Multi-objective meta-heuristics for a scheduling problem with sequence-dependent machine deterioration

Vívian Ludimila Aguiar Santos
Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil
Instituto Federal do Norte Minas Gerais
Rua Humberto Mallard 1355, Bairro Santos Dumont, 39274-140, Pirapora, MG, Brazil
vivian.santos@ifnmg.edu.br

Thales Francisco Mota Carvalho
Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais
Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil
Institute of Engineering, Science and Technology, Universidade Federal dos Vales do Jequitinhonha e Mucuri, Av. Um, 4.050 - Cidade Universitária, 39447-814, Janaúba, MG, Brazil
thales.mota@ufvjm.edu.br

Luciana Pereira de Assis
Computer Department, Universidade Federal dos Vales do Jequitinhonha e Mucuri
Rodovia MGT 367 - Km 583, 39100-000, Diamantina, MG, Brazil
lpassis@ufvjm.edu.br

Miri Weiss Cohen
Department of Software Engineering, Braude College of Engineering
Karmiel, 2161002, Israel
miri@braude.ac.il

Frederico Gadelha Guimarães
Machine Intelligence and Data Science (MINDS) Laboratory, School of Engineering, Universidade Federal de Minas Gerais, Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil
fredericoguimaraes@ufmg.br

ABSTRACT

This paper addresses a scheduling problem considering sequence-dependent machine deterioration, with the criterion of minimizing the makespan and total tardiness simultaneously. To handle this problem, we compare two different approaches to extending Iterated Local Search (ILS) meta-heuristic to multi-objective problems. In this paper, we develop the Pareto Iterated Local Search (PILS), which is a direct extension of the Pareto Local Search (PLS) heuristic that uses multi-objective local search. On the other hand, we use a hybrid meta-heuristic algorithm, named ILS/D, based on the Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D), and on the ILS. This paper aims to evaluate the performance of the algorithm used to solve the problem addressed, in an algorithm based only on local search meta-heuristic, since the problem has already been evaluated in algorithms based on population and hybrid meta-heuristics. Numerical experiments are carried out on test problems with different scales.

KEYWORDS. Machine deterioration. Sequence-depended. Multi-objective optimization.

https://proceedings.science/p/157513?lang=pt-br

1. Introduction

This paper considers a scheduling problem with deterioration in the machines. This problem consists in processing the jobs in unrelated parallel machines taking into account that each job executed causes a certain deterioration in the machine. This deterioration will decrease the performance of the machine, increasing the processing time of the next job to be processed. The deterioration of job processing times depends upon the specific jobs that have been processed by the machine previously. This concept is known as sequence-dependent deterioration. To the best of our knowledge, Ruiz-Torres et al. [2013] were the first authors who considered this concept in scheduling problems.

In the first paper, Ruiz-Torres et al. [2013] address the minimization of makespan. They consider that the deterioration of the processing time for a job depends on the specific jobs that have been previously processed by the machine and that the deterioration of the machine increases the processing times of the jobs. The authors show that for a single machine the problem can be solved in polynomial time and that for two or more machines the problem is NP-hard. They proposed a Mixed Integer Nonlinear Programming model (MINLP) and algorithms based on the Simulated Annealing (SA) meta-heuristic for a large number of instances.

In a second paper, Ruiz-Torres et al. [2015] address the problem of maximizing the percentage of jobs delivered on-time, i.e., minimizing the number of tardy jobs. As in the previous paper, the deterioration of each machine is a function of the sequence of jobs that were processed by the machine. Two machine programming strategies are combined with a set of sequencing algorithms to solve the problems of identical machines and unrelated machines. They proposed an MINLP model and tested a large set of problem instances that consider various levels of the number of jobs and machines, the due date, and the deterioration effect.

Santos et al. [2022] propose a multi-objective approach extending the problem defined by Ruiz-Torres et al. [2013, 2015]. They present a MINLP model for the problem that has two objectives to be minimized: the makespan and the total time of delay of the jobs. To solve the problem, they proposed the heuristic algorithm based on the decomposition and aggregation approach used in the Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) [Zhang e Li, 2007], in which the Iterated Local Search (ILS) meta-heuristic [Lourenço et al., 2003] is used as the search engine. The method developed is named Iterated Local Search Based on Decomposition (ILS/D). They compared the ILS/D, MOEA/D and Non-dominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al., 2002] algorithms. The results show that the ILS/D outperforms the MOEA/D and NSGA-II algorithms by a significant margin.

In this work, we address the same problem formulated by Ruiz-Torres et al. [2013, 2015]. However, we use the multi-objective perspective as proposed by Santos et al. [2022]. To solve the problem, we develop the Pareto Iterated Local Search (PILS) heuristic, which is a direct extension of the Pareto Local Search (PLS) heuristic that uses multi-objective local search. In the article by Santos et al. [2022], the problem addressed was evaluated with algorithms based on population meta-heuristics and with a hybrid algorithm based on population meta-heuristic and local search meta-heuristic. In this work, we seek to evaluate the performance of an algorithm based only on local search meta-heuristic. For this, the PILS algorithm is compared with ILS/D algorithm.

The main research question we want to explore is the following: which approach is more efficient as a way to extend ILS to the approached multi-objective problem? ILS/D relies on the decomposition and aggregation framework and employs a single-objective ILS to improve each sub-problem. Conversely, PILS follows the common ILS framework but relying on a Pareto-based local search. The answer to this question can contribute with an important implication for local

search-based meta-heuristics related to multi-objective problems.

