Sigma J Eng Nat Sci, Vol. 44, No. 4, pp. 2626–2640, August, 2026

Sigma Journal of Engineering and Natural Sciences
Web page info: https://sigma.yildiz.edu.tr
DOI: 10.14744/sigma.2026.2119

Research Article

Evaluating the performance of Iterated Greedy methods for the distributed permutation flowshop with sequence-independent setup times

Clarissa TARARAM de LAURENTYS[1], Marcelo Seido NAGANO[1,*], Fernando Luis ROSSİ[2]

[1]*Department of Production Engineering, University of São Paulo, São Paulo, 05508-010, Brazil*
[2]*Department of Management, Federal Institute of São Paulo, São Paulo, 01109-010, Brazil*

ARTICLE INFO

Article history
Received: 14 January 2025
Revised 22 March 2025
Accepted: 31 May 2025

Keywords:
Distributed Flowshop Scheduling; Iterated Greedy; Makespan; Sequence-Independent Setup Time

ABSTRACT

Given the strategic needs of meeting demand and minimizing production costs to remain competitive, companies have invested in more than one plant in different geographical spaces. Therefore, in addition to scheduling, organizations also deal with the designation of which factory will produce certain products. In more recent studies, the integration of distribution and scheduling has mostly driven the advancement of strategies to improve the overall performance of factory operations. The research addresses the distributed flowshop scheduling problem under sequence-independent setup times, a constraint that has not been widely studied in the literature. Sixteen Iterated Greedy methods were developed, combining multiple initialization, destruction, and reconstruction approaches, which makes it a comprehensive analysis of IG performance for this problem. The IG methods combine four initialization approaches, two destruction and two construction procedures. Experiments were conducted to examine the impact of two parameter settings in order to determine the optimal configuration for each method. The adopted performance objective is the total duration of production, commonly referred to as makespan. Computational experiments conducted on benchmark datasets revealed that IG7, which combines DESTR_2, NEH_3, and RECONST_1, consistently outperformed other variants, achieving an average ARPD improvement of 0.938. Results from all computational experiments suggest that the novel destruction procedure significantly influences the outcomes, with methods outperforming others across all analyzed instances.

Cite this article as: Tararam De Laurentys C, Nagano MS, Rossi FL. Evaluating the performance of Iterated Greedy methods for the distributed permutation flowshop with sequence-independent setup times. Sigma J Eng Nat Sci 2026;44(4):2626–2640.

INTRODUCTION

The flowshop scheduling problem is applicable across a range of production environments, including the chemical industry, aeronautical parts manufacturing, waste treatment [1], cider production [2], electronic circuits [3], concrete block manufacturing [4], earthmoving operations [5], iron and steel production and even the service sector [6]. Since the 1960s, significant efforts have been made to integrate setup duration into scheduling problems. Advances

***Corresponding author.**
*E-mail address: drnagano@sc.usp.br

This paper was recommended for publication in revised form by Editor-in-Chief Ahmet Selim Dalkilic

Published by Yıldız Technical University Press, İstanbul, Turkey
© Author. This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

in scheduling methodologies, such as genetic algorithms, simulated annealing, and decision support systems, have been instrumental in addressing these challenges. [7-9]. Furthermore, researchers like [10-13] have emphasized the importance of incorporating job sequence-independent setup time (SIST) into scheduling, highlighting its relevance to real-world applications.

The evolution towards distributed flowshop has been driven by a growing need to optimize production in complex and decentralized environments given the context of globalization. The article by [14] was the first to name the extension of the classic flowshop problem to the multi-factory environment as the Distributed Permutational Flowshop Problem (DPFSP). Distributed manufacturing in general is seen to reduce production costs and risks. The DPFSP consists of assigning n jobs in F factories. From there, the decision maker establishes the production sequence of jobs in each factory according to information about the particularities of the production environment.

Different conditions and restrictions were applied to DPFSP, for example, the no-wait flowshop condition was studied and presented by [15] using a branch-and-cut approach, [16] created a competitive memetic algorithm for DPFSP multiobjective and [17] proposed differential evolution metaheuristics for distributed flowshop with limited buffer. One of the major constraints studied for the DPFSP problem is the sequence dependent setup times (SDST). For instance, [18, 19] introduced discrete approaches utilizing bee colonization techniques. [20] developed a model using mixed integer linear programming (MILP) and constraint programming (CP). [21] put forward an iterated greedy algorithm, while [22] devised a metaheuristic employing a differential flight strategy.

More recently, cooperative scheduling strategies, such as those proposed by [23], and flexible assembly approaches highlighted by [24] demonstrate significant potential for complementing Iterated Greedy (IG) methods, particularly in handling complex constraints like sequence-independent setup times. Adaptive genetic algorithms, as detailed by [25], offer an interesting point of comparison, especially in scenarios requiring diverse search capabilities to optimize makespan. Furthermore, benchmarking results from studies such as [26, 27] provide a framework for validating IG performance in minimizing maximum tardiness and improving efficiency under hybrid setups. The systematic review by [28] underscores persistent gaps in distributed scheduling research, including the need to address integrated logistics and non-uniform factory setups.

Despite the rapid growth of literature on DPFSP, studies that incorporate the sequence-independent setup time (SIST) restriction were not found. As per [29] complete review of the distributed scheduling literature, within the constrained DPFSP only three types of setup time constrain were found. Which are: setups depending on the job and machine, sequence dependent setup times and machine and sequence-dependent setup time.

The study extends previous distributed scheduling research by exploring sequence independent setup times, a constraint largely overlooked in prior work, despite its relevance in real-world production environments. It emphasizes the increasing need for integrated scheduling strategies to handle complex production across geographically distributed facilities. Prior studies focused on sequence-dependent setup times (SDST) or multiobjective optimization, while this work uniquely targets SIST with a detailed evaluation of IG methods. By systematically testing 16 variations of IG, the research advances understanding of how different procedural choices (e.g., destruction, initialization) impact scheduling efficiency, providing actionable insights for factory management. The key highlights and novelties of this research are outlined below:

- Investigated sequence-independent setup times (SIST) in distributed flowshop scheduling, a less explored area.
- Developed 16 Iterated Greedy (IG) methods with diverse strategies for initialization, destruction, and reconstruction.
- Identified IG7 as the best-performing method, optimizing makespan effectively.
- Demonstrated the critical role of destruction procedures, with DESTR_2 achieving superior results.
- Provided practical insights into parameter tuning and scalability for large-scale problems.

The Distributed Permutational Flowshop Problem (DPFSP) is characterized by a collection N of n jobs that need to be processed across a group G of F factories. [14]. The assumptions underlying this study include each factory having an identical set M of m machines (similar to a standard PFSP), with fixed processing and setup times. All factories are capable of handling all jobs, and once a job j from set N is assigned to a factory F in set G, it cannot be moved to another factory; it must be completed in its initially assigned factory.

Within each factory, the problem is treated as PFSP considering the sequence-independent setup time (SIST) restriction. The PFSP-SIST scheduling problem involves a set of jobs $J = \{J_1, J_2,\ldots, J_n\}$, which constitutes a production batch, which goes through the production process that contains a set of machines in series $M = \{M_1, M_2,\ldots, M_m\}$. All jobs have the same path and perform m operations, one on each machine. Each job starts on the first machine and ends its cycle on the last machine. It is considered that $p_{i,j}$ is the processing time and that $s_{i,j}$ is the setup time of job J_j on machine M_i. [13].

The assumptions used are the following: all jobs and machines are available from the initial moment, each machine processes only one job at a time and each job is processed only once by the same machine, the processing and setup time are strictly positive, each of which must be scheduled, and each machine has the same sequence of jobs, therefore only permutational scheduling is allowed.

The objective of DPFSP is to find the optimal sequence that minimizes the makespan of the set G of factories,

C'_{max}. Within each factory the objective is to find the optimal sequence that minimize the makespan, C_{max}, under the selected assumptions and under the constraints of machine availability during the real scheduling time of the entire batch. [13]. The calculation of C_{max} e C'_{max} are presented in equations 2 and 3, respectively.

$$C_{i,j} = max_{1\leq j\leq n}\{C_{i,j-1} + s_{ij}, C_{i-1,j}\} + p_{i,j}, \quad i = 2, \dots, m; \; j = 2, \dots, n \tag{1}$$

$$C_{max} = C_F = max_{1\leq j\leq n}\{C_{m,j}\}, \; F = 2, \dots, G \tag{2}$$

$$C'_{max} = max_{1\leq F\leq G}\{C_1, C_2, \dots, C_G\} \tag{3}$$

The following notation is used throughout the paper to describe the Distributed Flowshop Scheduling Problem (DFSP) and the proposed Iterated Greedy (IG) methods:

- n: Total number of jobs.
- m: Total number of machines.
- f: Total number of factories.
- J_i: Job i, where i=1,2,…,n; i=1,2,…,n.
- M_j: Machine j, where j=1,2,…,m; j=1,2,…,m.
- F_k: Factory k, where k=1,2,…,f; k=1,2,…,f.
- $p_{i,j}$: Processing time of job i on machine j.
- $C_{i,j,k}$: Completion time of job i on machine j in factory k.
- C_{max}: Makespan, the total time required to complete all jobs across all factories.
- d: Number of jobs removed during the destruction phase.
- T: Temperature parameter in the acceptance criterion for simulated annealing.
- NEH_1, NEH_2, NEH_3, NEH_4: Initialization strategies based on the Nawaz-Enscore-Ham heuristic.
- DESTR_1, DESTR_2: Destruction strategies used to modify the current solution.
- RECONST_1, RECONST_2: Reconstruction strategies for reinserting jobs into the schedule.
- ARPD: Average Relative Percentage Deviation, a metric for comparing the performance of scheduling methods.

