Journal of Manufacturing Systems 56 (2020) 425–451

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Greedy randomized adaptive search for dynamic flexible job-shop scheduling

Adil Baykasoğlu[a,*], Fatma S. Madenoğlu[b], Alper Hamzadayı[c]

[a] Dokuz Eylül University, Department of Industrial Engineering, Izmir, Turkey
[b] Abdullah Gul University, Department of Management Science, Kayseri, Turkey
[c] Van Yüzüncü Yıl University, Department of Industrial Engineering, Van, Turkey

ARTICLE INFO

Keywords:
Flexible job shop scheduling
Rescheduling
Dynamic scheduling
GRASP

ABSTRACT

Dynamic flexible job shop scheduling problem is studied under the events such as new order arrivals, changes in due dates, machine breakdowns, order cancellations, and appearance of urgent orders. This paper presents a constructive algorithm which can solve FJSP and DFJSP with machine capacity constraints and sequence-dependent setup times, and employs greedy randomized adaptive search procedure (GRASP). Besides, Order Review Release (ORR) mechanism and order acceptance/rejection decisions are also incorporated into the proposed method in order to adjust capacity execution considering customer due date requirements. The lexicographic method is utilized to assess the objectives: schedule instability, makespan, mean tardiness and mean flow time. A group of experiments is also carried out in order to verify the suitability of the GRASP in solving the flexible job shop scheduling problem. Benchmark problems are formed for different problem scales with dynamic events. The event-driven rescheduling strategy is also compared with periodical rescheduling strategy. Results of the extensive computational experiment presents that proposed approach is very effective and can provide reasonable schedules under event-driven and periodic scheduling scenarios.

1. Introduction

Scheduling is a decision-making process [1], and has been described as assigning resources to jobs to fulfill the determined objectives. Scheduling is usually divided into two categories: static and dynamic [2]. All jobs are available for generating a schedule in a static scheduling environment. The generated schedule is assumed to be not modified after its execution [3]. The flexible job shop scheduling problem (FJSP) is a complex scheduling problem, many studies carried out on the static variant of this problem. FJSP is in the NP-hard problem class [4]. FJSP incorporates machine assignment that is for choosing a capable one in the set of machines for assigning to each operation and operation sequencing is to sequence the assigned operations on the machines with the aim of obtaining a feasible schedule [5].

Dynamic scheduling is a method in which different and new jobs emerge during the production period and thus updates have to be made in the previous schedule. Whenever an operation of any job is finished, it is moved out of the system. The unpredictable events, such as machine breakdown and repair may occur. Thus, a feasible schedule may become inapplicable after releasing it into the production area [3]. In today's production systems, most scheduling problems take place in such dynamic environments. In dynamic environments, rescheduling is necessary in order to update the ongoing schedule. Rescheduling is a short-term decision-making process that composes and updates the schedule according to the current state of the system and the overall system requirements [6]. Therefore, in order to minimize its impact on the system performance, an existing production schedule needs to be updated regularly in order to reduce the unexpected events' impact on the overall performance.

Order Review and Release (ORR) is a management system that permits passing the production order from the planning stage into the production stage. The customer orders arrive to the production system continuously, but arrival does not require releasing an order into the production environment instantly. On the contrary, a backlog file is called as pre-shop pool stores orders, decoupling the planning stage from the production stage [7]. With this way, ORR mechanism can be considered as the linkage between the production planning and the production control [8].

It is emphasized in Baykasoğlu and Ozsoydan [9] that the constructive approaches are more effective in many ways for dynamic combinatorial optimization problems than the evolutionary or population-based approaches, and each generation is re-initialized from

* Corresponding author.
E-mail addresses: adil.baykasoglu@deu.edu.tr (A. Baykasoğlu), selen.madenoglu@agu.edu.tr (F.S. Madenoğlu), alperhamzadayi@yyu.edu.tr (A. Hamzadayı).

https://doi.org/10.1016/j.jmsy.2020.06.005
Received 31 July 2019; Received in revised form 26 March 2020; Accepted 13 June 2020
Available online 09 July 2020
0278-6125/ © 2020 The Society of Manufacturing Engineers. Published by Elsevier Ltd. All rights reserved.

scratch in the constructive approaches. GRASP [10] is a constructive and multi-start strategy that uses both construction and improvement heuristics to generate high quality solutions. In GRASP, a solution is constructed step-by-step via a construction heuristic. Each time only one part of a solution is added to the constructed solution. The part of a solution to be added to the constructed solution is determined by randomly choosing a feasible element from a candidate list, in regard to a problem specific greedy rule.

Because of the complexity of FJSP, none of the exact solution methods has been proven to be effective so far, therefore, meta-heuristic algorithms may be preferred to construct feasible and good schedules. Brandimarte [11] presented a hybrid tabu search algorithm for FJSP. Baykasoğlu [5] and Baykasoğlu et al. [12] introduced a linguistic-based metaheuristic modeling for solving FJSP. Genetic and hybrid genetic algorithm [13–15], a variable neighborhood search algorithm [16], a variable neighborhood descent algorithm [17], an artificial immune algorithm [18], GRASP algorithm [19–21], a teaching–learning-based optimization algorithm [22], a discrete harmony search algorithm [23,24], a quantum-behaved particle swarm optimization [25], a novel biomimicry hybrid bacterial foraging optimization algorithm [26] are some of the metaheuristic based algorithms developed for solving FJSP so far. The reader can refer to Gao et al. [27] for the detailed review on swarm intelligence and evolutionary algorithms applications about solving FJSPs.

The dynamic FJSP (DFJSP) has been studied for many years and many solution techniques have been proposed for this problem. Kim and Kim [28] developed a simulation-based real time scheduling methodology to solve DFJSP. Branke and Mattfeld [29] proposed an evolutionary algorithm for DFJSP with periodic rescheduling policy under new order arrivals. Buyurgan and Saygin [30] proposed a multicriteria decision making framework that employs the analytical hierarchy process in advanced manufacturing systems for real-time scheduling and part routing. Gholami and Zandieh [31] incorporated simulation into a genetic algorithm to solve DFJSP with machine breakdowns under makespan and mean tardiness criteria. Fattahi and Fallahi [32] presented a mathematical model and a genetic algorithm by considering efficiency and stability objectives. Al-Hinai and ElMekkawy [33] presented a hybrid genetic algorithm under robustness and stability criteria. Rajabinasab and Mansour [34] proposed a multi agent system in order to obtain good quality and robust schedules under order arrivals, uncertain processing times, machine breakdown and process plan flexibility. The results of the proposed multi-agent-based approach were compared with the five dispatching rules from literature. A hybrid evolutionary algorithm is presented by Lin et al. [35] for dealing with a reactive FJSP under machine breakdowns, due dates change and total tardiness objective. He and Sun [36] proposed a new approach using genetic algorithm in order to obtain robust and stable performance for FJSP with machine breakdown. A heuristic approach for DFJSP was presented by Nie et al. [37]. In their approach gene-expression programming was employed along with their heuristic to generate competitive solutions under makespan, mean flow time and mean tardiness objectives. Negahban and Smith [38] provided a comprehensive review of discrete event simulation publications published between 2002 and 2013 with a particular focus on applications in manufacturing. Shen and Yao [39] presented an evolutionary algorithm and mathematical model for solving FJSP with order arrivals, machine breakdowns considering multiple efficiency and stability objectives. Ning et al. [40] presented a new multi-objective mathematical model using novel constraints. Periodic and event driven rescheduling policy were composed to solve DFJSP using a hybrid multi-phase quantum particle swarm algorithm. [23,24]) presented two stage artificial bee colony algorithm to solve FJSP with fuzzy processing times and new job insertion under makespan objective. Jin et al. [41] proposed a mixed integer linear programming model and non-dominated sorting genetic algorithm with three criteria for an integrated process planning and scheduling problem in order to analyze the impact of periodic and event-driven rescheduling strategies. [42,43]) presented a discrete Jaya algorithm for solving FJSP with new job insertion under bi-objective optimization. Uhlmann and Frazzon [44] reviewed the production rescheduling process and identified the integration among industries and real cases applications for rescheduling. [42,43]) proposed a discrete harmony search algorithm to solve scheduling and rescheduling problems with increasing processing time and new job insertion under the objectives of minimizing the maximum completion time and the mean of earliness and tardiness. Zadeh et al. [45] presented a heuristic model for solving DFJSP considering variable processing times under minimizing makespan. Zhou et al. [46] suggested three types of hyper-heuristic methods for FJSP with job arrivals. Discrete event simulation was used to assess the results for the proposed machine assignment rules and sequencing rules. The proposed approach included the surrogate assisted cooperative coevolution genetic programming. The results of the proposed approach were compared with the results of the well-known benchmark rules for machine assignment rules and sequencing rules. The results demonstrated that the proposed approach is competitive than others considering convergence speed and the quality of the results. Cao et al. [47] proposed an adaptive heterogeneous earliest finish time algorithm to address the makespan objective in FJSP with order arrivals. Ozturk et al. [48] proposed two approaches in combination with gene expression programming for DFJSP. Hu et al. [49] proposed a novel Petri-net-based dynamic scheduling approach and employed the deep Q-network and a graph convolutional network for solving dynamic scheduling problem of flexible manufacturing systems. Luo [50] addressed FJSP with new job insertions. A deep Q-network (DQN) is presented to determine the appropriate dispatching rules. Six composite dispatching rules are used to minimize the total tardiness. The results of the proposed approach perform significantly better than the proposed composite dispatching rules and other well-known dispatching rules for both trained and untrained production configurations.

To the best of our knowledge, in the literature, GRASP has not been tailored to generate new schedules during dynamic flexible shop floor environment. We noticed that few studies take into account more than one real-time situation such as order arrivals, machine breakdowns, order cancellations, due date changes, urgent order arrivals. Also, all of the previous papers don't consider machine capacity constraints in addition to sequence dependent setup times and dynamic events. This review reveals that no constructive algorithm is proposed for DFJSP so far. This paper fulfills this gap for this problem.

The main motivation of the present study is to propose a multi-start and constructive search strategy to solve FJSP and DFJSP by addressing several real-life situations. To the best of our knowledge, Giffler and Thompson's [51] (G&T) algorithm has not been integrated with GRASP to solve FJSP and DFJSP in the literature. Baykasoğlu and Karaslan [52] applied GRASP with G&T to dynamic job shop scheduling problems without considering flexibility. This paper also aims to extend authors previous study in order to analyze the effect of flexibility on scheduling performance by proving an effective scheduling mechanism for dynamic FJSP. In this paper, for the first time GRASP/G&T algorithm is utilized for solving FJSP and DFJSP with unexpected events such as changes in due dates, order cancellations etc. In DFJSP, sequence dependent setup times, machine capacity constraints, ORR mechanism, order acceptance/rejection decisions that alleviates the production capacity adjustment problem are also considered in order to better reflect the scheduling factors encountered in real manufacturing environments. The static FJSP is studied under the minimizing makespan objective, whereas a lexicographic method is utilized in DFJSP in order to assess four objectives. In lexicographic method, the objectives are ranked in the order of importance from best to worst. Firstly, the performance of the proposed GRASP in solving the static FJSP is tested on three groups of benchmarks from the literature and the generated results are compared with the best-known algorithms. Afterwards, the performance of the GRASP based event-driven policy and periodical rescheduling policy are compared with each other and with basic

dispatching rules over differing levels of rescheduling time interval, machine flexibility, shop utilization and due date tightness. It is observed that the periodical rescheduling strategy may provide better schedules with the penalty of being less responsive when it is compared to other methods.

Rest of the paper is organized as follows. Section two describes the details of DFJSP. Section three presents the proposed algorithm. Experimental study is described and performed in section four and last section comprises our conclusions.

2. Statement the proposed DFJSP

In this section, a detailed explanation of the problem is given.

2.1. Problem description

In the literature, it is usually assumed that all machines are constantly available during the production time. On the other hand, machines may not always be available in real-life production systems. Machine's unavailability intervals can be planned, and by doing that, some tasks (such as preventive maintenance, cleaning, etc.) may be performed within a planned time for each machine in advance. In this study, these kind of availability constraints are taken into consideration. In other words, the availability of each machine may be different from each other during a scheduling period. Contrary to the usual assumption of the FJJSP, some additional constraints are considered in this study to better reflect the real manufacturing environments. The sequence dependent and non-anticipatory setup times are also considered in this study. Other details of the studied problem are discussed in the subsequent sections.