The remainder of this paper is organized as follows. Section 2 presents the problem definition. The components of the proposed multi-objective algorithm are presented in Section 3. Section 4 presents the computational experiments and discussion. Lastly, Section 5 concludes the work.

2. Problem definition

The problem was defined by Ruiz-Torres et al. [2013, 2015] as follows. There is a set of n jobs, $N = \{1, \ldots, j, \ldots, n\}$, to be processed on a set of m unrelated parallel machines, $M = \{1, \ldots, k, \ldots, m\}$. All jobs are available for processing at time zero and job preemption is not allowed. Each machine can process at most one job at a time and can not stand idle until the last job assigned to it has been finished. The normal processing time of job j on machine k is defined by p_{jk}. The deteriorating effect of job j on machine k is given by d_{jk}, where, $0 \leq d_{jk} \leq 1, \forall j \in N$ and $k \in M$. The performance of the machine k in the position h (i.e., after executing the job that is in the position $h-1$) is given by: $q_{kh} = (1 - d_{(h-1)k}) \times q_{k(h-1)}$, where, $d_{(h-1)k}$ is the deteriorating effect of job in position $h-1$. It is assumed the machines start with no deterioration, thus the performance level to process the job in position 1 is 100%, i.e. $q_{k1} = 1$, for all $k \in M$. The actual processing of a job j processed on machine k in position h is determined by: $p'_{jk} = \frac{p_{jk}}{q_{kh}}$.There are n possible positions in each machine, and let A be the set of positions h. Each job j has a due date e_j. The job delay time on machine k at position h is defined by t_{kh}. The first objective of the problem is to minimize maximum processing time of all machines, known as makespan (C_{max}) and second objective is to minimize the total delay time (TDT) the jobs assigned to the machines.

To characterise an example problem we define a case study containing $n = 10$ jobs and $m = 3$ machines. Table 1 provides processing times (p_{jk}), deteriorating effects (d_{jk}) and the delivery dates of each job (e_j) are found in the last column. Figure 1 depicts two possible sequences, in which jobs are assigned to each machine. Each position is representing: the performance level of the machine when starting the job execution (above), actual processing time (p'_{jk}), job delay times (t_{kh}), and the total processing time at the end of each job (below). Jobs that are colored green are on-time and jobs that are colored red are late.

Table 1: A problem instance with $n = 10$ and $m = 3$

Job	p_{j1}	p_{j2}	p_{j3}	d_{j1}	d_{j2}	d_{j3}	e_j
1	68.7	44.1	56.7	0.0673	0.0545	0.0601	44.3
2	38.7	56.2	49.3	0.0939	0.0699	0.0708	42.6
3	81.5	93.6	16.3	0.0889	0.0796	0.0620	24.2
4	11.2	7.7	86.6	0.0934	0.0579	0.0588	40.3
5	30.5	57.8	63.2	0.0692	0.0709	0.0974	45.2
6	35.9	11.8	94.6	0.1000	0.0652	0.0862	34.1
7	5.9	99.3	79.0	0.0637	0.0861	0.0787	23.6
8	70.3	27.5	61.7	0.0565	0.0695	0.0860	51.1
9	49.0	9.5	70.4	0.0896	0.0896	0.0999	9.6
10	37.7	21.8	64.4	0.0718	0.0568	0.0637	31.4

Figure 1 shows scheduling 1 with jobs 7, 5, and 2 assigned to machine 1, the jobs 9, 6, 4, 10, and 8 assigned to machine 2, and the jobs 3 and 1 assigned to machine 3. The objective function values are: makespan = 95.16 and TDT = 144.12. In schedule 2, machine 2 which has the longest completion time (makespan) changes the order of jobs in order to reduce the makespan. We can

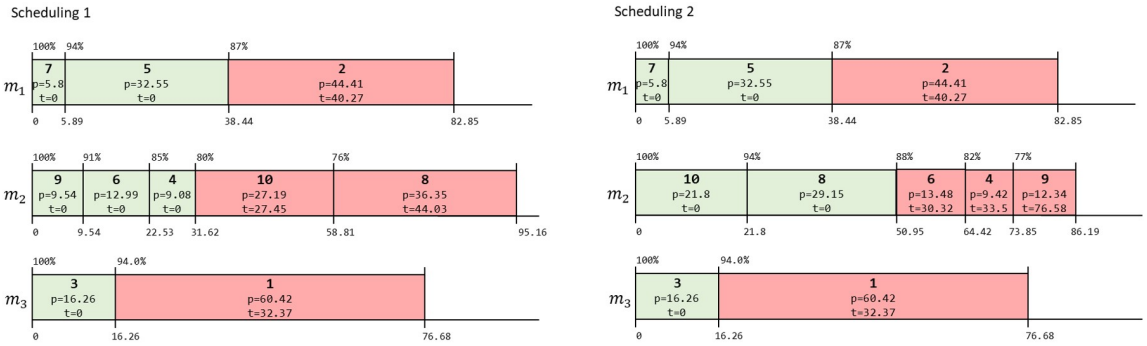


Figure 1: Scheduling for 10 jobs and 3 machines.

see that the new makespan value has been reduced to 86.19. However, in compensation, the new value of the TDT increased to 213.1. Therefore, there is a trade-off when considering the makespan versus the TDT of jobs. That is, the objectives are conflicting.