MATERIALS AND METHODS

Materials

The methodology in this study addresses the problem of distributed flowshop scheduling with sequence-independent setup times (SIST). It starts with defining the research problem, focusing on optimizing makespan in distributed manufacturing environments. Multiple Iterated Greedy (IG) algorithms are developed, using initialization procedures such as NEH variants and specific destruction and reconstruction mechanisms. These IG configurations, totaling 16 variations, are implemented and tested on benchmark datasets to assess their performance. The evaluation focuses on metrics like makespan to determine algorithm efficiency. The final step involves analyzing the results to identify the most effective configurations. This process is summarized in the flowchart (Fig. 1), which outlines each step, from problem definition to performance analysis, and highlights the sequence of the study's methodology.

Iterated Greedy (IG) algorithms have stood out in the literature for their simplicity and excellent performance for different variants of permutational flowshop problems. Unlike complex methods, IG stands as a straightforward iterative search method devoid of memory and elaborate structures. Its outcomes are easily reproducible and transferrable to various other problem domains. [30]. Some authors have already presented IG methods for DPFSP with good results [30-32]. It has six main steps. In the first, some constructive method is used to generate a complete initial solution, which, in the second stage, is subjected to a local search to optimize it. Then, the initial solution is destroyed, and it is reconstructed. Another local search is performed and, finally, an acceptance criterion is used to evaluate the new reconstructed solution. These last four steps are executed iteratively, so the process repeats until some stopping criterion is satisfied. An IG step-by-step sketch is provided in Figure 2. In the following sub-sections, the selected procedures for each step are explained.

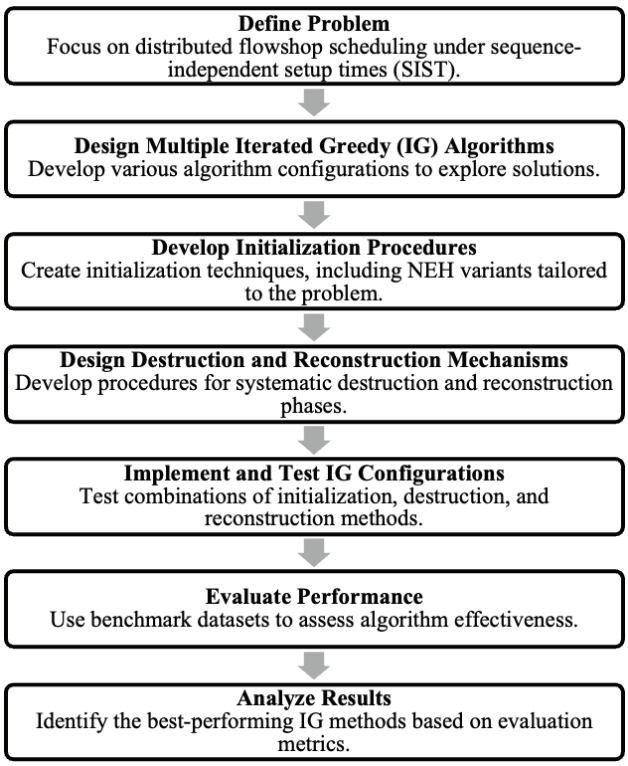

Figure 1. The flowchart outlines each step, from problem definition to performance analysis, and highlights the sequence of the study's methodology.

```
IG:
# Step 1: initialization procedure
π0 = Generate_Initial_Solution

# Step 2: Optimize solution
π = Local_Search (π0)

while (Stopping_Criterion not satisfied)

        # Step 3: Destroy solution
        πD = Destruction (π)

        # Step 4: Reconstruct solution:
        πR = Reconstruction (πD)

        # Step 5: Optimize solution
        πLS = Local_Search (πR)

        # Step 6: Evaluate new solution
        π = Acceptance_Criterion (πLS, π)
```

Figure 2. IG step-by-step sketch, identifying each of the six main steps.

Initialization Procedure

The IG method uses a high-performance constructive heuristic to generate the initial solution and, therefore, different variants of the NEH [33] are employed in most of the literature. [14] introduced NEH2 as a variant of the DPFSP problem, inspired by the original NEH method but distinguished by the incorporation of an assignment rule. This approach involves inserting each job into every conceivable position across all factories to identify the configuration that minimizes the makespan for the entire set of factories.

[30] included a simple addition to this method: following the insertion of an job in the best position across all F factories, the preceding or succeeding job (randomly chosen) is removed and evaluated in all positions within the same factory. This method, called NEH_en, brought better results with a statistically significant difference when compared to NEH2. For this article, both initialization methods will be tested with three additions: inclusion of sequence-independent setup time analysis, inclusion of priority rules in the first stage of the NEH and a tiebreaker strategy in the last stage.

[34] proposed and tested 48 heuristics based on the NEH for the sequence-independent setup time permutation flowshop problem (PFSP-SIST). The best results in relatively low computation time were obtained by Alg27 and Alg31. Both use in the fourth stage of the NEH the adapted tie-breaking strategy procedure proposed by [35] to the SIST constrain (TBFFs). This strategy unties the partial sequences by choosing the position that presents the minimum idle time. Alg27 combines this tie-breaking strategy with the use in the first stage of the NEH of the adapted priority rule proposed by [36] to the SIST constrain. This priority rule orders jobs by the non-increasing sum of the average, standard deviation, and the absolute value of the skewness (SKE), which measures the asymmetry of a probability distribution, of processing and setup times (PR7). Alg27 combines TBFFs with a similar priority rule, that orders jobs by the non-increasing sum of process time's average, standard deviation, and SKE, and sum of setup time's average and SKE (PR8).

For this article, the NEH2 and NEH_en are adapted by including TBFFs as a tie-breaking strategy and PR7 and PR8 as priority rules based on the good results shown in [34] study for the SIST constrain. Table 1 summarizes the initialization methods used (NEH_1, NEH_2, NEH_3 and NEH_4) and initialization components (NEH variant, priority rule and tie-breaking strategy).

Local Search

The initial solution is refined using the local search procedure. [30] proposed a search procedure, called LS3, which accounts for both factory assignment and job sequencing dimensions concurrently. In this procedure, an job is randomly extracted from the factory with the largest makespan and is inserted in all possible positions in all factories, including the one that was extracted. The job is allocated to the position that generates the lowest makespan in the set of factories and the process proceeds (without repetitions) up to all jobs in the highest makespan factory have been examined.

The results obtained by [30] show that, in addition to being faster in relation to CPU time requirements since it

Table 1. Evaluated IG initialization procedures

Initialization procedure	Initialization Components		
	NEH variant	Priority rule	Tie-breaking strategy
NEH_1	NEH2	PR_8	TB_{FFs}
NEH_2	NEH2	PR_7	TB_{FFs}
NEH_3	NEH2_en	PR_8	TB_{FFs}
NEH_4	NEH2_en	PR_7	TB_{FFs}

doesn't handle two neighborhoods in a loop, LS3 is statistically better when compared to the VND procedure, proposed by [14]. Therefore, the LS3 search was selected for this study.

Destruction

The destruction procedure involves the elimination of a specified number, denoted as d, of jobs from the initial solution. Prior research demonstrated that adjusting d allows algorithms to balance intensification and diversification. [30]. [32] proposed a destruction procedure in which jobs are removed one from each factory, from the one with the largest makespan to the smallest. If d > F, the remaining jobs (d - F) are randomly removed from the factories. [30] proposed a new destruction procedure, in which half of the jobs to be removed (d / 2) are removed from the highest makespan factory, while the remaining half is drawn randomly from the other factories (F - 1). Although the new procedure presents statistically better results than the previous one, both will be tested to evaluate the influence of the SIST restriction on the DPFSP problem.

The d value of the total jobs to be removed will be tested as a parameter with two fixed values (4 and 5), as per their better performance obtained in the research conducted by [30]. Table 2 summarizes the destruction procedures evaluated, where DESTR_1 denotes the order for job removal by descending order of factories using the makespan value and DESTR_1 removes the jobs from the highest makespan factory.

Reconstruction

The most common reconstruction process used in the literature [30-32, 37] is similar to the NEH procedure. Each of the d jobs removed in the destruction phase are tested in all positions of all factories to select the one that produces the lowest makespan of the set of factories. In accordance with NEH2_en, an addition to the method is proposed by [30], following the insertion of an job in the optimal position among all F factories, the preceding or succeeding job (randomly chosen) is extracted and evaluated in all positions within the same factory. Table 3 summarizes the RECONST_1 and RECONST_2 reconstruction procedures evaluated, which are based on NEH2 and NEH2_en, respectively.

