CAIO DE OLIVEIRA LOPES

ALGORITHMS FOR SOLVING THE JOB ROTATION PROBLEM WITH HETEROGENEOUS WORKERS

LAVRAS – MG

2025

CAIO DE OLIVEIRA LOPES

ALGORITHMS FOR SOLVING THE JOB ROTATION PROBLEM WITH HETEROGENEOUS WORKERS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Inteligência Artificial e Otimização, para a obtenção do título de Mestre.

Prof. Dr. Mayron César de Oliveira Moreira
Orientador

LAVRAS – MG
2025

Ficha Catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Lopes, Caio de Oliveira.
Algorithms for solving the job rotation problem with heterogeneous workers / Caio de Oliveira Lopes. - 2025.
92 p. : il.

Orientador: Mayron César de Oliveira Moreira

Dissertação (Mestrado Acadêmico) - Universidade Federal de Lavras, 2025.
Bibliografia.

1. Rotação de tarefas. 2. Balanceamento de linha de produção. 3. Trabalhadores heterogêneos. 4. Heurísticas. 5. Programação inteira mista. I. Moreira, Mayron César de Oliveira. II. Universidade Federal de Lavras. III. Título.

CAIO DE OLIVEIRA LOPES

ALGORITHMS FOR SOLVING THE JOB ROTATION PROBLEM WITH HETEROGENEOUS WORKERS

ALGORITHMS FOR SOLVING THE JOB ROTATION PROBLEM WITH HETEROGENEOUS WORKERS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Inteligência Artificial e Otimização, para a obtenção do título de Mestre.

APROVADA em 16 de julho de 2025.

Prof. Dr. Mayron César de Oliveira Moreira	UFLA
Prof. Pedro Belin Castellucci	UFSC
Prof. Vinicius Vitor dos Santos Dias	UFLA
Prof. Paulo Afonso Parreira Junior	UFLA

Prof. Dr. Mayron César de Oliveira Moreira
Orientador

LAVRAS – MG
2025

Dedico este trabalho à minha mãe, que com amor e dedicação me guiou em cada etapa da vida, inspirando-me a buscar sempre o melhor.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão à minha mãe, por todo o apoio, incentivo e presença constante ao longo da minha vida. Sua dedicação foi essencial para minha formação pessoal e acadêmica, e sem ela, esta conquista não seria possível.

Agradeço especialmente ao professor Mayron César de Oliveira Moreira, que mais uma vez assumiu o papel de orientador, agora nesta nova etapa da minha trajetória na pós-graduação. Sua orientação, paciência e incentivo foram fundamentais para a realização deste trabalho.

Deixo também meu agradecimento à professora Andreza Cristina Beezão Moreira, cuja dedicação e apoio durante a graduação tiveram um impacto marcante no meu desenvolvimento acadêmico e pessoal.

Sou grato à Universidade Federal de Lavras, em especial ao Programa de Pós-Graduação em Ciência da Computação, pela oportunidade de desenvolver este estudo. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), que contribuíram para a continuidade e a qualidade desta pesquisa.

A todos que, direta ou indiretamente, fizeram parte deste caminho, deixo meu sincero agradecimento. Espero ter retribuído, ao menos em parte, todo o apoio recebido, e que minhas atitudes e conquistas também tenham deixado boas lembranças.

"An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem."

(John Tukey)

RESUMO

Este estudo aborda o problema de balanceamento de linhas de produção e designação de trabalhadores em linhas de montagem com rotação de tarefas (JRALWABP). O problema em questão considera trabalhadores com heterogeneidade na execução de tarefas. O objetivo consiste em maximizar o número de tarefas distintas executadas pelos trabalhadores e minimizar o tempo de ciclo médio em relação a todos os períodos, sujeito às precedências de tarefas. Inicialmente, avaliamos duas formulações exatas de programação inteira mista (Modelos M1 e M2), considerando o tempo médio de ciclo como restrição. Embora essas formulações fossem úteis como referência, mostraram-se inviáveis para instâncias maiores, aumentando a necessidade de heurísticas robustas. Nesse contexto, desenvolvemos a estratégia denominada HAJR2, que evolui o algoritmo híbrido existente (HAJR1) por meio de um processo iterativo intercalando heurísticas como, GRASP, Busca Tabu e um algoritmo genético, além da introdução da heurística Pattern Injection Local Search (PILS). Os parâmetros da Busca Tabu e do algoritmo genético foram calibrados com Irace e a execução de ambos foi feita em mesma base, assegurando comparações justas. Os experimentos em quatro famílias de instâncias mostraram que HAJR2 supera HAJR1, aumentando a variedade média de tarefas atribuídas sem afetar o tempo de ciclo, e se mostra mais resistente em instâncias com alta taxa de incompatibilidade tarefa/trabalhador. Este trabalho refina o estado da arte na programação de rotação de trabalhadores heterogêneos, oferecendo uma base prática e extensível para aplicações reais em linhas de montagem.

Palavras-chave: rotação de tarefas; balanceamento de linha de produção; trabalhadores heterogêneos; heurísticas; programação inteira mista.

ABSTRACT

This study addresses the problem of assembly line balancing and worker assignment in job rotation scenarios (JRALWABP). The problem considers workers with heterogeneous task execution capabilities. The objective is to maximize the number of distinct tasks performed by each worker and to minimize the average cycle time across all periods, subject to task precedence constraints. Initially, we evaluated two exact mixed-integer programming formulations (Models M1 and M2), considering the average cycle time as a constraint. Although these formulations were useful as a reference, they proved infeasible for larger instances, highlighting the need for robust heuristics. In this context, we developed a strategy called HAJR2, which enhances the existing hybrid algorithm (HAJR1) through an iterative process that interleaves heuristics such as GRASP, Tabu Search, and a Genetic Algorithm, along with the introduction of the Pattern Injection Local Search (PILS) heuristic. The parameters of both the Tabu Search and Genetic Algorithm were tuned using Irace, and their execution was based on the same experimental setup, ensuring fair comparisons. Experiments on four instance families demonstrated that HAJR2 outperforms HAJR1 by increasing the average variety of tasks assigned without affecting the cycle time, and it proves more resilient in instances with a high task/worker incompatibility rate. This work refines the state of the art in heterogeneous worker job rotation scheduling and offers a practical, extensible foundation for real-world applications in assembly lines.

Keywords: job rotation; assembly line balancing; heterogeneous workers; heuristics; mixed-integer programming.

INDICADORES DE IMPACTO

O estudo desenvolvido tem como foco a proposição de algoritmos de otimização voltados ao apoio à tomada de decisão no planejamento da rotação de tarefas em sistemas produtivos. Em particular, são consideradas linhas de produção compostas por trabalhadores heterogêneos. Levando-se em conta que, em determinadas linhas, a variabilidade na execução das tarefas está associada à presença de pessoas com deficiência, esta dissertação propõe uma abordagem computacional capaz de tornar esses trabalhadores mais generalistas, assegurando uma produtividade mínima exigida pela indústria. A proposta representa uma estratégia para otimizar a fabricação em larga escala, ao mesmo tempo em que reduz a visibilidade de limitações funcionais no processo de aferição da produtividade. Em termos práticos, os gestores e trabalhadores de equipes com heterogeneidade funcional serão beneficiados pelas propostas apresentadas. A metodologia aqui apresentada pode ser de interesse para pesquisadores das áreas de Pesquisa Operacional e Ergonomia, que atuam na interface entre eficiência produtiva e fatores humanos. Por fim, este trabalho enquadra-se em importantes diretrizes institucionais: (i) Conforme as áreas temáticas da Política Nacional de Extensão, o estudo insere-se na categoria de "Tecnologia e Produção"; (ii) Alinhado aos 17 Objetivos de Desenvolvimento Sustentável (ODS) da Organização das Nações Unidas (ONU), esta dissertação se relaciona diretamente com o objetivo "Indústria, inovação e infraestrutura".

IMPACT INDICATORS

The study presented focuses on the design of optimization algorithms to support decision-making in the planning of task rotation within production systems. In particular, we consider production lines composed of heterogeneous workers. Bearing in mind that, in certain lines, variability in task execution is linked to the inclusion of persons with disabilities, this dissertation proposes a computational approach capable of making these workers more versatile, while ensuring the minimum productivity levels required by industry. The proposal represents a strategy to optimize large-scale manufacturing, at the same time reducing the visibility of functional limitations during productivity assessment. In practical terms, managers and team members operating under conditions of functional heterogeneity will benefit from the methods presented. The methodology introduced here may be of interest to researchers in the fields of Operations Research and Ergonomics, who work at the interface between productive efficiency and human factors. Finally, this work aligns with key institutional guidelines: (i) in accordance with the thematic areas of the National Extension Policy, the study falls under the category “Technology and Production”; (ii) aligned with the United Nations’ 17 Sustainable Development Goals (SDGs), this dissertation directly supports the goal “Industry, Innovation, and Infrastructure”.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

The Assembly Line Worker Assignment and Balancing Problem (ALWABP) has attracted significant attention from researchers, practitioners, and industry leaders due to its crucial role in industrial operations and workforce management. This complex challenge involves precisely optimizing the allocation of workers and tasks in assembly line production. Addressing ALWABP is essential for addressing practical concerns related to efficiency, productivity, and the strategic utilization of resources in manufacturing environments. As industries evolve and seek innovative solutions, ALWABP remains a key area of interest, requiring sophisticated approaches to balance workforce deployment and task assignment for sustained operational excellence.

Building on this foundation, the present research examines Job Rotation Assembly Line Worker Assignment and Balancing Problem (JRALWABP) as an extension of ALWABP, which directly impacts workflow within assembly lines, incorporating temporal dynamics by considering a planning horizon divided into T periods. In this context, a distinct ALWABP solution is generated for each period t, allowing workers to rotate across different stations or tasks over time. This temporal variation introduces new challenges and opportunities, as job rotation must be planned in a way that not only maintains feasible and efficient line balancing at each period, but also supports broader goals across the planning horizon. In this research, we aim to maximize the number of distinct tasks performed by each worker, promoting skill diversification and workforce flexibility.

In Section Chapter 2, examples of job rotation planning solutions will be presented to clarify key aspects of the problem. According to Comper *et al.* (2017), job rotation is closely linked to various benefits for workers, such as skill acquisition, improved job performance, increased autonomy, enhanced organizational flexibility, reduced physical and mental strain, greater job satisfaction, decreased monotony and boredom, prevention of musculoskeletal disorders, and lower absenteeism. Additionally, manufacturing systems must adapt to evolving market and product demands, as they are vital components of supply chains.

Keeping on a social perspective, several studies have investigated the health impacts of job rotation on workers, providing diverse insights into this multifaceted problem, as shown by Padula *et al.* (2017) and Comper *et al.* (2021). Job rotation also serves as an effective means to

support and integrate disabled (Moreira; Costa, 2013, e.g.) and older workers (Calzavara *et al.*, 2020, e.g.).

From a problem-solving perspective, job rotation within the Assembly Line Worker Assignment and Balancing Problem (ALWABP)—referred to as JRALWABP—was initially addressed by Costa & Miralles (2009), who employed a heuristic decomposition method supported by a *mixed-integer programming* (MIP) formulation. This work was later expanded by Moreira & Costa (2013), who proposed a hybrid method. The current study builds upon the contributions of Moreira & Costa (2013).

1.1 Motivation

The motivation for this research extends beyond academic or industrial considerations. It is grounded in a broader goal: to improve the quality of life for assembly line workers, enhance their workplace motivation, and create opportunities for professional development and training. Given that assembly line work forms the foundation of many industries, this research aims to offer new algorithmic approaches to support decision-makers in this environment.

1.2 Objectives

A central question guides this research: "Is it possible to introduce a novel algorithm for optimizing job rotation planning within the ALWABP context that surpasses the performance of the current benchmark?" The primary objective is to investigate the job rotation problem in ALWABP in-depth, enhancing existing algorithms and methodologies. This study extends the work of Moreira & Costa (2013), which introduced a hybrid method composed of three key steps: (*i*) generating a pool of feasible ALWABP solutions; (*ii*) selecting the most suitable solutions from the pool to construct a job rotation scheme; and (*iii*) applying a local search to refine the chosen solutions. This study proposes modifications to this structure. The specific goals guiding this research are outlined below:

a) **improvement of the solution selector mechanism:** Efficient task assignment is vital to assembly line performance. This research seeks to refine the solution selection mechanism, particularly in job rotation contexts, by leveraging bi-objective optimization instead of relying on average cycle time as a constraint;

b) **investigation of alternative solution generators:** To enhance the diversity and quality of solutions in the ALWABP solution pool, this study explores alternative algorithms for generating solutions. These will be assessed for their suitability to meet job rotation requirements, including modifications to the heuristic proposed by Moreira & Costa (2013);

c) **parameter tuning:** Effective configuration of algorithmic parameters is crucial for efficiently solving ALWABP. Neither Costa & Miralles (2009) nor Moreira & Costa (2013) explored this aspect. This research investigates parameter tuning using Irace (López-Ibáñez *et al.*, 2016) to determine optimal settings for each heuristic component;

d) **solution pattern identification:** This research aims to identify patterns that correlate elite ALWABP solutions with optimal problem-solving strategies. By utilizing Pattern Identification for Learning Solutions (PILS) (Arnold *et al.*, 2021), these patterns can inform the development of more effective and scenario-specific solutions;

e) **iterative method:** In Moreira & Costa (2013), the heuristic is executed once with relatively long execution times. This study proposes an iterative approach that utilizes shorter execution cycles and interleaves heuristics throughout the runtime. This method aims to enhance the exploration of the ALWABP solution space.

1.3 Contributions

In addition to the outlined objectives, this research addresses specific gaps in the literature, particularly the lack of advanced solution selection mechanisms and the limited exploration of alternative solution generators. The diversity and quality of solutions for each period directly impact the overall results. Moreover, parameter tuning remains underexplored in this field. This study aims to demonstrate that fine-tuning can lead to improved solutions by enhancing the parameters and methods employed. Additionally, the proposed iterative approach facilitates a more comprehensive exploration of the solution space, increasing the chances of discovering higher-quality solutions. These solutions not only aim to reduce cycle time but also prioritize job rotation by generating ALWABP solutions that contribute to task variety among workers.

1.4 Document Structure

This research is structured as follows: Chapter 2 presents the theoretical framework and core concepts that underpin the study. Chapter 3 reviews relevant literature and outlines the contributions that inform this research. Chapter 4 describes the methodology adopted to achieve the research objectives. Chapter 5 systematically presents the experimental results. Finally, Chapter 6 concludes the research by summarizing key findings and offering final insights.

2 THEORETICAL BASIS

This chapter presents the theoretical foundations of this study. The primary objective is to illuminate the conceptual framework. Through this exploration, the chapter aims to provide a comprehensive understanding of the theoretical landscape shaping the research and offer insights into the intellectual framework guiding the design and conduct of the study.

2.1 Problem Complexity

The Simple Assembly Line Balancing Problem (SALBP) is a significant challenge in combinatorial optimization, especially in the context of assembly line operations. In Miranda & Pereira (2019), the authors emphasize the widespread interest in assembly line balancing due to its industrial relevance and apparent simplicity. Building on this foundation, Miranda, Pereira & Vilà (2023) revisits SALBP, providing an alternative proof of the problem's NP-completeness using precedence constraints and introducing new benchmark instances for empirical analysis. Their results emphasize the importance of packing features in addressing the difficulty of SALBP and, at the same time, show the influential role played by the number of precedence constraints.

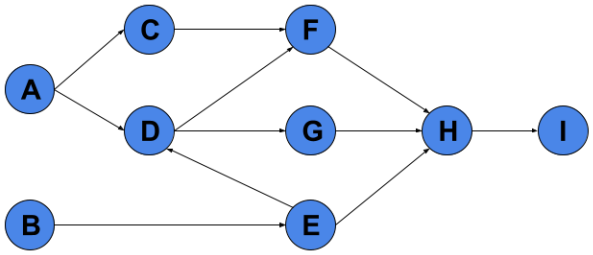
Extending the concept of NP-completeness of SALBP introduced by Miranda, Pereira & Vilà (2023), ALWABP encompasses not only the assignment of assembly operations to workstations but also the allocation of workers to these stations, thereby optimizing both production efficiency and human resource utilization. Given the complex nature of worker assignment and balancing, the introduction of precedence constraints in ALWABP significantly adds to the system's complexity. If we consider an instance of ALWABP where all workers perform tasks with the same execution time and all workers can execute all tasks, we fall into an instance of SALBP. Therefore, the decision version of ALWABP is also NP-complete.

As with SALBP, the precedence relations introduce a layer of dependencies that increases the difficulty of the problem. By drawing parallels to SALBP and its proven NP-complete status, it becomes clear that ALWABP inherits this computational complexity and presents a formidable challenge for efficient and optimal solution methods. This extension highlights the intricate interplay between task allocation, worker assignment, and precedence constraints in real-world assembly line scenarios, underscoring the need for advanced optimization techniques to address the NP-completeness of ALWABP.

2.2 Job Rotation Feasible Schedules

The examples presented are taken from Moreira (2015). Figure 2.1 shows a precedence graph with 9 tasks. In this case, for example, task D must be carried out before tasks F and G but after tasks A and E.

Figure 2.1 – Precedence graph with 9 tasks.



Source: Moreira (2015)

Consider Table 2.1 as the execution time matrix for an assembly line with 3 workers and 9 tasks. This scenario shows notable differences in task execution times between workers, as well as cases where they cannot perform certain tasks: workers w_1, w_2, and w_3 for tasks (F), (D and E), and (B), respectively. The examples in this section are based on Table 2.1.

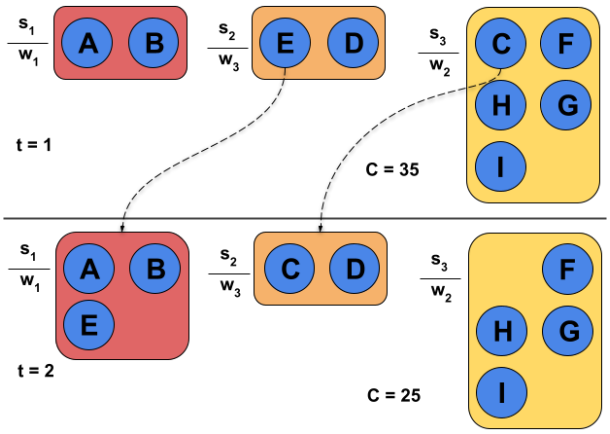
Table 2.1 – Task execution times matrix in ALWABP.

Workers	**Tasks**								
	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**
w_1	5	4	11	3	2	∞	10	30	15
w_2	7	4	10	∞	∞	12	3	1	9
w_3	40	∞	2	9	7	3	25	20	35

Source: Moreira (2015)

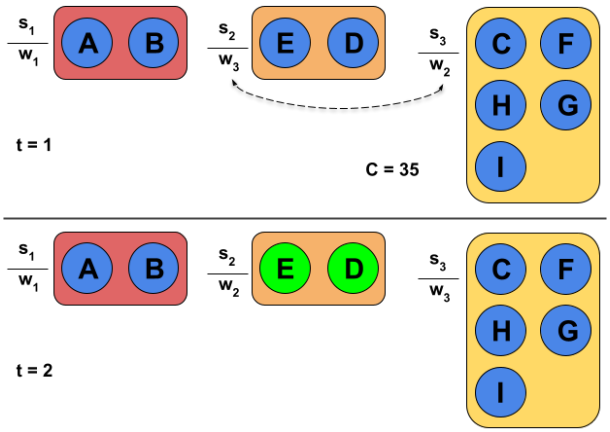
Based on the precedence graph in Figure 2.1, Figures 2.2, 2.3, and 2.4 show three examples with 3 workers, 9 tasks, and 2 periods. The infeasibilities in Figures 2.3 and 2.4, indicated by the green tasks, result from two distinct changes: exchanging workers between stations s_2 and s_3, and inserting task B into station s_2. In both cases, workers w_2 and w_3 cannot perform some tasks assigned to their stations in the last period. As noted earlier, the objective function (OF) aims to maximize the total number of distinct tasks executed by each worker across all periods.

Figure 2.2 – Feasible JRALWABP solution with 3 stations, 9 tasks, and 2 periods (OF = 11).



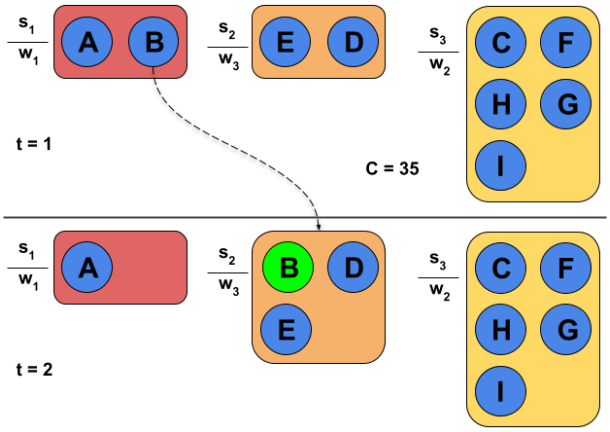
Source: Moreira (2015)

Figure 2.3 – JRALWABP solution with infeasibility at s_2 (tasks E and D, worker w_2) (OF = 16).



Source: Moreira (2015)

Figure 2.4 – JRALWABP solution with infeasibility at s_2 (task B, worker w_3) (OF = 10).



Source: Moreira (2015)

The examples presented in this section serve to illustrate the complexity and practical challenges associated with solving JRALWABP. By highlighting different task execution times and worker-task incompatibilities, these scenarios emphasize the need for careful planning and optimization to ensure feasible solutions. The infeasibilities observed in the later examples underscore the impact of worker-task restrictions and the role of precedence constraints, providing a concrete understanding of the difficulties involved in balancing assembly lines while respecting job rotation and worker allocation.