2.2. Event-driven rescheduling

Five common types of disruptions are taken into consideration: “machine breakdowns”, “new order arrivals”, “changes of the due dates”, “arrival of urgent orders”, and “order cancellation” for making decision about order rejection or acceptance in ORR mechanism.

2.2.1. Order acceptance or rejection in ORR

In production systems, not all new orders can be accepted for several issues, such as machine capacity constraints. The decision to accept or reject an order depends on arriving time, completion time, due date and available store capacity. A trial schedule can be constructed by considering these factors in order to make an effective decision. The order can be accepted if it may be shipped on the requested delivery time within the current capacity. The new coming order may be rejected if its processing requirements exceed the available capacity. This decision is used to effectively manage capacity and ensure customer satisfaction. As an ORR strategy, the forward finite loading mechanism [53] is used to decide whether a new order, which doesn’t begin processing is directed to production area or not. By considering the available capacity of the shop floor, the waiting orders are tentatively scheduled. Then, the decision for a certain order is made by considering delivery time of this order. It is assumed that all materials are available in the shop floor.

2.2.2. Machine breakdown

Whenever this unexpected event occurs, it is presumed that the ongoing operation has not been interrupted until it is completed. New schedule is generated for the non-started operations considering the repair time.

2.2.3. Change of the due dates

Although the due dates are declared in advance by the customers, the due dates may be postponed or retrieved. Whenever the customer updates the delivery time of any job, a new schedule is generated by considering the production constraints. The optimization is carried out by using the jobs waiting in ORR system and the non-started operations, and then the release decisions of the jobs waiting in ORR is made by considering the due dates of the jobs.

2.2.4. Order cancellation

Some orders may be cancelled by the customers. Whenever any order has been cancelled, the non-started operations list is updated and a new schedule is generated.

2.2.5. Rush order

Customers desire to obtain their order as soon as possible. Priority is given to the rush order; the ongoing schedule is renewed by considering this new situation.

2.3. Periodical rescheduling

The rescheduling occurs when a new dynamic event that changes the system status is encountered under event driven rescheduling strategy. The scheduling activity recurred at a fixed and predetermined time gathering all data from the shop floor under the periodic rescheduling strategy. No other important events can lead to rescheduling.

2.4. Objective functions

Lexicographic method (LM) [54] is a special case of the goal programming, in which the more important (upper level) goals are optimized before lower level goals are considered. In LM, the user is able to provide levels, or targets, of achievement for each objective and priorities the order in which goals are to be achieved. Many real-world optimization and decision-making problems are actually multi-objective. Lexicographic method (LM), deals with the achievement of objectives in a strict hierarchical order. This is true when some of our objectives are more important than others. Therefore, the first objective is taken into account primarily. The other objectives are naturally ranked according to their priorities: second-priority objectives, third priority objectives and so on. The performances of the solutions are measured using four metrics by making use of LM. The primary objective is minimizing mean tardiness. If the primary objective of the neighborhood solution is better than the primary objective of the current best solution, the best solution is updated. If the primary objective of the neighborhood solution and the primary objective of the current best solution are equal, then the secondary objective is evaluated. The secondary objective is minimizing schedule instability. When dynamic events occur, the non-started operations are scheduled with the proposed GRASP and this inevitably results in deviation from the previous schedule. The deviation from the previous schedule is a significant criterion for the performance of dynamic scheduling. For this reason, an instability criterion is added to the list of objectives to be optimized. The measure is the total deviation for all of the operations between the starting times in a new schedule and the old schedule. If the secondary objective of the neighborhood solution is better than the secondary objective of the current best solution, the current best solution is updated. The third objective is minimizing makespan and fourth objective is minimizing mean flow time. This evaluation procedure is adopted from Baykasoğlu [55], and the reader can refer to Baykasoğlu [55] for a detailed information about this procedure.

2.5. DFJSP with additional characteristics

In this section, the particular features of this research are explained. There are some assumptions in the model improvement: All jobs and machines are not available at the beginning of the scheduling horizon. Each operation can be operated on a set of machines and has its own due date. Transportation time is not considered. The due date and

Table 1
Sets and Parameters.

i, z→	job index
j, l→	operation index
k→	machine index
O_{ij}→	the working procedure (*operation*) *j* belonging to the job *i*
O_{ijk}→	the operation *j* belonging to job *i* to be executed on machine *k*
D_{ijk}→	sequence dependent setup time needed to process job *j* after job *i* on machine *k*
P_{ijk}→	the processing time of operation *j* belonging to job *i* on machine *k*
S_{ijk}→	the starting time of operation *j* belonging to job *i* on machine *k* (*Decision variable*)
F_{ij}→	the finishing time of operation *j* belonging to job *i* (*Decision variable*)
F_{ijk}→	the finishing time of operation *j* belonging to job *i* on machine *k* (*Decision variable*)
PS→	the partial schedule under construction
n→	total number of orders (*jobs*) in the schedule
JL→	jobs list
CL→	candidate operations list (*holds the operations of the jobs without predecessor in JL*)
FL→	list of the minimum finishing time information for the operations in CL

arrival time of jobs are unknown until they arrive. Sequence dependent setup time is considered. Before explaining the problem and the proposed GRASP with an illustrative example, the terms and decision variables are presented in Table 1.

2.6. Illustrative example

The proposed event-driven rescheduling policy is exemplified with an example as depicted in Fig. 1. The relevant data is shown in Table 2. The third column of Table 2 shows the machines on which the operations of the jobs can be processed and the processing times of the operations on the respective machines. For example, operation 1 of job 1 can be processed on machines 1, 2 or 3 with respect to the 2, 4 and 3 -h processing times, respectively. Each shift is 8 h, and each machine has machine specific unavailable time at the end of each shift. In this example, three shifts are illustrated, and jobs of type J2 and J1 are assumed to be processed on machines 1 and 2, respectively, just before the first shift. Thus, the setup times at the beginning of the first shift are taken as 1. Fig. 1(a) shows the original schedule with three machines and two orders. The following unexpected events happen: Machine 3 breaks down at time 5; Order 2 is called off at time 6; Order 3 gets at time 7 with due date 35; rush order (Order 4, which is identical with Job 1) arrives at time 8 and the delivery time of the third order is altered to 21.5 at time 8. Partial schedules are shown in Fig. 1(a). A feasible schedule is formed by the proposed event-driven rescheduling policy as it is demonstrated in Fig. 1(b).

3. GRASP for DFSSP: the central procedure

GRASP is an iterative algorithm that includes two main stages: construction and local search [10]. GRASP applies a greedy procedure in the course of construction phase of the algorithm so as to construct a feasible solution. Then, it applies local search methods for further improving the best solution detected so far.

3.1. Construction phase of GRASP

In the construction phase of the proposed GRASP, all available jobs (e.g., J_1, J_2, J_3, …, J_n) to be handled while constructing a new schedule are added to a list (called as the jobs list, JL) at each iteration. Then, all operations without predecessor of available jobs (e.g., O_{11}, O_{21}, O_{i1},…, O_{n1}) form the candidate operations list (CL). In this phase, G&T algorithm is incorporated into the GRASP in order to determine on which machines the operation in CL will be operated. Operations in CL are evaluated one by one in sequence from left to right. Trial schedules are generated for each next operation O_{ij} considering each alternative machine k on which that operation can be produced and considering the partial schedule under construction. The finishing time of the following operation is calculated for each alternative machine considering the starting time of the following operation on alternative machine, sequence dependent setup time, and processing time of the following operation for alternative machine (see Eq. (1)).

$$F_{ijk} = S_{ijk} + D_{ijk} + P_{ijk} \quad (1)$$

The operation is operated on machine k* which its finishing time is minimum, e.g, k* for O_{ij} = min (F_{ijk}, k = 1, 2,…, number of machine) (this assignment is not made in reality, it is a tentative assignment). Then, the minimum completion time information for operation *j*, F_{ijk^*}, is inserted in FL. This procedure is terminated when all of the operations in CL are evaluated in the same manner. The choice of the next operation that will be incorporated into the schedule is determined by evaluation of all operations in FL with respect to a greedy function. The utility of selecting operation in FL to be incorporated in the solution are evaluated by this function. Then, RCL is created with the alfa (α) parameter which adjust the covetousness and stochasticity of the algorithm and a threshold value g = ṣ + α(ṡ-ṣ), where ṣ = minimum value in FL and ṡ = maximum value in FL. This threshold value is used to build the RCL. One operation is chosen from RCL randomly at each iteration. This is the probabilistic part of the GRASP. After the selection of one of the operations in RCL, JL, CL and then FL are updated to collect the new information. These steps are iterated in the construction phase until all of the jobs in JL are scheduled. The construction phase of the proposed GRASP algorithm is outlined as follows:

Step 1: In order to construct the partial schedule (PS), initialize Gantt chart structure

Step 2: Collect jobs to be scheduled from the workshop and create JL using these jobs, and obtain relevant information about system state such as the due dates of jobs and their position from the shop floor before starting new scheduling

Step 3: Create candidate operations list CL by using the operations without predecessor belonging to jobs in JL

Step 4: Run the G&T algorithm in order to determine tentatively on which machines the operations in CL should be processed and determine the finishing time of each operation in CL on that tentatively assigned machine. Then, inset the finishing time information of each operation into an another list called FL (In this step, each operation in CL are tentatively tested one by one on all alternative machines where the operation would be processed, and one of these alternative machines is chosen for operation, resulting in a minimum completion time.).

Step 5: Acquire the finishing times of each operations in FL and determine the RCL so that;

ṣ = min {F_{ijk} (if they are all equal, choose one randomly)};

ṡ = max {F_{ijk} (if they are all equal, choose one randomly)};

g = ṣ + α(ṡ-ṣ);

RCL = {select all of the operations in FL whose finishing time are smaller or equals to g};

Step 6: Randomly choose one operation from RCL and put it to the partial schedule (PS) with the belonging attributes which are obtained in Step 4 (on which machine it will be produced, theirs starting time and finishing time information).

Step 7: Delete the chosen operation (chosen in step 6) of the related job in JL. If all of the jobs in JL are scheduled, go to Step 8. Else, go to Step 2.

Step 8: Calculate the LM based objective function of the generated PS.

A simple example for 2*2 flexible scheduling problem with the data provided in Table 3 is presented in order to better explain the working mechanism of the proposed GRASP algorithm. In this example, setup times are ignored while assigning the first job.

