Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD27278

A Literature Review of the Iterated Greedy Algorithm in Permutation Flow-shop Scheduling Problems

Zihao Chen

Shanghai University, Shanghai, China
17717346288@163.com

Abstract. This article is a literature review of iterated greedy (IG) applied to solving permutation flow-shop scheduling problem (PFSP) and distributed permutation flow-shop scheduling problem (DPFSP), which has profound and practical significance in job arrangement. Although various reviews of scheduling problems have been made, there are few review articles that focus on a certain algorithm. We first characterize the basic mathematical model of PFSP and DPFSP, as well as the Grantt chart to visualize a given solution. The article then provides a comprehensive review of the research development of IG applied in PFSP and DPFSP. In this progress, we focus on two major perspectives: (1) Classification and interpretation for the literature under a unified framework. (2) Unresolved issues and potential development based on a brief introduction of the algorithm. At the end of the review we also propose a brief introduction regarding the basic structure of iterated greedy algorithm. In the last part, the article leads a conclusion of the review and remark on the future development.

Keywords: Iterated greedy algorithm, permutation flow-shop scheduling, distributed flow-shop scheduling

1. Introduction

As two major variants of flow-shop scheduling problem, the permutation flow-shop scheduling problems (PFSPs) and the distributed permutation flow-shop scheduling problems (DPFSPs) have extensive applications in different kinds of industries including chemical, steel, food and electronic industry that need to arrange job sequences on the available machines. Shop scheduling problems assume that a sequence of independent jobs is going to be processed on a series of machines and a matrix $\{P_{ij}\}$ is set to denote the processing time of job i on machine j. The flow-shop variant assume that all the jobs are processed by the same machine order. The permutation constraints determines that the job sequence remains the same on all of the machines. As an extension of PFSP, the distributed environment allows multi-factory cooperation, where jobs are divided into a set of several independent subsequences $\{\pi^1, \cdots, \pi^n\}$ to be processed in independent factories $\{F_n\}$ $respectively$. An additional assumption is that the processing time of each job on each machine remains uniform in all factories.

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.LD27278

As a meta-heuristic method, the iterated greedy (IG) algorithm is developed to obtain optimal solution to NP-hard problems which cannot be solved in polynomial time and turns out to be effective in other classical problems including the travelling salesman problem. It was initially proposed in scheduling problems by Ruiz [1] in solving permutation flow shop scheduling (PFSP) problems. An IG algorithm consists of four basic steps: initialization, destruction, construction and local search.

This article aims at providing a literature review on the application of the IG algorithm to PFSP and DPFSP and contains articles published between 2007 and 2024 that either propose, improve or apply an IG-related method.

In section2, this article will introduce the problem description of PFSP and DPFSP using both mathematical model and Gantt chart. In section3, a detailed introduction of the IG algorithm, the review methodology of this article as well as the literature review on IG in PFSP and DPFSP based on the publication date and problem-wise framework will be given. Section4 will show the conclusions and remarks of the article.

2. Problem description

Due to its simplicity and effectiveness for a majority of PFSPs, the IG algorithm has been increasingly used in solving various kinds of PFSPs and DPFSPs as a generalization especially since 2019, when Ruiz [2] introduced IG to solve DPFSPs. According to the statistics based on Web of Science database, figure1 shows the number of articles using IG-related heuristics to solve PFSPs and DPFSPs on a yearly basis and the proportion of the two kinds of article is vividly depicted by the coloured bars. More interestingly, regarding the type of scheduling problems solved by IG, before 2019, In all articles that use IG-related heuristics to solve PFSPs and DPSFPs, the former outweighs the latter. While in most of the years after 2019, the latter dwarfs the former. From 2023 to the present, the proportion of articles using IG-related heuristics to solve DPFSPs take up more than 70%.