2.3 Algorithms for job rotation in ALWABP

This section provides a brief introduction to the algorithms presented in Costa & Miralles (2009) and Moreira & Costa (2013), which are of central importance to this research base.

2.3.1 Algorithm by Costa & Miralles (2009)

The authors Costa & Miralles (2009) propose an algorithm that uses a decomposition method based on a two-step approach to solve job rotations in ALWABP. The goal is to maximize the number of different tasks performed by each worker. In the first phase, the ALWABP problem is solved, and the solution obtained is assigned to the first rotation period. In the second phase, the problems for each subsequent period are addressed by modifying the objective function and introducing a constraint on the maximum cycle time. The algorithm performs sequential optimizations for each period, maximizing the variety of tasks assigned to each worker.

The algorithm employs a greedy strategy, and the objective function ensures that the algorithm only considers the variables associated with worker-task pairs that are not part of a previous solution. The cycle time limit is adjusted in each iteration to allow flexibility for later periods. The main feature of the algorithm is its ability to correct or partially correct potentially incorrect decisions made in previous phases, thanks to the independence between assignments in each period. The modified objective function leads the algorithm to reasonable global solutions.

2.3.2 Algorithm by Moreira & Costa (2013)

The paper presents several heuristic methods for solving the ALWABP, with a focus on generating high-quality single-period schedules, which form the foundation for creating effective job rotation schedules. The first approach discussed is the station-based constructive heuristics, initially developed by Moreira *et al.* (2012), based on the heuristics for the SALBP proposed by Scholl & Voss (1997). These heuristics sequentially add tasks to workstations while respecting precedence constraints using a heuristic priority rule. A total of 16 heuristic rules were adapted to ALWABP for task prioritization, along with three criteria for worker assignment.

Another method explored in the paper is a hybrid genetic algorithm. This algorithm utilizes constructive heuristics as decoders within a biased random-key genetic algorithm, where the values returned by these heuristics determine the fitness of individuals. The chromosome represents priority values for task-worker pairs. Additionally, the genetic algorithm incorporates mechanisms such as elitism, local search heuristics, and an immigration phase to maintain population diversity.

The Tabu Search algorithm, presented in Section 4.2.2, is also extended for ALWABP, incorporating both intensification and diversification procedures. Moves within this algorithm include task insertions, task swaps, and worker swaps. Infeasible solutions are allowed but are dynamically penalized; the algorithm employs aspiration criteria and penalty adjustments based on solution feasibility. It also includes three local search procedures and alternates between intensification and diversification movements to explore the solution space more effectively.

The paper further introduces a GRASP (Greedy Randomized Adaptive Search Procedure) method, presented in Section 4.2.1, which is a multi-start metaheuristic. GRASP involves an initial solution construction phase, followed by a solution improvement phase via local search. It relies on constructive heuristics for creating initial solutions and uses a restricted candidate list to select tasks during construction. The solution improvement phase incorporates a local search procedure similar to LS1, and this method iteratively refines solutions to produce different outcomes at each iteration.

Finally, a hybrid algorithm for job rotation scheduling is presented. This algorithm combines the aforementioned heuristic and metaheuristic approaches with a mathematical programming model. It creates a pool of solutions using the four heuristic methods. Then, it applies an integer linear programming model to select the optimal job rotation schedule, ensuring that

cycle time constraints are respected and the objective function is maximized. Two local search approaches based on large MIP neighborhoods are also employed to improve the objective function further. This integration of methods ensures that both the complexity of the problem and the practical needs of real-world assembly line operations are addressed robustly and flexibly, as outlined in the pseudocode in Figure 2.5.

Figure 2.5 – Hybrid Genetic Algorithm (HGA) for Job Rotation.

Algorithm 1 Hybrid Genetic Algorithm (HGA) for Job Rotation

```
Initialize the population with random solutions
Evaluate each solution using constructive heuristics
Store the best solutions in the elite set
while stopping_criterion_not_met() do
    Select individuals for reproduction
    Apply crossover to generate new individuals
    Apply mutation to explore new solution regions
    Apply local search to the newly generated individuals
    Evaluate the solutions and update the population
    Maintain the best solutions in the elite set
    if immigration_criterion_met() then
        Insert new random individuals into the population
Return the best solution found
```

Source: Author

2.4 Formal Definition

Mixed-integer programming (MIP) models are mathematical formulations used to solve optimization problems that involve both continuous and discrete variables. These models are particularly compelling when addressing complex decision-making scenarios that involve constraints and objectives to optimize and have been widely applied in various fields, including production scheduling, logistics, and resource allocation (Wolsey; Nemhauser, 1998; Conforti; Cornuéjols; Zambelli, 2014). The variables in MIP models can represent binary choices, such as task assignments or other discrete decisions, allowing the model to capture real-world constraints more accurately. In the context of assembly line systems, MIP models are highly suitable for representing problems such as the Assembly Line Worker Assignment and Balancing Problem (ALWABP), as they enable the efficient allocation of workers and tasks while respecting operational constraints.

In addressing the original constraints associated with the ALWABP, several considerations must be taken into account: (i) every task within the system must be executed; (ii) to each workstation must be assigned at least one task; (iii) each worker must be designated to a single workstation; (iv) each workstation receives a single worker; (v) tasks must respect the defined precedence; (vi) a worker can execute more than one task provided that the line cycle time is respected. After this brief explanation, this research will establish a formal definition of the ALWABP, using notations and definitions from Moreira & Costa (2013).

Concerning a context of simple *straight line* assembly system, let S be a set of ordered workstations, W be a set of workers, and $(N, \leq)$ be a partially ordered set of tasks. Consider F_i the sets of direct successors of task i and F_i^* the sets of all successors of task i. Each worker $w \in W \backslash W_i$ executes a task $i \in N$ in time p_{wi}, where W_i is the set of workers that are unable to execute task i. For periods and cycle time, define T as the set of periods, t as the index of each of these rotation periods, and $\overline{C}$ for the maximum mean allowed cycle time. A **period** refers to a smaller segment within a work period during which workers perform specific tasks. The division into periods is essential for implementing job rotation, as it allows workers to switch between different tasks, enhancing task variability and reducing fatigue and monotony.

For instance, if a work period lasts 8 hours, it could be divided into four periods of 2 hours each. During each period, indexed by t in the set T, workers are assigned to different workstations or tasks according to a predefined rotation plan. In this study, the aim is to explore the variability of task assignment while adhering to the maximum mean allowed cycle time, denoted by $\overline{C}$, ensuring that the workload is balanced throughout the workday. So, the variables are:

a) x_{swit}: Binary variable, equals 1 only if task i is assigned to worker w at workstation s at period t;

b) y_{swt}: Binary variable, equals 1 only if worker w is assigned to workstation s at period t;

c) z_{wi}: Binary variable, equals 1 only if worker w is executes task i in at least one period;

d) c_t: Continuous variable, represents the cycle time of period t.

$$\text{Max } z = \sum_{i \in N} \sum_{w \in W \backslash W_i} z_{wi} \quad \text{(Model 1 from Costa \& Miralles (2009))} \tag{2.1}$$

Subject to:

$$\sum_{s\in S}\sum_{w\in W\setminus W_i} x_{swit} = 1, \qquad \forall i \in N, \forall t \in T, \tag{2.2}$$

$$\sum_{i\in N}\sum_{w\in W\setminus W_i} x_{swit} \geq 1, \qquad \forall s \in S, \forall t \in T, \tag{2.3}$$

$$\sum_{s\in S} y_{swt} = 1, \qquad \forall w \in W, \forall t \in T, \tag{2.4}$$

$$\sum_{w\in W} y_{swt} = 1, \qquad \forall s \in S, \forall t \in T, \tag{2.5}$$

$$\sum_{s\in S|s\geq k}\sum_{w\in W\setminus W_i} x_{swit} \leq \sum_{s\in S|s\geq k}\sum_{w\in W\setminus W_j} x_{swjt}, \qquad \forall i \in N, j \in F_i, \forall k \in S, k \neq 1, \forall t \in T, \tag{2.6}$$

$$\sum_{i\in N|w\in W\setminus W_i} p_{wi}\cdot x_{swit} \leq C_t, \qquad \forall s \in S, \forall w \in W, \forall t \in T, \tag{2.7}$$

$$\sum_{t=1}^{|T|} C_t \leq |T|\overline{C}, \tag{2.8}$$

$$\sum_{i\in N|w\in W\setminus W_i} x_{swit} \leq |N| y_{swt}, \qquad \forall s \in S, \forall w \in W, \forall t \in T, \tag{2.9}$$

$$z_{wi} \leq \sum_{t\in T}\sum_{s\in S} x_{swit}, \qquad \forall i \in N, \forall w \in W\setminus W_i, \tag{2.10}$$

$$x_{swit} \in \{0,1\}, \qquad \forall s \in S, \forall i \in N, \forall w \in W\setminus W_i, \forall t \in T, \tag{2.11}$$

$$y_{swt} \in \{0,1\}, \qquad \forall s \in S, \forall w \in W, \forall t \in T, \tag{2.12}$$

$$z_{wi} \in \{0,1\}, \qquad \forall i \in N, \forall w \in W\setminus W_i. \tag{2.13}$$

$$C_t \geq 0, \qquad \forall t \in T. \tag{2.14}$$

The objective function (2.1) maximizes the number of different tasks executed by each worker. Constraints (2.2) – (2.9) enforce that the original constraints of the ALWABP, mentioned previously, are respected for each period. Constraint (2.8) guarantees that the mean cycle time of the rotation period does not exceed the maximum mean allowed cycle time $\overline{C}$. Finally, Constraint (2.10) computes the execution (or not) of the task i by worker w. Variables are presented in Constraints (2.11), (2.12) and (2.13).

2.5 Final remarks

In conclusion, this chapter provided an overview of the theoretical concepts that form the foundation of this study. By exploring the complexity of the Simple Assembly Line Balancing Problem (SALBP) and extending it to the more intricate ALWABP, we have highlighted the computational challenges posed by precedence constraints and worker-task assignments. Additionally, the job rotation examples illustrated the practical difficulties in balancing operational efficiency and human resource allocation. The introduction of various heuristic and metaheuristic algorithms, combined with mathematical programming, underscores the importance of advanced optimization techniques for solving these NP-complete problems.

3 LITERATURE REVIEW

This chapter presents a systematic literature review that aims to explore and synthesize a set of studies on job rotation in assembly lines, making a significant contribution to the existing body of knowledge, particularly due to the classifications it introduces. The research questions are:

a) **RQ1**: "Considering the context of production lines, what are the main job rotation variants? What are the constraints and the objective functions considered?";

b) **RQ2**: "How does the heterogeneity of production lines appear in job rotation planning?";

c) **RQ3**: "What are the primary solution methods for the variants reported in response to the first research question (**RQ1**)?".

The review is conducted using the Scopus database, a comprehensive bibliometric source well-suited for technology and engineering research (Baas *et al.*, 2020). The exclusion criterion considered: (*i*) qualitative articles on job rotation or those unrelated to algorithms; (*ii*) articles that do not belong to the "Coordination for the Improvement of Higher Education Personnel" (CAPES)[1] database. A total of 34 articles published between 2003 and 2025 were selected based on the following structured query:

Figure 3.1 – Query used on the Scopus database.

```
((TITLE-ABS-KEY(("job rotation") AND ("assembly system"OR "assembly
line*"OR "scheduling"OR "assignment"OR "sequencing") AND ("task"OR "job"OR
"worker"OR "skill*"OR "heterogeneous"OR "ergonomy"OR "human factor*") AND
("algorithm*"OR "optimization"OR "metaheuristic*"OR "mathematical
model*"OR "mathematical programming"OR "heuristic*"OR "hybrid")) AND
PUBYEAR > 2002 AND PUBYEAR < 2026) AND (LIMIT-TO (SRCTYPE, "j")) AND
(EXCLUDE (SUBJAREA, "MEDI") OR EXCLUDE (SUBJAREA, "AGRI") OR EXCLUDE
(SUBJAREA, "ENER") OR EXCLUDE (SUBJAREA, "ENVI") OR EXCLUDE (SUBJAREA,
"HEAL") OR EXCLUDE (SUBJAREA, "SOCI")) AND (LIMIT-TO(LANGUAGE, "English"))
AND (LIMIT-TO (DOCTYPE, "ar")))
```

Source: Author

[1] CAPES is responsible for expanding and consolidating graduate programs in Brazil. For more information, we recommend that the reader visits its webpage https://www.gov.br/capes/pt-br.

This review provides an in-depth analysis of selected studies to highlight trends, methodologies, and gaps in the literature. Surveys like Otto & Battaïa (2017), Katiraee *et al.* (2021), Battaïa & Dolgui (2022), and Boysen, Schulze & Scholl (2022) discuss various aspects, such as balancing and workforce differences. Aligned with Moreira & Costa (2013), this research primarily focuses on job rotation within ALWABP. The significance of job rotation has been linked to ergonomic factors and the well-being of assembly line workers, reinforcing the study's motivation (Padula *et al.*, 2017; Comper *et al.*, 2017; Comper *et al.*, 2021; Hashemi-Petroodi *et al.*, 2021). In the following section, you will find a summarized set of constraints, objective functions, heterogeneity factors, and solution methods. For complementary information regarding additional data not included here, please refer to Appendix A through Appendix D.

Also, these classifications are not inherently exclusive. A single study may fall under multiple categories, reflecting the multifaceted nature of job rotation. Some of these studies even address bi-objective optimization, further showcasing the complexity and diversity of the approaches in the literature. By presenting these nuanced classifications, this review provides a valuable framework for understanding the research landscape in job rotation. It opens pathways for more targeted and efficient solutions in future studies.

3.1 RQ1: Variants, Constraints, and Objective Functions in Job Rotation

This section examines the primary variants of job rotation problems within assembly lines, focusing on the constraints and objective functions employed. The review encompasses various approaches and models utilized in the selected studies.

3.1.1 Variants

Notable variants include the Job Rotation Scheduling Problem (JRSP) and the Ergonomic Job Rotation Scheduling Problem (EJRSP), which differ primarily in their emphasis on ergonomic factors. Job rotation plays a significant role in these studies. Figure 3.2 summarizes the variants found.

Figure 3.2 – The main variants concerning job rotation studies.

Variant	References
JRSP	(Tharmmaphornphilas; Norman, 2007; Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; Ayough; Hosseinzadeh; Motameni, 2020; Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021; Zadeh; Movahedi; Shayannia, 2022; Assunção *et al.*, 2022; Ayough *et al.*, 2025)
EJRSP	(Mossa *et al.*, 2016; Hochdörffer; Hedler; Lanza, 2018; Ospina-Mateus *et al.*, 2019; Botti; Mora, 2021; Battini *et al.*, 2022; Abdous *et al.*, 2025)

Source: Author

For instance, Tharmmaphornphilas & Norman (2007) proposes a methodology for creating robust job rotation schedules aimed at reducing the likelihood of low back injuries from lifting. This approach considers uncertain task demands and diverse worker profiles to simulate real-world scenarios. The study starts with deterministic problem versions solved via mathematical programming and extends to heuristic methods for stochastic versions. These methods can also be applied to maximize productivity and reduce exposure to environmental factors, such as excessive noise.

Similarly, Seçkiner & Kurt (2007) presents a novel solution to JRSP focusing on minimizing worker workload, particularly in hazardous jobs. This study highlights the complexity of creating effective rotation schedules in moderately sized service systems with varying customer demands. Using integer programming and a simulated annealing algorithm, the paper presents models and results for test problems, underscoring the efficiency of simulated annealing in solving combinatorial optimization problems. In the sequence, Seçkiner & Kurt (2008) explores the application of ant colony optimization (ACO) to minimize job workload in job rotation scheduling. By systematically varying job exposures among workers, job rotation helps mitigate occupational hazards associated with strenuous tasks. This paper pioneers the application of ACO to job rotation scheduling, aiming to develop an algorithm capable of handling diverse constraints, such as mandatory days off and workload balancing.

Mossa *et al.* (2016) propose an OCRA-based mixed integer nonlinear programming (MINLP) model to optimize job rotation schedules in environments characterized by low-load manual tasks with high repetition, such as assembly lines. This model aims to maximize production rates while balancing and minimizing human workloads within acceptable limits.

The authors employ the OCRA method to evaluate ergonomic risks, a well-recognized approach for assessing upper limb work-related musculoskeletal disorders (UL-WMSDs). The model, tested in an industrial case study, demonstrates its effectiveness in enhancing productivity and reducing ergonomic risks by incorporating workers' performance levels, which are influenced by training and skills.

The study of Hochdörffer, Hedler & Lanza (2018) examines EJRSP as a strategic tool to address challenges posed by demographic changes and workforce heterogeneity in manufacturing, particularly in the automotive industry. The paper focuses on developing a short-term staff planning system using a linear programming-based heuristic to generate comprehensive job rotation schedules. These schedules take into account workers' qualifications, ergonomic exposures, and recent work assignments, aiming to optimize workforce performance, reduce musculoskeletal injuries, and mitigate the impact of an aging workforce. Implemented and tested in a VBA-based software prototype, the approach proves beneficial in enhancing workforce flexibility and health management in dynamic manufacturing environments.

Ospina-Mateus *et al.* (2019) introduce a mathematical model for job rotation that considers ergonomic aspects in repetitive tasks, lifting, and awkward postures in high-variability manufacturing environments. The model, formulated as a multi-objective optimization problem integrating ergonomic constraints, is solved using an improved non-dominated sorting genetic algorithm. This algorithm enhances search convergence on the Pareto front by generating diversified results and maintaining solution diversity.

Also, Ayough, Hosseinzadeh & Motameni (2020) presents a comprehensive study on integrating JRSP and line-cell conversion problems in Seru systems. Highlighting JRSP's importance in enhancing system performance, the research introduces a nonlinear integer programming model to address job rotation scheduling complexities within the Seru framework. This dynamic assignment of operators significantly improves flow time. It reduces the number of required operators, highlighting the role of job rotation in enhancing operator skills, facilitating communication, and achieving faster and more reliable production flows.

In the study proposed by Ayough, Zandieh & Farhadi (2020), the authors investigate the impact of human factors on manufacturing cell performance, with a focus on JRSP. The study presents a multi-period nonlinear mixed-integer model that integrates job assignments and job rotation scheduling with balancing and production sequencing in U-shaped lean manufacturing cells. By considering dynamic operator performance metrics, such as learning, forgetting,

motivation, and boredom, the research highlights how these factors significantly influence cell design efficiency. Dynamic job rotation scheduling is shown to be essential for efficient lean cell design, addressing gaps in the literature that predominantly assumed static operator performance.

Extending previous work Ayough, Farhadi & Zandieh (2021), this study integrates human behavioral and cognitive parameters into lean cell manufacturing job rotation. This study introduces a set-based optimization framework that accounts for the dynamic performance of heterogeneous operators, influenced by factors such as learning, forgetting, motivation, and boredom. This comprehensive integration of human factors with job rotation scheduling provides new insights and practical guidelines for optimizing lean cell productivity.

In the same year, Botti & Mora (2021) introduced an integer linear programming model to optimize activity schedules for aged workers exposed to repetitive work risks. The model, designed for an existing manufacturing system, aims to reduce work-related musculoskeletal disorders (WMSDs) through a bi-objective approach that enhances ergonomic benefits and improves person-job fit. The proposed model ensures acceptable exposure levels to repetitive movement risks, adhering to current laws and international standards (ISO 11228-3, 2007).

Continuing this line of research, Zadeh, Movahedi & Shayannia (2022) explores workforce scheduling in an oil extraction project, emphasizing the importance of balancing system requirements with workforce needs while minimizing costs. The study presents a nonlinear integer model for JRSP, optimizing human resource allocation across different tasks while reducing fatigue and other associated costs. Furthermore, Battini *et al.* (2022) proposes a multi-objective job rotation scheduling model aligned with the Industry 5.0 paradigm. Departing from traditional efficiency-focused approaches, this model adopts a human-centric perspective, integrating socio-technical factors such as workers' experience, physical limitations, ergonomic risks, and perceived boredom. The primary aim is to optimize job assignments and rest-break plans to enhance worker well-being and satisfaction, aligning with Industry 5.0's broader goals of fostering a sustainable and resilient industry.

Still in 2022, Assunção *et al.* (2022) presents a formulation for JRSP tailored to the automotive industry's assembly lines. This study develops job rotation schedules to enhance diversity, ensure homogeneity, and mitigate exposure levels, taking into account biomechanical risk factors, workers' qualifications, and organizational aspects. Compared to manually created schedules, the proposed rotation plans yielded better results in increasing diversity, decreasing exposure, and balancing homogeneity while also being less time-consuming for team leaders.

This approach provides a reliable solution for job rotation in manufacturing, addressing the need for effective strategies to prevent work-related musculoskeletal disorders and improve worker well-being.

Considering now the most recent researches that fall under those criterias, Abdous *et al.* (2025) introduces a scenario-based ergonomic assembly line balancing model explicitly incorporating worker fatigue and recovery into the scheduling process. The authors formulate an Integer Linear Program that considers variability in operation times across multiple scenarios, ensuring that cycle time constraints are met under uncertainty. Ergonomic and fatigue constraints are embedded via a fatigue-and-recovery assessment that limits cumulative exertion and enforces rest allowances. The multi-objective focus balances minimizing health and ergonomic risks with maximizing productivity by keeping line efficiency high across all scenarios.

In contrast, Ayough *et al.* (2025) addresses the Job Rotation Scheduling Problem within Divisional Seru Production Systems (DSPS), aiming to minimize the maximum flow time across cells and rotation periods. The non-linear programming model embeds flow and sequencing constraints to ensure consistency of start and finish times for each worker–cell assignment, and allocation constraints enforce that each operator occupies exactly one cell per rotation period. A multi-functionality constraint guarantees that operators are only assigned to cells for which they are qualified, while rotation-period limits prevent excessive successive assignments without rotation. The single-objective focus is on minimizing overall makespan (maximum flow time), thus balancing workload across the DSPS.