Initialization

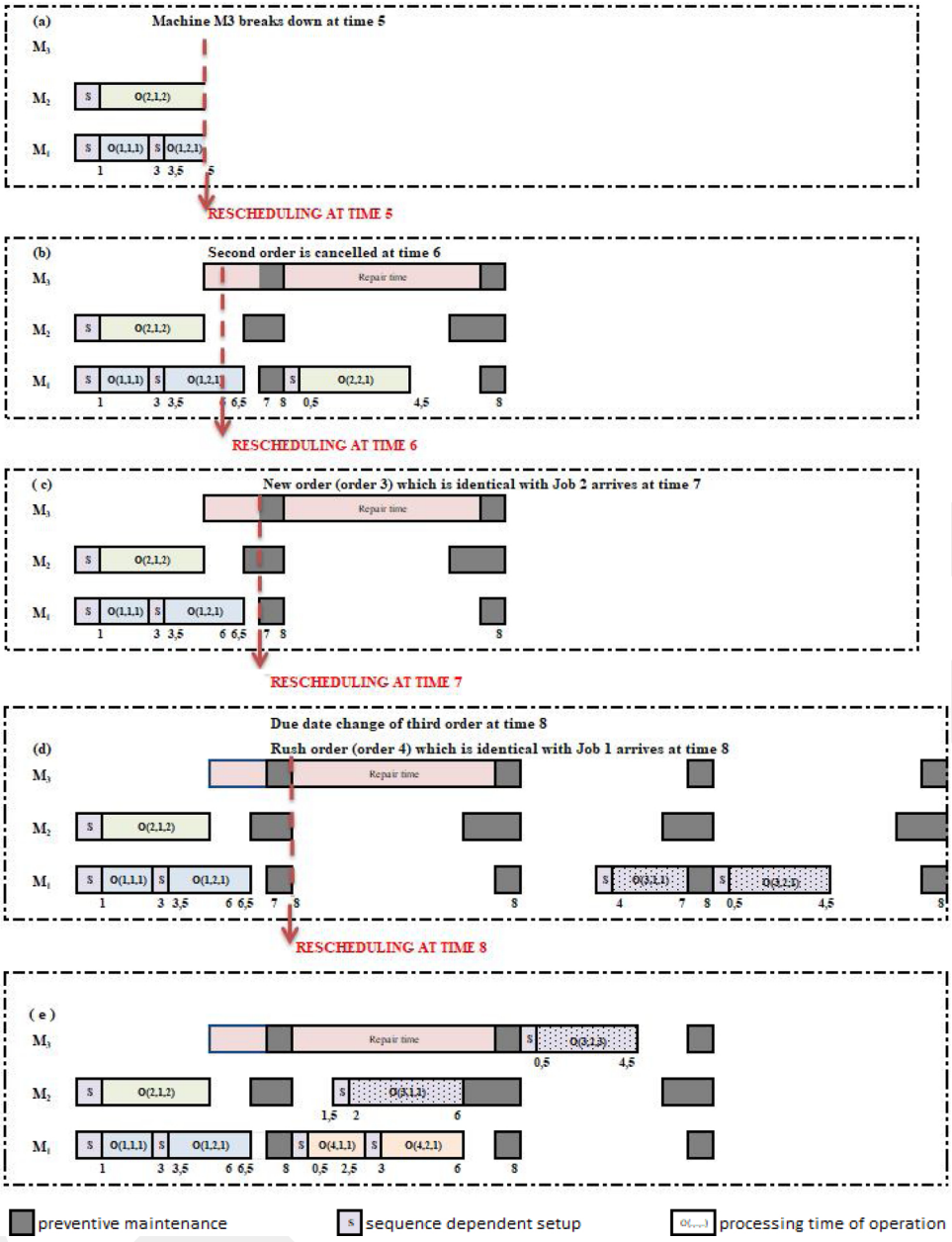

Fig. 1. (a) Partial schedule at time5 (b) Partial schedule at time6 (c) Partial schedule at time7 (d) Partial schedule at time8 (e) A final schedule for this example.

Table 2
Example problem data.

Jobs	Operation	Machine no/Processing Time(hour)
Job1	Operation1	1/2; 2/4; 3/3
	Operation2	1/3; 2/3; 3/3
Job2	Operation1	1/3; 2/4; 3/3
	Operation2	1/4; 2/5; 3/5

Setup Times

Machine1			Machine2		
	Job1	Job2		Job1	Job2
Job1	0.5	1	Job1	0.5	1
Job2	1	0.5	Job2	1	0.5

Table 3
Introductory example data.

Jobs	Operation	Machine no/Processing Time(hour)
J1	Operation1	1/10; 2/8
	Operation2	1/4; 2/7
J2	Operation1	1/7; 2/9
	Operation2	1/7; 2/5
J3	Operation1	1/6; 2/8
	Operation2	1/9; 2/6

Setup Times

Machine1				Machine2		
	J1	J2	J3	J1	J2	J3
J1	0	2	2	0	3	1
J2	3	1	1	2	0	2
J3	1	3	0	1	2	0

Step 1: PS = Ø;
Step 2: JL = {J1, J2}
Step 3: CL = {O_{11}, O_{21}}
Step 4: S_{111} = 0, S_{112} = 0, F_{111} = 10 (S_{111} + P_{111}), F_{112} = 8 (S_{112} + P_{112}), f* = min {F_{111}, F_{112}} = 8, k* = 2, FL = {F_{112}}; S_{211} = 0, S_{212} = 0, F_{211} = 7, F_{212} = 9, f* = min {F_{211}, F_{212}} = 7, k* = 1, FL = {F_{112}, F_{211}}
Step 5: ṣ = 7, ṡ = 8, α = 0.4, g = 7.4, FL = {F_{112} = 8, F_{211} = 7}, RCL = {O_{211}}
Step 6: PS = {O_{211}}
Step 7: Delete O_{21} from J2; go to Step 2.
Iteration 2
Step 2: JL = {J1, J2}
Step 3: CL = {O_{11}, O_{22}}
Step 4: S_{111} = 10 (7 (F_{211}) + 3 (D_{211})), S_{112} = 0, F_{111} = 20, F_{112} = 8, f* = min {F_{111}, F_{112}} = 8, k* = 2, FL = {F_{112}}; S_{221} = 8 (7 (F_{211}) + 1 (D_{221})), S_{222} = 7 (F_{211}), F_{221} = 15, F_{222} = 12, f* = min {F_{221}, F_{222}} = 12, k* = 2, FL = {F_{112}, F_{222}}
Step 5: ṣ = 8, ṡ = 12, α = 0.4, g = 9.6, FL = {F_{112} = 8, F_{222} = 12}, RCL = {O_{112}}
Step 6: PS = {O_{211}, O_{112}}
Step 7: Delete O_{11} from J1; go to Step 2.
Iteration 3
Step 2: JL = {J1, J2}
Step 3: CL = {O_{12}, O_{22}}
Step 4: S_{121} = 11 (8 (F_{112}) + 3 (D_{211})), S_{122} = 8 (8 (F_{112}) + 0 (D_{112})), F_{121} = 15, F_{122} = 15, f* = min {F_{121}, F_{122}} = 15, select one machine randomly k* = 1, FL = {F_{121}}; S_{221} = 8 (7 (F_{211}) + 1 (D_{221})), S_{222} = 11 (8 (F_{112}) + 3 (D_{122})), F_{221} = 15, F_{222} = 16, f* = min {F_{221}, F_{222}} = 15, k* = 1, FL = {F_{121}, F_{221}}
Step 5: ṣ = 15, ṡ = 15, α = 0.4, g = 15, FL = {F_{121} = 15, F_{221} = 15}, RCL = {O_{121}, O_{221}}, select one operation randomly (O_{121} is selected)
Step 6: PS = {O_{211}, O_{112}, O_{121}}
Step 7: Delete O_{12} from J1; go to Step 2.
Iteration 4
Step 2: JL = {J2}
Step 3: CL = {O_{22}}
Step 4: S_{221} = 17 (15 (F_{121}) + 2 (D_{121})), S_{222} = 11 (8 (F_{112}) + 3 (D_{122})), F_{221} = 24, F_{222} = 16, f* = min {F_{221}, F_{222}} = 16, k* = 2, FL = {F_{222}}
Step 5: ṣ = 16, ṡ = 16, α = 0.4, g = 16, FL = {F_{222} = 16}, RCL = {O_{222}}
Step 6: PS = {O_{211}, O_{112}, O_{121}, O_{222}}
Step 7: Delete O_{22} from J2. All of the jobs are now scheduled, then go to Step 8.

Step 8: In this step, the objective function value of the obtained PS is calculated by using LM. Then, the construction phase continues with the local search phase. Details of the calculation of the objective function are explained with the help of the example given below.

When an event that changes the current system status occurs, the current schedule becomes old-fashioned and a new schedule is required. After generating a new schedule, the run of the system is continued by assigning related attributes to the jobs according to the best solution vector that is found. Newly selected operations are incorporated into the partial schedule at each iteration during the construction phase of the new PS, until all operations are assigned. This constructive attribute of the algorithm makes it possible to adjust the sequence size when unexpected events (rush order, machine breakdown etc.) occur. Therefore, this algorithm is very effective in adapting to dynamic events. An example is given next in order to explain how the proposed mechanism is executed under a dynamic event: Assume that a new job (Job 3) is arrived to the production area at time 7, and the due dates of J1, J2 and J3 are 19, 15 and 22, respectively. The execution procedure of the proposed construction phase is presented below. The Gantt-chart is displayed in Fig. 2.

Initialization
Step 1: JL = {J1, J2, J3}
Step 2: CL = {O_{12}, O_{22}, O_{31}}
Step 3: S_{121} = 11, S_{122} = 8, F_{121} = 15, F_{122} = 15, f* = min {F_{121}, F_{122}} = 15, select one machine randomly k* = 2, FL = {F_{122}}; S_{221} = 8, S_{222} = 11, F_{221} = 15, F_{222} = 16, f* = min {F_{221}, F_{222}} = 15, k* = 1, FL = {F_{122}, F_{221}} ; S_{311} = 8 (7 (F_{211}) + 1 (D_{231})), S_{312} = 9 (8 (F_{112}) + 1 (D_{132})), F_{311} = 14, F_{312} = 17, f* = min {F_{311}, F_{312}} = 14, k* = 1, FL = {F_{122}, F_{221}, F_{311}}
Step 4: ṣ = 14, ṡ = 15, α = 0.4, g = 14.4, FL = {F_{122} = 15, F_{221} = 15, F_{311} = 14}, RCL = {O_{311}},
Step 6: PS = {O_{311}}
Step 7: Delete O_{31} from J3; go to Step 1.
Iteration 2
Step 2: JL = {J1, J2, J3}
Step 3: CL = {O_{12}, O_{22}, O_{32}}
Step 4: S_{121} = 14 + 1, S_{122} = 8 + 0, F_{121} = 19, F_{122} = 15, f* = min {F_{121}, F_{122}} = 15, k* = 2, FL = {F_{122}}; S_{221} = 17, S_{222} = 11, F_{221} = 24, F_{222} = 16, f* = min {F_{221}, F_{222}} = 16, k* = 2, FL = {F_{122}, F_{222}}; S_{321} = 14, S_{322} = 15, F_{321} = 23, F_{322} = 21, f* = min {F_{321}, F_{322}} = 21, k* = 2, FL = {F_{122}, F_{222}, F_{322}}
Step 5: ṣ = 15, ṡ = 21, α = 0.4, g = 17.4, FL = {F_{122} = 15, F_{222} = 16, F_{322} = 21}, RCL = {O_{122}, O_{222}}, select one operation randomly (O_{222} is selected)
Step 6: PS = {O_{311}, O_{222}}
Step 7: Delete O_{22} from J2; go to Step 2.
Iteration 3
Step 2: JL = {J1, J3}
Step 3: CL = {O_{12}, O_{32}}
Step 4: S_{121} = 15, S_{122} = 18, F_{121} = 19, F_{122} = 25, f* = min {F_{121}, F_{122}} = 19, k* = 1, FL = {F_{121}}; S_{321} = 14, S_{322} = 18, F_{321} = 23, F_{322} = 24, f* = min {F_{321}, F_{322}} = 23, k* = 1, FL = {F_{121}, F_{321}}
Step 5: ṣ = 19, ṡ = 23, α = 0.4, g = 20.6, FL = {F_{121} = 19, F_{321} = 23}, RCL = {O_{121}}
Step 6: PS = {O_{311}, O_{222}, O_{121}}
Step 7: Delete O_{12} from J1, go to Step 2.
Iteration 4
Step 2: JL = {J3}
Step 3: CL = {O_{32}}
Step 4: S_{321} = 21, S_{322} = 18, F_{321} = 30, F_{322} = 24, f* = min {F_{321}, F_{322}} = 24, k* = 2, FL = {F_{322}}
Step 5: ṣ = 24, ṡ = 24, α = 0.4, g = 24, FL = {F_{322} = 24}, RCL = {O_{322}}
Step 6: PS = {O_{311}, O_{222}, O_{121}, O_{322}}
Step 7: Delete O_{32} from J3. All of the jobs are now scheduled, then stop.

The constructed schedule (Fig. 2(a)) is conducted until time 7. The ongoing operations are not interrupted, but the rescheduling process is triggered in order to construct a new schedule as a new order arrived at time 7 (Fig. 2 (b)). Whenever a rescheduling process begins, all non-started operations are rescheduled according to the proposed algorithm as explained above (see also Fig. 3). The tardiness (max{0, completion time of the job *i*- due date of the job *i*}) of J1, J2 and J3 are calculated as 0, 1 and 2, respectively. So, the mean tardiness is 1 for these 3 jobs. The operations common to both the old and new schedules are O_{12} and O_{22}. The completion times of these operations in the old schedule are 15, 16, respectively. The completion times of them in the new schedule are 19 and 16, respectively. So, the value of instability criterion ($\sum_{\text{all elibible } O_{ij}} |F_{ij}^{\text{new schedule}} - F_{ij}^{\text{old schedule}}|$) is calculated as 4 (|19-15| + |16-16|). The makespan of this new schedule is 24-time unit. Since at least one operator of all three jobs is included to the new schedule, the mean flow time is calculated based on the starting times of the first operations of these jobs. The starting times of J1, J2 and J3 are 0, 0 and 8, and their completing times are 19, 16 and 24, respectively. So, the mean flow time is calculated as 17-time unit (((19-0) + (16-0) + (24-8)) / 3).

