcaraz et al. (2022).

As already discussed, the RCPSP is strongly NP-hard, and so is its multi-mode generalization. Additionally, the problem of finding a feasible solution in the multi-mode scenario is NP-complete when considering at least two non-renewable resources, as demonstrated by Kolisch and Drexl (1997). Consequently, this new variant of the MRCPSP is also NP-complete, and exact techniques may fail to find Pareto-optimal solutions for large or even medium-sized instances. We introduce in the following section a new genetic algorithm to find, or at least approximate, the set of non-dominated solutions for the problem.

4.4 The genetic algorithm

As described in Section 2.2.3, in the context of multi-objective optimization, the use of multi-objective evolutionary algorithms (MOEAs) is well-established. These algorithms are inspired by natural evolution and incorporate common structures such as selection, crossover, and mutation operators. These operators are successively applied to individuals in the population to reach the PF of the problem, or at least a close approximation. For the bi-objective MRCPSP_TDRCC, specific features must be designed to incorporate problem-specific knowledge, ensuring the efficiency of the technique. To address this, we have designed and implemented a metaheuristic based on the general template of the non-dominated sorting genetic algorithm II (NSGA-II) (Deb et al., 2002), which has been successfully applied to a wide range of problems across various contexts. This technique was already introduced in Chapter 3. Since this metaheuristic serves as the core algorithm in the experimental study conducted in this chapter, we present its general template—which remains standardized and independent of the specific problem—and provide a brief description.

We first apply the preprocessing procedure to the project data. Following this, the genetic algorithm is initiated by the generation of an initial population. Each individual is decoded into a schedule, and the objective values—makespan and total cost—are computed. The crowded-comparison operator (Deb et al., 2002) is then applied, assigning each solution a non-domination rank and a crowding distance to guide the selection process toward a uniformly spread-out PF.

Hereafter, the main loop of the algorithm begins and continues until the stopping criterion is met. The typical genetic operators—selection, crossover, and mutation—produce an offspring population of the same size as the initial one. The selection mechanism prioritizes individuals with the best values according to the crowded-comparison operator, so they will be selected more often than those with worse values. After evaluating the offspring individuals, both the parent and offspring populations are combined, resulting in a double-sized population. The crowded-comparison operator is then applied to this combined population. Based on their rank and crowding distance, a new population of the original size is selected. This procedure is summarized in Algorithm 2. Further details on the crowded-

Algorithm 2 NSGA-II template

```
1: i ← 0;
2: P_i ← create_initial_population(N);
3: evaluate_population(P_i);
4: crowded_comparison_operator(P_i);
5: while not stopping_criterion do
6:    Q_i ← selection(P_i);
7:    Q_i ← crossover(Q_i);
8:    Q_i ← mutation(Q_i);
9:    evaluate_population(Q_i);
10:   R_i ← P_i ∪ Q_i;
11:   crowded_comparison_operator(R_i);
12:   P_{i+1} ← reduce_population(R_i);
13:   i ← i + 1;
14: end while
```

comparison mechanism and the NSGA-II algorithm can be found in Section 3.3.2.

4.4.1 Solution encoding

As previously mentioned, designing features that incorporate problem-specific knowledge is essential to achieve good performance with our metaheuristic when solving this new problem variant. In Chapter 3, we describe a problem-specific encoding scheme for the bi-objective RCPSP_TDRC. In this section, we extend that encoding to suit the characteristics of our proposed variant.

First, in the context of the RCPSP, several encodings have been proposed to address this problem. The permutation-based encoding, known as the activity list (AL) representation (Hartmann, 1998), is the most common approach. This AL represents an ordering or permutation of the project activities, where each activity is placed between its latest predecessor and its earliest successor. To obtain the schedule, activities are sequenced according to the established order in the AL, always starting after the completion time of their predecessors, ensuring that precedence constraints are preserved. Hartmann and Kolisch (2000) demonstrate that this representation outperforms other existing encodings. Various authors subsequently add to this list of activities information regarding the sequencing of activities to construct the schedule, which improves the results obtained (see, e.g., Hartmann, 1999; Alcaraz and Maroto, 2001; Alcaraz and Maroto, 2006).

In the multi-mode context, a common approach involves extending the AL representation to capture the different execution modes of each activity. Several authors have used a double list, consisting of an AL and a mode assignment list (ML), where the ML specifies the execution mode for each activity when constructing the schedule (see, e.g., Hartmann, 2001; Alcaraz et al., 2003; Bouleimen and Lecocq, 2003; Fernandes et al., 2018; Zamanii, 2019). This representation remains widely adopted in the literature due to its simplicity and flexibility. Among others, Van Peteghem and Vanhoucke (2011) propose a double-list encoding combined with a scatter search algorithm, where resource scarceness parameters

are used to adapt local improvement strategies according to the scarcity intensity of renewable and non-renewable resources. In contrast, Coelho and Vanhoucke (2011) introduce a unified encoding based on a single priority list that simultaneously captures activity sequencing and mode selection through a satisfiability problem (SAT)-based procedure. Although effective in certain settings, this compact encoding has been reported to involve high computational costs, making it less suitable for large-sized instances (Fernandes et al., 2018).

Lastly, as detailed in Section 3.3.1, Alcaraz et al. (2022) propose an innovative encoding for the bi-objective RCPSP_TDRC, aiming to minimize both the makespan and the total cost of resource usage. In their work, the authors use a double list: an AL and a binary list representing the scheduling objective selected for each activity. The binary list captures the multi-objective nature of the problem, prioritizing the makespan when the bit is 0 and the total cost when the bit is 1. Hence, if the scheduling objective of an activity is the makespan, it will be scheduled as soon as resources are available and its predecessors have been completed. If the second objective is prioritized, i.e., the cost, the start of the activity may be delayed up to a specific limit, allowing it to be scheduled in a time slot where the cost is lower.

Combining both scenarios, a natural approach is to use a triple list to represent the individuals or solutions to the problem: an AL representation, a binary list indicating the scheduling objective, and a ML. At this point, we must determine how much we can delay the start of an activity when the priority objective is minimizing the total cost. To address this issue, we employ the classical earliest and latest starting times, which are computed using the activity durations provided by the ML. For a given activity $j \in V$, if the priority objective is minimizing the makespan, we start its execution as soon as possible, while preserving precedence and resource constraints. We denote this starting time as S_j^{mak}. If the scheduling objective for activity j is minimizing the total cost, we can schedule it within the discrete interval $\{t \in \mathbb{N} \mid S_j^{mak} \leq t \leq L_j\}$, in a time slot where the cost is cheaper and there is availability of resources. Therefore, the starting time of the activity can be delayed up to its latest starting time, ensuring that the time horizon, T, is not exceeded. However, this encoding has a drawback in the scheduling generation scheme, as demonstrated in the following example.

Figure 4.3 shows two feasible solutions encoded with the triple list representation outlined above for the project example presented in Figure 4.1. Recall that the renewable resource capacity is 6 in the even time slots and 5 in the odd ones. Additionally, the non-renewable resource capacity is assumed not to be exceeded, ensuring both solutions are feasible. Lastly, the time-dependent resource costs are displayed on the horizontal axis, and they have been simplified for the purpose of clarification. Therefore, no differentiation between $c_{1t}^{\mathcal{R}}$ and $c_{1t}^{\mathcal{W}}$ has been made, as their values coincide within the time horizon.

Focusing on Solution A, we observe that all activities prioritize minimizing the first objective, except for the last one, activity 7, which focuses on minimizing the total cost. Consequently, the first six activities are scheduled as early as possible, the moment all their predecessors are completed and resources are available. For the last activity, $S_7^{mak} = 13$ and its latest starting time $L_7 = 28$. Note that the mode assignment determines the duration of each activity, and thus its earliest and latest starting times. Within the time interval $\{t \in \mathbb{N} \mid 13 \leq t \leq 28\}$, the optimal time to schedule activity

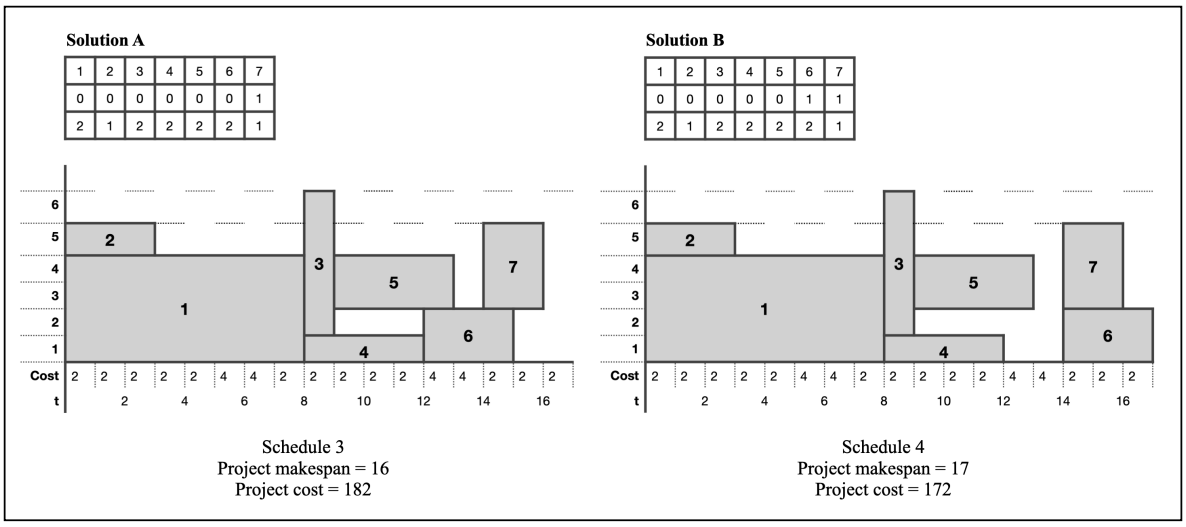


Figure 4.3: Example of a first attempt for the triple list representation.

7 is at time slot 14, where the cost is 2 until the activity is completed, a cost that cannot be reduced within this interval. Hence, activity 7 is delayed until time slot 14, as shown in the figure. Regarding Solution B, the scheduling objective for activity 6 is modified, shifting the priority to minimize the second objective. In this case, $S_6^{mak} = 12$ and $L_6 = 27$. The cheapest position within the interval $\{t \in \mathbb{N} \mid 12 \leq t \leq 27\}$ for activity 6 is also at time slot 14, where the cost is 2 during the execution of the activity. Consequently, activity 6 is delayed until time slot 14.

However, referring back to Schedule 1 in Figure 4.2, we observe that it cannot be obtained using the procedure described above. This schedule was generated by the exact method and illustrates a feasible solution that the described encoding is unable to reproduce. Specifically, activity 6 starts at time slot 13, which lies between the earliest possible starting time (12, corresponding to makespan minimization) and the cheapest one (14, corresponding to cost minimization). Under this encoding scheme, such intermediate positions are never considered, as activities are scheduled either at the earliest feasible time or at the time with the lowest cost. This example reveals a key limitation: when minimizing total cost, restricting the start time to only the cheapest slot within the interval $\{S_j^{mak}, \ldots, L_j\}$ may exclude feasible schedules that yield well-balanced compromises between objectives. As a result, the solution representation may fail to explore relevant regions of the search space, overlooking high-quality solutions and, thereby, deteriorating the performance of the metaheuristic.

To solve this problem, we need to allow more flexibility in choosing the cost position. At the same time, if we opt to schedule certain activities randomly within the interval, we must define a new value for the binary list to transmit this genetic material to the offspring generation. With this in mind, we replace the binary list with a newly defined scheduling objective list (OL), which takes three possible values, reflecting the bi-objective nature of the problem and adding flexibility to the scheduling generation scheme. The first two values, 0 and 1, are defined as before: 0 indicates that the priority is minimizing the makespan, while 1 indicates that the primary objective is minimizing the total cost. The third value is represented by the number 2 and indicates that a random position,

S_j^{rand}, is selected within the interval $\{S_j^{mak}, \ldots, L_j\}$. Therefore, if sufficient resources are available, activity j is scheduled at this random position. If not, we continue searching for a feasible time slot from that point to the end of the interval, and then from $S_j^{mak} + 1$ until S_j^{rand} is reached. In the worst-case scenario, S_j^{mak} is selected.

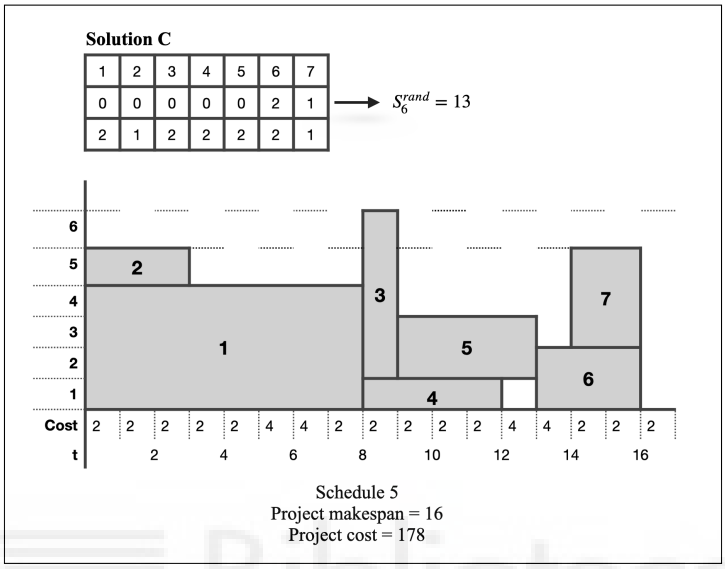


Figure 4.4: Example of our proposed triple list representation.

Figure 4.4 illustrates a feasible solution encoded using our new triple list representation for the project example shown in Figure 4.1. Specifically, it demonstrates how the scheduling generation scheme can position activity 6 in the middle, resulting in a schedule with the same makespan and total cost as Schedule 1 in Figure 4.2. Thus, this representation, along with its corresponding scheduling generation scheme, does not exclude feasible solutions, whether or not they lead to promising schedules.

To conclude, we focus on the scheduling generation scheme. The triple-list encoding designed for our metaheuristic introduces flexibility through the use of random positions—value 2 in the OL—allowing intermediate time slots between the earliest feasible starting time and the cheapest one to be randomly selected. Additional flexibility can be incorporated when the scheduling objective is cost minimization. In this case, instead of always selecting the time slot with the lowest cost, the algorithm subdivides the interval $\{S_j^{mak}, \ldots, L_j\}$ into several segments for each activity j, from which starting times are then selected. This mechanism enables the search to intensify around specific time slots that might otherwise be overlooked, thereby increasing the chances of discovering high-quality schedules. The specific strategy for subdividing and exploring this interval is empirically calibrated and tailored to the characteristics of our problem, as detailed in Section 4.5.2.

4.4.2 Initial population

After the search space is reduced through the preprocessing procedure, the genetic algorithm begins by creating the first generation of individuals, forming the initial population. For each individual, we

start by generating its AL. Activities are randomly selected from an eligible set, which is updated in each iteration with the activities whose predecessors have all been placed on the list. Next, the scheduling objective for each activity is randomly chosen from three possible values: 0 (makespan), 1 (total cost), and 2 (random position). Specifically, the probability of selecting value 2 is $\frac{1}{n}$, where n represents the number of activities in the project. The remaining two objectives—makespan and total cost—are selected with equal probabilities of $\frac{1}{2} - \frac{1}{2n}$. We assign a lower probability to the third value to prevent an excessive number of initially assigned random positions. This design choice is supported by preliminary experiments, which showed a clear deterioration in algorithm performance when too many activities were initially assigned to random positions. Regarding the ML, a common approach is to randomly select the execution mode for each activity from its available modes, i.e., the ones remaining after preprocessing. Nonetheless, among the obtained individuals, infeasible solutions may appear, compromising the quality of the initial population.

We first address infeasibilities related to non-renewable resources. In the multi-mode scenario, infeasible mode assignments with respect to non-renewable resource constraints can occur, and finding a feasible solution when multiple non-renewable resources are involved is an NP-complete problem. Hence, solutions under these conditions are allowed, but we aim to maximize the probability of obtaining feasible solutions in the initial population through a local search procedure previously used by Lova et al. (2009). They refer to this procedure as Minimum Normalized Resources (MNR), which involves computing the value

$$NR_{jm} = \sum_{l \in \mathcal{W}} \frac{w_{jlm}}{W_l}$$

for each activity j, and selecting the execution mode that minimizes this value. In their work, the authors demonstrate the superior performance of this technique, achieving feasible non-renewable mode assignments in over 95% of cases. When the obtained mode assignment is not feasible, we apply their randomized version of the MNR procedure. Initially, the ML is generated following the MNR criterion. Subsequently, the mode of up to $\frac{n}{2}$ activities is randomly altered. If the resulting mode assignment is feasible, the procedure stops. Otherwise, activities are randomly sorted, and each activity attempts to select one of its possible execution modes to reduce infeasibility. If a feasible mode assignment is achieved at this point, the process ends. If not, the procedure is repeated up to a specific number of attempts, which, following the experimental study conducted by Lova et al. (2009), is set to 200.

After obtaining a feasible non-renewable mode assignment—or infeasible, after carrying out the previous procedure—we focus on addressing infeasibilities related to renewable resources. Without loss of generality, we assume that a feasible non-renewable mode assignment has been obtained. As mentioned earlier, the multi-objective nature of the problem is incorporated to the solution encoding through the OL. In this context, the starting time of an activity may be delayed to a specific position either to minimize the total cost of the schedule or due to a randomly assigned time slot, reducing the number of possible time slots for starting its successors. Consequently, an infeasible schedule may occur if there are insufficient renewable resource units to execute these successors. To minimize such infeasibilities, we implement a local search mechanism. First, we evaluate the individuals in the initial population and select those that are infeasible with respect to renewable resource constraints. For each

selected individual, a random activity is chosen and its priority objective changes its value to one of the two remaining options. Specifically, value 2 is chosen with probability $\frac{1}{n}$, while values 0 and 1 are selected with equal probability. If the resulting individual is feasible, the procedure stops. Otherwise, the previous step is repeated up to a maximum number of n retries, with n being the number of activities in the project. Note that this local search does not affect the mode assignment, ensuring that no new infeasibilities related to non-renewable resources can arise after applying this mechanism.

4.4.3 Penalized Objective Function

In the context of EAs, a crucial aspect is the design of an effective mechanism to compare the quality of the solutions produced. In single-objective optimization, this is typically achieved by assigning each individual a fitness score that indicates how well it addresses the problem at hand. The selection operator is then biased towards individuals with better scores, increasing their probability of survival to the next generation. As discussed in Section 2.1.1, in EAs, this fitness score often corresponds to the objective function value itself or a transformation of it.

In MRCPSP, when the single-objective is to minimize the project completion time, the makespan of the schedule associated with an individual typically serves as its fitness score. However, in environments where infeasible individuals may persist, additional considerations are required. Alcaraz et al. (2003) propose a fitness function for the MRCPSP that penalizes infeasible individuals with respect to non-renewable resource constraints. Precisely, the fitness function ensures that the score of an infeasible individual is always worse than the one of any feasible solution, while still allowing for differentiation among infeasible solutions depending on the degree of infeasibility. This mechanism allows infeasible individuals with potentially valuable traits to pass them on to the next generation. It is important to highlight that, in the context of makespan minimization, and after removing non-executable modes with respect to renewable resource constraints, infeasibilities can only arise due to non-renewable resource constraints.

In contrast, when dealing with multi-objective optimization problems, the concept of fitness assignment becomes more sophisticated. As outlined in Section 2.2.3, MOEAs are generally categorized into three main paradigms: Pareto-based, indicator-based, and decomposition-based approaches. In Pareto-based MOEAs, fitness assignment is guided by the Pareto dominance relation, while also incorporating mechanisms to preserve diversity among solutions. Since our metaheuristic is based on NSGA-II, it follows this paradigm and uses the crowded-comparison operator for fitness assignment (see Section 3.3.2).

In this chapter, where both makespan and total cost of resource usage are minimized simultaneously, candidate solutions are first evaluated by computing these two objective values. However, to appropriately handle infeasible solutions—those violating renewable or non-renewable resource constraints—we incorporate a penalization mechanism. Based on the ideas of Alcaraz et al. (2003) for the single-objective case, we adjust both objective values by adding a penalization term based on the degree of infeasibility. This adjustment ensures that feasible solutions are favored during the selection process, while still allowing infeasible solutions to participate in the evolutionary process and contribute

valuable genetic material to future generations. Once the penalized objective values are computed, the NSGA-II assigns fitness following the crowded-comparison operator to guide the search toward a well-distributed PF.

We begin by describing the penalized function associated with the first objective, i.e., the makespan. Given an individual i, the penalized objective value $\hat{f}_1(i)$ is defined by Alcaraz et al. (2003) as follows:

$$\hat{f}_1(i) = \begin{cases} mak(i), & \text{if } SFT = 0, \\ max\_mak(P) + mak(i) - min\_CP + SFT(i), & \text{otherwise,} \end{cases}$$

where $mak(i)$ denotes the makespan of the schedule associated with individual i, and $max\_mak(P)$ represents the maximal makespan among all feasible schedules in population P. As such, $max\_mak(P)$ serves as an upper bound for the makespan of all feasible solutions in the population. To this value, we add the increase over the makespan due to the minimal critical path, min_CP, which is computed using the set of minimum duration modes for all activities. Lastly, $SFT(i)$ represents the number of non-renewable resource units that individual i exceeds its availability:

$$SFT(i) = \sum_{l \in \mathcal{W}} max\left\{0, \sum_{j \in V} (w_{jlm_j} - W_l)\right\},$$

where m_j denotes the execution mode of activity j as defined by the ML.

Following the same idea, we define $\hat{f}_2(i)$ as the penalized value of individual i with respect to the second objective function, i.e., the total cost of resource usage, as follows:

$$\hat{f}_2(i) = \begin{cases} cost(i), & \text{if } SFT = 0, \\ max\_cost(P) + cost(i) - cost\_LB + SFT(i), & \text{otherwise,} \end{cases}$$

where $cost(i)$ represents the total cost of resource usage for the schedule associated with individual i, and $max\_cost(P)$ denotes the maximal cost among all feasible schedules in population P. To this upper bound, we add the excess of non-renewable resources $SFT(i)$ and subtract a lower bound for the total project cost, $cost\_LB$. The lower bound is computed based on a hypothetical schedule where each activity is executed using resources at their minimum cost. Specifically, the minimum value of the time-dependent resource costs is calculated for each resource in the project. Moreover, we account for the minimal requirements for both renewable and non-renewable resources.

Building on these concepts, we define the penalized objective vector-valued function, $\hat{\mathbf{f}}$, which assigns to each individual i the penalized values of both objective functions:

$$\hat{\mathbf{f}}(i) = \begin{cases} (\hat{f}_1(i), \hat{f}_2(i)), & \text{if } i \text{ is feasible w.r.t. renewable resources,} \\ (\infty, \infty), & \text{otherwise.} \end{cases}$$

Thus, each individual i is represented by a bi-dimensional point, with one penalized objective value for each criterion: makespan and total cost of resource usage. Specifically, infeasible solutions that violate renewable resource constraints are assigned the worst possible objective vector, (∞, ∞), excluding them from the genetic process, as their survival probability is close to zero. In contrast,

infeasible solutions with respect to non-renewable resource constraints—but feasible regarding renewable resource constraints—are assigned penalized objective values based on their makespan and total cost, and not solely by the excess of non-renewable resources. Hence, two solutions with identical SFT but differing objective values will be treated differently during fitness assignment. Furthermore, this penalty mechanism ensures that infeasible individuals concerning non-renewable resources with better penalized objective values may exert a greater influence on the population than those exhibiting worse scores. That is, they have a higher probability of being selected and participating in the genetic process.

4.4.4 Selection operator

A fundamental component of genetic algorithms is the use of biologically inspired operators such as selection, crossover, and mutation. This section focuses on the selection operator, which is grounded in the natural principle of survival of the fittest: the idea that individuals with superior traits are more likely to persist and contribute to subsequent generations within the population. Consequently, the selection process must be designed to direct the search toward individuals with the most desirable features.

In this chapter, we adopt the selection procedure proposed by Deb et al. (2002) in the NSGA-II framework, which relies on standard binary tournament selection. This method involves randomly sampling pairs of individuals from the population to compete for survival. The tournament is guided by the crowded-comparison operator, which prioritizes individuals based on their non-domination rank and crowding distance. Specifically, the individual with the lower rank is selected, and in cases where both solutions belong to the same front, the winner of the tournament is the one with the higher crowding distance value. The procedure is applied iteratively over the current population until a new one of the same size is created. Elitism is introduced, thus guaranteeing that high-quality and diverse solutions are consistently represented in the next generation.

4.4.5 Crossover operator

Among the various genetic operators that participate in a genetic algorithm, crossover, also known as recombination, plays a fundamental role. It not only combines information from two parent solutions to produce two offspring that inherit their traits, but also aims to achieve a beneficial recombination. In detail, after the selection procedure, individuals in the current population are randomly paired to form couples—a mother and a father—who undergo the crossover operation with a specified probability, p_c. If a couple undergoes this procedure, two offspring—a daughter and a son—are produced, replacing their parents in the current population to maintain a constant population size. Otherwise, the parent solutions remain unchanged in the population.

Regarding the triple list introduced earlier for encoding solutions to our problem, the crossover operator must effectively manage and combine information from all three components. To achieve this, we implement a three-phase procedure. First, the two-point crossover proposed by Hartmann (1998) is applied to the ALs of the parents. Two non-negative crossover points, k_1 and k_2 ($1 \leq k_1 < k_2 \leq n$),

divide each AL into three segments. The son inherits the first k_1 positions from the father, exactly in the same order. The intermediate positions, from $k_1 + 1$ to k_2, are inherited from the mother, ensuring no duplicates and maintaining their relative order as in the mother's list. Lastly, the positions after the second crossover point, k_2, are inherited from the father, again maintaining their relative order and avoiding repetition. Precedence constraints are also preserved, ensuring that no activity appears before its predecessors. The daughter is generated analogously, but interchanging the parents. Second, the scheduling objective values for each activity in the offspring are directly inherited from the corresponding parent. Similarly, in the third phase, each activity in the offspring inherit the mode assignments from the parent from which they were derived. As a result, the offspring inherit a combination of traits from both parents.

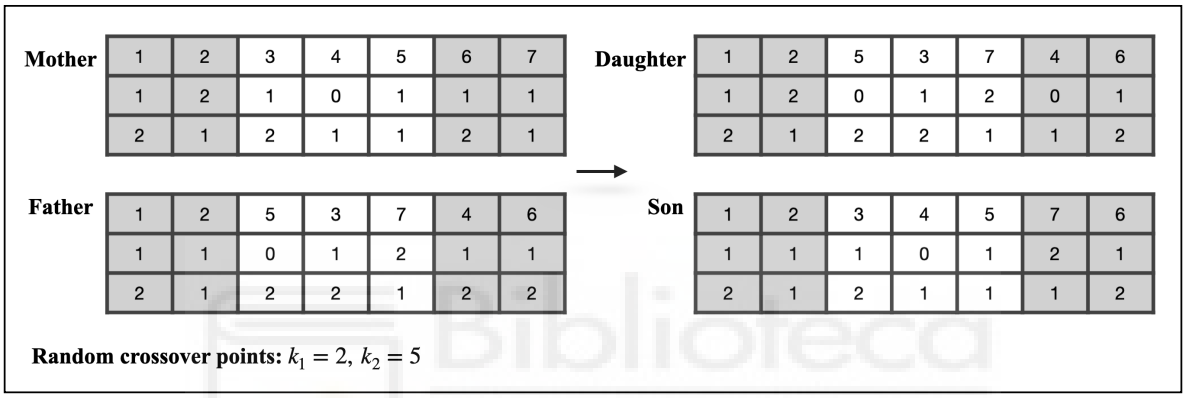


Figure 4.5: Crossover example.

Figure 4.5 illustrates the described procedure by combining two solutions from the problem example in Figure 4.1. The random crossover points, $k_1 = 2$ and $k_2 = 5$, divide the lists into three segments. To explain the three-phase crossover operator, we focus on the daughter, as the mechanism is analogous for the son. Regarding the AL, the first two positions of the daughter are inherited from the mother in the same order. Next, we look for the first three activities in the father that are not already present in the daughter, preserving their relative order in the father's list. Finally, the remaining positions are taken from the mother, again skipping already included activities and preserving order. The mode and scheduling objective of each activity are inherited from the same parent from whom the activity is taken in the AL. For example, activity 3 is inherited from the father, where the scheduling objective has value 1 and the mode assignment has value 2. These values are maintained in the daughter, as shown in the figure.

4.4.6 Mutation operator

Once the parent population is recombined to obtain a new population, the mutation mechanism is applied to every individual with a specified probability. The purpose of mutation is to introduce additional variability into the population, thereby preventing premature convergence and enabling the genetic algorithm to explore a broader solution space. By altering one or more positions within the

triple list encoding, the mutation operator introduces some extra variability and reintroduces genetic material that may have been lost. In detail, we have designed a three-phase mutation operator that is applied to all three components of the solutions representation: the AL, the OL, and the ML. Each individual in the current population undergoes this procedure.

In the first stage, the mutation mechanism employed is an extension of the insertion operator proposed by Alcaraz and Maroto (2001) for solving the RCPSP. Each activity in the list is inserted into a new position, between the last position of its predecessors and the first position of its successors, with probability p_m. This ensures that only precedence-feasible solutions are generated. After the insertion, the scheduling objective value and mode assignment of the activity remain unchanged. In the second stage, the scheduling objective value of each activity is altered among the remaining two options with probability p_m. Specifically, the value 2 is always selected with probability $\frac{1}{n}$, while values 0 and 1 are chosen with equal probability.

The third stage only affects the ML. Following Lova et al. (2009), the procedure depends on whether the current mode assignment is feasible with respect to non-renewable resource constraints. If the mode assignment is feasible, each activity randomly selects a new execution mode from its available modes with a mutation probability p_m. However, if the mode assignment is infeasible—meaning it violates the non-renewable resource constraints—a local search technique, referred to as massive mutation, is applied with probability p_{mas}. In this approach, each activity is randomly assigned a new execution mode from all available modes, regardless of its previous assignment. This process continues until a feasible mode assignment is identified or all activities in the list have been considered. If massive mutation is not applied, the individual undergoes the standard mutation operation instead.

4.4.7 Random replacement

In this section, we describe a random procedure commonly used in genetic algorithms to enhance performance. This operation is applied to each generation with a specified replacement probability, p_r. When performed, each individual in the current population has an exchange probability p_e of being replaced by a solution generated using the randomized version of the MNR procedure. The number of attempts to generate a random solution using this mechanism is set to 1. To minimize infeasibilities caused by renewable resource constraints, we apply the local search mechanism described above with a total of retries equal to 10% of the activities. This procedure helps restart the population or reintroduce variability, particularly when the algorithm is stuck in a local optimum or has prematurely converged.

4.5 Computational results

This section presents the results obtained from the computational experiments carried out in this chapter. We begin by describing the experimental setup, including the problem instances used and the methodological framework employed. Next, we provide a detailed account of the calibration procedures for both the exact technique and the genetic algorithm. Finally, we conduct a comprehensive analysis of the numerical results, using the various metrics included in this study to compare the performance

of our proposed metaheuristic with that of the exact method.

4.5.1 Experimental setup

To evaluate the performance of the proposed techniques, we rely on newly generated datasets specifically tailored to our variant of the bi-objective MRCPSP. In particular, we build on the multi-mode benchmark instances from the PSPLIB (Kolisch and Sprecher, 1997) and MMLIB (Van Peteghem and Vanhoucke, 2014) libraries, previously described in Section 2.4. These instances assume 2 renewable and 2 non-renewable resources, and 3 execution modes per activity. From PSPLIB, we utilize the J20 dataset, which contains small-sized sprojects with 20 activities. From MMLIB, we incorporate two of its three sets: MMLIB50 and MMLIB100, which feature projects with 50 and 100 activities, respectively. The J20 dataset comprises 640 instances labeled as J20Y_Z, where Y ranges from 1 to 64, identifying the parameter configuration, and Z ranges from 1 to 10, indicating the instance replicate. Due to conflicts between execution modes and resource constraints, some projects lack feasible solutions. These infeasible instances are removed, resulting in a final dataset of 554 instances. As for the MMLIB50 and MMLIB100 datasets, both consist of 540 instances with guaranteed feasibility and only efficient modes. Similarly, these instances are labeled as JXY_Z, where where X denotes the dataset size (50 or 100), Y ranges from 1 to 108, and Z ranges from 1 to 5.

