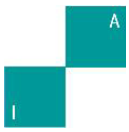
Inteligencia Artificial, 21(61) (2018), 111-123
doi: 10.4114/intartif.vol21iss61pp111-123



INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

aPaRT: A Fast Meta-Heuristic Algorithm using Path-Relinking and Tabu Search for Allocating Machines to Operations in FJSP Problem

Sahar Bakhtar, Hamid Jazayeriy*, Mojtaba Valinataj

Department of Computer Engineering, Babol Noshirvani University of Technology, Babol 47148-71167, Iran.
s.bakhtar@nit.ac.ir, jhamid@nit.ac.ir, m.valinataj@nit.ac.ir

* Corresponding author

Abstract This paper proposes a multi-start local search algorithm that solves the flexible job-shop scheduling (FJSP) problem to minimize makespan. The proposed algorithm uses a path-relinking method to generate near optimal solutions. A heuristic parameter, α, is used to assign machines to operations. Also, a tabu list is applied to avoid getting stuck into local optimums. The proposed algorithm is tested on two sets of benchmark problems (BRdata and Kacem) to make a comparison with the variable neighborhood search. The experimental results show that the proposed algorithm can produce promising solutions in a shorter amount of time.

Keywords: Job shop scheduling, Path-relinking, Local search, Tabu search, Makespan

1 Introduction

There are many scheduling problems which are very difficult to solve in a limited amount of execution time. The job shop scheduling problem (JSP) is one of the most popular scheduling types existing in practices. The JSP has been proven to be among the hardest combinatorial optimization problems [24, 10]. In JSP, a set of n jobs must be processed on m machines, where the processing of each job i consists of n_i operations. In addition, job i is composed of an ordered list of operations $O_{i1}, \ldots, O_{in_i}$ that should be executed based on the mentioned order. O_{ijk} is $j-th$ operation of job i that should be determined by the machine k where $k \in \{1, 2, \ldots, m\}$. P_{ijk} shows the processing time of $j-th$ operation of job i on machine k. Each machine is continuously available from the beginning of the scheduling process (time zero). FJSP is an extension of JSP where some machines are identical. There are several constraints on jobs and machines:

- *Total FJSP (T-FJSP)*: each operation can be processed by any machines (all the machines are equal).
- *Partial FJSP (P-FJSP)*: some operations can be processed on more than one machine (some machines are equal).

Makespan is the time needed to complete all jobs, and can be considered as one of the performance indicators for FJSP. The high level of the complexity of FJSP is the main reason for using heuristic or meta-heuristic algorithms to optimize FJSP. The main goal of this research is to achieve a feasible solution with optimized makespan in an appropriate time.

ISSN: 1137-3601 (print), 1988-3064 (on-line)

Local search is a popular meta-heuristic method for solving computationally hard optimization problems. There are many local search algorithms such as tabu search, variable neighborhood search(VNS), greedy randomized adaptive search procedures (GRASP), stochastic local search. A lot of studies have been conducted using local search method for solving FJSP [18, 32, 1, 17].

This paper proposes a meta-heuristic approach to optimize the makespan in FJSP problem. This approach tailored by path-relinking and tabu search methods. The proposed method is named path-relinking tabu (αPaRT) search.

According to the computational results, the proposed algorithm can work more efficient than the VNS algorithm. The proposed algorithm gives better results on all of the problem instances. The experimental results also show that the time needed to find best solutions is shorter in the αPaRT algorithm.

The reminder of this paper is organized as follows. Section 2 discusses related works. Then, section 3 presents the proposed algorithm, and section 4 describes the computational results obtained from applying proposed algorithm on the test datasets. Finally, section 5 provides the conclusion and suggestions for the future study.

2 Related works

Local search is the main part of heuristic algorithms. It is a practical tool and a common method to generate near-optimal solutions in a reasonable time for combinatorial optimization problems. There are many local search algorithms that have been used to optimize FJSP, such as tabu search, variable neighborhood search (VNS), greedy randomized adaptive search procedures (GRASP) and stochastic local search.

Variable neighborhood search (VNS) is one of the renowned meta-heuristic algorithms which has been successfully applied to solve optimization problems [22, 9]. It can escape from local optimums by changing neighborhood structures during the search process. There are lots of researches using VNS to solve FJSP [1, 5, 32]. Also, Amiri et al. [1] have developed a VNS algorithm to solve FJSP minimizing makespan. In this method, two neighborhood structures in terms of sequencing and three neighborhood structures related to assignment are employed to generate neighbouring solutions. The proposed algorithm in this study will be compared with Amiri's work [1].

Tabu search (TS) algorithm, proposed by Glover [6], has been successfully applied to a large number of combinatorial optimization problems [18, 19, 31]. It is usually applied in hybrid methods to solve optimization problems [18, 19, 11, 23, 14].