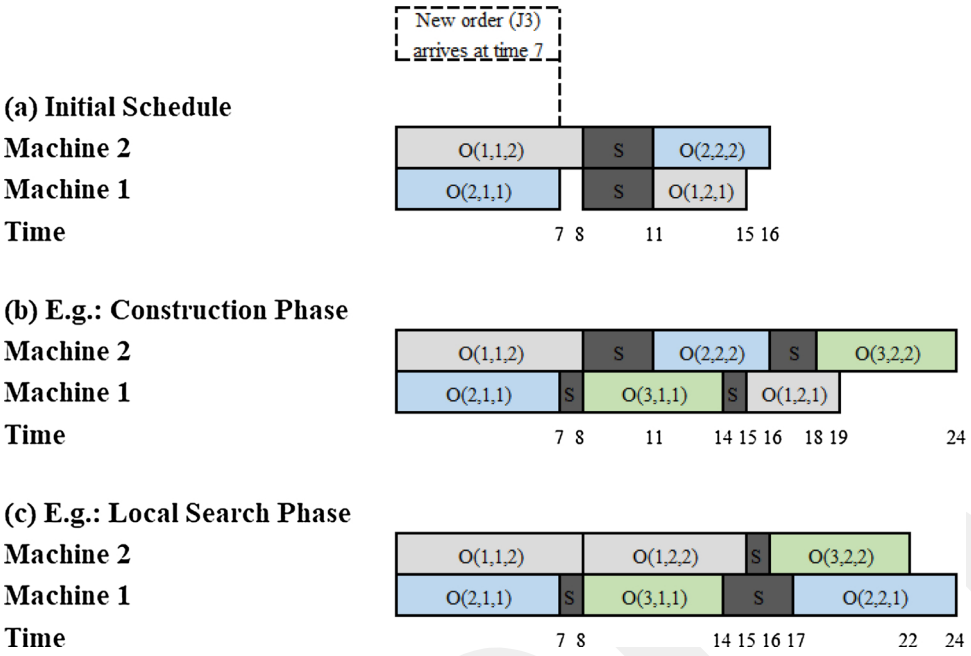

Fig. 2. An event-driven rescheduling example for FJSP with new order arrival.

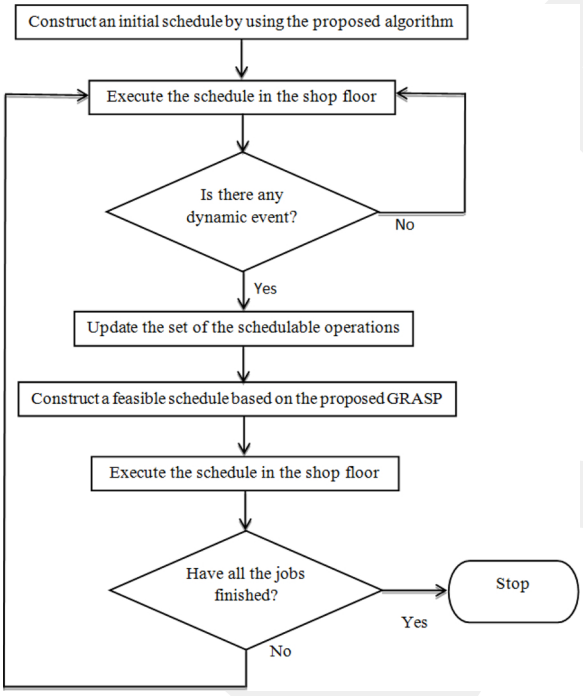

Fig. 3. A simplified flow diagram of the proposed approach for DFJSP.

Four performance criteria for this example can be denoted as {1, 4, 24, 17} according to the lexicographic method that is used in this paper. This solution is accepted as the current best solution and the algorithm continues its running with the local search stage.

3.2. Local search

The feasible solution generated during the construction stage of the algorithm is aimed to be further improved by examining its neighborhood solutions in the local search stage. The local search begins by converting the solution (partial schedule, PS) generated from the construction step to a permutation-based solution vector (PBSV) in order to obtain the feasible solutions throughout the algorithm, repeats its operation for a pre-determined number of iterations, *local_iteration*, and uses four different moves inside it. These moves are: (1) swapping contents of the two randomly selected positions in PBSV, (2) insertion (content of a randomly selected position of PBSV is removed and inserted between the randomly selected two adjacent positions in PBSV), (3) swapping the assigned machines between two randomly selected positions in PBSV, (4) changing the assigned machine of a randomly selected position in PBSV. In each iteration of the local search stage, one of these four moves are randomly determined and applied to the current PBSV for getting a new PBSV. After obtaining a new PBSV, a decoding method is performed on the new PBSV solution in order to convert it into a new PS for calculating its objective function value. If the solution quality of the obtained new PS is better than the previous one (current PBSV solution), the new PBSV solution is accepted as the current solution. Otherwise, the local search continues its search with the previous solution at hand. The further details of the proposed local search phase are given in the following.

3.2.1. Converting PS to PBSV (encoding)

Firstly, a fixed base vector (FBV) is created through PS obtained from the construction stage of the algorithm by considering the jobs and how many operations they have, and the same FBV is used for all encoding and decoding operations performed during the local search phase. The method of Amirthagadeswaran and Arunachalam [56] is used for generating FBV in which each number represents one operation of a job and each job will appear up to the operations it has. PS is generated as $\{O_{311}, O_{222}, O_{121}, O_{322}\}$ in the example given above after arriving a new job (J3) to the production area at time 7. In PS, J1, J2 and J3 have 1, 1 and 2 unscheduled operations, respectively. FBV is created as [1 2 3 3] beginning from the first job to the last one including the repetitions of the operators, where {1, 2, 3} represents $\{job_1, job_2, job_3\}$, respectively. For convenience, PS can be denoted as [3(1) 2(2) 1(1) 3(2)] via the job (machine) representation. The first job, 3, in PS,

represents the third position in FBV in which the value of 3 appears. Likewise, the second job, 2, in PS, represents the second position in FBV in which the value of 2 appears. So, the first and second positions in the permutation-based solution vector (PBSV) will be 3 and 2, respectively. When all steps are completed, PS ([3(1) 2(2) 1(1) 3(2)]) will be transformed into PBSV as [3(1) 2(2) 1(1) 4(2)] by preventing the repetition of the used positions in FBV.

3.2.2. Decoding PBSV to PS

After the obtainment of a new PBSV solution through one of the four moves, a decoding method is applied to the new PBSV solution in order to convert it into a new PS for calculating its objective value. Let us explain this situation with an example. Suppose that the move 1 is applied to PBSV of the above example and a new PBSV is obtained as [3(1) 1(2) 2(1) 4(2)] (the job content of the second and third positions of PBSV are changed). The first number, 3, in new PBSV, represents the third position in FBV (the same FBV is used throughout the local algorithm) in which the value of 3 appears. The second number, 1, in PBSV, represents the first position in FBV in which the value of 1 appears. Likewise, third and fourth numbers, 2 and 4, represent the second and fourth positions in FBV in which the values of 2 and 3 appear, respectively. PBSV, [3(1) 1(2) 2(1) 3(2)], will be transformed into PS as [3(1) 1(2) 2(1) 3(2)] and PS will be $\{O_{311}, O_{122}, O_{221}, O_{322}\}$. This encoding and decoding methods guarantee the feasibility throughout the local search algorithm. The first operation to be inserted to the new schedule is O_{31} and this operation is processed on the machine 1. The latest operation scheduled on this machine is O_{21} and F_{211} is 7. Since D_{231} is 1-time unit, S_{311} is 8. Processing time of O_{311} is 6-time unit and it will be completed at 14. The second operation in PS is O_{12} and it is scheduled to the machine 2. The operation has to be done just before O_{12} is O_{11}, and O_{11} is completed at 8. The latest operation scheduled on this machine is also O_{11}. Since D_{112} is 0-time unit, S_{122} is 8. Processing time of O_{122} is 7-time unit and it will be completed at 15. The third operation in PS is O_{22} and it is processed on machine 1. The operation has to be done just before O_{22} is O_{21}, and O_{21} is completed at 7. The latest operation scheduled on this machine is also O_{31} and it is completed at 14. The largest one between 7 and 14 is 14. So, the setup time of O_{22} will begin at 14. Since D_{321} is 3-time unit, S_{221} is 17. Processing time of O_{221} is 7-time unit and it will be completed at 24. The last operation in PS is O_{32} and it will be processed on the machine 2. The operation has to be done just before O_{32} is O_{31}, and O_{31} is completed at 14 on machine 1. The latest operation scheduled on the machine 2 is O_{12} and it is completed at 15. The largest one between 14 and 15 is 15. So, the setup time of O_{32} will begin at 15. Since D_{132} is 1-time unit, S_{122} is 16. Processing time of O_{322} is 6-time unit and it will be completed at 22 (see Fig. 2(c)). The tardiness of J1, J2 and J3 are 0, 9 and 0, respectively. So, the mean tardiness of this candidate solution is 3. The operations common to both the old and new schedules are O_{12} and O_{22}. The completion times of these operations in the old executed schedule are 15, 16, respectively. The completion times of these operations in the candidate schedule are 15 and 24, respectively. So, the value of instability criterion is calculated as 8(|15-15| + |24-16|). The makespan of this schedule is 24-time unit. The starting times of J1, J2 and J3 are 0, 0 and 8, respectively, and their completing times are 15, 24 and 22, respectively. So, the mean flow time is calculated as 43.67-time unit (((15-0) + (24-0) + (22-8))/3). Four performance metrics for this candidate solution is {3, 8, 24, 43.67}. Primary objective of the candidate solution is worse than the primary objective of the current best solution. So, the best solution will not be changed. After all the steps of the algorithm are completed, the newly constructed schedule is implemented until another unexpected event occurs. Calculation of the objective function value is explained in Algorithm 1.

for *op* = 1 : |PS| //(all operations in PS and at most number of orders (jobs), *n* × number of operations, *r* can be scheduled)
  if any previous operation (O_{ij^*}) of the operation in PS[*op*] (O_{ij}) is not scheduled
    → Determine the latest operation (O_{zl}) assigned to the same machine k with PS[*op*] (if there is)
    → Calculate the starting, (S_{ijk}) and finishing time (F_{ijk}) of PS[*op*] by considering setup time (D_{zik}), latest operation (O_{zl}) previously scheduled to the same machine (if there is), and the other related information
    → Add the selected operation to the tentative schedule
  else
    → Determine the finishing time (F_{zlk}) of the latest operation (O_{zl}) assigned to the same machine k with PS[*op*] (if there is)
    → Determine the finishing time (F_{ij^*k}) of the previous operation (O_{ij^*}) of PS[*op*] (O_{ij})
    → Calculate the starting (S_{ijk}) and finishing time (F_{ijk}) of PS[*op*] by considering the biggest finishing time between F_{zlk} and F_{ij^*k}, setup time (D_{zik}) and the other related information
    → Add the selected operation to the tentative schedule
  endif
endfor
→ Calculate the LM based objective function value for the generated PS

The steps of the local search phase of GRASP are outlined as follows:

Step 1: Construct a fixed base vector (FBV) by using the partial schedule (PS) obtained from the construction stage of the algorithm

Step 2: Covert PS into a permutation-based solution vector (PBSV) through FBV

Step 3: Set the number of iterations (*local_iteration*).

Step 4: Select a move from one of 4 moves randomly.

Step 5: Construct a new PBSV by applying the selected move.

Step 6: Decode the new PBSV into a new PS by using FBV

Step 7: Calculate objective function value of new PS.

Step 8: If the objective function of the candidate solution provides an improvement over the best solution available, replace them and update the best schedule.

Step 9: Repeat all of the steps until *i* reaches the local search iteration number.

The pseudo-code of the proposed algorithm is outlined in Algorithm 2.

