Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas

A mathematical formulation and heuristic algorithms for minimizing the makespan and energy cost under time-of-use electricity price in an unrelated parallel machine scheduling problem

Marcelo Ferreira Rego

Ouro Preto, Brazil
February 2022

Marcelo Ferreira Rego

A mathematical formulation and heuristic algorithms for minimizing the makespan and energy cost under time-of-use electricity price in an unrelated parallel machine scheduling problem

Advisor: Ph.D. Marcone Jamilson Freitas Souza

Co-advisor: Ph.D. Luciano Perdigão Cota

Thesis presented to the *Universidade Federal de Ouro Preto* in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Ouro Preto, Brazil

February 2022

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

R343m Rego, Marcelo Ferreira.
A mathematical formulation and heuristic algorithms for minimizing the makespan and energy cost under time-of-use electricity price in an unrelated parallel machine scheduling problem. [manuscrito] / Marcelo Ferreira Rego. - 2022.
72 f.

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza.
Coorientador: Dr. Luciano Perdigão Cota.
Tese (Doutorado). Universidade Federal de Ouro Preto. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.
Área de Concentração: Ciência da Computação.

1. Unrelated parallel machine. 2. Total energy cost. 3. Makespan. 4. Mixed-Integer Linear Programming. 5. MOVNS. 6. NSGA-II. 7. Multi-objective optimization. I. Cota, Luciano Perdigão. II. Souza, Marcone Jamilson Freitas. III. Universidade Federal de Ouro Preto. IV. Título.

CDU 004

Bibliotecário(a) Responsável: Luciana De Oliveira - SIAPE: 1.937.800

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E BIOLOGICAS
DEPARTAMENTO DE COMPUTACAO
PROGRAMA DE POS-GRADUACAO EM CIENCIA DA COMPUTACAO

FOLHA DE APROVAÇÃO

Marcelo Ferreira Rego

A mathematical formulation and heuristic algorithms for minimizing the makespan and energy cost under time-of-use electricity price in an unrelated parallel machine scheduling problem

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Doutor em Ciência da Computação

Aprovada em 18 de fevereiro de 2022

Membros da banca

Prof. Dr. Marcone Jamilson Freitas Souza - Orientador - Universidade Federal de Ouro Preto
Prof. Dr. - Luciano Perdigão Cota - Co-Orientador - Instituto Tecnológico Vale
Prof. Dr. Puca Huachi Vaz Penna - Universidade Federal de Ouro Preto
Prof. Dr. Igor Machado Coelho - Universidade Federal Fluminense
Prof. Dr. José Elias Cláudio Arroyo - Universidade Federal de Viçosa
Prof. Dr. Lucas de Souza Batista - Universidade Federal de Minas Gerais

Marcone Jamilson Freitas Souza, orientador do trabalho, aprovou a versão final e autorizou seu depósito no Repositório Institucional da UFOP em 17/04/2022

Documento assinado eletronicamente por **Puca Huachi Vaz Penna**, **COORDENADOR(A) DE CURSO DE PÓS-GRADUACÃO EM CIÊNCIA DA COMPUTAÇÃO**, em 17/05/2022, às 19:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0327565** e o código CRC **44637EE9**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: (31)3559-1641 - www.ufop.br

Dedico este trabalho aos meus pais Mario Borges *(in memoriam)* e Maria Deuzilene, à minha esposa Carol e à minha filha Alice.

Resumo

Em muitos países, o preço da energia varia de acordo com a política *time-of-use*. Como regra geral, é vantajoso financeiramente para as indústrias planejarem sua produção considerando essa política. Esta tese apresenta um novo problema de sequenciamento de máquinas paralelas não-relacionadas bi-objetivo com tempos de preparação dependentes da sequência, no qual os objetivos são minimizar o *makespan* e o custo total de energia considerando máquinas com diferentes modos de operação e que o preço da eletricidade segue a política *time-of-use*. Introduzimos uma formulação de programação linear inteira mista e aplicamos o método da soma ponderada para obter uma fronteira Pareto. Também desenvolvemos métodos de otimização multiobjetivo, baseados no *Multi-objective Variable Neighborhood Search* com procedimento de intensificação (chamado MOVNS2) e o *Non-dominated Sorting Genetic Algorithm* II (NSGA-II), para tratar instâncias grandes, com pelo menos 50 tarefas, uma vez que a formulação não pode resolvê-las em um tempo computacional aceitável para a tomada de decisão. Comparamos o desempenho dos algoritmos NSGA-II e MOVNS2 com dois algoritmos de otimização multiobjetivo da literatura, o MOVNS1 e o NSGA-I, em relação às métricas de hipervolume e *hierarchical cluster counting* (HCC). Os resultados mostraram que os métodos propostos são capazes de encontrar uma boa aproximação para a fronteira Pareto comparado com os resultados do método de soma ponderada em instâncias pequenas, de até 10 tarefas. Quando consideramos apenas as instâncias grandes, o MOVNS2 é superior ao MOVNS1, o NSGA-I e o NSGA-II em relação à métrica de hipervolume. Além disso, o NSGA-II supera os métodos de otimização multiobjetivo NSGA-I, MOVNS1 e MOVNS2 em relação à métrica HCC. Ambos os resultados apresentam um nível de confiança de 95%. Assim, o MOVNS2 proposto é capaz de encontrar soluções não-dominadas com boa convergência e o NSGA-II com boa diversidade.

Palavras-chave: Máquinas paralelas não-relacionadas, custo total de energia, *makespan*, Programação linear inteira mista, MOVNS, NSGA-II, Otimização multiobjetivo.

Abstract

In many countries, energy pricing varies according to the time-of-use policy. As a general rule, it is financially advantageous for the industries to plan their production considering this policy. This thesis introduces a new bi-objective unrelated parallel machine scheduling problem with sequence-dependent setup times, in which the objectives are to minimize the makespan and the total energy cost under machines with different operating modes and the time-of-use electricity price policy. We introduced a mixed-integer linear programming formulation and applied the weighted sum method to obtain the Pareto front. We also developed multi-objective methods, based on the Multi-objective Variable Neighborhood Search with intensification procedure (named MOVNS2) and Non-dominated Sorting Genetic Algorithm II (NSGA-II), to address large instances with at least 50 jobs since the formulation cannot solve it in acceptable computational time for decision-making. We compared the performance of the NSGA-II and MOVNS2 algorithms with two multi-objective algorithms of the literature, MOVNS1, and NSGA-I, concerning the hypervolume and hierarchical cluster counting (HCC) metrics. The results showed that the proposed methods are able to find a good approximation for the Pareto front compared with the presented results by the weighted sum method in small instances with up to 10 jobs. Considering only large instances, MOVNS2 is superior to MOVNS1, NSGA-I, and NSGA-II in the hypervolume metric. In addition, NSGA-II outperforms the NSGA-I, MOVNS1, and MOVNS2 multi-objective techniques concerning the HCC metric. Both results are with a 95% confidence level. Thus, the proposed MOVNS2 finds non-dominated solutions with good convergence and NSGA-II with good diversity.

Keywords: Unrelated parallel machine, total energy cost, makespan, Mixed-Integer Linear Programming, MOVNS, NSGA-II, Multi-objective optimization.

Acknowledgements

I am grateful to God for everything in my life. Some of them were blessings, and some were lessons. I thank you for every single thing sent my way.

My thanks to my mother, father (in memoriam), brother, sister, grandmother (in memoriam), and other relatives for their support and encouragement in my life.

My gratitude to my wife Caroline, a person I love and who accompanied me on this journey, always ready to help me.

I am grateful to my daughter Alice who always gives me reasons to smile and looking for happiness in this life.

My thanks to Professor Marcone for the excellent advice during the development of this work. I appreciate for friendship and encouragement, and especially for life teachings.

My thanks Professor Luciano for his valuable advice that made work a lot.

I am grateful UFVJM and UFOP for the opportunity to qualify.

My gratitude to lady Hercília, who provided me with a home in Ouro Preto.

My thanks to my friends who always encouraged me.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

ACO	Ant Colony Optimization
ACO-ATC	Ant Colony Optimization with Apparent Tardiness Cost
CEA	Combinatorial Evolutionary Algorithm
CPP	Critical Peak Pricing
DE	Differential Evolution
DR	Demand Response
EIA	Energy Information Administration
ET	Energy Tariff
GA	Genetic Algorithm
GRAPS	Greedy Randomized Adaptive Search Procedure
HV	Hypervolume
HCC	Hierarchical Cluster Counting
MILP	Mixed Integer Linear Programming
MO-ALNS	Multi-objective Adaptive Large Neighborhood Search
MO-ALNS/D	Multi-objective Adaptive Large Neighborhood Search with Decomposition
MDE	Memetic Differential Evolution
MOEA/D	Multi-objective Evolutionary Algorithm with Decomposition
MOPSO	Multi-objective Particle Swarm Optimization
MOSA	Multi-objective Simulated Annealing
MOVNS	Multi-objective Variable Neighborhood Search

NDS	Non-dominated set
NNIA	Nondominated Neighbor Immune Algorithm
NSGA-I	Non-dominated Sorting Genetic Algorithm I
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NUPMSP	
PEC	Partial Energy Cost
RTP	Real-time Pricing
RPD	Relative Percentage Deviation
SPEA-2	Strength Pareto Evolutionary Algorithm 2
TEC	Total Energy Cost
TOU	Time-of-use
UPMSP	Unrelated Parallel Machine Scheduling Problem
UPMSP-SDS	Unrelated Parallel Machine Scheduling Problem with Sequence-dependent Setup Times
VNS	Variable Neighborhood Search

Chapter 1

Introduction

In this chapter, we introduce the contextualization of the problem addressed in Section 1.1. The motivations are in Section 1.2, while the objectives and contributions are in Section 1.3. Finally, the work structure is in Section 1.5.

1.1 Contextualization

The industrial sector is one of the largest energy consumers in the world. According to EIA (2016), this sector consumes around 54% of the total energy delivered globally. The energy used in this sector comes in many forms, such as liquid fuels, natural gas, coal, electricity, and others.

The manufacturing industry transforms materials, energy, and information into goods and products (Fysikopoulos et al., 2014). Among the various forms of energy, electricity has been one of the most consumed by this sector. In China, for example, it consumes about 50% of the electricity produced in that country (Liu et al., 2014).

In recent years, electricity prices have continuously increased for manufacturing companies in industrialized countries (Willeke et al., 2016). In Norway, the industrial electricity price, including taxes, increased by 47% between 2017 and 2018 (BEIS, 2020). This rise impacts production costs and can reduce the competitiveness of companies. In countries that implement a pricing policy so that the energy price depends on the time-of-use, the reduction of electricity costs can occur through production planning that prioritizes periods when energy is less expensive.

Few studies address scheduling problems in which the energy price depends on the time-of-use tariffs. Among them, we mention Ebrahimi et al. (2020), Zeng et al. (2018), Wang et al. (2016), Shrouf et al. (2014), and Zhang et al. (2014), where the objective includes minimizing the total energy cost.

The variable operating mode is present in many real applications. For example, Fang et al. (2011) describe a manufacturing industry that cast iron plates with slots. In this example, the machines that cut the plates operate at different speeds.

On the other hand, among several scheduling environments, the unrelated parallel machine one has received much attention recently, given its wide applicability in the industry (Cota et al., 2019). In terms of performance measures, makespan minimization is one of the most common because this criterion aims at the good utilization of the machines (Pinedo, 2016). Lastly, the sequence-dependent setup times appear in many industrial and service applications (Kopanos et al., 2009). However, we found only work of Keshavarz et al. (2021) reported in the literature addressing the unrelated parallel machine scheduling problem with sequence-dependent setup times (UPMSP-SDS), considering reducing the makespan and the total energy cost. However, the previously cited work does not assume that the machines can operate at different speeds, nor that the machine power is part of the calculation of the energy cost of each job.

This thesis focuses on the problem of unrelated parallel machines with sequence-dependent setup time, considering the TOU and the machine operating mode to fill this gap in the literature.

We propose the weighted sum method to solve small instances and heuristic multi-objective optimization methods to treat large problem instances. The heuristic methods developed are based on NSGA-II and MOVNS, given the wide use and efficiency of these methods in related problems (Bektur, 2021; Afzalirad and Rezaeian, 2017; Bandyopadhyay and Bhattacharya, 2013; Arroyo et al., 2011; Sun et al., 2019; Wu and Che, 2020).

1.2 Motivations

The present study is motivated by at least two aspects: The first is the practical interest since there are several applications of this class of problem, for example, in the manufacturing industry (Deng et al., 2019; Ruiz et al., 2019; Shen et al., 2018), computing (Gotoda et al., 2012; Park and Dally, 2010; Wolf et al., 2008), or services (Hong et al., 2021; Senbel, 2019; Pour et al., 2018). Another relevant aspect is the theoretical interest since this problem belongs to the NP-hard class because it is a generalization of the identical parallel machine scheduling problem, which is NP-hard (Garey and Johnson, 1979).

1.3 Objective

1.3.1 General Objective

This work has as objective to present a mathematical model and heuristic algorithms to address the unrelated parallel machine scheduling problem to minimize the makespan and the total energy cost.

1.3.2 Specific Objectives

The specific objectives are the following:

- Present a new mathematical model for the unrelated parallel machine scheduling problem to minimize the makespan and the total energy cost.
- Implement a method based on the mathematical model able to find Pareto-optimal solutions for this problem.
- Build a procedure based on NSGA-II to treat large instances of the problem.
- Develop a method based on MOVNS to handle large instances of the problem.
- Compare the results of the proposed methods with literature methods.

1.4 Contributions

The main contributions of this work are the following:

- Introducing a new bi-objective unrelated parallel machine scheduling problem.
- Introducing a new mixed-integer linear programming formulation able to solve small instances of this problem.
- Proposing adapted versions of the NSGA-II and the MOVNS algorithms to treat large instances of this problem.
- Creating a set of instances for this problem.
- Performing an experimental study of the proposed methods.

1.5 Work structure

We organized the remainder of this thesis as follows: in Chapter 2, we review the literature. In Chapter 3, we detail the problem addressed, and introduce the proposed mathematical model. In Chapter 4, we show the adaptation of the NSGA-II and the MOVNS algorithms to the problem. In Chapter 5, we report the computational results, which include a comparison of the results of the proposed algorithms with the exact method on small instances and a comparison with other multi-objective algorithms on large instances. Finally, we present the conclusions and directions for future work in Chapter 6.

Chapter 2

Literature Review

Here, we provide basic concepts about scheduling problems and time-of-use policy in Sections 2.1 and 2.2, respectively. Then, in Section 2.3, we reviewed the main works found in the literature related to the problem addressed.

2.1 Scheduling problems

"*Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources*" (Pinedo, 2016). They are applicable in several scenarios. For example, to allocate machines in a production system to execute a job (Deng et al., 2019; Ruiz et al., 2019; Shen et al., 2018), allocate the processors of the computing system to perform specific calculations (Gotoda et al., 2012; Park and Dally, 2010; Wolf et al., 2008), or allocate workers to perform services according to customer demand (Hong et al., 2021; Senbel, 2019; Pour et al., 2018).

We can describe scheduling problems by three fields notation $(\alpha \mid \beta \mid \gamma)$ introduced by Graham et al. (1979), where the field α describes the machine environment and contains only one entry. The field β provides details of processing characteristics and restrictions. This field can contain one, several, or no entries. The field γ describes the objective to be minimized.

Next, we present some examples for each field. According to Pinedo (2016), we can cite some machine environments (α):

Single machine: There is a single machine to process each job j;

Identical Parallel machine: There are m identical machines in parallel. Job j requires a single operation and can be processed on any of the m machines.

Unrelated Parallel machine: This environment is a generalization of the previous one. There are m different machines in parallel. For each machine i, job j has a processing time p_{ij}.

Flow shop: There are m machines. Each job j requires one operation in each machine, to be processed. All jobs follow the same route, that is, they be processed first on machine 1, then on machine 2, and so on.

Parallel machine with different speeds: There are m machines in parallel with different speeds. The speed of each machine i is denoted by v_i . The processing time p_j of job j on machine i is equal to p_j/v_i.

In addition, we quote above some processing restrictions (β):

Release dates: Job j cannot start processing before its release date.

Preemptions: The job processing can stop at any time. In this way, it is possible to stop processing a job at any time and swap to another machine.

Sequence dependent setup times: It is the time required after job j to prepare the machine to process job k. This time depends on the sequence in which the jobs are allocated on machine.

Also according to Pinedo (2016); Chiaraviglio et al. (2011), we can list some objectives for the scheduling problems (γ):

Makespan: It is equal to the completion time of the last job, assuming that the completion time of any job is when it finishes;

Total weighted tardiness: This objective indicates the total tardiness generated by scheduling;

Total weighted earliness: It indicates the total earliness obtained by scheduling;

Maximum Lateness: Lateness measures whether given scheduling conforms to due dates and takes negative values to early jobs.

Energy consumption: The sum of energy consumed by machines to execute the jobs.

Energy cost: The sum of energy consumed by machines to execute the jobs multiplied by the energy tariff.

Following this notation, the problem addressed in this thesis is denoted by: $R_m \mid S_{ijk}, TOU \mid C_{max}, TEC$. The field R_m identifies that the machines are parallel and unrelated. S_{ijk} means that the setup time is sequence-dependent. TOU indicates that the problem considers time-of-use policy. Lastly, the field $< C_{max}, TEC >$ denotes that the objectives are makespan and total energy cost, respectively.

A relevant characteristic of the problem considered in this work, and found in real-world applications, is sequence-dependent setup time. For example, in the printing industry, a printer receives various orders (an order is considered a job). Each order has different colors, sizes, or paper types. Printers (machines) must be cleaned and reset when the color, size, or paper type of the following order to be processed is different from the current order. In this case, setup times depend on the processing sequence of the jobs (Huang et al., 2010).