Heuristic search procedures ,which need to find global optimal solutions in hard combinatorial optimization problems, usually require some classes of diversification to escape from local optimality. A good way to obtain diversification is to re-start (multi-start) the algorithm from a new solution [20, 21, 2]. Multi start method has two phases. The first phase generates a new solution and the second one seeks to improve the outcome [21]. The proposed algorithm in this essay is a multi-start algorithm.

The GRASP method is another local search with iterative structure that was developed in the late 1980s and introduced by Feo and Resende [4]. It is one of the most well known multi-start methods. GRASP was first used to solve computationally difficult set covering problems [4]. A GRASP is an iterative process. These methods has been applied on FJSP [27, 26].

Moreover, path-relinking is a subsidiary method for improving local search algorithms [7]. GRASP and scatter search are two popular local search algorithms that use path-relinking method [30, 28, 29, 15]. For example, Laguna and Marti in 1999 introduced path-relinking within GRASP as a way to improve multi-start methods. Path relinking has been suggested as an approach to integrate intensification and diversification strategies [8]. This approach generates new solutions by exploring trajectories that connect high-quality solutions by starting from one of these solutions, called an initiating solution, and generating a path in the neighborhood space that leads towards the other solutions, called guiding solutions. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions [16]. The paths are different because the move selection during the normal operation is generally greedy with respect to the objective function evaluation. For example, it is customary to adopt a move selection strategy that chooses the neighborhood move that minimizes (or maximizes) the objective function value in the local sense. During path relinking, however, the main goal is to incorporate attributes of the guiding solution (or solutions) while recording the objective function value at the same time [16]. The

purpose of performing relinking moves is to find improved solutions that were not in the neighborhood of solutions visited by the original path. Laguna and Marti used a GRASP and path-relinking method, for 2-Layer Straight Line Crossing minimization [16]. They have developed a heuristic procedure based on the GRASP methodology to provide high quality solutions to the problem of minimizing straight-line crossings in a 2-layer graph.

Also, in paper [2], along with a random and greedy method for initializing solutions, a path-relinking method have been proposed to solve FJSP and optimize overall makespan. However, the proposed algorithm in this paper differs from [2] in some respect. The proposed algorithm in this paper uses a path-relinking method to generate near optimal solutions. A heuristic parameter, α, is used to assign machines to operations. Also, a tabu list is applied to avoid getting stuck into local optimums.

Moreover, paper [11] applied path-relinking (PR) Tabu search (TS) algorithms to solve the MOFJSP. The work contributes to literature on the FJSP, TS, and multi-objective optimization. First, a multi-objective and hierarchical TS with back-jump tracking (TSAB) and local search are applied to generate a set of optimal solutions from initialized solutions; Then a PR in used to create more solutions from the set derived from TS; and an effective dimension-oriented intensification search (IS) mechanism is developed to improve the TS algorithm and add variety to solutions and also avoid solutions to get stuck in small areas. The proposed algorithm in [11] is called PRMOTS+IS.

In our proposed algorithm (αPaRT), first, a heuristic ,called α, is used to create initialized solutions; Then, a path-relinking is applied in a multi-start way to find a near optimal solution; To add diversity we have used multi-start method; Finally, a very simple TS and neighborhood search is employed to avoid getting stuck in near optimums. Although, PRMOTS+IS and our proposed algorithm (αPaRT) have used path-relinking and tabu search commonly, there are some significant differences between them. PaRT has introduced to create better initializing solutions. Besides, it has applied a multi-start method to add variety instead of Dimension-oriented intensification search(IS) in PRMOTS+IS. Also, we have employed a simple TS to avoid getting stuck in local optimums while in PRMOTS+IS, a multi objective and hierarchical TS with back-jump tracking is used to generate a set of optimal solutions.

Next section presents the proposed method with regard to minimize makespan in FJSP.

3 Proposed method

In this paper, a synthetic heuristic algorithm have been used to optimize FJSP. This algorithm consists of construction phase of GRASP, path-relinking, tabu search and some simple local search methods. The proposed algorithm is titled αPaRT (**pa**th-**r**elinking **t**abu search). Figure 1 shows the flow chart of the proposed αPaRT algorithm.

αPaRT algorithm iteratively generate near-optimal solutions. Each iteration is started with two solutions x and y. It should be mentioned that solution y is created in each iteration because of the multi-start quality of the proposed algorithm. Then, a path-relinking method is used to obtain a near-optimal solution. Path-relinking needs two input solutions. It starts from x, and makes a link to y. In the link between x and y, the best solution, x_{pr}, will be chosen. Then, a local search will be applied on x_{pr} to bring it out from local optimum(x_{mls}). The movement of this local search will be added to the $TabuList$. Next, a guided local search will be executed on x_{mls} to find a better solution(x_{ols}). This solution is an input for the next path-relinking. In each iteration, best solution will be chosen among x_{pr}, x_{mls} and x_{ols}. Figure 2 demonstrates the general process of the proposed algorithm by pseudo-code. The rest of this section describes the proposed method in details.