To solve the problem addressed, we developed a multi-objective heuristic algorithm: Pareto Iterated Local Search, called PILS, presented in the following sections. The second algorithm used to evaluate this problem is the ILS/D algorithm, proposed by Santos et al. [2022]. A detailed explanation of the ILS/D algorithm can be found in the work of Santos et al. [2022].

3. The proposed PILS algorithm

A natural way to extend ILS to multi-objective problems is to replace the local search heuristic by a Pareto-based local search one. Pareto Iterated Local Search (PILS), also known as multi-objective ILS, was first defined by Geiger [2006, 2008]. The PILS algorithm is a direct extension of the Pareto Local Search (PLS) meta-heuristic. The PLS meta-heuristic [Paquete et al., 2004] defines a Pareto dominance based local search for solving multi-objective combinatorial optimization problems. In this paper, we develop a PILS algorithm in order to contrast this approach with the decomposition and aggregation strategy used in ILS/D [Santos et al., 2022].

PILS has a basic structure and few parameters, moreover, it has proven to be a very successful approach to solve different scheduling problems. Geiger [2009] used the methodology of PILS algorithm to the multi-objective flow shop scheduling. They used two different approaches: first, two objectives were considered to be minimized: the makespan and the weighted total delay; while in the second, three objectives were examined: the makespan, the average time of completion and the average time of delay. Similarly, Geiger [2011] studied the permutation flow shop scheduling problems with multiple objectives and compared to other local search approaches. Xu e Lei [2014] studied a two-agent flow shop scheduling problem and proposed the PILS algorithm to minimize the makespan of jobs for the first agent and the total tardiness of jobs for the second agent simultaneously. Gomes et al. [2014] approach the resource-constrained project scheduling problem with precedence relations. They use the PILS algorithm among other multi-objective meta-heuristics to minimizing two criteria: the makespan and the total weighted start time of the activities. Rego et al. [2014] treat the single machine scheduling problem in which the setup time depends on the sequence and the job family, to minimize the makespan and the total weighted tardiness. They show that PILS is statistically better than Multi-objective Variable Neighborhood Search algorithm.

The PILS algorithm, developed in this work, has five steps: (i) Construction of initial solutions; (ii) Selecting a solution from the set of non-dominated solutions to be explored; (iii) Exploration of the neighborhood; (iv) Verification of the visited solutions; and (v) Perturbation.

https://proceedings.science/p/157513?lang=pt-br

A general description of the PILS algorithm is detailed in Algorithm 1. It includes three input parameters: τ (controls the maximum execution time), ϕ (defines the number of solutions created by the constructive procedure) and t (defines the fixed perturbation length). In line 1, the constructive procedure creates the initial solutions. These solutions form the set of non-dominated solutions (ND). Then, we randomly select one of the solutions in the set ND to be the current solution p (line 2). The iterations of the PILS algorithm are performed between lines 3 and 24 until the stop criterion (τ) is satisfied. During each iteration of the innermost loop, we explore the entire neighborhood of p, update the set ND (lines 6-8) and we check if a neighboring solution dominates p (line 9). If true, the solution p' becomes the new current solution (line 10); we reorder the neighborhoods in a random order (line 11) and return to the first neighborhood in the new generated order (line 12). Otherwise, we increase the value of i, that is, it moves on to the next neighborhood (line 14). At the end, we define the solution p as visited (line 17) and check if there is any non-visited solution in the set ND (line 18). If true, a non-visited solution is selected at random to become the current p solution (line 19). Otherwise, we randomly select a solution from the set ND (line 21) and a perturbation method is applied (line 22); Then, the solution p becomes the new current solution. Finally, the solution set ND is returned.

Algorithm 1 PILS

Require: τ, ϕ, t
1: **set** $ND \leftarrow$ constructiveProcedure(ϕ);
2: Select a solution $p \in ND$;
3: **while** (τ not satisfied) **do**
4: $i \leftarrow 1$;
5: **while** ($i \leq 6$) **do**
6: **for** $p' \in N_i(p)$ **do**
7: $ND \leftarrow$non-dominated solutions obtained from $ND \cup p'$;
8: **end for**
9: **if** ($\exists p' \in N_i(p) \mid p'$ dominates p) **then**
10: $p \leftarrow p'$;
11: Reorder neighborhoods $N_1, ..., N_6$ randomly;
12: $i \leftarrow 1$;
13: **else**
14: $i++$;
15: **end if**
16: **end while**
17: Set p as visited;
18: **if** ($\exists q \in ND \mid q$ was not visited yet) **then**
19: $p \leftarrow q$;
20: **else**
21: Select a solution $p \in ND$;
22: $p \leftarrow$ Perturbation (p, t);
23: **end if**
24: **end while**
25: **return** ND;

The constructive procedure, perturbation method and the neighborhood structures used by the PILS algorithm are explained in Section 3.1, Section 3.2 and Section 3.3, respectively.

3.1. Constructive Procedure

The constructive procedure is used to create initial solutions. Initially, we divided the number total of solutions into three groups of the same size. In the first group, individuals who have the shortest time to complete the job processing (makespan) will be generated, checking the best position and the best machine to assign a job. In the second group, individuals who have a shorter TDT for the delivery of jobs will be generated, checking the best position and the best machine to assign a job. In the third group, individuals will be randomly generated, that is, the position and

https://proceedings.science/p/157513?lang=pt-br

the machine to assign a job are chosen randomly. This division into three groups was necessary to have a diverse set of initial solutions, since it takes into account the two objective functions of the problem and also some random solutions.