→ Start the algorithm parameters, the Gantt charts related to the assignments of operations to machines and machine-operation timing charts, and make all problem data available (processing times, setup times, and all other data).
for *event_iter* = 1 : *max_event_iteration* (number of generated schedules during the executing of the algorithm)
  for *g_iter* = 1 : *grasp_iteration* //the given number of iterations for the GRASP algorithm
    → Collect the information about the latest status of the system
    → **Run** the construction phase of the proposed GRASP based algorithm
    → **Run** Local search phase of the proposed GRASP based algorithm
  endfor
  → Release the new schedule to the shop floor
endfor

Each new schedule can contain at most *n* orders (jobs) and each job can contain *r* operations in the worst-case scenario. The machine selection and sequencing of the operations of the orders are done at the same time in the construction phase and *r*n* times at most. There are at most *m* machines in the candidate machine list and the elements of the RCL is updated after each assigned operation *n* times at most. Thus, the overall complexity of the construction phase is $O(rmn^2)$ for one algorithm iteration (*grasp_iteration*). In the local search phase, the most time consuming process having $O(local\_iteration * r^2n^2)$ is the move of insertion. The total number of schedules during the execution of the algorithm will be as much as the events that cause to the generating of a new schedule (*max_event_iteration*). The overall worst case computational complexity of the algorithm is $O(max\_event\_iteration * grasp\_iteration * [(local\_iteration * r^2n^2) + (rmn^2)])$.

4. Experimental study

This section consists of four stages: a stage for the parameter calibration of the algorithm and three stages for the computational study to assess the performance of the proposed GRASP algorithm. In the first stage, the effective levels of each required parameter for the proposed algorithm runs are identified. In the second stage, the experiments are carried out on FJSP by using standard benchmark problems in order to reveal the performance of the proposed GRASP algorithm compared to other algorithms in the literature. Having observed that the algorithm is a powerful and competitive algorithm in the previous section, the third stage of the experiments are carried out on DFJSP and compare the proposed GRASP algorithm based event-driven rescheduling mechanism to the most common dispatching rules; EDD (earliest due date), SPT (shortest processing time), FIFO (first arrival at shop), SL (smallest slack), LPT (longest processing time), CR (smallest critical ratio). The fourth and final stage of computational study offers a comparison of the proposed GRASP algorithm between the event-driven and periodic rescheduling mechanism. It can be noted that few studies in the literature simultaneously take into account more than one real-time situation such as order arrivals, machine breakdowns, order cancellations, due date changes, urgent order arrivals and consider machine capacity constraints and sequence dependent setup times. Since there is no benchmark instances in the literature that contains all of the above-mentioned features for a direct comparison, experiments in the last two stages are carried out on three randomly generated test instances. Test instances have varying size, which are small (5 jobs x 5 machines), medium (10 jobs x 10 machines) and large (15 jobs x 15 machines), and given in Appendix A (see Table A1, Table A2 and Table A3). Extensive computational experiments are performed under the several degrees of problem flexibility, due date tightness and shop utilization. Machine sets are formed randomly and machine alternatives of operations (flexibility levels) for different problem sizes are given in Table 4. Flexibility levels of the machines are represented as follow: Small flexibility level: 1, Medium flexibility level: 2, Large flexibility level: 3. For example; 5 × 5 problem with flexibility 1 (small) means that each operation is conducted by a candidate machine set consisting of two machines. Non-flexible DJSSP is demonstrated with 0. The effects of resource utilization and due date tightness on the test problems are investigated at two levels. Low (-) machine unused time level, ranges [0.05 0.15] and High (+) machine unused time level ranges [0.20 0.50]. Tight (T) due date and loose (L) due date are examined.

4.1. Parameter calibration

Appropriate design of the parameters has significant impact on the efficiency of the proposed GRASP algorithm. This stage of the experimental study evaluates the demeanor of the proposed GRASP algorithm with different parameters. The proposed GRASP algorithm can be defined by the control parameter set $\pi = \{GI, LI, \alpha\}$ where; GI is the global iteration number, LI is the local iteration number and α is the alfa parameter. We employed a statistical design of experiments (DOE) approach [57] in order to find out the best possible parameter set. The DOE is an investigative method regarded highly both for its effectiveness and efficiency in the evaluation of the effect of multiple factors upon a process.

Table 4
Machine alternatives of operations for different problem sizes.

Problem	Machine flexibility		
	Small	Medium	Large
5 × 5	2	3	4
10 × 10	2	5	8
15 × 15	3	5	12

4.1.1. Parameter calibration for the static FJSP

A FJSP from Brandimarte's data-set [11] with medium complexity is chosen as 20 jobs-15 machines (Mk_10) to identify the effect of the control parameters. Each control parameter is determined through preliminary experiments and varies at four levels. The global iteration number (GI) has 5 possible values: 100, 200, 300, or 400, the local iteration number (LI) can be 100, 150, 200, or 250, and the alfa parameter (α) can be 0.3, 0.4, 0.5, or 0.6. Therefore, in order to permit the detection of a possible interaction of factor effects, a 4^3 full factorial experimental layout is used for carrying out the experiments. For each combination, 10 independent runs are carried out at each design point leading to a total of (64 × 10) 640 runs. In order to determine which control parameter effects are significant, a statistical analysis of variance (ANOVA) is conducted. As it can be seen from the results of ANOVA in Table 5, each of the main effects are found significant at the level of 5 %. Fig. 4 shows the plot of main effects for mean values obtained at each level. As a result, it is possible to recommend the values of GI = 300, LI = 200 and α = 0.4 as the efficient control parameters.

4.1.2. Parameter calibration for the dynamic FJSP

Similar to the parameter setting for the FJSP, the medium benchmark instance selected for parameter calibration of the DFJSP. The experiments are carried out in an environment where the machine flexibility level is taken as 0, the machine unused time level is accepted as low (-), and the due date tightness is accepted as tight (T). Likewise, 10 independent runs are carried out at each design point leading to a

Table 5
ANOVA for the FJSP.

Source of variation	Degree of freedom	Sum of squares	Mean of squares	F_{calc}	Prob($F > F_{calc}$)
Model	63	27128.6	430.61	8.14	0.00
Linear	9	24462.4	2718.04	51.37	0.00
GI	3	22677.4	7559.15	142.88	0.00
LI	3	1364.8	454.95	8.60	0.00
α	3	420.1	140.03	2.65	0.04
2-Way Interactions	27	1051.5	38.94	0.74	0.83
GI*LI	9	208.1	23.12	0.44	0.92
GI*α	9	399.9	44.43	0.84	0.58
LI*α	9	443.5	49.28	0.93	0.40
3-Way Interactions	27	1614.7	59.81	1.13	0.30
GI*LI*α	27	1614.7	59.81	1.13	0.30
Error	576	30474.3	52.91		
Total	639	57602.9			

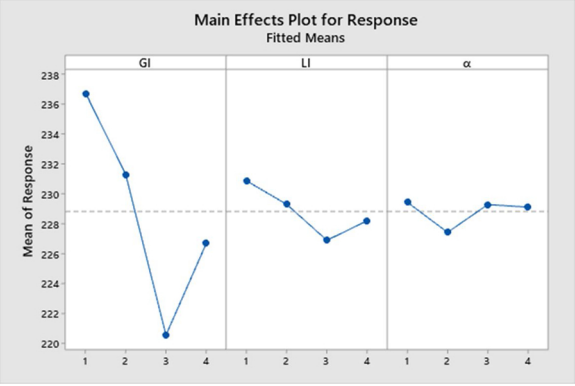

Fig. 4. Main effects plot (data means) for the FJSP.

Table 6
ANOVA for the DFJSP.

Source of variation	Degree of freedom	Sum of squares	Mean of squares	F_{calc}	Prob($F > F_{calc}$)
Model	63	45446.0	721.4	8.44	0.00
Linear	9	40256.5	4472.9	52.30	0.00
GI	3	35601.1	11867.0	138.76	0.00
LI	3	3880.8	1293.6	15.13	0.00
α	3	774.5	258.2	3.02	0.03
2-Way Interactions	27	2578.4	95.5	1.12	0.31
GI*LI	9	1188.6	132.1	1.54	0.13
GI*α	9	638.1	70.9	0.83	0.60
LI*α	9	751.8	83.5	0.98	0.46
3-Way Interactions	27	2611.1	96.7	1.13	0.29
GI*LI*α	27	2611.1	96.7	1.13	0.29
Error	576	49259.3	85.5		
Total	639	94705.3			

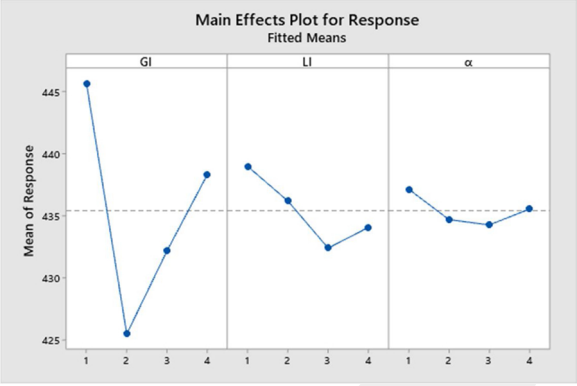

Fig. 5. Main effects plot (data means) for the DFJSP.

total of 640 runs and ANOVA is done as in Table 6. It is clearly seen from the plot of main effects for mean values obtained at each level in Fig. 5, the efficient control parameters for the DFJSP are set to 200 for GI, 200 for LI and 0.5 for α.

4.2. Experiment I

In this stage, we have conducted the experiments on three static FJSP benchmarks that are Kacem data [58], BRdata [11] and DPdata [59] in order to have an idea about the general attitude of the proposed GRASP. The FJSP benchmark sets include 31 problem instances and machine x job numbers of the problem instances are varying from 60 to 200. The proposed GRASP algorithm is run 10 times and the best obtained results are reported.

Computational results for the GRASP are displayed in Tables 7–9. The best values among the referred approaches depicted with bold values. Table 7 gives the result of the GRASP along with PSO + SA of [60], PVNS of [61], hGA of [17] and PSO of [62] for Kacem's instances.

The size of the problem is given in the first column, in which *n* and *m* represent the number of jobs and the number of machines, respectively. The second column represents the best makespan value obtained from PSO + SA, PVNS, hGA, PSO and proposed GRASP, respectively. Average CPU times of the proposed GRASP algorithm are provided in the last column. The proposed GRASP algorithm outperforms PSO + SA in 2 out of 3 instances, outperforms PVNS in 1 out of 3 instances, and outperforms PSO in 2 out of 3 instances. The GRASP algorithm performs similar to hGA for this data set.

Table 8 compares the best makespan of the GRASP with PVNS of [61], CDDS of [63], PSO of [62], SSPR of [64] for Brandimarte data set. The name and size of the instances are given in the first and second column of Tables 8 and 9. The third column shows the 'flexibility' that is defined in terms of the average number of alternative machines for each operation and the last column shows average CPU time of the proposes GRASP algorithm. The best lower bound of the instances are given in the fourth column of Tables 8 and 9. The results indicate that the proposed GRASP algorithm can obtain the best results for 7 out of 10 instances and shows a better performance than PVNS, CDDS and PSO.

Table 9 displays the results of the GRASP with hGA of [17], eGA of [14], CDDS of [63], SSPR of [64] and QPSO of [25] for Dauzere-Peres and Paulli's data set. The proposed GRASP algorithm is able to provide the best results for 4 instances out of 18 instances.

The proposed GRASP gives a performance close to the other algorithms for Dauzere-Peres and Paulli's instances. Statistical tests are conducted to better identify the distinction between these algorithms. The parametric test conditions that are the independence, normality and heteroscedasticity properties are not fulfilled perfectly. Therefore, the non-parametric statistical tests are conducted in order to analyze the results.

Derrac et al. [65] suggested usage of Friedman test that is a non-parametric equivalent of the parametric two-way analysis of variance for multiple comparisons (*N* x *N*). Test results depict that SSPR is the best algorithm with a rank of 2.11 and hGA is the worst algorithm with a rank of 5.14. Then, Iman-Davenport extension with $F_F = 10.24036$ and p-value = 0.000 is applied to detect if there are some difference among the algorithms. After detecting the differences among the algorithms, Nemenyi and Holm's post-hoc procedures are selected for the implementation of the Friedman test for being able to apply multiple comparisons process. The test statistic value of z for comparing the *i*th and *j*th algorithms is taken from the table of normal distribution. Table 10 lists all z-values, unadjusted values, and all of the post-hoc hypotheses. By applying Nemenyi's procedure with a level $\alpha = 0.1$, only six hypotheses are rejected, and the improvement of SSPR and QPSO over hGA and CDDS, and GRASP and eGA over hGA are identified. In addition, Holm procedure rejects improvement of GRASP over CCDS. The remaining eight hypotheses cannot be rejected by using these methods. However, when a level is changed as $\alpha = 0.05$, six hypotheses are rejected by Holm's method. Table 11 summarizes the rejected hypotheses for different α level. In the light of these tests, it can be concluded that the proposed GRASP algorithm is as effective and competitive as other algorithms in solving static FJSPs. This can be an indication of its applicability to dynamic scheduling problems.