3.1 Initialization

Lines 1 and 5 of the proposed algorithm in Figure 2 construct the initialization. In each iteration of the proposed method, a valid solution y is created by $InitialSolution()()$. This procedure has the responsibility to create solutions using the construction phase of the GRASP method. Creating a solution in $InitialSolution()()$ has two parts. (1) Allocating machines to operations and (2) sequencing operations. There are some practical criteria for each part of the creating solutions.

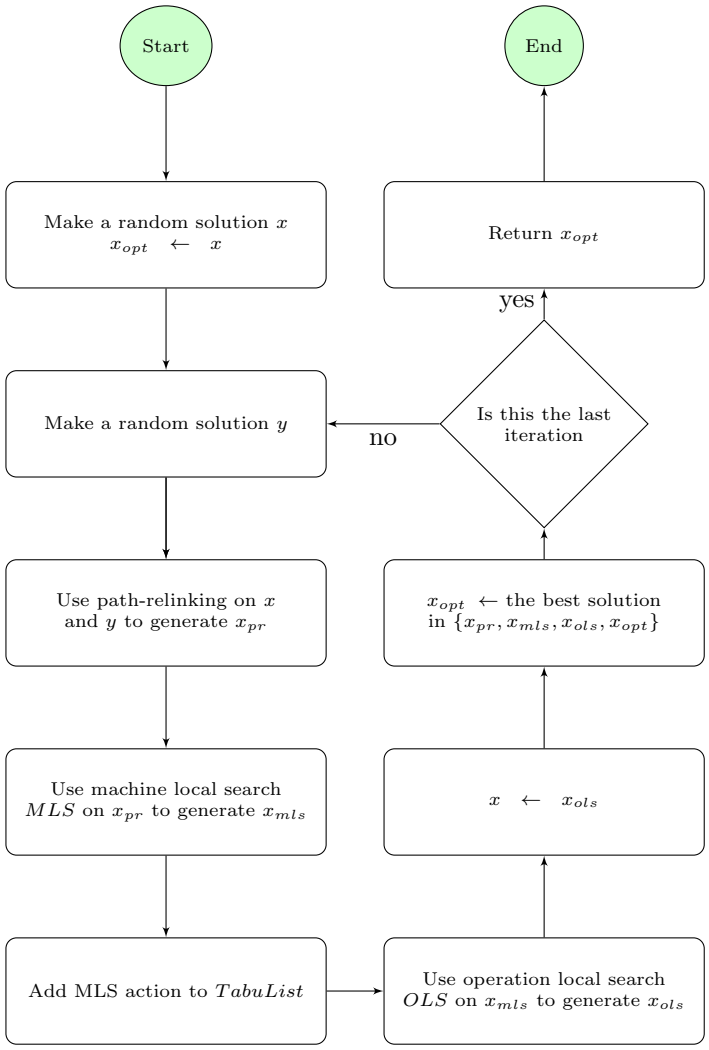


Figure 1: Flow chart of the proposed 'αPaRT' algorithm.

3.1.1 Allocating machines to operations

This step determines which machine should perform which operations. Recent studies have shown that the following methods are considered to assign a machine to an operation:

- Assign machines to operations randomly.
- Machines with lower processing time have higher priority to be allocated [25].
- Machines with minimum workload have higher priority to perform operations [1].

In this study a new heuristic is presented to assign a machine to an operation. In so doing, parameter α is defined to consider a synergetic effect of the two last methods.

$$\alpha_j = (p_{ij} + w_j)\frac{F}{f_j} \tag{1}$$

In Equation 1, p_{ij} shows processing time of operation i on machine j $(1 \leq j \leq m\ ,\ 1 \leq i \leq l)$. w_j shows the workload of machine j. Moreover, f_j is the minimum workload of machine j. F is the minimum workload of all machines.

$\alpha PaRTAlgorithm$

$x \leftarrow InitialSolution()()$
$x_{opt} \leftarrow x$
$l \leftarrow$ *the number of operations*
while *stopping condition is not satisfied* **do**
  $y \leftarrow InitialSolution()()$
  while *stopping condition is not satisfied* **do**
    $x_{pr} \leftarrow PathRelinking(x, y)$ ▷ path-relinking
  end while
  $[move, x_{mls}] \leftarrow MLS(x_{pr})$ ▷ local search
  $TabuList \leftarrow TabuList \cup move$ ▷ tabu search
  while $i < \frac{1}{10}.l$ **do**
    $x_{ols} \leftarrow OLS(x_{mls})$ ▷ local search
    $x_{opt} \leftarrow SolutionSelection(x_{pr}, x_{mls}, x_{ols}, x_{opt})$
  end while
  $x \leftarrow x_{ols}$
end while
return x_{opt}

Figure 2: Proposed algorithm with pseudo-code.