Acceptance Criterion

The acceptance criterion directly influences the equilibrium between intensification and diversification in the pursuit of the optimal solution. In addition to accepting those solutions that actually present improvements in terms of the makespan value found, it is possible to accept a worse solution than the current one. The aim of this criterion is to enable the search to shift away from the local optima already discovered and toward other potential optima that may yield better outcomes. A criterion popularly used in these cases is called simulated annealing, proposed by [37], in which a new solution is accepted if it is superior or equal; otherwise, acceptance is determined by the condition outlined in Equation 4.

$$random \leq \exp\left(-\frac{\left(C'_{max}(\pi_{LS}) - C'_{max}(\pi)\right)}{Temperature}\right) \quad (4)$$

$$Temperature = T * \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{i,j} + s_{i,j}}{n*m*10}, 0 < T < 1, \quad (5)$$

where random represents a number uniformly distributed between 0 and 1 and T is a parameter that needs adjustment. According to the results obtained in the research conducted by [30], the values of T = 0.1 and T = 0.2 have been shown in past work to provide a good balance between exploration and convergence, therefore, will be used in the computational experiments.

Stopping Criterion

The IG steps are repeated iteratively until the stopping criterion is reached. There are different stopping criteria used in the literature, most of which use some computational time factor related to multiplying the number of jobs by the number of machines [37-39], as IG requires more time for large instances. For this article, the stopping criterion will be defined by Equation 6.

$$Time\ limit = n * \frac{m}{2} * C \quad (6)$$

C is a parameter that needs adjustment. For this study the C value was set at 40 milliseconds, considering the good results obtained by [30].

Table 3. Evaluated IG reconstruction procedures

Reconstruction procedure	Procedure
RECONST_1	Similar to NEH2
RECONST_2	Similar to NEH2_en

Table 2. Evaluated IG destruction procedures

Destruction Procedure	Jobs' removal order
DESTR_1	Descending order of factories in relation to makespan value
DESTR _2	Half removed from the highest makespan factory, while the remaining half is drawn from the other factories.

Table 4. Evaluated IG methods

IG method	Initialization Procedure	Destruction Procedure	Reconstruction Procedure
IG 1	NEH_1	DESTR_1	RECONST_1
IG 2	NEH_2	DESTR_1	RECONST_1
IG 3	NEH_3	DESTR_1	RECONST_1
IG 4	NEH_4	DESTR_1	RECONST_1
IG 5	NEH_1	DESTR_2	RECONST_1
IG 6	NEH_2	DESTR_2	RECONST_1
IG 7	NEH_3	DESTR_2	RECONST_1
IG 8	NEH_4	DESTR_2	RECONST_1
IG 9	NEH_1	DESTR_1	RECONST_2
IG 10	NEH_2	DESTR_1	RECONST_2
IG 11	NEH_3	DESTR_1	RECONST_2
IG 12	NEH_4	DESTR_1	RECONST_2
IG 13	NEH_1	DESTR_2	RECONST_2
IG 14	NEH_2	DESTR_2	RECONST_2
IG 15	NEH_3	DESTR_2	RECONST_2
IG 16	NEH_4	DESTR_2	RECONST_2

Performance Metrics

In total, 16 IG variations were evaluated for the DPFSP-SIST, each one being tested for 2 values of T and 2 values of d. A total of 16 IG variations were evaluated for the DPFSP-SIST, with each method tested using two values for T and two values for d. Table 4 summarizes the composition of the evaluated methods, detailing the initialization procedure (NEH_1 to NEH_4), destruction procedure (DESTR_1, DESTR_2), and reconstruction procedure (RECONST_1, RECONST_2). The computational complexity of these variants is determined by their components, including initialization, destruction, reconstruction, and local search. As an example, simpler variants (e.g., DESTR_1 + NEH_1) favor lower computational overhead, while, more complex variants (e.g., DESTR_2 + NEH_3) are more computationally intensive due to randomness and dynamic job placement. This balance between complexity and performance highlights the importance of tailoring the algorithm to the size and characteristics of the scheduling problem, a topic further explored in the results section.

Each method will be tested using an adaptation of the instances from the [40] database, as it presents uniformly distributed values for processing and setup times, which results in instances that are more difficult to solve. The instances are adapted to the distribution condition, according to the identical factories' premises, each instance is used for F = {2, 3, 4, 5, 6, 7}. Processing times are uniformly distributed in the range [1,10] or [1,100] and setup times are uniformly distributed so that the maximum setup is 50%, 100% and 150% of the maximum processing time. To enable the execution of computational tests in a timely manner without compromising the variety and complexity of the instances, only the first 10 problems of each combination of m and n were used, with "large" instances being limited to n=150 and m=20. Considering this configuration, we have a total of 7,200 "small" instances and 2,160 "large" instances, adding up to a total of 9,360 instances. Furthermore, all tested methods use the Taillard acceleration [41] adaptation developed by [34] which includes the independent setup restriction.

The performance was evaluated using the Relative Percentage Deviation (RPD), calculated using Equation 7 where C_max (π_h) is the makespan obtained by a procedure for a given instance i and C_max^* is the best solution found among all methods compared for the instance i. As a result, the closer the solution obtained by method h is to the best solution for instance i, the lower the RPD will be.

$$RPD(C_{max}(\pi_h), i) = 100 * \left(\frac{C_{max}(\pi_h) - C^*_{max}}{C^*_{max}}\right); \quad h = \{1, 2, \dots, 16\}, \tag{7}$$

The Average Relative Percentage Deviation (ARPD) of method h considering N instances can be calculated using Equation 8.

$$ARPD\ (h\ , i) = \frac{\sum_{t=1}^{N} RPD\ (C_{max}(\pi_h), i)}{N}; \tag{8}$$

RESULTS AND DISCUSSION

The computational experiments were conducted using Python on a PC with an Intel Core i5-5200U 2.20 GHz CPU, 8 GB of RAM, and running Windows 10. The

outcomes of the experiments are consolidated in Tables 5-8, detailing the performance of each of the 16 algorithms according to the number of factories, jobs and machines in average and for each setup time distributions, 50%, 100% and 150%, respectfully. All outcomes are evaluated according to the number of factories, jobs and machines. When the setup time represents up to 100% of the processing time, IG8 presents the best results, 0.9286 and 0.9370, for 50% and 100%, respectively. However, in the case of representing 150%, that is, when the setup time is greater than the processing time, IG7 presents the best results, 0.9337. In average, IG7 presented the best results, 0.9382, followed by IG8, 0.9402.

Regarding the number of factories, it is possible to see that the complexity of the problem and, consequently, the ARPD of the solutions increase as the total number of factories increases. But, exclusively for methods with the DESTR_2 destruction type, when the total number of factories is a high value, f = 6 or f = 7, the ARPD decreases again. In other words, DESTR_2 helps the algorithm find better solutions even in complex scenarios with many factories. Regarding the number of jobs, the methods tend to find the best solutions and, therefore, lower ARPD values as the number of jobs increases. Therefore, IG presents better performance for large instances, with average results of 0.7323 compared to 1.2887 for small instances. Similarly, it is possible to see that the performance of the methods increases as the total number of machines increases. But for scenarios where the total number of machines is low, such as m = 2 or m = 3, performance is also good.

Although ARPD values are similar regarding the variation of factories, jobs and machines, the distribution of setup time in relation to processing time directly impacts the best method found. In cases where the setup is more relevant than the processing time, that is, it presents a distribution of 150%, the IG7 presents the best results for most scenarios. When the setup is less relevant, that is, distribution of 50% and 100%, the best method varies greatly depending on the scenario, and in only one case it is IG7 and on average it is IG8.

Regardless of the type of instance, it is noticeable that the main impact on the results is due to the destruction procedure choice, as represented in Figures 3 and 4. Figure 3 presents the mean plot for each of the 16 IG methods, and Figure 4 shows the ARPD for each for each initialization, destruction and reconstruction procedures. Procedure

Table 5. ARPD for each IG method for all instances, according to the number of factories, jobs and machines. The best results for each line are in bold