3.2. Perturbation procedure

The perturbation procedure is based on a circular reallocation of jobs between different machines. First, a set of t machines are randomly chosen: $\{m_1, m_2, ..., m_t\}$. This set has to contain the makespan machine. Next, a job is selected at random in each machine. The job of machine m_1 is inserted in machine m_2, the job of machine m_2 is inserted in machine m_3, so on until the job of machine m_t is inserted in machine m_1.

3.3. Neighborhood structures

There are 6 neighborhood structures used by the local search method, they can be divided into two groups: between different machines and on the same machine.

3.3.1. Neighborhood structures between different machines.

Four types of neighborhood structures were implemented among different machines, three of them explore solutions taking using the makespan objective, while the remaining use the TDT objective. Nevertheless, notice that, for a given solution, these structures may affect the valuation of both objectives. These structures are defined below:

- The *pairwise exchange* move consists of exchanging two jobs j_1 and j_2 belonging to two different machines m_c and m_k, respectively, in such a way that the first machine m_c is the machine with a maximum completion time (makespan machine).
- The *single insertion* move consists in removing a job j from the makespan machine m_c and inserting it in another machine m_k.
- The *two insertion* move consists in removing two jobs j_1 and j_2 from the makespan machine m_c and inserting it in another machine m_k.
- The *delay pairwise exchange* move consists in exchanging two jobs j_1 and j_2 belonging to two different machines m_d and m_k, respectively, in such a way that the first machine m_d is the machine with a maximum delay time.

3.3.2. Neighborhood structures on the same machine.

The main objective of neighborhood structures on the same machine is to reduce the TDT for jobs deliveries, thus obtaining more deliveries on-time. These structures are defined below:

- The *exchange* move consists in exchanging two jobs j_1 and j_2 on the same machine m_k.
- The *insertion* move consists in removing a job j from machine m_k and inserting it on the same machine m_k, in another position.

4. Computational experiments

In this section, the PILS is compared with the ILS/D algorithm proposed by Santos et al. [2022]. The heuristic algorithms ILS/D and PILS were coded in C++ and executed in five independent and identical computers (software and hardware), with configuration: Intel Core i7-870, CPU 2.93 GHz, 8GB of RAM, running Ubuntu 16.04.5 LTS. The performance of each algorithm is measured by computing the hyper-volume (HV) metric [Zitzler e Thiele, 1999]. In the case of two objective functions, the HV provides a value, which is defined as the area of the polygon bounded by the Pareto front of all feasible non-dominated candidate solutions and a given reference point. Thus, the higher the value of HV, the better the quality of the solutions [Vargas et al., 2016].

The algorithms were tested in 900 instances proposed by Ruiz-Torres et al. [2015], which are available at: `https://ruiz-torres.uprrp.edu/nt/`. The instances of Ruiz-Torres et al. [2015] use combinations of the following number of jobs (n) and number of machines (m) values: $n \in \{100, 200\}$ and $m \in \{5, 10, 20\}$. In all the instances, the processing times p_{jk} were generated randomly, with uniform distribution, over interval: [1; 100]. The deteriorating effects (d_{jk}) were also generated randomly, with uniform distribution over two intervals: [1%; 5%] and [5%; 10%]. The congestion ratio (CR) is used to establish the overall due date tightness, a higher CR would result in a greater number of job delays. The values of CR of the delivery dates e_j for each job j is given by: $CR \in \{2, 3.5, 5\}$. For each combination of n, m, processing time interval, deteriorating effect interval and congestion ratio of due dates, 25 instances were generated.

4.1. Analysis of parameters of the PILS algorithm

When an algorithm is calibrated on the same instances that will be later used for computational results and comparisons, we risk having biased or over fitted results. In order to remedy this problem we generate a calibration benchmark of 36 random instances with the possible combinations between n, m, d_{jk}, p_{jk} and CR, respectively ($2 \times 3 \times 2 \times 1 \times 3 = 36$). For calibration, the algorithms were executed 5 times for each instance. Calibration was performed using one factor at a time. For each parameter, a set of values was tested by setting the others.

The PILS algorithm has three input parameters to be defined as bellow. τ: controls the maximum execution time for algorithm; t: defines the perturbation length; ϕ: defines the number of solutions created by the constructive procedure. We use a natural stopping criterion (τ) which we set at a predefined elapsed CPU time following the expression $0.08 \times \frac{n}{m}$ seconds. In this expression, the CPU time is influenced by the number of jobs n and the number of machines m. For n fixed, the execution time increases as the value of m decreases.

In the ILS/D algorithm [Santos et al., 2022], the ILS method is executed for s subproblems and g generations, with the parameter τ for the ILS set to the CPU time of $0.08 \times \frac{n}{m}$ seconds. Therefore, to use the same computational budget for the PILS algorithm, it was necessary to include these values to the maximum execution time of the PILS algorithm. Therefore, the parameter τ for the PILS algorithm was set to the CPU time: $s \times g \times 0.08 \times \frac{n}{m}$ seconds. In the calibration of parameters of the ILS/D algorithm it was defined that $s = 40$ and $g = 50$. Thus, each execution of the PILS algorithm spent $50 \times 40 \times 0.08 \times \frac{n}{m}$ seconds.

The three tested values for the parameter ϕ are $\{20; 40; 60\}$ and the two tested values for the parameter t are $\{70\%\text{m}; 90\%\text{m}\}$. Note that there are a total of 6 parameter combinations to evaluate. As 5 executions were carried out for the 36 instances, totaling 1080 executions.