The machine with the lowest α will be selected to perform an operation. Figure 3 shows how machines are assigned to operations by introducing $AssignMachineToOperations()$. In each iteration, a machine will be selected to be allocated to an operation. In Figure 3, m is the number of machines and l is the total number of operations. Also, $\{S_i\}$ is a set of machines which have the ability to execute operation i. In FJSP, there are some operations that can be operate just by a specific machine. So, f_j is the time needed that machine j have to perform its corresponding operations.

3.1.2 Sequencing operations

FJSP can be solved by providing the sequence of operations. This paper proposes an combinatorial selection of operations. The selection can be done by the following policies:

- the random selection.
- the most remaining work selection(MRW) [3].
- the most number of remaining operations(MRO) [25].
- the shortest processing time(SPT) [3].

These policies may have different chances to be applied for operation selection. An experiment has been conducted to show , which percentage can achieve better solutions. This experiment has used different percentages to create new solutions for obtaining lower makespan. Table 1 illustrates the resulted makespan by applying the proposed algorithm with different percentages of random selection on BRdata. First column shows the name of each dataset and the other columns demonstrate the percentage of random selection to create new solutions. For example, if the percentage of random selection is 40, the other 60 percent will be equally divided among MRW, MRO and SPT. Therefore, in this study , the following chances are used: 5% for random selection and the remain 95% is equally divided as 31.6% for MRW, 31.6% for MRO and 31.6% for SPT.

As it can be seen from Table 1, good solutions are often generated when random operation selection is equal to 5%. Therefore, in this study , the following chances are used: 5% for random selection, 31.6% for MRW, 31.6% for MRO and 31.6% for SPT.

AssignMachineToOperations

$w_j \leftarrow 0\ (1 \leq j \leq m)$
Determine $\{f_1, f_2, \cdots, f_m\}$
$F \leftarrow \sum_{j=1}^{m} f_j$
$f'_j \leftarrow f_j/F\ (1 \leq j \leq m)$
for $i \leftarrow 1$ **to** l **do**
    $S_i \leftarrow$ *list of machines capable of performing operation* i
    $rand \leftarrow random(1, 100)$
    if $rand \leq 5$ **then**
        $M \leftarrow$ *Select a machine from set* $\{S_i\}$ *randomly*
    else
        for $j \leftarrow 1$ **to** $|S_i|$ **do**
            $\alpha_j \leftarrow (p_{ij} + w_j)/f'_j$
        end for
        $M \leftarrow \{j | \min_{j=1}^{|S_i|} \alpha_j\}$
    end if
    Assign machine M *to operation* i
    $w_j = w_j + p_{ij}$
end for

Figure 3: Assign machines to operations pseudo-code.

Table 1: Mean makespan on BRdata with different Random percentages of operation selection

	Random percentage of operation selection							
	0	0.03	**0.05**	0.10	0.20	0.30	0.50	1
$mk1$	42	41.3	**40.8**	42	42.8	43.1	45.2	46.8
$mk2$	28	27.8	**27.2**	27.5	29	30.4	31.2	33.6
$mk3$	204	204	**204**	204	204	204	204.4	205.1
$mk4$	62	61.8	**61**	61.4	62.2	63	64.3	67.8
$mk5$	175	174.5	**173**	173.9	175	175.4	177.6	179.9
$mk6$	61	60.6	**59.9**	61.7	61.9	63.4	65.8	68.1
$mk7$	142	141.3	**139.9**	141.6	142.6	142.9	144.1	146.8
$mk8$	523	523	**523**	523	523	523	523	523
$mk9$	310	**307.5**	307.6	308.8	310.6	312.1	314.8	317.4
$mk10$	210	210.4	**206.4**	207.6	210.5	212.8	214.8	218.9

3.2 Synthetic heuristics

Lines 6 to 16 of the proposed algorithm in Figure 2 construct the Synthetic heuristics. This section describes the main part of the algorithm. It consists of a multi start path-relinking method and a tabu search. Figure 4 demonstrates the synthetic heuristics including path-relinking, tabu-search and local searches on machines and operations.

3.2.1 Path-relinking

Path-relinking is the most important part of the proposed algorithm. Figure 5 demonstrates the process of $PathRelinking(x, y)$. In each iteration, path-relinking procedure needs two input solutions(x and y). Each of them consists of l operations $(x_1, \cdots, x_l)$ and $(y_1, \cdots, y_l)$. In $i-th$ iteration of the path-relinking algorithm, x_i will be replaced by y_i in solution x. After applying new operation in x, the rest of operations in x will be updated to make a valid solution for FJSP (x_{new}). After finishing the algorithm, solution x and y are the same. The best solution in the path(x_{pr}) will be chosen to move to the next step.