3.1.2 Constraints

This analysis identified fourteen distinct types of constraints, labeled from **C1** to **C14**. The most commonly addressed constraints include: **C1:** Time and Period Constraint, **C2:** Ergonomics and Fatigue Constraint, **C3:** Compatibility Constraint, and **C9:** Allocation and Assignment Constraint. Figure 3.3 summarizes the main constraints found in the studies. Also, less frequent constraints can be found in Appendix A.

Figure 3.3 – The main constraints concerning job rotation studies.

Constraints	**References**
Time and Period Constraint	(Costa; Miralles, 2009; Moreira; Costa, 2013; Ayough; Hosseinzadeh; Motameni, 2020; Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021; Botti; Mora, 2021; Mura; Dini, 2023; Al-Baiti; Cheaitou, 2024; Ospina-Mateus *et al.*, 2019; Mura; Dini, 2021; Shojaei Davood Jafariand; Dokohaki, 2023; Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; McDonald Kimberly P. Ellis; Koelling, 2009; Azizi; Liang, 2013; Abdous *et al.*, 2025)
Ergonomics and Fatigue Constraint	(Otto; Scholl, 2013; Mossa *et al.*, 2016; Botti; Mora, 2021; Mura; Dini, 2023; AlBaiti; Cheaitou, 2024; Mura; Dini, 2021; Zadeh; Movahedi; Shayannia, 2022; Bhadury; Radovilsky, 2006; Abdous *et al.*, 2025)
Compatibility Constraint	(Moreira; Costa, 2013; Hochdörffer; Hedler; Lanza, 2018; Botti; Mora, 2021; Mura; Dini, 2021; Asensio-Cuesta *et al.*, 2012; Mossa *et al.*, 2016; Assunção *et al.*, 2022; Battini *et al.*, 2022; Mura; Dini, 2023; AlBaiti; Cheaitou, 2024)
Allocation and Assignment Constraint	(Allwood; Lee, 2004; Corominas; Pastor; Rodríguez, 2006; Bhadury; Radovilsky, 2006; Butkovič; Lewis, 2007; Tharmmaphornphilas; Norman, 2007; Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; Asawarungsaengkul; Nanthavanij, 2008; Costa; Miralles, 2009; McDonald Kimberly P. Ellis; Koelling, 2009; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010; Asensio-Cuesta *et al.*, 2012; Moreira; Costa, 2013; Azizi; Liang, 2013; Otto; Scholl, 2013; Yaoyuenyong; Nanthavanij, 2006; Ayough *et al.*, 2025)

Source: Author

3.1.2.1 Time and Period Constraint

These constraints focus on time management and task sequencing, incorporating cycle times, full work periods, task precedence, and idle times. Studies such as Costa & Miralles (2009), Moreira & Costa (2013), Ayough, Hosseinzadeh & Motameni (2020), Ayough, Zandieh & Farhadi (2020), Ayough, Farhadi & Zandieh (2021), Botti & Mora (2021), Mura & Dini (2023), Abdous *et al.* (2025) address time constraints as cycle times. On the other hand, AlBaiti

& Cheaitou (2024) presents a specific constraint requiring a worker to be assigned for a full day if they are allocated in the first period of that day.

More commonly, Costa & Miralles (2009), Moreira & Costa (2013), Ospina-Mateus *et al.* (2019), Ayough, Zandieh & Farhadi (2020), Mura & Dini (2021), Mura & Dini (2023), Shojaei Davood Jafariand & Dokohaki (2023) discuss task precedence constraints, where a task can only begin after all preceding tasks have been completed. Additionally, Shojaei Davood Jafariand & Dokohaki (2023) addresses idle time constraints, intervals between tasks, and similar constraints, as seen in (Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; McDonald Kimberly P. Ellis; Koelling, 2009; Azizi; Liang, 2013; Botti; Mora, 2021). Abdous *et al.* (2025) also supports dynamic policies (rotation, scheduled breaks), but these are decision implementations and do not appear as explicit constraints in the ILP formulation.

3.1.2.2 Ergonomics and Fatigue Constraint

These constraints pertain to worker health and well-being, addressing ergonomic risk, fatigue, and noise exposure. A unified "Worker Health and Well-Being" constraint can monitor and limit these factors. Studies like Otto & Scholl (2013), Mossa *et al.* (2016), Botti & Mora (2021), Mura & Dini (2023), AlBaiti & Cheaitou (2024) present ergonomic constraints as risk levels. Conversely, studies such as Mura & Dini (2021), Zadeh, Movahedi & Shayannia (2022), Abdous *et al.* (2025) address these constraints in terms of fatigue levels. Mossa *et al.* (2016) explores balancing ergonomic risks among collaborators, while Bhadury & Radovilsky (2006) presents a specific constraint on maximum noise exposure for each worker.

3.1.2.3 Compatibility Constraint

These constraints ensure the proper matching of tasks with stations or workers, ensuring that tasks are allocated to the appropriate stations and that workers are assigned tasks compatible with their skills and abilities. Studies like Moreira & Costa (2013), Hochdörffer, Hedler & Lanza (2018), Botti & Mora (2021), Mura & Dini (2021) address task/workstation compatibility, while Asensio-Cuesta *et al.* (2012), Moreira & Costa (2013), Mossa *et al.* (2016), Hochdörffer, Hedler & Lanza (2018), Assunção *et al.* (2022), Battini *et al.* (2022), Mura & Dini (2021), Mura & Dini (2023), AlBaiti & Cheaitou (2024) discuss worker/task compatibility.

These constraints can appear simultaneously in a model, depending on the problem's granularity. Typically, more specific and granular problem definitions increase complexity but also depend on the solving methods applied. Some studies simplify the modeling process to address this complexity. An interesting example is Botti & Mora (2021), which presents workstation compatibility where each worker performs one specific task, effectively merging the notions of station and task allocation.

3.1.2.4 Allocation and Assignment Constraint

These constraints involve the allocation of tasks and workers within a period, presented in various forms. For instance, each task may be assigned to a single worker in a period (Allwood; Lee, 2004; Corominas; Pastor; Rodríguez, 2006; Bhadury; Radovilsky, 2006; Butkovič; Lewis, 2007; Tharmmaphornphilas; Norman, 2007; Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; Asawarungsaengkul; Nanthavanij, 2008; Costa; Miralles, 2009; McDonald Kimberly P. Ellis; Koelling, 2009; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010; Asensio-Cuesta *et al.*, 2012; Moreira; Costa, 2013; Azizi; Liang, 2013; Otto; Scholl, 2013).

Alternatively, each worker might be assigned to a single task in a period (Allwood; Lee, 2004; Corominas; Pastor; Rodríguez, 2006; Bhadury; Radovilsky, 2006; Butkovič; Lewis, 2007; Tharmmaphornphilas; Norman, 2007; Seçkiner; Kurt, 2008; Azizi; Liang, 2013; Ayough *et al.*, 2025), while in other contexts, workers can have multiple tasks in a period (McDonald Kimberly P. Ellis; Koelling, 2009; Costa; Miralles, 2009; Otto; Scholl, 2013; Azizi; Liang, 2013; Moreira; Costa, 2013).

Constraints also appear where a worker can have at most one task in a period (Seçkiner; Kurt, 2007; Asawarungsaengkul; Nanthavanij, 2008; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010). Additionally, these constraints can apply to workstations, as seen in Yaoyuenyong & Nanthavanij (2006), which presents specific characteristics where each worker is assigned to at most one station per period, and each station must have a single worker assigned to it per period.

3.1.3 Objective Functions

In total, twelve different types of objective functions were defined, ranging from **F1** to **F12**. The most frequent objective functions include: **F1:** Maximize productivity and

competence, **F2:** Minimize health and ergonomic risks, **F6:** Minimize overall operational costs. Figure 3.4 summarizes the main objective functions found in the studies. Also, less frequent objective functions can be found in Appendix B.

Figure 3.4 – The main objective functions concerning job rotation studies.

Objective Function	**References**
Maximize productivity and competence	(Allwood; Lee, 2004; Michalos *et al.*, 2010; Michalos; Makris; Mourtzis, 2011; Mossa *et al.*, 2016; Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021; Botti; Mora, 2021; Battini *et al.*, 2022; AlBaiti; Cheaitou, 2024; Abdous *et al.*, 2025)
Minimize health and ergonomic risks	(Asensio-Cuesta *et al.*, 2012; Otto; Scholl, 2013; Ospina-Mateus *et al.*, 2019; Botti; Mora, 2021; Assunção *et al.*, 2022; Battini *et al.*, 2022; Mura; Dini, 2023; AlBaiti; Cheaitou, 2024; Abdous *et al.*, 2025)
Minimize overall operational costs	(Bhadury; Radovilsky, 2006; Butkovič; Lewis, 2007; McDonald Kimberly P. Ellis; Koelling, 2009; Michalos *et al.*, 2010; Michalos; Makris; Mourtzis, 2011; Azizi; Liang, 2013; Hochdörffer; Hedler; Lanza, 2018; Mura; Dini, 2021; Zadeh; Movahedi; Shayannia, 2022; Mura; Dini, 2023; Shojaei Davood Jafariand; Dokohaki, 2023)

Source: Author

3.1.3.1 Maximize productivity and competence

This type of objective function was introduced by (Allwood; Lee, 2004; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010; Mossa *et al.*, 2016; Ayough; Zandieh; Farhadi, 2020; Botti; Mora, 2021; Ayough; Farhadi; Zandieh, 2021; Battini *et al.*, 2022; AlBaiti; Cheaitou, 2024; Abdous *et al.*, 2025) with a focus on productivity. The main differences lie in the specifics of how productivity is accounted for in each case.

On the other hand, Michalos *et al.* (2010), Michalos, Makris & Mourtzis (2011) present similar ideas but differently; they present and measure the notion of "competence" within the job rotation schedule, using a competency-based fuzzy model. In these studies, competence is defined as the ability of an operator to perform an assembly task without errors. Given its subjective nature, quantifying competence is a challenging task. Therefore, fuzzy logic is employed to handle this subjectivity.

3.1.3.2 Optimize health and minimize ergonomic risks

These objective functions are mostly related to ergonomic factors. Asensio-Cuesta *et al.* (2012) presented a model that aimed to minimize exposure to repetitive movements of the same muscle groups, an interesting approach, especially in manufacturing industries. Other studies may have contributed to similar aspects, but those were considering other factors, using concepts like “fatigue”, “boredom”, or even “exhaustion” levels.

Additionally, studies such as Bhadury & Radovilsky (2006), Azizi & Liang (2013), Ospina-Mateus *et al.* (2019), Botti & Mora (2021), Assunção *et al.* (2022), Battini *et al.* (2022), Abdous *et al.* (2025) were more comprehensive, aiming to minimize ergonomic risks and promote mental health, presenting similar approaches to understanding the concept of ergonomic risk for tasks performed by each worker. Some studies, such as Moreira & Costa (2009), Moreira & Costa (2013), present a more abstract vision to address the issue, focusing on maximizing the variability of tasks among workers.

Moreover, we have Mura & Dini (2023), which aimed to minimize noise exposure levels, presenting a well-based study and categorizing the noise exposure level. The noise exposure is evaluated in terms of the sound pressure level, according to OSHA 1910.95 regulations for occupational noise exposure. Very similar to what AlBaiti & Cheaitou (2024) did while trying to minimize the exposure to vibration, presenting how each machine used by the workers had a different impact level on this ergonomic aspect.

3.1.3.3 Minimize overall operational costs

These objective functions are less inclined to ergonomics than the previous ones presented in this section. It is possible to highlight studies like Bhadury & Radovilsky (2006), Butkovič & Lewis (2007), McDonald Kimberly P. Ellis & Koelling (2009), Michalos *et al.* (2010), Michalos, Makris & Mourtzis (2011), Azizi & Liang (2013), Hochdörffer, Hedler & Lanza (2018), Mura & Dini (2021), Zadeh, Movahedi & Shayannia (2022), Mura & Dini (2023), Shojaei Davood Jafariand & Dokohaki (2023), presenting the cost notion more simply, considering the number of hours worked, role, or even abstracting even more and considering a fixed “cost” to each worker assigned.

Some studies, such as McDonald Kimberly P. Ellis & Koelling (2009), Azizi & Liang (2013), are more specific and address less common notions, like training cost. Azizi & Liang (2013) also presented working flexibility cost and unmet demand cost, while McDonald Kimberly P. Ellis & Koelling (2009) have also considered inventory cost and even the cost called "poor quality". The cost of "poor quality" is incurred whenever defective units are produced.

3.2 RQ2: Heterogeneity in Job Rotation Planning

This section examines how the heterogeneity of production lines is taken into account in job rotation planning. It reviews how differences in tasks, skills, and worker attributes are incorporated into job rotation models and strategies. Figure 3.5 summarizes the main heterogeneous factors found in the studies.

Figure 3.5 – The main heterogeneity factors concerning job rotation studies.

Heterogeneity	**References**
Skill Level/Factor	(Corominas; Pastor; Rodríguez, 2006; Tharmmaphornphilas; Norman, 2007; McDonald Kimberly P. Ellis; Koelling, 2009; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010; Asensio-Cuesta *et al.*, 2012; Azizi; Liang, 2013; Mossa *et al.*, 2016; Hochdörffer; Hedler; Lanza, 2018; Ayough; Hosseinzadeh; Motameni, 2020; Botti; Mora, 2021; Assunção *et al.*, 2022; Mura; Dini, 2021; Battini *et al.*, 2022; Mura; Dini, 2023; AlBaiti; Cheaitou, 2024)
Learning Coefficient	(Allwood; Lee, 2004; Azizi; Liang, 2013; Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021; Shojaei Davood Jafariand; Dokohaki, 2023)
Task Execution Time	(McDonald Kimberly P. Ellis; Koelling, 2009; Costa; Miralles, 2009; Moreira; Costa, 2013; Mossa *et al.*, 2016; Ayough; Hosseinzadeh; Motameni, 2020; Botti; Mora, 2021; Abdous *et al.*, 2025; Ayough *et al.*, 2025)

Source: Author

3.2.1 Heterogeneity Factors

Fifteen heterogeneity factors were acknowledged, from **H1** to **H14**. The most common heterogeneity factors include: **H1:** Skill Level/Factor, **H2:** Learning Coefficient, **H12:** Task Execution Time. Also, less frequent heterogeneity factors can be found in Appendix C.

3.2.1.1 Skill Level/Factor

This heterogeneity is observed in most of the studies reviewed. Corominas, Pastor & Rodríguez (2006), Tharmmaphornphilas & Norman (2007), McDonald Kimberly P. Ellis & Koelling (2009), Nanthavanij, Yaoyuenyong & Jeenanunta (2010), Asensio-Cuesta *et al.* (2012), Azizi & Liang (2013), Mossa *et al.* (2016), Hochdörffer, Hedler & Lanza (2018), Ayough, Hosseinzadeh & Motameni (2020), Botti & Mora (2021), Assunção *et al.* (2022), Mura & Dini (2021), Battini *et al.* (2022), Mura & Dini (2023), AlBaiti & Cheaitou (2024) presented that workers have a skill level, and their assignment depends on that skill level. They are assigned to skills or workstations for which their level is a good match.

Although there are exceptions, such as Hochdörffer, Hedler & Lanza (2018), which also demonstrated the possibility of allocating under-qualified workers, provided that it is accompanied by an overqualified worker, to offer training to less qualified workers. However, the effectiveness of incorporating this idea into the mathematical model has not been proven.

3.2.1.2 Learning Coefficient

Studies like Allwood & Lee (2004), Azizi & Liang (2013), Ayough, Zandieh & Farhadi (2020), Ayough, Farhadi & Zandieh (2021), Shojaei Davood Jafariand & Dokohaki (2023) present the idea of a learning coefficient. It is used to apply a "real-time impact" during the optimization, creating a heterogeneity factor even if the workers previously did not have one. It considers this coefficient (k^a) to be used as a factor to impact the execution time of tasks and also as a risk-reducing factor in the ergonomic aspect.

3.2.1.3 Task Execution Time

This is a prevalent heterogeneity factor, presented in McDonald Kimberly P. Ellis & Koelling (2009), Costa & Miralles (2009), Moreira & Costa (2013), Mossa *et al.* (2016), Ayough, Hosseinzadeh & Motameni (2020), Botti & Mora (2021), Abdous *et al.* (2025), Ayough *et al.* (2025). Additionally, the task execution time difference can be combined with the idea presented in 3.2.1.2 to increase or reduce the difference between workers' execution times or even to create this difference in cases where workers have the same task execution time.

3.3 RQ3: Solution Methods for Job Rotation Variants

This section identifies the primary solution methods reported for the job rotation variants discussed in RQ1. It will provide an overview of the algorithms, heuristics, and metaheuristics employed in the selected studies to solve job rotation problems in assembly systems. Figure 3.6 summarizes the primary solution methods found in the studies. Also, less frequent solution methods can be found in Appendix D.

Figure 3.6 – The main solution methods concerning job rotation studies.

Solution method	**References**
Solver-based solution	(Bhadury; Radovilsky, 2006; McDonald Kimberly P. Ellis; Koelling, 2009; Costa; Miralles, 2009; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010; Mossa *et al.*, 2016; Hochdörffer; Hedler; Lanza, 2018; Botti; Mora, 2021; Battini *et al.*, 2022; Shojaei Davood Jafariand; Dokohaki, 2023; AlBaiti; Cheaitou, 2024; Abdous *et al.*, 2025)
Genetic algorithm	(Asawarungsaengkul; Nanthavanij, 2008; Asensio-Cuesta *et al.*, 2012; Moreira; Costa, 2013; Ospina-Mateus *et al.*, 2019; Assunção *et al.*, 2022; Mura; Dini, 2021)
Constructive heuristic	(Yaoyuenyong; Nanthavanij, 2006; Tharmmaphornphilas; Norman, 2007; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010; Otto; Scholl, 2013; Azizi; Liang, 2013; Moreira; Costa, 2013; Ayough; Farhadi; Zandieh, 2021)
Local searches	(Moreira; Costa, 2013; Otto; Scholl, 2013; Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021)

Source: Author

3.3.1 Algorithms, heuristics, and metaheuristics

Regarding the algorithms used to solve the presented problems, eighteen were identified, ranging from **A1** to **A17**. The most frequent algorithms are: **A2:** MIP Solver-based solution, **A5:** Genetic algorithm, **A7:** Constructive heuristic, **A8:** Local searches.

3.3.1.1 Solver-based solution

MIP Solver-based solutions are standard in job rotation planning due to their precision and ability to handle complex constraints. Studies like Bhadury & Radovilsky (2006), McDonald Kimberly P. Ellis & Koelling (2009), Costa & Miralles (2009), Nanthavanij, Yaoyuenyong & Jeenanunta (2010), Mossa *et al.* (2016), Hochdörffer, Hedler & Lanza (2018), Botti & Mora (2021), Battini *et al.* (2022), Shojaei Davood Jafariand & Dokohaki (2023), AlBaiti & Cheaitou (2024) employed solvers to optimize job rotations, leveraging their robust computational capabilities to find optimal or near-optimal solutions.

3.3.1.2 Genetic algorithm

Genetic algorithms (GAs) are popular due to their adaptability and efficiency in finding good solutions within a reasonable time. They mimic the process of natural selection to explore a vast solution space effectively. This approach is demonstrated in studies like Asawarungsaengkul & Nanthavanij (2008), Asensio-Cuesta *et al.* (2012), Moreira & Costa (2013), Ospina-Mateus *et al.* (2019), Assunção *et al.* (2022), Mura & Dini (2021), where GAs were used to optimize job rotations by evolving populations of solutions over successive generations.

3.3.1.3 Constructive heuristic

Constructive heuristics build solutions step by step, making locally optimal choices at each stage. This method is valued for its simplicity and speed. Notable applications are found in Yaoyuenyong & Nanthavanij (2006), Tharmmaphornphilas & Norman (2007), Nanthavanij, Yaoyuenyong & Jeenanunta (2010), Otto & Scholl (2013), Azizi & Liang (2013), Moreira & Costa (2013), Ayough, Farhadi & Zandieh (2021), where constructive heuristics efficiently tackled job rotation problems by incrementally constructing feasible solutions.

3.3.1.4 Local searches

Local search algorithms explore the neighborhood of current solutions to find improved ones, making them helpful in refining existing solutions. They are particularly effective for

problems with large solution spaces. Studies such as Moreira & Costa (2013), Otto & Scholl (2013), Ayough, Zandieh & Farhadi (2020), Ayough, Farhadi & Zandieh (2021) utilized local searches to enhance the quality of job rotation schedules, demonstrating their capability to converge on high-quality solutions through iterative improvements.

3.4 Final remarks

Upon reviewing the research questions for RQ1, which addresses the main variants of job rotation in production lines, the literature identifies two key types: the Job Rotation Scheduling Problem (JRSP) and the Ergonomic Job Rotation Scheduling Problem (EJRSP). These variants consider several constraints, including time and period constraints, ergonomic and fatigue constraints, compatibility constraints, and allocation and assignment constraints. The objective functions typically focus on maximizing productivity and worker competence, minimizing health and ergonomic risks, and reducing overall operational costs. For RQ2, the heterogeneity of production lines influences task rotation planning through variables such as worker skill levels, learning curves, and task execution times. Lastly, for RQ3, the primary solution methods discussed in the literature involve using MIP solvers, genetic algorithms, constructive heuristics, and local search techniques to address the complexities of job rotation and task assignment.