	IG 1	IG 2	IG 3	IG 4	IG 5	IG 6	IG 7	IG 8	IG 9	IG 10	IG 11	IG 12	IG 13	IG 14	IG 15	IG 16
f = 2	0.655	0.651	0.635	0.628	0.619	0.607	0.584	**0.581**	0.663	0.660	0.639	0.639	0.622	0.612	0.597	0.602
f = 3	1.043	1.065	1.030	1.023	0.872	0.890	**0.866**	0.875	1.053	1.071	1.048	1.043	0.893	0.911	0.883	0.872
f = 4	1.223	1.228	1.218	1.223	1.069	1.075	1.054	1.053	1.238	1.244	1.213	1.226	1.065	1.073	**1.047**	1.084
f = 5	1.443	1.447	1.402	1.431	**1.082**	1.115	1.090	1.089	1.460	1.459	1.434	1.440	1.094	1.101	1.119	1.086
f = 6	1.775	1.802	1.750	1.752	**1.045**	1.061	1.050	1.059	1.779	1.818	1.786	1.781	1.063	1.074	1.066	1.074
f = 7	2.104	2.076	2.055	2.059	0.999	1.022	0.986	**0.984**	2.111	2.057	2.086	2.056	1.004	1.014	0.996	0.991
mean	1.374	1.378	1.348	1.353	0.948	0.962	**0.938**	0.940	1.384	1.385	1.368	1.364	0.957	0.964	0.951	0.951
n = 15	1.659	1.650	1.647	1.617	**0.953**	1.001	0.973	0.977	1.685	1.687	1.683	1.659	0.986	1.010	1.018	1.019
n = 20	1.677	1.698	1.673	1.681	1.140	1.150	**1.121**	1.135	1.708	1.715	1.696	1.706	1.168	1.154	1.154	1.167
n = 25	1.477	1.491	1.458	1.500	1.045	**1.042**	1.055	1.043	1.510	1.492	1.498	1.497	1.050	1.071	1.074	1.063
n = 30	1.311	1.314	1.293	1.306	**0.935**	0.953	0.943	0.953	1.325	1.335	1.320	1.329	0.954	0.968	0.952	0.955
n = 50	0.993	0.972	0.961	0.947	0.814	0.795	0.792	0.781	0.938	0.956	0.947	0.918	0.790	0.771	0.757	**0.751**
n = 100	0.838	0.833	0.761	0.742	0.689	0.710	0.641	0.632	0.793	0.789	0.723	0.734	0.660	0.667	0.603	**0.600**
n = 150	0.719	0.727	0.632	0.639	0.633	0.630	0.533	0.539	0.689	0.684	0.617	0.604	0.591	0.592	0.511	**0.506**
mean	1.374	1.378	1.348	1.353	0.948	0.962	**0.938**	0.940	1.384	1.385	1.368	1.364	0.957	0.964	0.951	0.951
m = 2	1.462	1.428	1.446	1.444	**0.820**	0.857	0.846	0.837	1.489	1.487	1.502	1.448	0.860	0.853	0.866	0.876
m = 3	1.541	1.559	1.529	1.541	1.001	1.010	**0.985**	1.012	1.565	1.565	1.557	1.582	1.033	1.037	1.030	1.035
m = 4	1.593	1.616	1.577	1.568	1.103	1.107	1.102	**1.098**	1.610	1.610	1.604	1.596	1.101	1.127	1.111	1.117
m = 5	1.570	1.572	1.571	1.588	**1.094**	1.121	1.115	1.112	1.592	1.592	1.593	1.593	1.117	1.137	1.143	1.130
m = 6	1.488	1.517	1.465	1.488	1.074	1.087	**1.068**	1.075	1.531	1.532	1.490	1.518	1.088	1.099	1.097	1.099
m = 10	0.898	0.886	0.837	0.826	0.755	0.749	0.696	0.687	0.855	0.859	0.818	0.793	0.715	0.711	0.664	**0.662**
m = 20	0.802	0.802	0.732	0.726	0.670	0.675	0.615	0.615	0.759	0.761	0.707	0.712	0.646	0.642	0.584	**0.576**
mean	1.374	1.378	1.348	1.353	0.948	0.962	**0.938**	0.940	1.384	1.385	1.368	1.364	0.957	0.964	0.951	0.951

Table 6. ARPD for instances with setup times distributed so that the maximum setup is 50% of the maximum processing time. The best results for each line are in bold

	IG 1	IG 2	IG 3	IG 4	IG 5	IG 6	IG 7	IG 8	IG 9	IG 10	IG 11	IG 12	IG 13	IG 14	IG 15	IG 16
f = 2	0.683	0.684	0.656	0.659	0.639	0.627	0.613	**0.593**	0.677	0.676	0.662	0.672	0.642	0.629	0.610	0.628
f = 3	1.029	1.076	1.015	0.985	0.887	0.905	0.889	0.889	1.036	1.059	1.028	1.013	0.905	0.929	0.908	**0.882**
f = 4	1.170	1.213	1.172	1.182	1.082	1.102	1.063	1.072	1.182	1.197	1.150	1.178	1.058	1.078	**1.052**	1.087
f = 5	1.369	1.364	1.320	1.342	1.079	1.107	1.080	1.092	1.391	1.367	1.350	1.366	1.096	1.085	1.135	**1.071**
f = 6	1.562	1.622	1.581	1.553	**0.999**	1.033	1.020	1.009	1.582	1.611	1.553	1.588	1.008	1.045	1.008	1.019
f = 7	1.818	1.770	1.779	1.771	0.945	0.969	0.930	0.916	1.820	1.746	1.821	1.760	0.957	0.966	**0.910**	0.920
mean	1.272	1.288	1.254	1.249	0.939	0.957	0.933	**0.929**	1.281	1.276	1.261	1.263	0.944	0.955	0.937	0.935
n = 15	1.489	1.507	1.487	1.437	**0.906**	0.948	0.937	0.932	1.499	1.506	1.498	1.484	0.925	0.976	0.971	0.949
n = 20	1.502	1.535	1.531	1.501	1.110	1.132	1.096	**1.093**	1.526	1.537	1.522	1.535	1.112	1.115	1.106	1.141
n = 25	1.386	1.417	1.367	1.406	1.077	1.075	1.077	**1.054**	1.429	1.396	1.406	1.394	1.071	1.087	1.088	1.061
n = 30	1.252	1.269	1.228	1.251	**0.937**	0.980	0.962	0.978	1.274	1.268	1.257	1.295	0.981	0.995	0.964	0.972
n = 50	0.923	0.899	0.894	0.914	0.816	0.780	0.787	0.782	0.879	0.875	0.894	0.874	0.782	0.756	0.755	**0.726**
n = 100	0.835	0.818	0.747	0.714	0.672	0.704	0.624	0.618	0.777	0.776	0.689	0.699	0.681	0.651	0.604	**0.601**
n = 150	0.702	0.710	0.628	0.620	0.636	0.619	0.532	0.528	0.678	0.669	0.597	0.573	0.594	0.576	**0.500**	0.512
mean	1.272	1.288	1.254	1.249	0.939	0.957	0.933	**0.929**	1.281	1.276	1.261	1.263	0.944	0.955	0.937	0.935
m = 2	1.427	1.357	1.384	1.366	**0.873**	0.895	0.912	0.877	1.440	1.380	1.431	1.357	0.922	0.902	0.941	0.910
m = 3	1.411	1.482	1.437	1.424	1.005	1.010	**0.960**	1.022	1.442	1.467	1.427	1.486	0.996	1.058	1.006	1.001
m = 4	1.487	1.551	1.506	1.463	1.095	1.123	1.084	1.087	1.519	1.490	1.524	1.481	1.084	1.128	**1.073**	1.109
m = 5	1.391	1.419	1.409	1.420	**1.038**	1.093	1.097	1.050	1.387	1.430	1.407	1.431	1.063	1.076	1.089	1.081
m = 6	1.319	1.351	1.279	1.320	**1.027**	1.048	1.037	1.036	1.372	1.366	1.314	1.379	1.043	1.052	1.052	1.054
m = 10	0.849	0.843	0.821	0.790	0.738	0.731	0.680	0.679	0.825	0.818	0.796	0.768	0.713	0.686	0.656	**0.650**
m = 20	0.790	0.774	0.692	0.708	0.679	0.671	0.616	0.607	0.731	0.729	0.658	0.663	0.658	0.636	0.584	**0.577**
mean	1.272	1.288	1.254	1.249	0.939	0.957	0.933	**0.929**	1.281	1.276	1.261	1.263	0.944	0.955	0.937	0.935

Table 7. ARPD for instances with setup times distributed so that the maximum setup is 100% of the maximum processing time. The best results for each line are in bold