$PathRelinking(x, y)$ is used in the proposed algorithm. In each iteration there is a new solution y resulted from $InitialSolution()()$ and solution x resulted from the previous iteration of the proposed

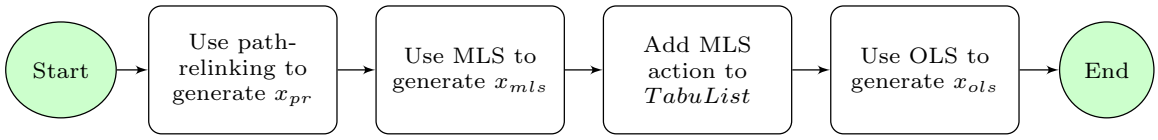


Figure 4: Steps in proposed αPaRT method to generate a new solution.

$Path - Relinking(x, y)$
1: **if** $evaluation(x) < evaluation(y)$ **then**
2: $x_{pr} = x$
3: **else**
4: $x_{pr} = y$
5: **end if**
6: **for** $i = 1$ **to** l **do**
7: *replace* x_i *by* y_i
8: x_{new} =*sort the rest of operation in* x
9: **if** $evaluation(x_{new}) < evaluation(x_{pr})$ **then**
10: $x_{pr} = x_{new}$
11: **end if**
12: **end for**
13: **return** x_{pr}

Figure 5: Pseudo-code of Path-relinking algorithm

algorithm to start $PathRelinking(x, y)$ again.

3.2.2 Tabu search

A tabu search method is used to avoid getting stuck at local optimums. After applying path-relinking, MLS local search is applied on x_{pr} to generate x_{mls} and bringing x_{pr} out of local optimum. The movement of MLS named *move* , is added to the $TabuList$. In the next iterations, this movement is forbidden. In each iteration, a movement is added to the $TabuList$ until, the number of iteration is finished or the best solution is obtained.

3.2.3 Local searches on machines and operations

In this part two local searches is introduced. These local searches work as follows:

- *Machine Local Search(MLS)*: This local search will change the machine that run operation i in the given solution x. At first, a machine with the longest finish time is selected. The time consumed by the selected machine determines the makespan. Afterwards, an operation is chosen from this machine randomly. This operation, i, will be assigned to another machine with the minimum α according to Equation 1. Solution x will change into x_{mls} by applying this local search.

- *Operation Local Search(OLS)*: This local search will change the sequence of operations on the given solution x. At first, machine with the longest finish time is selected. Then, an operation is chosen from this machine randomly. This operation will be substituted by the previous operation in the given solution x. If it is not feasible in respect of FJSP constraints, the selected operation will be substituted by the next operation. Solution x will change into x_{ols} by applying this local search.

3.3 Solution selection

This part updates best solution x_{opt} by evaluating the resulted solution x_{pr}, x_{mls} and x_{ols} and choosing the best solution in each iteration.

$$x_{opt} = best\{x_{pr}, x_{mls}, x_{ols}\} \tag{2}$$

Table 2: FJSP BRdata instances by [3] . (Available on http://www.idsia.ch/ monaldo/fjsp.html.

dataset	*njob*	*nmac*	*nop*	*meq*	*proc*
$mk1$	10	6	5-7	3	1-7
$mk2$	10	6	5-7	6	1-7
$mk3$	15	8	10-10	5	1-20
$mk4$	15	8	3-10	3	1-10
$mk5$	15	4	5-10	2	5-10
$mk6$	10	15	15-15	5	1-10
$mk7$	20	5	5-5	5	1-20
$mk8$	20	10	5-10	2	5-10
$mk9$	20	10	10-15	5	5-10
$mk10$	20	15	10-15	5	5-20

Table 3: FJSP Kacem data [13] .

dataset	*njob*	*nmac*	*tnop*
$Instance1$	4	5	12
$Instance2$	10	7	29
$Instance3$	10	10	30
$Instance4$	15	10	56

in Equation 2, x_{pr} is resulted from path-relinking, x_{mls} in obtains by local search on machines and x_{ols} is resulted from local search on operations.

4 Evaluation

This section describes the implementation and evaluation of the proposed algorithm. The proposed algorithm is applied on some famous datasets. Then, the resulted makespan is compared with other state-of-the-art algorithms.

4.1 Dataset

There are two common popular datasets which are used to evaluate FJSP. The first dataset consists of 15 test problems from Brandimarte in 1993 [3]. The data was randomly generated using a uniform distribution between given limits. The number of jobs ranges from 10 to 30, the number of machines ranges from 4 to 15 and the number of operations for each job ranges from 3 to 15. Table 2 shows the details of this dataset. In this dataset, *njob* is the number of jobs, *nmac* is the number of machines, *nop* is the number of operations per job which varies between a minimum and a maximum values, *meq* is the number of equal machines and *proc* is the processing time per operation that varies between a minimum and maximum values [3]. The problem dimension can be seen by $njob \times nmac \times nop$. According to Table 2, $mk9$ and $mk10$ have the largest dimensions. In this dataset, $mk1$ is the simplest and $mk10$ is the most complicated problem.