Note that none of the PSPLIB or MMLIB instances considered include time-dependent resource costs or time-varying resource capacities. These components must be generated to adapt the original instances to the specific requirements of our problem variant. To this end, we adopt the procedure outlined by Alcaraz et al. (2022), in which time-dependent resource costs are generated based on four distinct patterns of cost evolution over time. This procedure is described in detail in Section 3.4.1. In this study, we also apply these four patterns, as the PSPLIB and MMLIB instances involve two resources in each category. Specifically, patterns 1 and 4 are assigned to renewable resources 1 and 2, respectively, while patterns 2 and 3 are applied to non-renewable resources 1 and 2, respectively. Regarding the time-varying resource capacities for renewable resources, these are modified as follows. In 50% of cases, the original availability is retained. In the remaining 50%, resource availability is increased by up to 25%, truncating to the previous integer to ensure the increment does not exceed this limit. Hence, capacities lower than 4 are not modified. Lastly, to avoid introducing infeasibilities into originally feasible instances, reductions in resource availability are not considered. The GitHub repository `https://github.com/jalcarazs/Bi-objective-MRCPSP_TDRCC` contains all instances used in this chapter.

To evaluate the performance of our proposed metaheuristic, the experimental procedure is divided into two phases. In the first phase, we determine the most promising configuration for the genetic algorithm using a calibration set of instances. Specifically, we use only the J20 dataset, as it contains the largest number of instances for which we can obtain the exact PF using the exact method. For the calibration set, we select the first instance, J20Y_1, from each problem class, resulting in a total of 59 instances. Note that infeasible instances have been removed from the dataset. Therefore, if the first instance replicate is unavailable, we select the next available one. From these 59 instances, we use 6 to

assess the behavior of the exact method, as discussed in the following section. In the second phase, we compare our metaheuristic to the exact algorithm across three different evaluation sets. The first set consists of the second to fourth instances, J20Y_2, J20Y_3 and J20Y_4, for each problem class from the J20 dataset, among the available ones. This results in a total of 166 instances. The remaining two sets are derived from the MMLIB library. Specifically, for the MMLIB50 dataset, we select the first instance, J50Y_1, for each even-numbered problem class (i.e., where Y takes even values), leading to a total of 54 instances. This approach ensures that we include instances with all possible parameter configurations. For the MMLIB100 dataset, we select the first instance replicate, J100Y_1, at intervals of 12, resulting in a total of 10 instances. We select a smaller number of instances because the exact method requires significant computational time to solve these large projects, so we only present results for a few as examples. It is important to note that the exact method fails to identify any feasible solutions within the given time frame for certain problems in this dataset. In such cases, we proceed to the next problem class.

To conclude, in this chapter we consider seven performance indicators to evaluate the PF approximations obtained by the proposed metaheuristic: the overall non-dominated vector generation (OVNG), the C-metric (C), the Γ indicator (Γ), Zitzler's $\mathcal{M}_3^*$ metric ($\mathcal{M}_3^*$), the μ-indicator (μ), the additive ϵ-indicator (I_{ϵ^+}), and the hypervolume (HV). A detailed description of these metrics can be found in Section 2.1.1. Additionally, before computing any performance indicator, we apply linear normalization to the objective values of each solution in the PF approximation. Specifically, this normalization uses the minimum and maximum values of each objective, computed over the entire set of solutions obtained for a given problem instance.

4.5.2 Calibration of the algorithms

To evaluate the performance of an algorithm and compare it with another methodology, particularly for complex multi-objective problems, calibration is essential to ensure a fair and accurate comparison. Proper tuning guarantees that observed performance differences reflect true efficiency rather than discrepancies in parameter settings. This is especially critical when managing trade-offs between conflicting objectives. In the following, we outline the experimental setup for the exact method and provide a detailed account of the calibration process for the genetic algorithm.

The AUGMECON algorithm is implemented in C++ and integrated with IBM CPLEX 22.1 through Concert Technology. As described in Section 4.3, this approach involves solving a sequence of MILP problems for each instance. The number of problems to be addressed depends on the partitioning of the cost range, specifically the number of breakpoints used to divide this interval. With n breakpoints, the cost range is split into n subintervals, allowing the AUGMECON method to generate up to $n + 1$ Pareto-optimal solutions. It is worth noting that the cost range is defined by the cost values in the payoff table, i.e., the costs of the two most extreme Pareto solutions. For each instance, one extreme corresponds to the solution with the largest makespan and the smallest cost, while the other corresponds to the solution with the smallest makespan and the largest cost.

To determine an appropriate number of breakpoints, we conduct preliminary experiments on a

subset of instances from the J20 calibration dataset. Specifically, we select six instances from the calibration set at intervals of 10: J2010_1, J2020_1, J2030_1, J2040_1, J2050_1, and J2060_1. Each instance is solved using the AUGMECON algorithm with varying numbers of breakpoints: 10, 30, 50, 70, 90, 110, 130, 150, and 170. A time limit of 30 minutes is imposed for each MILP problem. The results are presented in Table 4.1 and illustrated in Figure 4.6.

Breakpoints	**J2010_1**	**J2020_1**	**J2030_1**	**J2040_1**	**J2050_1**	**J2060_1**	**Avg J20**
10	10	7	10	8	7	8	8.3
30	25	17	22	16	15	17	18.7
50	36	25	32	20	21	24	26.3
70	44	26	40	25	25	26	31.0
90	49	28	46	29	30	30	35.3
110	56	30	46	32	33	30	37.8
130	57	31	49	35	34	32	39.7
150	62	37	52	37	35	35	43.0
170	61	33	52	37	36	35	42.3

Table 4.1: Number of non-dominated solutions obtained for different numbers of breakpoints across six instances of the J20 dataset. The average values are reported in the last column.

From both Table 4.1 and Figure 4.6, we observe that the average number of Pareto solutions stabilizes around 150 breakpoints, with a slight decrease at 170. Based on this trend, we fix the number of breakpoints at 150 for all executions involving the J20 dataset. Nonetheless, for the larger datasets, MMLIB50 and MMLIB100, the computational complexity required adjustments. In these cases, we increase the time limit per breakpoint while reducing the total number of breakpoints. Specifically, for MMLIB50 instances, we set 100 breakpoints with a time limit of 1 hour per MILP execution. For the larger MMLIB100 instances, a maximum of 50 breakpoints is used, with a time limit of 2 hours per model. The chosen balance between problem complexity, number of breakpoints, and time limits proved to be adequate according to results from preliminary experiments.

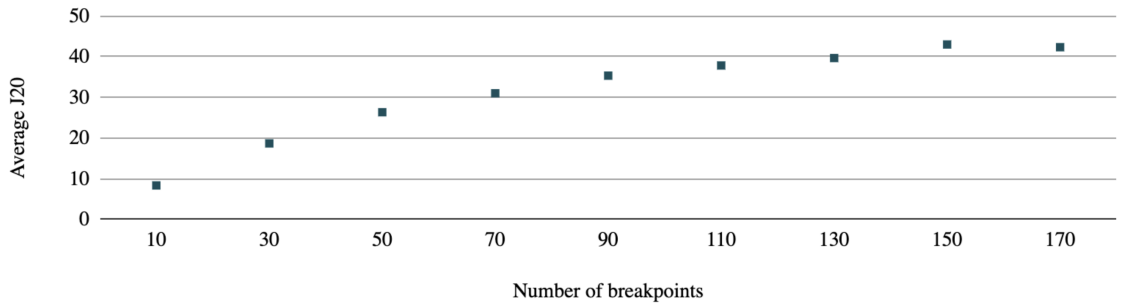


Figure 4.6: Average number of non-dominated solutions obtained for the six instances of the J20 dataset, for different number of breakpoints.

The genetic algorithm is implemented using the open-source jMetal framework (Durillo and Nebro, 2011; Nebro et al., 2015). This framework is extended to include the specific elements described above, such as solution encoding, genetic operators, and other tailored features. To configure the metaheuristic, we select a suitable set of parameter values and conduct a full factorial design of experiments, evaluating all possible combinations of factors and levels. Specifically, we consider six parameters: population size (N), crossover probability (p_c), mutation probability (p_m), replacement probability (p_r), exchange probability (p_e), and massive mutation probability (p_{mas}). Based on preliminary experiments, we define two levels for N (100 or 200) and two values for both crossover (0.7 or 0.9) and mutation ($\frac{1}{n}$ or $\frac{2}{n}$) probabilities, where n represents the number of activities in the project. For the random replacement procedure, we explore three levels for p_r (0, 0.05 or 0.1) and two levels for p_e (0.1 or 0.2). Notably, if $p_r = 0$, the random replacement procedure is not applied, regardless of the value of p_e, as no individuals in the current population are replaced with randomly generated ones. Consequently, when $p_r = 0$, the configuration remains unchanged whether p_e is set to 0.1 or 0.2. Lastly, we consider two levels for p_{mas} (0 or 0.1).

Attending to the different parameters and their levels, the factorial design for our metaheuristic results in a total of 80 possible configurations. In detail, when $p_r = 0$, the random replacement procedure is not applied, making p_e irrelevant. In this case, the factorial design yields $2 \times 2 \times 2 \times 1 \times 2 = 16$ possible configurations. For the remaining values of p_r, both levels of p_e are valid, resulting in $2 \times 2 \times 2 \times 2 \times 2 = 32$ configurations each. Summing these values gives a total of 80 possibilities. Each configuration is tested on the 59-instance calibration set, with three independent runs of the algorithm performed per instance. This amounts to a total of 14,160 runs of the genetic algorithm. The stopping criterion is set to 15 million evaluations per run, which results in a reasonable compromise between computation time and results quality.

As in Chapter 3, to compare all configurations with a certain level of confidence, we apply two statistical tests: analysis of variance (ANOVA) and Tukey's Honest Significant Difference (HSD) test for multiple comparisons. Both tests are conducted at a 95% confidence level. All necessary assumptions are checked, with no issues detected in the residuals, which are assumed to follow a Gaussian distribution. The response variables analyzed with ANOVA and Tukey's HSD test include the hypervolume, additive ϵ-indicator, μ-indicator, and C-metric, leading to four statistical models for each test. These four performance indicators are selected from the seven introduced in this chapter, as they collectively capture the four main desired properties—cardinality, convergence, distribution, and spread (see Section 2.1.1)—while enabling a more focused and interpretable statistical analysis. The remaining indicators are considered later in Section 4.5.3, where they support the qualitative evaluation of the proposed metaheuristics and contribute to a more comprehensive comparison with the exact method.

The factors (independent variables) for these models correspond to the various parameters considered during the calibration phase, along with their respective levels. A detailed description of the models and a comprehensive analysis of the results is provided in the Appendix A.3. Overall, the results reveal that all factors, except for the massive mutation probability (p_{mas}), are statistically significant. Precisely, the best results are observed when $p_r = 0$, indicating that the random replace-

ment procedure should not be applied. Table 4.2 summarizes the best configuration identified for our metaheuristic. Since p_{mas} is not statistically significant and the massive mutation mechanism increases the computational effort of the algorithm, we decide to set $p_{mas} = 0$. These findings suggest that the remaining genetic operators—namely, crossover and mutation—are sufficient to introduce the necessary variability into the population. As a result, the use of additional diversification mechanisms becomes unnecessary in this context.

	Parameters					
	N	p_c	p_m	p_r	p_e	p_{mas}
Best	200	0.7	$1.0/n$	0	-	0

Table 4.2: Best parameter configuration for the genetic algorithm based on the calibration results. The symbol "-" indicates that the parameter p_e is irrelevant.

To conclude the calibration of the metaheuristic, we focus on the procedure used to effectively select the starting time S_j of activity j when the scheduling objective is to minimize the total cost, as described in Section 4.4.1. This mechanism subdivides the interval of feasible starting times, $\{S_j^{mak}, \ldots, L_j\}$, and adjusts the selection probabilities based on the number of evaluations performed. Extensive computational experiments have shown that this procedure is essential to refine the schedule generation scheme and to prevent the exclusion of feasible schedules that could potentially lead to high-quality solutions.

We now detail how this mechanism operates for a given activity j. Assume that minimizing the total cost is the primary objective. The optimal cost position is determined within the interval $\{S_j^{mak}, \ldots, t'\}$, where $t' \in \mathbb{N}$ represents the upper bound of the possible starting times and is obtained using the following method:

- In 20% of cases, t' is randomly selected from $\{S_j^{mak} + 1, \ldots, L_j\}$.
- In the remaining 80%, t' is randomly chosen from specific subintervals, depending on the number of evaluations performed. Let the length of the interval of possible starting times for activity j be denoted by

$$\mathcal{L}_j = (L_j - S_j^{mak}) + 1.$$

  The procedure is then as follows:

  - During the first 5% of evaluations, it is selected from the interval $\{\frac{3}{4}\mathcal{L}_j + 1, \ldots, \mathcal{L}_j\}$.
  - In the next 5%, it is selected from $\{\frac{1}{2}\mathcal{L}_j + 1, \ldots, \frac{3}{4}\mathcal{L}_j\}$.
  - In the following 10%, it is selected from $\{\frac{1}{4}\mathcal{L}_j + 1, \ldots, \frac{1}{2}\mathcal{L}_j\}$.
  - In the subsequent 10%, it is selected from $\{\frac{1}{8}\mathcal{L}_j + 1, \ldots, \frac{1}{4}\mathcal{L}_j\}$.
  - Finally, during the last 70% of evaluations, it is selected from $\{S_j^{mak} + 1, \ldots, \frac{1}{8}\mathcal{L}_j\}$.

Notably, if $\mathcal{L}_j = 1$, then $L_j = S_j^{mak}$, implying that the optimal cost position coincides with the starting time determined by the makespan.

This procedure highlights a key challenge: the algorithm struggles to find feasible solutions in the leftmost part of the PF, i.e., schedules with low makespan values but high total cost value. A straightforward explanation for this issue lies in the design of the scheduling generation scheme, which allows for a broad range of possible starting time slots by using L_j as the right boundary of the interval. While this approach enables the algorithm to explore all potential starting times, it also results in the displacement of activities in the generated schedule when the focus shifts to minimizing total cost. Hence, we aim to identify cost-effective solutions without significantly delaying the starting times of activities. To achieve this, we assign a higher probability to selecting $t' \in \mathbb{N}$ within the interval $\{S_j^{mak} + 1, \ldots, \frac{1}{8}\mathcal{L}_j\}$. This adjustment is made during the final 70% of evaluations, when the solutions in the population, having been combined through genetic operators, exhibit strong properties. Finally, note that some operations may yield decimal values, which are truncated to integers during implementation, given that the makespan of any schedule is always an integer value.

4.5.3 Numerical results and analysis

This section presents the main computational results of this study. For each evaluation set we report the results obtained by the genetic algorithm and compare the generated fronts with those produced by the exact method. To achieve this, we utilized the several metrics outlined in Section 4.5.1, which measure different properties of these sets of non-dominated solutions. The objective is to demonstrate that the proposed metaheuristic performs effectively across all instance sizes, particularly for large instances where exact methods fail, making it the only viable alternative.

As in the calibration phase, the same stopping criterion is adopted for the J20 evaluation set, with a maximum of 15 million evaluations per run. Based on various tests performed, we find this limit to be a reasonable compromise between computation time and the quality of the results, hence it remains unchanged for both 50-activity and 100-activity instances. It is important to note that our problem formulation introduces a significantly higher level of complexity than the classical RCPSP, due to the inclusion of multiple modes, time-dependent costs, and, most notably, a multi-objective approach. Despite this, the total execution time required by our method remains substantially lower than that needed by AUGMECON. Likewise, we conduct three independent runs per instance, and all reported results represent the average values across these runs. Complete tables detailing all executions and the performance indicator values for every instance are provided in Appendix B.

Results for J20 dataset

We begin by discussing the results obtained for the J20 evaluation set. Table 4.3 presents the average values of all performance indicators considered in this study, calculated across the 166 instances in the evaluation set. Mean values are provided for both the exact method and the genetic algorithm. Note that the ϵ-indicator has been excluded from this table and the subsequent ones, as convergence is already captured by the hypervolume indicator, and the ϵ-indicator does not offer additional qualitative

insights into the produced fronts due to its consistently small and similar values. Further details on the ϵ-indicator values can be found in Appendix B.

	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	Time (h)
AUGMECON	39.49	0.03%	1.392	0.235	0.169	0.835	1.495
Genetic Algorithm	36.78	90.20%	1.292	0.198	0.153	0.812	0.166

Table 4.3: Average results for all instances in the J20 evaluation dataset.

As mentioned earlier, the metaheuristic is limited to 15 million evaluations per instance, while the AUGMECON method is constrained to 150 breakpoints and 30 minutes per breakpoint. On average, the CPU time required by the exact method is approximately nine times greater than that of the metaheuristic. Specifically, the genetic algorithm takes an average of about 10 minutes to solve the 20-activity instances, whereas the exact method requires approximately 1 hour and 30 minutes per instance. Lastly, we observe from the results in Table B1 that AUGMECON is unable to prove optimality within the time limit in 29 out of the 166 instances in the set.

A closer examination of the performance indicator values reveals that the OVNG—the average number of non-dominated points in the generated fronts—is similar for both methods. Despite differences in computational time, AUGMECON achieves an average OVNG of nearly 40 points, while the genetic algorithm yields approximately 37 points. Regarding the C-metric, we analyze it in both directions. Let Y^A represent the PF approximation (or the exact PF) obtained by the exact method, and Y^{GA} represent the one obtained by the genetic algorithm. In the first row, we compute the fraction of solutions from Y^A that are dominated by one or more solutions in Y^{GA}, denoted as $C(Y^{GA}, Y^A)$. In the second row, we compute the reverse relationship, $C(Y^A, Y^{GA})$. In the first scenario, the value is very low, equal to 0.03%. However, this value being greater than zero indicates that not all points found by the exact method are Pareto-optimal. Specifically, it implies that some solutions obtained by the genetic algorithm dominate at least one solution generated by the exact method. Examining Table B1 closely, we see that this situation arises for instance J2013_3. On the other hand, the average value of $C(Y^A, Y^{GA})$ is less than 100%—around 90%—which suggests that some solutions found by the metaheuristic are not dominated by the exact method's solutions. This indicates that the genetic algorithm performs well, even for these small-sized instances where the exact method is capable of finding the PF in most cases.

Next, we present three columns corresponding to Zitzler's $\mathcal{M}_3^*$ metric, the Γ indicator, and their ratio, the μ-indicator. Due to normalization, the maximum possible value for the $\mathcal{M}_3^*$ metric is $\sqrt{2} \approx 1.414$, which represents the maximum extent of the front, i.e., the best possible value. On average, the genetic algorithm achieves a value of 1.292 for this metric, which is quite close to the value obtained by AUGMECON, 1.392. In contrast, the metaheuristic outperforms AUGMECON in terms of the Γ-indicator, which measures the size of the holes in the front. As a result, the genetic algorithm achieves a better average value for the μ-indicator, which is 0.153. The exact method, on the other hand, achieves an average μ-value of 0.169. Although both values are very close, this indicates that the genetic algorithm produces better-spread approximations of the PF than the exact method.

Finally, the convergence and distribution properties are evaluated using the hypervolume indicator, which measures the volume of the objective space dominated by the considered front. The values are normalized, with the maximum possible value being 1, representing 100% of the total volume. The average results for both the exact method and the metaheuristic are very similar, each dominating over 80% of the total volume of the objective space.

To conclude the discussion of the J20 evaluation set, we present in Table 4.4 the average values of the performance indicators for those instances where AUGMECON is unable to obtain the exact PF within the established time limit. Without going into too much detail, we observe that the CPU time required by the exact method is now around 6 hours, while the genetic algorithm still requires an average of about 10 minutes. Regarding solution quality, the conclusions remain consistent with earlier observations. Precisely, we are now considering a total of 29 instances, which increases the value of the C-metric in the first row of the table, i.e., when measuring the coverage of the front produced by AUGMECON by the one provided by the metaheuristic. As for the average value of OVNG, it worsens for both algorithms, indicating that these problem instances pose a significant challenge.

	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	Time (h)
AUGMECON	35.52	0.17%	1.378	0.254	0.184	0.839	6.210
Genetic Algorithm	31.55	92.71%	1.264	0.206	0.165	0.797	0.163

Table 4.4: Average results for the instances in the J20 evaluation dataset where AUGMECON could not successfully solve all the MILP problems.

Overall, although the results remain similar, the substantial difference in computational effort leads us to conclude that the PF approximations provided by the genetic algorithm are quite good and that it can be considered a viable alternative to the exact method for solving this particular problem, even for the smallest instances considered. Figure 4.7 illustrates the PF approximations produced by both algorithms for two selected instances from the J20 evaluation dataset. As mentioned earlier, it is evident from the figure that both fronts are very similar, confirming that the metaheuristic demonstrates strong performance. Notably, AUGMECON obtains the exact front for J2062_3 (left), while for J2039_3 (right), it does not.

Results for MMLIB50 and MMLIB100 datasets

We now present the results for larger instances, starting with those containing 50 activities and later considering those with 100 activities, where the exact method begins to show poor performance with significantly higher computational times. Within the MMLIB50 evaluation dataset, which consists of 54 instances, AUGMECON was unable to compute the exact PF for any instance. Note that the number of breakpoints was reduced to 100, while the time limit was increased to 1 hour per MILP problem. Hence, the complexity increases significantly when considering 50 activities.

From the results in Table 4.5, we can observe a substantial difference in the CPU times required by both algorithms. AUGMECON requires an average of approximately 26 hours to generate the

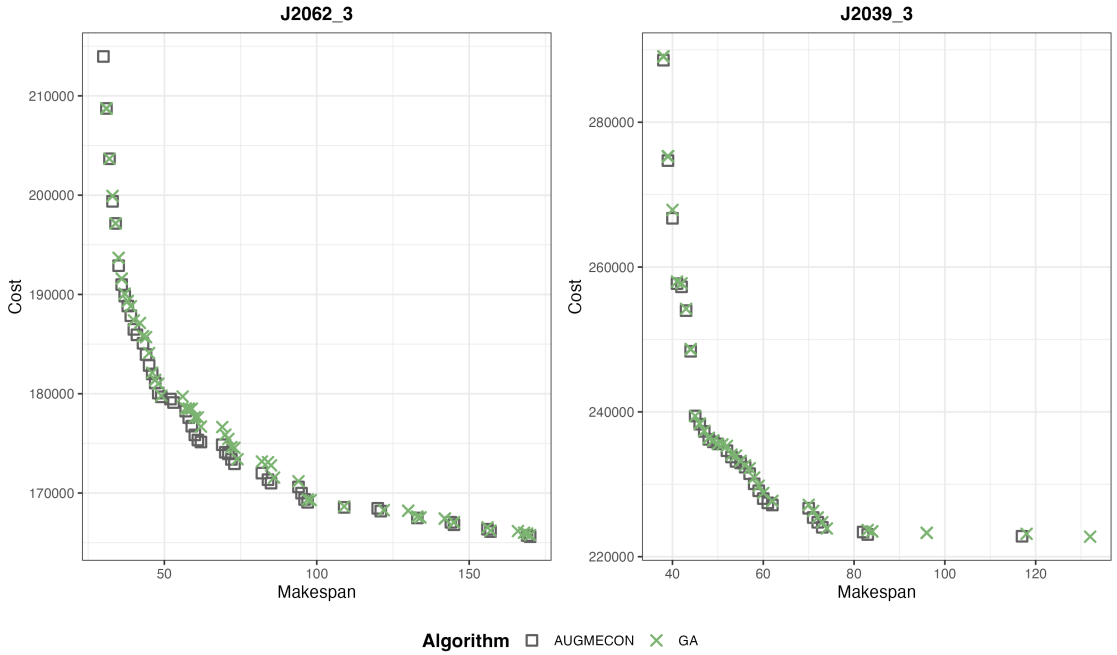


Figure 4.7: PF approximations generated by the exact method and the metaheuristic for two selected instances from the J20 evaluation dataset.

PF approximation for each instance, while the metaheuristic achieves this in less than 30 minutes on average. Additionally, the number of non-dominated solutions found also differs considerably. The exact method attains an average OVNG of about 30 points, whereas the genetic algorithm achieves nearly 62 points on average. This highlights the efficiency of the proposed algorithm, which can identify twice as many non-dominated solutions in less than 2% of the computational effort.

	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	Time (h)
AUGMECON	30.46	1.05%	1.29	0.253	0.196	0.814	26.237
Genetic Algorithm	61.77	83.17%	1.228	0.136	0.111	0.778	0.466

Table 4.5: Average results for all instances in the MMLIB50 evaluation dataset.

Regarding the C-metric, we observe that $C(Y^{GA}, Y^{A})$ exceeds 1%, showing an improvement over the J20 dataset. A closer look at Table B2 reveals that this percentage is nonzero for six instances, with notable values for J5074_1 (16.67%), J5076_1 (9.52%), J5080_1 (13.33%), and J5082_1 (9.52%). In contrast, the average $C(Y^{A}, Y^{GA})$ drops to approximately 83%, with instances such as J5086_1 and J5098_1 falling below 30%. These findings underscore that, in certain scenarios, the proposed metaheuristic is capable of producing solutions that dominate those generated by AUGMECON while simultaneously reducing the percentage of points dominated by the exact method's PF approximation.

Regarding the $\mathcal{M}_3^*$ metric, both algorithms show a very similar performance, 1.29 for the exact

method and 1.228 for the metaheuristic. However, the metaheuristic outperforms the exact method in both the Γ and μ indicators. Furthermore, the metaheuristic shows a substantial improvement in the Γ indicator, with an average value about 47% lower than that of the exact method. This contrasts with the 16% reduction observed in the J20 dataset, indicating a marked improvement in this larger-scale instances. Finally, the average hypervolume values are similar, with AUGMECON slightly outperforming the metaheuristic. This is likely due to the exact algorithm's ability to produce broader PF approximations, as reflected by the $\mathcal{M}_3^*$ metric—which often reaches a maximum value of $\sqrt{2} \approx 1.414$—thereby covering a larger portion of the objective space. Nevertheless, on average, the metaheuristic outperforms the exact method in terms of the number of non-dominated solutions and in a computation time reduced by a factor of 50.

Lastly, we present the results for the MMLIB100 evaluation dataset, which consists of 10 instances each with 100 activities. Although AUGMECON is constrained to 50 breakpoints and a 2-hour time limit per MILP problem, no exact PF was found for any of the instances. Table 4.6 shows the average results obtained in this final experimental study. For these large-scale instances, the exact method demonstrates notably poor performance compared to the metaheuristic. AUGMECON requires an average CPU time of nearly 74 hours, while the genetic algorithm only takes about 1 hour and 45 minutes, making our algorithm 42.4 times faster than the exact method.

	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	Time (h)
AUGMECON	2.40	0.00%	0.655	0.422	0.562	0.305	73.675
Genetic Algorithm	74.63	22.51%	1.203	0.155	0.129	0.700	1.737

Table 4.6: Average results for all instances in the MMLIB100 evaluation dataset.

The average OVNG for AUGMECON is around 2 points, whereas the metaheuristic identifies nearly 75 non-dominated solutions on average. Although the solutions produced by the exact method are not dominated, that is, the C-metric is 0% in the first row of the table, $C(Y^A, Y^{GA})$ drops to about 23%. This indicates that on average, only 23% of the solutions generated by the metaheuristic are dominated by those found by the exact method, highlighting the great performance of our genetic algorithm in terms of cardinality. Regarding distribution and spread properties, the fronts generated by AUGMECON are narrow, as indicated by the low average $\mathcal{M}_3^*$ metric, and exhibit a high average value of Γ, which is larger than in previous datasets. Consequently, the μ-indicator increases significantly for the exact algorithm, with the metaheuristic achieving a value approximately 77.04% lower. The metaheuristic outperforms the exact method in terms of HV, achieving an average value of 0.7 compared to 0.305, respectively. As seen in Table B3, AUGMECON identifies only one solution in 6 out of the 10 instances, making it impossible to compute certain performance indicator values for these cases.

To conclude the analysis, Figure 4.8 displays the PF approximations obtained by both AUGMECON and the metaheuristic for two selected instances: J5092_1 from the MMLIB50 dataset and J10087_1 from the MMLIB100 dataset. These examples illustrate the observations discussed above. Notably, the metaheuristic produces a larger number of non-dominated solutions, and these points are

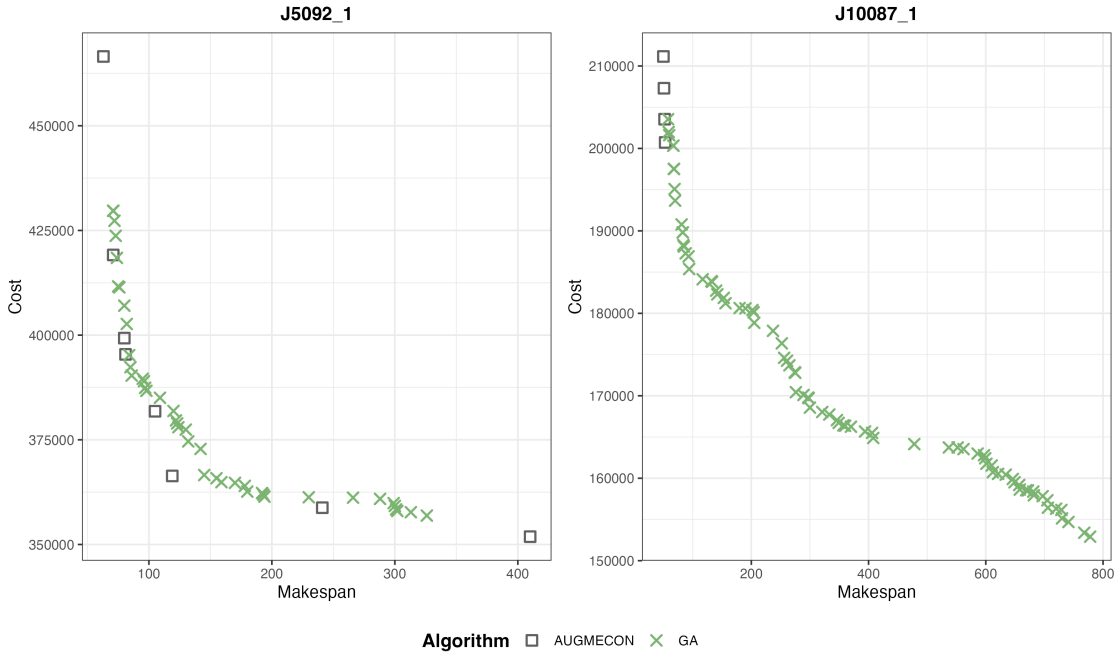


Figure 4.8: PF approximations generated by the exact method and the metaheuristic for two selected instances from the MMLIB50 and MMLIB100 datasets, respectively.

well-distributed throughout the objective space with fewer gaps between them. For J5092_1, while AUGMECON provides a good PF approximation in terms of the $\mathcal{M}_3^*$ metric, the points are more sparsely spread. In contrast, for J10087_1, the extent of the front is noticeably inferior. Overall, both graphs show how the metaheuristic generates fronts with much better properties than the exact method employing a drastically reduced computational effort.

4.6 Final remarks and future work

In this chapter, we introduce the bi-objective MRCPSP_TDRCC, which incorporates time-dependent resource costs and capacities. The objective is to simultaneously minimize both the makespan and the total cost of resource usage. To tackle this challenging optimization problem, we propose a mathematical formulation and explore an exact method based on the AUGMECON technique. For small instances, this exact algorithm can either obtain the set of Pareto-optimal solutions or provide a good approximation. However, through extensive experimentation, we demonstrate that this method struggles to achieve optimality within a reasonable time for larger problem sizes, specifically for problems with more than 20 activities.

To address this issue, we design a tailored genetic algorithm based on the NSGA-II framework, which effectively approximates the PF for this complex problem. Results, supported by rigorous performance quality indicators, show that our metaheuristic consistently outperforms the exact approach in providing high-quality approximations of the PF for medium- and large-sized instances, requiring

significantly less CPU time. Our computational experiments reveal that for instances with 20 activities, the metaheuristic requires approximately nine times less CPU time than the exact method, which fails to solve 29 out of the 166 instances to optimality within the time limit. For instances with 50 and 100 activities, the differences in CPU time are even more pronounced: AUGMECON requires, on average, nearly 26 hours per instance for MMLIB50 and 74 hours for MMLIB100. In contrast, the genetic algorithm completes its execution in less than 30 minutes for the 50-activity instances and under 2 hours for the largest ones. Moreover, the exact method is unable to compute the complete PF within the time limit for any instance in these two datasets.