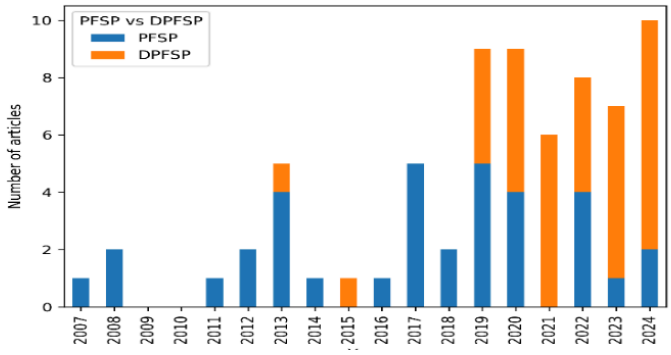

Figure 1. The Number of articles using IG to solve PFSPs and DPFSPs

2.1. Notations and solution representation

Generally speaking, the main characters to describe a PFSP or DPFSP minimizing make-span includes the processing time of each job on each machine and the total completion time until a job being processed on a certain machine. The related notations involved in this section are given in table1.

DOI: 10.54254/2755-2721/2025.LD27278

Table 1. Notations

Notations	Descriptions
π	the job factories
π^i	the job sequence in i-th factory
π^i_j	the j-th job in sequence π^i
$P_{\pi^i_j,k}$	the processing time of j-th job on k-th machine in i-th factory
$C_{\pi^i_j,k}$	the total completion time until j-th job being processed on k-th machine in i-th factory

Additionally, Gantt Chart is developed to visualize the solution of a given PFSP or DPFSP. in terms of a classic PFSP of 4-job, 3-machine as an example, it can be seen in Figure2 that the order of the job sequence does not change in all machines, reflecting the permutation nature. Due to the different processing time, waiting time exists between 1-st job and 3-rd job on 1-st machine, for example, which is called idle time. The make-span is labeled by the red line, meaning the completion time of the last job on the last machine.

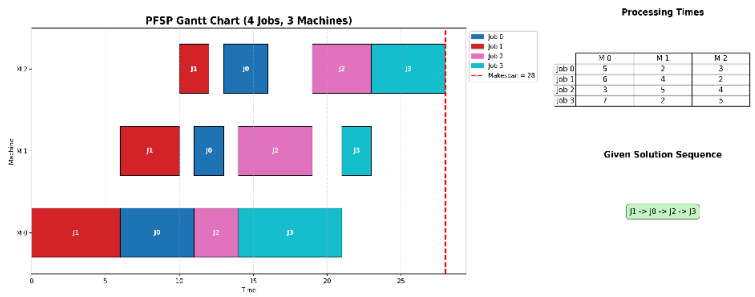

Figure 2. The Gantt chart of the solution of a given PFSP

For DPFSP, the situation is a bit more complicated since there are more than 1 factory. Similarly, the Gantt chart of a certain DPFSP is shown in Figure3, where F1-M1 means 1-st machine in 1-st factory. Clearly, in each factory, DPFSP can be degenerated into PFSP, and the final make-span of a DPFSP is the maximum of the make-span in all factories. It can be observed that with the same processing time, the make-span can be largely reduced by introducing more factories, reflecting the advantage of distribution characteristic.

DOI: 10.54254/2755-2721/2025.LD27278

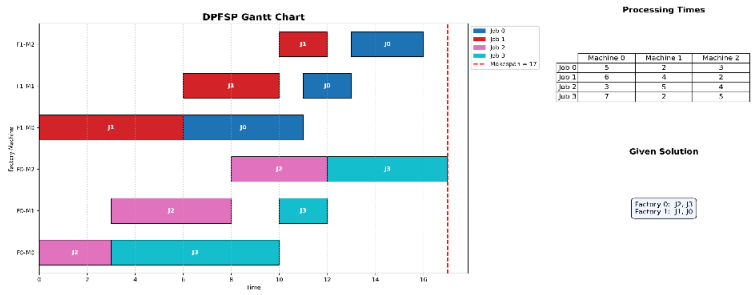

Figure 3. The Gantt chart of the solution of a given DPFSP

2.2. Mathematical model for PFSP and DPFSP

Both PFSPs and DPFSPs are NP-hard MILP problem, which means it is impossible to find an optimal solution in a polynomial time span. Since the DPFSP can be decomposed as a collection of PFSPs, the basic mathematical model of a n-job, m-machine and t-factory distributed permutation flow-shop scheduling problem is described as follows

$$\pi^i = \pi^i_1, ..., \pi^i_{n_i} \tag{1}$$

$$MinC_{max}(\pi) \tag{2}$$

$$s.t. C_{\pi^i_1,1} = P_{\pi_1,1} \tag{3}$$

$$C_{\pi^i_j,1} = C_{\pi^i_{j-1},1} + P_{\pi_j,1} \tag{4}$$

$$C_{\pi^i_1,k} = C_{\pi^i_1,k-1} + P_{\pi_1,k} \tag{5}$$

$$C_{\pi^i_j,k} = max(C_{\pi^i_{j-1},k}, C_{\pi^i_j,k-1}) + P_{\pi^i_j,k} \tag{6}$$