Second dataset is Kacem data. Kacem et al. designed four instances for the FJSP with total flexibility and varying numbers of operations per job [13]. An overview of the four instances is provided in Table 3. In this dataset, *tnop* indicates the total number of operations.

4.2 Evaluation metric

In order to conduct the experiments, the proposed algorithm is implemented in Matlab application. Assume $\tau_i(m_i)$ is the idle time of machine m_i and $\tau_c(m_i)$ is the time that machine m_i was performing related operations. In addition, T_i is the needed time for machine m_i to complete its process.

T_i will be obtain by Equation 3.

$$T_i = \tau_i(m_i) + \tau_c(m_i) \tag{3}$$

The maximum of set $\{T_1, T_2, \ldots, T_m\}$ is the total makespan.

$$makespan = max\{T_1, T_2, \ldots, T_m\} \tag{4}$$

In Equation 4 , m is the number of machines .

4.3 Computational results

This section describes the experimental tests used to evaluate the effectiveness of the proposed path-relinking algorithm.

At first, the quality of initial solutions will be examined. In this study, the parameter α is proposed as a heuristic to select a high priority machine to perform an operation. An experiment has been applied to show the effect of the proposed α on initializing solutions. This experiment contains calculating makespan by creating new solutions using the proposed α and comparing the results with the other state-of-the-art algorithms. Different algorithms create initial solutions using different structures. In this experiment, the methods of [1](VNS) and [19](TSPCB) are used to compare with the proposed heuristic, α.

Table 4 reports a comparison on obtained makespans by use of different initializing methods. First column displays the name of each datasets and the other columns show the average makespan of 100 times applying the proposed heuristic α, [1] and [19], respectively. Also, figure 6 shows the effectiveness of the proposed α on makespan.

Table 4: Mean makespan of initial solutions on BRdata

dataset	by proposed α	[1]	[19]
$mk1$	**58.82**	91.88	72.54
$mk2$	**43.33**	84.25	70.35
$mk3$	**246.31**	500.62	409.24
$mk4$	**106.99**	149.44	124.35
$mk5$	**253.59**	331.86	300.89
$mk6$	**109.29**	245.76	189.23
$mk7$	**232.71**	375.28	325.89
$mk8$	**624.75**	862.99	829.4
$mk9$	**489.75**	735.33	652.76
$mk10$	**356.52**	652.87	564.87

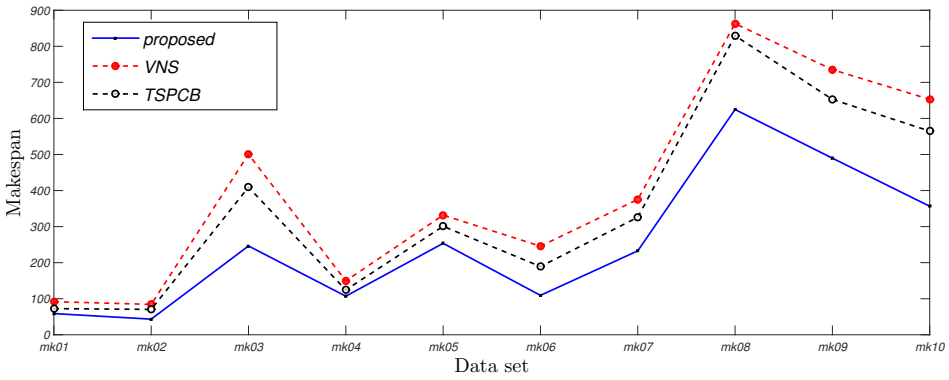


Figure 6: Mean makespan of the initial solutions on BRdata.

As it can be seen, the proposed heuristic is able to generate better initialized solutions which can make the process of reaching final solution faster.

After finding the best setups for the used local searches in proposed αPaRT, results obtained from αPaRT is compared with three other methods: VNS [1], GA [25] and TSPCB [19].

The non-deterministic nature of these algorithms made it necessary to carry out multiple runs on the same problem instance in order to obtain reasonable results. For each problem, the best solution is selected after fifty runs of the algorithms. Finally, the best solution for each instances is selected. in addition, for the proposed αPaRT and VNS the worst and average makespans are also reported in Table 5.

Table 5: Comparison of the makespan resulted from the proposed method (αPaRT) and the others.