Notably, our metaheuristic is capable of generating a wide range of non-dominated solutions, effectively covering a large portion of the objective space and providing a sharp approximation of the PF obtained by AUGMECON—even for small-sized instances where the exact algorithm is able to compute the complete PF in most cases. For the 20-activity instances, both methods achieve very similar values across several performance indicators such as OVNG, HV, and μ. For medium- and large-scale instances, the metaheuristic demonstrates a clear advantage. In particular, for MMLIB50 and MMLIB100, the genetic algorithm significantly outperforms the exact method in terms of OVNG—producing roughly twice as many non-dominated solutions for the 50-activity instances and a substantially higher number for the largest ones, where the exact method yields almost none. Additionally, the metaheuristic achieves superior results in terms of C, indicating that it consistently covers a larger portion of the PF and identifies a more diverse set of trade-off solutions. Given the inherent complexity of solving an NP-complete problem and its multi-objective nature, achieving an ideal trade-off between both objectives is unrealistic, as many compromise solutions are possible. Nonetheless, the proposed metaheuristic offers decision-makers a high-quality set of alternative solutions with a reasonable computational effort—a task that would be far from feasible using any exact technique in a real-world scenario.

Several promising avenues for future research remain. One natural extension is to compare the performance of different metaheuristics to further evaluate their suitability for this problem variant. Additionally, exploring alternative solution encodings or genetic operators could improve solution diversity and quality. Future work could also explore objective functions more closely aligned with specific industrial goals—such as minimizing delay penalties or early completion bonuses—increasing the model's applicability. Lastly, incorporating real-world project characteristics, such as uncertainty in processing times or cost structures, could lead to more practical and applicable problem variants.

Chapter 5

A robust optimization approach for scheduling with uncertain start-time dependent costs

Modern scheduling problems often require accounting for fluctuating operational costs and inherent uncertainty. This chapter introduces a novel robust optimization framework designed to tackle start-time dependent costs under uncertainty, extending classical scheduling models to better reflect real-world decision-making environments.

5.1 Motivation and structure

This chapter addresses a scheduling problem where costs vary over time and are subject to uncertainty. Building on the classical project scheduling problem (PSP) with start-time dependent costs, we propose a novel extension that incorporates execution-time dependent costs—where the cost of an activity varies throughout its processing time. This new definition better reflects real-world scenarios such as energy consumption or labor costs that fluctuate throughout the day. For example, running machinery during peak energy hours incurs higher costs, while scheduling labor during overtime hours results in increased wages. These execution-time costs are aggregated into start-time dependent costs, maintaining model tractability while enriching its realism.

Unlike the classical PSP with start-time dependent costs, which admits a polynomial-time solution (Möhring et al., 2003), our formulation introduces additional complexity by prohibiting parallel execution and removing precedence constraints from the project structure to create a more dynamic and flexible environment in which activities can be reordered based on cost efficiency or other operational criteria. As a result, we prove that our problem is NP-hard, even before incorporating uncertainty.

Note that the problem we are considering might be seen as a special case of the resource-constrained PSP with a unit capacity-resource (see, e.g.. Van Cauwelaert et al., 2020; Riedler et al., 2020; Liu et al., 2023) or even as a single-machine scheduling problem, where the sequence of activities must be determined (see, e.g., Baker, 1974; Pinedo, 2016; Shabtay, 2023). Unlike traditional single-machine

scheduling, which typically aims to minimize lateness costs or maximize earliness profits, our problem focuses on minimizing the total start-time dependent costs. These costs vary based on when an activity begins and are commonly found in project scheduling problems. Their impact is significant, as they influence one of the key objectives in project scheduling alongside makespan, namely the minimization of total cost. Therefore, we believe that the most appropriate context for addressing this problem is project scheduling rather than machine scheduling.

Since real-world scenarios are inherently uncertain, a deterministic approach may lead to suboptimal or unreliable schedules. Thus, we introduce uncertainty into the cost parameters of our model and address it using a robust approach (see Section 2.3). Following Bertsimas and Sim (2004), we consider budgeted uncertainty sets and propose a two-stage robust optimization formulation: in the first stage, a baseline schedule is established by determining the order in which activities should be executed. In the second stage, the actual cost values are realized, and the final schedule is computed, i.e., we assign specific starting times to each activity, while respecting the sequence determined in the first stage. The objective is to minimize the total cost of the resulting schedule, accounting for all potential first-stage activity permutations and potential adversarial scenarios. Additionally, we distinguish between two types of budgeted uncertainty sets: a discrete variant, which considers a finite set of cost scenarios, and a continuous variant, which allows convex combinations of such realizations.

This two-stage formulation arises whenever a sequence of activities must be determined in advance, while their specific starting times allow for some flexibility. As an example, in airport scheduling (Ikli et al., 2021), aircraft physically line up at the departure runway, which determines their departure order. The specific departure times, however, allow for some flexibility to adjust to the current circumstances. Similarly, consultation times—such as scheduled appointments with medical professionals—can often be delayed, but customers observe their order of arrival and consider changes in this sequence as unfair.

This chapter, drawing on the research by Rodríguez-Ballesteros et al. (2025b), delivers the following contributions. First, it introduces a novel NP-hard deterministic scheduling model with execution-time dependent costs and no precedence constraints. Second, it presents a compact robust optimization formulation for continuous budgeted uncertainty and an iterative solution procedure inspired by Zeng and Zhao (2013) for discrete budgeted uncertainty. Finally, it presents a comprehensive computational study demonstrating the efficiency of the compact model, which outperforms the iterative approach in speed and solution quality under both uncertainty settings.

The chapter is organized as follows. Section 5.2 introduces the deterministic model and robust framework. Section 5.3 presents the compact formulation for continuous budgeted uncertainty, while Section 5.4 details the iterative method for discrete scenarios. Section 5.5 provides the experimental analysis and discusses the results. Finally, concluding remarks are presented in Section 5.6.

5.2 Problem description

This section formally introduces the scheduling problem addressed in this chapter. We begin by describing the nominal version of the problem, which extends the classical PSP with start-time dependent

costs by removing precedence constraints and allowing activity costs to vary over their processing time. We also analyze the computational complexity of this novel problem. Next, we incorporate uncertainty into the cost structure and propose a two-stage robust optimization approach to address the resulting problem.

5.2.1 Nominal problem

We first recall the closely related PSP with start-time dependent costs as studied in Möhring et al. (2001); Möhring et al. (2003). We are given a finite time horizon $\mathcal{T} = \{0, \dots, T-1\}$ and a set of activities J. Each activity $j \in J$ has an integer duration d_j. Furthermore, we are given a time-activity start-time cost matrix $\mathbf{c}$, where each entry c_{jt} denotes the cost incurred by starting activity $j \in J$ at time $t \in \mathcal{T}$. Finally, we are also given a set of precedence relations $P \subseteq J \times J$, where $(i, j) \in P$ means that activity i must be scheduled before activity j. The task is to find a starting time for each activity, so that the total sum of start-time dependent execution times is minimized, where no preemption of activities is allowed. Note that this formulation assumes that the entire cost matrix $\mathbf{c}$ is part of the input, and hence T is polynomial in the input size. This differs from most other scheduling problems, where the time horizon can be exponential in the input size. In the original definition, activities can run in parallel, if this is not forbidden by a precedence constraint.

Following the notation introduced in Pritsker et al. (1969), the time-indexed 0/1-variables can be defined as

$$x_{jt} = \begin{cases} 1 & \text{if activity } j \text{ starts at time } t, \\ 0 & \text{otherwise,} \end{cases}$$

$\forall j \in J$ and $t \in \mathcal{T}$. The PSP with start-time dependent costs can be formulated as the following integer linear program:

$$\text{minimize} \quad \sum_{j \in J} \sum_{t \in \mathcal{T}} c_{jt} x_{jt}, \tag{5.1}$$

$$\text{subject to} \quad \sum_{t \in \mathcal{T}} x_{jt} = 1, \qquad \forall j \in J, \tag{5.2}$$

$$\sum_{s=t}^{T-1} x_{is} + \sum_{s=0}^{t+d_i-1} x_{js} \leq 1, \quad \forall (i, j) \in P,\ t \in \mathcal{T}, \tag{5.3}$$

$$x_{jt} \in \{0, 1\}, \qquad \forall j \in J,\ t \in \mathcal{T}. \tag{5.4}$$

The objective in (5.1) is to minimize the total cost. Constraints (5.2) ensure each activity is scheduled exactly once. Constraints (5.3) refer to the precedence relationship, ensuring no activity starts before its predecessors have been completed. Finally, constriants (5.4) restrict the decision variables to the subset $\{0, 1\}$. Note that in this formulation, an activity may start late in the time horizon, such that the sum of its start time and duration exceeds $T - 1$. For ease of notation, we allow this possibility. If undesired, additional constraints can be included to impose bounds on the earliest and latest starting times of each activity.

The complexity of this problem is quite well-understood. In fact, it can be solved as a linear programming (LP) problem, as the following lemma states (Chaudhuri et al., 1994).

Lemma 8. The polyhedron of problem (5.1)-(5.4) is integral.

This property states that it is possible to relax the integrality constraints (5.4) to obtain an LP formulation that has an optimal integral solution. In particular, Möhring et al. (2003) show that the problem can be solved in strongly polynomial time via a minimum cut formulation.

As commented above, our nominal problem takes as its basis the PSP with start-time dependent costs. An extension of this problem arises when we assume that certain costs w_{jt} must be paid whenever activity j is active at time t, referred to as execution-time dependent costs. In our setting, we begin with execution-time dependent costs, as they capture more realistic scenarios such as fluctuating energy or labor costs throughout the day. However, for modeling purposes, we recover the traditional PSP formulation by defining the total cost for starting activity j at time t as $c_{jt} = \sum_{s=t}^{t+d_j-1} w_{js}$, effectively transforming the problem into one with start-time dependent costs. This transformation allows us to use c_{jt} as the input cost parameters in our formulation, while preserving the problem's structure and computational complexity.

Furthermore, in our model, we remove all precedence constraints from the initial input and do not allow activities to be executed in parallel. That is, for every pair of distinct activities $i \neq j$, either $(i,j) \in P$ or $(j,i) \in P$, and P is acyclic; thus, the execution order is completely fixed. This setting reflects practical situations in project planning where activities are not governed by strict logical dependencies and must be executed sequentially due to resource or capacity limitations. In such contexts, tasks cannot run in parallel and must be arranged. Moreover, this structure allows for greater flexibility in reordering activities to optimize cost-related objectives. Given this setting, the nominal problem is defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{j\in J}\sum_{t\in\mathcal{T}} c_{jt}x_{jt} \\ \text{subject to} \quad & d_j x_{jt} + \sum_{j\neq i\in J}\sum_{s=t}^{t+d_j-1} x_{is} \leq d_j, \quad \forall j \in J, \quad t \in \mathcal{T}, \\ & (5.2), \text{and } (5.4). \end{aligned} \tag{5.5}$$

Constraints (5.5) ensure that once an activity is scheduled, it must be completed before another can begin, without requiring an explicit activity ordering. The only requirement is that no two activities can overlap in execution. Note that this formulation resembles clique-based time-indexed models used in single-machine scheduling, where non-overlapping time intervals form cliques (see, e.g., Sourd (2009) for exact algorithms in earliness-tardiness scheduling). However, unlike traditional approaches that focus on earliness-tardiness penalties, our model explicitly minimizes the sum of start-time dependent costs, adding a unique dimension to the problem.

At this point, we question if this new problem remains solvable in polynomial time. The following result demonstrates the complexity of the nominal problem.

Proposition 1. The non-preemptive scheduling problem with execution-time dependent costs and no precedence constraints is NP-hard and not approximable in polynomial time.

Proof. Let an instance of the strongly NP-hard 3-partition problem be given. It consists of a list S of

integers $a_1, \ldots, a_{3m}$ with $A = \sum_{i=1}^{3m} a_i/m$ and $A/4 < a_i < A/2$ for all $i \in \{1, \ldots, 3m\}$. The task is to find a partition into m triplets that all have the same sum.

We construct an instance of our scheduling problem as follows. We define activities $J = \{j_1, j_2, \ldots, j_{3m}\}$ with durations $d_j = a_j$. The time horizon is defined as $T = \sum_{j=1}^{3m} d_j + m$. Finally, the execution-time dependent costs are defined as

$$w_{jt} = \begin{cases} 1, & \text{if } t = n(A+1),\ n \in \mathbb{N}_{\geq 0} \\ 0, & \text{otherwise} \end{cases}$$

Figure 5.1 presents an example for $S = (3, 3, 3, 3, 4, 4)$. We have $m = 2$ and $A = 10$. Each square represents a time slot. Black squares are those where $w_{jt} = 1$. An optimal solution is to start activities j_1, j_2, and j_5 in the first 10 squares with costs zero, and start activities j_3, j_4, and j_6 in the remaining 10 squares with costs zero.

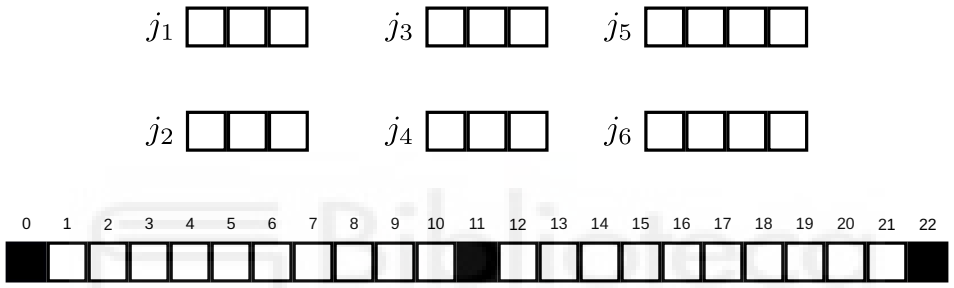


Figure 5.1: Example reduction for the proof of Proposition 1.

It follows that a schedule with costs 0 exists if and only if it is possible to partition activities into m sets $J_1, \ldots, J_m$ with $\sum_{j \in J_i} d_j = A$ for each i. It remains to show that this reduction is of polynomial time and space in the input size. Note that there are $3m \cdot (T+1)$ cost coefficients that need to be defined, where T is not polynomial in the binary encoding size of the input values a_i. However, as the 3-partition problem is strongly NP-hard, we may assume each a_i to be encoded in unary, which means that the reduction remains polynomial.

Finally, the inapproximability claim follows from the fact that the 3-partition instance is a yes-instance, if and only if a solution to the scheduling instance exists with costs 0. □

As a direct consequence of Proposition 1, we see that the problem (5.1)-(5.2)-(5.5)-(5.4) with start-time dependent costs is NP-hard as well.

5.2.2 Robust setting

In real-world applications, the start-time dependent costs considered in the nominal problem may not be known with certainty at the time of scheduling. External factors such as fluctuating energy tariffs, uncertain labor availability, or changing operational conditions can introduce significant cost uncertainty. To address this issue, we now extend the nominal scheduling problem to a two-stage robust setting, where we assume that the cost matrix $\mathbf{c}$ is uncertain. In the first stage, we do not determine

the complete schedule (i.e., the starting time of each activity), but only the order in which activities are to be processed. Once the actual costs are revealed, the second stage assigns starting times to each activity, subject to the previously fixed ordering. Note that the nominal problem described in Section 5.2.1 can be recovered by assuming that both stages are solved simultaneously under perfect information. In this sense, the robust model extends the nominal formulation by separating sequencing and timing decisions, while incorporating uncertainty in execution-time costs.

More formally, let y_{ij} be a binary variable that models if activity i precedes activity j. To obtain a complete ordering, we use the constraints

$$y_{ij} + y_{ji} = 1, \qquad \forall i, j \in J, \tag{5.6}$$

$$y_{ij} + y_{jk} \leq 1 + y_{ik}, \qquad \forall i, j, k \in J. \tag{5.7}$$

Let $\mathcal{Y} = \{\mathbf{y} \in \{0,1\}^{J \times J} : (5.6), (5.7)\}$. Furthermore, let $P(\mathbf{y}) = \{(i,j) \in J \times J : y_{ij} = 1\}$ for a given $\mathbf{y} \in \mathcal{Y}$. The set of schedules that adhere to the precedence relations implied by $\mathbf{y}$ is then defined as follows:

$$\mathcal{X}(\mathbf{y}) = \{\mathbf{x} \in \{0,1\}^{J \times \mathcal{T}} : \sum_{t \in \mathcal{T}} x_{jt} = 1, \qquad \forall j \in J,$$

$$\sum_{s=t}^{T-1} x_{is} + \sum_{s=0}^{t+d_i-1} x_{js} \leq 1, \qquad \forall (i,j) \in P(\mathbf{y}),\ t \in \mathcal{T}\}.$$

Let $\mathcal{U}$ be an uncertainty set, representing all possible cost scenarios $\mathbf{c}$ against which we aim to immunize the solution. The two-stage robust optimization approach that we study is to solve

$$\min_{\mathbf{y} \in \mathcal{Y}} \max_{\mathbf{c} \in \mathcal{U}} \min_{\mathbf{x} \in \mathcal{X}(\mathbf{y})} \sum_{j \in J} \sum_{t \in \mathcal{T}} c_{jt} x_{jt}$$

Note that if $\mathcal{U}$ consists of a single scenario, we recover the NP-hard nominal problem.

To define the uncertainty set, we follow the approach of budgeted uncertainty (Bertsimas and Sim, 2004). We assume that there is an interval $[\underline{c}_{jt}, \underline{c}_{jt} + \hat{c}_{jt}]$ of possible realizations for the cost of starting activity j at time t. We further assume that there is a bound Γ on the total sum of deviations. That is, we define the continuous budgeted uncertainty set to be

$$\mathcal{U}_C(\Gamma) = \left\{\mathbf{c} \in \mathbb{R}^{J \times \mathcal{T}} : c_{jt} = \underline{c}_{jt} + \hat{c}_{jt}\delta_{jt},\ \delta_{jt} \in [0,1],\ \sum_{t \in \mathcal{T}} \sum_{j \in J} \delta_{jt} \leq \Gamma\right\}, \tag{5.8}$$

and the discrete budgeted uncertainty set as

$$\mathcal{U}_D(\Gamma) = \left\{\mathbf{c} \in \mathbb{R}^{J \times \mathcal{T}} : c_{jt} = \underline{c}_{jt} + \hat{c}_{jt}\delta_{jt},\ \delta_{jt} \in \{0,1\},\ \sum_{t \in \mathcal{T}} \sum_{j \in J} \delta_{jt} \leq \Gamma\right\}. \tag{5.9}$$

If Γ is an integer, it is well-known that both uncertainty sets are equivalent for one-stage robust problems. This is not the case for two-stage problems: in the continuous version, the adversary can use fractional deviations, which may lead to worse outcomes for the decision-maker. As a result, the two uncertainty sets are no longer equivalent in general (Goerigk and Hartisch, 2024).

We close this section with a discussion of extensions of this setting. The "attack" possibilities for an adversary in the above definition (i.e., the decision where to increase costs) are restricted to

combinations of items and time slots. For example, it does not allow the adversary to perform one cost increase that affects all activities starting at a specific time, or all starting times of a specific activity. To this end, we can use scenarios of the form

$$c_{jt} = \underline{c}_{jt} + \hat{c}_{jt}\delta_{jt} + \tilde{c}_t\tilde{\delta}_t + \bar{c}_j\bar{\delta}_j,$$

with $\sum_{j\in J}\sum_{t\in\mathcal{T}}\delta_{jt} + \sum_{t\in\mathcal{T}}\tilde{\delta}_t + \sum_{j\in J}\bar{\delta}_j \leq \Gamma$. Furthermore, it can also be extended to execution-time dependent uncertainty, by using

$$c_{jt} = \sum_{s=t}^{t+d_j-1}\left(\underline{w}_{jt} + \hat{w}_{jt}\delta_{jt} + \tilde{w}_t\tilde{\delta}_t + \bar{w}_j\bar{\delta}_j\right).$$

To avoid additional notation, this chapter focuses on the base case defined by $\mathcal{U}_C(\Gamma)$ and $\mathcal{U}_D(\Gamma)$, although the approach could be extended to more general settings.

5.3 Compact formulation for continuous budgeted uncertainty

In the case of continuous budgeted uncertainty, we introduce a compact formulation that integrates scheduling decisions and worst-case cost realizations into a unified optimization model. This is done by dualizing the inner minimization problem and embedding the uncertainty set directly into the formulation. We begin by analyzing the inner problem:

$$\min_{\mathbf{x}\in\mathcal{X}(\mathbf{y})}\sum_{j\in J}\sum_{t\in\mathcal{T}}c_{jt}x_{jt},$$

for given $\mathbf{c}$ and $\mathbf{y}$. This is equivalent to problem (5.1)-(5.4), and can therefore be formulated as an LP problem. By taking its dual, we obtain the following model:

$$\text{maximize} \quad \sum_{j\in J}\alpha_j - \sum_{(i,j)\in P(\mathbf{y})}\sum_{t\in\mathcal{T}}\gamma_{(i,j),t} \tag{5.10}$$

$$\text{subject to} \quad \alpha_j - \sum_{\substack{(j,r)\in P(\mathbf{y})\\ r\in J}}\sum_{s=0}^{t}\gamma_{(j,r),s} - \sum_{\substack{(r,j)\in P(\mathbf{y})\\ r\in J}}\sum_{s=\max\{0,t-d_r+1\}}^{T-1}\gamma_{(r,j),s} \leq c_{jt}, \quad \forall j\in J,\ t\in\mathcal{T}, \tag{5.11}$$

$$\gamma_{(i,j),t} \geq 0 \quad \forall (i,j)\in P(\mathbf{y}),\ t\in\mathcal{T} \tag{5.12}$$

$$\alpha_j \text{ free} \quad \forall j\in J \tag{5.13}$$

To build the adversarial problem, we need to incorporate the optimization over the uncertainty set into the previous model. Precisely, we introduce the adversarial cost increase variables, δ_{jt}, $j\in J$,

$t \in \mathcal{T}$, obtaining the following LP formulation:

$$\text{maximize} \quad \sum_{j\in J} \alpha_j - \sum_{(i,j)\in P(\mathbf{y})} \sum_{t\in\mathcal{T}} \gamma_{(i,j),t} \tag{5.14}$$

$$\text{subject to} \quad \alpha_j - \sum_{\substack{(j,r)\in P(\mathbf{y})\\ r\in J}} \sum_{s=0}^{t} \gamma_{(j,r),s} - \sum_{\substack{(r,j)\in P(\mathbf{y})\\ r\in J}} \sum_{s=\max\{0,t-d_r+1\}}^{T-1} \gamma_{(r,j),s} \leq \underline{c}_{jt} + \hat{c}_{jt}\delta_{jt}, \quad \forall j \in J,\ t \in \mathcal{T}, \tag{5.15}$$

$$\sum_{t\in\mathcal{T}} \sum_{j\in J} \delta_{jt} \leq \Gamma, \tag{5.16}$$

$$0 \leq \delta_{jt} \leq 1, \quad \forall j \in J,\ t \in \mathcal{T} \tag{5.17}$$

$$\gamma_{(i,j),t} \geq 0 \quad \forall (i,j) \in P(\mathbf{y}),\ t \in \mathcal{T} \tag{5.18}$$

$$\alpha_j \text{ free} \quad \forall j \in J \tag{5.19}$$

Note that the dual variables α_j, $\gamma_{(i,j),t}$, and the cost increase variables δ_{jt} are continuous. Thus, the adversarial problem remains an LP problem. By dualizing constraints (5.14)-(5.19) and integrating the result with the first-stage problem via the precedence variables $\mathbf{y} \in \mathcal{Y}$, we derive the following compact mixed-integer linear programming formulation for the two-stage robust scheduling problem with continuous budgeted uncertain start-time dependent costs:

$$\text{minimize} \quad \sum_{j\in J} \sum_{t\in\mathcal{T}} \underline{c}_{jt} x_{jt} + \Gamma\pi + \sum_{j\in J} \sum_{t\in\mathcal{T}} \eta_{jt} \tag{5.20}$$

$$\text{subject to} \quad \sum_{t\in\mathcal{T}} x_{jt} = 1, \quad \forall j \in J \tag{5.21}$$

$$\sum_{s=t}^{T-1} x_{is} + \sum_{s=0}^{t+d_i-1} x_{js} \leq 1 + (1 - y_{ij}), \quad \forall i,j \in J,\ t \in \mathcal{T} \tag{5.22}$$

$$\pi + \eta_{jt} \geq \hat{c}_{jt} x_{jt}, \quad \forall j \in J,\ t \in \mathcal{T} \tag{5.23}$$

$$y_{ij} + y_{ji} = 1, \quad \forall i,j \in J \tag{5.24}$$

$$y_{ij} + y_{jk} \leq (1 + y_{ik}), \quad \forall i,j,k \in J \tag{5.25}$$

$$y_{ij} \in \{0,1\} \quad \forall i,j \in J \tag{5.26}$$

$$x_{jt} \geq 0, \quad \forall j \in J,\ t \in \mathcal{T} \tag{5.27}$$

$$\eta_{jt} \geq 0, \quad \forall j \in J, t \in \mathcal{T} \tag{5.28}$$

$$\pi \geq 0. \tag{5.29}$$

where the precedence binary variable $y_{ij} = 1$ if activity i is a predecessor of activity j; otherwise, $y_{ij} = 0$. Consequently, constraint (5.22) is only activated when activity i precedes activity j. The resulting compact formulation can therefore be interpreted as an extended total cost minimization problem, where a feasible schedule is constructed with the goal of minimizing the total cost across all possible first- and second-stage scenarios. In this setting, all activity permutations and their corresponding worst-case cost realizations are considered to determine the minimum total cost.

5.4 Iterative approach for discrete budgeted uncertainty

We now consider the case of discrete budgeted uncertainty. To formulate the adversarial problem, we follow the same approach as in the previous section. The only difference is that in formulation (5.14)-(5.19), the variables δ_{jt} are now binary instead of continuous. The resulting adversarial problem is as follows:

$$\text{maximize} \quad \sum_{j\in J}\alpha_j - \sum_{(i,j)\in P(\mathbf{y})}\sum_{t\in\mathcal{T}}\gamma_{(i,j),t} \tag{5.30}$$

$$\text{subject to} \quad \alpha_j - \sum_{\substack{(j,r)\in P(\mathbf{y})\\ r\in J}}\sum_{s=0}^{t}\gamma_{(j,r),s} - \sum_{\substack{(r,j)\in P(\mathbf{y})\\ r\in J}}\sum_{s=\max\{0,t-d_r+1\}}^{T-1}\gamma_{(r,j),s} \le \underline{c}_{jt} + \hat{c}_{jt}\delta_{jt}, \quad \forall j\in J,\ t\in\mathcal{T}, \tag{5.31}$$

$$\sum_{t\in\mathcal{T}}\sum_{j\in J}\delta_{jt} \le \Gamma, \tag{5.32}$$

$$\delta_{jt}\in\{0,1\}, \quad \forall j\in J,\ t\in\mathcal{T} \tag{5.33}$$

$$\gamma_{(i,j),t}\ge 0 \quad \forall (i,j)\in P(\mathbf{y}),\ t\in\mathcal{T} \tag{5.34}$$

$$\alpha_j \text{ free} \quad \forall j\in J \tag{5.35}$$

This means that we cannot use LP-duality to reach a compact problem reformulation. Instead, we propose an iterative solution method (Zeng and Zhao, 2013). For a subset $\mathcal{U}' = \{\mathbf{c}^1,\dots,\mathbf{c}^K\}\subseteq\mathcal{U}_D(\Gamma)$ of scenarios, we determine a robust solution by solving

$$\min_{\mathbf{y}\in\mathcal{Y}}\max_{k\in\mathcal{K}}\min_{\mathbf{x}\in\mathcal{X}(\mathbf{y})}\sum_{j\in J}\sum_{t\in\mathcal{T}}c^k_{jt}x_{jt},$$

where $\mathcal{K}=\{1,\dots,K\}$. This can be reformulated as the following optimization problem:

$$\text{minimize} \quad z \tag{5.36}$$

$$\text{subject to} \quad z \ge \sum_{j\in J}\sum_{t\in\mathcal{T}}c^k_{jt}x^k_{jt}, \quad \forall k\in\mathcal{K} \tag{5.37}$$

$$\sum_{t\in\mathcal{T}}x^k_{jt} = 1, \quad \forall j\in J, k\in\mathcal{K} \tag{5.38}$$

$$\sum_{s=t}^{T-1}x^k_{is} + \sum_{s=0}^{t+d_i-1}x^k_{js} \le 1+(1-y_{ij}), \quad \forall i,j\in J,\ t\in\mathcal{T}, k\in\mathcal{K} \tag{5.39}$$

$$y_{ij}+y_{ji}=1, \quad \forall i,j\in J \tag{5.40}$$

$$y_{ij}+y_{jk}\le(1+y_{ik}), \quad \forall i,j,k\in J \tag{5.41}$$

$$x^k_{jt}\in\{0,1\}, \quad \forall j\in J,\ t\in\mathcal{T}, k\in\mathcal{K} \tag{5.42}$$

$$y_{ij}\in\{0,1\} \quad \forall i,j\in J \tag{5.43}$$

Solving (5.36)-(5.43) provides us with a lower bound on the optimal value of the two-stage robust problem with respect to $\mathcal{U}_D(\Gamma)$, as only a subset of scenarios $\mathcal{U}'$ is considered. We obtain an upper

bound by solving the adversarial problem (5.30)-(5.35) for the current solution $\mathbf{y}$. If the resulting upper bound exceeds the lower bound, the corresponding scenario $\mathbf{c}$ is added to $\mathcal{U}'$, and the process is repeated. Otherwise, if the bounds coincide, the current solution is optimal. Since $\mathcal{U}_D(\Gamma)$ is finite, the algorithm converges in a finite number of iterations.

5.5 Computational results

In this section, we present computational tests conducted to evaluate the models presented in this chapter. We commence by outlining the instances used, followed by a description of the experimental setup employed for the five models under examination. We divide the experiments into three parts, based on the set of instances considered for each study, and provide a detailed analysis of the results obtained for each experiment.

5.5.1 Test data

We start by presenting the parameter settings used for generating the benchmark instances. While multiple sets of instances are available for the resource-constrained project scheduling problem (see, e.g., Kolisch and Sprecher, 1997), this does not seem to be the case for the considered problem without precedence constraints. Therefore, we need to generate our own data sets for the particular problem addressed in this chapter. Our nominal model outlined in Section 5.2.1 comprises specific elements. Firstly, akin to any scheduling problem, the primary feature involves the number of activities within each instance. For this parameter, we consider a wide range of possible values, $n \in \{5, 10, 15, 20, 25, 30\}$, thereby generating six benchmark sets (called n-sets) utilized in the first part of this study. Each set is composed of 20 instances, resulting in a total of 120 instances.

Once the size of the problems is established, the next step is to determine the duration of each activity. Activity durations d_j are assigned integer values randomly generated from a uniform distribution $U[1, 6)$. Typically, the planning horizon T is defined as the sum of the processing times of all activities within each instance. However, to regulate the tightness of this upper bound on the project makespan, we introduce a constant multiplier c (i.e., we set $T = c \cdot \sum_{j \in J} d_j$). The default value for this parameter is set to 1.2. We generate execution-time dependent costs w_{jt} using a uniform distribution $U[1, 10)$, and calculate start-time dependent costs $\underline{c}_{jt}$ as shown in Section 5.2.1. Furthermore, we choose the cost deviations $\hat{c}_{jt}$ as integer values randomly generated from a uniform distribution $U[1, 6)$. We set Γ to the default value of 4.

The parameters Γ and c are set to their default values to generate the 120 instances of the n-sets. Further experimentation on these parameters has also been conducted, where we discuss their impact on the behavior of the different models presented in this chapter. Specifically, with the number of activities fixed at $n = 15$, we explore the potential values of Γ, ranging from 1 to 20, resulting in twenty sets of 20 instances for each Γ value (called Γ-sets). Similarly, we examine c values spanning from 1.00 to 1.40, increasing by 0.05 for each step, thereby creating nine sets of 20 instances for each c value (called c-sets). Hence, we obtain 400 instances where we vary Γ, and 180 instances where we

vary c, yielding a total of 700 instances for the complete computational study.

All instances used in this chapter are available for download at the GitHub repository https://github.com/jalcarazs/Robust-optimization-uncertainty.