Another characteristic of the problem, also found in real-world applications, is energy pricing under the time-of-use policy. For example, Wang et al. (2016) present the case in a ceramic glass industry located in Shanghai, China. This company uses a furnace for heating and handling the glass. The primary energy used to heat the furnace is electricity, and its price varies according to the hour of the day.

2.2 Time-of-use electricity price

We can define Demand Response (DR) as the changes in electricity use by final consumers from their standard consumption profiles in response to changes in the price of electricity over time. Typically, the consumers are encouraged to reduce electricity usage at high wholesale market prices or when system reliability is compromised (Albadi and El-Saadany, 2007).

In the DR program, the customer signs a contract with the local utility to reduce their demand as and when requested. The advantage of this program for the utility

is the reduction of peak load, thus saving on expensive generation reserves. For the customer, the benefit is the cost-reducing provided by the local utility (Aalami et al., 2015).

DR programs are classified into one of two categories: price-based and incentive-based DR programs. The first category refers to change in electricity consumption by the end-use customer in response to dynamic prices. This category includes the TOU (time-of-use) rate, RTP (real-time pricing), and CPP (critical peak pricing), and they are entirely voluntary. The second category is designed by operators and includes Direct Load Control, Interruptible/Curtailable service, Demand Bidding, Emergency DR, Capacity Market, and Ancillary services market program. These programs give participating customers incentive payments and consider penalties for customers who enroll but do not respond in the needed time, depending on the program types and conditions (Falsafi et al., 2014). This thesis focuses on the TOU program, which is briefly introduced in the following.

The TOU is the most popular pricing method among DR strategies (Ding et al., 2016). It is a method of demand-side management in which the price varies hourly on the day. It is an alternative to the traditional time-invariant rate because it encourages consumers to change their electricity usage patterns. It serves as a cost-effective way to realize electricity demand response and reduce peak demand (Wang and Li, 2015).

We can divide the TOU into two groups based on the hour of the day, on-peak and off-peak. On-peak is the time of day when energy demand is highest. Off-peak rates are during the time of day when energy demand is lowest. Usually, the DR program offers a high price of electricity during on-peak periods and lower prices during off-peak periods (Cheng et al., 2019).

2.3 Related works

Here, we present a literature review with previous research that addressed scheduling problems and also considered objectives related to this work.

Some studies address the scheduling problem only to minimize energy consumption.

Shrouf et al. (2014) proposed a mathematical model for the single machine scheduling problem to minimize the total costs of energy considering continuous changes in energy prices (time-of-use), and preemption is not allowed. The planning horizon is divided

into several segments of equal length (called periods). However, the model cannot solve large instances within a reasonable computational time for decision-making. For this reason, they also proposed a genetic algorithm. The computational results indicated the possibility of reducing energy consumption by up to 30% when they compared the genetic algorithm solution and the "as soon as possible" heuristic solution.

Tsao et al. (2020) presented a fuzzy model integrated into a genetic algorithm for a single machine problem to minimize the total costs of energy, considering constraint carbon footprint, constraint makespan, variable electricity prices (time-of-use), and preemption is allowed. The processing time of the jobs depends on the allocation of sources. To deal with uncertainty related to resource allocation costs, they adopted a fuzzy approach combined with a Genetic Algorithm to decide machine status ("on" or "idle"), processing time, and jobs sequence. They tested the method in instances of up to 200 jobs. The results indicated a 4.20% reduction in total energy consumption compared to the traditional genetic algorithm.

Other studies address the scheduling problem aim minimizing energy consumption combined with a second objective.

Cota et al. (2018) proposed a mathematical model and applied a mathematical heuristic called multi-objective smart pool search for the UPMSP-SDS. The objective functions are to minimize the makespan and the total energy consumption. This study assumed operating mode on machines. In the experiments, they used a set of instances with up to fifteen jobs and five machines randomly generated. They adopted hypervolume and set coverage metrics to compare the proposed algorithm with the ϵ-constraint exact method. They concluded that the objectives are conflicting and that energy consumption is very important since it represents a cost for the industry.

Cota et al. (2019) introduced the MO-ALNS and MO-ALNS/D algorithms to treat instances of up to 250 jobs and 30 machines of the same problem described previously. The MO-ALNS algorithm is a multi-objective version of the Adaptive Large Neighborhood Search – ALNS (Ropke and Pisinger, 2006), in its turn, the MO-ALNS/D algorithm combines the multi-objective MOEA/D (Zhang and Li, 2007) with ALNS. The results show that the MO-ALNS/D algorithm was able to find better results than MO-ALNS in most instances in the hypervolume, set coverage, and Hierarchical Cluster Counting (HCC) (Guimarães et al., 2009) metrics.

Wu and Che (2019) proposed a memetic differential evolution (MDE) algorithm for the UPMSP in which the objectives are also to minimize the makespan and the

total energy consumption. They considered the operating mode but did not consider sequence-dependent setup times. The problem involves assigning jobs to machines and selecting an appropriate processing speed level for each job. They proposed a local search approach integrated with the DE algorithm to improve it. The computational results showed that the proposed approach significantly improves the basic DE. Also, the MDE outperforms the SPEA-2 and NSGA-II algorithms.

Liang et al. (2015) presented the Ant Colony Optimization algorithm with the Apparent Tardiness Cost (ACO-ATC) rule for the UPMSP, aiming to minimize the weighted sum of the total tardiness and the energy consumption. In this problem, machines need to wait until jobs are ready. However, it is necessary to decide whether the machine remains on or off during the wait. Turning off the machine and waiting until the job is ready saves energy. On the other hand, keeping the machine turned on and waiting for the next job saves the setup time required when turning on the machine. This problem considers setup time to the jobs, but it is not sequence-dependent. In the experiments, they compared the ACO-ATC results with the classic ACO and a GRASP-based algorithm (Feo and Resende, 1995) on 91 instances, with 5 runs for each instance. The proposed algorithm was better than the other approaches in most tested instances.

We found studies that only address the minimization of the total energy cost.

Ding et al. (2016) presented two approaches to UPMSP: the first introduces a time-interval-based Mixed Integer Linear Programming (MILP) formulation. The second is a reformulation of the problem using the Dantzig-Wolfe decomposition and a column generation heuristic. They considered the operating mode but did not consider the setup time for the problem. The objective is to minimize total energy cost. According to the results, the MILP formulation overcame the column generation method concerning solution quality and execution time when electricity prices stay stable for a relatively long period. On the other hand, the column generation method performed better when the electricity price frequently changed (i.e., every half hour). They performed computational experiments with 120 randomly generated instances.

Cheng et al. (2018) improved the formulation by Ding et al. (2016) by significantly reducing the number of decision variables. They performed computational experiments with 120 randomly generated instances as they could not obtain the instances of Ding et al. (2016). They reformulated some constraints such as job completion, machine availability, and non-preemption. They proposed a tighter and more compact model. The results showed that the new formulation achieves better results concerning the

solution quality and execution time.

Saberi-Aliabad et al. (2020) proposed the fix-and-relax heuristic algorithm in two stages for the same problem previously described. In the first stage of the algorithm, jobs are assigned to the machines, and the second one solves a scheduling problem on single machines. They tested their method in 360 instances randomly generated following the same parameter values as previous studies. They compared the proposed method with the algorithms of Che et al. (2017) and Cheng et al. (2018). The results showed that the fix-and-relax algorithm overcame the others.

Finally, we present studies that address the scheduling problem considering minimizing the energy cost combined with another objective.

Zeng et al. (2018) dealt with the bi-objective uniform parallel machine scheduling to minimize the total energy cost and the number of machines under the TOU electricity pricing. They considered the operating mode on machines. They proposed a new mathematical model and a heuristic algorithm for it. They adapted the heuristic proposed by Che et al. (2016) to the single machine problem. In this heuristic, the jobs are inserted in non-decreasing order of their processing time. In the mathematical formulation, it was used the number of machines m as a constraint. In this way, it is possible to obtain the Pareto front by changing the value of m. They compared the heuristic and the mathematical model results in instances with up to 60 jobs. Moreover, they tested the heuristic algorithm in instances with up to 5000 jobs, randomly generated, and in instances from a manufacturing company in Shaanxi Province, China. They concluded that the heuristic algorithm generates high-quality solutions within a reasonable time limit.

Moon et al. (2013) addressed the UPMSP under the TOU electricity pricing to minimize the weighted sum of makespan and energy cost. The problem does not consider the machine setup time. They present a hybrid genetic insertion algorithm. Computational tests indicate that the proposed algorithm reduces the total energy cost compared to the classical genetic algorithm. Cheng et al. (2019) presented a mathematical formulation and a genetic algorithm for a problem similar to described previously. The proposed model can solve the instances of Moon et al. (2013) quickly . They randomly generated instances with 2 to 3 machines and 5 to 9 jobs to verify the effectiveness of the proposed model. The results presented by their formulation overcome that of the genetic algorithm in terms of solution quality. Kurniawan et al. (2017) proposed a genetic algorithm with a delay mechanism for the same problem addressed by Moon et al. (2013).

The job delay mechanism improves the schedule since delaying the start of jobs avoids the high electricity price period. The proposed algorithm uses a probabilistic method to determine which jobs must be delayed and how long to delay. The proposed algorithm handled instances of up to 30 jobs and 15 machines. The results showed that the proposed method provided better solutions than the classical genetic algorithm.

Kurniawan et al. (2020) proposed a triple-chromosome genetic algorithm for the same problem described previously. The triple-chromosome represents the job sequencing, the job assignment, and the starting time of the job. The results showed that the proposed method performs better than other methods tested: classical genetic algorithm, double-chromosome genetic algorithm, and single-chromosome genetic algorithm.

Zhang et al. (2021) approached the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times (UPMSP-SMDST) with limited worker resources and learning effect, denoted by NUPMSP. In this problem, the workers who perform the machine setup have different skill levels. Due to the learning effect, this skill will increase until it reaches the maximum level. Also, the number of workers is limited. The objective is to minimize makespan and total energy cost. They proposed a new combinatorial evolutionary algorithm (CEA) and compared it to the neighbor immune algorithm (NNIA) and the NSGA-II. Based on the result obtained in 72 instances, they concluded that CEA outperformed the other algorithms in almost all instances.

Keshavarz et al. (2021) addressed UPMSP with sequence-dependent setup times to minimize makespan and energy consumption under TOU electricity pricing. They presented a mixed-integer bi-objective mathematical model and applied the ϵ-constraint method to solve small with up to 10 jobs and medium-sized instances with 12 to 45 jobs. They applied the Multiple Objective Particle Swarm Optimization algorithm (MOPSO) and the Multiple Objective Simulated Annealing algorithm (MOSA) to large-sized instances with 60 to 250 jobs. The results indicated that the MOSA performs better than the MOPSO.

The problem addressed in this work is similar to that defined in the study of Keshavarz et al. (2021). However, there are some differences. In the present work, we assume that the machine can operate in different modes to process a job. The operating mode affects the processing time and consequently the energy cost. In addition, we use the power of the machine to calculate the energy cost of a job. Table 2.1 summarizes the characteristics of scheduling problems treated by our work compared to literature references.

Table 2.1: Summary of characteristics addressed by our work compared to literature studies.

Reference	Unrelated parallel machines	Sequence-dependent setup	Makespan	Total energy cost	time-of-use	Multi-objective	Exact method	Metaheuristic method	Operating mode
Moon et al. (2013)	✓		✓	✓	✓	✓		✓	
Shrouf et al. (2014)				✓	✓		✓	✓	
Liang et al. (2015)	✓					✓	✓	✓	
Ding et al. (2016)	✓			✓	✓		✓		✓
Kurniawan et al. (2017)	✓		✓	✓	✓	✓		✓	
Cota et al. (2018)	✓	✓	✓			✓	✓		✓
Cheng et al. (2018)	✓			✓	✓		✓		
Zeng et al. (2018)				✓	✓	✓	✓	✓	✓
Cota et al. (2019)	✓	✓	✓			✓		✓	✓
Wu and Che (2019)	✓		✓			✓		✓	✓
Cheng et al. (2019)	✓		✓	✓	✓	✓	✓		
Tsao et al. (2020)				✓	✓		✓	✓	
Saberi-Aliabad et al. (2020)	✓			✓	✓		✓	✓	
Kurniawan et al. (2020)	✓	✓	✓	✓	✓	✓	✓	✓	
Zhang et al. (2021)	✓	✓	✓	✓	✓	✓		✓	
Keshavarz et al. (2021)	✓	✓	✓	✓	✓	✓	✓	✓	
Our proposal	✓	✓	✓	✓	✓	✓	✓	✓	✓

Chapter 3

Proposed Mathematical Model

In this chapter, we detail, in Section 3.1, the notation used to describe the problem. The mathematical formulation is in Section 3.2. Finally, we present a numerical example with dummy data for the problem addressed in Section 3.3.

3.1 Notation

To define the UPMSP-SDS, we characterize the problem in this section and introduce a MILP formulation to solve it.

The following are the characteristics of the problem addressed in this work:

- There are a set $N = \{1, \ldots, n\}$ of jobs, a set $M = \{1, \ldots, m\}$ of machines, and a set $L = \{1, \ldots, o\}$ of different operating modes, such that each operating mode $l \in L$ is associated with a multiplication factor of speed v_l and a multiplication factor of power λ_l;
- The machines are unrelated parallel. In other words, the processing time of job $j \in N$ can be different on each machine $i \in M$;
- There is a planning horizon that consists of a set $H = \{0, \ldots, |H|\}$ of time instants, and we must execute all jobs within this horizon;
- All jobs are available to be processed at the beginning of the planning horizon $h = 0$;

- Each job $j \in N$ must be allocated to exactly one machine $i \in M$;
- There is a processing time P_{ij} to execute a job $j \in N$ on a machine $i \in M$;
- There is a sequence-dependent setup time S_{ijk} to execute a job $k \in N$ after another job $j \in N$ on a machine $i \in M$;
- Each machine $i \in M$ has a power π_i at normal operating speed;
- The operating mode $l \in L$ of each job determines the multiplication factor of power (λ_l). It also determines the multiplication factor of speed (V_l), which, in turn, is related to the processing time of each job;
- There is a set D of days on the planning horizon H;
- Each day is discretized into $sizeD$ time intervals. For example, for discretizing a day in minutes, $sizeD = 1440$; for the discretization of one day in hours, $sizeD = 24$;
- To each day $t \in H$, we have a peak hour, which starts at the time $startp_t \in H$ and ends at the time $endp_t \in H$;
- ET^{off} and ET^{on} represent the energy tariff (\$/KWh) in off-peak and on-peak hours, respectively.

Table 3.1 presents the decision and auxiliary variables needed to model the problem.

Table 3.1: Decision and auxiliary variables for the problem

Name	Description
X_{ijhl}	Binary variable that assumes value 1 if the job j is allocated on the machine i at time h and in the operating mode l, and value 0, otherwise
PEC_t^{on}	Partial Energy Cost (\$) during the on-peak in day $t \in D$
PEC_t^{off}	Partial Energy Cost (\$) during the off-peak in day $t \in D$
$C_{\max}$	The maximum completion time of the jobs, also known as makespan
TEC	Total Energy Cost (\$)

3.2 Formulation

Based on the formulation of Pinto et al. (2019), we can define the problem through Equations (3.1) - (3.12).

$$\min \ C_{\max} \tag{3.1}$$

$$\min \ TEC \tag{3.2}$$

Subject to:

$$\sum_{i=1}^{m}\sum_{l=1}^{o}\sum_{h=0}^{|H|-\left\lceil\frac{P_{ij}}{V_l}\right\rceil} X_{ijhl} = 1 \qquad \forall j \in N \tag{3.3}$$

$$X_{ijhl} + \sum_{u=h}^{\min\left(h+\left\lceil\frac{P_{ij}}{V_l}\right\rceil+S_{ijk}-1,|H|\right)}\sum_{l_1=1}^{o} X_{ikul_1} \leq 1 \quad \forall i \in M, j \in N, k \in N, l \in L, j \neq k \tag{3.4}$$

$$C_{\max} \geq X_{ijhl} \times \left[h + \left\lceil\frac{P_{ij}}{V_l}\right\rceil\right], \qquad \forall i \in M, j \in N, h \in H, l \in L \tag{3.5}$$

$$PEC_t^{off} \geq \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{o} \frac{\lambda_l \times \pi_i \times ET^{off} \times 24}{sizeD} \times \tag{3.6}$$

$$\left\{ \sum_{h=sizeD\times(t-1)}^{startp_t-1} X_{ijhl} \times \right.$$

$$\left[\min\left(h + \left\lceil\frac{P_{ij}}{V_l}\right\rceil, startp_t\right) - h + \max\left(0, h + \left\lceil\frac{P_{ij}}{V_l}\right\rceil - endp_t - 1\right)\right]$$

$$+ \sum_{h=startp_t}^{endp_t-1} X_{ijhl} \times \left[\max\left(0, h + \left\lceil\frac{P_{ij}}{V_l}\right\rceil - endp_t - 1\right)\right]$$

$$\left. + \sum_{h=endp_t}^{|H|-1} X_{ijhl} \times \left\lceil\frac{P_{ij}}{V_l}\right\rceil \right\} \qquad \forall t \in D$$

$$PEC_t^{on} \geq \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{l=1}^{o} \frac{\lambda_l \times \pi_i \times ET^{on} \times 24}{sizeD} \times \tag{3.7}$$

$$\left\{ \sum_{h=sizeD\times(t-1)}^{startp_t-1} X_{ijhl} \times \right.$$

$$\left[\max\left(0, \min\left(h + \left\lceil \frac{P_{ij}}{V_l} \right\rceil - 1, endp_t\right) - (startp_t - 1)\right)\right]$$

$$\left. + \sum_{h=startp_t}^{endp_t-1} X_{ijhl} \times \left[\min\left(h + \left\lceil \frac{P_{ij}}{V_l} \right\rceil, endp_t + 1\right) - h\right] \right\} \qquad \forall t \in D$$

$$TEC \geq \sum_{t=1}^{sizeD} \left(PEC_t^{off} + PEC_t^{on}\right) \tag{3.8}$$

$$X_{ijhl} \in \{0, 1\} \qquad \forall i \in M, j \in N, h \in H, l \in L \tag{3.9}$$

$$C_{\max} \geq 0 \tag{3.10}$$

$$PEC_t^{off} \geq 0 \qquad \forall t \in D \tag{3.11}$$

$$PEC_t^{on} \geq 0 \qquad \forall t \in D \tag{3.12}$$

The objectives of the problem are to minimize, simultaneously, the makespan and the total energy cost, defined by Equations (3.1) and (3.2), respectively. The constraint set (3.3) ensures that every job $j \in N$ is allocated on a machine with a single operating mode and ends its execution inside the planning horizon. Constraints (3.4) define that if the job k is assigned to machine i immediately after the job j, then the start time of the job k must be greater than the sum of the end time of the job j and the setup time between them. It is important to highlight that the setup and processing times must satisfy the triangular inequality for the previous model to be valid (Rosa and Souza, 2009). The constraint set (3.5) determines a lower bound for the makespan. Constraints (3.6) and (3.7) define a lower bound in the partial energy cost for off-peak hours (PEC^{off}) and in the partial energy cost for on-peak hours (PEC^{on}), respectively. Note that a job can be partially executed in the on-peak hours and partially in the off-peak hours and that the total energy cost is directly related to the energy price and the job execution time. Constraint (3.8) ensures a lower bound for the total energy cost.