Based on the comprehensive analysis and review of the existing literature, we note that the problem addressed by Costa & Miralles (2009), Moreira & Costa (2013) corresponds to the JRALWABP considered in this study. The specific approach and methodology proposed in (Moreira; Costa, 2013) do not represent the majority of the approaches presented in the current research. Although extensive studies have been conducted in related areas, a lack of effective mechanisms remains for selecting the best solutions and generating diverse alternative solutions. These gaps are hypothetically crucial as the diversity and quality of solutions for each period directly impact the outcomes. Therefore, this study not only addresses these scientific opportunities but also has the potential to make substantial contributions to the field, highlighting its originality and significance.

In addition to addressing the identified gaps, this work aims to contribute to the existing body of knowledge by proposing new exact models and a hybrid framework, further developing and evolving ideas previously introduced in the literature. By combining precise methods from

Moreira & Costa (2013), Borba & Ritt (2014) with heuristic and metaheuristic approaches, the study seeks to enhance solution efficiency and quality, particularly in more complex instances of JRALWABP. This hybrid approach builds upon the foundational work of Moreira & Costa (2013) and introduces innovative mechanisms for solution generation and selection, ultimately aiming to enhance the diversity of solutions across multiple periods and improve the overall performance of the problem-solving process. Through this, the study contributes not only new theoretical insights but also practical advancements, further positioning itself within the broader context of systematic reviews and current research trends.

4 METHODOLOGY

This chapter presents the specific procedures, techniques, and approaches employed to provide a comprehensive overview of the research design. The focus is on detailing the steps taken to address the research objectives, clarifying the rationale behind the chosen methodologies, and offering insight into how the study is structured and executed. Through this presentation, readers will gain a deeper understanding of the thought processes and practical considerations that informed the research design, thereby fostering transparency and facilitating an evaluation of the study's rigor and validity. The goal is to enhance the solution strategy proposed by Moreira & Costa (2013) by incorporating new algorithmic features, as outlined in the following sections.

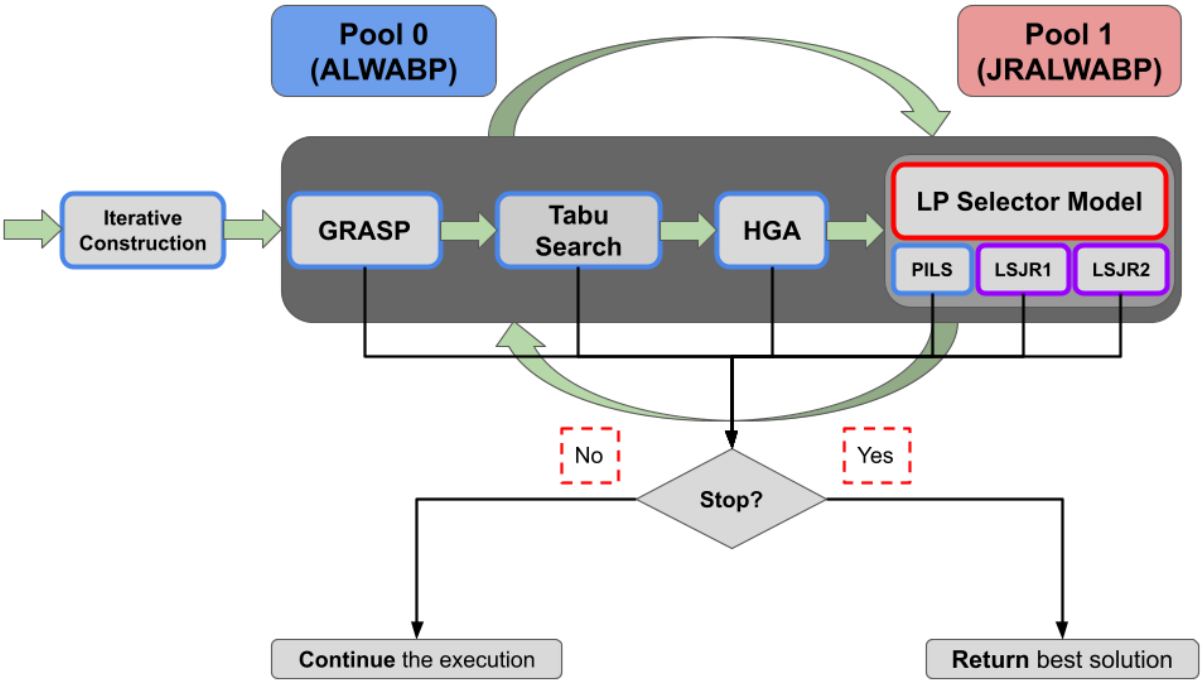
4.1 Pool Generation

The strategy of generating a diverse pool of solutions to enhance intensity during the search process has proven effective in the literature, as noted by (Jaradat; Ayob; Almarashdeh, 2016). In this phase, we implement an iterative process designed to develop a pool of solutions for ALWABP. The goal is to increase the diversity of the solutions generated within a specified timeframe, using heuristics to escape local minima. This approach not only improves the efficiency of generating the solution pool but also enriches the variety of solutions obtained, leading to a more thorough exploration of the optimization landscape. To illustrate the proposed algorithm, Figure 4.1 is presented, highlighting the key contributions of this study. For additional clarity, further information on Iterative Construction/GRASP, Tabu Search, and BRKGA can be found in Sections 4.2.1, 4.2.2 and 4.2.3, respectively.

Concerning the local searches, the procedures labeled LSJR1 and LSJR2 are mixed-integer programming (MIP) methods proposed by Moreira & Costa (2013). It is important to note that the local search methods, including the PILS, LSJR1, and LSJR2, will only be utilized if the JRALWABP solution has not improved during that specific iteration. Our approach enables iterative processing, allowing the local searches included in the "LP Selector Model" component to introduce new solutions into Pools 0 and 1. The colors used to outline each minor component indicate which pools it can contribute solutions to: Blue for Pool 0, Red for Pool 1, and Purple for Both Pools. The green arrows are pointing out the execution flow of the algorithm.

The flowchart serves as a visual guide to the innovative steps and methodologies that set our algorithm apart from existing approaches. By clearly delineating each phase of the process, we aim to provide a transparent overview of how our algorithm enhances current solutions. This visual representation not only highlights the originality of our proposal but also facilitates an understanding of its operational dynamics and potential impact. The Figure 4.1 details the HAJR2 flow. Also, Figure 4.2 details differences about Moreira & Costa (2013) approach, the updates made to HAJR1, and also our proposal, the HAJR2.

Figure 4.1 – Diagram presenting the study's algorithm proposal (HAJR2).



Source: Author

Figure 4.2 – Comparison of Moreira & Costa (2013), HAJR1, and HAJR2.

Feature	**Moreira & Costa (2013)**	**HAJR1**	**HAJR2**
Single iteration	✓	✓	✗
Single long heuristic execution	✓	✓	✗
30-minute time limit	✗	✓	✓
Unlimited iterations	✗	✗	✓
Multiple short heuristic executions	✗	✗	✓
Always applies LSJR1/LSJR2	✓	✓	✗
LSJR1/LSJR2 only when not improving	✗	✗	✓
Uses PILS heuristic	✗	✗	✓
Tabu calibrated using irace	✗	✓	✓
HGA calibrated using irace	✗	✓	✓

Source: Author

4.2 Proposal: Step by Step

Regarding the new solving approaches, we create fine-tuned versions of the GRASP and Tabu Search metaheuristics presented in Moreira & Costa (2013), utilizing IRace (López-Ibáñez *et al.*, 2016) since it is one of the most popular parametrization tools (Souza *et al.*, 2021). Instead of relying solely on the last generation of BRKGA, as mentioned by Moreira & Costa (2013), all heuristics receive as input one solution pool and can populate one or multiple pools. By combining iterative methods with innovative solution generation techniques, this proposal increases the number of different solutions found, providing a more robust foundation for subsequent stages of the optimization process.

4.2.1 Implemented Iterative Construction & GRASP

The Greedy Randomized Adaptive Search Procedure (GRASP) (Feo; Resende, 1995) is an iterative process consisting of construction and local search phases. During the construction phase, a feasible solution is built incrementally, where each element is selected based on a randomized greedy function. The local search phase then refines this solution to find a local

optimum. GRASP is particularly effective in solving combinatorial optimization problems where the solution space is vast, such as scheduling (Pinedo, 2022) and routing problems (Toth; Vigo, 2014). Its adaptability and efficiency make it suitable for generating diverse solutions in the context of job rotation, where multiple feasible schedules need to be explored.

Our GRASP implementation starts by reading the input pools and the parameter set. The core of the method is a loop over the 32 task-priority rules. See Appendix E for more details. For each rule, a neighborhood selection procedure is executed, attempting to construct a solution with the lowest possible cycle time. If the construction fails, the cycle time limit is incrementally increased until a solution is found. Upon successful construction, the BLM local search is applied to improve the solution further—note that this local search is exclusive to GRASP; the Iterative Construction approach only executes the construction strategy. If the resulting solution is feasible and better than the current best, it is added to the destination pool. After each cycle, the stopping criteria are updated, and the inner loop terminates when any of the predefined conditions are met. Once all 32 rules have been evaluated, the destination pool is returned.

The Iterative Construction, aside from the local search, also diverges a bit from GRASP during the construction process. While GRASP introduces randomness by selecting tasks probabilistically after ranking them with the task priority rules—allowing for different solutions across multiple runs—Iterative Construction follows a strictly greedy and deterministic approach. It adheres rigidly to the priority rules, always selecting the highest-priority task available at each step. As a result, it consistently produces the same solution regardless of the random seed, in contrast to GRASP's behavior.

Figure 4.3 – Greedy Randomized Adaptive Search Procedure (GRASP) pseudocode.

Algorithm 2 Greedy Randomized Adaptive Search Procedure (GRASP)

Require: Origin pool of solutions, destination pool
Ensure: Destination pool containing improved solutions
Select an initial solution from the origin pool
Evaluate the initial solution
for each neighborhood selection strategy **do**
    Initialize a temporary pool with the initial solution
    while stopping criteria not met **do**
        Apply the construction phase to generate a new solution
        Evaluate the constructed solution
        if the constructed solution is invalid **then**
            Continue to the next iteration
        Apply local search to improve the constructed solution
                ▷ IterativeConstruction does not use local search
        Evaluate the improved solution
        if the improved solution is feasible and penalty-free **then**
            Add it to the destination pool
        Update stopping criteria
return destination pool

Source: Author

4.2.2 Implemented Tabu Search

Tabu Search (Glover; Laguna, 1998) is a metaheuristic designed to guide local search procedures beyond local optima by using adaptive memory structures. It avoids cycles and encourages the exploration of diverse regions in the solution space by maintaining a list of tabu (forbidden) moves—recent modifications that are temporarily banned unless they meet an aspiration criterion. This makes Tabu Search especially effective for complex combinatorial problems such as job shop scheduling (Jain; al., 2022) and network design (Fortz; Thorup, 2000). In the context of job rotation in assembly lines, it efficiently navigates large and rugged solution landscapes, avoiding premature convergence and producing high-quality solutions.

Algorithm 4.4 outlines our adaptation of Tabu Search for the ALWABP. The process begins by identifying the best solution in the original pool, which is then used to initialize both the current and the best-known solution. The main loop iterates until a predefined stopping criterion is met, systematically exploring task and worker neighborhoods.

Each iteration begins with the selection of a neighborhood structure—either swap-and-reassign moves involving tasks or workers. The entire neighborhood is generated for the current solution, and each candidate move is evaluated by temporarily applying it. If the move is classified as tabu and fails the aspiration criterion—specifically, if it does not improve the global best—it is immediately undone and discarded. Otherwise, it is compared against other candidates to identify the best admissible move within the current neighborhood.

Once all moves have been considered, the best non-tabu move (if one exists) is permanently applied. The tabu list is then updated with this move, and its tenure is typically set proportional to the iteration limit, preventing short-term cycling. If the resulting solution is both feasible and penalty-free, it is added to the destination pool. Furthermore, if it represents an improvement over the current global best, the best solution is updated accordingly.

To escape local minima and encourage exploration, every 500 iterations the algorithm enters an `aspiration phase`, alternating between intensification and diversification strategies. During the `intensification phase`, the most frequently occurring task–station and worker–station assignments are fixed to reinforce good patterns. A focused local search (LS1) is performed over refined neighborhoods—`task-intensification` and `worker-intensification`—to exploit the promising region more thoroughly.

Conversely, in the `diversification phase`, the algorithm fixes the least frequently observed assignments, deliberately perturbing the solution by applying the `task-diversification` and `worker-diversification` neighborhoods. This aims to explore less-visited areas of the solution space. After perturbation and local search, the best resulting solution is selected. If this solution is both feasible and improves upon the global best, the solution and destination pool are updated.

After executing the selected intensification or diversification phase, the `intensification` flag is toggled to ensure alternating behavior in future checkpoints. Additionally, the current solution may be reset to the best solution found so far, reinforcing the exploitation of high-quality areas when no improvement occurs. This balance of strategic intensification and diversification continues until the stopping criteria are met, at which point the enriched destination pool is returned.

Figure 4.4 – Tabu Search pseudocode.

Algorithm 3 Tabu Search

Require: Origin solution, destination pool
Ensure: Destination pool with improved solutions
Set the best solution as the best solution from the origin pool
Initialize the current solution as the best solution
while stopping criteria not met **do**
    Select a neighborhood
    Build the neighborhood for the current solution
    Identify the best move in the neighborhood according to acceptance criteria
    if a valid best move is found **then**
        Apply the best move to the current solution
        if the current solution is feasible and penalty-free **then**
            Add it to the destination pool
            if it improves over the best solution **then**
                Update the best solution
        Update tabu list with the best move
    if intensification/diversification criteria are met **then**
        Apply perturbation strategies combining neighborhood search and local search
        Select the best solution after perturbation
        if the perturbated solution is feasible and improves the best **then**
            Update the best solution and destination pool
    Optionally reset current solution to best solution
return destination pool

Source: Author

4.2.3 Implemented BRKGA & HGA

Biased Random Key Genetic Algorithm (BRKGA) (Gonçalves; Resende, 2011; Londe *et al.*, 2023) is a variant of the Genetic Algorithm where solutions are encoded as vectors of random keys—real values in the range $[0, 1]$. A biased selection mechanism ensures that elite individuals (i.e., those with better fitness) have a higher probability of passing their genes to the next generation, increasing convergence speed while preserving diversity. This approach has proven effective for various combinatorial optimization problems, such as vehicle routing (Muthuswamy; al., 2022) and resource allocation (Chiarandini; al., 2010). In the context of job rotation, BRKGA can generate diverse and high-quality solutions by leveraging its robust genetic operators and preference for higher-quality individuals. Recent advances concerning random key optimizers can be found in (Chaves *et al.*, 2024).

Our implementation of BRKGA represents each candidate solution as a vector of real-valued keys in the interval $[0, 1]$, each corresponding to a decision variable. At each generation, the population is divided into three segments: a fixed fraction of the best-performing individuals (the elite), a small fraction of randomly generated individuals (mutants), and the remaining individuals generated through biased crossover. For each gene position j in the offspring, the value is inherited from the elite parent with probability β, and from the non-elite parent with probability $1 - \beta$. This biased reproduction scheme steers the search process toward promising regions of the solution space while preserving exploratory capability.

A key component is the decoding phase, in which each vector of keys is converted into a feasible solution using the Iterative Construction heuristic described in Section 4.2.1. In this context, the standard priority rules used by GRASP are replaced by the ranks induced by the random keys. Since both approaches rely on a priority matrix internally, the substitution is straightforward and allows BRKGA to integrate seamlessly with the construction logic.

The complete procedure is summarized in Algorithm 4.5. After initializing the population and evaluating the first generation, the algorithm enters a generational loop that continues until a stopping criterion is met (e.g., a maximum number of generations or a time limit). In each iteration, the best individual is decoded and evaluated. If the resulting solution is both feasible and penalty-free, it is added to the destination pool. Additionally, the current best solution is updated according to the acceptance criterion.

In the HGA (Hybrid Genetic Algorithm) variant, a local search phase is applied immediately after decoding the random keys, a multiple-best-improvement strategy (BLM), which intensifies the search around promising solutions. In contrast, BRKGA disables this feature by setting `local_search = None`, keeping the structure otherwise identical.

This modular implementation allows consistent integration across variants and facilitates experimentation with or without hybrid improvement.

Figure 4.5 – Biased Random-Key Genetic Algorithm (BRKGA) pseudocode.

Algorithm 4 Biased Random-Key Genetic Algorithm (BRKGA)

Require: Origin pool of solutions, destination pool
Ensure: Destination pool containing improved solutions
1: Initialize a random population of encoded solutions
2: Define the evolutionary problem with decoder and evaluation logic
3: Execute one initial generation to process the initial population
4: Identify the best solution from the initial population
5: **while** stopping criteria not met **do**
6: Perform one generation of the genetic algorithm
7: Decode the best individual from the current generation into a solution
▷ HGA: After decoding, a local search is applied
8: Evaluate the decoded solution
9: **if** the solution is feasible and penalty-free **then**
10: Add it to the destination pool
11: **if** the solution is accepted as new best (according to acceptance criteria) **then**
12: Update the best solution
13: **return** destination pool

Source: Author

4.2.4 Modifications to the Linear Programming Model of Moreira & Costa (2013)

The formulations employed by the "LP Selector Model" and the local searches LSJR1 and LSJR2 largely mirror those of Moreira & Costa (2013) as shown in Section 2.4, with two key enhancements. First, the parameter $\overline{C}$ is now a decision variable—its value is directly integrated into the objective function rather than serving as a fixed upper bound on cycle time. Second, the original objective, Equation 2.1, has been reformulated to jointly maximize the total number of distinct tasks performed by workers and minimize any increase in the maximum average cycle time, thereby addressing the problem's inherently multi-objective nature. Consequently, our revised objective function, replacing Equation 2.1, is:

$$\text{Max} \left[c_1 \left(\frac{\sum_{i \in N} \sum_{w \in W \setminus W_i} z_{wi}}{Z_{\max}} \right) \right] - \left[c_2 \left(\frac{\overline{C}}{C_{\max}} \right) \right] \tag{4.1}$$

Where:

a) $Z_{\max}$ represents the maximum possible variation in task assignments for a given instance, determined by the total number of tasks that each worker can perform;

b) $C_{\max}$ is the highest average cycle time recorded in Pool 0 to date;

c) c_1 and c_2 are weighting coefficients that can be inputted, with values ranging from 0 to 1. c_1 is provided as input, while c_2 is calculated as $1 - c_1$. These coefficients adjust the relative importance of each term in the objective function.

4.2.5 Local Searches

Originally, the local searches LSJR1 and LSJR2 were applied after the LP Selector Model was executed, only once. Considering the iterative nature of the proposal, the local searches now include the Pattern Injection Local Search (PILS) and these methods will only be applied if, in the current iteration, the LP Selector Model has not improved the current best JRALWABP solution known. The following subsections will provide a better understanding of their functionality.

4.2.5.1 LSJR1: Local Search for Job Rotation 1 from Moreira & Costa (2013)

The first procedure, named local search for job rotation 1 (LSJR1) (Moreira; Costa, 2013), renamed as "Single-Period Job Rotation Local Search", receives a solution J' and allows minor modifications to its structure. Specifically, the local search will enable tasks to remain in their current stations or to be assigned to immediately neighboring stations. This method is beneficial for fine-tuning task assignments in a job rotation context, ensuring that minor adjustments can lead to significant improvements in workload balance and reduced worker fatigue. The model, already considering the changes presented at Section 4.2.4 (Equation 4.1) is:

Subject to (2.2)–(2.14):

$$x_{swit} \leq \bar{x}_{swit} + \sum_{w' \in W \setminus W_i} \bar{x}_{s-1,w'it} + \sum_{w' \in W \setminus W_i} \bar{x}_{s+1,w'it}, \qquad \forall s \in S, \forall w \in W \setminus W_i, \forall i \in N, \forall t \in T, \tag{4.2}$$

$$y_{swt} = \bar{y}_{swt}, \qquad \forall s \in S, \forall w \in W, \forall t \in T. \tag{4.3}$$

Constraints (4.2) enforces variables x_{swit} to be set at zero if task i is not assigned to station s nor to any of its neighbors ($s-1$ and $s+1$) in the original solution J'. Note that variables with the indexes $(s-1)$ and $(s+1)$ must be ignored in the constraints if station s is the first or

last station in the line, respectively. Constraints (4.3) ensure that the workers are kept at their current positions.

4.2.5.2 LSJR2: Local Search for Job Rotation 2 from Moreira & Costa (2013)

The second procedure, named local search for job rotation 2 (LSJR2) (Moreira; Costa, 2013), here renamed as "Subperiod-Based Job Rotation Local Search", receives as input a feasible job rotation solution and a period $\hat{t}$. It then reduces the search space by fixing the solution at all periods except $\hat{t}$. This focused search approach effectively addresses specific time-period constraints, allowing for more precise adjustments within a given time frame. This operation is especially beneficial for optimizing task rotations, particularly when particular periods require more attention due to varying workloads or worker availability. The model, already considering the changes presented at Section 4.2.4 (Equation 4.1) is:

Subject to (2.2)–(2.14):

$$x_{swit} = \bar{x}_{swit}, \qquad \forall s \in S, \forall w \in W \backslash W_i, \forall i \in N, \forall t \in T \backslash \{\hat{t}\}, \tag{4.4}$$

$$y_{swt} = \bar{y}_{swt}, \qquad \forall s \in S, \forall w \in W, \forall t \in T. \tag{4.5}$$

In this case, Constraints 4.4 ensure that all tasks from a period different from $\hat{t}$ must remain in their original workstations. Finally, to guarantee the original allocations of workers in the $\bar{T}$ periods, constraints 4.5 are required.