5.5.2 Experimental setup

We now describe the experiments conducted in order to evaluate the performance of the five models considered in this study. Concerning the nominal problem formulation (5.1)-(5.2)-(5.4)-(5.5) presented in Section 5.2.1, we examine two settings for the start-time dependent costs. The first setting maintains the nominal values for the costs, while the second setting adjusts the costs to their worst-case scenario. Recall that worst-case costs are determined as $\underline{c}_{jt} + \hat{c}_{jt}$ for all activities and time slots. We label the first model as the nominal lower-bound (LB), as it establishes a lower limit for the total cost. Similarly, we denote the second setting as the nominal upper-bound (UB), as it offers an upper limit for the total cost.

Continuing with the robust setting, we have explored two approaches depending on the uncertainty set considered. Firstly, in Section 5.3, we have presented a compact formulation (5.20)-(5.29) for the two-stage robust project scheduling with continuous budgeted uncertain start-time dependent costs. On the other hand, as the variables δ_{jt} become binary instead of continuous in Section 5.4, we cannot combine the first and second-stage problems into a single compact formulation for discrete budgeted uncertainty. We apply an iterative solution method to address this issue, which shares solutions between the adversarial problem (5.30)-(5.35) and formulation (5.36)-(5.43). Throughout the computational experiments, we refer to the first robust model as the compact model (C), while the second solution method is referred to as the iterative solution discrete model (ID). Finally, we have also implemented the iterative solution method for the continuous scenario (IC), which considers continuous variables δ_{jt}. Note that this approach and the compact model C represent the same underlying problem; thus, optimal solutions coincide.

To evaluate the five models, we have established several criteria. Our focus lies on two main goals: understanding the behavior of the algorithms amidst uncertainty, and their speed (efficiency) in reaching the optimal value. Regarding the former, we need to test the quality of the solutions generated by the aforementioned models. In the context of our two-stage scheduling problems, a solution consists in a specific sequence for project tasks, i.e., the precedence constraints within the activities. To assess the quality of such ordering, we subject these precedence constraints to our two adversarial problems: formulation (5.14)-(5.19) with continuous budgeted uncertainty ($Cont.\ BU$) and formulation (5.30)-(5.35) with discrete budgeted uncertainty ($Disc.\ BU$). In this context, lower objective values denote superior performance under uncertainty. Additionally, we utilize the nominal lower-bound model (LB) to evaluate the solutions, assessing the quality of such precedence constraints in scenarios without uncertainty. Finally, we apply different time limits on the execution of these methods. We will highlight cases where a specific configuration does not achieve the optimal value, but only provides a feasible solution for the instance being considered.

To conclude, all the models proposed in this chapter have been coded in Java and solved using

Gurobi 10.0.3. The experimental study is structured into three primary experiments. In the first part, we concentrate on the n-sets to analyze how solutions behave as the instance size varies. Additionally, all executions are carried out with three different time limits: 30 minutes, 1 hour, and 2 hours. We aim to determine the maximum time for conducting the remaining experiments based on these limits. In a second and third part, we utilize the Γ-set and c-sets to delve deeper into the performance of our five models when we modify these parameters.

5.5.3 Results for n-sets

The aim of this section is to compare the performance of our two-stage robust formulation C and iterative methods IC and ID to the nominal models LB and UB. We begin by working with the n-sets to test the behavior of solutions as the number of activities varies. We examine six different activity sizes, with n ranging from 5 to 30, and each set consists of 20 instances. Moreover, Γ is set to 4 and c is set to 1.2, their default values.

		Optimal Value	**Evaluation**		
	n		*LB*	*Cont. BU*	*Disc. BU*
LB	5	44.25	44.25	54.79	53.85
	10	80.80	80.80	96.60	95.05
	15	113.45	113.45	130.67	129.65
	20	134.90	134.90	152.64	151.70
	25	166.25	166.25	184.98	183.60
	30	186.80	186.80	206.13	205.60
UB	5	56.80	44.95	54.14	53.30
	10	105.10	85.05	95.84	94.75
	15	147.40	120.00	132.28	131.45
	20	177.85	141.25	154.48	153.50
	25	219.65	175.95	189.24	188.20
	30	252.10	199.75	213.49	212.40

Table 5.1: Computational results for the nominal lower and upper bound optimization models, LB and UB, respectively.

Table 5.1 shows the results for the nominal models, LB and UB. Both models have successfully solved all 20 instances for each activity size to optimality. In detail, the rows of the table have been divided into two sections: the first section is dedicated to the LB model, and the second to the UB model. The *Optimal Value* column displays, for each problem size, the average optimal objective function value obtained by each configuration (i.e., the value reported by the model itself). We confirm that, on average, the optimal values are greater for UB, as it provides an upper-bound to the total cost of all executions. The remaining columns of Table 5.1 are dedicated to assess the performance of both solution methods. As previously mentioned, we evaluate the solutions using three different models: LB, *Cont. BU*, and *Disc. BU*. Each model is represented in the table by a column, displaying the average evaluations for each number of activities. It is clear that the LB column produces identical

average values when evaluating the LB optimal solutions, given that we are using the same model. We also note that solutions produced by UB are suboptimal when used in LB, which confirms that solutions are indeed different.

Regarding the *Cont. BU* and *Disc. BU* columns, we observe varying behavior in both nominal variants depending on the instance size. For smaller values, UB achieves better average evaluations. However, the behavior shifts starting from 15 activities, when the UB model yields worse results (a higher objective value) than the LB model. Recall that the value of Γ is set to 4 for all executions, controlling the maximum number of start-time dependent costs that can either reach their worst-case value (5.9) or worsen their value (5.8). This may explain the change in behavior, as the adversary can potentially target almost all costs for smaller activity numbers, whereas its attack may not be as effective for larger sizes at this Γ value. Clearly, in this latter scenario, an optimistic solution closer to the one provided by the LB model would be preferable over the one given by the UB variant. Nonetheless, for the smallest sizes, sticking to the worst-case scenario, i.e., adopting a pessimistic approach, would be more effective in dealing with uncertainty. Furthermore, we note that values in column *Cont. BU* are greater than those in column *Disc. BU*, as the adversary has a larger set of attacks to choose from.

		Evaluation								
		LB			*Cont. BU*			*Disc. BU*		
	n	30 min	1 h	2 h	30 min	1 h	2 h	30 min	1 h	2 h
C	5	44.75	44.75	44.75	53.75	53.75	53.75	52.80	52.80	52.80
	10	82.40	82.40	82.40	94.28	94.28	94.28	92.70	92.70	92.70
	15	114.90	114.90	114.90	129.09	129.09	129.09	128.15	128.15	128.15
	20	136.20	136.20	136.20	150.53	150.53	150.53	149.55	149.55	149.55
	25*	168.60	167.70	167.70	184.99	184.21	183.83	183.55	182.60	182.15
	30*	189.95	188.50	188.15	207.57	206.26	205.53	206.75	205.20	204.50
IC	5	44.75	44.75	44.75	53.75	53.75	53.75	52.80	52.80	52.80
	10*	82.30	82.45	82.55	94.52	94.44	94.41	92.95	92.95	92.90
	15*	114.20	114.15	114.20	129.92	129.82	129.66	128.70	128.60	128.45
	20*	135.90	136.30	136.05	151.94	151.60	151.30	150.75	150.55	150.25
	25*	166.45	166.55	166.65	184.88	184.86	184.81	183.50	183.40	183.40
	30*	187.00	187.10	187.35	205.82	205.78	205.57	205.10	205.00	204.70
ID	5	44.85	44.85	44.85	53.76	53.76	53.76	52.80	52.80	52.80
	10*	82.40	82.45	82.55	94.67	94.65	94.63	92.55	92.50	92.50
	15*	115.60	115.30	115.35	130.12	130.05	130.05	128.25	127.85	127.85
	20*	135.80	135.80	136.00	151.93	151.80	151.83	150.40	150.25	150.15
	25*	166.70	167.15	167.15	185.05	185.07	185.07	183.35	182.85	182.85
	30*	187.20	187.50	187.70	206.01	205.96	206.02	205.10	204.90	204.85

Table 5.2: Evaluation results for the compact (C) formulation and the iterative solution models, continuous (IC) and discrete (ID).

We now compare the performance of the three remaining solution approaches examined in this

study. Results are presented in Table 5.2. In particular, we analyze the compact formulation and the two iterative solution methods, which all incorporate uncertainty into their formulation. Firstly, we observe three horizontal sections dividing the table, each representing one of the solution methods being compared (C, IC, ID). For each of these models, we have provided the results for the three execution time limits specified above: 30 minutes, 1 hour, and 2 hours. Additionally, we have segregated the results based on the instance sizes, i.e., according to the n-sets. The column for average optimal values has been omitted from this table because the iterative methods consistently fail to achieve optimality across nearly all problem sizes. Furthermore, the compact formulation also fails to find optimal solutions for larger numbers of activities. We have marked with an asterisks those problem sizes for which we have encountered at least one instance that cannot be solved to optimality within any of the time limits. Consequently, only the evaluation columns (LB, $Cont.\ BU$, $Disc.\ BU$) are included in the table.

As previously mentioned, both the compact formulation and the continuous iterative solution method produce identical optimal values. Recall that the iterative method shares solutions between the robust model (5.36)-(5.43) and the adversarial problem with continuous budgeted uncertain costs, stopping the process once both models reach the same objective value (see Section 5.4). Thus, evaluating the optimal solutions achieved by the compact formulation on $Cont.\ BU$ yields the same objective values. For example, with the 15-set and across all three time limits, C obtains an average evaluation value of 129.09 on $Cont.\ BU$. Consequently, the average optimal value for problems within the 15-set also stands at 129.09. With this in mind, the average optimal values for C are implicitly integrated into the table, with exceptions for instances comprising 25 and 30 activities, where feasible solutions are only obtained for certain instances.

We now shift our attention to the iterative methods (IC, ID). These models have only reached optimality for the 5-set, which comprises 20 instances with five activities. Consequently, for the remaining problem sizes, lower and upper bounds do not coincide, and no optimal solution is reached. Hence, the average evaluation values displayed in the table are computed from the best feasible solutions found so far, i.e., the upper bound obtained by solving the adversarial problem with respect to the solution given by formulation (5.36)-(5.43). This clarifies why increasing the time limit enhances the evaluations of IC solutions on the $Cont.\ BU$ model, or at least prevents deterioration. For example, in the case of the 20-set, the average assessment values on $Cont.\ BU$ decrease from 151.94 for a time limit of 30 minutes to 151.30 for a time limit of 2 hours. Likewise, the assessment of ID solutions on the $Disc.\ BU$ model either improve or remain consistent. For instance, consider the column corresponding to the 15-set, where the evaluation value improves from 128.25 to 127.85. Note that the $Cont.\ BU$ columns would be identical for both the compact and iterative continuous solution methods if they were able to solve all instances within each n-set to optimality. Here, with the exception of the 5-set, IC solutions perform worse than C solutions. Regarding ID, it not only fails to achieve optimality for all problem sizes starting from 10, but we also demonstrate the superior performance of the compact formulation when evaluated on $Disc.\ BU$.

In Table 5.3, we display the relative gap between the lower and upper bounds for the iterative solution methods, continuous (IC) and discrete (ID). The results confirm the maximum time limit

	Relative gap (%)					
	IC			*ID*		
n	30 min	1 h	2 h	30 min	1 h	2 h
5	0.00	0.00	0.00	0.00	0.00	0.00
10	7.06	6.12	5.95	4.05	3.14	2.95
15	10.79	9.88	8.65	8.11	6.53	6.49
20	9.81	8.99	8.27	8.11	7.89	7.49
25	9.81	9.53	9.15	9.00	7.90	7.90
30	8.88	8.79	8.19	8.48	8.15	7.88

Table 5.3: Relative gap (%) for the iterative methods. Time limit is set to 2 hours.

of 2 hours yields best bounds achieved thus far in this study. Recall that the lower bound is obtained solving (5.36)-(5.43), while the upper bound originates from the adversarial problem, which may utilize either continuous or discrete budgeted uncertainty. Relative gaps are expressed as percentages, with a zero value indicating that the method has reached the optimal solution. Again, we observe that only instances consisting of 5 activities achieve optimality. Overall, we observe a more favorable performance in the discrete scenario case, which may be due to the smaller set of adversarial attacks.

		Average values								
		No. optimalities			**No. iterations**			**Iter. best solution**		
	n	30 min	1 h	2 h	30 min	1 h	2 h	30 min	1 h	2 h
IC	5	20	20	20	7.20	7.20	7.20	7.20	7.20	7.20
	10	1	2	2	11.85	13.75	16.35	5.30	6.75	7.20
	15	0	0	0	5.95	6.80	8.10	2.20	2.80	3.85
	20	0	0	0	4.30	5.20	6.40	1.75	2.35	2.95
	25	0	0	0	3.35	3.85	4.45	1.30	1.40	1.65
	30	0	0	0	2.85	3.35	3.80	1.65	1.60	2.00
ID	5	20	20	20	6.10	6.10	6.10	6.10	6.10	6.10
	10	2	7	8	11.15	12.50	13.85	6.50	8.15	10.40
	15	0	0	0	5.90	6.90	7.85	3.30	4.30	4.30
	20	0	0	0	4.70	5.45	6.20	2.55	2.55	2.75
	25	0	0	0	3.00	3.85	4.35	1.45	2.15	2.00
	30	0	0	0	2.90	3.25	3.65	1.90	2.05	2.25

Table 5.4: Statistics for the iterative solution methods.

From Table 5.4, we highlight some of the observations made above. In particular, we have conducted some calculations for both iterative methods (*IC*, *ID*). The *No. optimalities* columns focus on the number of optimal solutions achieved by both solution approaches when tackling instances within the n-sets. These results are segregated based on the number of activities and are presented for each time limit. We note that, within the 5-set, both iterative solution methods successfully achieved optimality

for all 20 instances in this category. However, the performance drastically declines when attempting to solve problems within the 10-sets. IC required a 2-hour time limit to achieve optimality for only two instances, while ID was capable of solving a total of eight instances within the same time period. Neither of the solution methods managed to achieve optimality for any instance within the remaining n-sets.

The *No. iterations* columns display the average total number of iterations performed by the iterative solution methods. Once again, these results are categorized based on the number of activities and the specified time limit. We observe an increasing trend in the number of iterations from instances with 5 activities to those in the 10-set. Since the iterative methods involve sharing solutions, this increases the number of constraints in (5.36)-(5.43). Hence, for larger instances, the average number of iterations decreases due to the exponential growth of this robust problem with each iteration. In fact, for the 25-set and 30-set, we are only able to perform 2 or 3 iterations on average.

Finally, we have included the *Iter. best solution* columns detailing the average iteration number at which the iterative methods have stopped when reaching optimality or finding the best feasible solution upon exceeding the time limit. As detailed above, the iterative solution methods employ the adversarial problem to assess the quality of the precedence constraints generated by the robust formulation (5.36)-(5.43) during the iterative process. Henceforth, although the time limit may be reached at a specific iteration, the reported best feasible solution might have been discovered earlier. Analyzing the results presented in the table, we notice that, on average, the best solutions are identified in iterations preceding the maximum number performed before terminating the process.

Considering this information, we can now provide a clearer interpretation of some of the results presented in Table 5.2. IC and ID models seem to outperform the compact formulation when evaluated on LB, as indicated by the lower objective values achieved. Nonetheless, when we combine the data from both Table 5.2 and Table 5.4, we recognize that the solutions generated by the iterative methods originate from a model that significantly differs from the compact one, which considers all possible attack scenarios from the adversary. For instance, in the case of the 25-set, IC needs around 1.65 iterations to achieve the best solution within a 2-hour time limit, while the average total number of iterations performed within this time frame is approximately 4.45. Recalling the construction of the iterative method, we recognize that the optimal value is then attained when accounting for only one deviation scenario for the start-time dependent costs, thus aligning more closely with the LB model than with the C one. Thus, for this 25-set, evaluating the solutions of IC on LB yields an average value of 166.45 (Table 5.2), which is closer to the value obtained for LB, 166.25 (Table 5.1), than to the value obtained for C, 168.60 (Table 5.2).

To conclude the analysis of the n-sets, Figure 5.2 displays the average time requirements (in seconds) of the five models examined in this study for solving the instances within these sets. Moreover, we have included three plots, one for each time limit: 30 minutes, 1 hour, and 2 hours. This figure effectively highlights how the iterative solution methods struggle to achieve optimality for instances with more than 10 activities, which has been largely discussed throughout this chapter. In particular, focusing on the third graph, it becomes evident that ID outperforms IC in solving the 10-set. As shown in Table 5.4, ID achieves optimality in 8 instances within a time limit of 2 hours, whereas it only solves

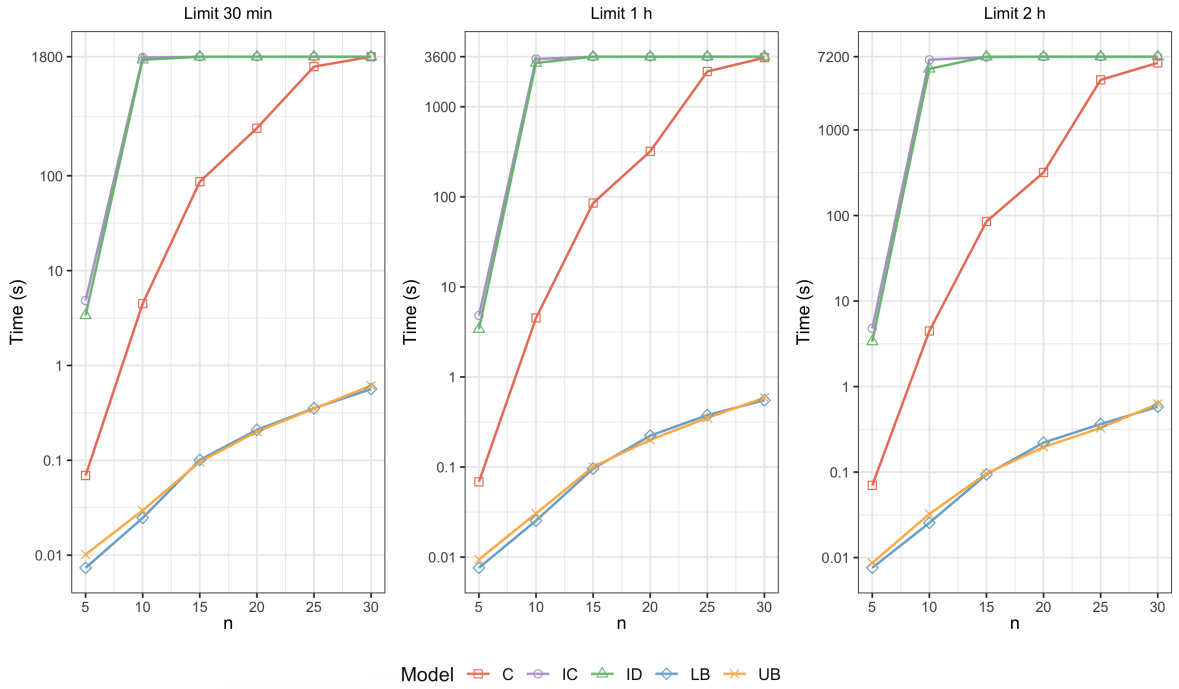


Figure 5.2: Average computational time in seconds for the five optimization models and the different time limit considered.

2 instances within a time limit of 30 minutes. Additionally, for the larger problems comprising 25 or 30 activities, the compact formulation also demonstrates limitations, as stated before.

In the light of these results, we set the time limit for all remaining experiments to 1 hour. Specifically, extending this time limit does not yield significantly better results, yet restricting all executions to 30 minutes appears inadequate. For example, in the case of the 10-set, although the overall performance does not show a significant improvement, analysis of Table 5.4 reveals that ID obtained a greater number of optimal solutions, namely 7, within a 1-hour time limit, compared to 2 optimalities within 30 minutes. Moreover, focusing on the results in Table 5.3, relative gaps consistently show better performance for a 1-hour time limit. At this stage, we must find a balance between solution quality and computational costs; hence, among the three time limits, the middle value is the best choice.

5.5.4 Results for Γ-sets

From this point onward, our focus shifts to the Γ-sets. Recall that we explore 20 possibilities for Γ, ranging from 1 to 20, which results in twenty sets conformed by 20 instances each, i.e. a total of 400 instances. In particular, we maintain the default value of c at 1.2, and all instances consist of $n = 15$ activities. We have selected this value for convenience, aiming to avoid working with excessively small instances while ensuring that the compact formulation can effectively solve to optimality the

various instances in this study. Regarding the models analyzed, we have excluded the iterative solution methods, continuous (IC) and discrete (ID), due to their poor performance in terms of solution quality and speed. They can only solve a few instances from the n-sets, limiting their usefulness to the smallest instances with 5 activities. Furthermore, the compact problem has shown better performance even in the discrete uncertain scenario, i.e. the average evaluations of C solutions on $Disc.\ BU$ outperform those obtained for the iterative discrete method.

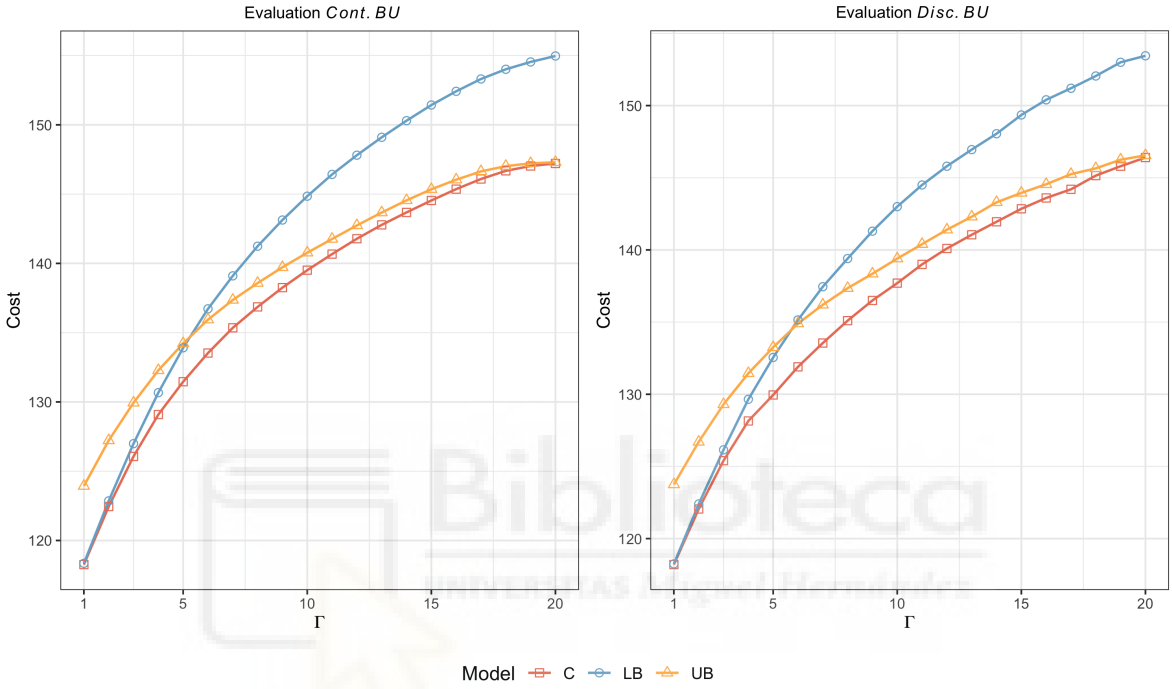


Figure 5.3: Average results for the different values of Γ.

The graphs in Figure 5.3 depict the main computational results concerning the Γ-sets. Each figure is dedicated to the evaluation results on the adversarial problem with continuous and discrete budgeted uncertainty, respectively. In detail, the total cost is plotted as a function of the Γ value. Results are displayed for the three models considered in this study: C, LB, and UB. For both adversarial problems, we observe a consistent trend in behavior. When Γ has a lower value, it significantly constrains the number of start-time dependent costs that can reach their worst-case value, resulting in good evaluations for the nominal lower-bound model. Nonetheless, as Γ exceeds 5, we notice a shift in the behavior of both nominal models, with the nominal upper-bound model offering better performance. Notably, in the scenario where Γ is set to 1, the assessment of C and LB solutions align. This trend is also evident in both graphs, where we observe a similar alignment between the evaluations of C and UB solutions in the case where $\Gamma = 20$. Consequently, we observe a convergence of the LB line to the C line on the left side and a convergence of the UB line to the C line on the right side, for both graphs. Overall, the compact formulation demonstrates superior performance compared to the behavior of the nominal models.

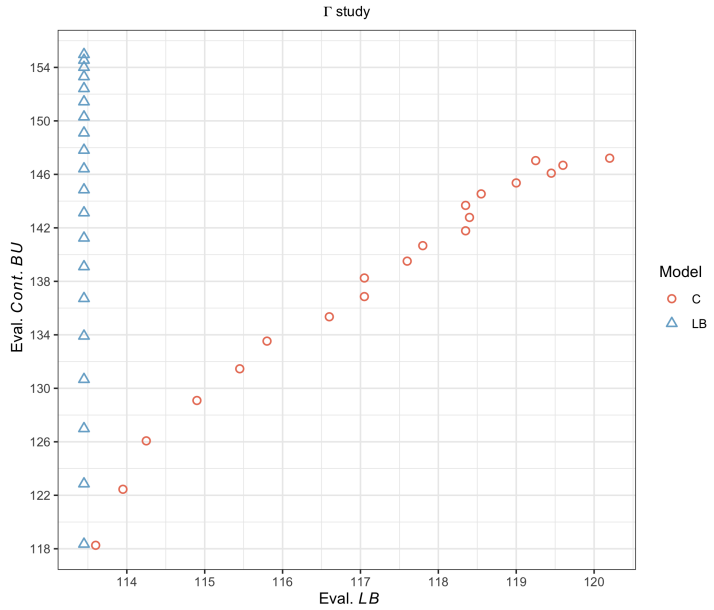


Figure 5.4: Comparison of the compact (C) and the nominal LB model (LB) with respect to the evaluation values.

To offer additional insights into the previous results, we have plotted in Figure 5.4 the evaluation values for the nominal lower-bound model and the compact formulation. In detail, we have represented LB values as a function of $Cont.\ BU$ values. We observe a total of 20 data points per model, with each data point corresponding to one value of Γ, ranging from 1 to 20. The primary objective of this figure is to examine how the trade-off between both evaluation values evolves as the value of Γ increases. It is noteworthy that lower values result in closer evaluations for both solution methods. However, as the value of Γ increases, the distance between the two points becomes greater. Since LB does not incorporate the parameter Γ into its formulation, its evaluation consistently remains the same. Specifically, it maintains a value of 113.45, as indicated in Table 5.1.

In contrast, the evaluations of C solutions on LB increase as the parameter Γ grows larger. As stated above, the inclusion of uncertainty in the model consistently degrades the performance concerning LB. However, what is notable to observe here is that the decline is more pronounced for the LB model in comparison to C when assessing the performance of both models on $Cont.\ BU$ (note that axes are scaled differently). Hence, the trade-off is favorable because the reduction in closeness to the lower bound for total cost is less significant than the improvement in dealing with the new uncertain scenario.

5.5.5 Results for c-sets

To wrap up this analysis, we shift our focus to the last set of instances, namely the c-sets. Recall that we consider nine possibilities for the c parameter, ranging from 1.0 to 1.40, with a step size of 0.05. For each potential value, we generate 20 instances, resulting in a total of 180 instances within these sets. Similar to the Γ-sets, we deal with problems consisting of $n = 15$ activities, and in this case, the

Γ parameter is set to 4, maintaining the default value used in the initial part of this study. Moreover, we have limited our analysis to the two nominal models, lower-bound (LB) and upper-bound (UB), along with the compact formulation (C).

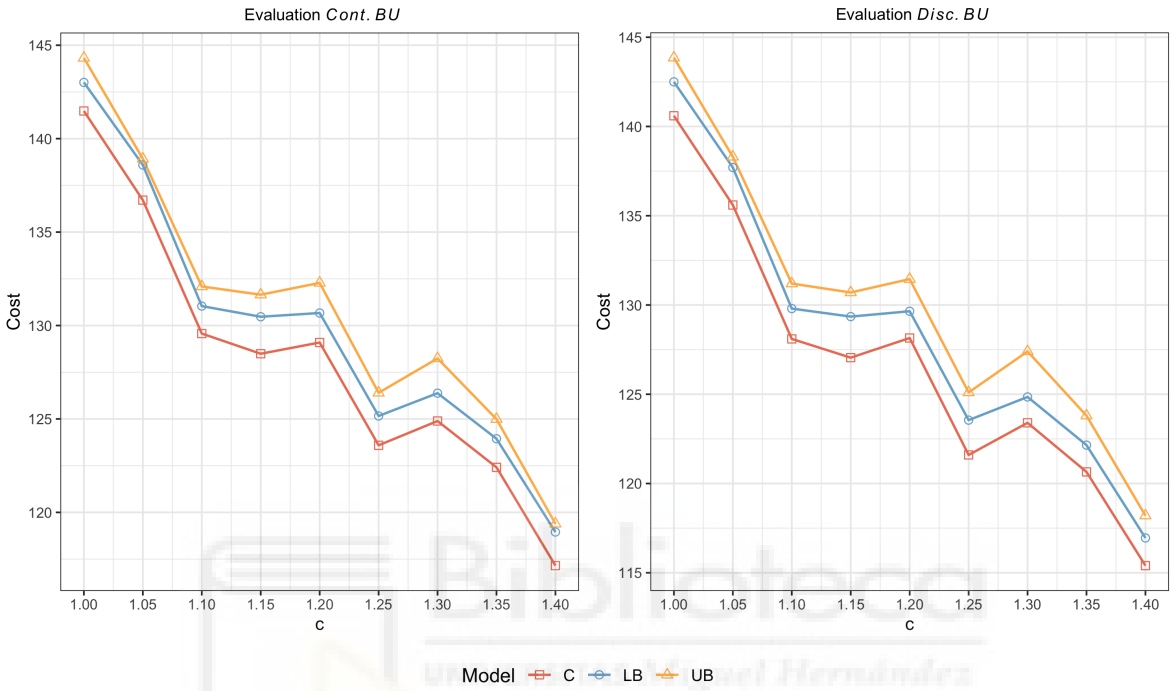


Figure 5.5: Average results for the different values of c.

Figure 5.5 presents the computational results from this final study. In detail, we have included two different graphs, similar to the ones for the Γ-sets, containing the evaluation results for both adversarial models, continuous ($Cont.\ BU$) and discrete ($Disc.\ BU$). In these graphs, the total cost is represented as a function of the c value, with the points corresponding to the three models mentioned earlier: C, LB, and UB. The results show that, for fixed values of the parameters n and Γ, adjusting the value of c appears to have minimal impact on the performance of these three models in terms of dealing with uncertainty. Despite some fluctuations due to randomness, their performance remains consistent. The only noticeable change in behavior is that, as the planning horizon increases, the objective values improve (decrease). This is attributed to the possibility of achieving a better ordering of the activities within the project as the number of time slots increases.

Overall, the computational study confirms the robustness and efficiency of the proposed compact formulation when handling uncertainty in start-time dependent costs. While iterative approaches can offer good solutions for small instances, their performance significantly declines as problem size increases. In contrast, the compact model consistently achieves high-quality solutions across all settings, making it a reliable approach for real-world applications where cost uncertainty plays a critical role.

5.6 Final remarks and future work

This chapter has explored a novel scheduling problem with start-time dependent costs under budgeted uncertainty, motivated by real-world scenarios such as fluctuating energy prices or labor tariffs. Unlike classical precedence-constrained scheduling models with start-time dependent costs—where the cost of initiating an activity varies across a discrete time horizon—the proposed nominal variant assumes no predefined ordering among activities and prohibits parallel execution. Furthermore, costs are modeled as varying throughout the processing time of each activity, leading to execution-time dependent costs. These are then aggregated into start-time dependent costs, aligning with their original formulation. While previous models with fixed precedence relations have been shown to be solvable in strongly polynomial time, we prove that this generalization is both NP-hard and not approximable. Notably, since the cost matrix is part of the input, and the time horizon is polynomial in its size, the complexity result relies on a reduction from a strongly NP-hard problem.