4.3. Experiment II

As mentioned earlier, there is no any algorithm in the literature that we can compare directly with our algorithm due to the complexity of the problem studied here. More than that, the objective function used in this study is completely different from the objective functions used in

Table 7
The results for Kacem's instances.

Problem size	PSO + SA [60]	PVNS [61]	hGA [17]	PSO [62]	**Proposed GRASP**	Average CPU time of the proposed GRASP
8 × 8	15	**14**	**14**	17	**14**	15.01
10 × 10	**7**	**7**	**7**	8	**7**	36.11
15 × 10	12	12	**11**	**11**	**11**	62.54

Table 8
Results for Brandimarte's data-set.

Problems	*n* x *m*	flex.	Lower Bound	PVNS [61]	CDDS [63]	PSO [62]	SSPR [64]	**Proposed GRASP**	Average CPU time of the proposed GRASP
Mk_01	10 × 6	2.09	36	**40**	**40**	41	**40**	**40**	10.74
Mk_02	10 × 6	4.10	24	**26**	**26**	**26**	**26**	**26**	12.15
Mk_03	15 × 8	3.01	204	**204**	**204**	207	**204**	**204**	22.82
Mk_04	15 × 8	1.91	48)	**60**	**60**	65	**60**	**60**	17.32
Mk_05	15 × 4	1.71	168	173	173	**171**	172	172	55.15
Mk_06	10 × 15	3.27	33	60	58	61	**57**	64	35.48
Mk_07	20 × 5	2.83	133	141	**139**	173	**139**	**139**	76.10
Mk_08	20 × 10	1.43	523	**523**	**523**	**523**	**523**	**523**	81.01
Mk_09	20 × 10	2.53	299	309	**307**	**307**	**307**	**307**	49.32
Mk_10	20 × 15	2.98	165	206	197	312	**196**	205	100.45

Table 9
Results for Dauzere-Peres and Paulli's data set.

Problems	*n* x *m*	flexibility	LB	hGA [17]	eGA [14]	CDDS [63]	SSPR [64]	QPSO [25]	**ProposedGRASP**	Average CPU time of the proposed GRASP
1a	10 × 5	1.13	2505	2518	2516	2518	**2505**	**2505**	**2505**	76.41
2a	10 × 5	1.69	2228	2231	2231	2231	**2229**	2230	2230	125.15
3a	10 × 5	2.56	2228	2229	2232	2229	**2228**	2229	2229	112.24
4a	10 × 5	1.13	2503	2515	2515	2503	2503	**2498**	2503	65.54
5a	10 × 5	1.69	2189	2217	2208	2216	2211	**2207**	**2207**	112.34
6a	10 × 5	2.56	2162	2196	2174	2196	2183	**2170**	**2170**	195.28
7a	15 × 8	1.24	2187	2307	**2217**	2283	2274	2264	2264	120.46
8a	15 × 8	2.42	2061	2073	2073	2069	**2064**	2073	2073	163.18
9a	15 × 8	4.03	2061	2066	2066	2066	**2062**	2066	2066	253.35
10a	15 × 8	1.24	2178	2315	**2189**	2291	2269	2205	2205	110.48
11a	15 × 8	2.42	2017	2071	2063	2063	2051	**2050**	**2050**	190.92
12a	15 × 8	4.03	1969	2030	2019	2031	**2018**	2019	2019	207.47
13a	20 × 10	1.34	2161	2257	**2194**	2257	2248	2253	2253	111.58
14a	20 × 10	2.99	2161	2167	2167	2167	**2163**	2167	2167	257.75
15a	20 × 10	5.02	2161	2165	2165	2165	**2162**	2165	2165	412.29
16a	20 × 10	1.34	2148	2256	**2211**	2256	2244	2252	2252	119.59
17a	20 × 10	2.99	2088	2140	**2109**	2140	2130	2134	2134	324.56
18a	20 × 10	5.02	2057	2127	**2089**	2127	2119	2123	2123	437.40

Table 10
Adjusted p-values.

i	Hypothesis	z	Unadjusted *p*	Nemenyi	Holm
1	hGA versus SSPR	4.7661588	0.0001000	0.0015000	0.0015000
2	CDDS versus SSPR	4.008919	0.0001000	0.0015000	0.0014000
3	hGA versus QPSO	3.6080268	0.0003000	0.0045000	0.0039000
4	hGA versus GRASP	3.4298526	0.0006000	0.0090000	0.0072000
5	hGA versus eGA	3.2071349	0.0013000	0.0195000	0.0143000
6	CDDS versus QPSO	2.8507866	0.0044000	0.0660000	0.0440000
7	CCDS versus GRASP	2.6726124	0.0075000	0.1125000	0.0675000
8	CDDS versus eGA	2.4498947	0.0143000	0.2145000	0.1144000
9	eGA versus SSPR	1.5590239	0.1190000	1	0.8330000
10	GRASP versus SSPR	1.3363062	0.1814000	1	1
11	QPSO versus SSPR	1.1581320	0.2468000	1	1
12	hGA versus CDDS	0.7572402	0.4489000	1	1
13	eGA versus QPSO	0.4008919	0.6885000	1	1
14	eGA versus GRASP	0.2227177	0.8238000	1	1
15	GRASP versus QPSO	0.1781742	0.8586000	1	1

Table 11
Multiple comparisons for different significance level (α).

Alg. Versus alg.	hGA		eGA		CDDS		SSPR		QPSO		GRASP	
	0.1	0.05	0.1	0.05	0.1	0.05	0.1	0.05	0.1	0.05	0.1	0.05
hGA	–	–	+	+	–	–	+	+	+	+	+	+
eGA	–	–	–	–	–	–	–	–	–	–	–	–
CDDS	–	–	–	–	–	–	+	+	+	+	+	–
SSPR	–	–	–	–	–	–	–	–	–	–	–	–
QPSO	–	–	–	–	–	–	–	–	–	–	–	–
GRASP	–	–	–	–	–	–	–	–	–	–	–	–

other dynamic flexible job shop scheduling studies. Thus, the performance comparisons of the proposed GRASP algorithm with the most well-known six dispatching rules: EDD, SPT, FIFO, SL, LPT and CR are made in this stage under different level of shop utilizations, due date tightness factors and flexibility levels.

The proposed GRASP algorithm employs the lexicographic method to evaluate generated schedules. Therefore, the first objective function (mean tardiness) is utilized to compare the proposed GRASP algorithm with heuristics. This is mainly due to the fact that in a lexicographic logic secondary objective can only be improved if primary objectives are not worsened. Therefore, it is not easy to compare other criteria directly. Moreover, the number of accepted orders must also be taken into account to make fair comparisons as number of accepted orders can be different for different rules/algorithms. Computational results of the experiments are given in Appendix B in detail. The summarization of the obtained results (mean tardiness and number of accepted orders) for each problem size is shown in Figs. 6–8. Considering all combinations of experiments given in Figs. 6–8, it can be clearly expressed that the proposed GRASP algorithm in most of the experiments in general significantly reduces the mean tardiness and allows more orders to be accepted compared to the dispatching rules. However, although the proposed GRASP algorithm in some experimental groups reduced the mean tardiness, some dispatching rules are more successful in accepting more orders by enduring higher mean tardiness. For 5 × 5, 10 × 10 and 15 × 15 problems, a maximum of 6, 12 and 20 orders are accepted, respectively. Considering Fig. 6 for the 5 × 5 problem, mean tardiness decreases and accepted order amount increases as the machine utilization and flexibility level increase, provided that the other conditions are the same. When the distribution rules are compared among themselves considering all combinations of experiments, a sharp distinction cannot be made between them. For the 5 × 5 problem with (-) T and

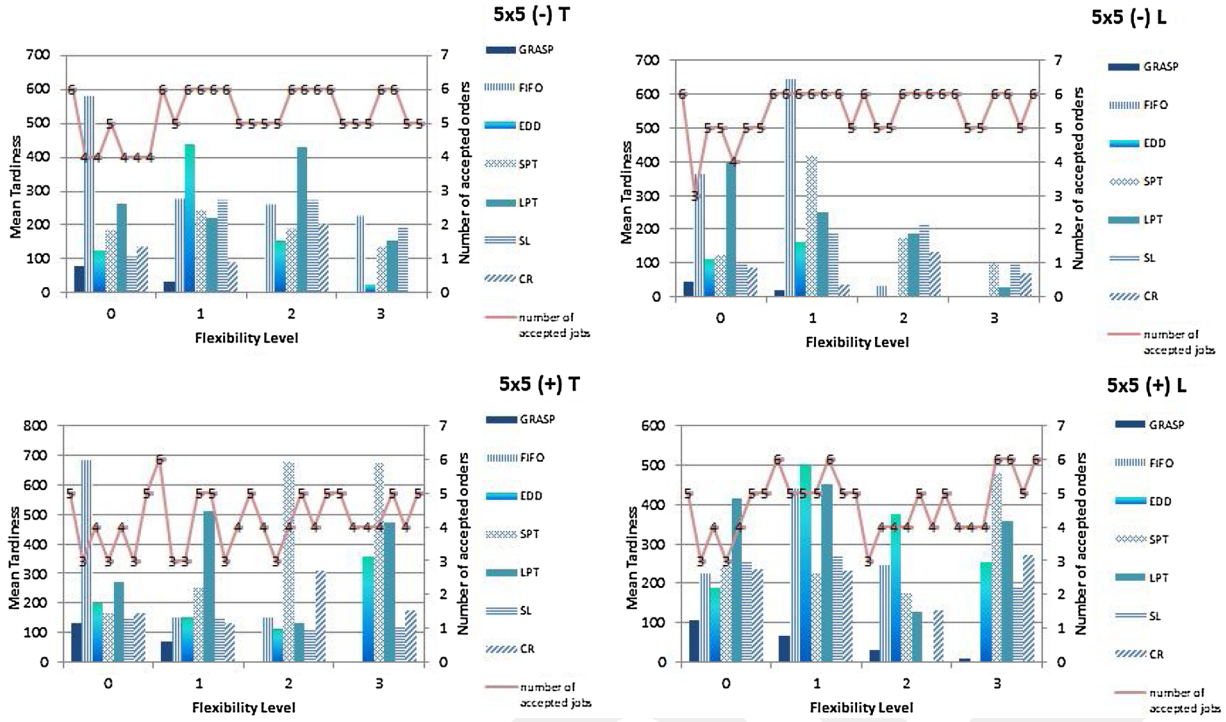

Fig. 6. The results for 5 × 5 problems.
***(-) low resource utilization, (+) high resource utilization, T-tight due date, L-loose due date.

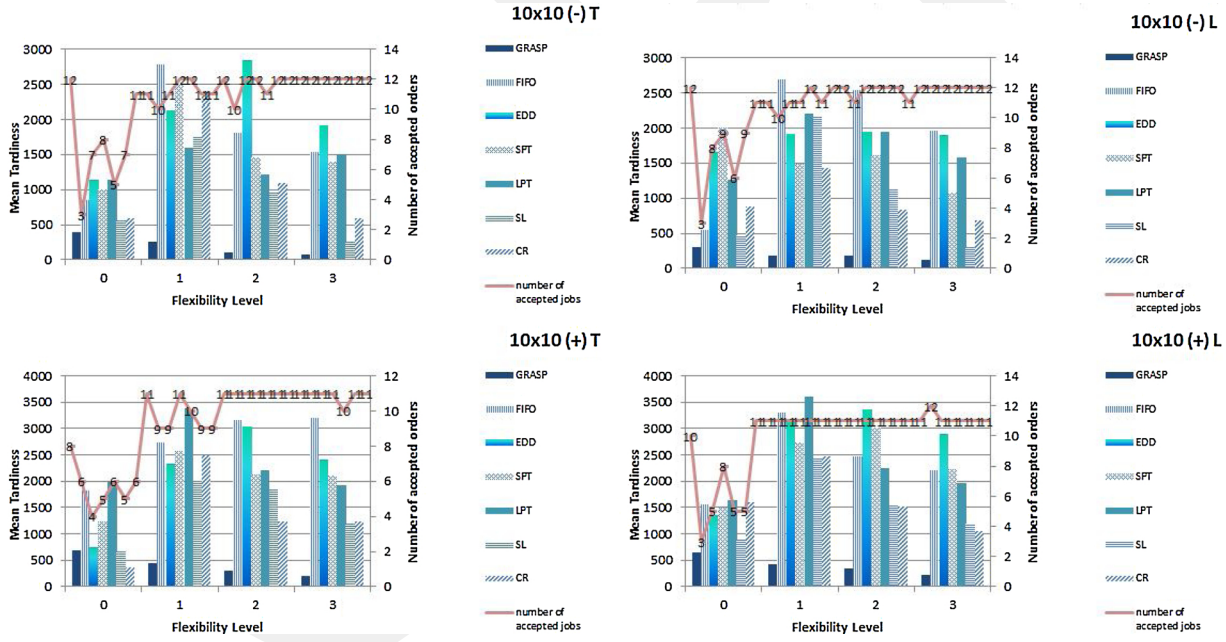

Fig. 7. The results for 10 × 10 problems.