4.2.5.3 Pattern Injection Local Search (PILS)

The Pattern Injection Local Search (PILS), introduced by Arnold *et al.* (2021) for pattern mining and injection, involves exploring high-order local search neighborhoods by utilizing frequent patterns from elite solutions to enhance the local search. The injection strategy identifies instance patterns within the context of the specific optimization problem, injecting these patterns to guide the search process toward promising regions of the solution space. This method is advantageous for complex optimization tasks where recognizing and exploiting patterns can significantly improve solution quality and search efficiency.

Our implementation of PILS begins by creating a working copy of the origin pool, from which patterns will be mined. The pool is then split into two subsets: the top-performing solutions (the elite set, E) and the remainder (the regular set, R). The number of elite solutions is determined by a fixed elite percentage $e_{\%}$, with a lower bound of at least one elite. The goal of this partition is to isolate high-quality structural features found in elite solutions and contrast them with those in more diverse or suboptimal candidates.

From both elite and regular sets, patterns of varying sizes are mined and evaluated based on their relative frequency. Only those patterns with frequency at least $f_{\min}$ are retained, and from these, the top-K most frequent patterns for each size are selected for subsequent injection. This frequency-based filtering ensures that the algorithm focuses on recurring and potentially useful building blocks.

In the injection phase, the algorithm iterates over a shuffled subset of elite solutions. For each solution, a total of M patterns are injected: a fraction ρ comes from elite patterns ($m_e = \lceil \rho \cdot M \rceil$), and the rest ($m_r = M - m_e$) from regular patterns. If the regular pool is exhausted, additional elite patterns are used to fill the quota. Each selected pattern is then applied to the current solution, generating a new candidate.

If the injected solution is feasible and penalty-free, it is evaluated and added to the destination pool. Optionally, a local search may be applied to the newly injected solution to further improve it. If the refined solution is still feasible and of sufficient quality, it is also added to the pool. After each pattern application, the move is undone to preserve the state for subsequent injections.

The algorithm continues this process until the stopping criterion is reached (e.g., based on time or iteration count), incrementing a counter between each iteration. Algorithm 4.6 outlines the overall procedure.

Figure 4.6 – Pattern Injection Local Search (PILS) pseudocode.

Algorithm 5 Pattern Injection Local Search (PILS)

```
Require: Origin pool of solutions, destination pool
Ensure: Destination pool containing improved solutions
while stopping criteria not met do
    Select elite solutions from the origin pool
    Select regular solutions from the origin pool
    Mine patterns from elite and regular solutions
    for each elite solution do
        for each pattern size do
            Select patterns from elites and regulars
            Complete selection with additional elite patterns if needed
            for each selected pattern do
                Inject the pattern into the solution
                if the resulting solution is feasible and penalty-free then
                    Add it to the destination pool
                if local search is applied then
                    Apply local search and evaluate
                    if the improved solution is feasible and penalty-free then
                        Add it to the destination pool
return destination pool
```

Source: Author

In the context of the ALWABP, frequent "patterns" are defined as unordered combinations of tasks that repeatedly occur together on the same workstation across a set of solutions. Consider $|N|$ the number of tasks and $|S|$ the number of stations, we automatically set the pattern size to:

$$p = \left\lceil \frac{|N|}{2 \cdot |S|} \right\rceil$$

We truncate p to an integer value. To mine patterns of size p from a solution s, we scan each workstation's assigned task list T, extract all $\binom{|T|}{p}$ subsets, sort each tuple to ensure uniqueness, and tally their frequencies over the elite pool. Only those patterns whose frequency exceeds a relative threshold $f_{\min}$ are retained for injection.

To inject a pattern into a target solution, we generate, for each candidate station w', a composite `MultipleMovement` consisting of:

a) a removal of each task t in the pattern from its current station $w \neq w'$;

b) an insertion of t into station w'.

Each movement is applied tentatively; if the resulting solution is feasible, it is accepted into the destination pool and optionally refined via a local search, then undone to restore the original solution before proceeding to the next station.

4.3 Development Details and Optimizations

The core execution engine—now dubbed the *Open Algorithm and Heuristic Framework* (OAHF)—was rebuilt from the ground up as a highly configurable platform capable of running a wide variety of heuristic strategies across multiple problem domains. This ambitious redesign carries its own set of challenges as we strive to balance flexibility with efficiency.

Where Moreira & Costa (2013) implemented their solution in C, relying on minimal data structures (plain `structs` and static arrays) to achieve maximum performance at the cost of adaptability, our Python-based framework adopts a modular, object-oriented architecture, preserving the framework's adaptability and delivering good performance. Fundamental abstractions such as `Solution`, `Evaluator`, `Evaluation`, `Pool` and `Metaheuristic` are exposed as base classes; concrete behaviors (e.g., `AlwabpSolution`, `JobRotationSolution`) are introduced through inheritance and composition, making it straightforward to plug in new problem definitions, neighborhood operators, or search schemes.

Configuration is driven by a single parameter file, which specifies all input data and orchestrates the selection of methods, including neighborhood types, neighborhood selection lists (such as sequential, random, or circular), solution pools, and the metaheuristics to run. Each component is assigned a unique identifier, establishing a clear hierarchy and dependency graph that the framework parses at startup.

Because each move in a heuristic—especially in a fully parameterized environment—can trigger expensive constraint checks or evaluations, we implemented extensive memoization to optimize performance. Hash codes for solutions are cached, and each transition (solution A + move $m \Rightarrow$ solution B) is stored in a lookup "graph", enabling us to retrieve results for repeated or reverse moves quickly. This neighborhood map not only accelerates move evaluation but also aids in managing precedence graphs, whether direct or inverted (Moreira; Costa, 2013).

For the ALWABP solution, the hash is computed by first flattening the assignment state into a single, deterministic sequence of integers. We iterate over each station in sorted order, append the station's identifier, then append its assigned tasks (also sorted) followed by the worker

assigned to that station. After processing all stations, we append the graph-orientation flag. This yields a single tuple of integers whose contents uniquely capture the current station–task–worker configuration plus orientation. We then call Python's built-in `hash()` on that tuple, caching the result so that subsequent calls return the precomputed value without recomputing the flattening. In the JRALWABP variant, we treat each period's ALWABP solution as an atomic component and simply form a tuple of those period-solution objects.

Breaking down the complexity we're talking about, regarding Python's built-in `hash()`, for immutable types, like `int`, hash is O(1), since the result is obtained from the number itself. For tuples, the hashing is approximately the sum of the hashes of its elements ($O(m)$, where m is the number of items). Table 4.7 details each step's complexity and clarifies why the caching helped so much in the hashing process.

Let $|S|$ denote the number of stations, and n_i the number of tasks assigned to station i. The total number of assigned tasks is $|N| = \sum_{i=1}^{|S|} n_i$. The variable K refers to the length of the final flattened list used for hashing, which includes each station ID, all sorted tasks, each assigned worker, and the orientation flag. Therefore, $K = 2|S| + |N| + 1$, which simplifies to $O(|S| + |N|)$ for asymptotic analysis.

Figure 4.7 – Complexity breakdown of the `solution_hash` property.

Step	**Complexity**
Sort station keys	$O(\|S\| \log \|S\|)$
Sort tasks per station	$\sum_{i=1}^{\|S\|} O(n_i \log n_i)$
Build flat list	$O(\|S\| + \|N\|)$
Convert to tuple and compute hash	$O(\|S\| + \|N\|)$
Total (first call)	$O\left(\|S\| \log \|S\| + \sum_{i=1}^{\|S\|} n_i \log n_i + \|S\| + \|N\|\right)$
Subsequent calls	$O(1)$ (memoized result)

Source: Author

Finally, to further narrow the gap between Python's expressive power and the raw speed of a compiled language, critical modules are transpiled and compiled with Cython. This hybrid

approach preserves the framework's adaptability while delivering robust, near-C performance where it matters most.

4.4 Final Remarks

In summary, this chapter has detailed the systematic construction of our hybrid optimization framework—from the generation of a diverse solution pool through iterative heuristics and metaheuristic variants to the precise parametrization and interleaving of Tabu Search, GRASP, and BRKGA methods. We have also introduced the modifications to the original LP Selector Model from Moreira & Costa (2013), elevating cycle time to a decision variable and balancing multi-objective criteria to drive both task variety and efficiency. The incorporation of a novel local search to the JRALWABP (PILS), also keeping the previously used ones (LSJR1, LSJR2) and the development of the modular OAHF platform further reinforce the adaptability and rigor of our approach. Together, these components form a cohesive methodology designed to explore the search space comprehensively and transparently, laying a solid foundation for the empirical validations and comparative analyses that follow.

5 RESULTS

In this chapter, we present the outcomes of our study on the Job Rotation Assembly Line Worker Assignment and Balancing Problem (JRALWABP). We begin by describing the benchmark instances and their main characteristics (Section 5.1). Then, we report the results of computational experiments involving two exact mixed-integer programming formulations, focusing on feasibility rates, optimality gaps, and solver performance (Sections 5.2.1 and 5.2.1.1). Given the challenges faced by a commercial solver in finding feasible solutions for most large instances, Section 5.2.2 compares the adapted implementation from (Moreira; Costa, 2013) with our proposed version of the hybrid algorithm tailored for the job rotation context. Section 5.3 shows a Pareto front analysis concerning some instances tested. We highlight the strengths of the heuristic–mathematical framework, especially in handling more complex and demanding scenarios in Section 5.4.

5.1 Benchmark

The mathematical models and the hybrid algorithms were tested on a set of instances proposed by Chaves, Lorena & Cristóbal (2007). These data are adapted from SALBP instances and grouped into four families: Roszieg, Heskia, Tonge, and Wee-Mag, each containing 80 instances. The data generation occurs so that each family contains 10 instances for each combination of three experimental factors: the number of workers, the variability of task execution times, and the number of infeasible task-worker assignment pairs. In each situation, each of these factors assumes a "low" or "high" level.

As described in Chaves, Lorena & Cristóbal (2007), for instances with low variability, task execution times are obtained with a uniform distribution in the range $[1, t_i]$, where t_i is the task execution time of the original SALBP instance. In the case of high variability, the range used is $[1, 3t_i]$. The "low" and "high" levels associated with the number of infeasible task-worker pairs are up to 10% and 20%, respectively, and some "high" instances have a bit more than 20% infeasibles. Table 5.1 lists the main characteristics of each family of instances. *Order Strength* is understood as a metric relative to the density of the precedence graph of instances. Thus, given a precedence network represented by a topologically ordered graph $G = (N, E)$, where $E = \{(i, j); j \in F_i\} \subseteq N^2$, its *Order Strength* ($OS$) is defined as $OS = \frac{2|E|}{|N|(|N|-1)}$. The higher the

value of OS, the denser the precedence network tends to be. Regarding the number of periods, this work analyzed $|T| = 2$, one of the values selected by Moreira & Costa (2013), and of faster execution. For $\overline{C}$ of each instance, the best cycle time reported in the work of Borba & Ritt (2014) is used, one of the best sources for bounds in the literature of the problem [1], adding tolerances of 5%, 10%, 25%, and 50% to relax the upper bound for each instance and observe the impact at the job rotation solutions. Considering 2 models per instance, there are a total of 8 distinct tests for each problem instance.

Table 5.1 – Instance Characteristics.

Family	$\|N\|$	$\|W\|$	*Order Strength*
Roszieg	25	4 (groups of 1 to 4) or 6 (groups of 5 to 8)	71.67
Heskia	28	4 (groups of 1 to 4) or 7 (groups of 5 to 8)	22.49
Tonge	70	10 (groups of 1 to 4) or 17 (groups of 5 to 8)	59.42
Wee-Mag	75	11 (groups of 1 to 4) or 19 (groups of 5 to 8)	22.67

Source: Moreira (2015)

5.2 Computational Experiments

In this section, we present the methodology and results of our computational experiments. These findings will provide essential insights and contribute to a deeper understanding of the research problem, offering a foundation for future studies.

5.2.1 Mathematical Models

We consider two formulations for the JRALWABP. The first one, named Model 1 (M1), has been shown in Section 2.4. Model M1 has $O(|S||N||W||T|)$ variables and $O(|S||N|^2|T|)$ constraints. This study introduces an adaptation of the model introduced by Borba & Ritt (2014) for the JRALWABP. Thus, let $d_{vwt} \in \{0, 1\}$ be a decision variable equal to 1 if worker $v \in W$ must precede worker $w \in W$ in period $t \in T$. In addition, the task-to-worker assignment variable is redefined as $x_{wit} \in \{0, 1\}$ equal to 1 if task i is assigned to worker w in period t. Model 2 (M2) is presented below.

[1] ALWABP Repository *online*: https://github.com/mrpritt/ALWABP.

Model M2: maximize $$z = \sum_{i \in N} \sum_{w \in W \setminus W_i} z_{wi} \tag{5.1}$$

Subject to:

(2.8), (2.13), (2.14), **and**

$$\sum_{w \in W \setminus W_i} x_{wit} = 1 \qquad \forall i \in N, \forall t \in T \tag{5.2}$$

$$d_{uwt} \geq d_{uvt} + d_{vwt} - 1 \qquad \forall \{u, v, w\} \subseteq W^3, |\{u, v, w\}| = 3, \forall t \in T \tag{5.3}$$

$$\sum_{i \in N | w \in W \setminus W_i} p_{wi} x_{wit} \leq C_t \qquad \forall w \in W, \forall t \in T \tag{5.4}$$

$$d_{vwt} \geq x_{vit} + x_{wjt} - 1 \qquad \forall i \in N, \forall j \in F_i, \forall v \in W \setminus W_i, \forall w \in W \setminus W_j, \forall t \in T \tag{5.5}$$

$$d_{vwt} + d_{wvt} \leq 1 \qquad \forall v \in W, \forall w \in W \setminus \{v\}, \forall t \in T \tag{5.6}$$

$$z_{wi} \leq \sum_{t \in T} x_{wit} \qquad \forall i \in N, \forall w \in W \setminus W_i \tag{5.7}$$

$$x_{wit} \in \{0, 1\} \qquad \forall i \in N, \forall w \in W \setminus W_i, \forall t \in T \tag{5.8}$$

$$d_{vwt} \in \{0, 1\} \qquad \forall v \in W, \forall w \in W \setminus \{v\}, \forall t \in T \tag{5.9}$$

Model M2 continues with the objective of maximizing the number of distinct tasks performed by workers over the planning horizon. Constraints (5.2) ensure that each task is allocated to a worker in each period. The precedence relationship between workers in the straight-line layout of the production system is established by constraints (5.3). Like constraints (2.7), inequalities (5.4) define the cycle time variable for each period. Task precedences are guaranteed by constraints (5.5), while constraints (5.6) prohibit bidirectional relationships (symmetry) in the positioning of workers on the assembly line. The coupling of variables z and x is established by constraints (5.7). The domain of the variables is indicated by constraints (5.8)-(5.9). Model M2 has $O(|N||W||T|)$ variables and $O(|W|^2|N|^2|T|)$ constraints.

5.2.1.1 Mathematical Models: Results

The mathematical models were implemented in C#, and all numerical tests were conducted on an Intel Core i7 11th generation processor (i7-1165G7), with 2.80GHz, 16GB of RAM, and Windows 10 (x64) operating system. For solving the integer-linear models, Gurobi 11.01 was used, with a single thread, a time limit of 3,600 seconds, and an optimality gap tolerance of 10^{-3}, since we're dealing mostly with integer values. All tests, complete solutions, and Gurobi execution logs are publicly available in the study's repository.[2]. The LP files are not included in the repository as they exceed the size allowed by the data storage platform. However, such files can be generated with the provided source code.

This study presents an analysis of all instances (considering tolerances of 5%, 10%, 25%, and 50%) in which both models, M1 and M2, found at least one feasible solution. Table 5.2 shows, for each family (column **Family**) and tolerance indicator (column **Perc**): (*i*) the number of instances where both models obtained at least one feasible solution (column **#M1/M2**); (*ii*) the number of instances where model M1 obtained at least one feasible solution and model M2 did not (column **#M1**); (*iii*) the number of instances where M2 obtained at least one feasible solution and M1 did not (column **#M2**); and (*iv*) the average optimality gaps and also the standard deviation, in percentage levels (column $\overline{\boldsymbol{Gap}} \pm \boldsymbol{\sigma(Gap)}$).

Evaluating the metrics reported in columns **#M1** and **#M2**, it is noted that both models were effective for the Heskia family. For Roszieg, with tolerances of 5% and 10%, the models had close results. Model M1 obtained more feasible solutions relative to the Tonge family instances, unlike the Wee-Mag family instances, whose resolution via model M2 was more effective. Thus, we can infer that in this context, for the Heskia and Roszieg families, there is little difference between models M1 and M2, as they are less complex instances than the examples seen in the other families.

The column $\overline{\boldsymbol{Gap}} \pm \boldsymbol{\sigma(Gap)}$ shows that for the Roszieg instance family, model M1 was superior to model M2 in cases where the average cycle time tolerance does not exceed 5% and 10% of the reference value. For the values of 25% and 50%, the results of both formulations are similar. In the Heskia family, the quality of the solutions of the two models coincides in all tested scenarios. Models M1 and M2 did not obtain feasible solutions for some scenarios related to large instances belonging to the Tonge and Wee-Mag families. This occurred in cases

[2] Directory containing the results: https://github.com/caio-de-oliveira-lopes/Job_Rotation_Algorithms

where the tolerances corresponded to 5% and 10% of the reference cycle time. In these cases, it is observed that the allowable cycle time is close to the optimal value obtained in an equivalent instance in the context of ALWABP. Therefore, the difficulty of the task rotation model to deal with such a limitation is understandable, given that finding feasible solutions for such instances in ALWABP is notoriously challenging, as reported in the literature by Borba & Ritt (2014).

Considering the other tolerance values, it is observed that formulations M1 and M2 had similar behaviors. Model M1 was superior to model M2 with 25% tolerance in the Tonge and Wee-Mag families. When the desired cycle time increase reaches almost 50%, it can be stated that the solutions of both models are practically equivalent. Additionally, the table reinforces the idea that the Tonge instance set is the most challenging to solve, as it has a high number of workers, tasks, and OS of the precedence graph.

Figures 5.1, 5.2, 5.3, and 5.4 present the average number of nodes visited by the branch-and-cut algorithm implemented in the Gurobi solver, and the average number of simplex iterations performed for each Model M1 and M2, respectively, when solving the instances in the scenarios above. The x-axis represents the families and the applied tolerances, while the y-axis shows the average number of nodes visited. It is observed that the number of nodes explored to solve model M1 is higher for the Tonge and Wee-Mag instance families (Figure 5.1), whereas for model M2, the number of nodes visited is greater for the Roszieg family.

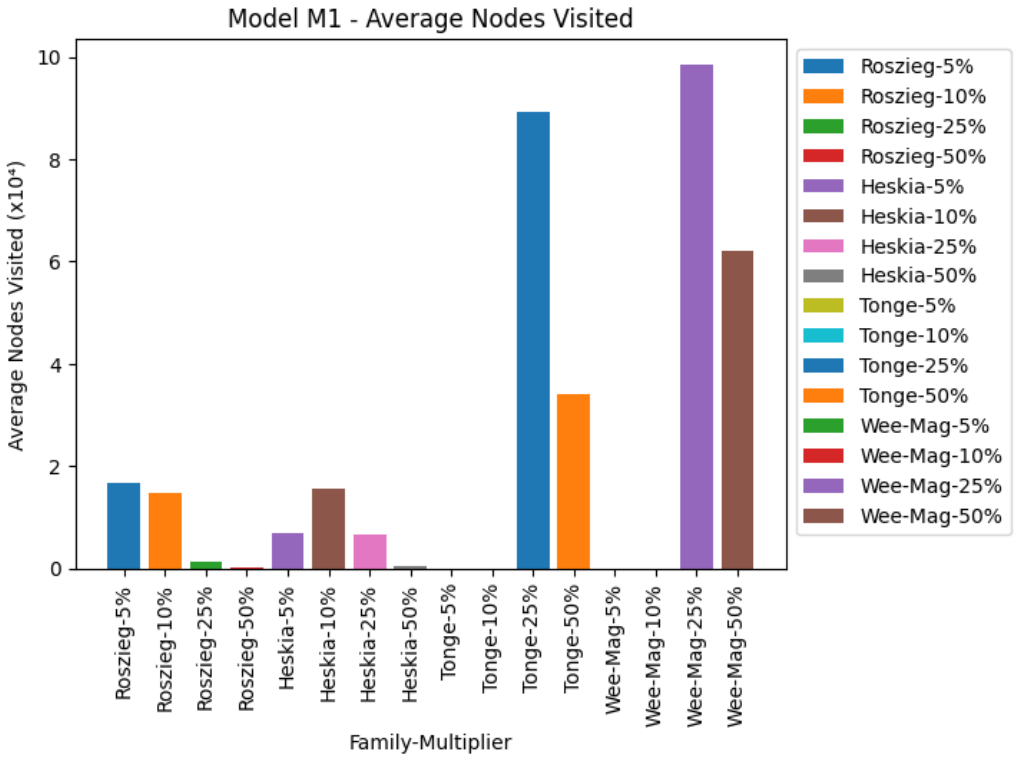
These results suggest that model M1 is more impacted by the number of tasks and workers in the instances, while model M2 is more influenced by the OS value of each instance. This indicates that the set of complementary inequalities related to precedence constraints, proposed in Borba & Ritt (2014), has the potential to reduce the number of nodes explored by model M2.

Table 5.2 – Comparisons between models M1 and M2.