Dataset		Proposed αPaRT method			VNS			GA	TSPCB
		best	worst	mean	best	worst	mean		
BRdata	mk1	**40**	42	40.8	**40**	42	40.9	**40**	**40**
	mk2	**26**	31	27.2	**26**	32	27.6	**26**	**26**
	mk3	**204**	204	204	**204**	204	204	**204**	**204**
	mk4	**60**	65	61	**60**	64	61.4	**60**	62
	mk5	**172**	175	173	173	176	174.5	173	**172**
	mk6	**59**	62	59.9	60	68	62	63	65
	mk7	**139**	141	139.9	140	142	140.8	**139**	140
	mk8	**523**	523	523	**523**	523	523	**523**	**523**
	mk9	**307**	310	307.6	**307**	312	309.9	311	310
	mk10	**204**	210	206.4	207	215	209.5	212	214
Kacem data	*Instance*2	**14**	14	14	**14**	14	14	-	-
	*Instance*3	**7**	7	7	**7**	7	7	-	-
	*Instance*4	**11**	12	11.7	12	12	12	-	-

In general, Table 5 indicates the proposed αPaRT is at least as good as the other methods in all cases (results on Kacemdata are not reported by some of studies). Turning to details, the proposed αPaRT gives the best makespan in instances $mk06$ and $mk10$ among the other three algorithms listed. Besides, with regard to Table 5, it is apparent that the proposed αPaRT have better average makespan than VNS in all cases. It should be noticed that because of lack of information about the results of GA and TSPCB, the worst and average of them are neglected.

Furthermore, the best, worst and average makespans of the resulted non-dominated solutions, which were reported in appendix of paper [11] for BRdata, are shown in Table 6. With regard to Table 6, in a single-objective perspective, PaRT has achieved much better makespans in comparison with PRMOTS+IS.

Table 6: The best, worst and average makespans

Dataset		αPaRT			PRMOTS+IS[11]		
		best	worst	mean	best	worst	mean
BRdata	mk1	**40**	42	40.8	**40**	45	41.8
	mk2	**26**	31	27.2	27	33	29
	mk3	**204**	204	204	**204**	330	260.9
	mk4	**60**	65	61	63	146	85.2
	mk5	**172**	175	173	174	209	174.1
	mk6	**59**	62	59.9	63	106	77.7
	mk7	**139**	141	139.9	141	217	158.1
	mk8	**523**	523	523	523	587	551.8
	mk9	**307**	310	307.6	310	310	454
	mk10	**204**	210	206.4	222	210	308

Moreover, to evaluate the time complexity, a comparison of the proposed αPaRT and VNS is reported

in Table 7. In this comparison, for each instance of the dataset, running of the proposed αPaRT is terminated when it reaches to VNS result.

Table 7: Time comparison of the running proposed αPaRT method and VNS

		VNS		αPaRT		
dataset	instance	makespan	time(s)	makespan	time(s)	improvement
BRdata	$mk1$	40	87.2	40	**70.3**	24%
	$mk2$	26	5173.1	26	**5170.7**	0.4%
	$mk3$	204	68	204	**1.5**	4400%
	$mk4$	60	11442	60	**11233**	1.8%
	$mk5$	173	11546	173	**11517**	0.2%
	$mk6$	59	12666	59	**12087**	4%
	$mk7$	140	11031	140	**11002**	0.2%
	$mk8$	523	86.9	523	**70.2**	23%
	$mk9$	307	72142	307	**71154**	1.3%
	$mk10$	207	12602	207	**11577**	8%
Kacemdata	$Instance2$	14	216.9	14	**194.2**	11%
	$Instance3$	7	4786.7	7	**4657.2**	2.7%
	$Instance4$	12	7545.4	12	**7502.3**	0.5%

Computational results in Table 7 show that almost in all cases the proposed αPaRT algorithm could achieve the same solutions faster than the VNS algorithm.

5 Conclusion

This paper introduces a path-relinking search algorithm for the flexible job-shop scheduling problem. Minimization of makespan is considered as the objective function. This algorithm uses a random-greedy structure to create initial solutions. Then, path-relinking and tabu search methods are applied to obtain near-optimal solutions. Finally, the proposed algorithm is tested on BRdata introduced in [3] and Kacem data presented in [12]. The computational results demonstrate that the proposed algorithm achieves improvements compared to VNS, GA and TSPCB. These improvements include lower average makespan in comparison with VNS almost in all cases and also the lower or at least equal amount of best makespan compared to GA and TSPCB in all instances. Additionally, it is revealed that the proposed method is much faster than VNS.

There are some directions for future works. Path-relinking may be used as a part of an evolutionary algorithm. Another interesting direction would be the evaluation of the proposed method in a multi-objective problem. for example in FJSP, makespan, tardiness, workload or energy could be considered as objectives.

References

[1] M Amiri, M Zandieh, M Yazdani, and A Bagheri. A variable neighbourhood search algorithm for the flexible job-shop scheduling problem. *International journal of production research*, 48(19):5671–5689, 2010.