The six configurations tested for PILS do not present significant differences when compared to each other, presenting the following hyper-volume averages: 0.726, 0.700, 0.701, 0.711, 0.723, 0.720, respectively. Although the configurations are statistically equal, we choose the configuration 05, which presents a high value for the average of the hyper-volume (0.723) and lower standard deviation (0.12). Therefore, for the final experiments of PILS, we use the parameters used in the configuration 05: $\tau = 50 \times 40 \times 0.08 \times \frac{n}{m}$ seconds, $\phi = 60$ and $t = 70\%$ of m.

4.2. Comparison between the results of the ILS/D and PILS algorithms

After calibrating the developed PILS algorithm, we need to compare the performance of the algorithms ILS/D and PILS. The two algorithms were tested in 900 instances proposed by Ruiz-Torres et al. [2015] and were run 5 times with the same elapsed CPU time as stopping criterion. The indicator used to compare the results of the two algorithms is the HV and the reference point is the largest solution found for each of the objectives. All solutions found by the two algorithms

https://proceedings.science/p/157513?lang=pt-br

were used to identify the minimum and maximum values of each objective in the estimated Pareto fronts. The values of the HV indicator were normalized between 0 and 1.

Table 2 shows the average HV encountered by the compared algorithms for all instances. The average HV are grouped by instance size ($n \times m \times d_{jk} \times CR$), where for each combination we have 25 instances. Bold values represent best results. We can see that the ILS/D algorithm presents the best results for the test problems in groups 100×5, 200×20, 200×10 and 200×5. On the other hand, the PILS algorithm presents better results for the test problems 100×10. And for the test problems 100×20, the two algorithms show similar results. The results suggest that the ILS/D algorithm performs better than the PILS algorithm, because the ILS/D algorithm found the best result in most instances (80% of cases).

Table 2: Results of the average HV indicator for the ILS/D and PILS algorithms.

n	m	d_{jk}	CR	ILS/D	PILS	n	m	d_{jk}	CR	ILS/D	PILS
100	20	[1%, 5%]	2	**0.6309**	0.5802	200	20	[1%, 5%]	2	**0.7975**	0.7731
			3.5	**0.6544**	0.6293				3.5	**0.8278**	0.8035
			5	0.6267	**0.6529**				5	**0.7149**	0.6880
		[5%, 10%]	2	**0.6476**	0.6331			[5%, 10%]	2	**0.8666**	0.8150
			3.5	**0.7062**	0.6824				3.5	**0.7800**	0.7379
			5	0.6533	**0.7085**				5	**0.7158**	0.6527
	10	[1%, 5%]	2	**0.8495**	0.7972		10	[1%, 5%]	2	0.9175	**0.9218**
			3.5	**0.7820**	0.7768				3.5	**0.6544**	0.6458
			5	0.7357	**0.7724**				5	**0.6645**	0.6213
		[5%, 10%]	2	0.8552	**0.8703**			[5%, 10%]	2	**0.7577**	0.7548
			3.5	0.7352	**0.7874**				3.5	**0.7616**	0.7141
			5	0.7334	**0.7773**				5	**0.7544**	0.7302
	5	[1%, 5%]	2	**0.7924**	0.7363		5	[1%, 5%]	2	**0.7614**	0.6387
			3.5	**0.7521**	0.7225				3.5	**0.7427**	0.6267
			5	**0.7468**	0.7038				5	**0.7261**	0.6070
		[5%, 10%]	2	**0.7604**	0.7356			[5%, 10%]	2	**0.7948**	0.7372
			3.5	**0.7343**	0.7199				3.5	**0.8011**	0.7544
			5	**0.7316**	0.7153				5	**0.7989**	0.7575

We verified that the data does not follow a normal distribution using the *Shapiro-Wilk* test [Shapiro e Wilk, 1965]. The *Kruskal-Wallis* test is the non-parametric method alternative to ANOVA for one factor [Kruskal e Wallis, 1952]. In order to validate the results obtained by the two algorithms and check if the observed differences are statistically significant, we apply the *Kruskal-Wallis* non-parametric test for the groups of instance size ($n \times m$), where for each combination we have 150 instances, that is, 150 samples and only the average result was considered.

The value obtained for the p-value in the *Kruskal-Wallis* test was less than 0.05 for 100×5, 200×20, 200×10 and 200×5. The results suggest there is statistical difference between the results of the HV indicator for compared algorithms at 95% confidence level. On the other hand, for the test problems 100×20 and 100×10, the result p-value obtained in the *Kruskal-Wallis* test was bigger than 0.05, which shows that there is no difference between the algorithms for these instances.