	IG 1	IG 2	IG 3	IG 4	IG 5	IG 6	IG 7	IG 8	IG 9	IG 10	IG 11	IG 12	IG 13	IG 14	IG 15	IG 16
f = 2	0.640	0.655	0.636	0.620	0.616	0.601	0.585	**0.582**	0.676	0.663	0.636	0.635	0.612	0.619	0.610	0.603
f = 3	1.036	1.066	1.037	1.052	**0.871**	0.893	0.874	0.878	1.071	1.094	1.061	1.061	0.885	0.905	0.871	0.874
f = 4	1.211	1.213	1.206	1.209	1.063	1.030	1.037	**1.021**	1.261	1.253	1.222	1.214	1.044	1.059	1.034	1.064
f = 5	1.440	1.455	1.402	1.425	**1.053**	1.096	1.107	1.055	1.442	1.457	1.415	1.418	1.085	1.090	1.096	1.067
f = 6	1.812	1.806	1.775	1.786	1.062	1.071	1.074	1.078	1.806	1.863	1.819	1.826	**1.051**	1.085	1.086	1.082
f = 7	2.139	2.165	2.093	2.141	1.019	1.025	1.013	1.007	2.157	2.157	2.091	2.134	**0.995**	1.039	1.006	1.014
mean	1.380	1.393	1.358	1.372	0.947	0.952	0.948	**0.937**	1.402	1.414	1.374	1.381	0.945	0.966	0.951	0.951
n = 15	1.667	1.670	1.651	1.647	**0.966**	1.007	0.983	0.983	1.708	1.724	1.681	1.684	0.983	0.994	1.022	1.028
n = 20	1.708	1.755	1.696	1.742	1.159	1.161	1.136	**1.118**	1.752	1.782	1.728	1.771	1.185	1.181	1.161	1.172
n = 25	1.488	1.503	1.471	1.513	1.023	**1.015**	1.077	1.051	1.527	1.523	1.498	1.497	1.018	1.059	1.071	1.076
n = 30	1.302	1.311	1.311	1.313	0.932	0.924	0.953	0.943	1.335	1.348	1.335	1.332	**0.924**	0.968	0.947	0.937
n = 50	0.983	0.977	0.969	0.938	0.809	0.805	0.789	0.798	0.951	0.975	0.939	0.930	0.793	0.778	0.767	**0.757**
n = 100	0.819	0.819	0.739	0.724	0.677	0.688	0.643	0.623	0.792	0.780	0.728	0.718	0.636	0.681	0.592	**0.585**
n = 150	0.716	0.719	0.628	0.638	0.627	0.623	0.521	0.521	0.676	0.687	0.591	0.603	0.581	0.596	0.499	**0.483**
mean	1.380	1.393	1.358	1.372	0.947	0.952	0.948	**0.937**	1.402	1.414	1.374	1.381	0.945	0.966	0.951	0.951
m = 2	1.436	1.408	1.437	1.459	**0.754**	0.804	0.792	0.787	1.478	1.505	1.456	1.445	0.792	0.813	0.813	0.819
m = 3	1.602	1.627	1.575	1.613	1.039	1.038	1.037	**1.025**	1.625	1.621	1.623	1.640	1.062	1.053	1.059	1.090
m = 4	1.591	1.629	1.577	1.602	1.098	1.103	1.159	1.133	1.623	1.651	1.585	1.647	**1.093**	1.149	1.129	1.145
m = 5	1.578	1.608	1.567	1.601	1.101	1.115	1.116	1.106	1.620	1.640	1.599	1.611	1.112	1.135	1.140	**1.100**
m = 6	1.501	1.526	1.504	1.493	1.108	1.073	1.083	**1.068**	1.558	1.556	1.539	1.510	1.078	1.103	1.109	1.113
m = 10	0.905	0.878	0.820	0.816	0.759	0.749	0.691	0.679	0.856	0.876	0.808	0.787	0.708	0.727	0.657	**0.654**
m = 20	0.774	0.799	0.737	0.717	0.650	0.662	0.611	0.616	0.757	0.751	0.697	0.713	0.632	0.644	0.581	**0.563**
mean	1.380	1.393	1.358	1.372	0.947	0.952	0.948	**0.937**	1.402	1.414	1.374	1.381	0.945	0.966	0.951	0.951

Table 8. ARPD for instances with setup times distributed so that the maximum setup is 150% of the maximum processing time. The best results for each line are in bold

	IG 1	IG 2	IG 3	IG 4	IG 5	IG 6	IG 7	IG 8	IG 9	IG 10	IG 11	IG 12	IG 13	IG 14	IG 15	IG 16
f = 2	0.641	0.613	0.614	0.607	0.603	0.593	**0.555**	0.568	0.635	0.641	0.618	0.609	0.612	0.589	0.569	0.576
f = 3	1.063	1.054	1.036	1.030	0.859	0.873	**0.834**	0.859	1.051	1.060	1.055	1.055	0.889	0.899	0.869	0.859
f = 4	1.289	1.256	1.277	1.280	1.062	1.093	1.062	1.066	1.273	1.282	1.267	1.287	1.094	1.083	**1.055**	1.102
f = 5	1.520	1.522	1.485	1.527	1.113	1.141	**1.083**	1.119	1.547	1.553	1.537	1.535	1.102	1.129	1.125	1.118
f = 6	1.951	1.978	1.895	1.918	1.075	1.078	**1.055**	1.089	1.948	1.980	1.986	1.930	1.130	1.092	1.104	1.120
f = 7	2.354	2.293	2.293	2.264	1.031	1.071	**1.013**	1.030	2.358	2.269	2.347	2.273	1.060	1.038	1.073	1.040
mean	1.470	1.453	1.433	1.438	0.957	0.975	**0.934**	0.955	1.469	1.464	1.468	1.449	0.981	0.972	0.966	0.969
n = 15	1.819	1.773	1.802	1.766	**0.987**	1.049	0.999	1.014	1.848	1.831	1.869	1.810	1.052	1.059	1.062	1.081
n = 20	1.821	1.806	1.793	1.799	1.150	1.156	**1.132**	1.193	1.847	1.826	1.837	1.811	1.207	1.165	1.193	1.189
n = 25	1.556	1.552	1.536	1.581	1.035	1.037	**1.010**	1.025	1.572	1.556	1.590	1.601	1.062	1.065	1.062	1.053
n = 30	1.377	1.361	1.339	1.354	0.937	0.954	**0.915**	0.940	1.366	1.389	1.369	1.361	0.958	0.943	0.946	0.955
n = 50	1.073	1.041	1.020	0.989	0.817	0.801	0.800	0.764	0.985	1.019	1.009	0.949	0.795	0.780	**0.750**	0.769
n = 100	0.860	0.864	0.796	0.789	0.718	0.738	0.656	0.656	0.811	0.811	0.751	0.786	0.662	0.670	0.613	**0.612**
n = 150	0.738	0.752	0.640	0.660	0.637	0.649	0.545	0.567	0.713	0.697	0.664	0.637	0.599	0.602	0.534	**0.523**
mean	1.470	1.453	1.433	1.438	0.957	0.975	**0.934**	0.955	1.469	1.464	1.468	1.449	0.981	0.972	0.966	0.969
m = 2	1.523	1.517	1.517	1.506	0.833	0.871	**0.832**	0.848	1.549	1.575	1.618	1.543	0.866	0.846	0.843	0.897
m = 3	1.609	1.569	1.574	1.586	**0.958**	0.982	0.958	0.991	1.627	1.606	1.621	1.621	1.039	1.001	1.027	1.015
m = 4	1.700	1.668	1.648	1.640	1.114	1.095	**1.064**	1.075	1.686	1.689	1.703	1.661	1.126	1.104	1.131	1.097
m = 5	1.740	1.688	1.736	1.743	1.142	1.155	**1.132**	1.181	1.768	1.706	1.774	1.738	1.176	1.200	1.199	1.209
m = 6	1.645	1.673	1.613	1.650	1.089	1.140	**1.083**	1.120	1.661	1.675	1.616	1.667	1.143	1.140	1.130	1.129
m = 10	0.939	0.937	0.870	0.871	0.767	0.767	0.717	0.703	0.885	0.883	0.851	0.823	0.724	0.721	**0.679**	0.682
m = 20	0.843	0.834	0.767	0.754	0.681	0.692	0.617	0.622	0.788	0.802	0.766	0.758	0.647	0.648	**0.585**	0.588
mean	1.470	1.453	1.433	1.438	0.957	0.975	**0.934**	0.955	1.469	1.464	1.468	1.449	0.981	0.972	0.966	0.969

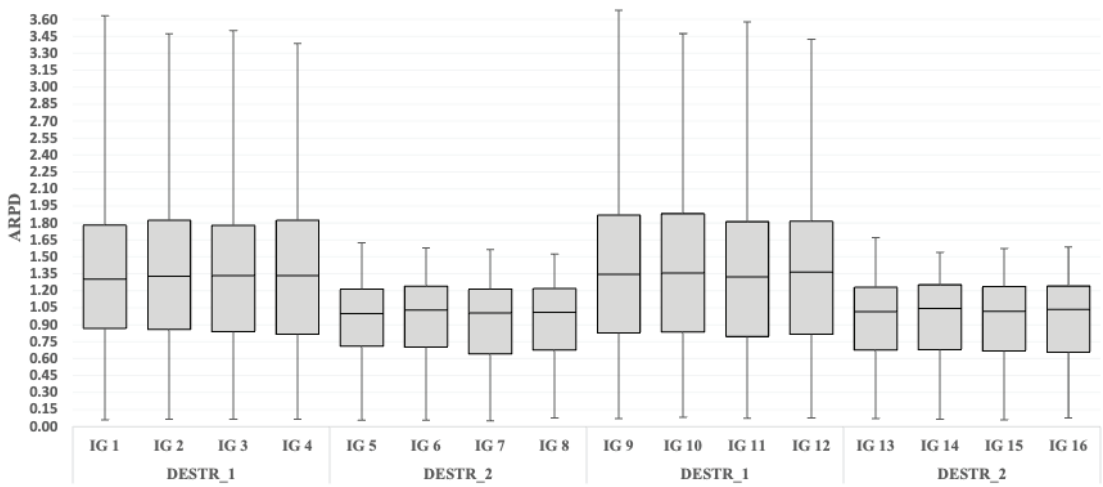

Figure 3. Mean plot for each of the 16 IG methods.

DESTR_2 provides up to 47.6% better average results when compared to DESTR_1. Among the methods with the same destruction procedure, the results variation is low, up to 2,5% on average. This shows that the choice of initialization and reconstruction procedures has little effect on the performance of the IG method, with the NEH_3 initialization procedure and the RECONST_2 reconstruction procedure being slightly superior to the others.