To address the inherent uncertainty in practical settings, we model cost variability using a robust optimization approach. In particular, we consider budgeted uncertainty sets and develop a two-stage robust optimization framework. In the first stage, the sequence of activities is fixed, while in the second stage, precise starting times are determined once actual costs are revealed. Leveraging the polynomial solvability of the precedence-constrained version, we derive compact formulations for the adversarial problems under both continuous and discrete budgeted uncertainty. Specifically, for the continuous case, we introduce a compact mixed-integer linear programming formulation. Conversely, for discrete budgeted uncertainty, we propose an iterative method based on adversarial scenario analysis.

The experimental study is conducted on three benchmark sets specifically generated for this work. First, the n-sets comprise 120 instances that vary by problem size, i.e., the number of activities. Second, the Γ-sets contain 400 instances with different robustness levels, controlled by the parameter Γ, which limits the total deviation from nominal costs. Lastly, the c-sets encompass 180 instances with varying time horizon tightness, adjusted through a scaling factor c applied to the total duration of the activities. The five models evaluated—LB, UB, C, ID, and IC—are assessed based on their robustness and computational efficiency, measured by their speed in achieving optimal solutions.

Extensive computational results highlight that the compact formulation consistently outperforms iterative methods in terms of robustness and scalability. The compact model solves all n-set instances optimally for problems with up to 20 activities, and maintains high-quality solutions even for larger problem sizes. In contrast, iterative methods reach optimality only for instances with up to 5 activities and exhibit significant scalability issues as the problem size increases. Notably, solutions obtained with the compact model, when evaluated under $Disc.\ BU$, often yield better results than those produced by ID, demonstrating superior robustness even when the solutions are generated under different uncertainty paradigms. Furthermore, the results show a clear advantage of robust solutions, offering a favorable trade-off between nominal performance and worst-case protection compared to optimistic (LB) and pessimistic (UB) nominal strategies. The analysis of the c-sets confirms the methods' robustness under varying time horizons, although it also reveals greater variability due to the randomness introduced.

Building on these findings, several directions for future research naturally emerge. Potential extensions include incorporating additional sources of uncertainty—such as uncertain processing times—or adopting alternative paradigms, such as stochastic scheduling or fuzzy programming. Exploring recoverable robustness frameworks, which allow limited rescheduling after the realization of uncertainty, also represents a promising avenue. Furthermore, extending the proposed model to accommodate richer constraints—such as explicit precedence relations, resource availability, or multi-mode execution—would significantly enhance its practical applicability. Finally, the development of dedicated heuristics or metaheuristics may facilitate the efficient resolution of larger, industrial-scale instances, thus helping to bridge the gap between theoretical models and real-world implementation.

Chapter 6
Conclusions

Project scheduling represents a critical challenge for organizations seeking to manage resources efficiently, control costs, and deliver timely outcomes in increasingly dynamic and uncertain environments. This thesis has addressed several advanced variants of the project scheduling problem to better capture complexities encountered in practical environments, such as time-dependent costs, multi-mode execution, fluctuating resource availabilities, and cost uncertainty. The primary contributions span rigorous mathematical modeling, algorithmic innovation, and extensive computational analyses, providing robust and practical tools for complex scheduling scenarios. This final chapter summarizes the key contributions and insights derived from this thesis, and briefly reflects on the future research avenues outlined in the preceding chapters.

The first line of research, presented in Chapter 3, explores an extension of the classical resource-constrained project scheduling problem (RCPSP), namely the bi-objective RCPSP with time-dependent resource costs (RCPSP_TDRC), which considers two objectives: minimizing the project makespan and the total cost of resource usage. By rigorously evaluating seven state-of-the-art multi-objective evolutionary algorithms (MOEAs), this study demonstrates their ability to efficiently generate high-quality approximations of the Pareto front (PF)—or a reference PF when the true one is unknown—within a reasonable computational time. Overall, all metaheuristics produce accurate PF approximations. In particular, the non-dominated sorting genetic algorithm II (NSGA-II) exhibits superior performance across the experimental analyses conducted in this chapter, consistently achieving statistically significant improvements. In contrast, the multi-objective evolutionary algorithm based on decomposition (MOEA/D) and the indicator-based evolutionary algorithm (IBEA) perform considerably worse. These results reinforce the relevance of evolutionary algorithms as effective tools for generating diverse and competitive trade-off solutions, essential for informed decision-making in complex scheduling scenarios.

Building upon these ideas, the second line of research, developed in Chapter 4, introduces the multi-mode RCPSP with time-dependent resource costs and capacities (MRCPSP_TDRCC), further enriching the scheduling framework. This problem variant combines multi-objective optimization with time-varying resource conditions and alternative execution modes for activities. The chapter presents a comprehensive mathematical formulation and proposes an exact solution method based on the augmented ϵ-constrained approach (AUGMECON). While effective for small instances, the exact method shows clear limitations when tackling larger, real-world-sized problems—particularly when the number

of activities exceeds 20. To address these limitations, a tailored genetic algorithm inspired by NSGA-II is developed, demonstrating strong performance in approximating the PF for medium and large-scale instances. By consistently outperforming the exact method, the metaheuristic underscores the critical role of problem-specific evolutionary strategies in solving complex, large-scale, multi-objective scheduling problems that reflect realistic industrial scenarios. Moreover, the metaheuristic is able to closely approximate the PF obtained by AUGMECON even for small-sized instances, further validating its robustness and effectiveness across different problem sizes.

To conclude, the third line of research, developed in Chapter 5, adopts a robust approach to tackle a scheduling problem with start-time dependent costs under budgeted uncertainty. This framework reflects practical situations such as variable energy prices or labor costs, and advances beyond classical deterministic models. Unlike traditional precedence-constrained scheduling problems, the proposed nominal variant assumes no predefined ordering of activities and prohibits parallel execution. While models with fixed precedence relations are solvable in strongly polynomial time, we show that this generalization is NP-hard and hard to approximate. Additionally, the model incorporates execution-time dependent costs—costs that vary throughout an activity's duration—which can be aggregated into start-time dependent costs to recover the original formulation. In the uncertain setting, a two-stage robust optimization framework is proposed: the activity ordering is fixed in the first stage, and precise starting times are assigned in the second, once actual costs are revealed. This structure suits applications like airport scheduling or medical appointments, where activity sequences are fixed in advance but timing remains flexible. We introduce a compact mixed-integer linear programming formulation to address continuous budgeted uncertainty, along with an iterative scenario-based method for the discrete case. Computational results confirm the compact model's robustness and scalability, even within the discrete framework, highlighting the advantages of explicitly incorporating uncertainty into scheduling decisions. These contributions enhance schedule reliability and flexibility under realistic conditions.

As discussed in greater depth throughout the previous chapters, this work opens several promising directions for future research. Building on the foundations established here, subsequent studies could further enrich model realism by incorporating additional sources of uncertainty or exploring alternative paradigms—such as stochastic or fuzzy approaches—for uncertainty modeling. Comparative analyses of advanced metaheuristics, particularly in the multi-mode setting, may help identify the most effective strategies for solving increasingly complex instances. Likewise, the development of dedicated heuristics or metaheuristics for the uncertainty-based model, or the inclusion of resource-related constraints, could expand its practical reach. Applying these frameworks to real-world case studies would offer a valuable step toward closing the gap between theory and application.

In conclusion, this thesis provides a strong methodological foundation for advancing scheduling optimization by emphasizing the integration of realism and adaptability into project scheduling methodologies. Through the incorporation of practical complexities—such as multiple objectives, time-varying constraints, multi-mode execution, and uncertainty—into comprehensive optimization frameworks, the research contributes to both theoretical advancement and practical capability. The proposed models and algorithms facilitate the generation of diverse and well-informed trade-off solutions, supporting

strategic planning in environments characterized by resource constraints and conflicting objectives. As a result, the proposed approaches can support organizations in improving operational efficiency, strengthening robustness, and adapting more effectively to the dynamics of real-world project environments. This synthesis of theoretical and applied perspectives offers a solid basis for future developments toward more flexible, reliable, and scalable scheduling frameworks.

Conclusiones

La planificación de proyectos constituye un desafío fundamental para las organizaciones que aspiran a gestionar sus recursos de forma eficiente, controlar los costes y cumplir los plazos establecidos en contextos cada vez más dinámicos e inciertos. Esta tesis ha abordado diversas variantes avanzadas del problema de planificación de proyectos con el objetivo de reflejar mejor las complejidades propias de entornos reales, tales como los costes dependientes del tiempo, la pesencia de múltiples modos de ejecución, la variabilidad en la disponibilidad de recursos y la incertidumbre en el contexto operativo. Las contribuciones principales de este trabajo combinan una rigurosa modelización matemática, innovación algorítmica y un exhaustivo análisis computacional, dando lugar a herramientas robustas y aplicables para afrontar problemas complejos de planificación. Este capítulo final resume las aportaciones más relevantes e ideas desarrolladas a lo largo de la tesis, además de presentar brevemente algunas líneas futuras discutidas en capítulos anteriores.

La primera línea de investigación, expuesta en el Capítulo 3, estudia una extensión del problema clásico de planificación de proyectos con recursos limitados (RCPSP): el RCPSP biobjetivo con costes de recursos dependientes del tiempo (RCPSP_TDRC), cuyo propósito es minimizar simultáneamente la duración total del proyecto y el coste global derivado del uso de los recursos. A través de una evaluación exhaustiva de siete algoritmos evolutivos multi-objetivo (MOEAs) de última generación, este estudio evidencia la capacidad de estos algoritmos para aproximar de manera eficiente y precisa el frente de Pareto (PF)—o un frente de referencia cuando el verdadero no es conocido—en tiempos computacionales reducidos. En términos generales, todas las metaheurísticas consideradas alcanzan aproximaciones precisas del PF. En particular, el algoritmo NSGA-II destaca claramente en los distintos experimentos realizados, logrando mejoras estadísticamente significativas de forma consistente. Por el contrario, enfoques como MOEA/D e IBEA exhiben un desempeño notablemente inferior. Estos resultados subrayan la utilidad de los algoritmos evolutivos como herramientas eficaces para obtener soluciones diversas y competitivas, esenciales para la toma de decisiones en entornos complejos de planificación.

Basándose en estos avances, la segunda línea de investigación, desarrollada en el Capítulo 4, introduce el RCPSP multi-modo con costes y capacidades dependientes del tiempo (MRCPSP_TDRCC), enriqueciendo aún más el marco de planificación de proyectos. Esta variante combina la optimización multi-objetivo con condiciones de recursos variables a lo largo del tiempo y múltiples modos de ejecución por actividad. En el capítulo se propone una formulación matemática del problema junto con un

método exacto de resolución basado en la técnica AUGMECON (*augmented ϵ-constrained*). Aunque este método resulta eficaz en instancias pequeñas, presenta claras limitaciones cuando se abordan problemas de mayor escala, particularmente cuando el número de actividades supera las 20. Para superar estas dificultades, se desarrolla un algoritmo genético adaptado e inspirado en NSGA-II, que muestra un desempeño destacado en la aproximación del PF para instancias medianas y grandes. Esta metaheurística no solo supera sistemáticamente al método exacto, sino que también logra aproximaciones precisas del PF incluso en instancias pequeñas, reforzando así su robustez y eficacia frente a diversos tamaños del problema. De esta forma, se evidencia claramente la utilidad de estrategias evolutivas diseñadas específicamente para abordar contextos industriales complejos y realistas.

Finalmente, la tercera línea de investigación, abordada en el Capítulo 5, se centra en un enfoque robusto para resolver un problema de planificación de proyectos con costes dependientes del instante de inicio de las actividades, bajo condiciones de incertidumbre presupuestada. Este marco refleja situaciones prácticas como la variabilidad en los precios de la energía o los costes laborales, superando las limitaciones de los modelos deterministas clásicos. A diferencia de los problemas tradicionales con restricciones de precedencia, la variante nominal propuesta no impone un orden predefinido entre las actividades, y además prohíbe su ejecución en paralelo. Mientras que los modelos con precedencias fijas pueden resolverse en tiempo fuertemente polinómico, se demuestra que esta generalización constituye un problema NP-duro y difícil de aproximar. Además, el modelo incorpora costes dependientes del tiempo de ejecución—que varían a lo largo de la duración de cada actividad—y que pueden agregarse como costes dependientes del inicio para recuperar la formulación original. Para tratar la incertidumbre, se propone un marco de optimización robusta en dos etapas: en la primera, se fija el orden de las actividades y, una vez revelados los costes reales, se determinan los instantes exactos de inicio. Esta estructura resulta adecuada para aplicaciones como la programación en aeropuertos o las citas médicas, donde el orden debe establecerse con antelación, pero los horarios pueden mantenerse flexibles. En particular, se propone una formulación compacta de programación lineal entera mixta para el caso de incertidumbre presupuestada continua, así como un método iterativo basado en escenarios para su variante discreta. Los resultados computacionales confirman la robustez y escalabilidad del modelo compacto, incluso bajo incertidumbre discreta, y ponen de manifiesto las ventajas de considerar explícitamente la incertidumbre en la toma de decisiones de planificación. Estas contribuciones incrementan la fiabilidad y flexibilidad de los calendarios en contextos inciertos y realistas.

Como ya se ha discutido con mayor detalle en capítulos anteriores, este trabajo abre diversas líneas prometedoras para futuras investigaciones. Partiendo de los modelos y algoritmos desarrollados, se podrían incorporar nuevas fuentes de incertidumbre, explorar enfoques alternativos como la optimización estocástica o fuzzy, o adaptar los marcos existentes a contextos más realistas mediante la inclusión de restricciones adicionales de recursos o relaciones de precedencia. Asimismo, el análisis comparativo de metaheurísticas avanzadas para el problema multi-modo o el desarrollo de estrategias heurísticas específicas para entornos inciertos constituyen vías con alto potencial. Finalmente, la aplicación práctica de estos modelos en casos reales ayudaría a validar su utilidad y fortalecería el vínculo entre la teoría y la práctica.

En definitiva, esta tesis proporciona una base metodológica sólida para avanzar en la optimización

aplicada a la planificación de proyectos, enfatizando la integración de realismo y adaptabilidad en los modelos propuestos. Al incorporar múltiples objetivos, restricciones variables en el tiempo, ejecución multi-modo e incertidumbre dentro de marcos completos de optimización, esta investigación contribuye tanto al desarrollo teórico como a la capacidad práctica del área. Los modelos y algoritmos presentados permiten generar soluciones diversas y bien fundamentadas, facilitando la toma de decisiones estratégicas en entornos marcados por limitaciones de recursos y objetivos contrapuestos. En conjunto, los enfoques propuestos pueden ayudar a las organizaciones a mejorar su eficiencia operativa, aumentar la robustez de sus planes y adaptarse mejor a las dinámicas reales de los proyectos. Esta síntesis entre perspectivas teóricas y aplicadas constituye un punto de partida sólido para futuros desarrollos orientados al diseño de marcos de planificación más flexibles, fiables y escalables.

Appendix A
Statistical Tests Results

In this appendix, we include complementary statistical analyses that support the experimental findings discussed in Chapters 3 and 4. The results presented here serve to reinforce the conclusions drawn from the calibration and comparison of metaheuristics.

A.1 Calibration of the metaheuristics in Section 3.4.3

This section illustrates the application of the ranking-based decision criterion introduced in Section 3.4.2, referred to as the discarded criterion, where a predefined order among performance indicators is used to identify the best configuration. We apply this procedure to the J120 calibration set using NSGA-II as a case study. For the sake of brevity, we include only the statistical results necessary to support the final decision. Therefore, we do not report the full two-way ANOVA results for all indicators, but only for those that contribute to identifying the best configuration. The analysis is conducted in `R`.

Factor	Df	Sum Sq	Mean Sq	F value	Pr(>F)
config	11	0.0097	0.000884	13.7693	$<$ 2e-16***
instance	119	21.3833	0.179691	2799.7960	$<$ 2e-16***
config:instance	1309	0.0933	0.000071	1.1102	0.01265*
Residuals	2880	0.1848	0.000064		

Table A1: Two-way ANOVA results for the HV response across NSGA-II configurations on the J120 calibration set. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Following the ordering of the performance indicators established by the discarded criterion, we begin the analysis with the hypervolume (HV). A two-way ANOVA is applied to the results obtained for the different NSGA-II configurations on the J120 calibration set. As shown in Table A1, both the instance and configuration factors are statistically significant at the 1% level, while their interaction is significant at the 5% level, indicating that the configuration influences the observed HV values.

To further examine these differences, Tukey's HSD test is applied. Figure A1 presents the mean HV values with 99% confidence intervals and grouping letters. Each configuration label includes the name of the metaheuristic—NSGA-II in this case—the population size N, the crossover probability p_c, and a scaling factor (1.0 or 2.0) that, divided by the number of activities n, defines the mutation probability p_m. As shown, several configurations fall into the first treatment group (group a), indicating no statistically significant differences among them in terms of average HV values. Therefore, no single configuration can be selected as the best based on this metric alone.

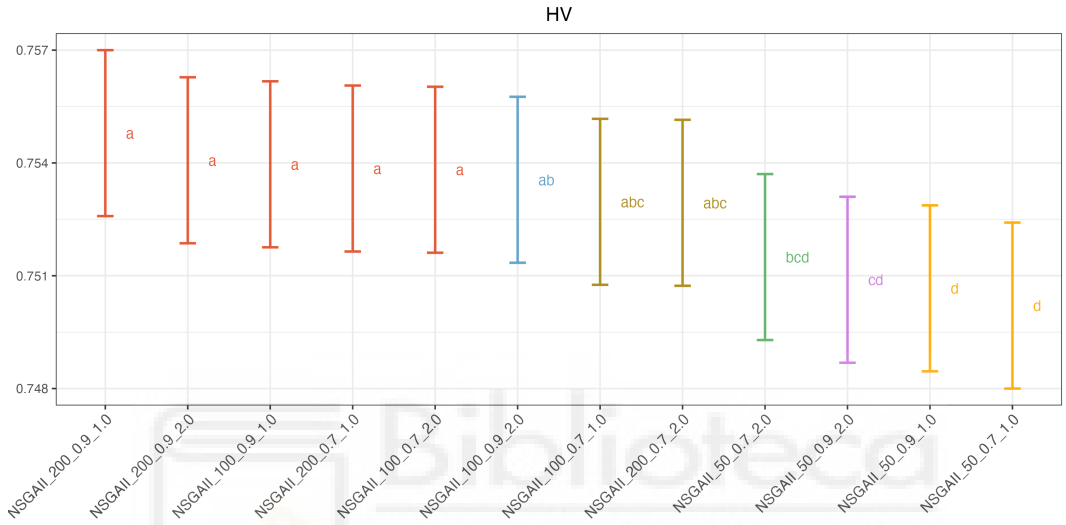


Figure A1: Hypervolume mean plots with Tukey's HSD 99% confidence intervals for NSGA-II calibration on the J120 benchmark set.

We then proceed to the second performance indicator in the list: spread (Δ). The same methodology is applied. As reported in Table A2, the results of the two-way ANOVA again reveal significant effects for all factors, including their interaction. Figure A2 displays the results of Tukey's HSD test for this metric using the same visual format. Unlike the HV case, the analysis of spread allows us to clearly identify a best-performing configuration. By focusing on those configurations that were previously part of the top-performing group for HV (group a), we find that configuration `NSGAII_200_0.9_2.0` achieves statistically superior results compared to the rest in terms of spread.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	11	62.683	5.6985	1230.0566	< 2.2e-16***
instance	119	18.603	0.1563	33.7437	< 2.2e-16***
config:instance	1309	17.204	0.0131	2.8369	< 2.2e-16***
Residuals	2880	13.342	0.0046		

Table A2: Two-way ANOVA results for the Δ response across NSGA-II configurations on the J120 calibration set. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

As a result, the calibration procedure for NSGA-II on the J120 dataset yields a configuration with an initial population size of 200, a crossover probability of 0.9, and a mutation probability of $\frac{2}{n}$, where n is the number of activities in the project. This procedure is systematically repeated for each of the seven metaheuristics considered in the experimental setup, and independently for both the J120 and J60 calibration sets. For each combination of algorithm and dataset, the best-performing configuration is selected. These calibrated settings are then used in the comparison phase, where the performance of the metaheuristics is statistically evaluated across benchmark instances.

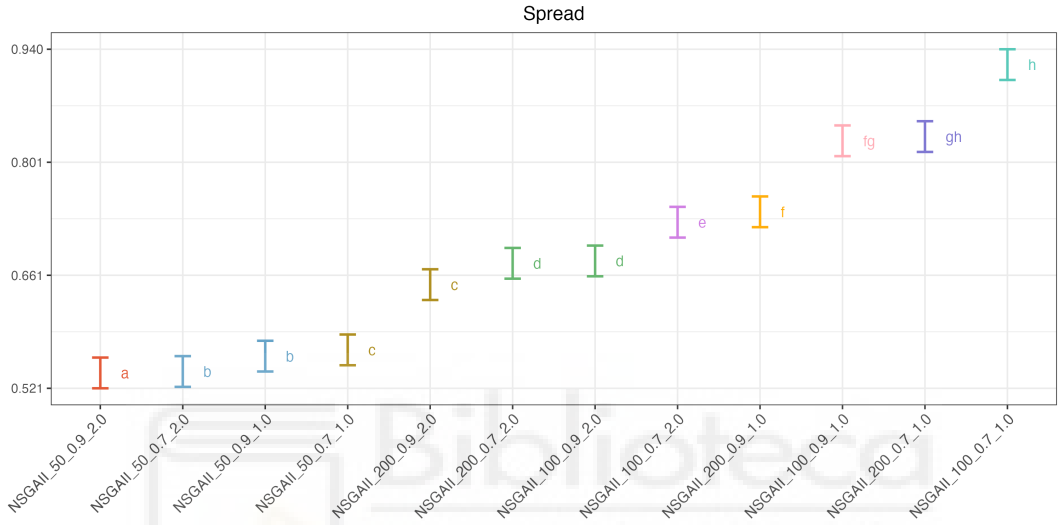


Figure A2: Spread mean plots with Tukey's HSD 99% confidence intervals for NSGA-II calibration on the J120 benchmark set.

A.2 Comparison of the best configurations in Section 3.4.4

This section presents the detailed results of the statistical analyses conducted to compare the best configurations previously identified for each of the seven metaheuristics during the calibration phase. In particular, we report the outputs of the two-way ANOVA and the mean plots with Tukey's HSD 99% confidence intervals. These results complement the conclusions drawn in Section 3.4.4, providing additional evidence to support the ranking established based on the discarded criterion.

A.2.1 J60 dataset

We begin with the results for the J60 evaluation dataset. As described in Section 3.4.4, a two-way ANOVA is applied independently to each of the five performance indicators, following the model structure introduced in Model (3.9). Factor A represents the set of problem instances—384 levels—while factor B corresponds to the seven metaheuristic configurations. Note that all necessary hypotheses are checked, with no problems in the residuals found. The analysis is conducted in `R`.

Tables A3 to A7 summarize the ANOVA results for each metric: hypervolume (HV), spread (Δ), additive ϵ-indicator (I_{ϵ^+}), modified inverted generational distance (IGD^+), and μ-indicator (μ). In all cases, the main effects and their interaction are statistically significant, confirming that the performance indicators are influenced by both the algorithm and the specific instance, as well as their interaction.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	6	252.880	42.147	1.1077e+05	< 2.2e-16***
instance	383	51.472	0.134	3.5322e+02	< 2.2e-16***
config:instance	2298	19.497	0.008	2.2299e+01	< 2.2e-16***
Residuals	5376	2.045	0.000		

Table A3: Two-way ANOVA results for the response HV. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	6	479.54	79.924	9.1651e+03	< 2.2e-16***
instance	383	71.23	0.186	2.1327e+01	< 2.2e-16***
config:instance	2298	53.52	0.023	2.6705e+00	< 2.2e-16***
Residuals	5376	46.88	0.009		

Table A4: Two-way ANOVA results for the response Δ. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	6	299.152	49.859	71009.715	< 2.2e-16***
instance	383	12.799	0.033	47.595	< 2.2e-16***
config:instance	2298	37.710	0.016	23.371	< 2.2e-16***
Residuals	5376	3.775	0.001		

Table A5: Two-way ANOVA results for the response I_{ϵ^+}. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

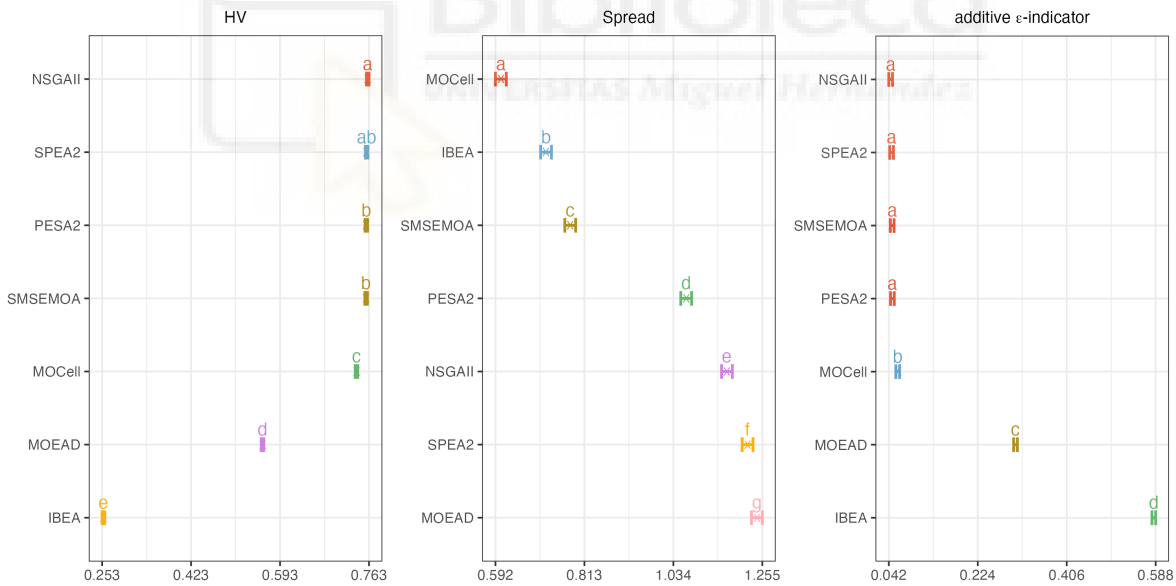
Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	6	142.387	23.7312	6.5854e+04	< 2.2e-16***
instance	383	6.690	0.0175	4.8473e+01	< 2.2e-16***
config:instance	2298	24.627	0.0107	2.9739e+01	< 2.2e-16***
Residuals	5376	1.937	0.0004		

Table A6: Two-way ANOVA results for the response IGD^+. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Factor	Df	Sum Sq	Mean Sq	F value	Pr(>F)
config	6	197.444	32.907	7.2604e+03	< 2.2e-16***
instance	383	26.222	0.068	1.5106e+01	< 2.2e-16***
config:instance	2298	17.239	0.008	1.6551e+00	< 2.2e-16***
Residuals	5376	24.366	0.005		

Table A7: Two-way ANOVA results for the response μ. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

To complement the statistical results, Figure A3 presents the mean plots for each performance indicator, including Tukey's HSD 99% confidence intervals for the best configurations of the metaheuristics. The algorithms are ranked according to the treatment groups identified by the statistical test; thus, group a appears first and corresponds to those techniques with the best average values for the performance indicators. It is important to note that when HSD groups differ, the mean values are significantly different. This representation therefore provides a clear overview of the best-performing algorithms for each metric considered, along with the corresponding range of indicator values.



These graphs not only validate the findings from Section 3.4.4, but also reinforce the robustness of NSGA-II, which ranks as the top-performing metaheuristic in four out of the five performance indicators considered. In contrast, MOEA/D and IBEA consistently underperform, with IBEA showing acceptable results only for the Δ indicator. Meanwhile, the remaining metaheuristics—SPEA2, MOCell, PESA-II, and SMS-EMOA—demonstrate competitive and reliable performance, confirming their suitability for addressing this problem variant and consistently achieving high-quality results.

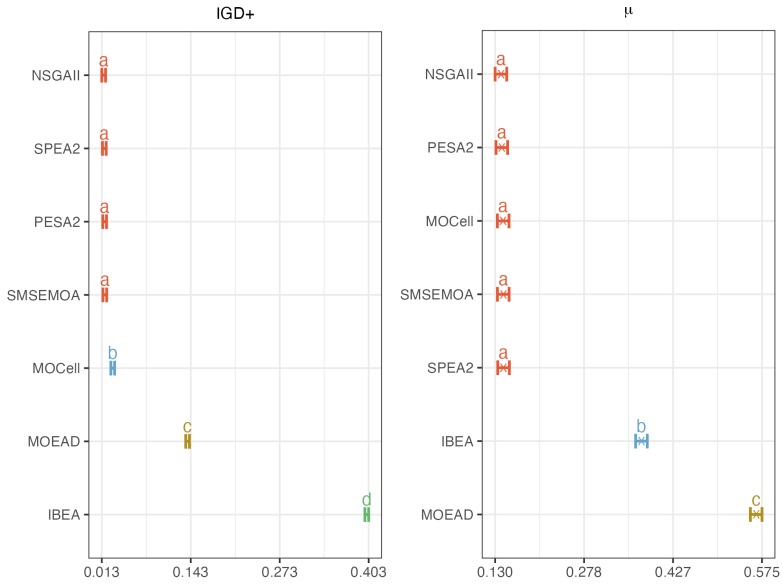


Figure A3: Hypervolume, spread, additive ϵ-indicator, IGD^+, and μ-indicator mean plots with Tukey's HSD 99% confidence intervals for the best configurations of the metaheuristics.

A.2.2 J120 dataset

We proceed with the results for the J120 evaluation dataset. As with the J60 set, a two-way ANOVA is independently applied to each of the five performance indicators, following the model structure defined in Model (3.9). Here, factor A represents the set of problem instances—480 in total—while factor B refers to the seven metaheuristic configurations under comparison. All necessary assumptions were verified, and no issues were detected in the residual analysis. The statistical analyses are conducted in R.

Factor	Df	Sum Sq	Mean Sq	F value	Pr(>F)
config	6	237.063	39.510	1.3162e+05	< 2.2e-16***
instance	479	33.954	0.071	2.3614e+02	< 2.2e-16***
config:instance	2874	7.868	0.003	9.1198	< 2.2e-16***
Residuals	6720	2.017	0.000		

Table A8: Two-way ANOVA results for the response HV. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Factor	Df	Sum Sq	Mean Sq	F value	Pr(>F)
config	6	375.99	62.665	11684.1212	< 2.2e-16***
instance	479	19.68	0.041	7.6591	< 2.2e-16***

config:instance	2874	29.47	0.010	1.9120	< 2.2e-16***
Residuals	6720	36.04	0.005		

Table A9: Two-way ANOVA results for the response Δ. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	6	283.207	47.201	1.0967e+05	< 2.2e-16***
isntace	479	3.684	0.008	1.7868e+01	< 2.2e-16***
config:instance	2874	8.955	0.003	7.2394	< 2.2e-16***
Residuals	6720	2.892	0.000		

Table A10: Two-way ANOVA results for the response I_{ϵ^+}. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	6	108.553	18.0921	1.4673e+05	< 2.2e-16***
instance	479	0.796	0.0017	1.3483e+01	< 2.2e-16***
config:instance	2874	3.170	0.0011	8.9457	< 2.2e-16***
Residuals	6720	0.829	0.0001		

Table A11: Two-way ANOVA results for the response IGD^+. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
config	6	219.083	36.514	13096.4272	< 2.2e-16***
instance	479	10.272	0.021	7.6913	< 2.2e-16***
config:instance	2874	11.200	0.004	1.3977	< 2.2e-16***
Residuals	6720	18.736	0.003		

Table A12: Two-way ANOVA results for the response μ. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Tables A8 to A12 report the ANOVA results for each performance indicator. As for the J60 evaluation set, the main effects and their interaction are statistically significant in every case.

In order to get additional insights from the above results, Figure A4 provides the mean plots of the five performance indicators with Tukey's HSD 99% confidence intervals for the selected best configurations of the metaheuristics. These graphical results complement and reinforce the conclusions

Figure A4: Hypervolume, spread, additive ϵ-indicator, IGD^+, and μ-indicator mean plots with Tukey's HSD 99% confidence intervals for the best configurations of the metaheuristics.

presented in Section 3.4.4, confirming NSGA-II as the most robust and consistently high-performing metaheuristic across all indicators. Among the remaining techniques, PESA-II emerges as a strong alternative, closely followed by MOCell and SMS-EMOA, which also demonstrate solid and reliable performance. Conversely, MOEA/D and IBEA remain the least effective algorithms, consistently ranking in the lowest treatment groups across the metrics considered. Together with the findings for the J60 dataset, these results further strengthen the overall conclusions of this study and validate the reliability of the proposed comparison framework across different problem scales.