Family	**Perc.**	**#M1/M2**	**#M1**	**#M2**	$\overline{Gap} \pm \sigma Gap$	
					M1	**M2**
Roszieg	5%	79	1	0	0.0% ± 0.0%	5.3% ± 10.5%
	10%	79	1	0	0.0% ± 0.0%	1.9% ± 5.0%
	25%	80	0	0	0.0% ± 0.0%	0.1% ± 0.5%
	50%	80	0	0	0.0% ± 0.0%	0.0% ± 0.0%
Heskia	5%	80	0	0	0.0% ± 0.0%	0.0% ± 0.0%
	10%	80	0	0	0.0% ± 0.2%	0.0% ± 0.0%
	25%	80	0	0	0.0% ± 0.0%	0.0% ± 0.0%
	50%	80	0	0	0.0% ± 0.0%	0.0% ± 0.0%
Tonge	5%	0	0	0	−% ± −%	−% ± −%
	10%	0	2	0	−% ± −%	−% ± −%
	25%	4	9	1	0.2% ± 0.3%	1.1% ± 1.1%
	50%	34	10	2	0.0% ± 0.0%	0.0% ± 0.0%
Wee-Mag	5%	0	0	0	−% ± −%	−% ± −%
	10%	0	0	0	−% ± −%	−% ± −%
	25%	14	0	14	0.0% ± 0.2%	1.3% ± 1.7%
	50%	49	7	12	0.0% ± 0.1%	0.0% ± 0.1%

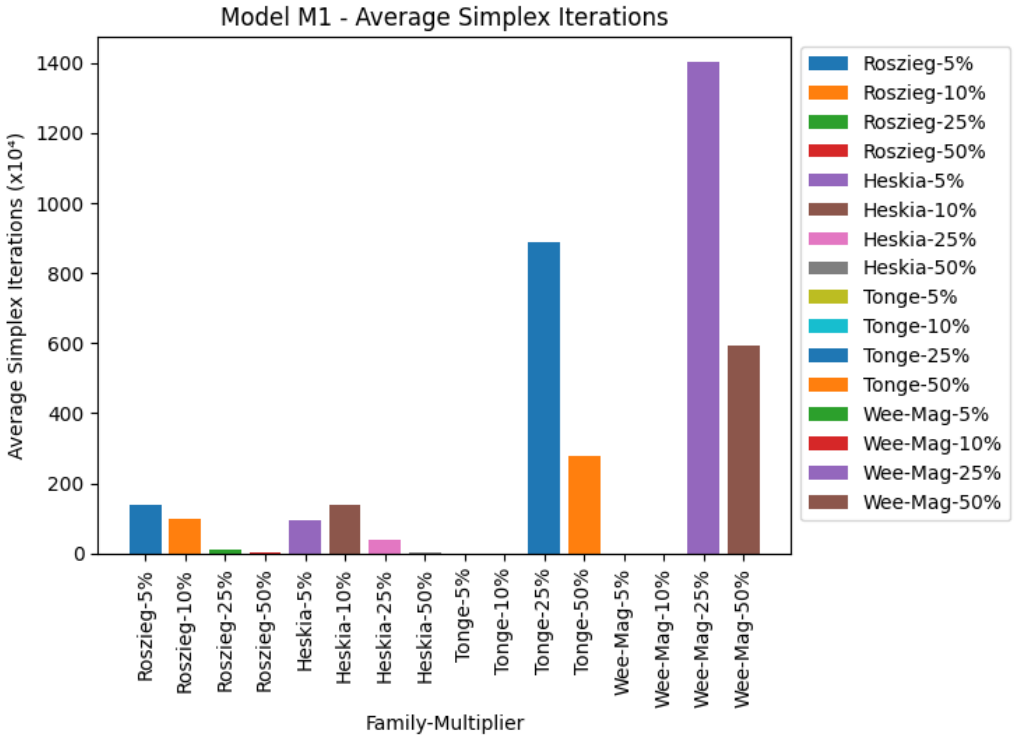
Source: Lopes & Moreira (2024)

Figure 5.1 – Average Nodes Visited for Model M1.



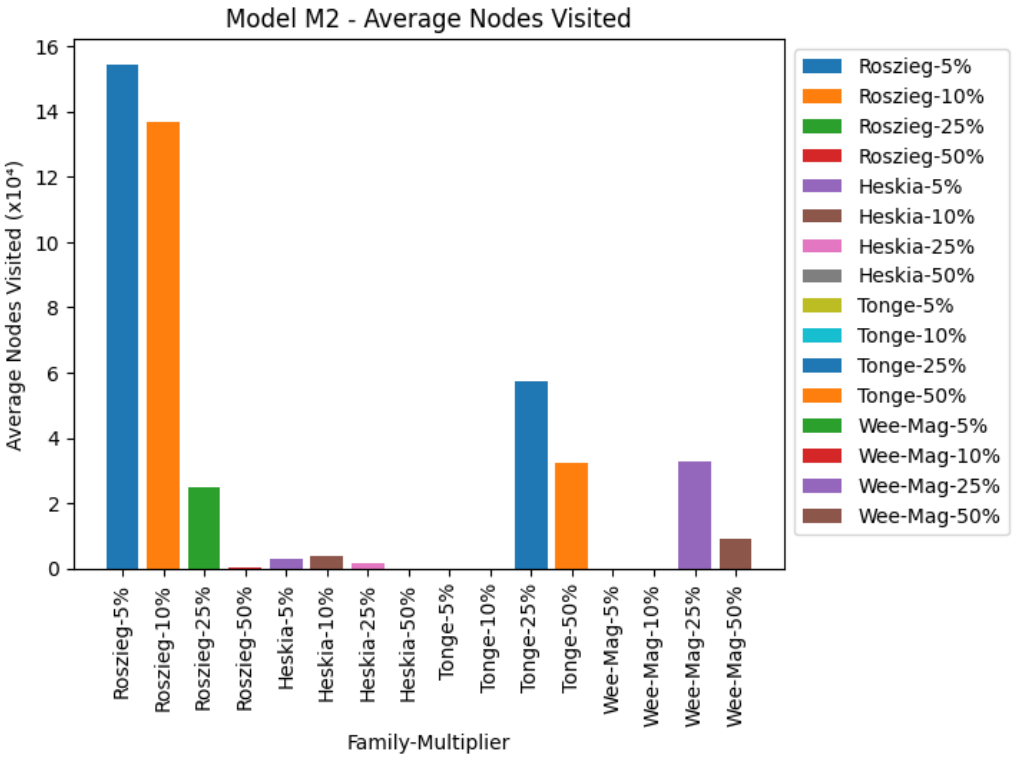
Source: Lopes & Moreira (2024).

Figure 5.2 – Average Simplex Iterations for Model M1.



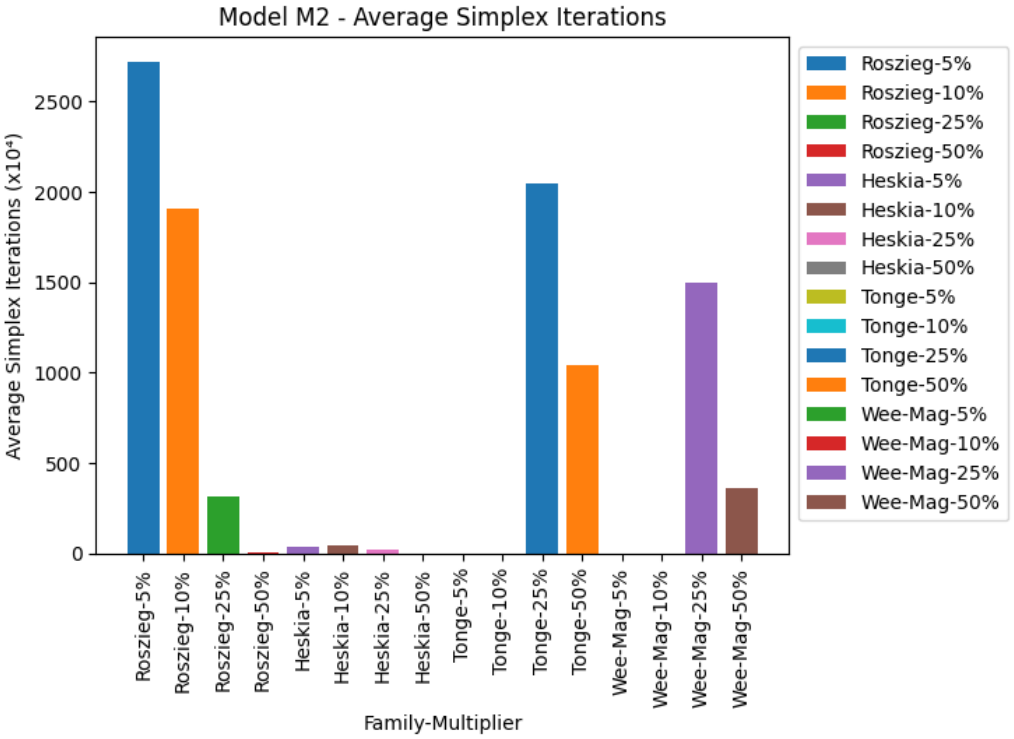
Source: Lopes & Moreira (2024).

Figure 5.3 – Average Nodes Visited for Model M2.



Source: Lopes & Moreira (2024).

Figure 5.4 – Average Simplex Iterations for Model M2.



Source: Lopes & Moreira (2024).

5.2.2 Hybrid Algorithm

As shown in Section 5.2.1.1, purely exact methods still exhibit difficulties when tackling more complex instances. A key reason is that the exact model fails to recognize the independence of each planning period: even a feasible solution for a single period can be replicated across all periods (yielding a valid schedule, far from the optimal). To overcome this limitation, we adopt a hybrid strategy in which linear programming is used both to select high-quality heuristic ALWABP solutions to compose a JRALWABP solution and to guide local searches on a JRALWABP solution through strategic variable fixings.

The hybrid algorithm originally proposed by Moreira & Costa (2013) consists of a single iteration combining: (i) a deterministic constructive method; (ii) a GRASP metaheuristic (i.e., the constructive method enhanced by random move selection); (iii) a tabu search; (iv) solutions from the last generation of a hybrid genetic algorithm (HGA); (v) a LP selector model; and (vi) two local search routines, LSJR1 and LSJR2. Notably, none of the parameters in the 2013 proposal were tuned using Irace or any analogous calibration tool. In contrast, the approach presented in this work—detailed in Section 4.1 and depicted in Figure 4.1—offers a refined alternative, and our computational experiments will directly compare the two proposals.

5.2.2.1 Hybrid Algorithm: Results

The entire structure needed to execute both algorithms was implemented in Python and gives the possibility to compile the entire project in Cython (for speedup purposes), and all numerical tests were conducted on an Intel(R) Xeon(R) CPU E5-2609 v2, with 2.50GHz, and 4GB of RAM, and Debian GNU/Linux 12 (bookworm) Arch: x86_64 operating system. For solving the integer-linear models, Gurobi 12.0.2 was used with the default configurations, except for the optimality gap tolerance, set to 10^{-3}. All tests, complete pools, final solutions, and execution logs are publicly available in the study's repository.[3].

In this section, the original approach proposed by Moreira & Costa (2013) is referred to as **HAJR1**, while the method proposed in this study is denoted as **HAJR2**. Here, both algorithms are using the changes mentioned in Section 4.2.4 using $c_1 = 0.6$ and $c_2 = 0.4$. The parameters for HGA and Tabu Search were tuned through Irace (López-Ibáñez *et al.*, 2016),

[3] Directory containing the results: https://github.com/caio-de-oliveira-lopes/OAHF

using a set of 30 generated ALWABP instances and considering each algorithm individually, only running the Iterative Construction beforehand. The original instances and the generated ones are both present in the repository too.

The training instances were generated using SALBP-1 instances from Otto, Otto & Scholl (2013). We selected 30 benchmark instances—10 small, 10 medium, and 10 large—from the 2013 SALBP dataset [4], and within each size group, we classified instances into three difficulty levels: 3 less tricky, 3 tricky, and 4 very tricky. For the small instances, the number of workers was uniformly varied between 4 and 7. For the medium and large instances, the lower and upper bounds were determined based on the Tonge and Wee-Mag lower bounds: the smaller of the two served as the lower bound, and the larger as the upper bound. Task processing times were scaled by random factors between 3 and 5 to simulate real-world variability. Finally, worker-task incompatibilities were introduced by marking 10%–20% of tasks as infeasible for each worker, ensuring that each worker was unable to perform at least one task while every task remained feasible for at least one worker.

Table 5.3 summarizes the parameter configurations identified by Irace as optimal for both the Hybrid Genetic Algorithm (HGA) and the Tabu Search. For HGA, the stopping criterion is time-based (180s), during which a population of 25 individuals evolves; an elite fraction of 0.1 preserves the top 10 % of solutions across generations, and a bias factor of 0.7 controls the balance between exploration and exploitation in selection. In contrast, the Tabu Search employs an iteration-based stopping criterion (5,000 its) and an aspiration trigger set to 500 iterations, meaning that every 500 iterations the algorithm revisits its intensification or diversification strategy to escape local optima. These settings reflect the differing dynamics of the two metaheuristics: HGA's time-driven stop aligns with its population-based nature, whereas Tabu Search leverages iteration counts and periodic aspiration checks to guide its neighborhood exploration. The other option for tuning was GRASP, but due to having only one important parameter, the size of the candidates list, we've chosen not to do it. Accordingly, the candidate list includes all tasks whose priority ordering rule value is lower or equal to the threshold, defined as:

$$\text{threshold} = c_{\min} + ((1 - greediness) \cdot (c_{\max} - c_{\min}))$$

[4] https://assembly-line-balancing.de/salbp/benchmark-data-sets-2013/

where $c_{\min}$ and $c_{\max}$ are the minimum and maximum values in the list, respectively.

Table 5.3 – Best configurations obtained by *Irace* for HGA and Tabu Search.

HGA				**Tabu Search**	
StopCriteria (s)	**Population**	**Elite**	**Bias**	**StopCriteria (it)**	**Aspiration Trigger (it)**
180	25	0.1	0.7	5000	500

Source: Author

The final parameter definitions were stored in two JSON files in our GitHub repository: heuristic_definition_hajr.json for HAJR1 and heuristic_definition_iterative_hajr.json for HAJR2. Both algorithms stop with a 30-minute execution; the difference lies in HAJR2 using shorter stop criteria (in time) and iterating until the 30 minutes are over. HAJR1, on the other hand, runs only once, so the component algorithms use a higher time limit as stop criteria.

All configuration files—including the experimental seed files, Irace tuning outputs, and additional parameters required for replication—are available at https://github.com/caio-de-oliveira-lopes/OAHF/tree/main/Parameters. Furthermore, the repository's main directory contains the scripts and draft documents used to generate the tables and supplementary materials presented in this paper. Figure 5.5 provides explanations for the column headings used in Tables 5.4 and 5.5, to facilitate their interpretation.

Figure 5.5 – Explanation of each column presented in the experimental results tables.

Column	**Description**
Family	Indicates the instance family.
$\boldsymbol{\lvert N \rvert}$	Number of tasks in the instance.
$\boldsymbol{\lvert W \rvert}$	Number of workers in the instance.
Inf (%)	Percentual of infeasible worker-task relations in the instance.
$\boldsymbol{\overline{NDT} \pm \sigma(NDT)}$	Average number of distinct tasks executed and its standard deviation.
$\boldsymbol{\overline{C} \pm \sigma(C)}$	Average cycle time and its standard deviation.
HAJR1, HAJR2	Refer to the proposed algorithms being evaluated.
Method	The method applied, such as Iterative Construction, GRASP, Tabu Search, HGA, PILS, LSJR1 and LSJR2.

Source: Author

The results summarized in Table 5.4 reveal that HAJR2 consistently attains a slightly higher average number of distinct tasks $\overline{NDT}$ than HAJR1 across most instance families. For

low infeasibility rates (Inf = 10%), the improvement is modest—typically within the respective standard deviations—but systematic (e.g., Roszieg-6 and Heskia-7 see gains of about 2 tasks on average). Cycle times $\overline{C}$ remain essentially equivalent between the two algorithms, with differences rarely exceeding the observed variation (for instance, Tonge-17 shows a 0.2 unit difference in $\overline{C}$, which is negligible). In these tests, we've considered $|T| = 4$, the median value used by Moreira & Costa (2013), from the set $\{2, 4, 7\}$.

When the infeasibility rate increases to Inf = 20%, HAJR2 still tends to produce more distinct tasks, although some specific configurations (e.g., Roszieg-4) exhibit slight decreases (-0.4 tasks). Interestingly, HAJR2 sometimes incurs longer average cycle times (e.g., +6.9 in Tonge-10), while in other cases it reduces $\overline{C}$, as seen in Heskia-4, where it drops by 3.0 units. In the most challenging Wee-Mag instances, HAJR2 achieves an increase of roughly 16 tasks in $\overline{NDT}$ with essentially no penalty in $\overline{C}$, demonstrating its robustness under tight constraints.

Table 5.5 breaks down the average percentage contribution of each heuristic component to the best solution found. These values were computed by analyzing how many of the $|T|$ component solutions (with $|T| = 4$, as previously mentioned) within each JRALWABP solution were generated by each heuristic method. If a method contributed x out of the $|T|$ solutions, it was credited with a contribution of $x/|T|$ for that instance. This process was repeated across 80 instances and 3 random seeds per instance family, totaling 240 cases.

Under HAJR1, the combination "PILS + LSJR1 + LSJR2" overwhelmingly dominates (approximately 78–96%), whereas in HAJR2 this dominance is lessened to 67–92%. Instead, HAJR2 attributes much larger shares to the GRASP phase—up to 30.3% in Roszieg—and to the initial Iterative Construction (e.g., 7.2% in Tonge), indicating that these stages play a substantially more active role in refining solutions. Tabu Search and HGA remain minor contributors in both algorithms (usually below 4%).

Altogether, these findings suggest that HAJR2's rebalancing of heuristic effort—distributing weight more evenly among Iterative Construction, GRASP, and the local search intensification—leads to consistent gains in solution diversity ($\overline{NDT}$) without penalizing, and in many cases even improving, the overall cycle time $\overline{C}$. The larger contributions of the constructive and metaheuristic phases in HAJR2 appear especially beneficial for handling instances with higher infeasibility rates.

Table 5.4 – Summary of results across families of instances.

Family	$\|N\|$	$\|W\|$	**Inf** (%)	$\overline{NDT} \pm \sigma(NDT)$		$\overline{C} \pm \sigma(C)$	
				HAJR1	**HAJR2**	**HAJR1**	**HAJR2**
Roszieg	**25**	**4**	**10**	83.4 ± 6.0	84.6 ± 5.3	32.3 ± 5.0	32.6 ± 4.9
			20	72.7 ± 8.0	74.4 ± 7.5	35.0 ± 13.0	35.4 ± 13.4
		6	**10**	92.8 ± 3.8	94.9 ± 3.6	18.9 ± 3.5	19.0 ± 3.6
			20	89.1 ± 4.9	90.8 ± 5.8	16.1 ± 3.6	16.0 ± 3.1
Heskia	**28**	**4**	**10**	89.7 ± 4.7	88.1 ± 5.6	180.5 ± 50.6	176.8 ± 49.5
			20	83.8 ± 5.5	83.6 ± 5.6	182.5 ± 45.6	179.6 ± 45.9
		7	**10**	95.4 ± 3.7	96.3 ± 5.2	73.5 ± 21.9	74.1 ± 21.3
			20	94.9 ± 6.0	95.2 ± 6.8	67.1 ± 22.5	67.7 ± 22.6
Tonge	**70**	**10**	**10**	252.4 ± 10.9	250.0 ± 13.2	191.5 ± 50.3	198.4 ± 53.0
			20	253.3 ± 8.3	248.5 ± 14.3	173.3 ± 42.1	179.0 ± 43.7
		17	**10**	236.3 ± 0.6	236.7 ± 0.6	89.0 ± 0.2	88.8 ± 0.0
			20	240.8 ± 13.1	244.1 ± 11.7	71.4 ± 20.6	72.6 ± 20.5
Wee-Mag	**75**	**11**	**10**	258.8 ± 7.9	255.4 ± 25.8	57.4 ± 15.8	57.0 ± 18.0
			20	263.1 ± 10.5	266.1 ± 12.8	53.0 ± 13.5	54.5 ± 14.5
		19	**20**	242.4 ± 33.7	258.4 ± 32.5	22.3 ± 4.0	22.3 ± 8.3

Source: Author

Table 5.5 – Average heuristic contributions (%) to best solutions found.