Constraints (3.9)-(3.12) define the domain of the decision and auxiliary variables of the problem.

We generalize the model proposed by Pinto et al. (2019) in the present work to solve planning horizons with more than one day. Furthermore, our model makes it possible to discretize the day in any time interval and not only in 10-minute intervals.

The calculation of the energy cost of a job j depends on its execution time during the on-peak and off-peak time. Thus, there are six possible cases:

Case 1: The job j starts and ends before the on-peak hours;

Case 2: The job j starts before the on-peak hours and ends in the on-peak hours;

Case 3: The job j starts and ends in the on-peak hours;

Case 4: The job j starts during the on-peak hours and ends after the on-peak hours;

Case 5: The job j starts and ends after the on-peak hours;

Case 6: The job j starts before the on-peak hours and ends after the on-peak hours.

3.3 Numerical Example

To illustrate cases 1 to 5, let Figure 3.1. It shows the execution of five jobs $N = \{2, 4, 1, 5, 3\}$ in the scheduling of a single machine $i = 1$ in a single operating mode $l = 1$ on day $t = 1$ of the planning horizon. Let the start of the on-peak hours ($startp_1$) equal to 18; the end of the on-peak hours ($endp_1$) equal to 21; the multiplication factor of power (λ_l) equal to 1; the energy consumption of machine at normal operating (π_1) equal to 100; the energy tariff in the on-peak hours (ET^{on}) equal to $0.10\$/KWh$ and in the off-peak hours (ET^{off}) equal to $0.05\$/KWh$; the multiplication factor of speed v_l equal to 1. In this example, we consider discretization in hours. This figure shows that jobs 4, 1, and 5 are performed in the on-peak hours, partially or totally, and jobs 2 and 3, in turn, in the off-peak hours.

For this example, Eqs. (3.7) and (3.6) are reduced to Eqs. (3.13) and (3.14) below:

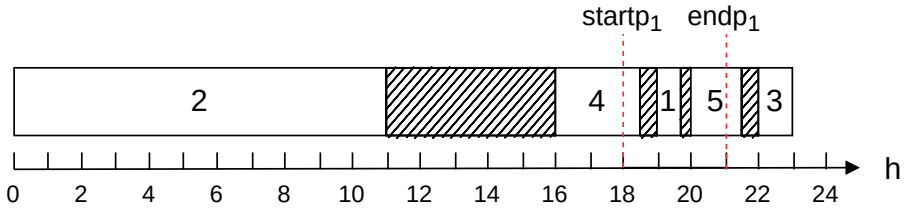


Figure 3.1: Example to illustrate the calculation of the energy cost on a machine

$$PEC_1^{on} = \underbrace{\sum_{j=1}^{n} \frac{1 \times 100 \times 0.10 \times 24}{24}}_{\text{Parcel 1(a)}} \times \tag{3.13}$$

$$\underbrace{\left\{ \sum_{h=0}^{18-1} X_{1jh1} \times \left[\max\left(0, \min\left(h + \left\lceil \frac{P_{1j}}{1} \right\rceil - 1, 21\right) - (18-1)\right)\right]\right.}_{\text{Parcel 2(b)}}$$

$$\underbrace{\left. + \sum_{h=18}^{21-1} X_{1jh1} \times \left[\min\left(h + \left\lceil \frac{P_{1j}}{1} \right\rceil, 21+1\right) - h\right]\right\}}_{\text{Parcel 3(c)}}$$

$$PEC_1^{off} = \underbrace{\sum_{j=1}^{n} \frac{1 \times 100 \times 5 \times 24}{24}}_{\text{Parcel 1(d)}} \times \tag{3.14}$$

$$\underbrace{\left\{ \sum_{h=0}^{18-1} X_{1jh1} \times \left[\min\left(h + \left\lceil \frac{P_{1j}}{1} \right\rceil, 18\right) - h + \max\left(0, h + \left\lceil \frac{P_{1j}}{1} \right\rceil - 21 - 1\right)\right]\right.}_{\text{Parcel 2(e)}}$$

$$\underbrace{+ \sum_{h=18}^{21-1} X_{1jh1} \times \left[\max\left(0, h + \left\lceil \frac{P_{1j}}{1} \right\rceil - 21 - 1\right)\right]}_{\text{Parcel 3(f)}}$$

$$\underbrace{\left. + \sum_{h=21}^{24-1} X_{1jh1} \times \left\lceil \frac{P_{1j}}{1} \right\rceil \right\}}_{\text{Parcel 4(g)}}$$

Table 3.2 illustrates the contribution of each job to the total energy cost, according to

the example in Figure 3.1. The column "# Job" represents the job, the column "Case" shows the contemplated case, and the columns "Contr. on-peak" and "Contr. off-peak" show the contributions of the job to the energy cost of each job in the on-peak and off-peak hours, respectively.

Table 3.2: Energy cost by job in the Example of Figure 3.1

# Job	Case	Contr. on-peak hours	Contr. off-peak hours
2	1	$\underbrace{10}_{(a)} \times (\underbrace{0}_{(b)} + \underbrace{0}_{(c)}) = 0$	$\underbrace{5}_{(d)} \times (\underbrace{11}_{(e)} + \underbrace{0}_{(f)} + \underbrace{0}_{(g)}) = 55$
4	2	$\underbrace{10}_{(a)} \times (\underbrace{1}_{(b)} + \underbrace{0}_{(c)}) = 10$	$\underbrace{5}_{(d)} \times (\underbrace{2}_{(e)} + \underbrace{0}_{(f)} + \underbrace{0}_{(g)}) = 10$
1	3	$\underbrace{10}_{(a)} \times (\underbrace{0}_{(b)} + \underbrace{1}_{(c)}) = 10$	$\underbrace{5}_{(d)} \times (\underbrace{0}_{(e)} + \underbrace{0}_{(f)} + \underbrace{0}_{(g)}) = 0$
5	4	$\underbrace{10}_{(a)} \times (\underbrace{0}_{(b)} + \underbrace{1}_{(c)}) = 10$	$\underbrace{5}_{(d)} \times (\underbrace{0}_{(e)} + \underbrace{1}_{(f)} + \underbrace{0}_{(g)}) = 5$
3	5	$\underbrace{10}_{(a)} \times (\underbrace{0}_{(b)} + \underbrace{0}_{(c)}) =$	$\underbrace{5}_{(d)} \times (\underbrace{0}_{(e)} + \underbrace{0}_{(f)} + \underbrace{1}_{(g)}) = 5$

The total energy cost found to the schedule shown in Figure 3.1 is 105.

To illustrate case 6, consider Figure 3.2. It shows the execution of three jobs $N = \{2, 1, 3\}$ on a single machine $i = 1$ in operating mode $l = 1$ during day $t = 1$ of the planning horizon. Let also the start of the on-peak hours ($startp_1$) equal to 18; the end of the on-peak hours ($endp_1$) equal to 21; the multiplication factor of power (λ_l) equal to 1; the energy consumption of machines at normal operation (π_1) equal to 100; the energy tariff in the on-peak hours (ET^{on}) equal to 0.10 $\$/KWh$ and in the off-peak hours (ET^{off}) equal to 0.05 $\$/KWh$; and the multiplication factor of speed equal to 1. Such as in the previous example, we consider discretization in hours. This figure shows that job 1 is performed in the on-peak hours and jobs 2 and 3, in turn, in the off-peak hours.

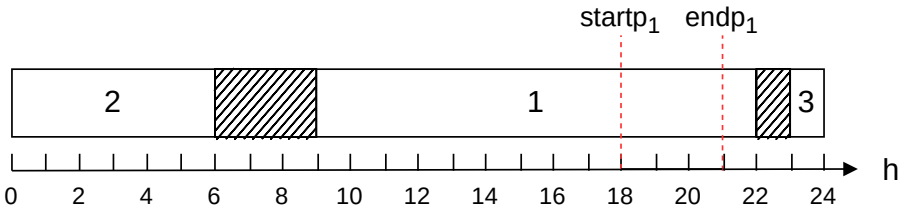


Figure 3.2: Schedule example for case 6

The contribution of job 1 to the energy cost in the on-peak hours is 30, and the contribution to the cost in the off-peak hours is 50.

Thus, calculating similarly to the previous example, we conclude that the total energy cost for the schedule shown in Figure 3.2 is 155.

Chapter 4

Proposed Algorithms

In this chapter, we present the algorithms to treat the problem in this study. In Section 4.1, we detail the Weighted Sum Method. In Section 4.2, we show the representation and evaluation of the solution used by metaheuristics algorithms. In Sections 4.3 and 4.4, we present the NSGA-II and MOVNS algorithms, respectively.

4.1 Weighted Sum Method

We used the weighted sum method (Marler and Arora, 2004) to solve the multi-objective optimization problem addressed using a mathematical programming solver. This method converts the multi-objective problem into a single objective problem using the weighted sum of the objectives.

For this, consider Equation (4.1):

$$\min \quad z(X) = \left[\alpha \times \left(\frac{C_{\max}(X)}{|H|}\right) + (1-\alpha) \times \left(\frac{TEC(X)}{Cost_{\max}}\right)\right] \tag{4.1}$$

where:

- α: Real number in range [0, 1];

- $|H|$: Represents the cardinality of the set H;
- $Cost_{\max}$: It is the estimate for the maximum energy cost used to normalize the total energy cost.

and:

$$Cost_{\max} = n \times \max\left(\frac{P_{ij}}{V_l}\right) \times PEC^{on} \times \max(\pi_i)$$

The problem constraints are those defined by Equations (3.3)-(3.12).

Algorithm 4.1 describes all the steps of the weighted sum method implemented.

Algorithm 4.1: Weighted Sum Method

input: $\Delta = \{\alpha_1, \alpha_2, \cdots, \alpha_{sv}\}$, time_limit

```
1 NDS ← ∅
2 foreach α_i ∈ Δ do
3 |   model_result ← RunWeightedSumModel(α_i, time_limit)
4 |   s.Makespan ← GetMakespan(model_result)
5 |   s.TEC ← GetTEC(model_result)
6 |   NDS ← AddSolution(s)
7 end
8 return NDS
```

Algorithm 4.1 receives the set Δ with the values for α and the time limit as input. In line 1, we initialize the non-dominated set (NDS) as empty. Then, we execute the loop defined between lines 2-7 for each value α. In line 3, we obtain the result from the execution of the model. Then, we get the Makespan and TEC values resulting from the model execution. Then, in line 6, we add the solution obtained to the NDS. Finally, in line 8, the method returns the generated non-dominated set.

In the weighted sum method, the decision-maker must define a weight for each objective function. The value of this weight reflects the relative importance of each objective in the overall solution. We adopted several combinations of weights to find the most significant number of optimal Pareto solutions to the problem addressed.

We used the following parameters for Algorithm 4.1:

- The set $\Delta = \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ with the possible values

for α;

- The time limit for each execution of the mathematical model, defined as $\mathsf{time\_limit} = 800 \times n \times \ln(m)$ seconds for each α value, where m is the number of machines, and n is the number of jobs;
- $sizeD = 144$: To discretize the day at intervals of 10 minutes each.

Since the problem addressed is a generalization of the identical parallel machine scheduling problem, which is NP-hard (Garey and Johnson, 1979), we can conclude that it also belongs to this problem class. For this reason, we used heuristic multi-objective algorithms to treat it. In this chapter, we present the proposed NSGA-II in Section 4.3 and the MOVNS2 in Section 4.4.

4.2 Representation and evaluation of the solution

We represent a solution by m lists. In each position of these lists, we have a pair of keys. The first value indicates the job, and the second is associated with the operating mode. We exemplify in Figure 4.1 the representation of a solution for an instance with two machines and six jobs. We allocate jobs 6, 4, and 2 on machine M1 and define the operating modes 3, 1, and 1 in this order. In addition, we have allocated jobs 3, 5, and 1 on machine M2 and set the operating modes 2, 3, and 1 in this order. Note that the rectangles represent the jobs, and the circles represent the operating modes.

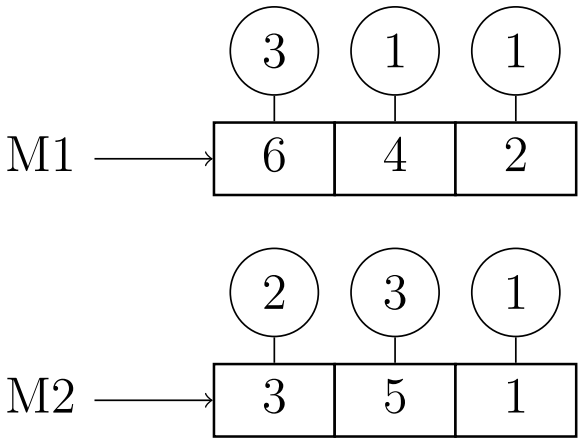


Figure 4.1: Representation of the solution

We evaluate a solution according the Equations (3.1) and (3.2).

4.3 NSGA-II

The NSGA-II algorithm was proposed by Deb et al. (2002). There are in literature many reports of successful use of this algorithm (Deb et al., 2007; Liu et al., 2014; Wang et al., 2017; Babazadeh et al., 2018). This algorithm is an alternative to the exact method described in the previous section to find an approximation of the Pareto-optimal front in large instances in an adequate computational time for decision-making.

Algorithm 4.2 describes how the implemented NSGA-II works.

Algorithm 4.2: NSGA-II

input: size_{pop}, prob_{mut}, stopping_criterion
$\mathsf{P}_0 \leftarrow$ Generate initial population of size_{pop} individuals
$\mathsf{Q}_0 \leftarrow \emptyset$
$t \leftarrow 0$
while stopping_criterion not satisfied **do**
  $\mathsf{R}_t \leftarrow \mathsf{P}_t \cup \mathsf{Q}_t$
  $\mathcal{F} \leftarrow$ `Fast non-dominated sorting` (R_t)
  $\mathsf{P}_{t+1} \leftarrow \emptyset$
  $i \leftarrow 1$
  while $|\mathsf{P}_{t+1}|+|\mathcal{F}_i| \leq \mathsf{size}_{pop}$ **do**
    `Compute Crowding Distance` $(\mathcal{F}_i)$
    $\mathsf{P}_{t+1} \leftarrow \mathsf{P}_{t+1} \cup \mathcal{F}_i$
    $i \leftarrow i + 1$
  end
  if $|\mathsf{P}_{t+1}| < \mathsf{size}_{pop}$ **then**
    `Sort` $(\mathcal{F}_i, \prec_{obj})$
    $j = 1$
    while $|\mathsf{P}_{t+1}| < \mathsf{size}_{pop}$ **do**
      $\mathsf{P}_{t+1} \leftarrow \mathsf{P}_{t+1} \cup \mathcal{F}_i[j]$
      $j \leftarrow j + 1$
    end
  end
  $\mathsf{Q}_{t+1} \leftarrow$ `Crossover`(P_{t+1})
  $\mathsf{Q}_{t+1} \leftarrow$ `Mutation`$(\mathsf{Q}_{t+1}, \mathsf{prob}_{mut})$
  $t \leftarrow t + 1$
end
NDS $\leftarrow$ non-dominated solutions of P_t
return NDS

Algorithm 4.2 receives the following input parameters: the population size (size_{pop}), the probability of mutation (prob_{mut}), and the stopping_criterion. In line 1, we create an initial population P_0. Then, in the main loop (lines 4–25), we combine the parent P_t and offspring Q_t to generate a new population R_t (line 5). In line 6, we apply the fast non-dominated sorting method to divide the population R_t into non-dominated sets, called fronts, $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k$. A front $\mathcal{F}_i$ dominates another $\mathcal{F}_j$, if and only if, $i < j$ and $\mathsf{R}_t = \mathcal{F}_1 \cup \mathcal{F}_2 \ldots \mathcal{F}_k$. In lines 9–13, we select the best fronts of $\mathcal{F}$ to include in the population P_{t+1}. We repeat this procedure as long as it is possible to include a new front in P_{t+1} without exceeding the population size. Then we check the size of the population obtained. If it is not exactly size_{pop}, we sort the next front i of $\mathcal{F}$ that has not yet been included in P_{t+1}, according to the crowding distance, and we select the first $\mathsf{size}_{pop} - |\mathsf{P}_{t+1}|$ individuals to population P_{t+1}. In lines 22 and 23, we apply the crossover and mutation operators to generate the new population Q_{t+1}, with size_{pop} individuals.

The following subsections describe the Fast non-dominated sorting, the Crowding Distance procedures, how to generate an initial population, the crossover operators, and the mutation operators, respectively.