Figure 5 shows the mean plot for all 16 IG methods for different values of parameter d and T. The variation of the parameters d and T has minimum impact on the performance, with the best configuration being on average

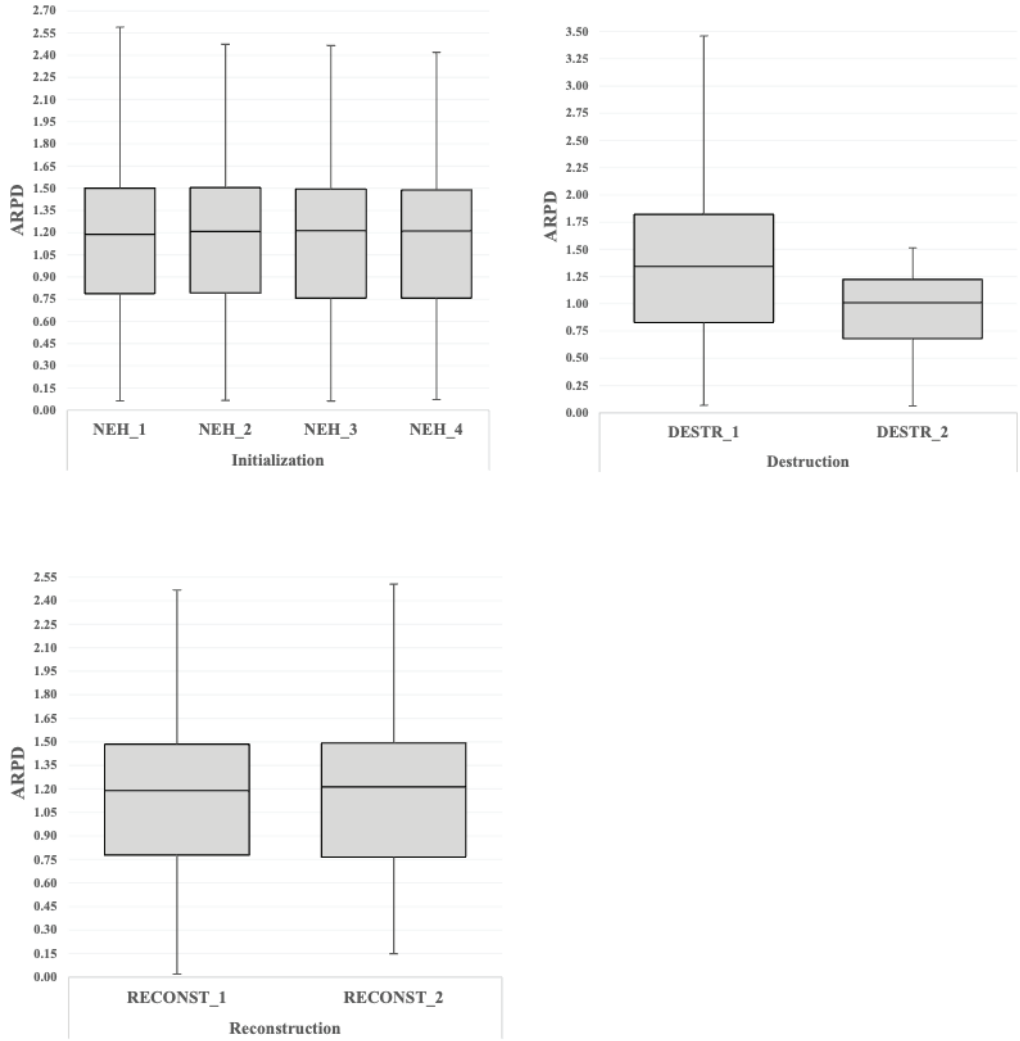

Figure 4. Mean plot for each initialization, destruction and reconstruction procedures.

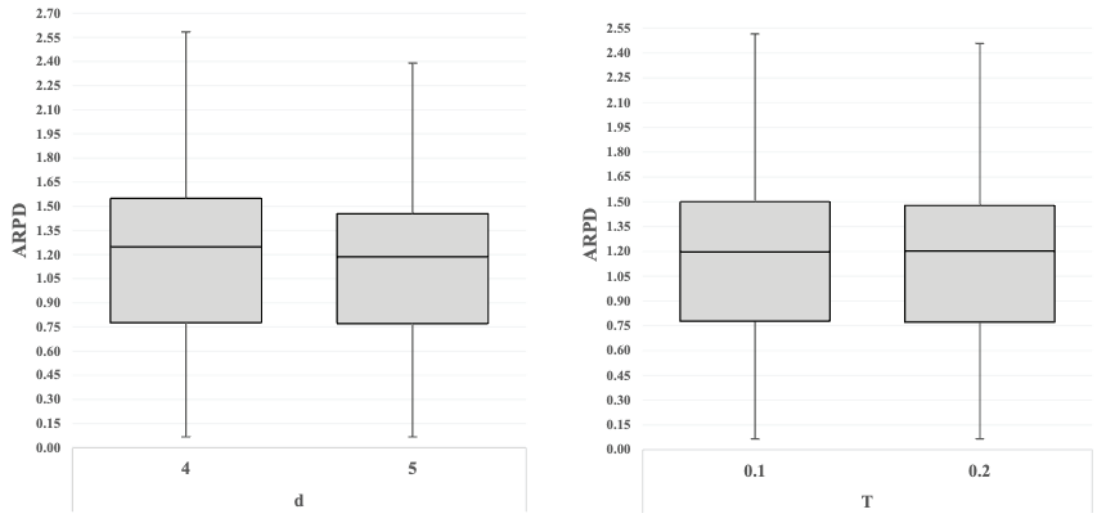

Figure 5. Mean plot for all 16 IG methods for parameter d and T.

d = 5, 1.1345, and T = 0.2, 1.1545. However, unlike the parameter T, parameter d when evaluated in each of the 16 methods shows relevant behavior. On average and for small instances, Figures 6 and 7, respectively, the value of d = 5 obtains better ARPDs for methods with DESTR_1, while for methods with DESTR_2 d = 4 presented better results. On the contrary, for large instances, Figure 8, d = 5 performs better in all but one method, IG15. This may mean that for smaller instances, in addition to the choice of destruction procedure having a great influence on the

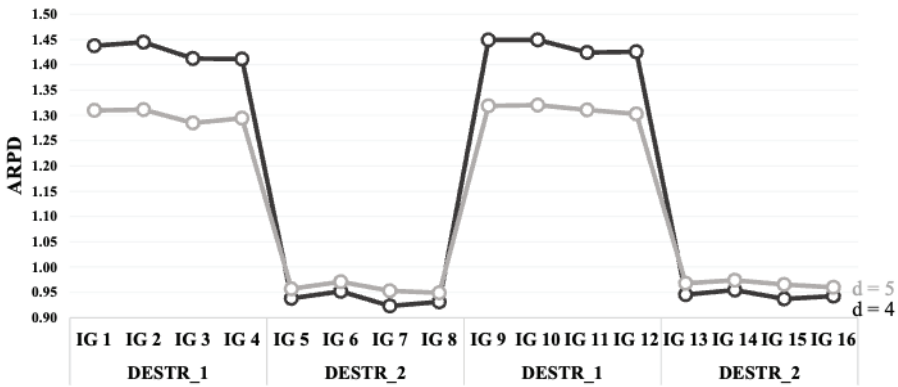

Figure 6. ARPD for the IG methods and different d parameters for all instances.

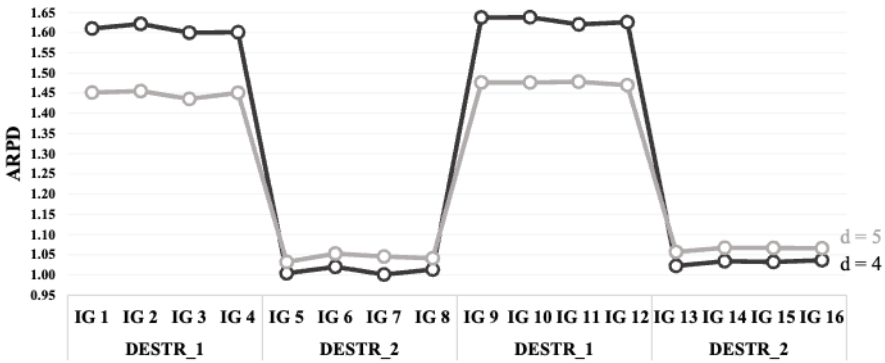

Figure 7. ARPD for the IG methods and different d parameters in small instances.

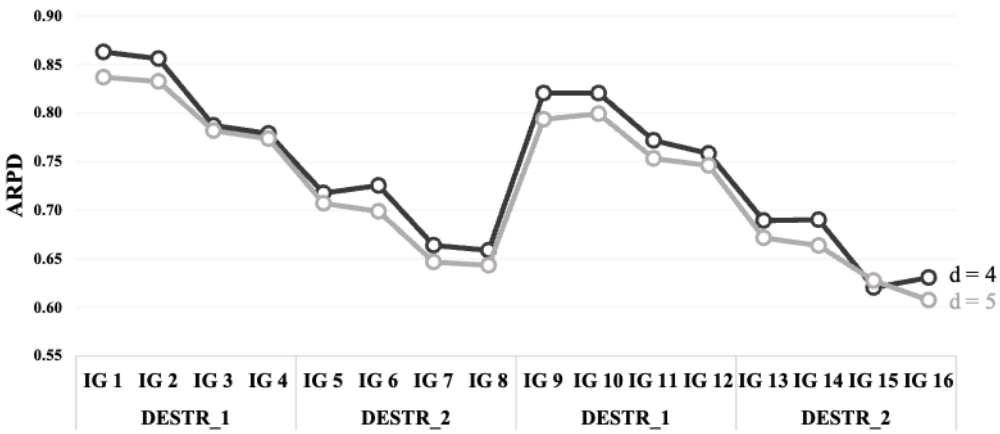

Figure 8. ARPD for the IG methods and different d parameters in large instances.