To identify a significant difference between the results of paired algorithms, the *Box-plot of Kruskal-Wallis groups* test was done with 95% of confidence level. Figure 2 shows the test *Box-plot of Kruskal-Wallis groups* from the comparison of the algorithms ILS/D and PILS for the test problems 100×20 (Figure 2-a), 100×10 (Figure 2-b), 100×5 (Figure 2-c), 200×20 (Figure 2-d), 200×10 (Figure 2-e) and 200×5 (Figure 2-f). In Figure 2 (c), (d), (e) and (f), since the interval for the ILS/D algorithm is classified as group a and the interval for the PILS algorithm is

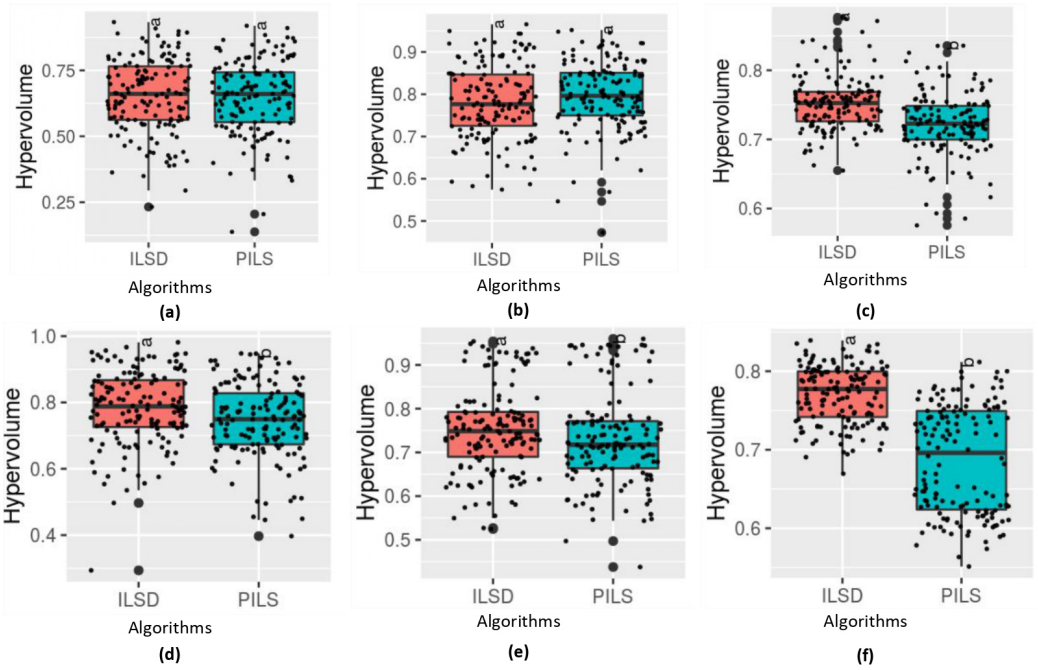


Figure 2: Box plot results (*Kruskal-Wallis Test*) of the ILS/D and PILS algorithms for the HV indicator, of the test problems $n \times m$.

classified as group b, the mean of the PILS algorithm is significantly different. That is, the ILS/D algorithm developed showed superior performance overcoming the PILS algorithm. However, for the test problems 100×20 and 100×10, we can see in Figure 2 (a) and 2 (b), that the algorithms do not present significant differences when compared to each other, since all the settings are part of the group a. In general, the result of *Box-plot of Kruskal-Wallis groups* test suggests that ILS/D is statistically better than the other algorithm in a pairwise comparison.

A graphical analysis of the algorithms convergence was performed. Six instances ($n \times m$) were randomly selected to illustrate the results. Figure 3 shows the results of the best execution among the five executions of the selected instances, for the test problems 100×20 (Figure 3-a), 100×10 (Figure 3-b), 100×5 (Figure 3-c), 200×20 (Figure 3-d), 200×10 (Figure 3-e) and 200×5 (Figure 3-f). Figure 3 suggests that the algorithms ILS/D and PILS have a good convergence towards the Pareto front. On the other hand, Figure 3 shows that a limitation of the PILS algorithm is that there is no control of diversification in the Pareto front approximation. Although the PILS algorithm obtains a better hyper-volume value for some instances, we can verify that there is not much extension in the Pareto front approximation. Therefore, it can be observed that the Pareto front approximation of ILS/D is better than the one of PILS, probably because of benefiting from the implicit parallelism of a population-based search and decomposition.

4.3. Algorithms validation

The main goal of this section is to validate the results of PILS and ILS/D algorithms for the addressed problem. For that, these algorithms are executed for forty-eight small sized instances proposed in this work. These instances use combinations of the following n and m values: $n \in \{6, 7\}$ and $m \in \{2, 3, 4\}$, additionally, $n \in \{8\}$ and $m \in \{2, 3\}$. By combining these parameters with the other parameters (p_{jk}, d_{jk}, CR), we generate 48 instances in total.