4.3.1 Fast non-dominated sorting

The NSGA-II algorithm uses the Fast Non-Dominated Sorting method to sort a population. It has complexity $O(n\_obj \times \mathsf{size}_{pop}^2)$, where n_obj is the number of objectives and size_{pop} is the size of the population. This method receives a population P as an input parameter and returns a set of non-dominated fronts $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k)$.

The Fast Non-Dominated Sorting method is defined by Algorithm 4.3:

In the loop between lines 1 and 15 of Algorithm 4.3, we select each p solution of P. In the loop between lines 4 and 11, we choose each q solution of P, which q is different from p. Then, we check if p dominates q. If yes, we insert the solution q into the set S_p. In contrast, if q dominates p, we increment n_p by one. Next, we check which solutions have the value n_p equal to zero (line 12). The solutions with the value n_p equal to zero are non-dominated, so they must be included in the $\mathcal{F}_1$ front (line 13).

Then, the algorithm starts another loop (lines 17 – 29) to generate the others fronts. We create each front $\mathcal{F}_i$ in this loop, selecting each q non-dominated solution that is not yet on another front. We repeat this loop to generate new fronts from P as long as possible.

Algorithm 4.3: Fast Non-Dominated Sorting

```
   input: P
 1 foreach p ∈ P do
 2 |   S_p ← ∅
 3 |   n_p = 0
 4 |   foreach q ∈ P do
 5 |   |   if (p ≺ q) then
 6 |   |   |   S_p ← S_p ∪ {q}
 7 |   |   end
 8 |   |   else if (q ≺ p) then
 9 |   |   |   n_p ← n_p + 1
10 |   |   end
11 |   end
12 |   if (n_p = 0) then
13 |   |   F_1 ← F_1 ∪ {p}
14 |   end
15 end
16 i ← 1
17 while F ≠ ∅ do
18 |   Q ← ∅
19 |   foreach p ∈ F_i do
20 |   |   foreach q ∈ S_p do
21 |   |   |   n_q ← n_q − 1
22 |   |   |   if (n_q = 0) then
23 |   |   |   |   Q ← Q ∪ q
24 |   |   |   end
25 |   |   end
26 |   end
27 |   i ← i + 1
28 |   F_i ← Q
29 end
30 return F
```

4.3.2 Crowding Distance

Crowding distance is an NSGA-II mechanism responsible for preserving the diversity of the obtained non-dominated set of solutions. The crowding distance value of a solution estimates the density of solutions around that solution (Raquel and Naval, 2005).

To estimate the density of solutions around a certain point in the population, we calculate the average distance between its two adjacent points for each objective. The

distance metric for point i estimates the perimeter of the largest cuboid covering this point, not including any other points in the population. Solutions located close to regions with fewer points receive a higher value than those located close to regions with more points in the objective space (Deb et al., 2002).

The method to calculate the value for the crowding distance of each solution is presented by Algorithm 4.4.

Algorithm 4.4: *crowding distance*

input: $\mathcal{I}$
1 $l \leftarrow |\mathcal{I}|$
2 **foreach** $i \in \mathcal{I}$ **do**
3 $\quad \mathcal{I}[i] \leftarrow 0$
4 **end**
5 **foreach** $m \in \mathcal{M}$ **do**
6 $\quad \mathcal{I} \leftarrow \texttt{Sort}(\mathcal{I}, obj)$
7 $\quad \mathcal{I}[1].distance \leftarrow \infty$
8 $\quad \mathcal{I}[l].distance \leftarrow \infty$
9 $\quad$ **for** $i = 2$ to $(l - 1)$ **do**
10 $\quad\quad \mathcal{I}[i].distance \leftarrow \mathcal{I}[i].distance + (\mathcal{I}[i+1].obj - \mathcal{I}[i-1].obj)/(f_{obj}^{\max} - f_{obj}^{\min})$
11 $\quad$ **end**
12 **end**
13 **return** $\mathcal{I}$

Algorithm 4.4 takes as input a set of solutions $\mathcal{I}$. This set has size l, and, initially, we assigned the value zero to the crowding distance of all individuals. Let loop defined between the lines 5 – 12. First, we sort the solutions of the set $\mathcal{I}$ (line 6), considering each objective obj, then we assign infinity to the crowding distance value of the first (line 7) and the last solution of the set $\mathcal{I}$ (line 8). Next, in the loop defined between lines 9 – 11, we calculate the crowding distance for other solutions of the set $\mathcal{I}$ (line 10). The loop defined between the lines (5 – 12) is repeated for each obj objective. The method returns the crowding distance value for each solution of the set $\mathcal{I}$.

4.3.3 Initial Population

The initial population of the NSGA-II contains size_{pop} individuals. Two of them are constructed through a greedy strategy, one of which considers only the objective of minimizing the makespan. In this case, we always choose the operating mode related to the highest speed factor. The other individual considers only the total energy cost. In the second case, we choose the operating mode related to the lowest consumption factor. The other individuals (size_{pop} - 2) of the initial population are randomly generated.

Algorithm 4.5 describes the greedy strategy used to generate individuals to the initial population.

Algorithm 4.5: Greedy Constructive Heuristic

```
   input: N, n, obj
1  s ← ∅
2  for i = 1 to n do
3  |   j ← random job ∈ N
4  |   N ← N \ {j}
5  |   (i_best, pos_best) ← GreedyChoice(s, j, obj)
6  |   s ← Insert(s, j, i_best, pos_best)
7  end
8  return s
```

Algorithm 4.5 starts with an empty initial individual, that is, without any jobs allocated (line 1). The loop between lines 2 and 7 allocates each job $j \in \mathsf{N}$ on the machines. Therefore, we randomly select a job j, which has not yet been allocated (line 3). Then, we identified the best machine i_{best} of the individual s and the best position pos_{best} to insert this job (line 5). In this case, we consider one to each generated solution: minimize the makespan or the total energy cost. Then, we allocate job j in position pos_{best} on machine i_{best} in individual s (line 6). At the end of the procedure, we return a feasible individual s (line 8).

4.3.4 Crossover

We used the binary tournament selection method to choose each pair of individuals for the crossover operator. We run two tournaments with two individuals each and select the winner of each tournament for the crossover. In our approach, the dominant individual wins the tournament. If both individuals are non-dominated, we randomly choose an objective and use it to define the winning individual.

Figure 4.2 illustrates the crossover between two individuals. Note the following associations between the reproduction process and the modeling of the problem addressed: an individual is associated with a problem solution, and a gene is related to a job in scheduling.

After selecting two individuals named parent 1 and parent 2, respectively, we applied the crossover operator to generate new individuals. We adopted the One Point Order Crossover operator from Vallada and Ruiz (2011) adapted to the parallel machine problem. We describe its operation below:

1. We define, at random, the crossover points of each machine, as shown in Figure 4.2(a);
2. We generate two offspring. The first receives the genes to the left of the crossover point defined on each machine of parent 1. The second gets the genes to the right, as shown in Figure 4.2(b);
3. We mark in parent 2 the genes present in each offspring, as shown in Figure 4.2(c);
4. We add the unmarked genes of parent 2 to offspring 1 and 2. We add these genes in the position that results in the lowest value for the objective function, whereas this problem has two objective functions, so we randomly select one at each crossover. In the end, we will have two new individuals, as shown in Figure 4.2(d).

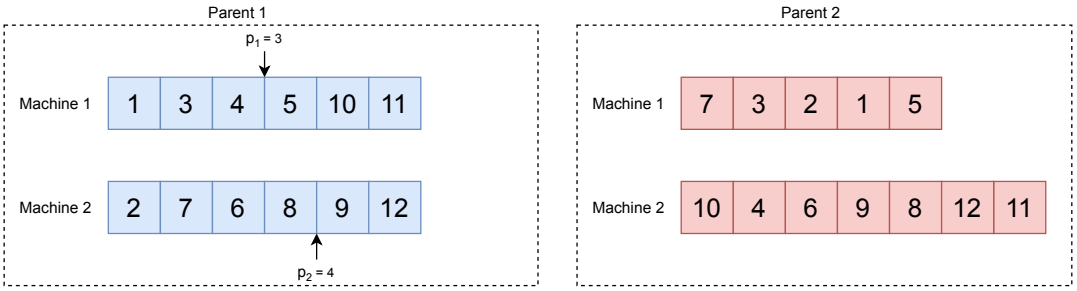
The offspring always inherit the parent's operating modes.

We repeat this procedure until to generate size_{pop} new individuals.

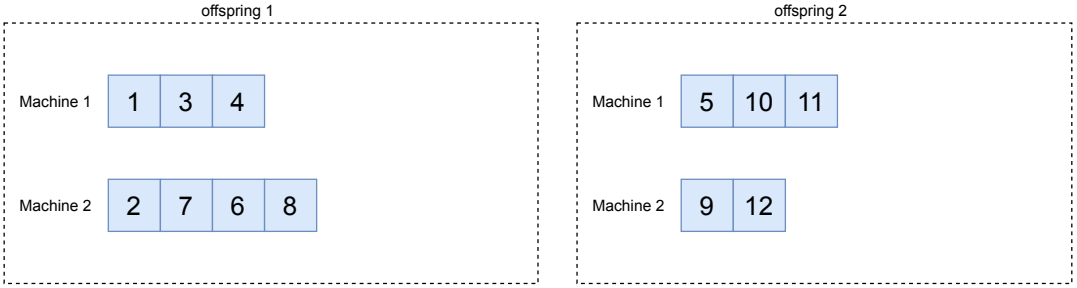
4.3.5 Mutation

We implemented three mutation operators (Swap, Insert, and Operating mode change), described below. These operators maintain the population's genetic diversity and reduce the chances of the algorithm getting stuck at a local optimum.

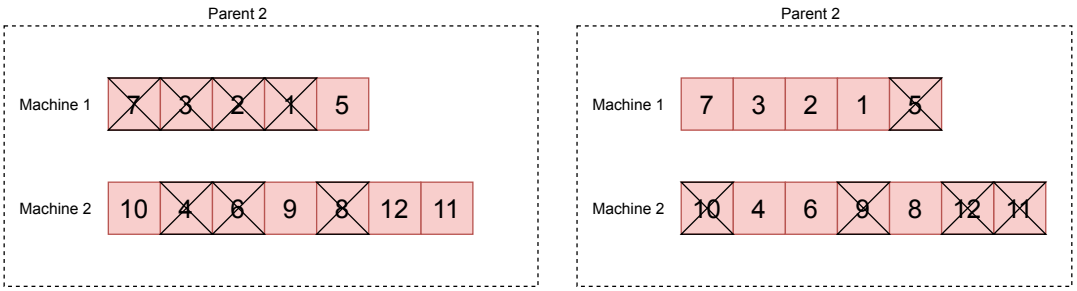
Each individual in a given population has a probability of $prob_{mut}$ of getting mutated. The mutation consists of applying an operator to a select individual. We chose the



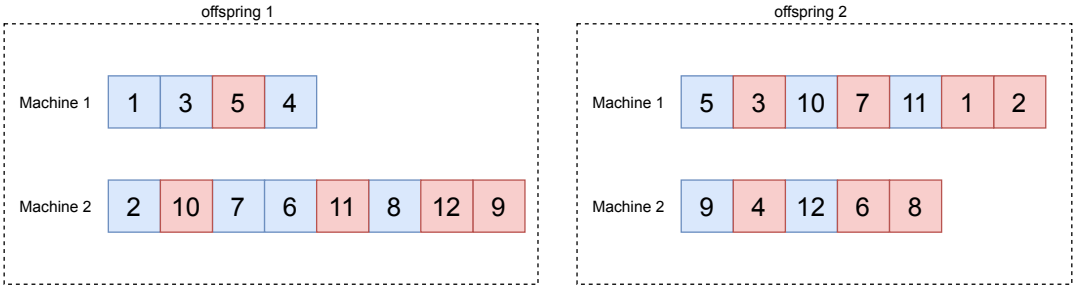
(a) Selecting parents and crossover points



(b) Copy part of genes from parent 1 to each offspring



(c) Mark in parent 2 the genes present in each offspring



(d) Complete the genes of each offspring with the genes of parent 2

Figure 4.2: Crossover adapted from Vallada and Ruiz (2011)

operator to be applied randomly. In this work, we propose the following mutation operators:

4.3.5.1 Swap

The swap operator works by randomly choosing a job j_1, initially allocated in position a on machine i_1 and another job j_2 allocated in position b on machine i_2. Then, we allocate job j_1 in position b on machine i_2. Further, we allocate job j_2 in position a on machine i_1.

Figure 4.3 illustrates the swap between two jobs, j_1 and j_2. They are initially allocated on machines i_1 and i_2, respectively. After swapping, we allocate job j_2 on machine i_1 and job j_1 on machine i_2.

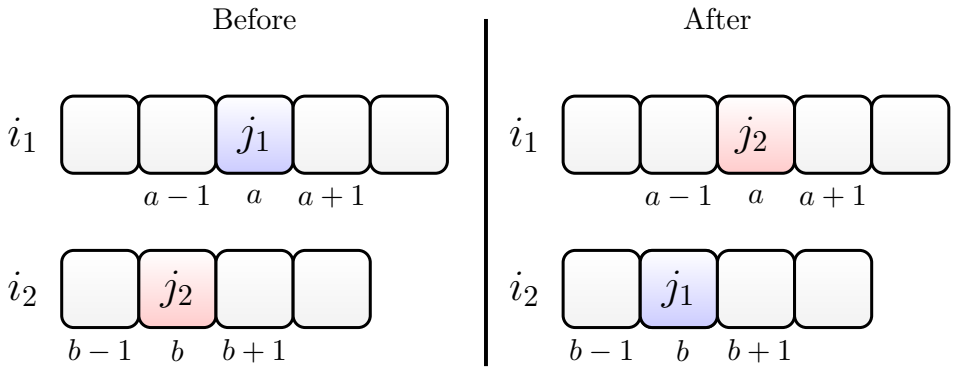


Figure 4.3: Swap move between jobs j_1 and j_2

4.3.5.2 Insertion

The insertion operator consists of randomly choosing a job j_1 allocated at position a on machine i_1 and randomly choosing position b of another machine i_2. Then job j_1 is removed from machine i_1 and inserted into position b on machine i_2.

Figure 4.4 illustrates this operator. The left side shows the scheduling before, and the right side shows it after the insertion.

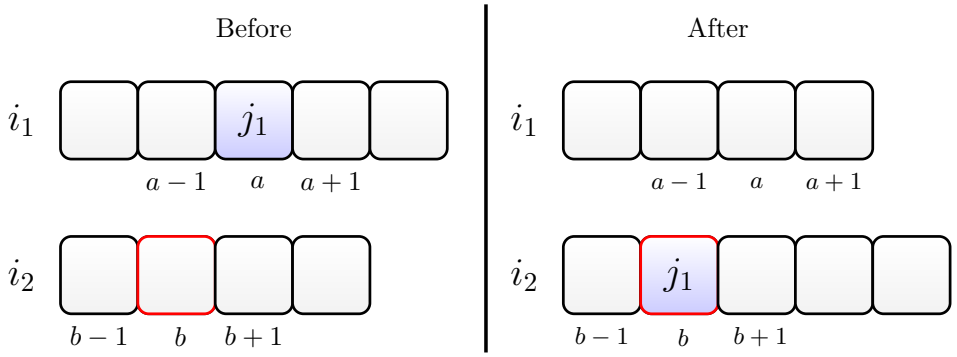
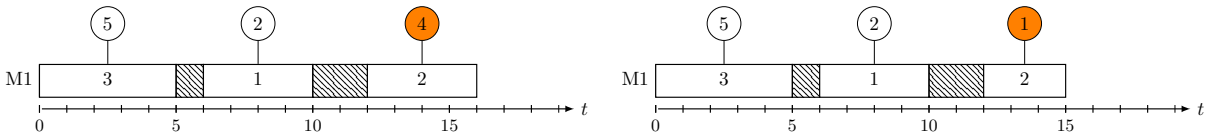


Figure 4.4: Insertion operator of job j_1 on machine i_2

4.3.5.3 Operating mode change

In the operating mode change operator, we randomly select a job and change its operating mode at random.

Figure 4.5 illustrates the application of this operator in a scheduling which involves 3 jobs. As can be seen, job 2, which is in the third position on machine 1, has operating mode 4. After the application of this operator, the job changes to operating mode 1.



(a) Before (b) After

Figure 4.5: Example of the mode change operator

The algorithms NSGA-I and NSGA-II were implemented by performing a mutation with a probability equal to $prob_{mut}$.

4.4 MOVNS

The MOVNS was developed by Geiger (2008). It is a multi-objective local search algorithm based on the VNS algorithm (Mladenović and Hansen, 1997). According to Hansen and Mladenović (2001), the VNS is a metaheuristic simple and effective to com-

binatorial problems. Its main idea is to apply systematic changes of neighborhoods both in a descent phase, to find a local minimum, and in a perturbation phase to escape from the corresponding valley.

We show the basic steps of VNS through Algorithm 4.6:

Algorithm 4.6: Basic VNS

```
   input: 𝒩, stopping_criterion
1  s ← generate an initial solution
2  while stopping_criterion not satisfied do
3  |  for k = 1 to max_n do
4  |  |  s' ← Shaking(s, 𝒩_k)
5  |  |  s'' ← Local search(s, s')
6  |  |  if f(s'') < f(s) then
7  |  |  |  s ← s''
8  |  |  |  k ← 1
9  |  |  end
10 |  |  else
11 |  |  |  k ← k + 1
12 |  |  end
13 |  end
14 end
15 return s
```

We adapted two versions of the multi-objective local search algorithm MOVNS for the problem addressed in this work. The first is the basic version with five neighborhood structures, here named MOVNS1, and the second adds the intensification procedure of Arroyo et al. (2011) to MOVNS, here called MOVNS2. We present the pseudocode of the MOVNS2 in Algorithm 4.7.