[2] Sahar Bakhtar, Hamid Jazayeriy, and Mojtaba Valinataj. A multi-start path-relinking algorithm for the flexible job-shop scheduling problem. In *Information and Knowledge Technology (IKT), 2015 7th Conference on*, pages 1–6. IEEE, 2015.

[3] Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations research*, 41(3):157–183, 1993.

[4] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[5] Jie Gao, Linyan Sun, and Mitsuo Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35(9):2892–2907, 2008.

[6] Fred Glover. Tabu search: A tutorial. *Interfaces*, 20(4):74–94, 1990.

[7] Fred Glover and Manuel Laguna. Tabu search principles. In *Tabu Search*, pages 125–151. Springer, 1997.

[8] Fred Glover, Manuel Laguna, and Rafael Martí. Fundamentals of scatter search and path relinking. *Control and cybernetics*, 29:653–684, 2000.

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

[10] Anant Singh Jain and Sheik Meeran. *A multi-level hybrid framework for the deterministic job-shop scheduling problem.* PhD thesis, Citeseer, 1998.

[11] Shuai Jia and Zhi-Hua Hu. Path-relinking tabu search for the multi-objective flexible job shop scheduling problem. *Computers & Operations Research*, 47:11–26, 2014.

[12] Imed Kacem, Slim Hammadi, and Pierre Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(1):1–13, 2002.

[13] Imed Kacem, Slim Hammadi, and Pierre Borne. Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and computers in simulation*, 60(3):245–276, 2002.

[14] Shuhei Kawaguchi and Yoshikazu Fukuyama. Reactive tabu search for job-shop scheduling problems considering peak shift of electric power energy consumption. In *Region 10 Conference (TENCON), 2016 IEEE*, pages 3406–3409. IEEE, 2016.

[15] Manuel Laguna. Scatter search. In *Search Methodologies*, pages 119–141. Springer, 2014.

[16] Manuel Laguna and Rafael Marti. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11(1):44–52, 1999.

[17] Jun-Qing Li, Quan-Ke Pan, and Jing Chen. A hybrid pareto-based local search algorithm for multi-objective flexible job shop scheduling problems. *International Journal of Production Research*, 50(4):1063–1078, 2012.

[18] Jun-qing Li, Quan-ke Pan, and Yun-Chia Liang. An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, 59(4):647–662, 2010.

[19] Jun-Qing Li, Quan-Ke Pan, PN Suganthan, and TJ Chua. A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem. *The international journal of advanced manufacturing technology*, 52(5-8):683–697, 2011.

[20] Rafael Martí. Multi-start methods. In *Handbook of metaheuristics*, pages 355–368. Springer, 2003.

[21] Rafael Martí, Mauricio GC Resende, and Celso C Ribeiro. Multi-start methods for combinatorial optimization. *European Journal of Operational Research*, 226(1):1–8, 2013.

[22] Nenad Mladenović and Pierre Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

[23] Yasuhiko Morinaga, Masahiro Nagao, and Mitsuru Sano. Balancing setup workers' load of flexible job shop scheduling using hybrid genetic algorithm with tabu search strategy. *International Journal of Decision Support Systems*, 2(1-3):71–90, 2016.

[24] Wim PM Nuijten and Emile HL Aarts. A computational study of constraint satisfaction for multiple capacitated job shop scheduling. *European Journal of Operational Research*, 90(2):269–284, 1996.

[25] F Pezzella, G Morganti, and G Ciaschetti. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 35(10):3202–3212, 2008.

[26] M Rajkumar, P Asokan, N Anilkumar, and T Page. A grasp algorithm for flexible job-shop scheduling problem with limited resource constraints. *International Journal of Production Research*, 49(8):2409–2423, 2011.

[27] M Rajkumar, P Asokan, and V Vamsikrishna. A grasp algorithm for flexible job-shop scheduling with maintenance constraints. *International Journal of Production Research*, 48(22):6821–6836, 2010.

[28] Mauricio GC Resende, Rafael Martí, Micael Gallego, and Abraham Duarte. Grasp and path relinking for the max–min diversity problem. *Computers & Operations Research*, 37(3):498–508, 2010.

[29] Mauricio GC Resende, Celso C Ribeiro, Fred Glover, and Rafael Martí. Scatter search and path-relinking: Fundamentals, advances, and applications. In *Handbook of metaheuristics*, pages 87–107. Springer, 2010.

[30] Mauricio GC Resendel and Celso C Ribeiro. Grasp with path-relinking: Recent advances and applications. In *Metaheuristics: progress as real problem solvers*, pages 29–63. Springer, 2005.

[31] Mohammad Saidi-Mehrabad and Parviz Fattahi. Flexible job shop scheduling with tabu search algorithms. *The International Journal of Advanced Manufacturing Technology*, 32(5-6):563–570, 2007.

[32] M Yazdani, M Amiri, and M Zandieh. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, 37(1):678–687, 2010.