Table 9. ARPD of IG methods with DESTR_2 for each parameter configuration and instance type. The best results are in green, the worst in red, with the optimal outcome and the least favorable in bold

		ARPD							
d	**T**	**IG 5**	**IG 6**	**IG 7**	**IG 8**	**IG 13**	**IG 14**	**IG 15**	**IG 16**
5	0.1	0.968	0.972	0.982	**0.989**	0.957	0.959	0.969	0.968
5	0.2	0.946	0.970	0.954	0.959	0.949	0.939	0.962	0.953
4	0.1	0.938	0.971	0.953	0.951	**0.923**	0.932	0.931	0.934
4	0.2	0.939	0.933	0.938	0.958	**0.923**	0.931	0.944	0.952
		ARPD for small instances							
d	**T**	**IG 5**	**IG 6**	**IG 7**	**IG 8**	**IG 13**	**IG 14**	**IG 15**	**IG 16**
5	0.1	1.042	1.054	1.074	**1.086**	1.050	1.052	1.072	1.072
5	0.2	1.022	1.052	1.040	1.048	1.041	1.029	1.061	1.060
4	0.1	1.005	1.047	1.034	1.030	1.002	1.014	1.023	1.025
4	0.2	1.004	**0.993**	1.011	1.038	1.000	1.013	1.041	1.047
		ARPD for large instances							
d	**T**	**IG 5**	**IG 6**	**IG 7**	**IG 8**	**IG 13**	**IG 14**	**IG 15**	**IG 16**
5	0.1	0.720	0.699	0.674	0.664	0.649	0.649	0.624	0.618
5	0.2	0.693	0.698	0.669	0.663	0.644	0.637	0.630	**0.596**
4	0.1	0.715	0.718	0.683	0.691	0.658	0.660	0.623	0.628
4	0.2	0.720	**0.733**	0.696	0.690	0.669	0.657	0.618	0.633

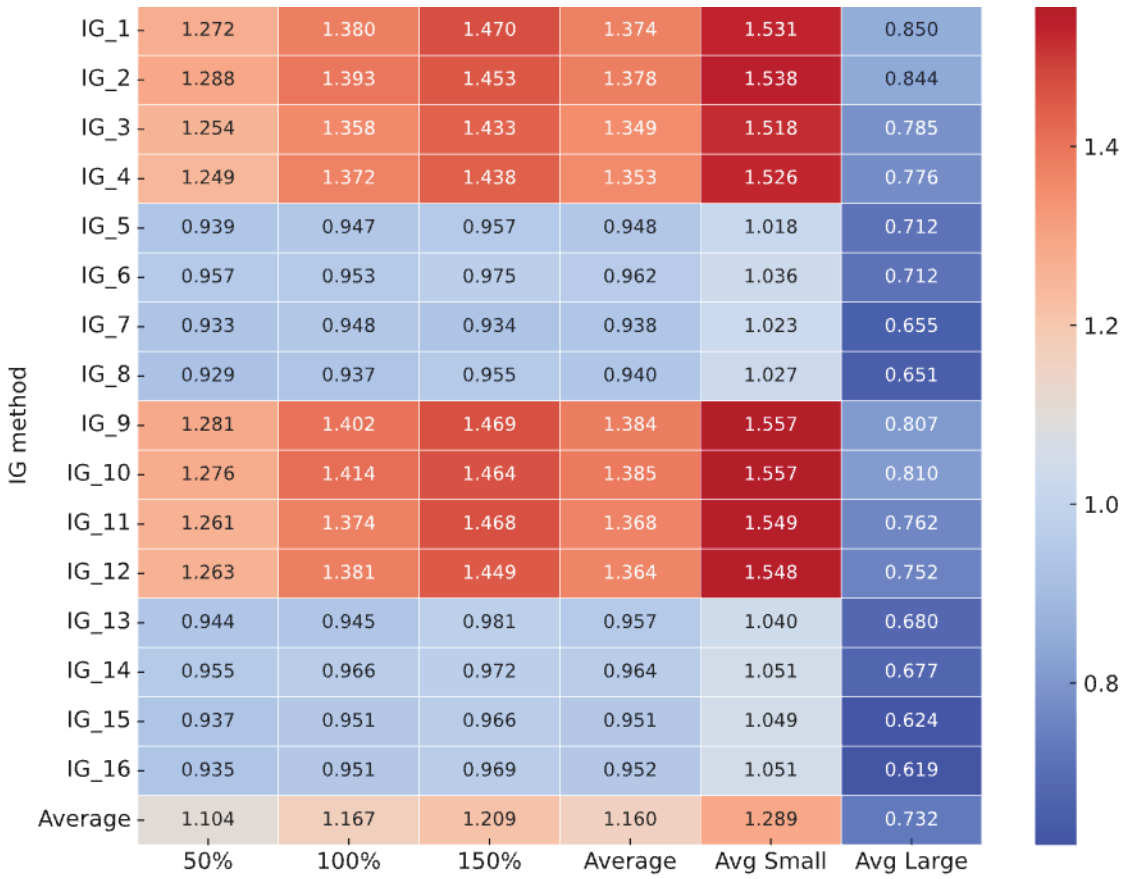

Figure 9. Heatmap illustrating the values of different IG methods across various conditions (50%, 100%, 150%, Average, Avg Small, Avg Large). The color gradient represents the magnitude of the values.

performance of the algorithm, the number of jobs to be removed must be defined according to the type of destruction procedure. For DESTR_1 the more jobs removed the better the performance and for DESTR_2, the fewer jobs removed, the better the performance. As most of the tests were conducted on smaller instances, the average results exhibit a similar trend. Worth noting, when dealing with larger instances requiring the sequencing of numerous jobs, it is advantageous to remove more jobs, regardless of the type of destruction. A possible explanation for the overall superior performance of DESTR_2 lies in its partial random removal of jobs, which reduces the method's greediness. This approach increases the likelihood of escaping local optima and exploring new search spaces, potentially leading to better solutions and further improving overall performance of the method.

The T factor directly impacts the acceptance criteria. The higher its value, the greater the chance of accepting a worse solution than the current one, allowing the search to shift away from the local optima already discovered and toward other potential optima that may yield better outcomes. From the results of the best methods, which utilize the DESTR_2 destruction procedure, shown in Table 9, for both small and large instances, T of 0.2 performed better than 0.1 for almost all algorithms. For small instances, the configuration with d = 4 and T = 0.2 yields the most favorable outcomes, with IG6 exhibiting the best ARPD of 0.993. For large instances, the configuration with d = 5 and T = 0.2 yields the best results, with IG16 achieving the best ARPD of 0.596. However, on average the best method was IG13, ARPD of 0.923, with the configuration d = 4 and T = 0.1 or T = 0.2. Therefore, once again it is important to analyze the production scenario before defining the best configuration and consequently the best method to be used. Figure 9 presents a performance heatmap comparing all IG methods across small and large instances, highlighting the performance of IG7 in comparison to other methods.

CONCLUSION

This research focused on solving the Distributed Permutation Flowshop Scheduling Problem (DPFSP) under the constraint of sequence-independent setup times (SIST)—a factor with significant practical relevance but limited attention in the current literature. A total of 16 Iterated Greedy (IG) variants were developed by systematically combining four initialization procedures, two destruction strategies, and two reconstruction methods. Additionally, two key parameters were evaluated: the number of jobs removed during the destruction phase (d), and the temperature parameter (T) influencing the acceptance criterion.

The results clearly show that the destruction procedure plays a central role in the overall performance of IG methods. While the choice of initialization and reconstruction methods had minimal impact, the DESTR_2 strategy consistently yielded better outcomes across all tested scenarios. For smaller instances, the best results were achieved with d = 4 and T = 0.2, while in larger instances, increasing the number of removed jobs to d = 5 maintained superior performance. Among all configurations, the combination of NEH_3 (initialization), DESTR_2 (destruction), and RECONST_1 (reconstruction)—identified as IG7—produced the best average results, confirming the effectiveness of this specific configuration. These findings align with previous research by [30], reinforcing the idea that job removal and rescheduling strategies critically influence distributed scheduling performance.

Another important conclusion is the necessity of aligning the IG configuration with the characteristics of the production environment. The IG methods demonstrated greater effectiveness in large-scale instances, as reflected in lower ARPD values, suggesting strong scalability. Moreover, the relevance of setup time within the scheduling environment should guide the choice of method. IG7 was more effective in scenarios where setup time had a substantial impact, while IG8 performed better when processing time was equally or more critical.