$$C_{max}(\pi) = max(C_{\pi^i_{n_i},m}), fori = 1, ..., t \tag{7}$$

In this model, equation (1) means job sequence in i-th factory contains jobs from 1 to n_i . formula 2 means that the objective function is make-span, where π refers to job sequences in all factories $\pi^1, \cdots, \pi^t$. Constraint 3 means the total completion time of 1-st job on 1-st machine is just the processing time of 1-st job on 1-st machine. Constraint 4 means on the first machine, the completion time of j-th job equals to the completion time of (j-1)-th job plus the processing time of it. Constraint 5 means on k-th machine, the completion time of 1-st job equals to the completion time of 1-st job on (k-1)-th machine plus the processing time of it on k-th machine. Constraint 6 means that the completion time of j-th job on k-th machine equals to the maximum of $C_{\pi_{j-1},k}$ and $C_{\pi_j,k-1}$ plus the processing time of j-th job on k-th machine. To understand this intuitively, it can be known in figure2 that J0 on M1 can be processed as long as J1 has been processed on M1 and J0 has been processed on M0. Equation (7) means that the total make-span of job sequence π is just the maximum of the total completion time in all factories.

For a PFSP, since there is only one factory, the index i can be uniformly set to 1 and $n_i = n$. The equation (7) can be degenerated to $C_{max}(\pi) = C_{\pi^1_n,m}$.

DOI: 10.54254/2755-2721/2025.LD27278

3. Literature review on IG in PFSPs and DPFSPs

3.1. Notations

According to Graham et al [3], the characteristics of a scheduling problem can be described by three parameters denoted as $\alpha|\beta|\gamma$, which describe the machine environment, constraints and performance criteria respectively. For example, $\mathrm{DF}|\,\mathrm{prmu},\,\mathrm{no\text{-}idle},\mathrm{DW}|\mathrm{TWET}$ refers to a distributed permutation flow-shop scheduling problem with no-idle and due-window constraints and its objective function is to minimize total weighted earliness and tardiness. For a PFSP, it can be described as $F\,|\,prmu$, and a DPFSP can be described as $DF\,|\,prmu$. Some of the commonly used values of β and γ involved in this review will be showed in table2 and table3.

Table 2. Notations ofβ

The value of β	Descriptions
prmu	Permutation
nwt	no-wait
no-idle	no-idle
DW	due window
SDST/SIST	sequence-(in)dependent setup times

Table 3. Notations ofγ

The value of γ	Descriptions
$\mathrm{C}_{\max}$	Makespan
$\sum \mathrm{T}$	Total Tardiness
TWT	Total weighted tardiness
TEC	Total energy consumption
TFT	Total flowtime
TWEC	Total weighted tardiness Time
TWET	Total weighted earliness and tardiness

3.2. Review methodology

The following criteria are set to limit the range of this review and clarify the standard of which articles will be selected and analyzed. First, the working environment of the problem mentioned in this article is limited to PFS and DPFS. Other variants of scheduling problems including flexible flow-shop and cell scheduling problem are not included in this review.

In terms of the application of IG, an article is selected in this review if it proposes a novel IG-related algorithm to solve a PFS or DPFS, improves an existing IG algorithm to obtain a better solution, or integrates the IG algorithm with other algorithms to foster its efficiency.

Regarding the searching standard, all the articles involved in this review were searched by the keywords permutation flow-shop, the iterated greedy algorithm or the distributed permutation algorithm within the database Web of Science and the time span is between 2007 and 2024.

Table4 respectively summarize the PFSPs- and DPFSPs-related articles in chronological order. The following review of the literature given in Table4 will be divided into two parts as PFSPs and

DOI: 10.54254/2755-2721/2025.LD27278

DPFSPs.