The MOVNS2 receives as input: the stopping criterion, the neighborhood structures ($\mathcal{N} = \mathcal{N}_1, \ldots \mathcal{N}_5$), the initial size of the solution set (size_{set}), the destruction size n_r, and the shaking level (shake_{level}). It returns the non-dominated set obtained (NDS). Initially, in line 1 of Algorithm 4.7, we use the method of Scheffé (1958) to create a vector with size_{set} weights. In line 2, we generate an initial solution to each weight W_i, using the constructive method of Subsection 4.4.1. This set is composed of size_{set} solutions. We repeat the steps defined between lines 3 – 14 as long as the stopping criterion is not satisfied. In lines 4 and 5, we select, randomly, a solution s of the NDS, apply the shaking procedure defined in Subsection 4.4.3, and generate a neighbor solution s′. We

Algorithm 4.7: MOVNS2

Input: stopping_criterion, $\mathcal{N}$, size_{set}, n_r, shake_{level}

1 $\mathsf{W} \leftarrow$ generate a vector with size_{set} weights by the Scheffé (1958) method
2 $\mathsf{NDS} \leftarrow$ generate an initial solution for each weight W_i
3 **while** stopping_criterion not satisfied **do**
4 $\mathsf{s} \leftarrow$ random solution from NDS
5 $\mathsf{s}' \leftarrow \mathtt{Shaking}(\mathsf{s}, \mathsf{shake}_{level})$
6 **for** $k = 1$ **to** 5 **do**
7 **foreach** $\mathsf{s}'' \in \mathcal{N}_k(\mathsf{s}')$ **do**
8 $\mathsf{NDS} \leftarrow \mathtt{AddSolution}(\mathsf{s}'')$
9 **end**
10 **end**
11 $\mathsf{s} \leftarrow$ random solution from NDS
12 $\mathsf{NDS}' \leftarrow \mathtt{Intensification}(\mathsf{s}, \mathsf{n}_r)$
13 $\mathsf{NDS} \leftarrow$ non-dominated solutions obtained from $\mathsf{NDS} \cup \mathsf{NDS}'$
14 **end**
15 **return** NDS

explore the neighborhood of s' considering each neighborhood structure k (lines 6 – 10). We update the NDS with each neighbor solution of s' (line 8). Next, we apply the intensification procedure described in Subsection 4.4.4 in a random solution of NDS (line 12). In the end, the MOVNS2 Algorithm returns the non-dominated set obtained.

Next, we detail the initial solution (Subsection 4.4.1), the neighborhood structures (Subsection 4.4.2), the shaking (Subsection 4.4.3) and intensification procedures (Subsection 4.4.4).

4.4.1 Initial solution

We present in Algorithm 4.8 the strategy for generating each solution of the initial set.

Algorithm 4.8 gets as input the weight W_i for the objective function. It starts with an empty initial solution (line 1). Then, the loop between lines 2 and 7 allocates each job j on the machines. For this, we randomly select a job j not yet assigned (line 3). Then, we choose the best machine i_{best}, position pos_{best}, and operating mode l_{best} to insert job j in solution s (line 5). To make this choice, we consider the weighted sum of the objectives of the problem with weight W_i. Then, we allocate job j in position

Algorithm 4.8: Greedy Constructive Heuristic Weighted

```
   input: W_i
1  s ← ∅
2  for k = 1 to n do
3  |  j ← random job ∈ N
4  |  N ← N \ {j}
5  |  (i_best, pos_best, l_best) ← GreedyChoiceWeighted(s, j, W_i)
6  |  s ← Insert(s, j, i_best, pos_best, l_best)
7  end
8  return s
```

pos_{best} on machine i_{best} with operating mode l_{best} in solution s (line 6). At the end of this procedure, we return a feasible solution s (line 8).

4.4.2 Neighborhood Structures

We describe below the five neighborhood structures used to explore the solution space of the problem:

Swap on the same machine ($\mathcal{N}_1$): In this operator, we select two jobs, j_1 and j_2, allocated, respectively, in positions x and y on machine i, and reallocate job j_1 in position y and job j_2 in position x on the same machine i.

Swap between different machines ($\mathcal{N}_2$): This structure consists of selecting a job j_1 allocated on machine i_1 in position x and another job j_2 that is in position y on machine i_2. Then, we allocate job j_1 in position y on machine i_2, and we allocate job j_2 on machine i_1 in position x.

Insertion on the same machine ($\mathcal{N}_3$): It starts selecting job j_1 that is initially in position x on machine i. Then, we choose another position y on the same machine. Finally, we remove job j_1 from the initial position and reinsert it in position y on machine i.

Insertion between different machines ($\mathcal{N}_4$): In this structure, we select job j_1 allocated in position x on machine i_1 and select position y on machine i_2. Then, remove job j_1 and insert it in position y on machine i_2.

Change operating mode ($\mathcal{N}_5$): This neighborhood structure selects job j on machine i and then changes its operating mode.

4.4.3 Shaking procedure

The shaking procedure is an important phase of a VNS-based algorithm. According to Hansen et al. (2017), the purpose of this procedure, when used within a VNS heuristic, is to avoid getting stuck in a local minimum. The simple shaking procedure consists in selecting a random solution from the k-th neighborhood structure of solution s. In the shaking procedure of this work, we apply shake_{level} moves chosen among those described in Subsection 4.4.2 to the current solution.

4.4.4 Intensification of Arroyo et al. (2011)

We present, in Algorithm 4.9, the intensification procedure of the MOVNS Algorithm by Arroyo et al. (2011). It was mentioned in line 12 of Algorithm 4.7.

The input of this procedure is the solution s and the destruction size n_r. The output

is the non-dominated set obtained.

Algorithm 4.9: Intensification of Arroyo et al. (2011)

Input: n_r, s

1 $\mathsf{s}_p \leftarrow \mathsf{s}$
2 $\mathsf{s}_r \leftarrow \emptyset$
3 **for** $k \leftarrow 1$ **to** n_r **do**
4 | Remove random job j from the solution s_p
5 | Insert job j into s_r
6 **end**
7 NDS' $\leftarrow \mathsf{s}_p$
8 **foreach** *job* $\in \mathsf{s}_r$ **do**
9 | NDS'' $\leftarrow \emptyset$
10 | **foreach** solution $\mathsf{s}_p{}' \in$ NDS' **do**
11 | | Insert the job in all positions of $\mathsf{s}_p{}'$
12 | | Evaluate each partial solution resulting $\mathsf{s}_p{}''$
13 | | NDS'' $\leftarrow$ non-dominated solutions obtained of NDS'' $\cup \{\mathsf{s}_p{}''\}$
14 | **end**
15 | NDS' $\leftarrow$ NDS''
16 **end**
17 **return** NDS'

First, in Algorithm 4.9, we initialize the partial solution (s_p) from the current solution s and initialize as empty the set of removed jobs (s_r), in lines 1 and 2, respectively.

Then, we perform the destruction phase (lines 3 – 6), in which we randomly remove n_r jobs from the partial solution s_p. Then we insert the jobs removed from the s_p in the set of removed jobs s_r and keep the remaining $\mathsf{n} - \mathsf{n}_r$ jobs in s_p. At the end, we have the partial solution s_p and set of removed jobs s_r generated in the destruction phase.

In line 7, we initialize with s_p the non-dominated set of partial solutions NDS'.

Then, between lines 8-16, we insert each job of s_r in all positions of the partial solution s_p. To each new partial solution s_p'' generated, we update NDS'' (line 13).

At the end (line 15), we have the non-dominated set NDS' with feasible solutions.

Chapter 5

Computational Experiments

This section is organized as follows. Subsections 5.1 and 5.2 describe the instances and the metrics used to assess the quality of the set of non-dominated solutions generated by the algorithms. Subsection 5.3 shows the parameter calibration of the algorithms. Subsection 5.4 reports the results.

We coded the algorithms in the C++ language and implemented the mathematical model with the Gurobi 7.0.2 API (Gurobi Optimization, 2020). We performed the tests on a microcomputer with the following configurations: Intel (R) Core (TM) i7-4510U processor with a clock frequency of 2 GHz, 16 GB of RAM, and 64-bits Ubuntu 19.10 operating system.

Furthermore, we compared the performance of the NSGA-II and MOVNS2 algorithm with two multi-objective algorithms: MOVNS1 of Geiger (2008) and NSGA-I of Srinivas and Deb (1994). The NSGA-I uses the same crossover operators, mutation operators, stopping criterion, and initial population of the NSGA-II. In turn, the MOVNS1 uses the same neighborhood structures, stopping criterion, initial solution of the MOVNS2 algorithm.

5.1 Instances Generation

Since, as far as we know, there is no set of instances in the literature for the problem addressed, we adapted two instance sets from the literature that deal with similar problems. The first one, called set1, is a subset of the small instances of Cota et al. (2018)

satisfying the triangular inequality, in which we add information about the energy price on-peak and off-peak hours. The second set, named set2, is also a subset of the large instances of Cota et al. (2018), in which we included instances of 750 jobs. Table 5.1 shows the characteristics of these sets of instances, which are available in Rego et al. (2021).

Table 5.1: Instance characteristics

Parameter	set1	set2	Based on
n	6, 7, 8, 9, 10	50, 250, 750	Vallada and Ruiz (2011), Cota et al. (2018)
m	2	10, 20	Vallada and Ruiz (2011), Cota et al. (2018)
o	3	5	Mansouri et al. (2016), Ahilan et al. (2013), Cota et al. (2018)
P_{ij}	$U[1, 99]$	$U[1, 99]$	Vallada and Ruiz (2011), Cota et al. (2018)
S_{ijk}	$U[1, 9]$	$U[1, 9]$, $U[1, 124]$	Vallada and Ruiz (2011), Cota et al. (2018)
π_i	$U[40, 200]$	$U[40, 200]$	Cota et al. (2018)
V_l	1.2, 1, 0.8	1.2, 1.1, 1, 0.9, 0.8	Mansouri et al. (2016), Ahilan et al. (2013), Cota et al. (2018)
λ_l	1.5, 1, 0.6	1.5, 1.25, 1, 0.8, 0.6	Mansouri et al. (2016), Ahilan et al. (2013), Cota et al. (2018)

5.2 Metric description

The comparison of different multi-objective optimization algorithms involves choosing the aspects to be evaluated. Zitzler et al. (2000) suggest three aspects that can be identified and measured: convergence, extension, and distribution. Convergence refers to the proximity of this set to the Pareto-optimal front or to the reference set. In turn, the extension assesses the breadth of the region covered by this set of non-dominated solutions. The distribution refers to the uniformity of the spacing between the solutions within the set. Some authors consider that uniformity and diversity form a single aspect

called diversity (Yan et al., 2007). In the present work, we will use convergence and diversity.

The hypervolume metric can evaluate both convergence and diversity (Shang et al., 2020), and the HCC metric, in turn, considers only diversity (Guimarães et al., 2009).

5.2.1 Hypervolume

The hypervolume or S metric is a measure of quality often used to compare results from multi-objective algorithms and it was proposed by Zitzler and Thiele (1998). This metric provides a combined estimate of convergence and diversity of a set of solutions (Deb, 2014). The hypervolume of a non-dominated set measures the area covered or dominated by this set's points, limited by a Reference Point (RP). In maximization problems, it is common to use the point (0; 0), while in minimization problems, an upper bound, also known as the Nadir point, is used to limit this area. In Figure 5.1, the shaded area defines the hypervolume of the set of non-dominated solutions $\mathcal{A}$ for a problem with two objective functions. The point $(max_x; max_y)$ defines the upper limit. We denote by $HV(\mathcal{A})$ the hypervolume of a set of non-dominated solutions $\mathcal{A}$ relative to a reference point (Deb, 2014).

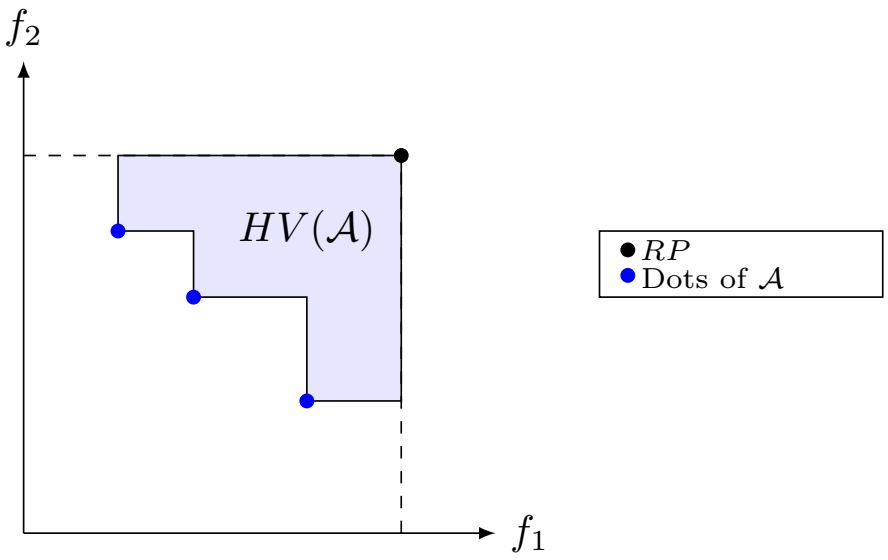


Figure 5.1: Hypervolume for set $\mathcal{A}$

5.2.2 HCC

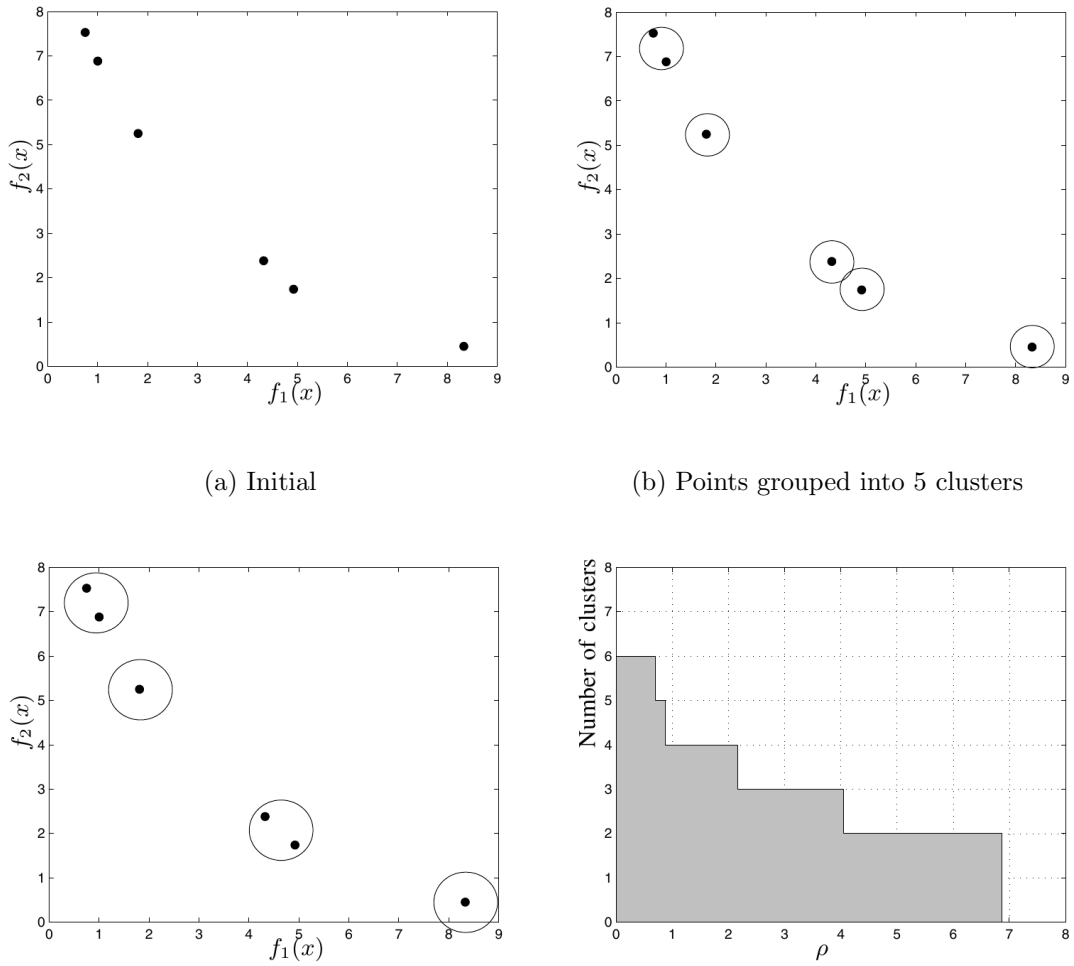
Hierarchical Cluster Counting (HCC) is a metric proposed by Guimarães et al. (2009) to evaluate the quality of non-dominated sets obtained by multi-objective optimization algorithms. It is based on hierarchical clustering techniques, such as the Sphere Counting (SC) (Wanner et al., 2006) metric. According to Guimarães et al. (2009), the diversity of a non-dominated set is directly proportional to the HCC value calculated for it.

We calculate the HCC for a set of points $\mathcal{A}$ as follows:

1. Initially, we create a grouping for each point in the set and consider that each group created is a sphere of radius equal to zero;
2. Then, we calculate the minimum distances of fusion, which is a new assumed value for the radius of the spheres capable of decreasing the number of clusters;
3. We group the points into the same cluster;
4. We repeat steps 2 and 3 until all the points belong to the same grouping;
5. We obtain the HCC value by adding, in each iteration, the product between the distances of fusion and the amount of grouping formed.

Consider Figure 5.2, which illustrates the steps to calculate the HCC for a six-point non-dominated set. Figure 5.2(a) shows the first cluster in which each point is in a different sphere with radius zero. Figure 5.2(b) shows the points grouped into five spheres, each with radius r_1. Figure 5.2(c) shows the points grouped into four spheres, each with radius r_2. Figure 5.2(d) shows, in the Cartesian plane, the relationship between the number of clusters and the radius of each cluster. The gray region area represents the value of the HCC metric for the set shown in Figure 5.2.