(+) L test combinations and machine flexibility level of 2 and 3, while the proposed GRASP algorithm reduced the mean tardiness more, some dispatching rules are more successful in accepting more orders with higher mean tardiness. As the problem size increases, the proposed GRASP algorithm comes to the fore again in terms of its performance. Except for a few exceptional cases, the SL and CR rules perform best among the dispatching rules for the 10 × 10 and 15 × 15 problems. As an exception, for example, it can be seen Fig. 7, the FIFO rule accepted more orders than the other methods and algorithms for the 10 × 10 problem with (+) L test combination when the machine flexibility is 3. Although the performance of the dispatching rules depends on the problem size, the proposed GRASP algorithm generally provides better performance than all without being affected by the problem size. In general, although the proposed GRASP algorithm gives a better performance than six dispatching rules used in this paper, some exceptional cases may arise due to varying problem sizes and test conditions.

In addition, results of all combinations are examined by ANOVA for testing the statistical significance. The results of ANOVA are shown in Table 12. By considering all problem sizes (p-value < 0.05), *'shop utilization'* and *'degree of problem flexibility'* are found as the important factors for scheduling environment, whereas 'due date tightness' is found as the negligible factor. Furthermore, there are some interactions

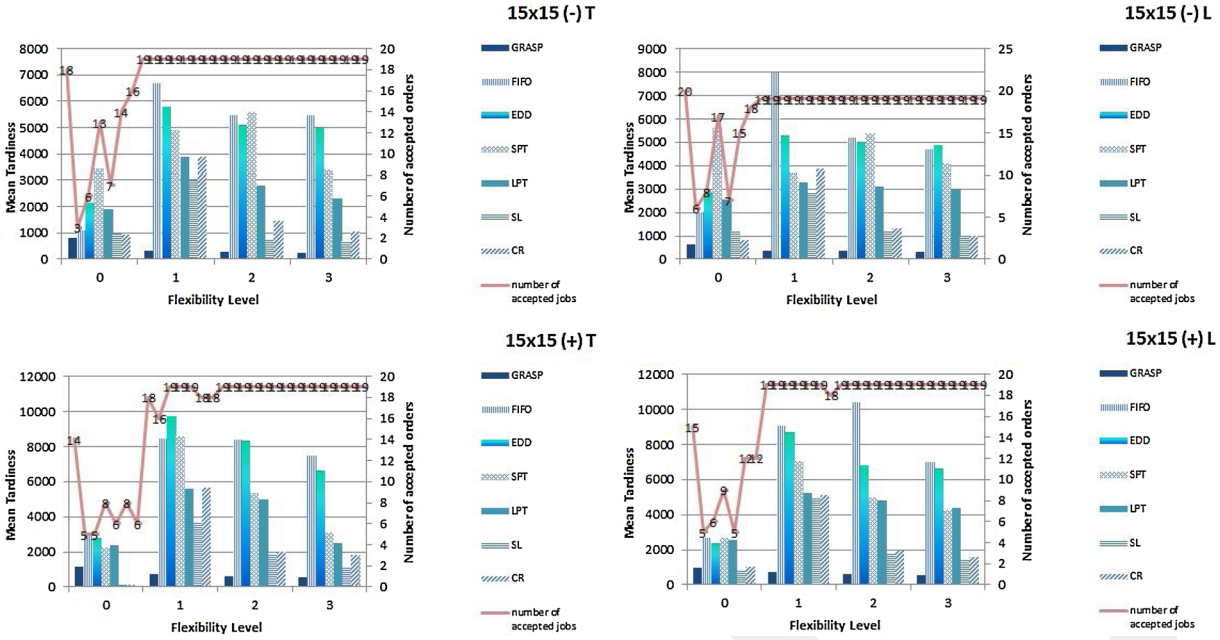

Fig. 8. The results for 15 × 15 problems.

Table 12
ANOVA results.

Source/ Problem sizes	P (5 × 5)	P (10 × 10)	P (15 × 15)
Shop_utilization	0.000	0.000	0.000
Duedate_tightness	0.098	0.487	0.284
Flexibility	0.000	0.000	0.000
Shop_utilization * Duedate_tightness	0.017	0.303	0.189
Shop_utilization * Flexibility	0.000	0.000	0.240
Duedate_tightness * Flexibility	0.000	0.019	0.011

between the factors. For example, the interaction between *'due date tightness'* and *'flexibility'* is shown in Table 12 (p-value < 0.05) for 5 × 5, 10 × 10 and 15 × 15.

4.4. Experiment III

This stage of the computational study compares the proposed GRASP algorithm under the event-driven and the periodical re-scheduling mechanism. The periodical scheduling periods are considered as 120 min, 180 min, and 240 min. The detailed comparisons of

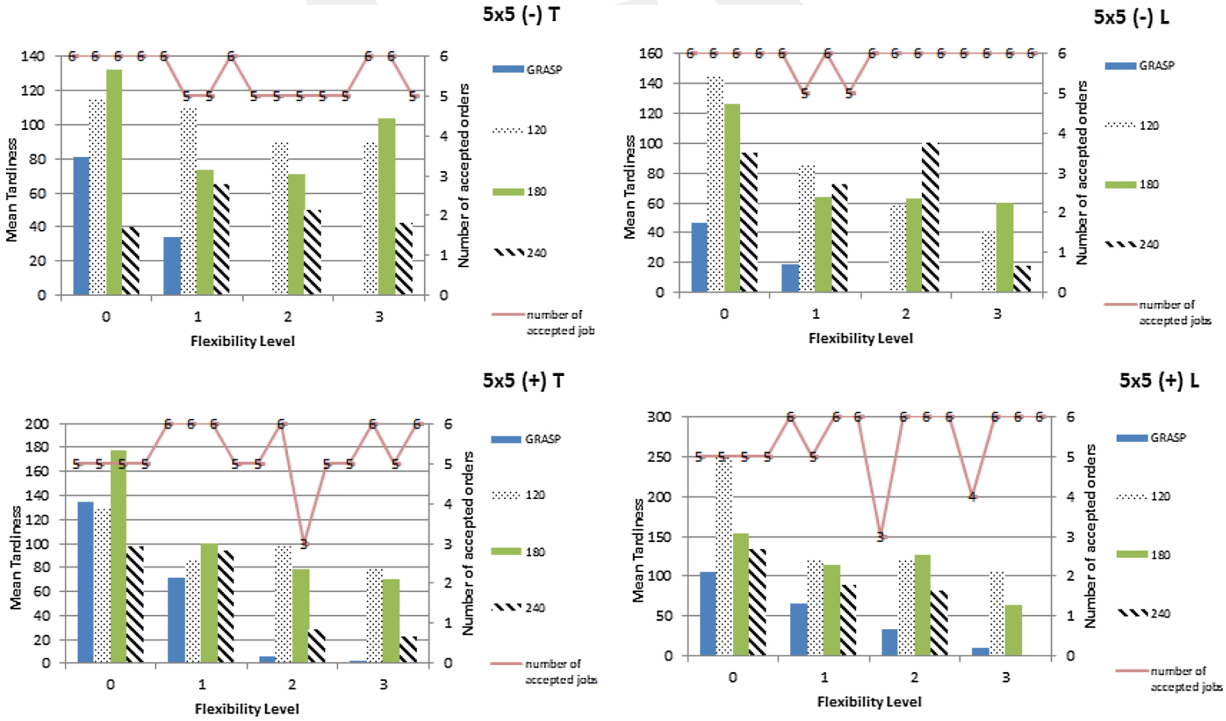

Fig. 9. The results of the rescheduling policies for 5 × 5 problems.
***(-) low resource utilization, (+) high resource utilization, T-tight due date, L-loose due date.

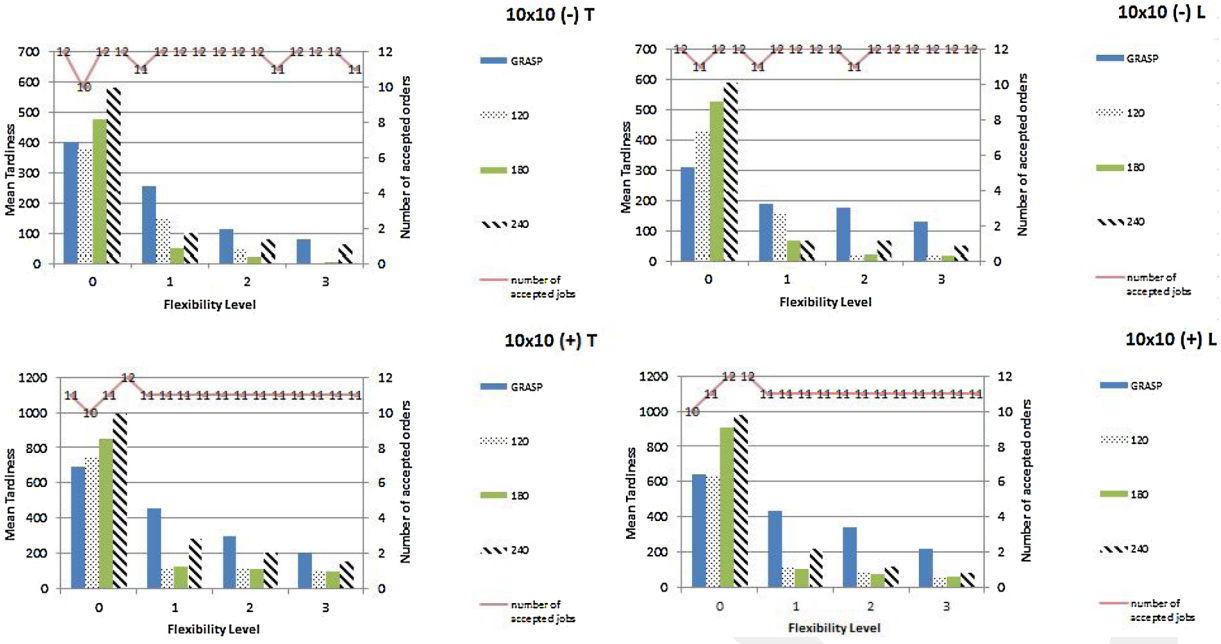

Fig. 10. The results of the rescheduling policies for 10 × 10 problems.