DOI: 10.54254/2755-2721/2025.LD27278

3.3. IG in PFSP

Table 4. Application of IG in PFSPs and DPFSPs('-' means this content is not mentioned in the article or there is no specific name for the content mentioned in the article)

Problems	Criteria	Constraints	Reference	Initialization	LocalSearch
PFSP	C_{max}	-	R.Ruiz et al[1]	NEH	ILS
PFSP	$C_{max} - TFT$	-	J.M.Framinan et al [4]	NEH	PEH
PFSP	$C_{max} - TWT$, $C_{max} - TFT$	SDST	M.Ciavotta et al [5]	NEH/Rajendran's heuristic	-
PFSP	TEC-TFT	L	H.Oztop et al [6]	PFH_NEH(x)	FI-LS
PFSP	∑T-TEC	no-wait	D.Yuksel et al [7]	FRB5	ILS
PFSP	TWT	d_j	Li et al [8]	TSNEH	LS_1-RLSHybrid_LS
PFSP	$C_{max} - TFT$	nwt	D.Yuksel et al [9]	-	ILS
DPFSP	C_{max}	-	Lin et al [10]	NEH2	-
DPFSP	C_{max}	-	V.FernandezViagas et al [11]	NEH2	LS1,RLS1,RLS2
DPFSP	C_{max}	-	R.Ruiz et al [2]	NEH2_en	LS_3
DPFSP	TWET	DW	Jing et al [12]	ANEH	ITI
DPFSP	C_{max}	SDST	Huang et al [13]	NEH_F	LSB_1
DPFSP	C_{max}	PM	Mao et al [14]	INEH_dp	ISLS,ESLS,EILS
DPFSP	C_{max}	no-idle	F.L.Rossi et al [15]	NEH-RN	LS_3
DPFSP	TFT	mixed no-idle	Li et al [16]	$DFRB4_1$	RLS,SLS
DPFSP	TFT	no-idle	Li et al [17]	H_{11}	InsertProductLS, InsertJobLS, InsertProductRfLS, InsertJobRfLS, SwapProductRfLS, SwapJobRfLS
DPFSP	TFT	-	Zhao et al[18]	CNEH-VND	LocalSearch_Group and LocalSearch_Job

DOI: 10.54254/2755-2721/2025.LD27278

DPFSP	TWFT	no-idle and DW	K.Mousighichi et al [19]	NEH-EDDWET	IS

Ruben Ruiz [1] first introduced the solution representation of IG to solve PFSP with make-span criterion by iterating two major phase, construction and destruction. In the destruction phase, a given number n of d jobs was removed randomly from the permutation π of n jobs to create two sub sequences: π_D of $\pi - d$ jobs and π_R of d jobs. Then the jobs in π_R were reinserted into π_D by the NEH heuristic to yield a improved solution. It is worth mentioning that the NEH heuristic and its variants are widely applied in initialization and destruction-construction phase in most IG algorithms. The ILS algorithm was then applied to update the candidate solution by evaluating the representative neighborhood of the permutation π. Additionally, Taillard acceleration (which is also applied in Michele Ciavotta et al [5], Ruben Ruiz et al [2] and many other papers) and an iterative improvement procedure were used to foster the efficiency. The aforementioned iteration continued until a specific acceptance criterion T based on simulated annealing was satisfied. In conclusion, the IG algorithm only needs two parameter T and d, thus had better conceptual simplicity. It also performed well in the experimental design analysis.

Jose M. Framinan and Rainer Leisten [4] introduced a multi-criteria IG to solve $F|prmu|C_{max}, \sum TFT$. To extend the IG algorithm to multi-objective PFSPs, a set of non-dominated solutions, denoted as BS, is first generated by NEH and FL heuristic. In each iteration, the solutions in BS underwent the destruction and construction phases and resulted in a new set CS, then BS was updated by CS based on the dominance of the solutions. This IG algorithms outperformed MOSA-II, the best algorithm for the problem at that time, in the experimental analysis.

Michele Ciavotta et al [5] introduced Restarted Iterated Pareto Greedy (RIPG) as an extension of original IG to solve $F|SIST, prmu|C_{max}, TWT$, $F|SDST, prmu|C_{max}, TFT$. Similar to Jose and Rainer, a set of non-dominated solutions to respectively optimize the two objectives were generated by NEH heuristic and Rajendran's heuristic. The innovative part of RIPG is that it applied a restart phase where a restart was executed when the working set remained unchanged during a large number of iterations.