We present, in Figure 5.2, some steps to exemplify the calculation of this metric. In Figure 5.2(a), we show a Pareto front composed of six points, and each point represents one cluster, with $\rho = 0$. In Figure 5.2(b), we can see five clusters considering $\rho \approx 0.7$. In



(a) Initial

(b) Points grouped into 5 clusters

(c) Points grouped into 4 clusters

(d) Number of clusters by ρ size

Figure 5.2: Example of how to calculate the HCC metric (Guimarães et al., 2009)

Figure 5.2(c), we have four clusters with $\rho \approx 0.9$. The calculation ends when all points are in a single cluster. Finally, Figure 5.2(d) shows a graph with cluster numbers formed according to ρ. Note that as the value of ρ increases, the number of clusters decreases.

5.3 Tuning of algorithms' parameters

The parameter values used in the algorithms can affect their performance. Therefore, we use the Irace package (López-Ibáñez et al., 2016) to find the best values for these parameters. Irace is a software encoded in the R language that automatically performs an iterative procedure to find the appropriate optimization algorithm settings.

The parameters were tested in the following instances: 6_2_1439_3_S_1-9, 10_2_1439_3_S_1-9, 50_10_1439_5_S_1-9, 250_10_1439_5_S_1-124 and 750_20_1439_5_S_1-9.

Table 5.2 shows the test scenarios used. In the first column, we present the algorithm name; in the second column, the description of each parameter; in the third column, the set of values tested for each parameter; and in the fourth column, the best value returned by Irace.

Table 5.2: Test scenarios for algorithms' parameters

Method	Description	Tested values	Irace best value
NSGA-II	Population size ($size_{pop}$)	80, 90, 100, 110	110
	Probability of mutation	0.04, 0.05, 0.06, 0.07	0.05
NSGA-I	Population size ($size_{pop}$)	80, 90, 100, 110	80
	Probability of mutation	0.04, 0.05, 0.06, 0.07	0.06
MOVNS2	Initial set size ($size_{set}$)	20, 25, 30, 35	25
	Perturbation level	2, 4, 6, 8	6
	Destruction level	2, 4, 6, 8	6
MOVNS1	Initial set size ($size_{set}$)	25, 30, 35, 40	30
	Perturbation level	2, 4, 6, 8	4

5.4 Results

In this section, we presented the results of two experiments used to evaluate the performance of proposed algorithms. First, we present the MOVNS2, NSGA-II, and weighted sum method results in instances with up to 10 jobs and 2 machines (set1). Then, we report the results of NSGA-II and MOVNS2 algorithms in larger instances, with up to 750 jobs and 20 machines (set2), and compare with others literature algorithms. In both cases, we executed the algorithms 30 times in each instance. The time limit for each execution of the metaheuristics was defined as $time_{limit} = n \times ln(m)$ seconds, where m and n are the number of machines and jobs, respectively.

We used the Relative Percentage Deviation (RPD_i^{HV}) to evaluate the HV metric for each method Alg and instance i. It is calculated by Equation (5.1):

$$RPD_i^{HV}(Alg) = \frac{HV_i^{RS} - HV_i^{v}}{HV_i^{RS}}, \tag{5.1}$$

where HV_i^{RS} is the hypervolume value of the reference set in 30 executions of the algorithm Alg in the instance i. v can assume three values: min, max, and avg, representing, respectively, the smallest, the largest, and the average of the hypervolume in 30 executions of the algorithm in the instance i.

Since we not known the optimal Pareto front, we define a reference set to each instance compose by all non-dominated solutions obtained by the all tested algorithms. The reference set is also known as Pareto front (Arroyo et al., 2011).

5.4.1 Results in the set1

In this subsection, we reported the results of the algorithms NSGA-II, MOVNS2, and the weighted sum method in the set of instances set1.

Table 5.3 shows the reference set data for these instances. In this table, the first two columns display the instance identifier and name, respectively. The next two columns present the number of jobs and machines, respectively. The fifth column shows the hypervolume of this reference set. Finally, the last column presents the reference point ($C_{\max}$; TEC) used to calculate the hypervolume of each instance.

Table 5.3: Summary of reference set data in the set1

# ID	# Instance	n	m	HV	RP
1	6_2_1439_3_S_1-9	6	2	8,795.00	(250; 259.82)
2	7_2_1439_3_S_1-9	7	2	15,918.00	(400; 260.68)
3	8_2_1439_3_S_1-9	8	2	4,825.00	(260; 321.17)
4	9_2_1439_3_S_1-9	9	2	31,080.00	(450; 389.38)
5	10_2_1439_3_S_1-9	10	2	35,448.00	(500; 382.46)

Tables 5.4 and 5.5 present the method results concerning the RPD^{HV} and the HCC metrics. In these tables, the first column identifies the instance. The second and third columns show the upper bound (UB) and the time, in seconds, of the exact method. The next three columns display the minimum, maximum and average values, respectively, concerning the MOVNS2 method. The seventh column presents the standard deviation of the MOVNS2 results. The next three columns show the minimum, maximum and average values, respectively, concerning the NSGA-II method. The 11th column displays the standard deviation of the NSGA-II results. Finally, the 12th column presents the time, in seconds, of the NSGA-II and MOVNS2 algorithms.

Table 5.4: Summary of RPD^{HV} and runtime of the proposed methods in the set1. The best average values are highlighted in bold.

	Exact		MOVNS2				NSGA-II				
# ID	UB (%)	time (s)	min (%)	max (%)	*avg* (%)	*sd*	min (%)	max (%)	*avg* (%)	*sd*	time (s)
1	0.01	172.09	0.00	0.00	**0.00**	0.00	0.00	0.03	0.00	0.01	4.16
2	0.01	549.86	0.00	0.00	**0.00**	0.00	0.00	0.02	0.00	0.01	4.85
3	0.00	2,140.40	0.00	0.00	**0.00**	0.00	0.00	0.21	0.02	0.04	5.54
4	0.03	8,312.92	0.00	0.00	**0.00**	0.00	0.00	0.05	0.02	0.02	6.24
5	0.03	39,396.63	0.00	0.00	**0.00**	0.00	0.00	0.07	0.02	0.02	6.93

We can see in Table 5.4 that in the set of instances set1, the RPD^{HV} of the MOVNS2 algorithm is lower or equal in all cases (min, max e *avg*) compared to the NSGA-II and exact method. We can also verify that the standard deviation of the MOVNS2 algorithm in all instances is equal to zero. In other words, in these instances, all MOVNS2 executions obtained the same set non-dominated. Moreover, the execution time of the NSGA-II and the MOVNS2 is much less than that of the exact algorithm.

Table 5.5: Summary of HCC value and runtime of proposed the methods in the set1. The best average values are highlighted in bold.

# ID	Exact		MOVNS2				NSGA-II				time (s)
	UB	time (s)	min	max	*avg*	*sd*	min	max	*avg*	*sd*	
1	209.32	172.09	239.72	239.72	239.72	0.00	223.81	274.13	**241.52**	8.49	4.16
2	428.11	549.86	525.03	525.03	525.03	0.00	513.94	535.51	**525.33**	3.43	4.85
3	142.59	2,140.40	206.10	206.10	**206.10**	0.00	135.43	242.25	194.88	23.14	5.54
4	359.65	8,312.92	575.89	575.89	**575.89**	0.00	408.26	707.81	556.11	71.95	6.24
5	708.26	39,396.63	848.28	848.28	**848.28**	0.00	474.27	869.75	721.59	98.26	6.93

Concerning Table 5.5, we noted that the NSGA-II algorithm performed better than others in two comparisons (ID 1 and 2). If we consider only the instances with more than seven jobs (ID 3, 4, and 5), the MOVNS2 was superior in diversity metric.

Figure 5.3(a) and 5.3(b) presents the non-dominated sets obtained by NSGA-II, MOVNS2, and the exact method in two randomly selected instances. The first instance has 6 jobs and 2 machines, and the second has 10 jobs and 2 machines. In this figure, the fill green circles, the blue "x", and the red circles represent the solutions of the exact method, the MOVNS2, and NSGA-II, respectively. The x-axis represents the makespan, and the y-axis represents the total energy cost.

We can notice in Figure 5.3(a) that the MOVNS2 and NSGA-II non-dominated set contain all the solutions found by the exact method, plus two additional solutions. In this example, the three methods have the same amplitude, and the MOVNS2 and NSGA-II were able to find a set of solutions with higher cardinality. On the other hand, Figure 5.3(b) shows that MOVNS2, NSGA-II, and the exact method found 16, 14, and 8 non-dominated solutions, respectively. The MOVNS2 solutions dominate two of the NSGA-II solutions. In this example, the MOVNS2 and the exact method showed better amplitude than the NSGA-II, but the MOVNS2 obtained higher cardinality.

Considering these results, we observed that the MOVNS2 and NSGA-II find good quality solutions and require less computational time than the exact method.

5.4.2 Results in the set2

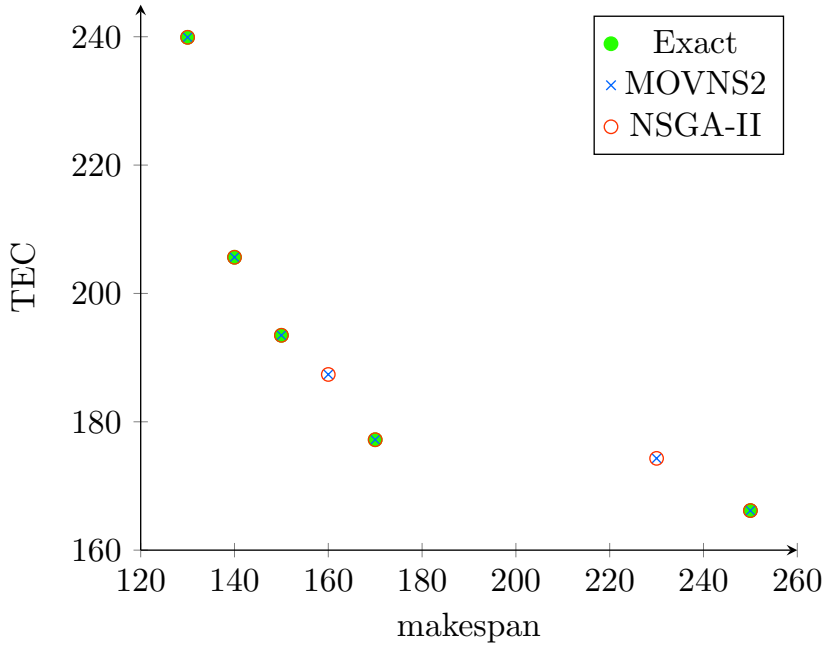
Here, we presented the results of the NSGA-II and MOVNS2 algorithms in the set of instances set2.

Table 5.6 shows the reference set data for the instances of set2. Its organization follows the same description as the previous section's tables.

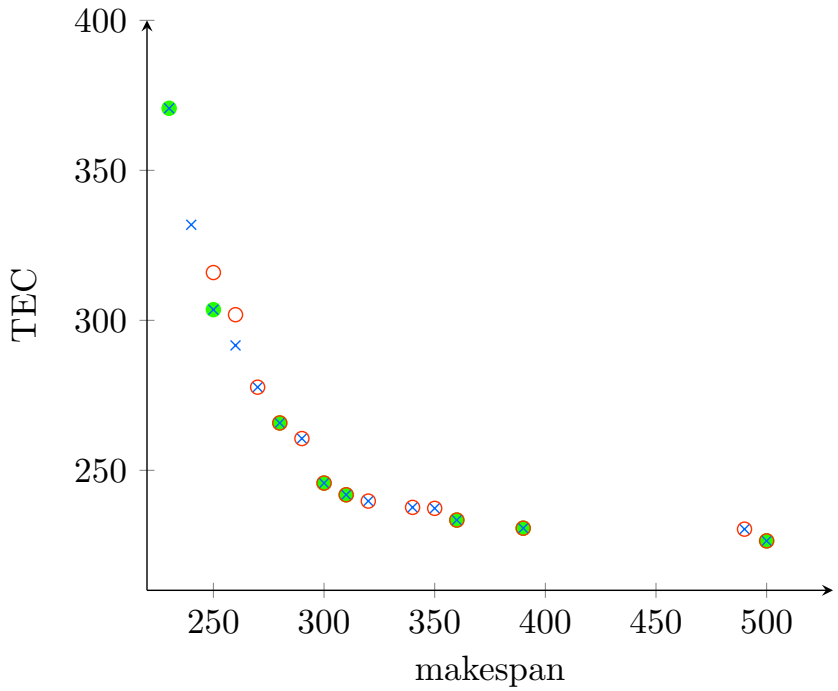
Tables 5.7 and 5.8 report the RPD^{HV} and HCC metric values, respectively, to the proposed algorithms in the set of instances set2.

The results in Table 5.7 indicate that, in the mean, the MOVNS2 algorithm performs better than the NSGA-II, in all instances considering the RPD^{HV} metric. The low standard deviation values presented by MOVNS2 indicate that it is relatively stable.

The results in Table 5.8 indicate that, in the mean, the MOVNS2 algorithm per-



(a) Instance with 6 jobs and 2 machines



(b) Instance with 10 jobs and 2 machines

Figure 5.3: Example of Fronts found by NSGA-II, MOVNS2 and Exact methods

Table 5.6: Summary of reference set data in the set2

# ID	# Instance	n	m	HV	RP
1	50_10_1439_5_S_1-9	50	10	65,171.00	(289; 452.65)
2	50_10_1439_5_S_1-124	50	10	257,472.00	(539; 909.56)
3	50_20_1439_5_S_1-9	50	20	20,554.00	(121; 323.34)
4	50_20_1439_5_S_1-124	50	20	212,473.00	(474; 642.57)
5	250_10_1439_5_S_1-9	250	10	1,603,978.00	(1457; 2245.16)
6	250_10_1439_5_S_1-124	250	10	8,501,314.00	(5049; 2930.02)
7	250_20_1439_5_S_1-9	250	20	821,506.00	(525; 2570.86)
8	250_20_1439_5_S_1-124	250	20	6,536,232.00	(2919; 3179.80)
9	750_10_1439_5_S_1-9	750	10	8,500,037.00	(3758; 5630.80)
10	750_10_1439_5_S_1-124	750	10	111,007,003.00	(19483; 9442.33)
11	750_20_1439_5_S_1-9	750	20	4,992,770.00	(1556; 5573.70)
12	750_20_1439_5_S_1-124	750	20	37,616,516.00	(7847; 7065.71)

Table 5.7: Summary of RPD^{HV} and runtime to the instances of set2 in the proposed algorithms. The best average values are highlighted in bold.

# ID	MOVNS2				NSGA-II				time (s)
	min (%)	max (%)	*avg* (%)	*sd*	min (%)	max (%)	*avg* (%)	*sd*	
1	0.01	0.05	**0.03**	0.01	0.10	0.24	0.17	0.03	115.13
2	0.04	0.09	**0.07**	0.01	0.07	0.19	0.12	0.03	115.13
3	0.01	0.06	**0.03**	0.01	0.19	0.31	0.24	0.04	149.79
4	0.01	0.04	**0.02**	0.01	0.09	0.19	0.14	0.02	149.79
5	0.01	0.02	**0.01**	0.00	0.08	0.11	0.10	0.01	575.65
6	0.02	0.03	**0.02**	0.00	0.02	0.05	0.03	0.00	575.65
7	0.01	0.03	**0.02**	0.00	0.11	0.16	0.14	0.01	748.93
8	0.01	0.02	**0.01**	0.00	0.04	0.05	0.04	0.00	748.93
9	0.01	0.03	**0.02**	0.00	0.07	0.08	0.08	0.00	1,726.94
10	0.01	0.03	**0.02**	0.00	0.05	0.06	0.05	0.00	1,726.94
11	0.01	0.01	**0.01**	0.00	0.09	0.12	0.10	0.01	2,246.80
12	0.01	0.02	**0.01**	0.00	0.06	0.07	0.07	0.00	2,246.80

forms better than the NSGA-II in three instances (ID 2, 6, and 7) for the HCC metric. Moreover, the NSGA-II is superior in nine instances for this metric.

Table 5.8: Summary of HCC and runtime to the instances of set2 in the proposed algorithms. The best average values are highlighted in bold.

# ID	MOVNS2				NSGA-II				time (s)
	min	max	*avg*	*sd*	min	max	*avg*	*sd*	
1	1,012.20	1,477.80	1,151.40	98.19	1,253.90	1,715.60	**1,446.94**	128.44	115.13
2	1,113.80	2,286.20	**1,674.68**	303.65	1,063.80	2,224.00	1,653.94	323.72	115.13
3	336.57	477.62	401.53	38.68	279.32	659.35	**431.34**	89.45	149.79
4	1,288.60	1,733.70	1,437.69	117.01	1,325.60	2,530.40	**1,861.56**	276.50	149.79
5	5,292.00	7,950.80	6,474.27	585.46	8,268.30	10,702.62	**9,187.19**	535.43	575.65
6	8,755.10	15,637.53	**11,600.85**	1,901.10	9,866.60	12,470.58	11,042.90	669.22	575.65
7	2,157.50	3,135.60	**2,767.26**	203.35	1,878.40	4,193.80	2,556.17	514.60	748.93
8	4,543.90	11,054.71	7,745.17	1,712.55	7,360.40	10,702.12	**8,790.46**	872.03	748.93
9	8,541.80	13,796.15	10,525.14	1,820.08	20,218.49	23,418.80	**21,967.44**	858.86	1,726.94
10	24,361.20	66,683.14	43,214.28	12,745.63	53,824.74	75,091.86	**60,472.87**	6,184.92	1,726.94
11	4,312.80	6,002.60	5,302.41	393.51	10,215.91	15,284.62	**12,925.00**	1,473.66	2,246.80
12	14,984.12	26,785.21	20,141.95	4,116.04	26,060.88	33,175.70	**29,703.90**	1,663.63	2,246.80

We report, in Tables 5.9 and 5.10, the mean values of RPD^{HV} and HCC metrics, respectively, of the NSGA-I, NSGA-II, MOVNS1, and MOVNS2 algorithms in the set of instances set2.