A.3 Calibration of the NSGA-II in Section 4.5.2

In this section, we present the ANOVA results for the calibration of NSGA-II discussed in Section 4.5.2. Note that all necessary hypotheses are checked, with no problems in the residuals found. The analysis is conducted in `R`, fitting a separate model for each response variable, corresponding to each performance indicator. To reduce unnecessary complexity, we applied the `step()` function, retaining only the factors and interactions that significantly contribute to explaining variability in each response. We begin by examining the model with hypervolume as the response variable.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
N	1	0.057	0.0571	5.8715	0.01540*
p_c	1	0.001	0.0012	0.1239	0.72488
p_m	1	0.000	0.0001	0.0092	0.92340
p_r	2	28.994	14.4972	1491.4579	< 2.2e-16***
p_e	1	0.213	0.2134	21.9503	2.824e-06***
N:p_m	1	0.055	0.0547	5.6268	0.01770*
p_c:p_m	1	0.040	0.0398	4.0975	0.04296*
p_m:p_r	2	0.058	0.0288	2.9581	0.05195
p_m:p_e	1	0.056	0.0562	5.7826	0.01620*
Residuals	14148	137.521	0.0097		

Table A13: ANOVA results for the response HV. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

As shown in Table A13, only three factors are statistically significant: population size, replacement probability, and exchange probability. Additionally, some interaction effects significantly impact the response. In contrast, the main effects of both crossover and mutation probabilities are not statistically significant, and the probability of massive mutation has no impact on the response variable, meaning it does not contribute to explaining the variability of the hypervolume.

Next, Table A14 analyzes the results of the model with the ϵ-indicator as the response variable. The results confirm that, once again, the main effects of population size, replacement probability, and exchange probability are the only statistically significant. Similarly, the probability of massive mutation is not incorporated into the model, meaning it does not contribute to explaining the variability of $I_{\epsilon+}$.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
N	1	0.072	0.0715	15.1875	9.779e-05***
p_c	1	0.010	0.0097	2.0700	0.1502460
p_m	1	0.005	0.0046	0.9672	0.3253848
p_r	2	46.498	23.2489	4937.7881	< 2.2e-16***
p_e	1	0.386	0.3863	82.0459	< 2.2e-16***

N:p_m	1	0.052	0.0523	11.1184	0.0008569***
p_c:p_m	1	0.031	0.0314	6.6791	0.0097648**
N:p_r	2	0.043	0.0216	4.5826	0.0102438*
p_m:p_r	2	0.072	0.0359	7.6284	0.0004885***
N:p_e	1	0.002	0.0023	0.4902	0.4838533
p_m:p_e	1	0.037	0.0369	7.8336	0.0051355**
p_r:p_e	1	0.029	0.0291	6.1911	0.0128508*
N:p_m:p_r	2	0.014	0.0070	1.4884	0.2257774
N:p_m:p_e	1	0.013	0.0125	2.6554	0.1032209
N:p_r:p_e	1	0.000	0.0002	0.0508	0.8216358
p_m:p_r:p_e	1	0.000	0.0002	0.0474	0.8276282
N:p_m:p_r:p_e	1	0.010	0.0097	2.0539	0.1518343
Residuals	14138	66.567	0.0047		

Table A14: ANOVA results for the response I_{ϵ^+}. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Table A15 presents the results of the model with the μ-indicator as the response variable. Once again, we confirm that the massive mutation has no effect on the response variable, as it is not included in the model. The main effects of all factors are statistically significant, except for crossover probability, which has a high p-value. Additionally, several interaction effects remain in the final model (after applying the `step()` function), highlighting their contribution to explaining variability in the response.

Factor	**Df**	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
N	1	1.335	1.335	108.2534	< 2.2e-16***
p_c	1	0.001	0.001	0.0410	0.839456
p_m	1	0.394	0.394	31.9058	1.650e-08***
p_r	2	103.055	51.528	4177.2528	< 2.2e-16***
p_e	1	1.956	1.956	158.6037	< 2.2e-16***
N:p_c	1	0.004	0.004	0.3176	0.573043
N:p_m	1	0.025	0.025	2.0645	0.150791
p_c:p_m	1	0.000	0.000	0.0323	0.857420
N:p_r	2	0.440	0.220	17.8536	1.803e-08***
p_c:p_r	2	0.048	0.024	1.9336	0.144661
p_m:p_r	2	0.039	0.020	1.5822	0.205560
N:p_e	1	0.002	0.002	0.1237	0.725016
p_c:p_e	1	0.002	0.002	0.1832	0.668614
p_m:p_e	1	0.031	0.031	2.4818	0.115192
p_r:p_e	1	0.111	0.111	8.9891	0.002721**
N:p_c:p_m	1	0.048	0.048	3.8611	0.049437*

N:p_c:p_r	2	0.000	0.000	0.0108	0.989251
N:p_m:p_r	2	0.035	0.017	1.4182	0.242174
p_c:p_m:p_r	2	0.001	0.000	0.0212	0.979051
N:p_c:p_e	1	0.010	0.010	0.7977	0.371807
N:p_m:p_e	1	0.003	0.003	0.2213	0.638056
p_c:p_m:p_e	1	0.018	0.018	1.4499	0.228566
N:p_r:p_e	1	0.003	0.003	0.2469	0.619283
p_c:p_r:p_e	1	0.055	0.055	4.4296	0.035338*
p_m:p_r:p_e	1	0.002	0.002	0.1717	0.678622
N:p_c:p_m:p_r	2	0.022	0.011	0.8854	0.412584
N:p_c:p_m:p_e	1	0.000	0.000	0.0280	0.867208
N:p_c:p_r:p_e	1	0.001	0.001	0.0496	0.823700
N:p_m:p_r:p_e	1	0.004	0.004	0.2903	0.590064
p_c:p_m:p_r:p_e	1	0.014	0.014	1.0967	0.295008
N:p_c:p_m:p_r:p_e	1	0.032	0.032	2.5820	0.108105
Residuals	14120	174.174	0.012		

Table A15: ANOVA for the response μ. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

Lastly, in Table A16, we focus on the model with the C-metric (C) as the response variable. The results indicate that all factors are statistically significant, except for the massive mutation probability, which is excluded from the model after the reduction procedure. Interactions between factors also show a significant effect on the response variable.

Factor	Df	Sum Sq	Mean Sq	F value	Pr(>F)
N	1	0.0298	0.02981	18.164	2.040e-05***
p_c	1	0.0168	0.01677	10.219	0.001393**
p_m	1	0.8479	0.84792	516.581	< 2.2e-16***
p_r	2	4.2802	2.14011	1303.827	< 2.2e-16***
p_e	1	0.0407	0.04073	24.811	6.398e-07***
p_m:p_r	2	0.0756	0.03781	23.037	1.027e-10***
p_m:p_e	1	0.0069	0.00693	4.219	0.039991*
Residuals	14150	23.2259	0.00164		

Table A16: ANOVA for the Response C. Significance codes: *** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

To conclude, the ANOVA results confirm that the massive mutation probability does not significantly contribute to explain the variability of any response variable. Consequently, this procedure will be excluded from the algorithm's configuration. In contrast, the remaining five factors—population size, crossover probability, mutation probability, replacement probability, and exchange probability—have a statistically significant impact on performance. To identify the most promising values for each

factor and achieve the best average performance in our genetic algorithm, we apply Tukey's HSD test. As in the previous analysis, a separate model is fitted for each metric to evaluate statistical significance and determine the most adequate parameter levels.

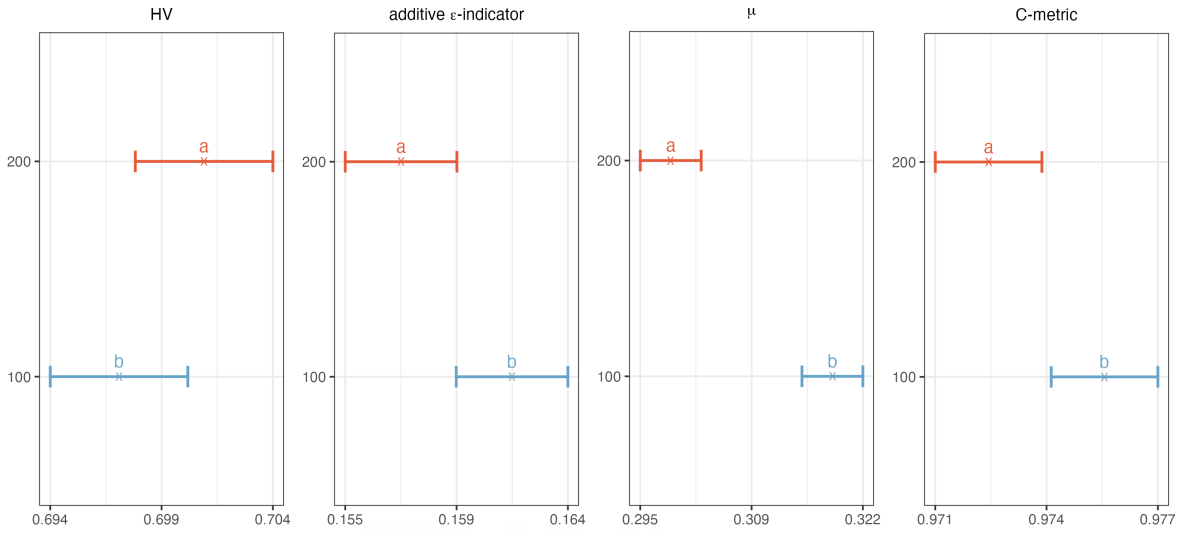


Figure A5: Hypervolume, additive ϵ-indicator, μ-indicator, and C-metric mean plots with Tukey's HSD 95% confidence intervals for population size calibration.

Figure A5 presents the means plot of the four response variables (HV, ϵ-indicator, μ-indicator, and C-metric) with Tukey's HSD 95% confidence intervals for population size. These plots identify the parameter levels associated with statistically superior performance, as indicated by their inclusion in the first treatment group. Specifically, population size has two possible values: 100 and 200. Based on the confidence intervals, we conclude that a population size of 200 yields superior performance, as it consistently falls within the first treatment group across all four statistical tests. Moreover, no inconsistencies are observed among the performance indicators.

In contrast, the results for crossover and mutation probabilities indicate that, in some of the tests performed, no statistically significant differences were found between parameter levels. Specifically, crossover probability has two possible values (0.7 or 0.9), while mutation probability has two levels ($\frac{1}{n}$ or $\frac{2}{n}$), with n being the number of project activities. Figure A6 shows that both crossover probability levels belong to the first treatment group in three of the four statistical tests. Only in the means plot of C we observe that the levels are separated into different groups, with 0.7 belonging to the first group. Therefore, we select a crossover probability of 0.7, as it demonstrates superior performance in one test while showing no significant difference in the others. Similarly, Figure A7 reveals that for two of the four statistical tests, no significant differences exist between the levels of mutation probability. However, for the means plot of the μ-indicator and the C-metric, two distinct treatment groups emerge, with $\frac{1}{n}$ consistently appearing in the first group. Thus, we select $\frac{1}{n}$ as the mutation probability.

Figures A8 and A9 present the results for replacement and exchange probabilities, respectively.

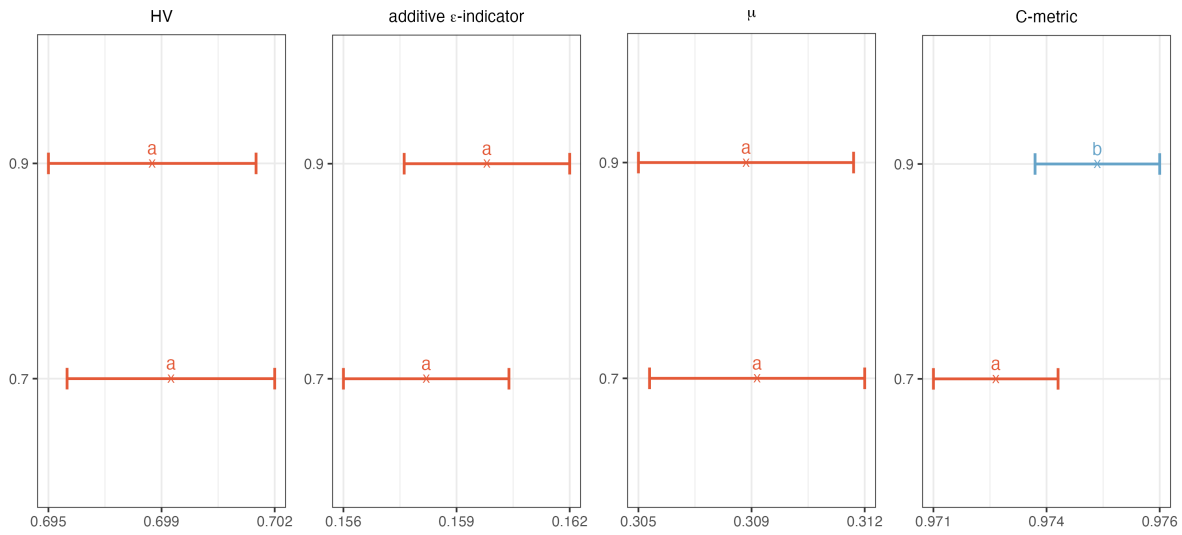


Figure A6: Hypervolume, additive ϵ-indicator, μ-indicator, and C-metric mean plots with Tukey's HSD 95% confidence intervals for crossover probability calibration.

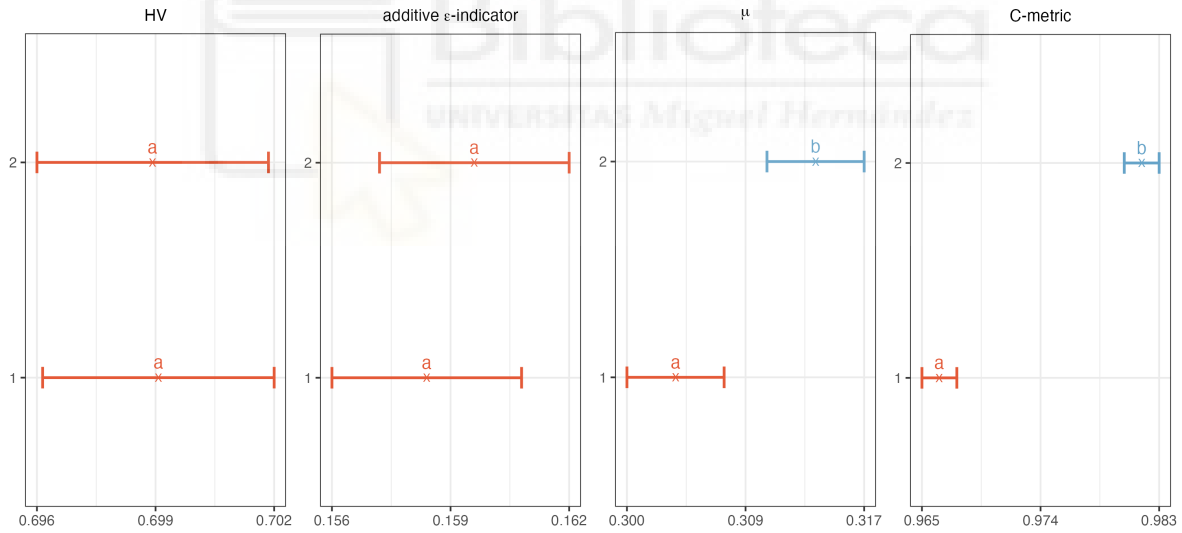


Figure A7: Hypervolume, additive ϵ-indicator, μ-indicator, and C-metric mean plots with Tukey's HSD 95% confidence intervals for mutation probability calibration.

The replacement probability can take values of 0, 0.05 or 0.1, while the exchange probability has two possible values: 0.1 or 0.2. As shown in Figure A8, a replacement probability of 0 consistently belongs to the first treatment group across all four Tukey's HSD tests. Consequently, we set this value, meaning that the replacement procedure will not be performed in our genetic algorithm. Similarly, the selected

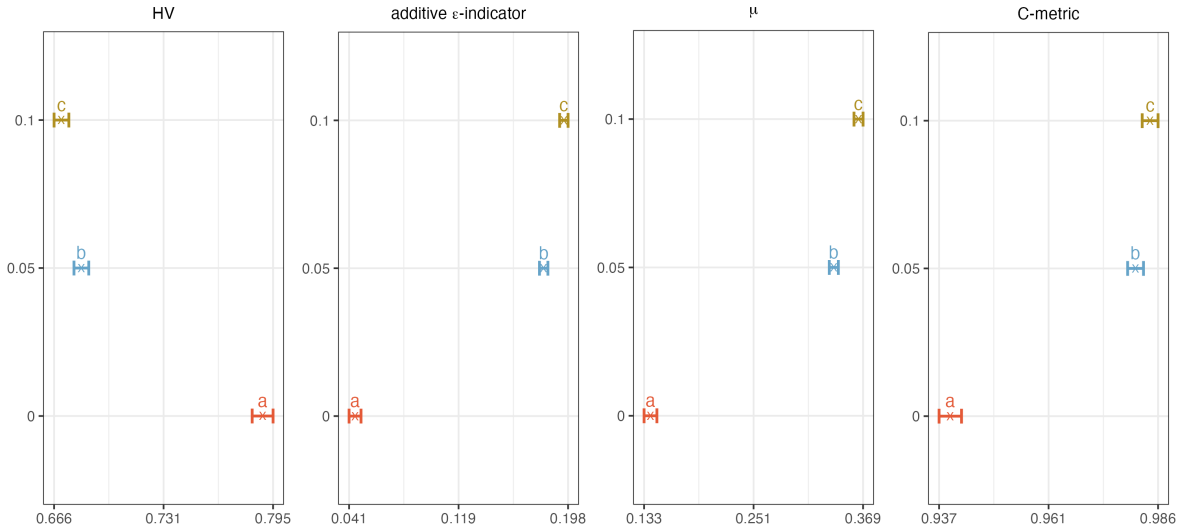


Figure A8: Hypervolume, additive ϵ-indicator, μ-indicator, and C-metric mean plots with Tukey's HSD 95% confidence intervals for replacement probability calibration.

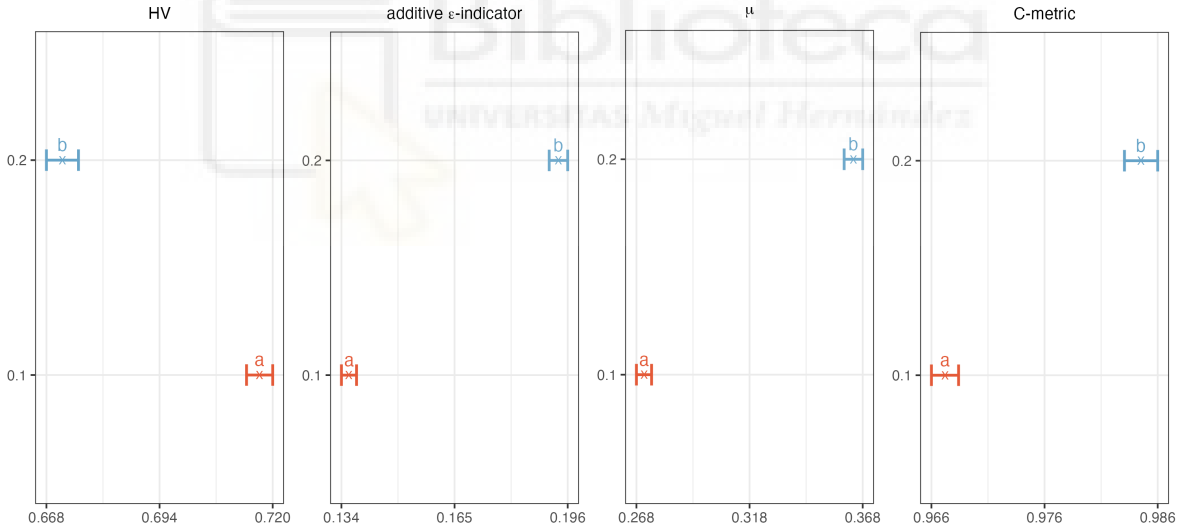


Figure A9: Hypervolume, additive ϵ-indicator, μ-indicator, and C-metric mean plots with Tukey's HSD 95% confidence intervals for exchange probability calibration.

exchange probability is 0.1, as it is included in the first group for all statistical tests. Nonetheless, it is important to note that the exchange procedure is not executed if the replacement probability is set to zero, as the former depends on the latter.

In summary, we select a population size of 200 individuals, a crossover probability of 0.7, a mutation

probability of $\frac{1}{n}$, and a replacement probability of zero. Consequently, the replacement procedure is not implemented in our genetic algorithm. Additionally, the ANOVA results confirm that the massive mutation parameter is not statistically significant; therefore, no further analysis is conducted for this factor.

Appendix B
Full tables of computational results for Chapter 4

In this appendix, we present the full set of results from the computational study conducted in Section 4.5.3. All reported times are given in hours. We begin with the evaluation set derived from the J20 dataset. The table below summarizes the values obtained for all performance indicators by both the exact technique and the genetic algorithm. Instances where AUGMECON failed to compute the exact PF are marked with the symbol (*).

	AUGMECON							Genetic Algorithm							
Problem	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	Time	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	$I_{\epsilon+}$	Time
J203_5	42	0.00 %	1.414	0.089	0.063	0.7	0.1	41	73.33 %	1.39	0.095	0.068	0.674	0.053	0.2
J209_2(*)	47	0.00 %	1.414	0.378	0.267	0.788	3.26	42.67	84.38 %	1.152	0.09	0.078	0.769	0.041	0.19
J209_3	34	0.00 %	1.414	0.18	0.128	0.786	0.15	28.67	73.65 %	1.374	0.346	0.252	0.751	0.072	0.17
J209_5	39	0.00 %	1.414	0.3	0.212	0.793	2.75	27	77.65 %	1.09	0.199	0.183	0.756	0.072	0.18
J2010_2	52	0.00 %	1.401	0.084	0.06	0.654	0.35	44.33	90.40 %	1.367	0.087	0.063	0.627	0.065	0.18
J2010_3	43	0.00 %	1.399	0.391	0.279	0.855	1.05	42.67	98.50 %	1.148	0.111	0.097	0.834	0.032	0.18
J2010_4	29	0.00 %	1.414	0.326	0.231	0.884	0.15	31	95.86 %	1.088	0.118	0.108	0.872	0.016	0.17
J2011_2	70	0.00 %	1.395	0.106	0.076	0.704	0.49	62.33	94.01 %	1.395	0.107	0.076	0.686	0.038	0.17
J2011_3	57	0.00 %	1.384	0.089	0.064	0.727	1.07	42	98.52 %	1.398	0.1	0.071	0.704	0.037	0.18
J2011_4	63	0.00 %	1.414	0.106	0.075	0.712	1.05	55.33	96.36 %	1.385	0.112	0.081	0.683	0.043	0.16
J2012_2	47	0.00 %	1.414	0.326	0.23	0.801	0.22	43	92.16 %	1.356	0.321	0.237	0.785	0.029	0.17
J2012_3	38	0.00 %	1.413	0.231	0.164	0.883	0.22	48.33	63.15 %	1.362	0.195	0.143	0.879	0.015	0.18
J2012_4	48	0.00 %	1.414	0.258	0.183	0.826	0.25	48.67	97.32 %	1.337	0.248	0.186	0.816	0.018	0.17
J2013_2(*)	22	0.00 %	1.414	0.593	0.419	0.855	11.3	27.33	90.31 %	1.242	0.274	0.22	0.826	0.084	0.16
J2013_3(*)	33	3.03 %	1.402	0.245	0.175	0.838	9.87	32	91.07 %	1.309	0.203	0.155	0.781	0.081	0.16
J2013_4(*)	33	0.00 %	1.414	0.205	0.145	0.861	2.01	30.67	96.94 %	1.317	0.268	0.203	0.833	0.056	0.16
J2014_2(*)	24	0.00 %	1.414	0.235	0.166	0.83	5.45	20.33	91.98 %	1.193	0.121	0.102	0.782	0.081	0.15
J2014_3(*)	23	0.00 %	1.412	0.545	0.386	0.886	1.67	18	98.15 %	1.072	0.153	0.142	0.852	0.063	0.16
J2014_4	22	0.00 %	1.408	0.209	0.148	0.878	0.27	20.33	85.43 %	1.396	0.383	0.274	0.853	0.041	0.17
J2015_2	34	0.00 %	1.409	0.271	0.193	0.893	2.44	30.33	88.24 %	1.155	0.279	0.242	0.862	0.046	0.18
J2015_3	40	0.00 %	1.411	0.255	0.181	0.883	1.06	40.33	80.47 %	1.371	0.192	0.14	0.877	0.014	0.18
J2015_4	30	0.00 %	1.409	0.286	0.203	0.89	0.34	29	89.86 %	1.409	0.254	0.18	0.879	0.033	0.17
J2016_2	27	0.00 %	1.147	0.185	0.161	0.941	0.45	27	87.92 %	1.213	0.275	0.227	0.933	0.012	0.17
J2016_3	37	0.00 %	1.389	0.178	0.128	0.862	0.27	35.67	92.68 %	1.381	0.165	0.119	0.846	0.038	0.18
J2016_4	36	0.00 %	1.399	0.253	0.181	0.858	0.46	30.67	96.84 %	1.393	0.253	0.181	0.843	0.021	0.18
J2017_2	52	0.00 %	1.394	0.239	0.172	0.768	0.85	54.67	95.11 %	1.392	0.249	0.179	0.741	0.046	0.17
J2017_3	52	0.00 %	1.414	0.229	0.162	0.789	0.44	56.67	98.86 %	1.407	0.223	0.159	0.777	0.028	0.16
J2017_4	51	0.00 %	1.414	0.235	0.166	0.746	1	32.67	95.94 %	1.381	0.213	0.154	0.722	0.043	0.16
J2018_2	69	0.00 %	1.414	0.144	0.102	0.655	0.32	57	97.11 %	1.383	0.145	0.105	0.637	0.042	0.16
J2018_3	52	0.00 %	1.39	0.224	0.161	0.786	0.15	52.33	94.37 %	1.321	0.185	0.14	0.775	0.02	0.15
J2018_4	43	0.00 %	1.414	0.14	0.099	0.752	0.39	37.33	88.58 %	1.336	0.156	0.117	0.707	0.083	0.15

J2019_2	22	0.00 %	1.414	0.293	0.207	0.755	0.11	24.33	59.33 %	1.406	0.293	0.209	0.748	0.02	0.16
J2019_3	45	0.00 %	1.408	0.245	0.174	0.835	0.23	38.33	79.41 %	1.378	0.248	0.18	0.819	0.035	0.17
J2019_4	55	0.00 %	1.404	0.112	0.08	0.649	0.13	45.33	99.29 %	1.381	0.114	0.082	0.623	0.045	0.16
J2020_2	41	0.00 %	1.414	0.21	0.149	0.766	0.07	43.33	94.67 %	1.395	0.21	0.15	0.752	0.033	0.17
J2020_3	54	0.00 %	1.414	0.143	0.101	0.795	0.11	51	92.33 %	1.4	0.143	0.102	0.78	0.027	0.16
J2020_4	48	0.00 %	1.414	0.228	0.162	0.809	0.16	48.33	85.72 %	1.403	0.228	0.163	0.799	0.024	0.16
J2021_2(*)	34	0.98 %	1.201	0.187	0.155	0.862	9.25	30	89.84 %	1.359	0.225	0.166	0.799	0.087	0.15
J2021_3(*)	35	0.00 %	1.408	0.2	0.142	0.795	4.49	31.33	95.83 %	1.384	0.108	0.078	0.746	0.082	0.16
J2021_4(*)	38	0.00 %	1.414	0.312	0.22	0.845	4.1	30.33	88.15 %	1.384	0.314	0.227	0.788	0.067	0.15
J2022_2	30	0.00 %	1.405	0.198	0.141	0.867	0.58	25.67	97.38 %	1.152	0.207	0.18	0.838	0.05	0.15
J2022_3	32	0.00 %	1.207	0.257	0.213	0.893	0.24	31	93.71 %	1.292	0.237	0.183	0.878	0.027	0.16
J2022_4(*)	35	0.00 %	1.373	0.341	0.248	0.889	2.06	29.67	96.74 %	1.194	0.252	0.211	0.867	0.063	0.17
J2023_2	47	0.00 %	1.404	0.091	0.065	0.778	0.58	41	92.81 %	1.395	0.094	0.067	0.751	0.043	0.16
J2023_3	35	0.00 %	1.387	0.298	0.215	0.888	0.16	29.33	100.00 %	1.409	0.362	0.257	0.872	0.03	0.16
J2023_4	35	0.00 %	1.344	0.094	0.07	0.796	0.22	32.33	96.02 %	1.352	0.11	0.081	0.757	0.062	0.17
J2024_2	33	0.00 %	1.342	0.323	0.241	0.89	0.19	37.33	92.85 %	1.376	0.319	0.232	0.878	0.026	0.18
J2024_3	32	0.00 %	1.363	0.184	0.135	0.85	0.08	27.33	100.00 %	1.393	0.177	0.127	0.827	0.05	0.17
J2024_4	30	0.00 %	1.312	0.212	0.161	0.855	0.1	28.67	87.19 %	1.384	0.164	0.119	0.834	0.046	0.16
J2025_2	35	0.00 %	1.414	0.261	0.184	0.913	0.08	28	65.79 %	1.089	0.267	0.246	0.889	0.045	0.16
J2025_3	44	0.00 %	1.35	0.15	0.111	0.881	0.23	37.67	86.97 %	1.367	0.143	0.105	0.87	0.023	0.15
J2025_4	41	0.00 %	1.414	0.167	0.118	0.802	0.26	37.33	80.36 %	1.4	0.182	0.13	0.783	0.055	0.16
J2026_2	41	0.00 %	1.414	0.387	0.274	0.9	0.08	61.33	64.79 %	1.154	0.135	0.117	0.899	0.01	0.16
J2026_3	32	0.00 %	1.414	0.267	0.189	0.805	0.06	29.33	95.59 %	1.406	0.265	0.189	0.795	0.017	0.17
J2026_4	52	0.00 %	1.414	0.117	0.082	0.799	0.25	59.67	77.15 %	1.409	0.128	0.091	0.793	0.022	0.16
J2027_2	47	0.00 %	1.414	0.251	0.177	0.874	0.11	51	81.98 %	1.414	0.246	0.174	0.865	0.024	0.16
J2027_3	40	0.00 %	1.372	0.272	0.198	0.84	0.1	44.33	78.27 %	1.377	0.273	0.198	0.834	0.016	0.16
J2027_4	33	0.00 %	1.407	0.426	0.303	0.902	0.11	40	95.98 %	1.337	0.414	0.31	0.891	0.025	0.17
J2028_2	36	0.00 %	1.412	0.158	0.112	0.839	0.08	36.67	93.79 %	1.393	0.159	0.115	0.826	0.031	0.16
J2028_3	31	0.00 %	1.407	0.181	0.129	0.876	0.08	29.67	78.72 %	1.393	0.157	0.113	0.866	0.026	0.16
J2028_4	39	0.00 %	1.411	0.272	0.192	0.826	0.1	37	82.31 %	1.382	0.276	0.2	0.807	0.032	0.15
J2029_2(*)	43	0.00 %	1.409	0.093	0.066	0.818	1.45	43.67	99.28 %	1.359	0.133	0.098	0.777	0.077	0.15
J2029_3(*)	35	0.00 %	1.269	0.294	0.232	0.86	2.2	31	95.78 %	1.358	0.21	0.155	0.81	0.077	0.15
J2029_4(*)	46	0.00 %	1.414	0.096	0.068	0.798	6.42	41	95.12 %	1.382	0.102	0.074	0.746	0.078	0.14
J2030_2	27	0.00 %	1.414	0.231	0.164	0.911	0.16	30	91.14 %	1.362	0.203	0.149	0.901	0.035	0.14
J2030_3	36	0.00 %	1.283	0.111	0.087	0.822	0.58	27.33	89.15 %	1.349	0.199	0.148	0.766	0.096	0.15
J2030_4	39	0.00 %	1.41	0.177	0.126	0.863	0.67	38	98.29 %	1.369	0.196	0.143	0.831	0.076	0.16
J2031_2	24	0.00 %	1.408	0.402	0.285	0.862	0.28	23	84.32 %	1.316	0.32	0.243	0.845	0.039	0.15
J2031_3	35	0.00 %	1.286	0.142	0.111	0.835	0.15	32.67	93.94 %	1.338	0.178	0.133	0.808	0.055	0.14
J2031_4	33	0.00 %	1.402	0.299	0.213	0.847	0.28	28.67	89.87 %	1.352	0.239	0.177	0.809	0.062	0.16
J2032_2	38	0.00 %	1.4	0.192	0.137	0.844	0.09	35.33	94.49 %	1.409	0.192	0.136	0.825	0.052	0.15
J2032_3	35	0.00 %	1.372	0.31	0.226	0.864	0.15	30.67	85.27 %	1.381	0.483	0.35	0.853	0.029	0.16
J2032_4	32	0.00 %	1.377	0.253	0.183	0.886	0.12	36.67	93.80 %	1.395	0.277	0.198	0.872	0.033	0.14
J2033_2	37	0.00 %	1.414	0.179	0.127	0.872	1.13	34.67	94.39 %	1.316	0.25	0.19	0.862	0.026	0.18
J2033_3	51	0.00 %	1.414	0.116	0.082	0.731	1.46	43.33	90.76 %	1.297	0.116	0.089	0.699	0.052	0.16
J2033_4	49	0.00 %	1.392	0.105	0.075	0.836	0.42	38.67	74.39 %	1.298	0.211	0.163	0.815	0.04	0.18
J2034_2	33	0.00 %	1.414	0.316	0.224	0.896	0.53	31.67	73.21 %	1.149	0.173	0.151	0.888	0.019	0.18
J2034_3	36	0.00 %	1.414	0.262	0.185	0.905	0.76	45	87.91 %	1.297	0.196	0.151	0.892	0.028	0.17
J2034_4	24	0.00 %	1.407	0.376	0.267	0.929	0.16	30.67	92.49 %	1.233	0.26	0.211	0.924	0.009	0.18
J2035_2	47	0.00 %	1.359	0.209	0.154	0.874	0.09	61.33	87.47 %	1.383	0.225	0.162	0.868	0.015	0.18
J2035_3	39	0.00 %	1.414	0.171	0.121	0.82	0.37	28.67	77.30 %	1.136	0.105	0.092	0.808	0.034	0.17
J2035_4	32	0.00 %	1.386	0.275	0.198	0.922	1.15	31.67	48.01 %	1.144	0.458	0.4	0.908	0.028	0.19
J2037_2(*)	32	0.00 %	1.414	0.205	0.145	0.856	8.7	29.33	96.62 %	1.288	0.365	0.283	0.815	0.063	0.17
J2037_3(*)	34	0.98 %	1.4	0.279	0.199	0.831	7.23	33.67	92.20 %	1.199	0.119	0.099	0.786	0.079	0.18
J2037_4	38	0.00 %	1.414	0.31	0.219	0.885	2.03	35	99.10 %	1.178	0.201	0.17	0.83	0.057	0.18
J2038_2(*)	34	0.00 %	1.333	0.214	0.161	0.905	4.15	31.33	90.62 %	1.265	0.363	0.287	0.872	0.045	0.17
J2038_3(*)	33	0.00 %	1.414	0.376	0.266	0.9	3.12	23	89.59 %	0.96	0.323	0.337	0.885	0.034	0.18
J2038_4(*)	31	0.00 %	1.414	0.288	0.204	0.839	3.73	24.33	82.67 %	1.206	0.338	0.28	0.821	0.027	0.18
J2039_2	31	0.00 %	1.414	0.55	0.389	0.94	0.39	34	100.00 %	0.637	0.12	0.188	0.918	0.031	0.19
J2039_3(*)	31	0.00 %	1.194	0.288	0.241	0.928	3.6	34	82.79 %	1.165	0.227	0.194	0.918	0.019	0.18
J2039_4	27	0.00 %	1.414	0.484	0.342	0.893	0.24	22	85.55 %	1.156	0.352	0.305	0.878	0.035	0.19
J2040_2	27	0.00 %	1.414	0.613	0.434	0.944	0.4	24.67	76.21 %	1.079	0.225	0.208	0.939	0.018	0.18
J2040_3	19	0.00 %	1.373	0.734	0.535	0.952	0.52	17.67	67.69 %	0.8	0.227	0.284	0.946	0.018	0.19
J2040_4	23	0.00 %	1.351	0.592	0.438	0.942	1.94	27.67	63.69 %	0.941	0.239	0.254	0.94	0.017	0.18
J2041_2(*)	47	0.00 %	1.383	0.1	0.072	0.727	7.2	38.67	93.18 %	1.323	0.113	0.086	0.676	0.099	0.15
J2041_3	42	0.00 %	1.414	0.13	0.092	0.804	2.11	38.67	96.45 %	1.336	0.106	0.079	0.774	0.065	0.16