With the idea to take energy consumption into consideration, M.Fatih Taşgetiren et al [6] in two of his papers used energy-efficient IG to solve $F_m|prmu, L|TEC, TFT$ (Here L means processing speed levels). To measure the energy consumption of each job processed on each machine, a job-based speed-scaling structure was introduced. Additionally, a novel initialization algorithm, denoted as PFH_NEH(x) was introduced by combing the NEH and PF heuristic. In Damla Yuksel et al[7], the IG algorithm was integrated with MO-DABC, which in computational results outperformed other algorithms.

Qiu-Ying Li [10] introduced an effective Two-Stage Iterated Greedy (ETSIG) algorithm to solve $F|prmu, d_j|TWT$ (Here d_j means soft due windows). Due to the nature of the problem, the original NEH2 method was extended to two-stage NEH(TSNEH) In TSNEH, the jobs were categorized into α and α' by Earliest Due Date (EDD) and Earliest Weighted Due Date(EWDD), representing the seed sequences of hard-due-date jobs and soft-due-date jobs respectively. Jobs in the two seed sequences were then inserted into the solution Π_0 stage by stage. In line with the two-stage concept of the method, an extra parameter ρ is introduced to foster computational efficiency in the second stage.

In recent years, several articles have started to integrate reinforcement learning methods with classical IG to further enhance the effectiveness.

DOI: 10.54254/2755-2721/2025.LD27278

Yüksel et al [9] introduced a $\mathrm{BC\text{-}IG_{QL}}$ to solve $\mathrm{F|prmu, nwt|C_{max}, TFT}$ by integrating block insertion heuristic algorithm and Q-learning guided algorithm with IG. The idea of block insertion is to use block move, a group of consecutive jobs, instead of swap or insertion to improve efficiency. The latter featured a real-time reward-penalty reception based on the selected action, and was used in construction and destruction phase.

3.4. IG in DPFSP

Shih-Wei Lin et al [10] first introduced an IG-related algorithm, known as modified IG(MIG), to solve $\mathrm{DF_m|prmu|C_{max}}$. In MIG, two major amendments were made. The first amendment was that the acceptance criteria T was not a constant parameter but a sinking value to generate better quality solutions. The second amendment was that the removed elements in each iteration became dynamic to improve the stability of the solution quality.

Victor Fernandez-Viagas and Jose M.Framinan [11] introduced a bounded-search iterated greedy (BSIG) algorithm to solve $\mathrm{DF_m|prmu|C_{max}}$. The BSIG algorithm was based on the lower and upper bound of make-span and the main idea was to discard the candidates whose lower bound exceeded the current best solution. It's worth noting that three local search methods were used together to better improve the solution, which to some extent added to the complexity.

Rubén Ruiz et al [2] questioned if BSIG could outperformed SS under the same situation and introduced a two-stage IG to solve $\mathrm{DF_m|prmu|C_{max}}$, which aimed at simplifying the IG procedure in DPFSPs. In the initialization phase and reconstruction phase, NEH2_en was introduced and applied on the basis of $FRB4_k$ where either the former or the latter job was randomly selected and tested after each job insertion within the same factory. In the destruction phase, a simple operator was applied by particularly removing half of the selected jobs from the factory with largest make-span. As for local search, LS_3 was introduced on the basis of VND(a) in Ruben Ruiz et al [20] and BSIG mentioned above. After applying the improved IG, a simplified nested IG was applied again only to the factory generating the $\mathrm{C_{max}}$ using the original initialization and LS_1 in the second stage.

Inheriting the idea of the two-stage IG [2]and BSIG [11], Fernando Luis Rossi and Marcelo Seido Nagano [15] introduced an IG-RN heuristic to solve $\mathrm{DF|prmu, SDST, no\text{-}idle|C_{max}}$. The modification was that the nested IG was applied to two factories with largest make-span and the iteration in each factory was bounded by CPU-time. In the reconstruction phase, the factories were ordered in non-descending order of their make-span $\mathrm{C^f_{max}}$. Moreover, a NEH_RN initialization algorithm was proposed.

In order to avoid locally optimal solution, a restart scheme had been widely applied in consequent articles.

Jiang-Ping Huang [13] introduced an iterated greedy algorithm with a restart scheme (IGR) to solve $\mathrm{DF|prmu, SDST|C_{max}}$. Similar to the idea of Ruiz et al [2], the IGR algorithm applied a restart scheme including two operators to create new solutions and four operators to improve the current solutions in each iteration. Moreover, NEH_F initialization algorithm based on the original NEH and LSB_1 and LSB_2 local search based on the critical factory $\mathrm{F_c}$ were introduced.