Table 5.9: Average RPD^{HV} to the instances of set2 in the tested algorithms. The best values are highlighted in bold.

#ID	MOVNS1 (%)	MOVNS2 (%)	NSGA-I (%)	NSGA-II (%)
1	**0.02**	0.03	0.18	0.17
2	0.08	**0.07**	0.12	0.12
3	**0.03**	**0.03**	0.26	0.24
4	0.04	**0.02**	0.14	0.14
5	0.02	**0.01**	0.13	0.10
6	0.03	**0.02**	0.04	0.03
7	**0.02**	**0.02**	0.17	0.14
8	0.02	**0.01**	0.05	0.04
9	**0.02**	**0.02**	0.10	0.08
10	**0.02**	**0.02**	0.06	0.05
11	**0.01**	**0.01**	0.16	0.10
12	**0.01**	**0.01**	0.07	0.07
Mean	0.03	**0.02**	0.12	0.11

As shown in Table 5.9, the MOVNS2 achieved the best average results regarding hypervolume in eleven instances. In the mean, the MOVNS1 was superior in seven instances. Both obtained the best result in six instances. The MOVNS2 achieved the best results for this metric.

On the other hand, in Table 5.10, the NSGA-II found the best results in seven instances of set2 in the HCC metric. The NSGA-I had better performance in three instances. The MOVNS1 was superior in two instances. In the mean, the NSGA-II presented the best result for the HCC metric.

These results indicate that the NSGA-II algorithm outperforms NSGA-I, MOVNS1, and MOVNS2 algorithms concerning this metric.

Figures 5.4(a) and 5.4(b) illustrate the Pareto front obtained from each algorithm

Table 5.10: Average HCC to the instances of set2 in the tested algorithms. The best values are highlighted in bold.

#ID	MOVNS1	MOVNS2	NSGA-II	NSGA-I
1	1,282.79	1,151.40	**1,446.94**	1,422.89
2	**2,042.02**	1,674.68	1,653.94	1,844.78
3	402.08	401.53	**431.34**	397.67
4	1,689.60	1,437.69	1,861.56	**1,885.60**
5	6,708.21	6,474.27	**9,187.19**	8,270.10
6	**12,017.58**	11,600.85	11,042.90	10,844.27
7	3,006.93	2,767.26	2,556.17	**3,429.42**
8	8,550.94	7,745.17	**8,790.46**	8,739.29
9	9,880.29	10,525.14	**21,967.44**	19,413.19
10	32,607.87	43,214.28	60,472.87	**61,504.40**
11	5,763.31	5,302.41	**12,925.00**	11,851.43
12	18,301.29	20,141.95	**29,703.90**	29,611.39
Mean	8,521.08	9,369.72	**13,503.31**	13,267.87

in two different instances. The first instance has 50 jobs and 20 machines, and the second has 750 jobs and 10 machines. As can be seen, the NSGA-II produced sets of non-dominated solutions with good diversity compared to other algorithms. In its turn, the MOVNS1 and MOVNS2 algorithms converge well.

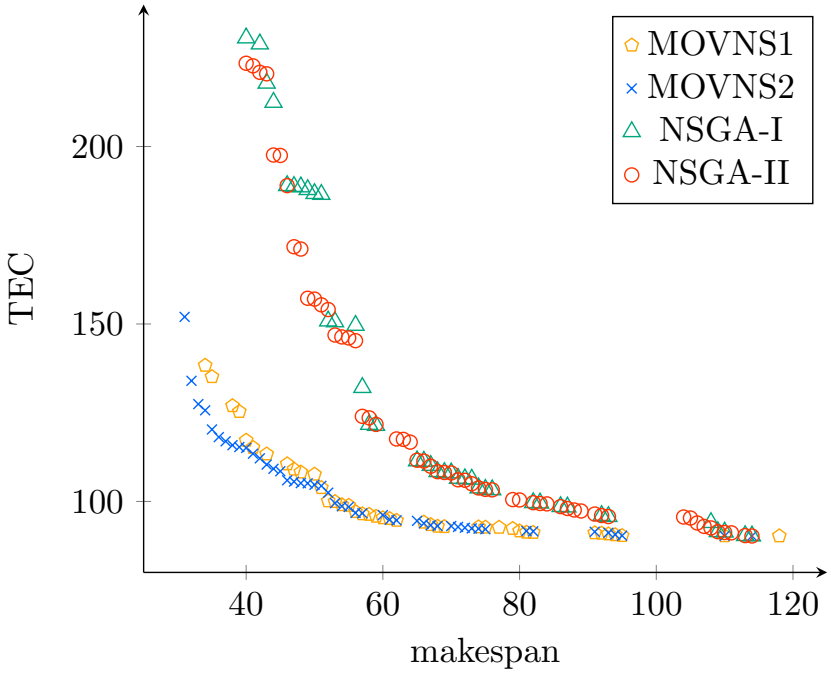
In industrial problems, as treated in this work, the goal is to find a Pareto front that optimizes the objective functions; in other words, a Pareto front with good convergence. Thus, we can conclude that the MOVNS2 algorithm presents the best performance for the addressed problem.

5.4.3 Statistical Analysis

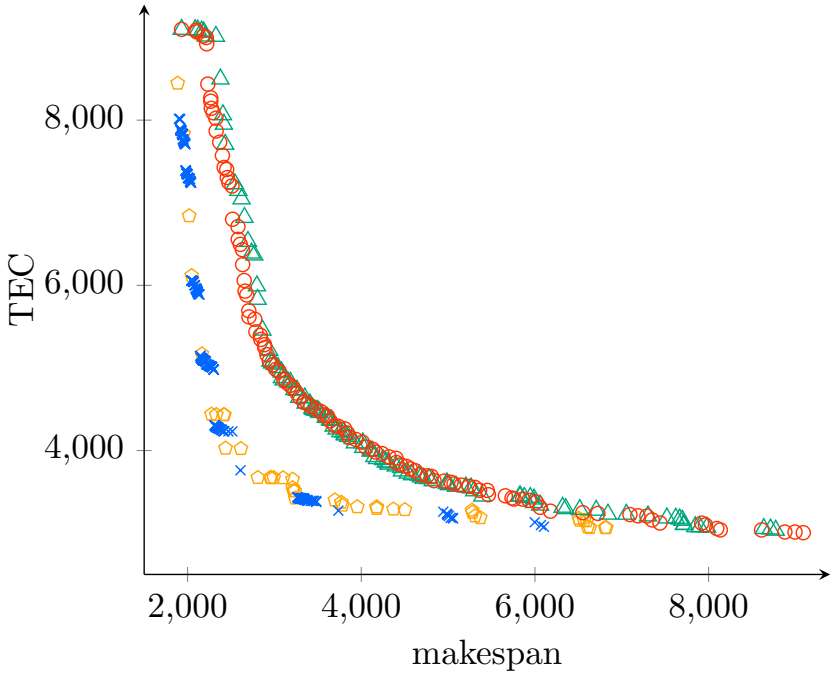
We performed an exploratory analysis to understand the samples data before performing the statistical test.

Figure 5.5 (a)-(b) shows the boxplot of the RPD^{HV} and HCC results, respectively.

Before performing the hypothesis tests, we need to choose the test type, parametric or non-parametric. Generally, parametric tests are more powerful; however, to use them,

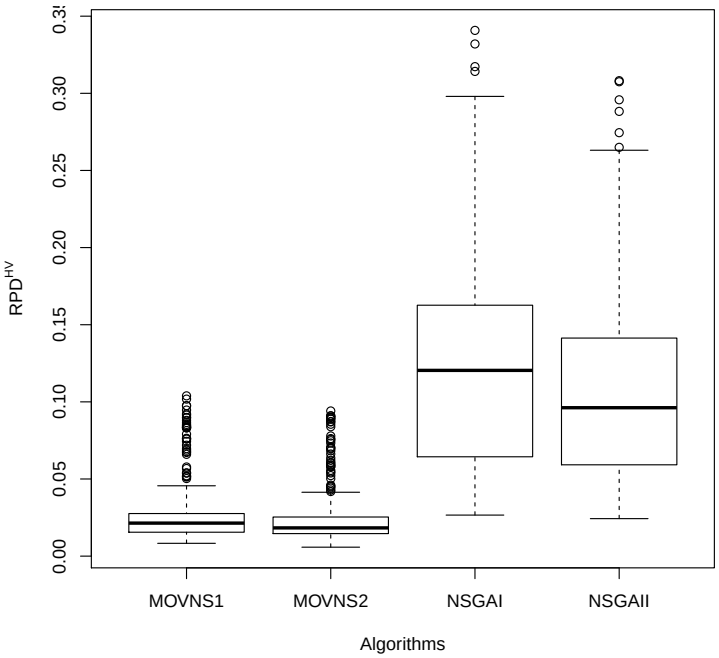


(a) Instance with 50 jobs and 20 machines

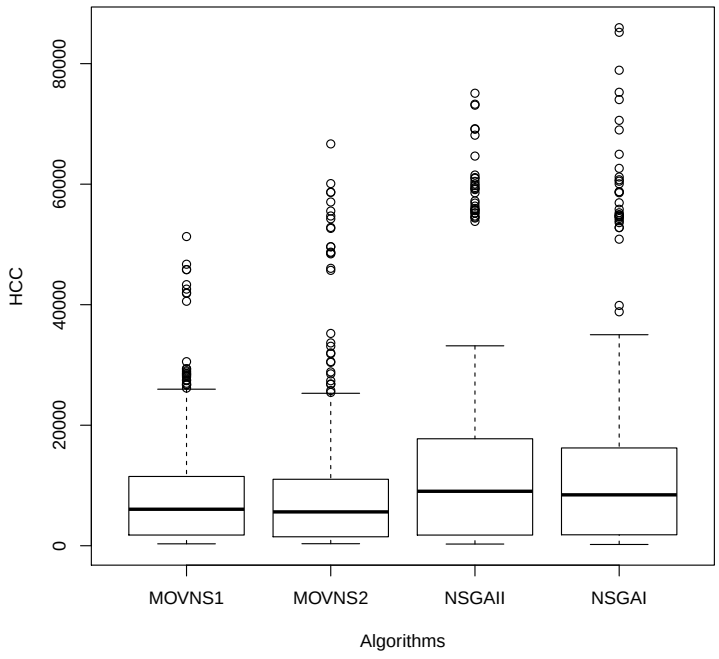


(b) Instance with 750 jobs and 10 machines

Figure 5.4: Example of the Pareto front obtained from each algorithm



(a) Boxplot of the RPD^{HV} results



(b) Boxplot of the HCC results

Figure 5.5: Boxplots of the results

it is necessary to satisfy three assumptions (García et al., 2010):

1. Normality: Every sample must originate from a population with normal distribution,
2. Independence: The samples must be independent of each other,
3. Homoscedasticity: There must be equality of variances across samples.

We applied the Shapiro-Wilk normality test to the samples with the RPD^{HV} and HCC values from each algorithm and showed its results in Table 5.11.

Table 5.11: p-values of the Shapiro-Wilk normality test concerning RPD^{HV} and HCC values

Algorithm	p-value	
	RPD^{HV}	HCC
MOVNS1	2.2e-16	2.2e-16
MOVNS2	2.2e-16	2.2e-16
NSGA-I	1.35e-10	2.2e-16
NSGA-II	4.413e-12	2.2e-16

With a confidence level of 95% ($\alpha = 0.05$), we can say that the results presented in Table 5.11 do not present evidence that the results of the algorithms come from a population with normal distribution.

Thus, to verify if the differences between the results presented by the algorithms are statistically significant, we performed the non-parametric Friedman test with Bonferroni correction.

The test presented a p-value equal to 2.2e-16. Therefore, we can conclude that the observed difference is statistically significant from the test result.

Thus, we applied the Paired Wilcoxon signed-rank non-parametric test (Wilcoxon, 1945) to identify the pairs of results that present the differences. Table 5.12 reports the results of this test obtained by the MOVNS1, MOVNS2, NSGA-I, and NSGA-II algorithms for the RPD^{HV} and HCC values samples.

Table 5.12: p-values of the paired Wilcoxon signed-rank test concerning RPD^{HV} and HCC values ($\alpha = 0.05$).

group 1	group 2	p-value	
		RPD^{HV}	HCC
MOVNS1	MOVNS2	2e-16	0.03497
MOVNS1	NSGA-I	7.8e-15	2e-16
MOVNS1	NSGA-II	1.1e-07	2e-16
MOVNS2	NSGA-I	0.028	2e-16
MOVNS2	NSGA-II	2.2e-06	2e-16
NSGA-I	NSGA-II	2e-16	0.00056

According to Table 5.12, there is a significant statistical difference between all algorithms concerning RPD^{HV} and HCC metrics. Thus, these tests confirm the results in Tables 5.9 and 5.10, indicating that MOVNS2 outperforms all other algorithms regarding the RPD^{HV} metric, and the NSGA-II is superior concerning the HCC metric.

Chapter 6

Conclusions

This thesis addressed the unrelated parallel machine scheduling problem with sequence-dependent setup times for minimizing the makespan and total energy cost under time-of-use electricity price.

To solve it, we developed a mixed-integer linear programming formulation and applied the weighted sum method to generate sets of non-dominated solutions to the problem. Considering that this formulation could not solve larger instances of the problem, we adapted heuristic algorithms to deal with them.

To test the methods, we adapted instances of the literature to contemplate the problem's characteristics addressed. We divided these instances into two groups. The first group consists of small instances with up to 10 jobs and 2 machines, while the second group contains large instances, with up to 750 jobs and 20 machines. We evaluated the methods concerning the hypervolume and HCC metrics.

Initially, we used part of the set of instances to tuning the parameter values of the algorithms. To this end, we used the Irace package.

We validated the NSGA-II and MOVNS2 results in small instances, by comparing them with the results of the exact method. The algorithms showed good convergence and diversity. Besides, it spent much shorter CPU time than that required by the exact method.

We compare the proposed NSGA-II and MOVNS2 with the NSGA-I and MOVNS1 of the literature in instances with up to 750 jobs and 50 machines. The results showed that the MOVNS2 outperforms MOVNS1, NSGA-I, and NSGA-II algorithms concerning

the hypervolume metric. Further, the NSGA-II is superior to MOVNS1, MOVNS2, and NSGA-I algorithms regarding the HCC metric. Both results are with a 95% confidence level. Thus, the proposed MOVNS2 and NSGA-II algorithms find non-dominated solutions with good convergence and diversity, respectively. As we address an industrial problem in this work, we concluded that the MOVNS2 presents the best performance since it finds Pareto fronts with the best convergence.

As future work, we suggest testing other crossover and mutation operators for the NSGA-II. Besides, we intend to implement other multi-objective algorithms, such as Strength Pareto Evolutionary Algorithm 2 (SPEA2) and Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D). Moreover, we indicate to test mechanisms that improve the diversity of solutions generated by MOVNS2.

Appendix A

Publications

As result of this work, the following publications were produced:

Title: *A mathematical formulation and an NSGA-II algorithm for minimizing the makespan and energy cost under time-of-use electricity price in an unrelated parallel machine scheduling*

Authors: Marcelo Ferreira Rego, Júlio Cesar Evaristo Moreira Pinto, Luciano Perdigão Cota, and Marcone Jamilson Freitas Souza

Journal: PeerJ Computer Science

Issn: 2376-5992

Year: 2022

Qualis Computer Science: A2

DOI: 10.7717/peerj-cs.844

Title: *Smart General Variable Neighborhood Search with Local Search based on Mathematical Programming for solving the Unrelated Parallel Machine Scheduling Problem*

Authors: Marcelo Ferreira Rego and Marcone Jamilson Freitas Souza

Conference: 21th International Conference on Enterprise Information Systems (ICEIS 2019)

Address: Heraklion, Creta, Grécia

Year: 2019

Month: 3-5 May

Qualis Computer Science: B2

The following book chapter was published:

Title: *A Hybrid Algorithm for the Unrelated Parallel Machine Scheduling Problem*

Authors: Marcelo Ferreira Rego and Marcone Jamilson Freitas Souza

Booktitle: Revised selected papers of the 21st International Conference, ICEIS 2019

Series: Lecture Notes in Business Information Processing

Publisher: Springer International Publishing

Year: 2020

Isbn: 978-3-030-40783-4

Editors: Filipe J., Śmiałek M., Brodsky A., Hammoudi S.

Bibliography

Aalami, H. A.; Moghaddam, M. Parsa and Yousefi, G. R. (2015). Evaluation of nonlinear models for time-based rates demand response programs. *International Journal of Electrical Power & Energy Systems*, v. 65, p. 282–290.

Afzalirad, Mojtaba and Rezaeian, Javad. (2017). A realistic variant of bi-objective unrelated parallel machine scheduling problem: Nsga-ii and moaco approaches. *Applied Soft Computing*, v. 50, p. 109–123.

Ahilan, C; Kumanan, Somasundaram; Sivakumaran, N and Dhas, J Edwin Raja. (2013). Modeling and prediction of machining quality in CNC turning process using intelligent hybrid decision making tools. *Applied Soft Computing*, v. 13, n. 3, p. 1543–1551.

Albadi, Mohamed H and El-Saadany, Ehab F. (2007). Demand response in electricity markets: An overview. *2007 IEEE power engineering society general meeting*, p. 1–5, Tampa, FL, USA. IEEE.

Arroyo, J. E. C.; Ottoni, R. S. and Oliveira, A. P. (2011). Multi-objective variable neighborhood search algorithms for a single machine scheduling problem with distinct due windows. *Electronic Notes in Theoretical Computer Science*, v. 281, p. 5–19.

Babazadeh, Hossein; Alavidoost, MH; Zarandi, MH Fazel and Sayyari, ST. (2018). An enhanced nsga-ii algorithm for fuzzy bi-objective assembly line balancing problems. *Computers & industrial engineering*, v. 123, p. 189–208.