Algorithm	Method	Roszieg	Heskia	Tonge	Wee-Mag
HAJR1	**Iterative Construction**	2.2	2.8	3.0	15.6
	GRASP	5.5	8.0	0.1	0.2
	Tabu Search	1.5	0.6	0.6	5.8
	HGA	0.1	0.0	0.0	0.0
	LSJR1	76.1	40.8	3.0	0.9
	LSJR2	14.6	47.7	93.2	77.4
HAJR2	**Iterative Construction**	1.9	3.8	7.2	2.2
	GRASP	30.3	27.0	14.0	3.1
	Tabu Search	0.8	1.5	3.6	2.5
	HGA	0.2	0.0	0.3	0.0
	PILS	9.5	2.6	0.3	0.1
	LSJR1	50.5	41.2	14.8	3.2
	LSJR2	6.8	24.0	59.8	88.9

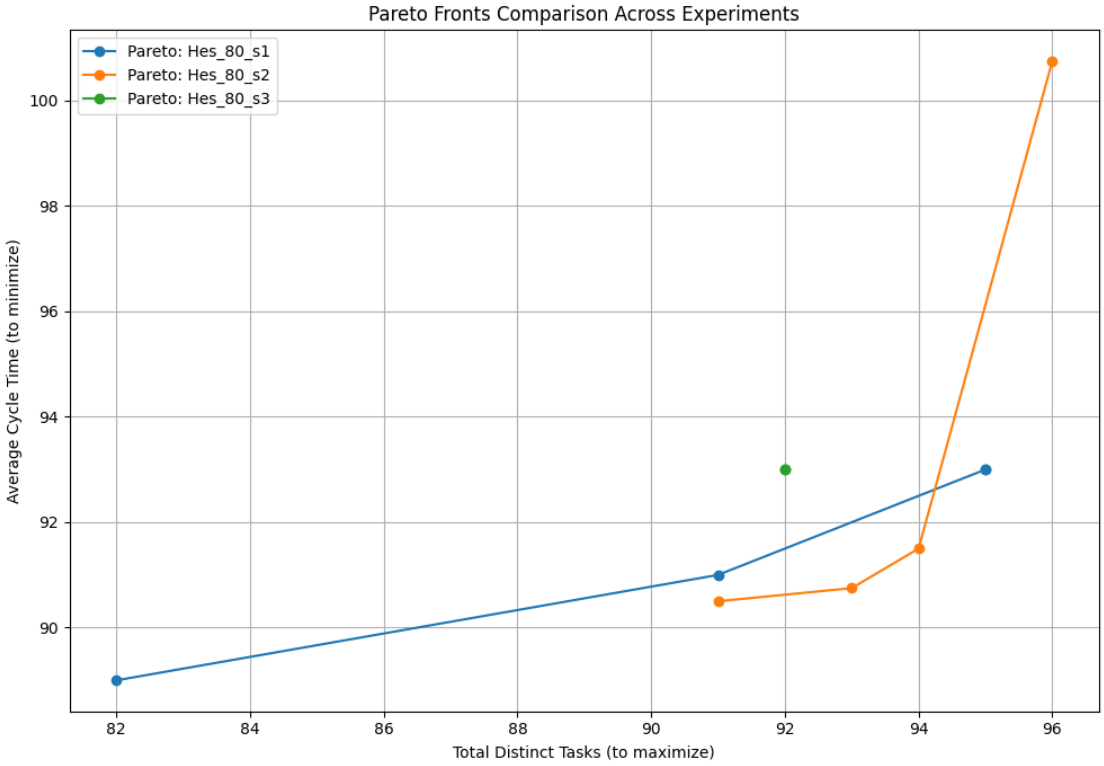
Source: Author

5.3 Pareto Front Analysis

In a bi-objective context, we say that solution A *dominates* solution B if A is at least as good as B in both objectives (i.e. $\overline{NDT}(A) \geq \overline{NDT}(B)$ and $C(A) \leq C(B)$) and strictly better in at least one of them. A solution is *Pareto-optimal* (non-dominated) if no other feasible solution dominates it. The set of all Pareto-optimal solutions forms the *Pareto front*, representing the efficient trade-off between objectives (Ehrgott, 2005; Luc, 2016; Miettinen, 1999).

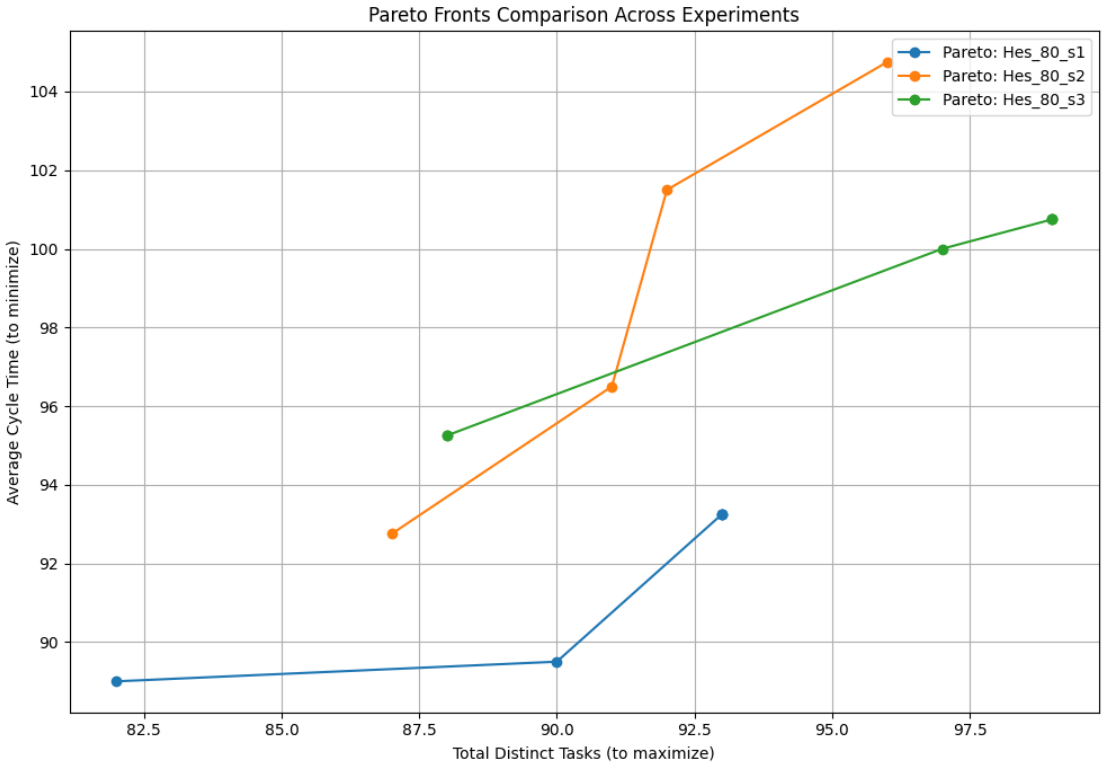
To illustrate the trade-off between maximizing task diversity ($\overline{NDT}$) and minimizing cycle time (C), we selected one small instance (Heskia) and one large instance (Wee-Mag), each solved with both HAJR1 and HAJR2 over three random seeds, showing only non-dominated solutions.

Figure 5.6 – Heskia (small), HAJR1.



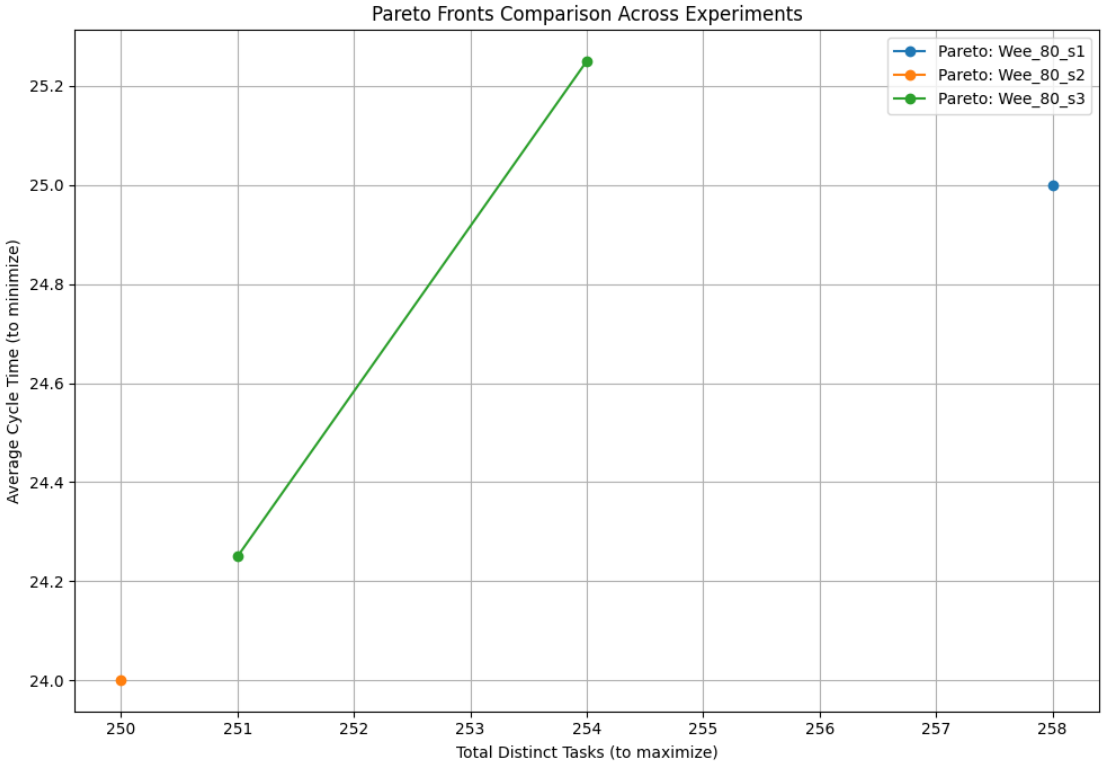
Source: Author

Figure 5.7 – Heskia (small), HAJR2.



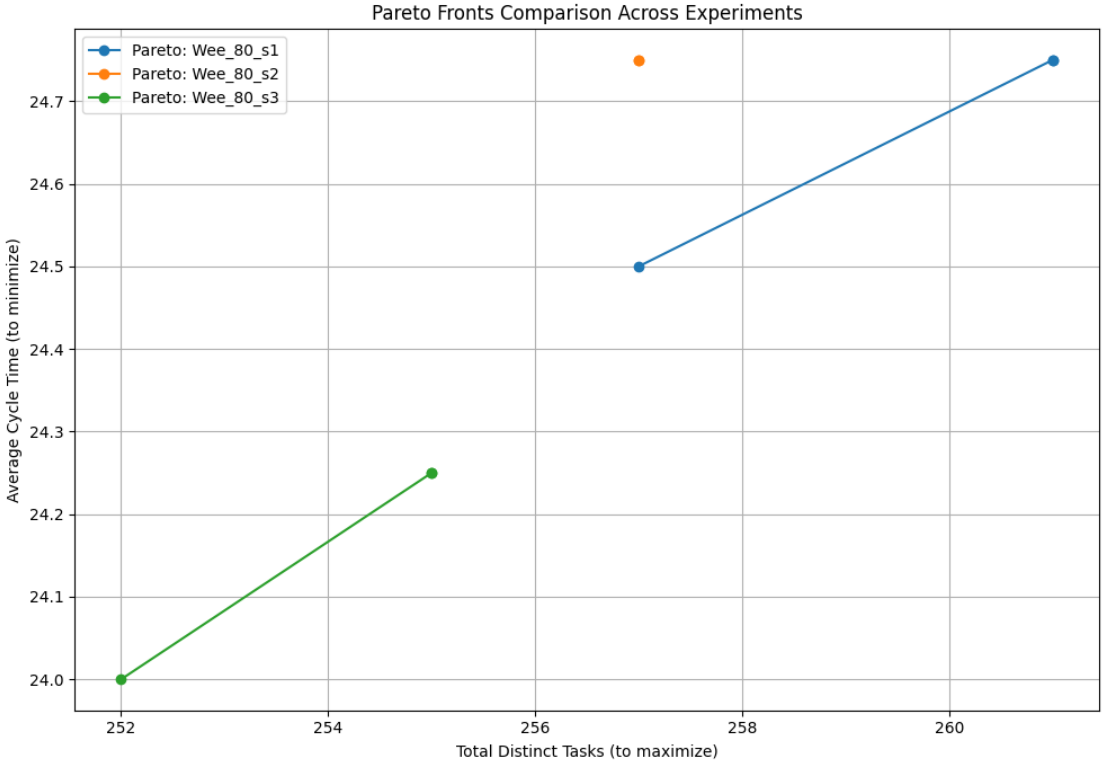
Source: Author

Figure 5.8 – Wee-Mag (large), HAJR1.



Source: Author

Figure 5.9 – Wee-Mag (large), HAJR2.



Source: Author

Figure 5.6–5.7 shows that for the small Heskia instance the frontiers are tight under both algorithms, indicating limited trade-off space. In contrast, Figures 5.8–5.9 display broader frontiers for the large Wee-Mag instance, reflecting a wider range of feasible balances between cycle time and task diversity.

5.4 Final Remarks

This chapter has provided a comprehensive analysis of the computational results obtained from both exact and hybrid approaches to solving the JRALWABP. The experimental evaluation demonstrates the inherent difficulty of the problem, especially for larger and more constrained instances. While the two proposed mixed-integer linear programming formulations (Models M1 and M2) showed complementary strengths, their performance remains limited by scalability and solution times, particularly for complex instance families such as Tonge and Wee-Mag.

Recognizing that a single upper bound on cycle time might render the selector model infeasible when handling a diverse solution pool, the second set of experiments reframed the problem as a bi-objective optimization. This change allowed the model to balance cycle-time minimization against the diversity of feasible task allocations, avoiding the pitfalls of an overly restrictive time limit in the solution pool.

In contrast, the hybrid algorithms, and especially the newly proposed HAJR2, proved more robust in tackling larger instances and those with higher infeasibility rates. The enhancements incorporated into HAJR2, including the refined balance among construction and local search heuristics, resulted in improvements in the number of distinct tasks performed ($\overline{NDT}$) without sacrificing average cycle time ($\overline{C}$). Moreover, the detailed analysis of heuristic contributions indicates that HAJR2's performance gains stem from a more effective use of early-stage constructive methods and GRASP, highlighting the importance of diversity in the solution pool.

Overall, the results validate the efficacy of hybrid methods for addressing JRALWABP, especially in real-world settings where exact solutions may be computationally infeasible. The findings also support the continued exploration and refinement of hybrid frameworks as promising tools for balancing worker rotation, productivity, and operational constraints in assembly line environments.

6 CONCLUSION

This study tackled the Job Rotation Assembly Line Worker Assignment and Balancing Problem (JRALWABP), which combines the challenges of heterogeneous worker capabilities with the need for systematic rotation policies to maintain productivity and fairness. Our primary goal was to examine existing exact and heuristic methods, identify their strengths and limitations, and propose improved algorithms capable of delivering high-quality solutions on realistic instance sets.

It is worth recalling how each chapter contributed to fulfilling the objectives of this study. In Chapter 2 we established the NP-complete nature of the JRALWABP, highlighted the practical challenges in finding feasible schedules, and motivated the use of hybrid optimization methods. Chapter 3 reviewed the main problem variants, key constraint types, and objective functions, as well as the role of worker heterogeneity in shaping solution strategies. Chapter 4 introduced our novel iterative hybrid framework, extending the approach of Moreira & Costa (2013).

Regarding the results, we revisited one exact mixed-integer programming formulation (Model M1) and proposed another (Model M2) to evaluate their feasibility and performance on benchmark instances. We consider the average cycle time overall periods as a capacity constraint. While these models provided valuable lower bounds and occasional optimal solutions for small and medium-sized cases, they proved computationally prohibitive on larger, more constrained instances, highlighting the need for robust heuristic approaches.

Building on the hybrid algorithm already existing in literature, referred to here as HAJR1, we developed an enhanced strategy here referred to as HAJR2. Unlike HAJR1's single-shot execution of constructive heuristics, GRASP, Tabu Search and hybrid genetic components followed by an LP-based selector, HAJR2 interleaves these components in an iterative loop over the entire runtime. We also introduced our version of the Pattern Injection Local Search (PILS) heuristic for ALWABP, which effectively exploits elite solution patterns to both diversify and intensify the search. HGA and Tabu Search parameter settings were fine-tuned via Irace. All algorithms were implemented within a unified Python/Cython framework to ensure a fair comparison.

Extensive computational experiments on four families of ALWABP instances demonstrated that HAJR2 consistently outperforms HAJR1. In particular, HAJR2 achieves a higher average number of distinct tasks per worker without compromising cycle time, and it scales

more gracefully to instances with high infeasibility rates. An analysis of heuristic contributions confirmed that the iterative interleaving and the PILS addition were the keys to HAJR2's performance gains, especially on the most challenging benchmarks.

Since the GRASP metaheuristic demonstrates a potential to provide solutions for the final job rotation schedule, future works may extend this algorithm to explore different initial solutions and local search schemes as a single solution builder of the pool. Irace can still be used to tune GRASP and specially PILS, which has several parameters to be tuned. Aside from all this, we can also develop an instance space analysis to understand better the behavior of HAJR1 and HAJR2 over the benchmark.

Overall, this research refines the state of the art in job rotation scheduling for heterogeneous workforces, offering a practical, extensible algorithmic platform for real-world assembly line planning.

REFERENCES

ABDOUS, M. A. *et al.* Scenario-based optimization and simulation framework for human-centered assembly line balancing. **International Journal of Production Economics**, Elsevier, v. 282, p. 109513, 4 2025. ISSN 0925-5273. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0925527324003700.

ALBAITI, N. N. S.; CHEAITOU, A. Optimizing worker productivity and the exposure to hand-arm vibration: a skill-based job rotation model. **Journal of Industrial and Production Engineering**, Taylor & Francis, v. 41, n. 2, p. 121–139, 2024. Disponível em: https://doi.org/10.1080/21681015.2023.2270985.

ALLWOOD, J.; LEE, W. The impact of job rotation on problem solving skills. **International Journal of Production Research**, v. 42, p. 865–881, 03 2004.

ARNOLD, F. *et al.* PILS: Exploring high-order neighborhoods by pattern mining and injection. **Pattern Recognition**, v. 116, p. 10795, 2021. ISSN 0031-3203. Disponível em: https://www.sciencedirect.com/science/article/pii/S0031320321001448.

ASAWARUNGSAENGKUL, K.; NANTHAVANIJ, S. Heuristic genetic algorithm for workforce scheduling with minimum total worker-location changeover. **International Journal of Industrial Engineering : Theory Applications and Practice**, v. 15, 12 2008.

ASENSIO-CUESTA, S. *et al.* A genetic algorithm for the design of job rotation schedules considering ergonomic and competence criteria. **International Journal of Advanced Manufacturing Technology**, v. 60, p. 1161–1174, 06 2012.

ASSUNÇÃO, A. *et al.* A genetic algorithm approach to design job rotation schedules ensuring homogeneity and diversity of exposure in the automotive industry. **Heliyon**, v. 8, n. 5, p. e09396, 2022. ISSN 2405-8440. Disponível em: https://www.sciencedirect.com/science/article/pii/S2405844022006843.

AYOUGH, A.; FARHADI, F.; ZANDIEH, M. The job rotation scheduling problem considering human cognitive effects: an integrated approach. **Assembly Automation**, ahead-of-print, 05 2021.

AYOUGH, A.; HOSSEINZADEH, M.; MOTAMENI, A. Job rotation scheduling in the seru system: shake enforced invasive weed optimization approach. **Assembly Automation**, ahead-of-print, 02 2020.

AYOUGH, A. *et al.* Modeling workers rotation in divisional seru production systems. **Computers & Industrial Engineering**, Pergamon, v. 205, p. 111141, 7 2025. ISSN 0360-8352. Disponível em: https://www-sciencedirect-com.ez26.periodicos.capes.gov.br/science/article/pii/S0360835225002876?via%3Dihub.

AYOUGH, A.; ZANDIEH, M.; FARHADI, F. Balancing, sequencing, and job rotation scheduling of a u-shaped lean cell with dynamic operator performance. **Computers & Industrial Engineering**, v. 143, p. 106363, 02 2020.

AZIZI, N.; LIANG, M. An integrated approach to worker assignment, workforce flexibility acquisition, and task rotation. **The Journal of the Operational Research Society**, v. 64, 02 2013.

BAAS, J. *et al.* Scopus as a curated, high-quality bibliometric data source for academic research in quantitative science studies. **Quantitative Science Studies**, v. 1, n. 1, p. 377–386, 02 2020. ISSN 2641-3337. Disponível em: https://doi.org/10.1162/qss\%5Fa\%5F00019.

BATTAÏA, O.; DOLGUI, A. Hybridizations in line balancing problems: A comprehensive review on new trends and formulations. **International Journal of Production Economics**, Elsevier B.V., v. 250, 8 2022. ISSN 09255273.

BATTINI, D. *et al.* Towards industry 5.0: A multi-objective job rotation model for an inclusive workforce. **International Journal of Production Economics**, v. 250, n. C, 2022. Disponível em: https://ideas.repec.org/a/eee/proeco/v250y2022ics092552732200202x.html.

BHADURY, J.; RADOVILSKY, Z. Job rotation using the multi-period assignment model. **International Journal of Production Research - INT J PROD RES**, v. 44, p. 4431–4444, 10 2006.

BORBA, L.; RITT, M. A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. **Computers and Operations Research**, v. 45, p. 87–96, 2014.

BOTTI, M. C. L.; MORA, C. Modelling job rotation in manufacturing systems with aged workers. **International Journal of Production Research**, Taylor & Francis, v. 59, n. 8, p. 2522–2536, 2021. Disponível em: https://doi.org/10.1080/00207543.2020.1735659.

BOYSEN, N.; SCHULZE, P.; SCHOLL, A. **Assembly line balancing: What happened in the last fifteen years?** [*S.l.*]: Elsevier B.V., 2022. 797–814 p.

BUTKOVIČ, P.; LEWIS, S. On the job rotation problem. **Discrete Optimization**, v. 4, n. 2, p. 163–174, 2007. ISSN 1572-5286. Disponível em: https://www.sciencedirect.com/science/article/pii/S1572528606000910.

CALZAVARA, M. *et al.* Ageing workforce management in manufacturing systems: state of the art and future research agenda. **International Journal of Production Research**, Taylor and Francis Ltd., v. 58, p. 729–747, 2 2020. ISSN 1366588X.

CHAVES, A.; LORENA, L.; CRISTÓBAL, M. Clustering search approach for the assembly line worker assignment and balancing problem. **37th International Conference on Computers and Industrial Engineering 2007**, v. 2, 01 2007.

CHAVES, A. A. *et al.* A random-key optimizer for combinatorial optimization. **arXiv preprint arXiv:2411.04293**, 2024.

CHIARANDINI, M.; AL. et. A biased random-key genetic algorithm for the resource-constrained project scheduling problem. **Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation**, ACM, p. 175–182, 2010.

COMPER, M. L. C. *et al.* Effectiveness of job rotation for preventing work-related musculoskeletal diseases: a cluster randomised controlled trial. **Occupational and environmental medicine**, BMJ Publishing Group Ltd, v. 74, p. 545–552, 8 2017. ISSN 1470-7926. Disponível em: http://www.ncbi.nlm.nih.gov/pubmed/28250047.