Akin to the IGR mentioned above, Jia-yang Mao [14] introduced a multi-start iterated greedy (MSIG) algorithm to solve $\mathrm{DF|prmu, PM|C_{max}}$ (Here PM means preventive maintainance). MSIG set an extra iteration parameter α , and if the local solution didn't improve in α consecutive iterations, it would be abandoned and the iteration would be restarted from the heuristic initialization. Additionally, a NEH_dp initialization algorithm was proposed based on NEH2_en by

DOI: 10.54254/2755-2721/2025.LD27278

Rubén Ruiz et al [2]. In the insertion process of NEH2_dp, each available position had a certain possibility of k to be ignored. Like M.Framinan [11], the local search consisted of three methods: ISLS, ESLS and EILS.

Yuan-Zhen Li et al [16] introduced an adaptive iterated greedy algorithm(AIG) to solve $DF|prmu, mixed\ no\text{-}idle|TFT$ by applying several modifications. One innovation of the article was the swap-based local search (SLS). In SLS, two jobs in any factories were randomly swapped, then the neighboring solutions were tested to find a better solution. In the initiation phase, a new method, called $DFRB4_1$, was constructed on the basis of $FRB4_k$ introduced by Ruiz et al[21] for PFSPs. In the deconstruction phase, two parameters were constructed depending on the total flow time and completion time to decide the probability of each factory and each job being selected in the insertion procedure. Similar to [13], a restart algorithm was introduced to avoid partial optimal solution.

The common point of all IG algorithms mentioned above is that they first produce an initial solution, and all the consequent phases are depended on it. This fact means none of the algorithms is population-based. Thus, further modifications focus on the popularization of the solutions.

Yuan-Zhen Li et al [17] introduced a referenced iterated greedy (RIG) algorithm to solve $DAF|prmu, no\text{-}idle|TFT$ (DAF means Distributed assembly flow-shop). According to the different constraints of the DAMNIPFSP, six local search algorithms were designed including four referenced local search algorithms. Three of them were for products and the others were for jobs. Two reference-based reconstruction algorithms were also proposed. The key idea of referenced local search and reconstruction was that the selection was not random but based on a product sequence λ^0 and a reference solution sol^0 .

Hui Zhao et al [18] introduced a cooperative population-based iterated greedy (CPIG) algorithm to solve $DF|fmls, prmu|TFT$ (Here fmls means group scheduling). The population-based algorithm ensured that the destruction-construction and local search were carried out in a diversified population. Due to its nature, the DPFGSP was regarded as two sub-problems: group assignment among factories and the job processing in each group, referring to two sub-populations. The cooperative algorithm ensured that the two sub-populations were interconnected, which in this paper was controlled by a critical parameter $DCRate$.

Similar to the trend seen in PFSPs, machine-learning techniques have also been used with modified IG algorithms to yield better computational results.

Kasra Mousighichi and Mualla Gonca Avci [19] introduced a hybrid iterated greedy (HIG) algorithm to solve $DF|prmu, no\text{-}idle, DW|TWET$ by integrating the classical local search methods with Tabu search method, denoted as intermediate search (IS). Another innovation was the model construction of this new variant of DPFSP. Four different models based on sequence and position were constructed and each model excels under specific cases while underperforming in others, which reflects high robustness in practical application. To describe the no-idle constraints and due window, No-idle adjustment and Gap insertion were also developed. Moreover, different strategies were implemented in the destruction and construction phase.

3.5. The iterated greedy algorithm

With respect to Ruiz and Stu¨tzle [1], the following is a brief introduction of the four main steps of the IG.

Initialization: Instead of random generation, the initial job sequence is usually obtained by a well-designed algorithm. One of the commonly used initialization algorithm is the NEH heuristic first

DOI: 10.54254/2755-2721/2025.LD27278

proposed by Nawaz et al. [22] in 1983, whose modified versions can be widely founded in a significant number of related articles. The original NEH includes two steps. The first step is to sort the jobs in the descending order of their total processing time on all the machines. The second step is to schedule the first two jobs in the sorted list in the order that minimizes the make-span and then iteratively insert each subsequent job from the sorted list into the current partial schedule at the position that results in the minimum make-span.