Bandyopadhyay, Susmita and Bhattacharya, Ranjan. (2013). Solving multi-objective parallel machine scheduling problem by a modified nsga-ii. *Applied Mathematical Modelling*, v. 37, n. 10-11, p. 6718–6729.

BEIS,. Industrial electricity prices in the IEA 2020.; U.K. Department for Business Energy Industrial Strategy, (2020). Available at https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/895162/table_531.xlsx, accessed on February 28, 2020.

Bektur, Gulcin. (2021). An nsga-ii-based memetic algorithm for an energy-efficient unrelated parallel machine scheduling problem with machine-sequence dependent setup times and learning effect. *Arabian Journal for Science and Engineering*, p. 1–16.

Che, Ada; Zeng, Yizeng and Lyu, Ke. (2016). An efficient greedy insertion heuristic for energy-conscious single machine scheduling problem under time-of-use electricity tariffs. *Journal of Cleaner Production*, v. 129, p. 565–577.

Che, Ada; Zhang, Shibohua and Wu, Xueqi. (2017). Energy-conscious unrelated parallel machine scheduling under time-of-use electricity tariffs. *Journal of Cleaner Production*, v. 156, p. 688–697.

Cheng, Junheng; Chu, Feng and Zhou, Mengchu. (2018). An improved model for parallel machine scheduling under time-of-use electricity price. *IEEE Transactions on Automation Science and Engineering*, v. 15, n. 2, p. 896–899.

Cheng, Junheng; Wu, Peng and Chu, Feng. 25-27 September(2019). Mixed-integer programming for unrelated parallel machines scheduling problem considering electricity cost and makespan penalty cost. *2019 International Conference on Industrial Engineering and Systems Management (IESM)*, p. 1–5, Shanghai, China. IEEE.

Chiaraviglio, Luca; Mellia, Marco and Neri, Fabio. (2011). Minimizing isp network energy cost: Formulation and solutions. *IEEE/ACM Transactions On Networking*, v. 20, n. 2, p. 463–476.

Cota, Luciano P; Coelho, Vitor N; Guimarães, Frederico G and Souza, Marcone J F. (2018). Bi-criteria formulation for green scheduling with unrelated parallel machines with sequence-dependent setup times. *International Transactions in Operational Research*, v. 28, p. 996–1017.

Cota, Luciano P; Guimarães, Frederico G; Ribeiro, Roberto G; Meneghini, Ivan R; de Oliveira, Fernando B; Souza, Marcone J F and Siarry, Patrick. (2019). An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem. *Swarm and Evolutionary Computation*, v. 51, p. 100601.

Deb, Kalyanmoy. (2014). Multi-objective optimization. Burke, Edmund K. and Kendall, Graham, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, p. 403–449. Springer US, Boston, MA.

Deb, Kalyanmoy; Pratap, Amrit; Agarwal, Sameer and Meyarivan, TAMT. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, v. 6, n. 2, p. 182–197.

Deb, Kalyanmoy; Rao N., Udaya Bhaskara and Karthik, S. (2007). Dynamic multi-objective optimization and decision-making using modified nsga-ii: a case study on hydro-thermal power scheduling. Obayashi, Shigeru; Deb, Kalyanmoy; Poloni, Carlo; Hiroyasu, Tomoyuki and Murata, Tadahiko, editors, *Evolutionary Multi-Criterion Optimization*, p. 803–817, Matsushima, Japan. Springer Berlin Heidelberg.

Deng, Wu; Xu, Junjie and Zhao, Huimin. (2019). An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. *IEEE access*, v. 7, p. 20281–20292.

Ding, J.; Song, S.; Zhang, R.; Chiong, R. and Wu, C. (2016). Parallel machine scheduling under time-of-use electricity prices: New models and optimization approaches. *IEEE Transactions on Automation Science and Engineering*, v. 13, n. 2, p. 1138–1154.

Ebrahimi, Ahmad; Jeon, Hyun Woo; Lee, Seokgi and Wang, Chao. (2020). Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system: A comparison of four metaheuristic algorithms. *Computers & Industrial Engineering*, v. 141, p. 106295.

EIA,. May(2016). International energy outlook 2016 with projections to 2040. Technical report DOE/EIA-0484, U.S. Energy Information Administration, U.S. Department of Energy, Washington. Available at https://www.eia.gov/outlooks/ieo/pdf/0484(2016).pdf, accessed on February 1, 2020.

Falsafi, Hananeh; Zakariazadeh, Alireza and Jadid, Shahram. (2014). The role of demand response in single and multi-objective wind-thermal generation scheduling: A stochastic programming. *Energy*, v. 64, p. 853–867.

Fang, Kan; Uhan, Nelson; Zhao, Fu and Sutherland, John W. (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems*, v. 30, n. 4, p. 234–240.

Feo, Thomas A and Resende, Mauricio G. C. (1995). Greedy randomized adaptive search procedures. *Journal of global optimization*, v. 6, n. 2, p. 109–133.

Fysikopoulos, Apostolos; Pastras, Georgios; Alexopoulos, Theocharis and Chryssolouris, George. (2014). On a generalized approach to manufacturing energy efficiency. *The International Journal of Advanced Manufacturing Technology*, v. 73, n. 9-12, p. 1437–1452.

García, Salvador; Fernández, Alberto; Luengo, Julián and Herrera, Francisco. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, v. 180, n. 10, p. 2044–2064.

Garey, Michael R and Johnson, David S. (1979). *Computers and intractability. A guide to the theory of NP-completeness*, volume 174. Freeman, San Francisco.

Geiger, Martin Josef. (2008). Randomised variable neighbourhood search for multi objective optimisation. *4th EU/ME Workshop: Design and Evaluation of Advanced Hybrid Meta-Heuristics*, p. 34–42, Nottingham, United Kingdom.

Gotoda, Shohei; Ito, Minoru and Shibata, Naoki. 13–16 May(2012). Task scheduling algorithm for multicore processor system for minimizing recovery time in case of single node fault. *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, p. 260–267, Ottawa, Canada. IEEE. doi: 10.1109/CCGrid.2012.23.

Graham, R. L.; Lawler, E. L.; Lenstra, J. K. and Kan, A. H. G. Rinnooy. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. v. 5, p. 287–326.

Guimarães, F. G.; Wanner, E. F. and Takahashi, R. H. C. (2009). A quality metric for multi-objective optimization based on hierarchical clustering techniques. *2009 IEEE Congress on Evolutionary Computation*, p. 3292–3299, Trondheim, Norway.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, (2020). Available at http://www.gurobi.com , accessed on February 4, 2020.

Hansen, Pierre and Mladenović, Nenad. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, v. 130, n. 3, p. 449–467.

Hansen, Pierre; Mladenović, Nenad; Todosijević, Raca and Hanafi, Saïd. (2017). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, v. 5, n. 3, p. 423–454.

Hong, Feng; Chen, Hao; Cao, Bin and Fan, Jing. (2021). A moead-based approach to solving the staff scheduling problem. Gao, Honghao; Wang, Xinheng; Iqbal, Muddesar; Yin, Yuyu; Yin, Jianwei and Gu, Ning, editors, *Collaborative Computing: Networking, Applications and Worksharing*, p. 112–131, Pittsburgh, PA USA. Springer International Publishing.

Huang, Simin; Cai, Linning and Zhang, Xiaoyue. (2010). Parallel dedicated machine scheduling problem with sequence-dependent setups and a single server. *Computers & Industrial Engineering*, v. 58, n. 1, p. 165–174.

Keshavarz, Taha; Karimi, Erfaneh and Shakhsi-Niaei, Majid. (2021). Unrelated parallel machines scheduling with sequence-dependent setup times to minimize makespan and tariff charged energy consumption. *Advances in Industrial Engineering*, v. 55, n. 1, p. 91–113.

Kopanos, Georgios M; Laínez, José Miguel and Puigjaner, Luis. (2009). An efficient mixed-integer linear programming scheduling framework for addressing sequence-dependent setup issues in batch plants. *Industrial & Engineering Chemistry Research*, v. 48, n. 13, p. 6346–6357.

Kurniawan, B; Gozali, AA; Weng, W and Fujimura, S. (2017). A genetic algorithm for unrelated parallel machine scheduling minimizing makespan cost and electricity cost under time-of-use (tou) tariffs with job delay mechanism. *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, p. 583–587, Singapore. IEEE.

Kurniawan, Bobby; Chandramitasari, Widyaning; Gozali, Alfian Akbar; Weng, Wei and Fujimura, Shigeru. (2020). Triple-chromosome genetic algorithm for unrelated parallel machine scheduling under time-of-use tariffs. *IEEJ Transactions on Electrical and Electronic Engineering*, v. 15, n. 2, p. 208–217.

Liang, Peng; Yang, Hai-dong; Liu, Guo-sheng and Guo, Jian-hua. (2015). An ant optimization model for unrelated parallel machine scheduling with energy consumption and total tardiness. *Mathematical Problems in Engineering*, v. 2015.

Liu, Ying; Dong, Haibo; Lohse, Niels; Petrovic, Sanja and Gindy, Nabil. (2014). An investigation into minimising total energy consumption and total weighted tardiness in job shops. *Journal of Cleaner Production*, v. 65, p. 87–96.

López-Ibáñez, Manuel; Dubois-Lacoste, Jérémie; Cáceres, Leslie Pérez; Birattari, Mauro and Stützle, Thomas. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58.

Mansouri, S Afshin; Aktas, Emel and Besikci, Umut. (2016). Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. *European Journal of Operational Research*, v. 248, n. 3, p. 772–788.

Marler, R Timothy and Arora, Jasbir S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, v. 26, n. 6, p. 369–395.

Mladenović, Nenad and Hansen, Pierre. (1997). Variable neighborhood search. *Computers & operations research*, v. 24, n. 11, p. 1097–1100.

Moon, Joon-Yung; Shin, Kitae and Park, Jinwoo. (2013). Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. *The International Journal of Advanced Manufacturing Technology*, v. 68, n. 1-4, p. 523–535.

Park, Jongsoo and Dally, William J. June 13–5(2010). Buffer-space efficient and deadlock-free scheduling of stream applications on multi-core architectures. *Proceedings of the Twenty-Second Annual ACM Symposium on Parallelism in Algorithms and Architectures*, p. 1–10, New York, NY, USA. Association for Computing Machinery.

Pinedo, Michael L. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing, New York.

Pinto, Julio Cesar Evaristo Moreira; Cota, Luciano Perdigão; Guimarães, Frederico G. and Souza, Marcone Jamilson Freitas. (2019). Uma formulação de programação matemática para minimizar o makespan e o custo de energia em um problema de sequenciamento em máquinas paralelas. *Anais do LI Simpósio Brasileiro de Pesquisa Operacional*, volume 2, p. 107764, Campinas, Brazil. SOBRAPO.

Pour, Shahrzad M; Drake, John H; Ejlertsen, Lena Secher; Rasmussen, Kourosh Marjani and Burke, Edmund K. (2018). A hybrid constraint programming/mixed integer programming framework for the preventive signaling maintenance crew scheduling problem. *European Journal of Operational Research*, v. 269, n. 1, p. 341–352.

Raquel, Carlo R. and Naval, Prospero C. (2005). An effective use of crowding distance in multiobjective particle swarm optimization. *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, p. 257–264, Washington DC USA. Association for Computing Machinery.

Rego, Marcelo Ferreira; Cota, Luciano Perdigão and Souza, Marcone Jamilson Freitas. Instances for the upmsp with sequence-dependent setup times under time-of-use electricity price, (2021). Available at https://github.com/marcelofr/UPMSP_ME_INSTANCE, accessed on February 4, 2022.

Ropke, Stefan and Pisinger, David. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, v. 40, n. 4, p. 455–472.

Rosa, Bruno Ferreira and Souza, Marcone Jamilson Freitas. 01-04 September(2009). Uma nova formulação de programação matemática indexada no tempo para uma classe de problemas de sequenciamento em uma máquina. *Anais do XLI Simpósio Brasileiro de Pesquisa Operacional*, volume 41, p. 2898–2909, Porto Seguro, Brazil. SOBRAPO.

Ruiz, Rubén; Pan, Quan-Ke and Naderi, Bahman. (2019). Iterated greedy methods for the distributed permutation flowshop scheduling problem. *Omega*, v. 83, p. 213–222.

Saberi-Aliabad, Hossein; Reisi-Nafchi, Mohammad and Moslehi, Ghasem. (2020). Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs. *Journal of Cleaner Production*, v. 249, p. 119393.

Scheffé, Henry. (1958). Experiments with mixtures. *Journal of the Royal Statistical Society: Series B (Methodological)*, v. 20, n. 2, p. 344–360.

Senbel, Samah. July 16–17(2019). Novel recursive technique for finding the optimal solution of the nurse scheduling problem. Arai, Kohei; Bhatia, Rahul and Kapoor, Supriya, editors, *Intelligent Computing*, p. 359–375, London, United Kingdom. Springer.

Shang, Ke; Ishibuchi, Hisao; He, Linjun and Pang, Lie Meng. (2020). A survey on the hypervolume indicator in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, v. 25, n. 1, p. 1–20.

Shen, Liji; Dauzère-Pérès, Stéphane and Neufeld, Janis S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, v. 265, n. 2, p. 503–516.

Shrouf, Fadi; Ordieres-Meré, Joaquin; García-Sánchez, Alvaro and Ortega-Mier, Miguel. (2014). Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *Journal of Cleaner Production*, v. 67, p. 197–207.

Srinivas, Nidamarthi and Deb, Kalyanmoy. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, v. 2, n. 3, p. 221–248.

Sun, Xiang; Guo, Shunsheng; Guo, Jun and Du, Baigang. (2019). A hybrid multi-objective evolutionary algorithm with heuristic adjustment strategies and variable neighbor-hood search for flexible job-shop scheduling problem considering flexible rest time. *IEEE Access*, v. 7, p. 157003–157018.

Tsao, Yu-Chung; Thanh, Vo-Van and Hwang, Feng-Jang. (2020). Energy-efficient single-machine scheduling problem with controllable job processing times under differential electricity pricing. *Resources, Conservation and Recycling*, v. 161, p. 104902.

Vallada, Eva and Ruiz, Rubén. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, v. 211, n. 3, p. 612–622.

Wang, Hongfeng; Fu, Yaping; Huang, Min; Huang, George Q and Wang, Junwei. (2017). A nsga-ii based memetic algorithm for multiobjective parallel flowshop scheduling problem. *Computers & Industrial Engineering*, v. 113, p. 185–194.

Wang, Shijin; Liu, Ming; Chu, Feng and Chu, Chengbin. (2016). Bi-objective optimization of a single machine batch scheduling problem with energy cost consideration. *Journal of Cleaner Production*, v. 137, p. 1205–1215.

Wang, Yong and Li, Lin. (2015). Time-of-use electricity pricing for industrial customers: A survey of us utilities. *Applied Energy*, v. 149, p. 89–103.

Wanner, Elizabeth F; Guimaraes, Frederico G; Takahashi, Ricardo HC and Fleming, Peter J. (2006). A quadratic approximation-based local search procedure for multi-objective genetic algorithms. *2006 IEEE International Conference on Evolutionary Computation*, p. 938–945, Vancouver, BC, Canada. IEEE.

Wilcoxon, Frank. (1945). Some uses of statistics in plant pathology. *Biometrics Bulletin*, v. 1, n. 4, p. 41–45.

Willeke, Stefan; Ullmann, Georg and Nyhuis, Peter. (2016). Method for an energy-cost-oriented manufacturing control to reduce energy costs: Energy cost reduction by using a new sequencing method. *2016 International Conference on Industrial Engineering, Management Science and Application (ICIMSA)*, p. 1–5, Jeju, Korea (South). IEEE.

Wolf, Joel; Bansal, Nikhil; Hildrum, Kirsten; Parekh, Sujay; Rajan, Deepak; Wagle, Rohit; Wu, Kun-Lung and Fleischer, Lisa. December 1-5(2008). Soda: An optimizing scheduler for large-scale stream-based distributed computer systems. Issarny, Valérie and Schantz, Richard, editors, *Middleware 2008*, p. 306–325, Berlin, Heidelberg. Springer Berlin Heidelberg.

Wu, Xueqi and Che, Ada. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega*, v. 82, p. 155–165.

Wu, Xueqi and Che, Ada. (2020). Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. *Omega*, v. 94, p. 102117.

Yan, Jingyu; Li, Chongguo; Wang, Zhi; Deng, Lei and Sun, Demin. (2007). Diversity metrics in multi-objective optimization: Review and perspective. *2007 IEEE International Conference on Integration Technology*, p. 553–557, Shenzhen, China. IEEE. doi: 10.1109/ICITECHNOLOGY.2007.4290378.

Zeng, YiZeng; Che, Ada and Wu, Xueqi. (2018). Bi-objective scheduling on uniform parallel machines considering electricity cost. *Engineering Optimization*, v. 50, n. 1, p. 19–36.

Zhang, Hao; Zhao, Fu; Fang, Kan and Sutherland, John W. (2014). Energy-conscious flow shop scheduling under time-of-use electricity tariffs. *CIRP Annals*, v. 63, n. 1, p. 37 – 40.

Zhang, Like; Deng, Qianwang; Lin, Ruihang; Gong, Guiliang and Han, Wenwu. (2021). A combinatorial evolutionary algorithm for unrelated parallel machine scheduling problem with sequence and machine-dependent setup times, limited worker resources and learning effect. *Expert Systems with Applications*, v. 175, p. 114843.

Zhang, Qingfu and Li, Hui. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, v. 11, n. 6, p. 712–731.

Zitzler, E. and Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms – a comparative case study. *Parallel problem solving from nature-PPSN V*, p. 292–301, Amsterdam, Netherlands. Springer.

Zitzler, Eckart; Deb, Kalyanmoy and Thiele, Lothar. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, v. 8, n. 2, p. 173–195.