https://proceedings.science/p/157513?lang=pt-br

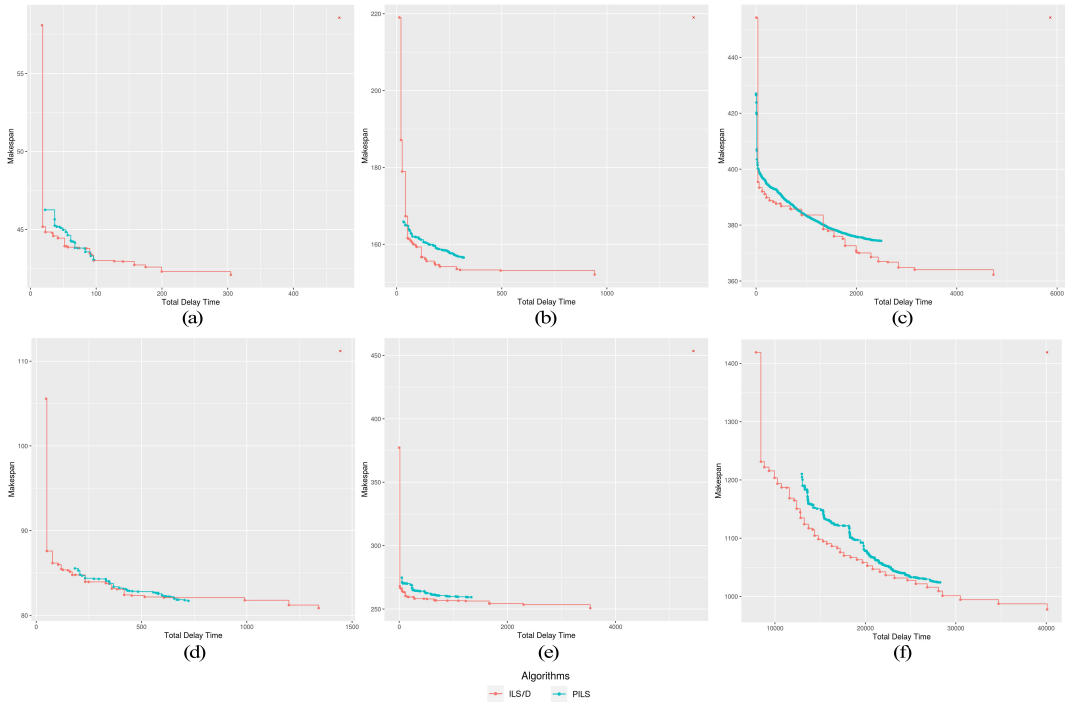


Figure 3: Estimates of the Pareto front obtained with the two algorithms for the test problems $n \times m$.

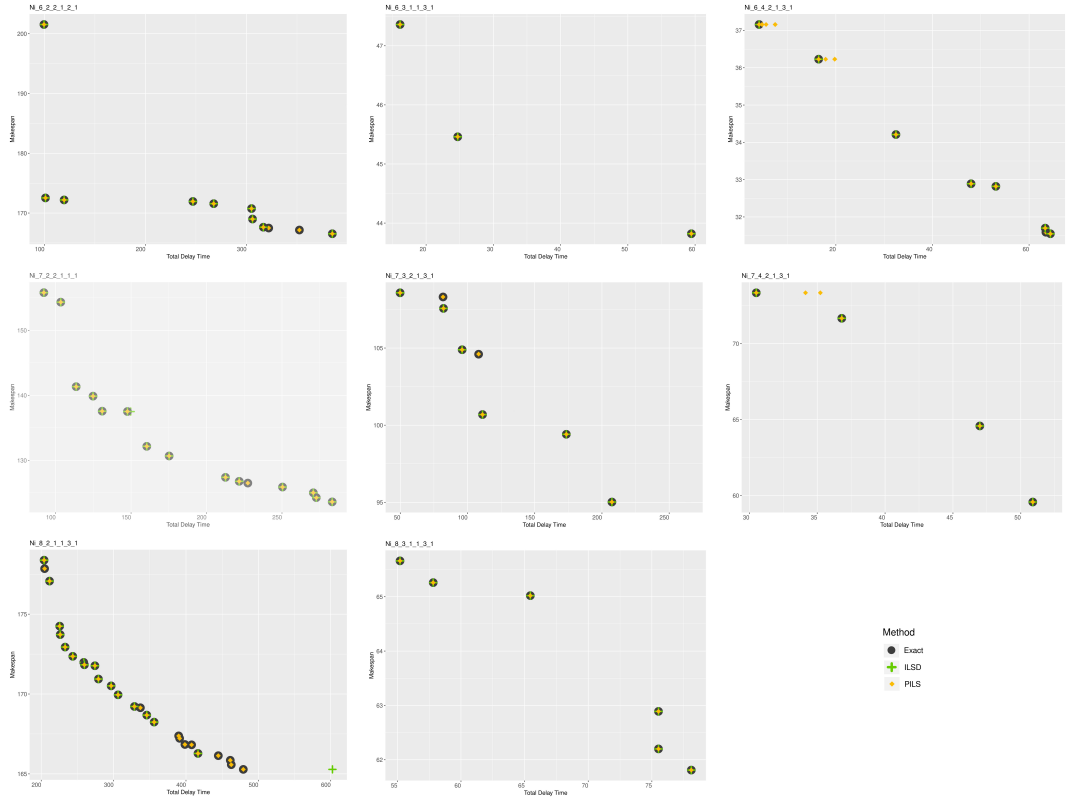


Figure 4: Pareto-optimal front (black circles) and the fronts returned by ILS/D (green '+') and PILS (yellow rectangle) algorithms.

The results of the algorithms are compared with the optimal values obtained by enumeration technique. By counting all the exacts points of Pareto-optimal front obtained by the enumeration technique, we verified that, on average, the ILS/D and PILS algorithms find 92% and 99% equal points, respectively. Figure 4 shows the results obtained by enumeration technique and by the PILS and ILS/D algorithms for eight instances randomly selected of each combination n and m. From the illustration we can see that the results achieved by the algorithms are similar to the enumeration technique. These results confirm the validity of the PILS and ILS/D algorithms for the problem, showing that they are able to converge to correct Pareto-optimal front solutions.

4.4. Discussion

In general, the results suggest that the PILS algorithm presents approximate results of the ILS/D algorithm, specifically, for the test problems 100×20 and 100×10, in the pairwise comparison. In some cases, the PILS algorithm achieves better average hyper-volume than the ILS/D algorithm. In addition, the PILS algorithm presents a good convergence of the Pareto front approximation and the PILS algorithm can obtain exact results for small sized instances.