Due to its complexity and unresolved properties such as ties, some acceleration methods and tie-breaking rules are widely applied to improve the efficiency. For instance, Liu et al [23] introduced a new tie-breaking and priority rule and there are also literature reviews of different modifications.

Destruction: The destruction phase is set to the initial solution denoted as π by randomly removing d jobs from it, resulting in two sequences: π_D of n-d jobs and π_R of d jobs. It is worth noting that in recent articles, the removed jobs are always not randomly chosen but through certain algorithms including greedy selection and blocked selection mentioned in Kasra Mousighichi and Mualla Gonca Avci [19].

Reconstruction: The reconstruction phase is to reinsert the jobs in π_R into π_D through certain algorithms like the NEH algorithms mentioned above or other possible insertions.

Local Search: The local search phase is to search a potential optimal solution within the neighborhood of the solution obtained by the reconstruction phase. Similarly, there are plenty of potential algorithms that can be used, among which a classical one is based on the insertion neighborhood of the job sequence π mentioned in Ruiz et al [1]

4. Conclusions

This article introduces a brief review of the development and incumbent status of PFSP, DPFSP, the IG algorithm and their variants. As the main problems and algorithm mentioned in this article, the classical mathematical model of PFSP and DPFSP, the notation to describe a PFSP or DPFSP regarding its constraints and objectives as well as the basic parts of IG algorithm are introduced in detail. In the literature review, sixteen selected articles are categorized into PFSP and DPFSP. The analysis and evaluation of each article mostly focus on its innovative ideas including the modification of certain phase in IG, novel algorithms leading to better computational results of IG and the change of the algorithm structure to solve a new variant of DPFSP or PFSP.

During the development of IG in PFSPs, it can be seen that more objectives and constraints are taken into consideration, leading to extension of initialization, destruction-construction and local search methods. Similar trend can also be seen in DPFSPs. However, the application of IG to solve bi-objective DPFSPs with more constraints have not been massively explored to the best of our knowledge. Thus, one possible aspect is to apply the IG algorithms to more variants and test its effectiveness compared with other heuristics.

Regarding the algorithm structure, although some changes has come up like the restart scheme, multi-start scheme, nested IG and cooperative population IG, most of the ideas are based on Ruiz's article in 2017. Thus there's still plenty of room for exploration. Integrating other heuristics such as tabu search and Q-learning might be a potential aspect and if the methods to deal with bi-objective PFSPs like PEH could be applied to bi or multi-objective DPFSPs, more fruitful results might be obtained.

DOI: 10.54254/2755-2721/2025.LD27278

References

[1] Ruben Ruiz and Thomas Stutzle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. EUROPEAN JOURNAL OF OPERATIONAL RESEARCH, 177(3): 2033–2049, MAR 16 2007.

[2] Ruben Ruiz, Quan-Ke Pan, and Bahman Naderi. Iterated greedy methods for the distributed permutation flowshop OMEGA-INTERNATIONAL scheduling problem. JOURNAL OF MANAGEMENT SCIENCE, 83: 213–222, MAR 2019.

[3] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, Discrete Optimization II, volume 5 of Annals of Discrete Mathematics, pages 287–326. Elsevier, 1979.

[4] Jose M. Framinan and Rainer Leisten. A multi objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria. OR SPECTRUM, 30(4): 787–804, OCT 2008.

[5] Michele Ciavotta, Gerardo Minella, and Ruben Ruiz. Multi-objective sequence dependent setup times per mutation flowshop: A new algorithm and a com prehensive study. EUROPEAN JOURNAL OF OPERATIONAL RESEARCH, 227(2): 301–313, JUN 1 2013.

[6] HandeOztop, M.FatihTasgetiren, Deniz Tursel Eliiyi, Quan-Ke Pan, and Levent Kandiller. An energy efficient permutation flowshop scheduling problem. EXPERT SYSTEMS WITH APPLICATIONS, 150, JUL 15 2020.

[7] Damla Yuksel, M. Fatih Tasgetiren, Levent Kandiller, and Liang Gao. An energy-efficient bi-objective no-wait permutation flowshop scheduling problem to minimize total tardiness and total energy consumption. COMPUTERS & INDUSTRIAL ENGINEERING, 145, JUL 2020.