J2041_4	53	0.00 %	1.414	0.165	0.116	0.809	0.71	38	88.77 %	1.378	0.202	0.147	0.787	0.043	0.17
J2042_2	48	0.00 %	1.414	0.203	0.143	0.867	0.31	43.67	87.26 %	1.279	0.118	0.092	0.859	0.018	0.17
J2042_3	31	0.00 %	1.414	0.244	0.173	0.894	0.2	37.67	82.37 %	1.124	0.172	0.153	0.879	0.029	0.19
J2042_4	41	0.00 %	1.376	0.232	0.169	0.823	1	36.33	89.35 %	1.362	0.275	0.202	0.78	0.072	0.18
J2043_2(*)	45	0.00 %	1.409	0.309	0.22	0.852	1.22	38.33	90.71 %	1.287	0.265	0.206	0.816	0.057	0.17
J2043_3	52	0.00 %	1.414	0.246	0.174	0.842	1.29	48.67	95.30 %	1.174	0.099	0.084	0.832	0.02	0.17
J2043_4	46	0.00 %	1.414	0.164	0.116	0.812	0.21	44.67	97.18 %	1.218	0.107	0.088	0.792	0.038	0.17
J2044_2	32	0.00 %	1.414	0.387	0.274	0.905	0.14	38.33	87.62 %	1.114	0.135	0.121	0.896	0.014	0.17
J2044_3	41	0.00 %	1.414	0.333	0.235	0.916	0.33	54	82.78 %	1.07	0.132	0.124	0.908	0.022	0.17
J2044_4	50	0.00 %	1.414	0.257	0.182	0.869	0.38	54.33	92.64 %	1.207	0.163	0.135	0.861	0.015	0.18
J2045_2(*)	37	0.00 %	1.414	0.182	0.129	0.825	2.5	33.33	89.81 %	1.339	0.116	0.087	0.802	0.054	0.19
J2045_3(*)	38	0.00 %	1.414	0.226	0.16	0.817	14.31	30.67	95.96 %	1.224	0.195	0.159	0.777	0.072	0.16
J2045_4(*)	36	0.00 %	1.332	0.224	0.168	0.85	11.08	29.33	89.58 %	1.307	0.192	0.147	0.773	0.088	0.16
J2046_2	34	0.00 %	1.384	0.268	0.193	0.905	0.45	35.33	84.51 %	1.238	0.238	0.192	0.894	0.022	0.17
J2046_3(*)	42	0.00 %	1.414	0.29	0.205	0.883	9.77	28.33	98.96 %	1.022	0.111	0.109	0.853	0.058	0.16
J2046_4	32	0.00 %	1.414	0.394	0.278	0.927	2.71	34.33	99.02 %	1.117	0.281	0.251	0.914	0.031	0.18
J2047_2	30	0.00 %	1.378	0.293	0.213	0.9	0.37	25.67	83.42 %	1.029	0.197	0.191	0.882	0.027	0.16
J2047_3	34	0.00 %	1.414	0.533	0.377	0.934	0.21	34.33	82.09 %	1.235	0.43	0.348	0.928	0.015	0.17
J2047_4	22	0.00 %	1.414	0.334	0.236	0.868	0.13	21.33	91.02 %	1.047	0.155	0.148	0.856	0.041	0.17
J2048_2	41	0.00 %	1.349	0.2	0.148	0.902	0.27	39.67	88.39 %	1.399	0.23	0.165	0.895	0.018	0.17
J2048_3	36	0.00 %	1.387	0.255	0.184	0.879	0.27	32.67	88.88 %	1.351	0.238	0.176	0.861	0.045	0.18
J2048_4	27	0.00 %	1.414	0.271	0.191	0.872	0.43	26	90.10 %	1.201	0.241	0.2	0.839	0.05	0.17
J2049_2	54	0.00 %	1.414	0.088	0.062	0.779	0.33	49.33	75.33 %	1.411	0.096	0.068	0.767	0.03	0.17
J2049_3	54	0.00 %	1.399	0.117	0.084	0.812	1.4	51	99.39 %	1.246	0.116	0.093	0.792	0.034	0.17
J2049_4	28	0.00 %	1.276	0.247	0.194	0.884	0.87	21.33	89.09 %	1.235	0.333	0.27	0.866	0.029	0.17
J2050_2	44	0.00 %	1.414	0.095	0.067	0.736	0.6	35.33	97.21 %	1.384	0.094	0.068	0.704	0.046	0.16
J2050_3	42	0.00 %	1.402	0.214	0.153	0.833	0.33	36.67	77.86 %	1.299	0.178	0.137	0.822	0.028	0.17
J2050_4	58	0.00 %	1.41	0.216	0.154	0.823	0.84	58.33	97.77 %	1.323	0.165	0.124	0.809	0.023	0.16
J2051_2	46	0.00 %	1.414	0.144	0.102	0.705	0.1	36.33	97.30 %	1.39	0.139	0.1	0.67	0.051	0.18
J2051_3	37	0.00 %	1.407	0.148	0.105	0.806	0.11	38.67	97.43 %	1.401	0.143	0.102	0.788	0.041	0.15
J2051_4	62	0.00 %	1.403	0.09	0.064	0.769	2.07	62	90.47 %	1.351	0.106	0.079	0.751	0.045	0.16
J2052_2	57	0.00 %	1.414	0.24	0.17	0.804	0.28	51	95.54 %	1.304	0.187	0.143	0.793	0.023	0.17
J2052_3	54	0.00 %	1.322	0.117	0.088	0.822	0.25	50	87.55 %	1.339	0.15	0.112	0.812	0.031	0.16
J2052_4	40	0.00 %	1.397	0.253	0.181	0.869	0.08	37.33	95.59 %	1.108	0.109	0.099	0.851	0.029	0.17
J2053_2(*)	33	0.00 %	1.393	0.209	0.15	0.818	11.53	26	100.00 %	1.284	0.305	0.238	0.75	0.09	0.16
J2053_3(*)	28	0.00 %	1.253	0.115	0.092	0.799	14.95	32.33	95.82 %	1.391	0.228	0.164	0.721	0.114	0.16
J2053_4(*)	42	0.00 %	1.414	0.173	0.122	0.816	9.49	41.33	89.49 %	1.318	0.109	0.082	0.784	0.066	0.16
J2054_2	44	0.00 %	1.379	0.182	0.132	0.868	0.41	37	95.49 %	1.405	0.159	0.113	0.843	0.049	0.16
J2054_3	37	0.00 %	1.414	0.297	0.21	0.842	1.21	28.67	96.88 %	1.176	0.158	0.134	0.791	0.081	0.16
J2054_4	31	0.00 %	1.357	0.365	0.269	0.93	0.54	32.67	98.99 %	1.393	0.386	0.277	0.914	0.034	0.16
J2055_2	32	0.00 %	1.373	0.245	0.178	0.826	1.38	31	98.85 %	1.219	0.116	0.095	0.772	0.089	0.15
J2055_3	30	0.00 %	1.414	0.646	0.457	0.943	1.51	29	91.98 %	0.715	0.145	0.202	0.923	0.024	0.16
J2055_4	32	0.00 %	1.412	0.26	0.184	0.904	0.16	33.33	95.01 %	1.287	0.206	0.16	0.892	0.023	0.16
J2056_2	47	0.00 %	1.406	0.3	0.214	0.881	0.34	35.67	96.38 %	1.187	0.288	0.243	0.861	0.029	0.17
J2056_3	35	0.00 %	1.402	0.298	0.213	0.834	0.97	25	93.53 %	1.407	0.31	0.221	0.792	0.06	0.17
J2056_4	40	0.00 %	1.414	0.255	0.181	0.901	0.31	36.67	99.05 %	1.289	0.255	0.198	0.883	0.028	0.17
J2057_2	40	0.00 %	1.405	0.09	0.064	0.755	0.28	36.33	97.30 %	1.399	0.097	0.069	0.728	0.041	0.15
J2057_3	58	0.00 %	1.371	0.09	0.066	0.739	0.37	45.33	94.90 %	1.358	0.099	0.073	0.718	0.064	0.16
J2057_4	41	0.00 %	1.413	0.213	0.151	0.856	0.11	35.67	87.98 %	1.366	0.199	0.146	0.839	0.032	0.17
J2058_2	37	0.00 %	1.397	0.177	0.127	0.745	0.17	33.67	100.00 %	1.351	0.183	0.136	0.715	0.051	0.16
J2058_3	61	0.00 %	1.408	0.144	0.103	0.753	0.14	53.33	100.00 %	1.39	0.145	0.104	0.728	0.039	0.16
J2058_4	35	0.00 %	1.414	0.13	0.092	0.674	0.23	28	89.48 %	1.397	0.13	0.093	0.654	0.037	0.15
J2059_2	59	0.00 %	1.404	0.15	0.107	0.736	0.19	46.67	93.70 %	1.385	0.15	0.108	0.721	0.037	0.16
J2059_3	52	0.00 %	1.409	0.07	0.05	0.692	0.15	49.67	95.37 %	1.393	0.08	0.057	0.682	0.026	0.16
J2059_4	43	0.00 %	1.393	0.151	0.109	0.78	0.08	38.67	100.00 %	1.398	0.16	0.115	0.759	0.054	0.16
J2060_2	36	0.00 %	1.414	0.375	0.265	0.88	0.28	24	74.64 %	1.246	0.375	0.301	0.854	0.06	0.16
J2060_3	53	0.00 %	1.388	0.082	0.059	0.767	0.2	47.33	97.26 %	1.383	0.097	0.07	0.746	0.043	0.17
J2060_4	41	0.00 %	1.386	0.13	0.093	0.732	0.14	28.33	95.38 %	1.342	0.156	0.117	0.705	0.057	0.16
J2061_2(*)	39	0.00 %	1.414	0.155	0.109	0.77	3.99	33	97.03 %	1.37	0.158	0.115	0.691	0.108	0.16
J2061_3	38	0.00 %	1.414	0.142	0.1	0.831	1.47	41	89.62 %	1.313	0.093	0.071	0.801	0.068	0.15
J2061_4	39	0.00 %	1.414	0.109	0.077	0.829	0.86	41.33	95.19 %	1.384	0.099	0.072	0.799	0.051	0.14
J2062_2	37	0.00 %	1.396	0.188	0.135	0.827	0.52	29	93.28 %	1.365	0.176	0.129	0.781	0.071	0.16
J2062_3	49	0.00 %	1.414	0.108	0.077	0.846	0.75	49.67	94.02 %	1.284	0.097	0.075	0.833	0.034	0.16
J2062_4	33	0.00 %	1.414	0.272	0.193	0.86	0.2	32	93.93 %	1.228	0.127	0.103	0.848	0.026	0.15
J2063_2	43	0.00 %	1.414	0.154	0.109	0.83	0.31	42	98.43 %	1.382	0.144	0.104	0.812	0.049	0.16

J2063_3	33	0.00 %	1.385	0.148	0.107	0.819	0.22	29	96.66 %	1.403	0.131	0.093	0.802	0.039	0.16
J2063_4	31	0.00 %	1.407	0.119	0.084	0.813	0.09	32	97.92 %	1.403	0.138	0.099	0.785	0.059	0.15
J2064_2	41	0.00 %	1.403	0.138	0.098	0.842	0.25	38	94.82 %	1.347	0.129	0.096	0.822	0.035	0.17
J2064_3	71	0.00 %	1.412	0.104	0.074	0.784	0.41	58	97.74 %	1.401	0.117	0.084	0.762	0.033	0.17
J2064_4	30	0.00 %	1.339	0.108	0.081	0.818	0.17	29	100.00 %	1.407	0.14	0.1	0.794	0.047	0.15
Average	39.49	0.03 %	1.392	0.235	0.169	0.835	1.495	36.78	90.20 %	1.292	0.198	0.153	0.812	0.043	0.166

Table B1: Experimental results for the J20 evaluation dataset.

Next, we present the computational results from the experimental study conducted on the MMLIB50 evaluation dataset. For this set of instances, the exact algorithm failed to compute any exact PF.

	AUGMECON							**Genetic Algorithm**							
Problem	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	Time	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	$I_{\epsilon+}$	Time
J502_1	45	0.00 %	1.39	0.15	0.108	0.854	35.71	71.33	100.00 %	1.227	0.091	0.074	0.795	0.061	0.64
J504_1	37	0.00 %	1.414	0.21	0.149	0.881	22.67	59.33	100.00 %	1.157	0.117	0.101	0.835	0.055	0.57
J506_1	54	0.00 %	1.414	0.213	0.15	0.809	15.05	60.33	100.00 %	1.339	0.124	0.093	0.748	0.068	0.63
J508_1	31	0.00 %	1.254	0.201	0.16	0.87	20.34	57.33	97.20 %	1.332	0.189	0.142	0.79	0.073	0.45
J5010_1	25	0.00 %	1.234	0.143	0.116	0.923	16.39	45	100.00 %	1.261	0.212	0.168	0.845	0.084	0.57
J5012_1	35	1.90 %	1.408	0.216	0.154	0.791	30.43	62.33	94.50 %	1.332	0.088	0.066	0.727	0.067	0.46
J5014_1	48	0.00 %	1.414	0.372	0.263	0.815	30.32	79.67	100.00 %	1.063	0.065	0.061	0.76	0.051	0.75
J5016_1	53	0.00 %	1.412	0.128	0.091	0.799	20.73	67.67	100.00 %	1.176	0.086	0.073	0.732	0.066	0.58
J5018_1	73	0.00 %	1.29	0.057	0.044	0.736	23.82	68.67	100.00 %	1.374	0.139	0.101	0.672	0.063	0.55
J5020_1	41	0.00 %	1.414	0.135	0.095	0.86	22.13	73.67	100.00 %	1.127	0.069	0.062	0.813	0.057	0.64
J5022_1	41	0.00 %	1.353	0.156	0.116	0.858	38.33	59	100.00 %	1.225	0.094	0.077	0.783	0.066	0.6
J5024_1	47	0.00 %	1.414	0.082	0.058	0.829	15.85	77.33	100.00 %	1.36	0.087	0.064	0.78	0.062	0.43
J5026_1	42	0.00 %	1.414	0.204	0.144	0.828	15.86	70.33	98.59 %	1.058	0.092	0.087	0.775	0.06	0.7
J5028_1	68	0.00 %	1.414	0.091	0.064	0.671	35.81	74.67	100.00 %	1.314	0.099	0.076	0.597	0.083	0.67
J5030_1	52	0.00 %	1.414	0.155	0.11	0.796	17.76	80.67	100.00 %	1.202	0.095	0.079	0.741	0.059	0.54
J5032_1	50	0.00 %	1.414	0.201	0.142	0.821	20.03	78	100.00 %	1.221	0.091	0.075	0.74	0.072	0.61
J5034_1	47	0.00 %	1.412	0.146	0.103	0.815	21.31	57.33	100.00 %	1.207	0.158	0.131	0.728	0.087	0.56
J5036_1	63	0.00 %	1.254	0.081	0.065	0.804	17.65	77	100.00 %	1.357	0.122	0.09	0.746	0.061	0.54
J5038_1	38	0.00 %	1.414	0.399	0.282	0.844	30.97	86.33	99.63 %	1.102	0.095	0.086	0.808	0.049	0.55
J5040_1	32	0.00 %	1.414	0.164	0.116	0.76	30.48	75.33	95.96 %	1.24	0.113	0.091	0.705	0.066	0.42
J5042_1	42	0.00 %	1.414	0.128	0.091	0.809	34.44	64.33	100.00 %	1.301	0.146	0.112	0.764	0.074	0.44
J5044_1	27	0.00 %	1.256	0.192	0.153	0.749	36.03	70.33	92.50 %	1.368	0.106	0.077	0.686	0.073	0.45
J5046_1	15	0.00 %	1.154	0.344	0.298	0.764	24.03	65.67	79.73 %	1.383	0.104	0.075	0.721	0.072	0.34
J5048_1	14	0.00 %	1.412	0.592	0.419	0.701	19.3	73.33	69.02 %	1.355	0.128	0.095	0.696	0.065	0.41
J5050_1	19	0.00 %	1.414	0.42	0.297	0.724	16.2	82.67	53.54 %	1.132	0.072	0.063	0.717	0.055	0.6
J5052_1	40	0.00 %	1.414	0.261	0.185	0.823	33.43	65	89.76 %	1.17	0.093	0.079	0.784	0.064	0.48
J5054_1	45	0.00 %	1.293	0.188	0.145	0.769	33.87	60	100.00 %	1.339	0.123	0.092	0.716	0.068	0.44
J5056_1	17	5.88 %	1.254	0.295	0.235	0.701	26.08	71.67	67.89 %	1.342	0.073	0.054	0.696	0.069	0.49
J5058_1	17	0.00 %	1.098	0.216	0.197	0.813	21.28	69.33	73.61 %	1.289	0.078	0.061	0.775	0.079	0.44
J5060_1	44	0.00 %	1.303	0.234	0.18	0.815	40.34	83.33	92.15 %	1.374	0.105	0.076	0.771	0.066	0.37
J5062_1	47	0.00 %	1.414	0.167	0.118	0.826	28.57	65	100.00 %	1.115	0.148	0.133	0.771	0.064	0.53
J5064_1	53	0.00 %	1.414	0.131	0.093	0.787	16.52	73.33	100.00 %	1.191	0.102	0.085	0.737	0.075	0.52
J5066_1	44	0.00 %	1.414	0.155	0.11	0.852	28.64	67	100.00 %	1.211	0.095	0.078	0.809	0.062	0.45
J5068_1	17	0.00 %	1.414	0.304	0.215	0.867	19.02	56.67	99.45 %	1.156	0.128	0.111	0.811	0.071	0.48
J5070_1	30	0.00 %	1.412	0.238	0.169	0.89	18.14	54.67	100.00 %	1.142	0.174	0.152	0.824	0.08	0.46
J5072_1	35	0.00 %	1.41	0.163	0.116	0.881	28.31	46.33	100.00 %	1.235	0.133	0.108	0.814	0.09	0.39
J5074_1	10	16.67 %	1.157	0.389	0.336	0.784	30.61	54	28.96 %	1.144	0.103	0.09	0.804	0.046	0.42
J5076_1	7	9.52 %	1.111	0.358	0.322	0.755	33.09	53.67	31.85 %	1.256	0.13	0.104	0.773	0.051	0.32
J5078_1	16	0.00 %	1.38	0.255	0.185	0.836	35.08	59.67	91.20 %	1.264	0.129	0.102	0.795	0.06	0.37
J5080_1	5	13.33 %	1.107	0.307	0.278	0.654	25.01	61.67	44.58 %	1.348	0.124	0.092	0.649	0.081	0.32
J5082_1	7	9.52 %	1.074	0.351	0.327	0.82	18.56	53.67	43.93 %	1.366	0.268	0.196	0.83	0.047	0.32
J5084_1	5	0.00 %	1.095	0.387	0.354	0.706	27.13	35.33	33.09 %	1.384	0.187	0.135	0.736	0.049	0.27
J5086_1	6	0.00 %	0.752	0.265	0.352	0.718	25.11	63.67	25.78 %	1.23	0.109	0.089	0.84	0.057	0.36
J5088_1	15	0.00 %	1.414	0.477	0.337	0.872	20.64	42	92.00 %	1.06	0.285	0.269	0.832	0.089	0.35
J5090_1	21	0.00 %	1.362	0.736	0.54	0.809	47.74	51.67	72.01 %	1.298	0.178	0.137	0.801	0.062	0.33

J5092_1	8	0.00 %	1.414	0.487	0.344	0.85	32.22	44.67	57.68 %	1.077	0.17	0.158	0.839	0.067	0.38
J5094_1	6	0.00 %	0.962	0.493	0.512	0.86	21.51	38.33	52.79 %	1.106	0.347	0.314	0.864	0.093	0.33
J5096_1	15	0.00 %	0.928	0.175	0.189	0.837	35.1	55	77.05 %	1.196	0.192	0.161	0.851	0.069	0.29
J5098_1	10	0.00 %	0.912	0.28	0.307	0.87	14.26	58	11.35 %	1.009	0.156	0.154	0.898	0.046	0.51
J50100_1	25	0.00 %	1.414	0.304	0.215	0.894	28.9	47.67	91.68 %	1.039	0.136	0.131	0.866	0.04	0.42
J50102_1	13	0.00 %	0.918	0.125	0.136	0.85	25.22	48.33	69.50 %	1.276	0.141	0.11	0.85	0.07	0.36
J50104_1	10	0.00 %	1.11	0.236	0.212	0.93	42.64	37.67	89.00 %	1.112	0.271	0.244	0.892	0.076	0.36
J50106_1	14	0.00 %	1.132	0.353	0.312	0.885	22	41.33	75.24 %	1.172	0.165	0.141	0.855	0.078	0.35
J50108_1	24	0.00 %	1.364	0.164	0.12	0.882	26.14	39.33	100.00 %	1.18	0.224	0.19	0.806	0.077	0.35
Average	30.46	1.05 %	1.29	0.253	0.196	0.814	26.237	61.77	83.17 %	1.228	0.136	0.111	0.778	0.067	0.466

Table B2: Experimental results for the MMLIB50 evaluation dataset.

Lastly, we present the results obtained for the MMLIB100 evaluation dataset. Once again, the AUGMECON method was unable to compute any exact PF for this set of instances. Specifically, there are six instances where the exact method could produce only a single feasible solution. In these cases, certain performance indicators could not be calculated, and the corresponding values are represented with the symbol "-".

	AUGMECON							**Genetic Algorithm**							
Problem	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	Time	OVNG	C	$\mathcal{M}_3^*$	Γ	μ	HV	$I_{\epsilon+}$	Time
J1001_1	1	0.00 %	-	-	-	0.000	20.43	89.67	0.00 %	1.147	0.076	0.067	0.709	0.052	2.17
J10013_1	3	0.00 %	0.901	0.739	0.82	0.874	24.48	37.33	79.42 %	1.109	0.274	0.247	0.816	0.09	2.18
J10025_1	9	0.00 %	0.657	0.142	0.217	0.636	59.59	73.33	45.79 %	1.165	0.108	0.093	0.608	0.101	2.66
J10037_1	2	0.00 %	0.881	0.741	0.841	0.646	66.31	80.33	51.77 %	1.312	0.155	0.118	0.632	0.163	2
J10049_1	1	0.00 %	-	-	-	0.000	77.32	115	0.00 %	1.126	0.055	0.049	0.646	0.084	1.67
J10061_1	1	0.00 %	-	-	-	0.000	106.85	61.67	0.00 %	1.189	0.278	0.234	0.682	0.05	1.95
J10073_1	1	0.00 %	-	-	-	0.688	108.41	84	44.80 %	1.287	0.088	0.069	0.712	0.145	1.37
J10087_1	4	0.00 %	0.179	0.066	0.369	0.179	76.56	74.67	2.11 %	1.276	0.104	0.082	0.716	0.031	0.99
J10098_1	1	0.00 %	-	-	-	0.000	107.12	84.33	0.00 %	1.115	0.059	0.052	0.753	0.067	1.38
J100108_1	1	0.00 %	-	-	-	0.022	89.68	46	1.25 %	1.302	0.353	0.271	0.726	0.114	1
Average	2.40	0.00 %	0.655	0.422	0.562	0.305	73.675	74.63	22.51 %	1.203	0.155	0.129	0.700	0.090	1.737

Table B3: Experimental results for the MMLIB100 evaluation dataset.

References

Afshar-Nadjafi, B. and Majlesi, M. (2014). Resource constrained project scheduling problem with setup times after preemptive processes. *Computers & Chemical Engineering*, 69:16–25. https://doi.org/10.1016/j.compchemeng.2014.06.012.

Alba, E. and Dorronsoro, B. (2008). On the effects of structuring the population. In *Cellular Genetic Algorithms*, pages 37–46. Springer US, Boston, MA. https://doi.org/10.1007/978-0-387-77610-1_3.

Alcaraz, J. (2024). Redesigning a NSGA-II metaheuristic for the bi-objective support vector machine with feature selection. *Computers & Operations Research*, 172:106821. https://doi.org/10.1016/j.cor.2024.106821.

Alcaraz, J., Anton-Sanchez, L., and Saldanha-da-Gama, F. (2022). Bi-objective resource-constrained project scheduling problem with time-dependent resource costs. *Journal of Manufacturing Systems*, 63:506–523. https://doi.org/10.1016/j.jmsy.2022.05.002.

Alcaraz, J. and Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research*, 102:83–109. https://doi.org/10.1023/A:1010949931021.

Alcaraz, J. and Maroto, C. (2006). A hybrid genetic algorithm based on intelligent encoding for project scheduling. In Józefowska, J. and Weglarz, J., editors, *Perspectives in Modern Project Scheduling*, volume 92, pages 249–274. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-33768-5_10.

Alcaraz, J., Maroto, C., and Ruiz, R. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *The Journal of the Operational Research Society*, 54(6):614–626. https://doi.org/10.1057/palgrave.jors.2601563.

Arnold, K., Gosling, J., and Holmes, D. (2005). *The Java programming language*. Addison Wesley Professional, 4 edition. ISBN 978-0321349804.

Artigues, C., Leus, R., and Talla Nobibon, F. (2013). Robust optimization for resource-constrained project scheduling with uncertain activity durations. *Flexible Services and Manufacturing Journal*, 25:175–205. https://doi.org/10.1007/s10696-012-9147-2.

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., and Salomon, L. (2021). Performance indicators in multiobjective optimization. *European Journal of Operational Research*, 292(2):397–422. https://doi.org/10.1016/j.ejor.2020.11.016.

Audet, C., Savard, G., and Zghal, W. (2008). Multiobjective optimization through a series of single-objective formulations. *SIAM Journal on Optimization*, 19(1):188–210. https://doi.org/10.1137/060677513.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Oxford, UK.

Baker, K. R. (1974). *Introduction to Sequencing and Scheduling*. John Wiley & Sons, Inc., New York, USA.

Ballestín, F., Barrios, A., and Valls, V. (2011). An evolutionary algorithm for the resource-constrained project scheduling problem with minimum and maximum time lags. *Journal of Scheduling*, 14:391–406. https://doi.org/10.1007/s10951-009-0125-9.

Ballestín, F. and Blanco, R. (2015). Theoretical and practical fundamentals. In Schwindt, C. and Zimmermann, J., editors, *Handbook on Project Management and Scheduling Vol. 1*, pages 411–427. Springer, Cham. ISBN 978-3-319-05443-8, https://doi.org/10.1007/978-3-319-05443-8_19.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust optimization*, volume 28. Princeton University Press. ISBN 9780691143682.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 99:238–252. https://doi.org/10.1007/BF01386316.

Bendotti, P., Chrétienne, P., Fouilhoux, P., and Pass-Lanneau, A. (2023). The anchor-robust project scheduling problem. *Operations Research*, 71(6):2267–2290. https://doi.org/10.1287/opre.2022.2315.

Bertsimas, D. and den Hertog, D. (2022). *Robust and Adaptive Optimization*. Dynamic Ideas LLC. ISBN 978-1-7337-8852-6.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53. https://doi.org/10.1287/opre.1030.0065.

Bianco, L. and Caramia, M. (2012). An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations. *European Journal of Operational Research*, 219(1):73–85. https://doi.org/10.1016/j.ejor.2011.12.019.

Blazewicz, J., Lenstra, J., and Kan, A. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24. https://doi.org/10.1016/0166-218X(83)90012-4.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308. https://doi.org/10.1145/937503.937505.

Boctor, F. F. (1996). Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34(8):2335–2351. https://doi.org/10.1080/00207549608905028.

Bold, M. and Goerigk, M. (2021). A compact reformulation of the two-stage robust resource-constrained project scheduling problem. *Computers & Operations Research*, 130:105232. https://doi.org/10.1016/j.cor.2021.105232.