Overall, the IG methodologies proposed in this study offer practical strategies for improving scheduling performance in distributed manufacturing systems. These approaches are particularly relevant for industries aiming to minimize delays, optimize machine utilization, and streamline production operations. The benchmarking results provide valuable guidance for both practitioners and researchers by identifying the most effective combinations of algorithmic components. Key conclusions include:

1. Optimal Initialization and Reconstruction: NEH_3 and RECONST_1 were the most effective in generating strong initial and rebuilt solutions, improving makespan outcomes.
2. Destruction Procedure Effectiveness: DESTR_2 consistently outperformed DESTR_1, emphasizing the importance of thoughtful disruption in iterative search strategies.
3. Parameter Sensitivity: Performance varied significantly with changes to d and T, with d = 5 and T = 0.2 emerging as the most robust configuration for large instances.
4. IG7 as the Best Variant: Among all tested methods, IG7 achieved the lowest ARPD values, demonstrating its general effectiveness across different scenarios.
5. Scalability: The IG framework showed strong scalability, delivering better relative performance in larger problem instances.

Future work could extend these findings by exploring new configurations within the IG framework, such as alternative values for d, T, or cycle limits (C), or by applying the method to heterogeneous factory environments to more closely mirror real-world systems. Additionally, the IG methods developed here could be applied to integrated production-distribution systems, incorporating logistics and delivery scheduling. As this study focused solely on

makespan minimization, another important avenue for research is to incorporate multi-objective criteria, including total tardiness, energy consumption, or cost-related factors, to better reflect the complex trade-offs encountered in actual manufacturing environments.

ACKNOWLEDGEMENT

This research was partially supported by grants 308847/2026-1 and 404819/2023-0 from the National Council for Scientific and Technological Development (CNPq), Brazil.

AUTHORSHIP CONTRIBUTIONS

Authors equally contributed to this work.

DATA AVAILABILITY STATEMENT

The authors confirm that the data that supports the findings of this study are available within the article. Raw data that support the finding of this study are available from the corresponding author, upon reasonable request.

CONFLICT OF INTEREST

The authors declared no conflict of interest.

ETHICS

There are no ethical issues with the publication of this manuscript.

STATEMENT ON THE USE OF ARTIFICIAL INTELLIGENCE

Artificial intelligence was not used in the preparation of the article.

REFERENCES

[1] Martinez S, Hall NG, Sriskandarajah C. Complexity of flowshop scheduling problems with a new blocking constraint. Eur J Oper Res 2006;169:855–864. [CrossRef]

[2] Trabelsi W, Sauvey C, Sauer N. Heuristics and metaheuristics for mixed blocking constraints flowshop scheduling problems. Comput Oper Res 2012;39:2520–2527. [CrossRef]

[3] Gong H, Tang L, Duin CW. A two-stage flowshop scheduling problem on a batching machine and a discrete machine with blocking and shared setup times. Comput Oper Res 2010;37:960–969. [CrossRef]

[4] Grabowski J, Pempera J. Sequencing of jobs in some production system. Eur J Oper Res 2000;125:535–550. [CrossRef]

[5] Yu AJ, Seif J. Minimizing tardiness and maintenance costs in flowshop scheduling by a lower-bound-based GA. Comput Ind Eng 2016;97:26–40. [CrossRef]

[6] Hall NG, Sriskandarajah C. A survey of machine scheduling problems with blocking and no-wait in process. Oper Res 1996;44:510–525. [CrossRef]

[7] Ercan T, Kayakutlu G. Job scheduling with the help of dominance properties and genetic algorithm on hybrid flow shop problem. Sigma J Eng Nat Sci 2015;33:345–355.

[8] Sarıçiçek İ. Multi-objective scheduling by maximizing machine preferences for unrelated parallel machines. Sigma J Eng Nat Sci 2020;38:405–420.

[9] Alcan G, Gündüz M, Akdoğan M. Scheduling in energy systems. Sigma J Eng Nat Sci 2015;6:67–76.

[10] Yang W, Liao CJ. Survey of scheduling research involving setup times. Int J Syst Sci 1999;30:143–155. [CrossRef]

[11] Allahverdi A, Ng CT, Cheng TCE, Kovalyov MY. A survey of scheduling problems with setup times or costs. Eur J Oper Res 2008;187:985–1032. [CrossRef]

[12] Allahverdi A. A survey of scheduling problems with no-wait in process. Eur J Oper Res 2016;255:665–686. [CrossRef]

[13] Belabid J, Aqil S, Karam A. Solving permutation flowshop scheduling problem with sequence-independent setup time. J Appl Math 2020;2020. [CrossRef]

[14] Naderi B, Ruiz R. The distributed permutation flowshop scheduling problem. Comput Oper Res 2010;37:754–768. [CrossRef]

[15] Avci M, Avci MG, Hamzadayi A. A branch-and-cut approach for the distributed no-wait flowshop scheduling problem. Comput Oper Res 2022;148. [CrossRef]

[16] Deng J, Wang L. A competitive memetic algorithm for multi-objective distributed permutation flowshop scheduling problem. Swarm Evol Comput 2017;32:121–131. [CrossRef]

[17] Zhang G, Xing K. Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion. Comput Oper Res 2019;108:33–43. [CrossRef]

[18] Huang J, Pan QK, Gao L, Zhang C. Effective constructive heuristics and discrete bee colony optimization for distributed flowshop with setup times. Eng Appl Artif Intell 2021;97. [CrossRef]

[19] Yu Y, Guo H, Pan QK. A discrete artificial bee colony method based on variable neighborhood structures for the distributed permutation flowshop problem with sequence-dependent setup times. Swarm Evol Comput 2022;75. [CrossRef]

[20] Karabulut K, Pan QK, Guo H, Zhang L. An evolution strategy approach for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. Comput Oper Res 2022;142. [CrossRef]

[21] Huang J, Pan QK, Gao L. An effective iterated greedy method for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. Swarm Evol Comput 2020;59. [CrossRef]
[22] Guo H, Pan QK, Zhang L. An effective metaheuristic with a differential flight strategy for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. Knowl Based Syst 2022;242. [CrossRef]
[23] Han B, Pan QK, Gao L. A cooperative iterated greedy algorithm for the serial distributed permutation flowshop scheduling problem. Int J Prod Res 2024. [CrossRef]
[24] Meng Q, Qiu D, Liu Y. Two-stage assembly flow shop scheduling problem with sequence-independent setup times. IEEE 19th International Conference 2023. [CrossRef]
[25] Qiao Y, Wu NQ, He YF. Adaptive genetic algorithm for two-stage hybrid flow-shop scheduling with sequence-independent setup time and no-interruption requirement. Expert Syst Appl 2022. [CrossRef]
[26] Elissaouy O, Allali K. Minimizing the maximum tardiness for a permutation flow shop problem under the constraint of sequence-independent setup time. RAIRO Oper Res 2024. [CrossRef]
[27] Ying KC, Lin SW, Chen KF. Minimising makespan in distributed assembly hybrid flowshop scheduling problems. Int J Prod Res 2024. [CrossRef]
[28] İnce N, Deliktaş D, Selvi İH. A comprehensive literature review of the flowshop group scheduling problems: Systematic and bibliometric reviews. Int J Prod Res 2024. [CrossRef]
[29] Perez-Gonzalez P, Framinan JM. A review and classification on distributed permutation flowshop scheduling problems. Eur J Oper Res 2023;312:1–21. [CrossRef]
[30] Ruiz R, Pan QK, Naderi B. Iterated greedy methods for the distributed permutation flowshop scheduling problem. Omega 2019;83:213–222. [CrossRef]
[31] Fernandez-Viagas V, Framinan JM. A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem. Int J Prod Res 2015;53:1111–1123. [CrossRef]
[32] Lin SW, Ying KC, Huang CY. Minimising makespan in distributed permutation flowshops using a modified iterated greedy algorithm. Int J Prod Res 2013;51:5029–5038. [CrossRef]
[33] Nawaz M, Ensore E Jr, Ham I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 1982;11:91–95. [CrossRef]
[34] Laurentys CT, Nagano MS. Evaluating procedures in the NEH heuristic for the PFSP - SIST. J Proj Manag 2024;9:17–26. [CrossRef]
[35] Fernandez-Viagas V, Ruiz R, Framinan JM. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. Eur J Oper Res 2017;257:707–721. [CrossRef]
[36] Liu W, Jin Y, Price M. A new improved NEH heuristic for permutation flowshop scheduling problems. Int J Prod Econ 2017;193:21–30. [CrossRef]
[37] Ruiz R, Stützle T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 2007;177:2033–2049. [CrossRef]
[38] Juan AA, Faulin J, Ruiz R, Barrios BB, Caballe S. Using iterated local search for solving the flow-shop problem: Parallelization, parametrization, and randomization issues. Int Trans Oper Res 2013;21:103–126. [CrossRef]
[39] Katragjini K, Vallada E, Ruiz R. Flowshop rescheduling under different types of disruption. Int J Prod Res 2013;51:780–797. [CrossRef]
[40] Ruiz R, Allahverdi A. Some effective heuristics for no-wait flowshops with setup times to minimize total completion time. Ann Oper Res 2007;156:143–171. [CrossRef]
[41] Taillard E. Benchmarks for basic scheduling problems. Eur J Oper Res 1993;64:278–285. [CrossRef]