[8] Qiu-Ying Li, Quan-Ke Pan, Liang Gao, Hong Yan Sang, Xian-Xia Zhang, and Wei-Min Li. The constrained permutation flowshop problem: An effective two-stage iterated greedy algorithm to minimize weighted tardiness. SWARM AND EVOLUTIONARYCOMPUTATION, 91, DEC2024.

[9] Damla Yuksel, Levent Kandiller, and Mehmet Fatih Tasgetiren. Q-learning guided algorithms for bi criteria minimization of total flow time and makespan in no-wait permutation flowshops. SWARM AND EVOLUTIONARYCOMPUTATION, 89, AUG2024.

[10] Shih-Wei Lin, Kuo-Ching Ying, and Chien-Yi Huang. Minimizing makespan in distributed permutation flow shops using a modified iterated greedy algorithm. INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH, 51(16): 5029–5038, AUG 1 2013.

[11] Victor Fernandez-Viagas and Jose M. Framinan. A bounded-search iterated greedy algorithm for the dis tributed permutation flowshop scheduling problem. INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH, 53(4): 1111–1123, FEB 16 2015

[12] Xue-Lei Jing, Quan-Ke Pan, Liang Gao, and Yu-Long Wang. An effective iterated greedy algorithm for the distributed permutation flowshop scheduling with due windows. APPLIED SOFT COMPUTING, 96, NOV 2020.

[13] Jiang-Ping Huang, Quan-Ke Pan, and Liang Gao. An effective iterated greedy method for the dis tributed permutation flowshop scheduling problem with sequence-dependent setup times. Swarm and Evolutionary Computation, 59: 100742, 2020.

[14] Jia-yang Mao, Quan-ke Pan, Zhong-hua Miao, and Liang Gao. An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance. EXPERT SYSTEMS WITH APPLICATIONS, 169, MAY 1 2021.

[15] Fernando Luis Rossi and Marcelo Seido Nagano. Heuristics and iterated greedy algorithms for the distributed mixed no-idle flowshop with sequence-dependent setup times. COMPUTERS & INDUSTRIAL ENGINEERING, 157, JUL 2021. 10th International Symposium on Intelligent Manufacturing and Service Systems (IMSS), Sakarya, TURKEY, SEP 09-11, 2019.

[16] Yuan-Zhen Li, Quan-Ke Pan, Jun-Qing Li, Liang Gao, and M. Fatih Tasgetiren. An adaptive iterated greedy algorithm for distributed mixed no-idle permutation flowshop scheduling problems. SWARM AND EVOLUTIONARYCOMPUTATION, 63, JUN2021.

[17] Yuan-Zhen Li, Quan-Ke Pan, Ruben Ruiz, and Hong Yan Sang. A referenced iterated greedy algorithm for the distributed assembly mixed no-idle permutation flowshop scheduling problem with the total tardiness criterion. KNOWLEDGE-BASED SYSTEMS, 239, MAR52022.

[18] Hui Zhao, Quan-Ke Pan, and Kai-Zhou Gao. A cooperative population-based iterated greedy algorithm for distributed permutation flowshop group scheduling problem. ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE, 125, OCT 2023.

[19] Kasra Mousighichi and Mualla Gonca Avci. The distributed no-idle permutation flowshop scheduling problem with due windows. COMPUTATIONAL & APPLIED MATHEMATICS, 43(4), JUN 2024.

DOI: 10.54254/2755-2721/2025.LD27278

[20] B. Naderi and Ruben Ruiz. The distributed permutation flowshop scheduling problem. COMPUTERS & OPERATIONS RESEARCH, 37(4): 754–768, APR 2010.
[21] Shahriar Farahmand Rad, Rubén Ruiz, and Naser Boroojerdian. New high performing heuristics for minimizing makespan in permutation flowshops. Omega, 37(2): 331–345, 2009.
[22] Muhammad Nawaz, E Emory Enscore, and Inyong Ham. A heuristic algorithm for the m-machine, n-job f low-shop sequencing problem. Omega, 11(1): 91–95, 1983.
[23] Weibo Liu, Yan Jin, and Mark Price. A new improved neh heuristic for permutation flowshop scheduling problems. INTERNATIONAL JOURNAL OF PRODUCTION ECONOMICS, 193: 21–30, NOV 2017.