Bold, M. and Goerigk, M. (2022a). A faster exact method for solving the robust multi-mode resource-constrained project scheduling problem. *Operations Research Letters*, 50(5):581–587. https://doi.org/10.1016/j.orl.2022.08.003.

Bold, M. and Goerigk, M. (2022b). Investigating the recoverable robust single machine scheduling problem under interval uncertainty. *Discrete Applied Mathematics*, 313:99–114. https://doi.org/10.1016/j.dam.2022.02.005.

Bold, M. and Goerigk, M. (2024). Recoverable robust single machine scheduling with polyhedral uncertainty. *Journal of Scheduling*. https://doi.org/10.1007/s10951-024-00828-7.

Bougeret, M., Pessoa, A. A., and Poss, M. (2019). Robust scheduling with budgeted uncertainty. *Discrete Applied Mathematics*, 261:93–107. https://doi.org/10.1016/j.dam.2018.07.001.

Bouleimen, K. and Lecocq, H. (2003). 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. https://doi.org/10.1016/S0377-2217(02)00761-0.

Bruni, M., Di Puglia Pugliese, L., Beraldi, P., and Guerriero, F. (2017). An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations. *Omega*, 71:66–84. https://doi.org/10.1016/j.omega.2016.09.009.

Bruni, M., Di Puglia Pugliese, L., Beraldi, P., and Guerriero, F. (2018). A computational study of exact approaches for the adjustable robust resource-constrained project scheduling problem. *Computers & Operations Research*, 99:178–190. https://doi.org/10.1016/j.cor.2018.06.016.

Carlier, J., Moukrim, A., and Sahli, A. (2018). Lower bounds for the event scheduling problem with consumption and production of resources. *Discrete Applied Mathematics*, 234:178–194. https://doi.org/10.1016/j.dam.2016.05.021.

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51. https://doi.org/10.1007/BF00940812.

Chaari, T., Chaabane, S., Aissani, N., and Trentesaux, D. (2014). Scheduling under uncertainty: Survey and research directions. In *2014 International Conference on Advanced Logistics and Transport (ICALT)*, pages 229–234, Hammamet, Tunisia. IEEE. https://doi.org/10.1109/ICAdLT.2014.6866316.

Chakrabortty, R. K., Sarker, R. A., and Essam, D. L. (2016). Multi-mode resource-constrained project scheduling under resource disruptions. *Computers & Chemical Engineering*, 88:13–29. https://doi.org/10.1016/j.compchemeng.2016.01.004.

Chanas, S. and Kuchta, D. (1998). Discrete fuzzy optimization. In Słowiński, R., editor, *Fuzzy Sets in Decision Analysis, Operations Research and Statistics*, pages 249–280. Springer US, Boston, MA. https://doi.org/10.1007/978-1-4615-5645-9_8.

Chang, G. J. and Edmonds, J. (1985). The poset scheduling problem. *Order*, 2(2):113–118. https://doi.org/10.1007/BF00334849.

Chaudhuri, S., Walker, R. A., and Mitchell, J. E. (1994). Analyzing and exploiting the structure of the constraints in the ilp approach to the scheduling problem. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2(4):456–471. https://doi.org/10.1109/92.335014.

Coelho, J. and Vanhoucke, M. (2011). Multi-mode resource-constrained project scheduling using rcpsp and sat solvers. *European Journal of Operational Research*, 213(1):73–82. https://doi.org/10.1016/j.ejor.2011.03.019.

Coello Coello, C. A. (2005). An introduction to evolutionary algorithms and their applications. In Ramos, F. F., Larios Rosillo, V., and Unger, H., editors, *Advanced Distributed Systems*, pages 425–442. Springer Berlin Heidelberg. https://doi.org/10.1007/11533962_39.

Colorni, A., Dorigo, M., Maniezzo, V., Varela, F., and Bourgine, P. (1992). Distributed optimization by ant colonies. In *Proceedings of the European Conference on Artificial Life*, pages 134–152.

Corne, D. W., Jerram, N. R., Knowles, J. D., and Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 283–290, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. ISBN 1558607749.

Custódio, A. L., Madeira, J. F. A., Vaz, A. I. F., and Vicente, L. N. (2011). Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21(3):1109–1140. https://doi.org/10.1137/10079731X.

Dai, H., Cheng, W., and Guo, P. (2018). An improved tabu search for multi-skill resource-constrained project scheduling problems under step-deterioration. *Arabian Journal for Science and Engineering*, 43:3279–3290. https://doi.org/10.1007/s13369-017-3047-4.

Damak, N., Jarboui, B., Siarry, P., and Loukil, T. (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research*, 36(9):2653–2659. https://doi.org/10.1016/j.cor.2008.11.010.

Danloup, N., Allaoui, H., and Goncalves, G. (2018). A comparison of two meta-heuristics for the pickup and delivery problem with transshipment. *Computers & Operations Research*, 100:155–171. https://doi.org/10.1016/j.cor.2018.07.013.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, Inc., USA. ISBN 047187339X.

Deb, K. (2011). Multi-objective optimisation using evolutionary algorithms: An introduction. In Wang, L., Ng, A. H. C., and Deb, K., editors, *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, pages 3–34. Springer, London. https://doi.org/10.1007/978-0-85729-652-8_1.

Deb, K., Agrawal, S., Pratap, A., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197. https://doi.org/10.1109/4235.996017.

Demeulemeester, E., De Reyck, B., and Herroelen, W. (2000). The discrete time/resource trade-off problem in project networks: A branch-and-bound approach. *IIE Transactions*, 32:1059–1069. https://doi.org/10.1023/A:1013785108131.

Demeulemeester, E. and Herroelen, W. (2002). *Project Scheduling: A Research Handbook*. Springer New York, NY, 1 edition. https://doi.org/10.1007/b101924.

Durillo, J. J. and Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771. https://doi.org/10.1016/j.advengsoft.2011.05.014.

Durillo, J. J., Nebro, A. J., Coello, C. A. C., Garcia-Nieto, J., Luna, F., and Alba, E. (2010). A study of multiobjective metaheuristics when solving parameter scalable problems. *IEEE Transactions on Evolutionary Computation*, 14(4):618–635. https://doi.org/10.1109/TEVC.2009.2034647.

Dächert, K., Klamroth, K., Lacour, R., and Vanderpooten, D. (2017). Efficient computation of the search region in multi-objective optimization. *European Journal of Operational Research*, 260(3):841–855. https://doi.org/10.1016/j.ejor.2016.05.029.

Ehrgott, M. (2005). *Multicriteria optimization*, volume 491. Springer Science & Business Media.

Eiben, A. E. and Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Springer. https://doi.org/10.1007/978-3-662-44874-8.

Emmerich, M., Beume, N., and Naujoks, B. (2005). An EMO algorithm using the hypervolume measure as selection criterion. In Coello Coello, C. A., Hernández Aguirre, A., and Zitzler, E.,

editors, *Evolutionary Multi-Criterion Optimization*, pages 62–76, Berlin, Heidelberg. Springer. ISBN 978-3-540-31880-4.

Emmerich, M. and Deutz, A. (2018). A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. *Natural Computing*, 17:585–609. https://doi.org/10.1007/s11047-018-9685-y.

Fahmy, A. M. (2016). Optimization algorithms in project scheduling. In *Optimization Algorithms – Methods and Applications*. InTech. https://doi.org/10.5772/63108.

Fernandes, A., Rodrigues, C., and da Costa, F. A. (2018). A path-relinking algorithm for the multi-mode resource-constrained project scheduling problem. *Computers & Operations Research*, 92:145–154. https://doi.org/10.1016/j.cor.2018.01.001.

Fernandes, G. A. and de Souza, S. R. (2021). A matheuristic approach to the multi-mode resource constrained project scheduling problem. *Computers & Industrial Engineering*, 162:107592. https://doi.org/10.1016/j.cie.2021.107592.

Filatovas, E., Kurasova, O., Redondo, J., and Fernández, J. (2020). A reference point-based evolutionary algorithm for approximating regions of interest in multiobjective problems. *TOP*, 28(2):402–423. https://doi.org/10.1007/s11750-019-00535-z.

Florez, L., Castro-Lacouture, D., and Medaglia, A. L. (2013). Sustainable workforce scheduling in construction program management. *Journal of the Operational Research Society*, 64(8):1169–1181. https://doi.org/10.1057/jors.2012.164.

Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence Through Simulated Evolution*. Wiley, Chichester, WS.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Forrest, S., editor, *Proccedings of the CGA-93: Fith International Conference on Genetic Algorithms*, pages 416–423, San Mateo, CA. Morgan Kaufmann.

Fonseca, C. M. and Fleming, P. J. (1996). On the performance assessment and comparison of stochastic multiobjective optimizers. In Voigt, H.-M., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 584–593, Berlin, Heidelberg. Springer. ISBN 978-3-540-70668-7.

Geiger, M. J. (2017). A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. *European Journal of Operational Research*, 256(3):729–741. https://doi.org/10.1016/j.ejor.2016.07.024.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166. https://doi.org/10.1111/j.1540-5915.1977.tb01074.x.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549. https://doi.org/10.1016/0305-0548(86)90048-1.

Glover, F. (1989). Tabu search—part i. *ORSA Journal on Computing*, 1(3):190–206. https://doi.org/10.1287/ijoc.1.3.190.

Glover, F. (1990). Tabu search—part ii. *ORSA Journal on Computing*, 2(1):4–32. https://doi.org/10.1287/ijoc.2.1.4.

Goerigk, M. and Hartisch, M. (2024). *An Introduction to Robust Combinatorial Optimization*, volume 361 of *International Series in Operations Research & Management Science*. Springer. https://doi.org/10.1007/978-3-031-61261-9.

Goldratt, E. (1997). *Critical Chain*. The North River Press.

Govindan, K., Jafarian, A., and Nourbakhsh, V. (2019). Designing a sustainable supply chain network integrated with vehicle routing: A comparison of hybrid swarm intelligence metaheuristics. *Computers & Operations Research*, 110:220–235. https://doi.org/10.1016/j.cor.2018.11.013.

Gröflin, H., Liebling, T., and Prodon, A. (1982). Optimal subtrees and extensions. In Bachem, A., Grötschel, M., and Korte, B., editors, *Bonn Workshop on Combinatorial Optimization*, volume 66 of *North-Holland Mathematics Studies*, pages 121–127. North-Holland. https://doi.org/10.1016/S0304-0208(08)72447-2.

Guiffrida, A. and Nagi, R. (1998). Fuzzy set theory applications in production management research: A literature survey. *Journal of Intelligent Manufacturing*, 9:39–56. https://doi.org/10.1023/A:1008847308326.

Habibi, F., Barzinpour, F., and Sadjadi, S. (2018). Resource-constrained project scheduling problem: Review of past and recent developments. *Journal of Project Management*, 3:55–88. https://doi.org/10.5267/j.jpm.2018.1.005.

Hall, N. G. and Posner, M. E. (2004). Sensitivity analysis for scheduling problems. *Journal of Scheduling*, 7(1):49–83. https://doi.org/10.1023/B:JOSH.0000013055.31639.f6.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics*, 45(7):733–750. https://doi.org/10.1002/(SICI)1520-6750(199810)45:7<733::AID-NAV5>3.0.CO;2-C.

Hartmann, S. (1999). *Project Scheduling under Limited Resources: Models, Methods, and Applications*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-58627-9.

Hartmann, S. (2001). Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102:111–135. https://doi.org/10.1023/A:1010902015091.

Hartmann, S. (2015). Time-varying resource requirements and capacities. In Schwindt, C. and Zimmermann, J., editors, *Handbook on Project Management and Scheduling Vol.1*, pages 163–176, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-05443-8_8.

Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14. https://doi.org/10.1016/j.ejor.2009.11.005.

Hartmann, S. and Briskorn, D. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 297(1):1–14. https://doi.org/10.1016/j.ejor.2021.05.004.

Hartmann, S. and Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2):394–407. https://doi.org/10.1016/S0377-2217(99)00485-3.

Hazır, O. and Ulusoy, G. (2020). A classification and review of approaches and methods for modeling uncertainty in projects. *International Journal of Production Economics*, 223:107522. https://doi.org/10.1016/j.ijpe.2019.107522.

Herroelen, W. and Leus, R. (2004). Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42:1599–1620. https://doi.org/10.1080/00207540310001638055.

Herroelen, W. and Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306. https://doi.org/10.1016/j.ejor.2004.04.002.

Hill, A., Lalla-Ruiz, E., Voß, S., and Goycoolea, M. (2019). A multi-mode resource-constrained project scheduling reformulation for the waterway ship scheduling problem. *Journal of Scheduling*, 22:173–182. https://doi.org/10.1007/s10951-018-0578-9.

Holland, J. H. (1962). Outline for a logical theory of adaptive systems. *Journal of the ACM*, 9(3):297–314. https://doi.org/10.1145/321127.321128.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Ikli, S., Mancel, C., Mongeau, M., Olive, X., and Rachelson, E. (2021). The aircraft runway scheduling problem: A survey. *Computers & Operations Research*, 132:105336. https://doi.org/10.1016/j.cor.2021.105336.

Ishibuchi, H., Masuda, H., Tanigaki, Y., and Nojima, Y. (2015). Modified distance calculation in generational distance and inverted generational distance. In Gaspar-Cunha, A., Henggeler Antunes, C., and Coello, C. C., editors, *Evolutionary Multi-Criterion Optimization*, pages 110–125, Cham. Springer International Publishing. https://doi.org/978-3-319-15892-1.

Jarboui, B., Damak, N., Siarry, P., and Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, 195(1):299–308. https://doi.org/10.1016/j.amc.2007.04.096.

Jie, L., Liu, W., Sun, Z., and Teng, S. (2017). Hybrid fuzzy clustering methods based on improved self-adaptive cellular genetic algorithm and optimal-selection-based fuzzy c-means. *Neurocomputing*, 249:140–156. https://doi.org/10.1016/j.neucom.2017.03.068.

Kasperski, A. and Zieliński, P. (2019). Risk-averse single machine scheduling: complexity and approximation. *Journal of Scheduling*, 22:567–580. https://doi.org/10.1007/s10951-019-00599-6.

Katoch, S., Chauhan, S. S., and Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80:8091–8126. https://doi.org/10.1007/s11042-020-10139-6.

Kelley, J. E. and Walker, M. R. (1959). Critical-path planning and scheduling. In *Papers Presented at the December 1–3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '59 (Eastern), pages 160–173. Association for Computing Machinery. https://doi.org/10.1145/1460299.1460318.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948. https://doi.org/10.1109/ICNN.1995.488968.

Knowles, J. and Corne, D. (1999). The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*, volume 1, pages 98–105, Washington, DC, USA. IEEE Press. ISBN 0-7803-5536-9, https://doi.org/10.1109/CEC.1999.781913.

Knowles, J. D. and Corne, D. W. (2000). Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172. https://doi.org/10.1162/106365600568167.

Kolisch, R. and Drexl, A. (1997). Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, 29:987–999. https://doi.org/10.1023/A:1018552303415.

Kolisch, R. and Sprecher, A. (1997). PSPLIB - A project scheduling problem library: OR software - ORSEP operations research software exchange program. *European Journal of Operational Research*, 96(1):205–216. https://doi.org/10.1016/S0377-2217(96)00170-1.

Koster, A. M., Segschneider, J., and Ventsch, N. (2024). γ-robust optimization of project scheduling problems. *Computers & Operations Research*, 161:106453. https://www.sciencedirect.com/science/article/pii/S0305054823003179.

Kouvelis, P. and Yu, G. (1997). *Robust Discrete Optimization and Its Applications*, volume 14 of *Nonconvex Optimization and Its Applications*. Springer Science & Business Media. https://doi.org/10.1007/978-1-4757-2620-6.

Koza, J. R. (1992). *Genetic Programming*. MIT Press, Cambridge, MA.

Kursawe, F. (1990). A variant of evolution strategies for vector optimization. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature (PPSN)*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197. Springer. https://doi.org/10.1007/BFb0029752.

Leyman, P. and Vanhoucke, M. (2016). Payment models and net present value optimization for resource-constrained project scheduling. *Computers & Industrial Engineering*, 91:139–153. https://doi.org/10.1016/j.cie.2015.11.008.

Lgelmund, G. and Radermacher, F. J. (1983a). Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*, 13(1):29–48. https://doi.org/10.1002/net.3230130103.

Lgelmund, G. and Radermacher, F. J. (1983b). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, 13(1):1–28. https://doi.org/10.1002/net.3230130102.

Li, H. and Zhang, H. (2013). Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. *Automation in Construction*, 35:431–438. https://doi.org/10.1016/j.autcon.2013.05.030.

Li, Z. and Ierapetritou, M. (2008). Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, 32(4):715–727. https://doi.org/10.1016/j.compchemeng.2007.03.001.

Li, Z., Lin, X., Zhang, Q., and Liu, H. (2020). Evolution strategies for continuous optimization: A survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 56:100694. https://doi.org/10.1016/j.swevo.2020.100694.

Liu, Y., Jin, S., Zhou, J., and Hu, Q. (2023). A branch-and-bound algorithm for the unit-capacity resource constrained project scheduling problem with transfer times. *Computers & Operations Research*, 151:106097. https://doi.org/10.1016/j.cor.2022.106097.

Lova, A., Tormos, P., Cervantes, M., and Barber, F. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, 117(2):302–316. https://doi.org/10.1016/j.ijpe.2008.11.002.

Lozano, J. A., naga, P. L., Inza, I., and Bengoetxea, E. (2006). *Towards a New Evolutionary Computation: Advances in Estimation of Distribution Algorithms*. Springer. https://doi.org/10.1007/3-540-32494-1.

Maghsoudlou, H., Afshar-Nadjafi, B., and Niaki, S. T. A. (2016). A multi-objective invasive weeds optimization algorithm for solving multi-skill multi-mode resource constrained project scheduling problem. *Computers & Chemical Engineering*, 88:157–169. https://doi.org/10.1016/j.compchemeng.2016.02.013.

Martin, O., Otto, S. W., and Felten, E. W. (1991). Large-step Markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–326.

Mavrotas, G. (2009). Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465. https://doi.org/10.1016/j.amc.2009.03.037.

Miettinen, K. (1998). *Nonlinear multiobjective optimization*. Springer, New York, 1 edition. https://doi.org/10.1007/978-1-4615-5563-6.

Möhring, R. H., Schulz, A. S., Stork, F., and Uetz, M. (2001). On project scheduling with irregular starting time costs. *Operations Research Letters*, 28(4):149–154. https://doi.org/10.1016/S0167-6377(01)00064-5.

Möhring, R. H., Schulz, A. S., Stork, F., and Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science*, 49(3):330–350. https://doi.org/10.1287/mnsc.49.3.330.12737.

Möhring, R. H. and Stork, F. (1997). State-of-the-art-survey—stochastic programming: Computation and applications. *INFORMS*, 9(2):111–229. https://doi.org/10.1287/ijoc.9.2.111.

Möhring, R. H. and Stork, F. (2000). Linear preselective policies for stochastic project scheduling. *Mathematical Methods of Operations Research*, 52(3):501–515. https://doi.org/10.1007/s001860000095.

Moukrim, A., Quilliot, A., and Toussaint, H. (2015). An effective branch-and-price algorithm for the preemptive resource constrained project scheduling problem based on minimal interval order enumeration. *European Journal of Operational Research*, 244(2):360–368. https://doi.org/10.1016/j.ejor.2014.12.037.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., and Alba, E. (2009). MOCell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746. https://doi.org/10.1002/int.20358.

Nebro, A. J., Durillo, J. J., and Vergne, M. (2015). Redesigning the jMetal multi-objective optimization framework. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1093–1100, New York, USA. Association for Computing Machinery. https://doi.org/10.1145/2739482.2768462.

Palacio, J. D. and Larrea, O. L. (2017). A lexicographic approach to the robust resource–constrained project scheduling problem. *International Transactions in Operational Research*, 24(1–2):143–157. https://doi.org/10.1111/itor.12301.

Pass-Lanneau, A., Bendotti, P., and Brunod-Indrigo, L. (2024). Exact and heuristic methods for anchor-robust and adjustable-robust RCPSP. *Annals of Operations Research*, 337:649–682. https://doi.org/10.1007/s10479-023-05537-6.

Patterson, J. H., Slowinski, R., Talbot, F. B., and Weglarz, J. (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. In Slowinski, R. and Weglarz, J., editors, *Advances in Project Scheduling*, pages 3–28, Amsterdam. Elsevier. https://doi.org/10.1016/B978-0-444-87358-3.50005-5.

Pinedo, M. L. (2016). *Scheduling: Theory, Algorithms, And Systems*. Springer Cham, 6 edition. https://doi.org/10.1007/978-3-031-05921-6.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57. https://doi.org/10.1007/s11721-007-0002-0.

Pourghaderi, A. R., Torabi, S. A., and Talebi, J. (2008). Scatter search for multi-mode resource-constrained project scheduling problems. In *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 163–167.

Pritsker, A. A. B., Watters, L. J., and Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108. https://doi.org/10.1287/mnsc.16.1.93.

R Development Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rahimi, A., Karimi, H., and Afshar-Nadjafi, B. (2013). Using meta-heuristics for project scheduling under mode identity constraints. *Applied Soft Computing*, 13(4):2124–2135. https://doi.org/10.1016/j.asoc.2012.11.002.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Technical Report Library Translation 1122, Royal Aircraft Establishment.

Riedler, M., Jatschka, T., Maschler, J., and Raidl, G. R. (2020). An iterative time-bucket refinement algorithm for a high-resolution resource-constrained project scheduling problem. *International Transactions in Operational Research*, 27(1):573–613. https://doi.org/10.1111/itor.12445.

Rodríguez-Ballesteros, S., Alcaraz, J., and Anton-Sanchez, L. (2024). Metaheuristics for the bi-objective resource-constrained project scheduling problem with time-dependent resource costs: An experimental comparison. *Computers & Operations Research*, 163. https://doi.org/10.1016/j.cor.2023.106489.

Rodríguez-Ballesteros, S., Alcaraz, J., and Anton-Sanchez, L. (2025a). Multi-mode resource-constrained project scheduling problem with time-dependent resource costs and capacities: A bi-objective approach. *Expert Systems With Applications*. Under review.

Rodríguez-Ballesteros, S., Goerigk, M., Alcaraz, J., and Anton-Sanchez, L. (2025b). A robust optimization approach for scheduling with uncertain start-time dependent costs. *Computers & Operations Research*. Under review.

Roundy, R. O., Maxwell, W. L., Herer, Y. T., Tayur, S. R., and Getzler, A. W. (1991). A price-directed approach to real-time scheduling of production operations. *IIE Transactions*, 23(2):149–160. https://doi.org/10.1080/07408179108963850.

Sabuncuoglu, I. and Karabuk, S. (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems*, 18(4):268–283. Special issue on scheduling: From Research Into Practice. https://doi.org/10.1016/S0278-6125(00)86630-3.

Sahli, A., Carlier, J., and Moukrim, A. (2016). Comparison of mixed integer linear programming models for the event scheduling problem with consumption and production of resources. *IFAC-PapersOnLine*, 49(12):1044–1049. 8th IFAC Conference on Manufacturing Modelling, Management and Control (MIM 2016). https://doi.org/10.1016/j.ifacol.2016.07.580.

Said, L. B., Bechikh, S., and Ghedira, K. (2010). The r-dominance: A new dominance relation for interactive evolutionary multicriteria decision making. *IEEE Transactions on Evolutionary Computation*, 14(5):801–818. https://doi.org/10.1109/TEVC.2010.2041060.

Salto, C. and Alba, E. (2019). Cellular genetic algorithms: Understanding the behavior of using neighborhoods. *Applied Artificial Intelligence*, 33(10):863–880. https://doi.org/10.1080/08839514.2019.1646005.

Samikoglu, Ö., Honkomp, S. J., Pekny, J. F., and Reklaitis, G. V. (1998). Sensitivity analysis for project planning and scheduling under uncertain completions. *Computers & Chemical Engineering*, 22:S871–S874. European Symposium on Computer Aided Process Engineering-8, https://doi.org/10.1016/S0098-1354(98)00169-0.

Sankaran, J., Bricker, D., and Juang, S.-H. (1999). Strong fractional cutting-plane algorithm for resource-constrained project scheduling. *International Journal of Industrial Engineering: Theory Applications and Practice*, 6(2):99–111.

Schlünz, E. B., Bokov, P. M., and van Vuuren, J. H. (2016). A comparative study on multiobjective metaheuristics for solving constrained in-core fuel management optimisation problems. *Computers & Operations Research*, 75:174–190. https://doi.org/10.1016/j.cor.2016.06.001.

Schnabel, A., Kellenbrink, C., and Helber, S. (2018). Profit-oriented scheduling of resource-constrained projects with flexible capacity constraints. *Business Research*, 11:329–356. https://doi.org/10.1007/s40685-018-0063-5.

Schnell, A. and Hartl, R. F. (2016). On the efficient modeling and solution of the multi-mode resource-constrained project scheduling problem with generalized precedence relations. *OR Spectrum*, 38(2):283–303. https://doi.org/10.1007/s00291-015-0419-6.

Schutt, A., Feydy, T., Stuckey, P. J., and Wallace, M. (2013). Solving rcpsp/max by lazy clause generation. *Journal of Scheduling*, 16:273–289. https://doi.org/10.1007/s10951-012-0285-x.

Seada, H. and Deb, K. (2015). U-NSGA-III: A unified evolutionary optimization procedure for single, multiple, and many objectives: Proof-of-principle results. In Gaspar-Cunha, A., Henggeler Antunes, C., and Coello, C. C., editors, *Evolutionary Multi-Criterion Optimization*, pages 34–49, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-319-15892-1_3.

Seeley, T. D. (1995). *The Wisdom of the Hive*. Harvard University Press, Cambridge, MA. ISBN 9780674953765.

Shabtay, D. (2023). A new perspective on single-machine scheduling problems with late work related criteria. *Annals of Operations Research*, 322(2):947–966. https://doi.org/10.1007/s10479-022-04806-0.

Shahsavar, A., Najafi, A. A., and Niaki, S. T. A. (2015). Three self-adaptive multi-objective evolutionary algorithms for a triple-objective project scheduling problem. *Computers & Industrial Engineering*, 87:4–15. https://doi.org/10.1016/j.cie.2015.04.027.

Shen, H. and Li, X. (2013). Cooperative discrete particle swarms for multi-mode resource-constrained projects. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 31–36.

Siew Mooi, S. L., Md Sultan, A. B., Sulaiman, M., Mustapha, A., and Leong, K. Y. (2017). Crossover and mutation operators of genetic algorithms. *International Journal of Machine Learning and Computing*, 7(1):9–12. https://doi.org/10.18178/ijmlc.2017.7.1.611.

Singh, G. and Gupta, N. (2022). *A Study of Crossover Operators in Genetic Algorithms*, pages 17–32. Springer Singapore. https://doi.org/10.1007/978-981-16-3128-3_2.

Slowiński, R. and Hapke, M. (2000). *Scheduling under Fuzziness*, pages 167–198. Physica-Verlag, Heidelberg, Germany.

Sourd, F. (2009). New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal on Computing*, 21:167–175. https://doi.org/10.1287/ijoc.1080.0287.

Sprecher, A. and Drexl, A. (1998). Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107:431–450. https://doi.org/10.1016/S0377-2217(97)00348-2.

Sprecher, A., Hartmann, S., and Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *OR Spektrum*, 19:195–203. https://doi.org/10.1007/BF01545587.

Storn, R. and Price, K. (1997). Differential evolution –A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359. https://doi.org/10.1023/A:1008202821328.

Stroustrup, B. (2013). *The C++ Programming Language*. Addison-Wesley, 4th edition.

Suresh, V. and Chaudhuri, D. (1993). Dynamic scheduling – a survey of research. *International Journal of Production Economics*, 32(1):53–63. https://ideas.repec.org/a/eee/proeco/v32y1993i1p53-63.html.

Szelke, E. and Kerr, R. M. (1994). Knowledge-based reactive scheduling. *Production Planning and Control*, 5(2):124–145. https://doi.org/10.1080/09537289408919480.

Tabrizi, B. H. (2018). Integrated planning of project scheduling and material procurement considering the environmental impacts. *Computers & Industrial Engineering*, 120:103–115. https://doi.org/10.1016/j.cie.2018.04.031.

Talbi, E.-G. (2009). *Metaheuristics: From Design to Implementation*. John Wiley & Sons, Hoboken, NJ, USA. ISBN 978-0-470-27858-1.

Talbot, F. B. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The non-preemptive case. *Management Science*, 28(10):1091–1213. https://doi.org/10.1287/mnsc.28.10.1197.

Tirkolaee, E. B., Goli, A., Hematian, M., and Sadeghian, R. (2019). Multi-objective multi-mode resource constrained project scheduling problem using pareto-based algorithms. *Computing*, 101:547–570. https://doi.org/10.1007/s00607-018-00693-1.

Toffolo, T. A. M., Santos, H. G., Carvalho, M. A. M., and Soares, J. A. (2016). An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *Journal of Scheduling*, 19:295–307. https://doi.org/10.1007/s10951-015-0422-4.

Van Cauwelaert, S., Dejemeppe, C., and Schaus, P. (2020). An efficient filtering algorithm for the unary resource constraint with transition times and optional activities. *Journal of Scheduling*, 23:431–449. https://doi.org/10.1007/s10951-019-00632-8.

Van Peteghem, V. and Vanhoucke, M. (2011). Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. *Journal of Heuristics*, 17:705–728. https://doi.org/10.1007/s10732-010-9152-0.

Van Peteghem, V. and Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, 235(1):62–72. https://doi.org/10.1016/j.ejor.2013.10.012.

Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Air Force Institute of Technology.

Vanhoucke, M. and Coelho, J. (2019). Resource-constrained project scheduling with activity splitting and setup times. *Computers & Operations Research*, 109:230–249. https://doi.org/10.1016/j.cor.2019.05.004.

Vieira, G. E., Herrmann, J. W., and Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, 6:39–62. https://doi.org/10.1023/A:1022235519958.

Wang, Q., Liu, C., and Zheng, L. (2020). A column-generation-based algorithm for a resource-constrained project scheduling problem with a fractional shared resource. *Engineering Optimization*. https://doi.org/10.1080/0305215X.2019.1610946.

Whitley, L. D. (1993). Cellular genetic algorithms. In *International Conference on Genetic Algorithms*. https://doi.org/10.1007/978-981-10-7497-4_5.

Wu, C. W., Brown, K. N., and Beck, J. C. (2009). Scheduling with uncertain durations: Modeling β-robust scheduling with constraints. *Computers & Operations Research*, 36(8):2348–2356. https://doi.org/10.1016/j.cor.2008.08.008.

Yepes-Borrero, J. C., Perea, F., Ruiz, R., and Villa, F. (2021). Bi-objective parallel machine scheduling with additional resources during setups. *European Journal of Operational Research*, 292(2):443–455. https://doi.org/10.1016/j.ejor.2020.10.052.

Zaman, F., Elsayed, S., Sarker, R., Essam, D., and Coello, C. A. C. (2021). Pro-reactive approach for project scheduling under unpredictable disruptions. *IEEE Transactions on Cybernetics*, 52(11):11299–11312. https://doi.org/10.1109/TCYB.2021.3097312.

Zamani, R. (2013). An evolutionary search procedure for optimizing time–cost performance of projects under multiple renewable resource constraints. *Computers & Industrial Engineering*, 66(2):451–460. https://doi.org/10.1016/j.cie.2013.07.010.

Zamanii, R. (2019). An effective mirror-based genetic algorithm for scheduling multi-mode resource constrained projects. *Computers & Industrial Engineering*, 127:914–924. https://doi.org/10.1016/j.cie.2018.11.031.

Zeng, B. and Zhao, L. (2013). Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters*, 41(5):457–461. https://doi.org/10.1016/j.orl.2013.05.003.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731. https://doi.org/10.1109/TEVC.2007.892759.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland. ISBN 3826568311.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195. https://doi.org/10.1162/106365600568202.

Zitzler, E., Knowles, J., and Thiele, L. (2008). *Quality Assessment of Pareto Set Approximations*, pages 373–404. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-88908-3_14.

Zitzler, E. and Künzli, S. (2004). Indicator-based selection in multiobjective search. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J. E., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII*, pages 832–842, Berlin, Heidelberg. Springer. ISBN 978-3-540-30217-9.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm. Technical report, ETH Zurich, Computer Engineering and Networks Laboratory. https://doi.org/10.3929/ethz-a-004284029.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271. https://doi.org/10.1109/4235.797969.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132. https://doi.org/10.1109/TEVC.2003.810758.