COMPER, M. L. C. *et al.* Influence of adherence to autonomous job rotation on musculoskeletal symptoms, occupational exposure, and work ability. **International Journal of Industrial Ergonomics**, Elsevier, v. 84, p. 103165, 7 2021. ISSN 0169-8141.

CONFORTI, M.; CORNUÉJOLS, G.; ZAMBELLI, G. **Integer programming**. [*S.l.*]: Springer, 2014.

COROMINAS, A.; PASTOR, R.; RODRÍGUEZ, E. Rotational allocation of tasks to multifunctional workers in a service industry. **International Journal of Production Economics**, v. 103, n. 1, p. 3–9, 2006. ISSN 0925-5273. Disponível em: https://www.sciencedirect.com/science/article/pii/S092552730500143X.

COSTA, A. M.; MIRALLES, C. Job rotation in assembly lines employing disabled workers. **International Journal of Production Economics**, v. 120, p. 625–632, 8 2009. ISSN 09255273.

EHRGOTT, M. **Multicriteria Optimization**. 2nd. ed. Berlin: Springer, 2005. ISBN 978-3-540-23098-8.

FEO, T.; RESENDE, M. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, v. 6, p. 109–133, 03 1995.

FORTZ, B.; THORUP, M. Internet traffic engineering by optimizing ospf weights. **IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings (Cat. No.00CH37064)**, IEEE, v. 2, p. 519–528, 2000.

GLOVER, F.; LAGUNA, M. Tabu search. In: ____. **Handbook of Combinatorial Optimization: Volume1–3**. Boston, MA: Springer US, 1998. p. 2093–2229. ISBN 978-1-4613-0303-9. Disponível em: https://doi.org/10.1007/978-1-4613-0303-9\%5F33.

GONÇALVES, J.; RESENDE, M. Biased random-key genetic algorithms for combinatorial optimization. **J. Heuristics**, v. 17, p. 487–525, 10 2011.

HASHEMI-PETROODI, S. E. *et al.* Workforce reconfiguration strategies in manufacturing systems: a state of the art. **International Journal of Production Research**, Taylor and Francis Ltd., v. 59, p. 6721–6744, 2021. ISSN 1366588X.

HOCHDÖRFFER, J.; HEDLER, M.; LANZA, G. Staff scheduling in job rotation environments considering ergonomic aspects and preservation of qualifications. **Journal of Manufacturing Systems**, v. 46, p. 103–114, 2018. ISSN 0278-6125. Disponível em: https://www.sciencedirect.com/science/article/pii/S0278612517301462.

JAIN, A. K.; AL. et. A review on job shop scheduling with a focus on techniques used and recent trends. **International Journal of Production Research**, Taylor & Francis, v. 60, p. 6827–6854, 2022.

JARADAT, G.; AYOB, M.; ALMARASHDEH, I. The effect of elite pool in hybrid population-based meta-heuristics for solving combinatorial optimization problems. **Applied Soft Computing**, Elsevier, v. 44, p. 45–56, 2016.

KATIRAEE, N. *et al.* Consideration of workers' differences in production systems modelling and design: State of the art and directions for future research. **International Journal of Production Research**, Taylor and Francis Ltd., v. 59, p. 3237–3268, 2021. ISSN 1366588X.

LONDE, M. A. *et al.* **Biased Random-Key Genetic Algorithms: A Review**. 2023. Disponível em: https://arxiv.org/abs/2312.00961.

LOPES, C. de O.; MOREIRA, M. C. O. Uma análise experimental de modelos matemáticos para a resolução do planejamento de rotação de tarefas em linhas de produção com trabalhadores heterogêneos. In: **SBPO 2024 – LVI SBPO, Novembro, 04-07, 2024, Fortaleza, CE, Brasil**. [*S.l.*]: Submitted to the Brazilian Symposium on Operational Research (SBPO), 2024. p. 1–12.

LÓPEZ-IBÁÑEZ, M. *et al.* The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, Elsevier, v. 3, p. 43–58, 2016.

LUC, D. T. Pareto optimality. In: LUC, D. T. (Ed.). **Multiobjective Linear Programming: An Introduction**. 1st. ed. Cham, Switzerland: Springer, 2016. cap. 4, p. 85–118. ISBN 978-3-319-21091-9.

MCDONALD KIMBERLY P. ELLIS, E. M. V. A. T.; KOELLING, C. P. Development and application of a worker assignment model to evaluate a lean manufacturing cell. **International Journal of Production Research**, Taylor & Francis, v. 47, n. 9, p. 2427–2447, 2009. Disponível em: https://doi.org/10.1080/00207540701570174.

MICHALOS, G.; MAKRIS, S.; MOURTZIS, D. A web based tool for dynamic job rotation scheduling using multiple criteria. **CIRP Annals**, v. 60, n. 1, p. 453–456, 2011. ISSN 0007-8506. Disponível em: https://www.sciencedirect.com/science/article/pii/S0007850611000382.

MICHALOS, G. *et al.* Dynamic job rotation for workload balancing in human based assembly systems. **CIRP Journal of Manufacturing Science and Technology**, v. 2, n. 3, p. 153–160, 2010. ISSN 1755-5817. Sustainable Development of Manufacturing Systems. Disponível em: https://www.sciencedirect.com/science/article/pii/S1755581710000167.

MIETTINEN, K. Multiobjective optimization. In: BRANKE, J. *et al.* (Ed.). **Interactive Methods for Multiobjective Optimization**. Berlin: Springer, 1999, (Lecture Notes in Computer Science, v. 5252). p. 27–46.

MIRANDA, E. Álvarez; PEREIRA, J. On the complexity of assembly line balancing problems. **Computers & Operations Research**, v. 108, p. 182–186, 2019. ISSN 0305-0548. Disponível em: https://www.sciencedirect.com/science/article/pii/S0305054819300826.

MIRANDA, E. Álvarez; PEREIRA, J.; VILà, M. Analysis of the simple assembly line balancing problem complexity. **Computers & Operations Research**, v. 159, p. 106323, 2023. ISSN 0305-0548. Disponível em: https://www.sciencedirect.com/science/article/pii/S0305054823001879.

MOREIRA, M.; COSTA, A. A minimalist yet efficient tabu search algorithm for balancing assembly lines with disabled workers. In: . [*S.l.: s.n.*], 2009.

MOREIRA, M. C. d. O. **Problema de balanceamento de linhas de produção e integração de trabalhadores**. Tese (Doutorado) — Universidade de Sao Paulo, Agência USP de Gestão da Informação Acadêmica (AGUIA), 2015. Disponível em: http://dx.doi.org/10.11606/T.55.2016.tde-08012016-145627.

MOREIRA, M. C. O.; COSTA, A. M. Hybrid heuristics for planning job rotation schedules in assembly lines with heterogeneous workers. **International Journal of Production Economics**, v. 141, p. 552–560, 2013.

MOREIRA, M. C. O. *et al.* Simple heuristics for the assembly line worker assignment and balancing problem. **Journal of Heuristics**, v. 18, p. 505–524, 2012.

MOSSA, G. *et al.* Productivity and ergonomic risk in human based production systems: A job-rotation scheduling model. **International Journal of Production Economics**, v. 171, p. 471–477, 2016. ISSN 0925-5273. Challenges for Sustainable Operations – Selected papers of ICPR 2013. Disponível em: https://www.sciencedirect.com/science/article/pii/S092552731500225X.

MURA, M.; DINI, G. Job rotation and human–robot collaboration for enhancing ergonomics in assembly lines by a genetic algorithm. **The International Journal of Advanced Manufacturing Technology**, v. 118, 10 2021.

MURA, M. D.; DINI, G. Improving ergonomics in mixed-model assembly lines balancing noise exposure and energy expenditure. **CIRP Journal of Manufacturing Science and Technology**, v. 40, p. 44–52, 2023. ISSN 1755-5817. Disponível em: https://www.sciencedirect.com/science/article/pii/S1755581722001730.

MUTHUSWAMY, K.; AL. et. A hybrid brkga and lns algorithm for the vehicle routing problem with time windows. **Computers & Industrial Engineering**, Elsevier, v. 165, p. 107939, 2022.

NANTHAVANIJ, S.; YAOYUENYONG, S.; JEENANUNTA, C. Heuristic approach to workforce scheduling with combined safety and productivity objective. **International Journal of Industrial Engineering: Theory, Applications and Practice**, v. 17, n. 4, Sep. 2010. Disponível em: https://journals.sfu.ca/ijietap/index.php/ijie/article/view/370.

OSPINA-MATEUS, H. *et al.* Application of genetic algorithm to job scheduling under ergonomic constraints in manufacturing industry. **Journal of Ambient Intelligence and Humanized Computing**, v. 10, p. 2063–2090, 05 2019.

OTTO, A.; BATTAÏA, O. Reducing physical ergonomic risks at assembly lines by line balancing and job rotation: A survey. **Computers & Industrial Engineering**, v. 111, p. 467–480, 2017. ISSN 0360-8352. Disponível em: https://www.sciencedirect.com/science/article/pii/S0360835217301572.

OTTO, A.; OTTO, C.; SCHOLL, A. Systematic data generation and test design for solution algorithms on the example of salbpgen for assembly line balancing. **European Journal of Operational Research**, v. 228, n. 1, p. 33–45, 2013. ISSN 0377-2217. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221713000039.

OTTO, A.; SCHOLL, A. Reducing ergonomic risks by job rotation scheduling. **OR Spectrum**, v. 35, 07 2013.

PADULA, R. S. *et al.* Job rotation designed to prevent musculoskeletal disorders and control risk in manufacturing industries: A systematic review. **Applied Ergonomics**, Elsevier, v. 58, p. 386–397, 1 2017. ISSN 0003-6870.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. [*S.l.*]: Springer Nature, 2022.

SCHOLL, A.; VOSS, S. Simple assembly line balancing–heuristic approaches. **Journal of Heuristics**, v. 2, p. 217–244, 12 1997.

SEÇKINER, S. U.; KURT, M. A simulated annealing approach to the solution of job rotation scheduling problems. **Applied Mathematics and Computation**, Elsevier, v. 188, p. 31–45, 5 2007. ISSN 0096-3003.

SEÇKINER, S. U.; KURT, M. Ant colony optimization for the job rotation scheduling problem. **Applied Mathematics and Computation**, Elsevier, v. 201, p. 149–160, 7 2008. ISSN 0096-3003.

SHOJAEI DAVOOD JAFARIAND, M. K. A.; DOKOHAKI, P. Developing and solving the multi-objective flexible and sustainable job shop scheduling problem with reverse flow and job rotation considerations in uncertain situations. **Journal of Optimization in Industrial Engineering**, Islamic Azad University, Qazvin Branch, v. 16, n. 35, 2023. ISSN 2251-9904. Disponível em: sanad.iau.ir/fa/Article/951334.

SOUZA, M. *et al.* Acviz: A tool for the visual analysis of the configuration of algorithms with irace. **Operations Research Perspectives**, Elsevier, v. 8, p. 100186, 2021.

THARMMAPHORNPHILAS, W.; NORMAN, B. A methodology to create robust job rotation schedules. **Annals OR**, v. 155, p. 339–360, 08 2007.

TOTH, P.; VIGO, D. **Vehicle Routing: Problems, Methods, and Applications**. [*S.l.*]: SIAM, 2014.

WOLSEY, L. A.; NEMHAUSER, G. L. **Integer and combinatorial optimization**. [*S.l.*]: John Wiley & Sons, 1998.

YAOYUENYONG, S.; NANTHAVANIJ, S. Hybrid procedure to determine optimal workforce without noise hazard exposure. **Computers & Industrial Engineering**, v. 51, n. 4, p. 743–764, 2006. ISSN 0360-8352. Disponível em: https://www.sciencedirect.com/science/article/pii/S0360835206001446.

ZADEH, N.; MOVAHEDI, M.; SHAYANNIA, S. Human resource scheduling in project management using the simulated annealing algorithm with the human factors engineering approach. **Discrete Dynamics in Nature and Society**, v. 2022, p. 1–7, 06 2022.

APPENDIX

APPENDIX A – LESS FREQUENT CONSTRAINTS ON STUDIES

Figure 1 – Less frequent constraints on studies.

Constraints	**References**
Minimum Productivity Limit	(McDonald Kimberly P. Ellis; Koelling, 2009; Mossa *et al.*, 2016)
Energy Limitation Constraint	(Assunção *et al.*, 2022; Battini *et al.*, 2022; Mura; Dini, 2021; Mura; Dini, 2023)
Limitation of Tasks Allocated Successively	(Bhadury; Radovilsky, 2006; Ospina-Mateus *et al.*, 2019; Ayough; Zandieh; Farhadi, 2020; Mura; Dini, 2021; Mura; Dini, 2023; Ayough *et al.*, 2025, 2025)
Task Similarity	(Assunção *et al.*, 2022; Battini *et al.*, 2022; Mura; Dini, 2021; Zadeh; Movahedi; Shayannia, 2022)
Maximum Tasks/Worker Limit	(Ayough; Hosseinzadeh; Motameni, 2020)
Multi-functionality and Skills Constraint	(Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; McDonald Kimberly P. Ellis; Koelling, 2009; Azizi; Liang, 2013; Ayough *et al.*, 2025)
Training Constraint	(Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; McDonald Kimberly P. Ellis; Koelling, 2009; Azizi; Liang, 2013)
Hazard Exposure Constraint	(Corominas; Pastor; Rodríguez, 2006; Asawarungsaengkul; Nanthavanij, 2008; Nanthavanij; Yaoyuenyong; Jeenanunta, 2010)
Job Severity Constraint	(Corominas; Pastor; Rodríguez, 2006; Tharmmaphornphilas; Norman, 2007)
Flow and Sequencing Constraint	(Allwood; Lee, 2004; Ayough *et al.*, 2025)

Source: Author

APPENDIX B – LESS FREQUENT OBJECTIVE FUNCTIONS ON STUDIES

Figure 2 – Less frequent objective functions on studies.

Objective Function	**References**
Minimize task completion time	(Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; Ayough; Hosseinzadeh; Motameni, 2020; Shojaei Davood Jafariand; Dokohaki, 2023)
Minimize workforce allocation	(Yaoyuenyong; Nanthavanij, 2006; Asawarungsaengkul; Nanthavanij, 2008; Ayough; Hosseinzadeh; Motameni, 2020; Ayough; Farhadi; Zandieh, 2021; Shojaei Davood Jafariand; Dokohaki, 2023)
Minimize environmental impact	(Shojaei Davood Jafariand; Dokohaki, 2023)
Balance workload distribution	(Michalos *et al.*, 2010; Michalos; Makris; Mourtzis, 2011; Assunção *et al.*, 2022; Mura; Dini, 2021; Mura; Dini, 2023; Ayough *et al.*, 2025)
Minimize repetitive tasks and fatigue	(Bhadury; Radovilsky, 2006; Costa; Miralles, 2009; Michalos *et al.*, 2010; Michalos; Makris; Mourtzis, 2011; Otto; Scholl, 2013; Assunção *et al.*, 2022; Battini *et al.*, 2022; Zadeh; Movahedi; Shayannia, 2022)
Minimize workflow inefficiencies	(Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021)
Minimize and balance travel distances	(Michalos *et al.*, 2010; Michalos; Makris; Mourtzis, 2011)
Minimize job severity	(Tharmmaphornphilas; Norman, 2007; Asawarungsaengkul; Nanthavanij, 2008)
Minimize penalties over unmet constraints	(Corominas; Pastor; Rodríguez, 2006)

Source: Author

APPENDIX C – LESS FREQUENT HETEROGENEITY FACTORS ON STUDIES

Figure 3 – Less frequent heterogeneity factors on studies.

Heterogeneity	**References**
Gender	(Mura; Dini, 2021; Mura; Dini, 2023)
Age	(Mura; Dini, 2021; Mura; Dini, 2023)
Height	(Mura; Dini, 2021; Mura; Dini, 2023)
Weight	(Mura; Dini, 2021; Mura; Dini, 2023)
Task Execution Cost	(Bhadury; Radovilsky, 2006; Butkovič; Lewis, 2007; Seçkiner; Kurt, 2007; Seçkiner; Kurt, 2008; McDonald Kimberly P. Ellis; Koelling, 2009; Zadeh; Movahedi; Shayannia, 2022)
Physical Limitations	(Asensio-Cuesta *et al.*, 2012; Battini *et al.*, 2022)
Rest Time	(Botti; Mora, 2021; Battini *et al.*, 2022)
Energy Limit	(Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021; Battini *et al.*, 2022)
Forgetfulness Level	(Azizi; Liang, 2013; Ayough; Zandieh; Farhadi, 2020; Ayough; Farhadi; Zandieh, 2021)
Task Number Limit	(Ayough; Hosseinzadeh; Motameni, 2020)
Task Demand	(Tharmmaphornphilas; Norman, 2007)

Source: Author

APPENDIX D – LESS FREQUENT SOLUTION METHODS ON STUDIES

Figure 4 – Less frequent solution methods on studies.

Solution Method	**References**
GAMS	(Shojaei Davood Jafariand; Dokohaki, 2023; AlBaiti; Cheaitou, 2024; Ayough *et al.*, 2025)
ϵ-constraint method	(Botti; Mora, 2021; Battini *et al.*, 2022; Shojaei Davood Jafariand; Dokohaki, 2023; AlBaiti; Cheaitou, 2024)
The whale optimization algorithm (WOA)	(Shojaei Davood Jafariand; Dokohaki, 2023)
Simulated annealing	(Seçkiner; Kurt, 2007; Zadeh; Movahedi; Shayannia, 2022)
Shake-enforced invasive weed optimization (SE-IWO)	(Ayough; Hosseinzadeh; Motameni, 2020)
Tabu search	(Otto; Scholl, 2013)
Random-choice enumerative method	(Michalos *et al.*, 2010; Michalos; Makris; Mourtzis, 2011)
Simulation model	(Allwood; Lee, 2004; McDonald Kimberly P. Ellis; Koelling, 2009; Abdous *et al.*, 2025)
Ant colony	(Seçkiner; Kurt, 2008)
Exact polynomial algorithm for special cases	(Butkovič; Lewis, 2007)
Branch & Bound procedures	(Yaoyuenyong; Nanthavanij, 2006)
Hybrid algorithm	(Bhadury; Radovilsky, 2006)
Assignment algorithm	(Corominas; Pastor; Rodríguez, 2006)
Iterative Dichotomic Search algorithm	(Abdous *et al.*, 2025)
Invasive Weed Optimization (IWO)	(Ayough *et al.*, 2025)

Source: Author

APPENDIX E – TASK PRIORITY RULES BY MOREIRA ET AL. (2012)

These rules derive from six priority rules suggested by Scholl & Voss (1997); the proposed adaptations lead to eleven rules for the ALWABP derived from existing rules for the SALBP. Furthermore, Moreira *et al.* (2012) also proposes five new rules that try to reflect the structure of the ALWABP by using the original task execution times per worker. The idea is to prioritize tasks when workers that quickly execute these tasks are being considered. The full set of priority rules is listed and described in the following, where $\bar{w}_i \in \arg\min_{w \in W} t_{wi}$ is some fastest worker in the execution of task i and $t_i^- = \min_{w \in W} t_{wi}$, $t_i^+ = \max_{w \in W} t_{wi}$, $\bar{t}_i = \sum_{w \in W} t_{wi}/|W|$ are the minimum, maximum, and mean execution time of task i, respectively. Unfeasible task-worker pairs enter with an execution time $\bar{c}$ into t_i^+ and $\bar{t}_i$, where $\bar{c}$ is the cycle time under consideration.

Figure 5 – Task priority rules used in ALWABP.

Rule	**Description**
MaxF	Descending number of followers, $\lvert F_i^* \rvert$
MaxIF	Descending number of immediate followers, $\lvert F_i \rvert$
MaxTime$^-$	Descending minimum task times, t_i^-
MaxTime$^+$	Descending maximum task times, t_i^+
Max$\overline{\text{Time}}$	Descending average task times, $\bar{t}_i$
MinTime$^-$	Ascending minimum task times, t_i^-
MinTime$^+$	Ascending maximum task times, t_i^+
Min$\overline{\text{Time}}$	Ascending average task times, $\bar{t}_i$
MaxPW$^-$	Descending minimum positional weights, $pw_i^- = t_i^- + \sum_{h \in F_i^*} t_h^-$
MaxPW$^+$	Descending maximum positional weights, $pw_i^+ = t_i^+ + \sum_{h \in F_i^*} t_h^+$
Max$\overline{\text{PW}}$	Descending average positional weights, $pw_i = \bar{t}_i + \sum_{h \in F_i^*} \bar{t}_h$
MinD	Ascending difference to best worker, $t_{wi} - t_{\bar{w}_i i}$
MinR	Ascending ratio to best worker, $t_{wi}/t_{\bar{w}_i i}$
MaxFTime	Descending number of followers per time, $\lvert F_i \rvert / t_{wi}$
MaxIFTime	Descending number of immediate followers per time, $\lvert F_i^* \rvert / t_{wi}$
MinRank	Ascending rank of worker's execution time, $\lvert \{w' \in W \mid t_{w'i} < t_{wi}\} \rvert$

Source: Author

Rules MinD and MinR prioritize tasks for which the current worker is faster than other workers, while MaxFTime and MaxIFTime balance two desired characteristics of a task: it should be executed quickly by the current worker and it should make the largest number of tasks available. Rule MinRank gives preference to tasks with a short execution time compared to other workers, but does not depend on concrete execution times. The other rules follow the original rationale. Thus, by combining the sixteen task priority rules with the **FORWARD** and **BACKWARD** precedence graphs, we obtain a total of thirty-two task priority rules used during Iterative Construction and GRASP.