However, the results obtained show that a distinguishing mark of the ILS/D algorithm is the sectorization of search in the Pareto front approximation, even when it does not find a better value for the hyper-volume indicator. In this case, the decomposition does not allow the solutions found by the local search structures to quickly converge to a restricted search space (local optimal). This makes it possible to explore a greater extent of the Pareto front approximation.

Therefore, we can conclude that extending the algorithm based on the local search meta-heuristic to a hybrid approach, which also uses a population-based meta-heuristic, allows for finding a better Pareto front approximation. This is probably due to the benefit of employing a neighborhood-based local search and because of benefiting from the implicit parallelism based on population and decomposition like the MOEA/D algorithm.

The data of the problem, as well as all the solutions and results obtained are available on the website `http://vivian.tmserver.com.br/scheduling_ilsdxpils`.

5. Conclusions

This work addressed the unrelated parallel machine scheduling problem with sequence-dependent machine deterioration. We use the multi-objective perspective, with two optimization criteria: the makespan minimization and the minimization of TDT, simultaneously. To solve the problem, a heuristic algorithm is developed: PILS. The PILS algorithm was compared with the ILS/D algorithm. The algorithms were evaluated on 900 benchmark instances from the literature. The experiment results indicate that the heuristics ILS/D presents superior results compared to the PILS algorithm in most cases. The main conclusion that can be drawn from this study is that the decomposition and aggregation approach can be a helpful way to extend ILS to multi-objective problems. Furthermore, we can conclude that hybrid algorithms can obtain higher quality solutions in comparison with other existing meta-heuristics in literature.

Acknowledgements

The authors would like to thank the institutions UFMG, UFVJM, IFNMG for the support; and the program for qualification of the Brazilian institution IFNMG - Campus Pirapora (Programa de Bolsas para Qualificação de Servidores - PBQS, in Portuguese), for the support and incentive in the development of this work.

This work has been supported by the Brazilian agencies (i) National Council for Scientific and Technological Development (CNPq), Grant no. 312991/2020-7; (ii) Coordination for the Improvement of Higher Education Personnel (CAPES), and (iii) Foundation for Research of the State of Minas Gerais (FAPEMIG, in Portuguese), Grant no. APQ-01779-21. MINDS Laboratory - `https://minds.eng.ufmg.br/`.

https://proceedings.science/p/157513?lang=pt-br

References

Deb, K., Pratap, A., Agarwal, S., e Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Geiger, M. J. (2006). The pils metaheuristic and its application to multi-objective machine scheduling. *Multicriteria Decision Making and Fuzzy Systems Theory, Methods and Applications*, p. 43–58.

Geiger, M. J. (2008). Foundations of the pareto iterated local search metaheuristic. *arXiv preprint arXiv:0809.0406*.

Geiger, M. J. (2009). Improvements for multi-objective flow shop scheduling by pareto iterated local search. *arXiv preprint arXiv:0907.2993*.

Geiger, M. J. (2011). Decision support for multi-objective flow shop scheduling by the pareto iterated local search methodology. *Computers & industrial engineering*, 61(3):805–812.

Gomes, H. C., das Neves, F. d. A., e Souza, M. J. F. (2014). Multi-objective metaheuristic algorithms for the resource-constrained project scheduling problem with precedence relations. *Computers & Operations Research*, 44:92–104.

Kruskal, W. H. e Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621.

Lourenço, H. R., Martin, O. C., e Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, p. 320–353. Springer.

Paquete, L., Chiarandini, M., e Stützle, T. (2004). Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study. In *Metaheuristics for multiobjective optimisation*, p. 177–199. Springer.

Rego, M. F., Souza, M. J. F., Coelho, I. M., e Arroyo, J. E. C. (2014). Multi-objective algorithms for the single machine scheduling problem with sequence-dependent family setups. In *Soft Computing in Industrial Applications*, p. 117–127. Springer.

Ruiz-Torres, A. J., Paletta, G., e Pérez, E. (2013). Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects. *Computers & Operations Research*, 40(8):2051–2061.

Ruiz-Torres, A. J., Paletta, G., e Perez-Roman, E. (2015). Maximizing the percentage of on-time jobs with sequence dependent deteriorating process times. *International Journal of Operations Research and Information Systems (IJORIS)*, 6(3):1–18.

Santos, V. L. A., Carvalho, T. F. M., de Assis, L. P., Weiss-Cohen, M., e Guimarães, F. G. (2022). Multi-objective iterated local search based on decomposition for job scheduling problems with machine deterioration effect. *Engineering Applications of Artificial Intelligence*, 112:104826. ISSN 0952-1976.

Shapiro, S. S. e Wilk, M. B. (1965). An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611.

Vargas, D., Lemonge, A., Barbosa, H., e Bernardino, H. (2016). Um algoritmo baseado em evolução diferencial para problemas de otimização estrutural multiobjetivo com restrições. *Revista Internacional de Métodos Numéricos para Cálculo y Diseño en Ingeniería*, 32(2):91–99.

Xu, X. Q. e Lei, D. M. (2014). A parallel iterated local search for two-agent flow shop scheduling. In *Advanced Materials Research*, volume 926, p. 3476–3484. Trans Tech Publ.

Zhang, Q. e Li, H. (2007). Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6):712–731.

Zitzler, E. e Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271.

https://proceedings.science/p/157513?lang=pt-br