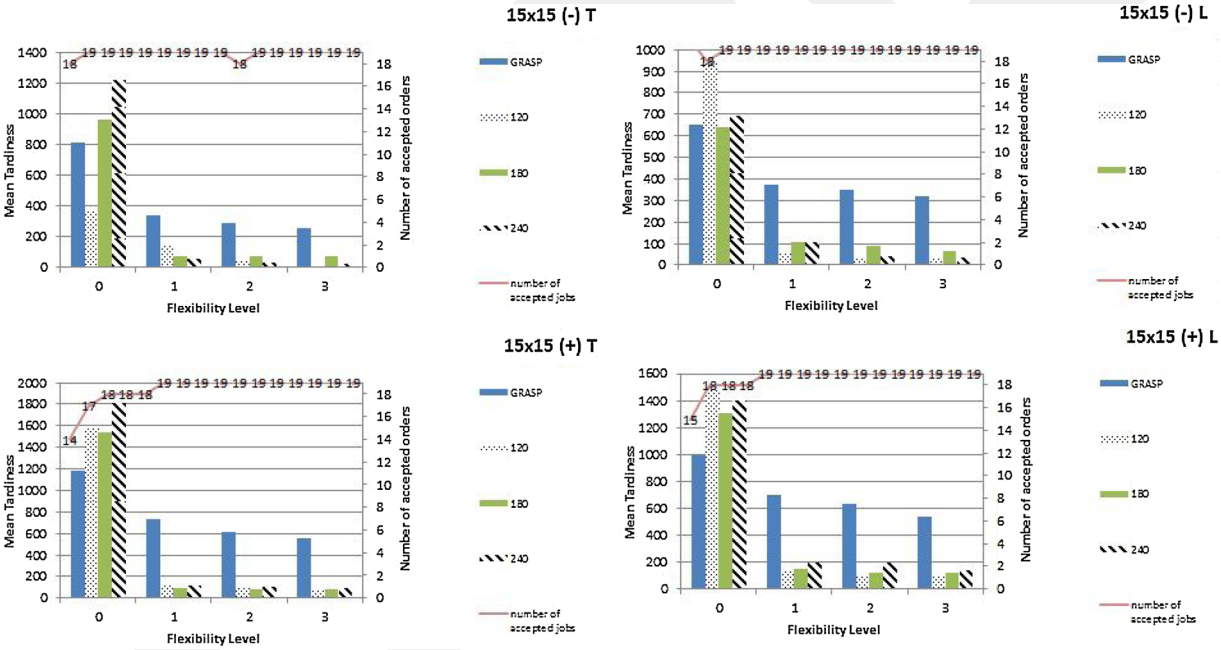

Fig. 11. The results of the rescheduling policies for 15 × 15 problems.

periodical and event-driven rescheduling mechanisms are shown in Tables B4–B6 in Appendix. The summarization of the obtained results (mean tardiness and number of accepted orders) for each problem size is shown in Figs. 9–11. For the 5 × 5 problem (see Fig. 9), the event-driven rescheduling gives the better performance in reducing the mean tardiness for all experiment settings, with the exception of (-) T experiment combination with the flexibility level of 0, for which periodical rescheduling with 240 min shows a better performance, (+) T experiment combination with the flexibility level of 0, for which periodical rescheduling with 240 min shows a better performance, and (+) L experiment combination with the flexibility level of 3, for which periodical rescheduling with 240 min shows a better performance. The periodic rescheduling is generally more successful in accepting more orders by enduring the higher mean tardiness when it is compared to the event-driven rescheduling for the 5 × 5 problem. The mean tardiness generally decreases as machine flexibility level is increased in the same experiment combination. However, in some cases, the decrease in mean tardiness causes a decrease in the number of accepted orders. For example, when flexibility has decreased from 0 to 3 in the (+) L experiment combination, the number of orders accepted has also decreased, despite the decrease in the mean tardiness. The decrease in machine unavailability time and increase in due date generally have a positive effect on minimizing mean tardiness. For 10 × 10 and 15 × 15 problems (see Figs. 10 and 11), regardless of the time periods considered, the periodic rescheduling is usually more successful than the event-driven rescheduling in minimizing the mean tardiness and maximizing the accepted orders. It can be seen from Figs. 9–11 that flexibility generally helps to improve scheduling performance with

degradation of the mean tardiness or the increment of the number of the accepted orders for both periodic and event driven rescheduling mechanisms.

To compare the event-driven rescheduling mechanism with the periodical rescheduling mechanism statistically, Friedman test is also conducted. The null hypothesis for Friedman's test remarks the equality of medians between the algorithms. Then, Holm's post-hoc procedures are selected for the execution of the Friedman test in order to apply multiple comparisons process. According to this hypothesis, the best scheduling method has emerged as the event-driven rescheduling mechanism, while the worst scheduling method has emerged as the periodical rescheduling mechanism with 120 scheduling frequency for the 5×5 problems (with all combinations). For 10×10 and 15×15 problems, the best scheduling method has emerged as the periodical rescheduling mechanism with 120 scheduling frequency, while the worst scheduling method has emerged as the event-driven rescheduling mechanism.

In general, although the periodic rescheduling gives a better performance than the event-driven rescheduling, some exceptional cases may arise due to varying problem sizes and test conditions. The periodical rescheduling mechanism gathers all of the data from the production environment at the specific period and evaluates this information during the rescheduling process. This inevitably enables it to provide better results and the production decision may be more accurate because of the accumulated information. However, event-driven rescheduling mechanism takes into account dynamic events instantly and this may not lead effective schedules in some occasions. Results clearly show that the determination of the rescheduling mechanism is a critical decision for dynamic scheduling performance.

5. Conclusions

In the literature, GRASP has not been tailored to generate new schedules during dynamic flexible shop floor environment. We noticed that few studies take into account more than one real-time situation such as order arrivals, machine breakdowns, order cancellations, due date changes, urgent order arrivals. Also, all of the previous papers don't consider machine capacity constraints in addition to sequence dependent setup times and dynamic events. This paper bridges this gap for this problem and the main motivation of the present study is to propose a multi-start and constructive search strategy to solve FJSP and DFJSP by addressing several real-life situations. Besides, Order Review Release (ORR) mechanism and order acceptance/rejection decisions are also incorporated into the proposed method in order to adjust capacity execution considering customer due date requirements. The relative capability of the constructed schedules is evaluated by using lexicographic method. Extensive computational studies are conducted along with statistical performance analyses. Firstly, the capability of the proposed algorithm is investigated for solving benchmark problems. Satisfactory results are obtained. The proposed algorithm is employed to solve the dynamic scheduling problems over different levels of machine flexibility, shop utilization and due date tightness with event-driven rescheduling mechanism. Results are compared with dispatching rules. It is investigated that the proposed algorithm is able to obtain better performance by accepting more orders. Finally, event driven and periodical rescheduling mechanisms, which are utilized within the proposed algorithm, are compared. It is observed that the periodical rescheduling strategy can provide better schedules with the penalty of being less responsive in the most cases. As a future study, the problem can be extended by integrating maintenance scheduling and a better capacity management strategy can be developed by making use of the revenue management principles.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

In Tables A1–A3 event type shows order arrival, machine breakdown, order cancellation, rush order, change of due date, respectively. For example; J1 gets as an order at time 50; M1 fails at time 120; the repair operation for this machine takes 30 min; 4th order is cancelled at time 350; the 7th order reaches at time 100 as a urgent order; the delivery time of the 6th order is switched at time 120 and the revised delivery time is determined by customer as 800 min. Sequence dependent setup times are uniformly distributed (8,15) and minor setup time as 10 min is use when the operation is first processed operation on the machine. Processing times, which are given in the Tables A1–A3, are uniformly distributed. Capable machine sets of the operations are assigned randomly. Each shift is considered 480 min. The data sets are available for download at http://web.deu.edu.tr/baykasoglu/DFJSP_data.rar.

Table A1
Data for 5×5 DFJSP.

Jobs	Processing Times (min)					Dynamic events	Occurrence Time (min)	Event Type[a]	Information about dynamic events
J1	U(21,45)	U(35,41)	U(53,68)	U(21,45)	U(55,82)	1	50	1	J1- Order 6
J2	U(55,75)	U(31,55)	U(12,31)	U(42,60)	U(39,55)	2	100	1	J3- Order 7
J3	U(34,42)	U(64,78)	U(19,30)	U(92,100)	U(83,110)	3	120	2	M1-repair time 30 min
J4	U(87,100)	U(69,82)	U(87,95)	U(93,110)	U(60,75)	4	240	2	M2-repair time 42min
J5	U(98,115)	U(44,60)	U(49,66)	U(96,102)	U(19,25)	5	310	2	M3-repair time 51min
Machine Capacity(machine-min)			M1-450 M2-420 M3-450 M4-440 M5-450			6	210	2	M4-repair time 22 min
						7	300	2	M5-repair time 43 min
						8	350	3	Order 4
						9	100	4	Order 7
						10	120	5	Order 6-Due date 800min

[a] 1:New order arrival, 2: Machine breakdown, 3: Order cancellation, 4: Rush order, 5: Change of due date.

Table A2
Data for 10 × 10 DFJSP.

Jobs	Processing Times(min)										Dynamic events	Occurrence Time (min)	Event Type[a]	Information about dynamic events
J1	U(89,114)	U(79,96)	U(99,107)	U(53,57)	U(66,75)	U(70,90)	U(62,71)	U(81,96)	U(61,88)	U(72,85)	1	100	1	J5-Order 11
J2	U(74,85)	U(89,102)	U(84,100)	U(91,108)	U(91,101)	U(65,75)	U(91,96)	U(64,66)	U(79,93)	U(71,84)	2	150	1	J8- Order 12
J3	U(78,90)	U(88,92)	U(76,88)	U(68,74)	U(62,73)	U(51,74)	U(73,108)	U(92,118)	U(89,113)	U(81,96)	3	200	1	J6- Order 13
J4	U(58,69)	U(81,92)	U(94,102)	U(61,71)	U(87,94)	U(62,74)	U(63,68)	U(70,81)	U(84,97)	U(75,82)	4	120	2	M1-repair time 30 min
J5	U(59,64)	U(57,62)	U(60,70)	U(81,92)	U(80,92)	U(89,103)	U(90,105)	U(64,70)	U(61,73)	U(67,79)	5	240	2	M2-repair time 42 min
J6	U(91,98)	U(56,63)	U(85,97)	U(83,96)	U(68,78)	U(59,68)	U(56,67)	U(60,69)	U(77,87)	U(78,93)	6	310	2	M3-repair time 51 min
J7	U(80,86)	U(84,96)	U(62,73)	U(86,100)	U(76,85)	U(70,79)	U(99,109)	U(54,60)	U(97,108)	U(90,107)	7	210	2	M4-repair time 22 min
J8	U(95,99)	U(53,62)	U(67,74)	U(78,93)	U(69,81)	U(92,105)	U(89,104)	U(97,105)	U(96,110)	U(93,109)	8	300	2	M5-repair time 43 min
J9	U(70,74)	U(57,65)	U(93,109)	U(54,64)	U(62,70)	U(53,63)	U(100,114)	U(85,91)	U(86,90)	U(62,71)	9	150	2	M6-repair time 42 min
J10	U(81,86)	U(59,64)	U(61,66)	U(57,65)	U(65,69)	U(77,87)	U(70,78)	U(53,60)	U(68,79)	U(52,57)	10	210	2	M7-repair time 36 min
Machine Capacity(machine-min)				M1-400 M2-420M3-450M4-440M5-450 M6-460 M7-450							11	420	2	M8-repair time 65 min
				M8-440 M9-450 M10-420							12	320	2	M9-repair time 80 min
Order Arrival Time (job-min)				J1-0 J2-0 J3-0 J4-0 J5-0 J6-20 J7-40 J8-55 J9-63 J10-80							13	520	2	M10-repair time 45 min
											14	550	3	Order 11
											15	80	4	Order 10
											16	100	5	Order 8-Due date 2500 min

Table A3
Data for 15 × 15 DFJSP.

Jobs	Machine Sequences and Processing Times(min)															
J1	U(80,95)	U(99,114)	U(91,105)	U(57,67)	U(61,72)	U(92,107)	U(74,88)	U(99,117)	U(63,73)	U(96,110)	U(77,90)	U(95,113)	U(61,73)	U(59,68)	U(73,87)	
J2	U(80,92)	U(77,90)	U(75,86)	U(67,77)	U(98,113)	U(66,75)	U(52,61)	U(89,106)	U(74,89)	U(96,113)	U(84,100)	U(99,117)	U(52,61)	U(68,80)	U(87,104)	
J3	U(100,113)	U(66,79)	U(80,95)	U(76,91)	U(69,82)	U(53,62)	U(88,101)	U(57,67)	U(96,113)	U(90,106)	U(74,88)	U(100,117)	U(72,88)	U(96,114)	U(56,66)	
J4	U(57,65)	U(73,87)	U(51,61)	U(97,114)	U(52,62)	U(58,63)	U(75,81)	U(56,61)	U(82,90)	U(76,82)	U(99,108)	U(64,69)	U(88,96)	U(88,94)	U(66,71)	
J5	U(61,72)	U(62,73)	U(77,88)	U(67,78)	U(59,70)	U(75,82)	U(64,68)	U(87,94)	U(54,59)	U(60,64)	U(79,86)	U(73,80)	U(78,85)	U(64,68)	U(74,81)	
J6	U(53,63)	U(95,112)	U(97,116)	U(58,68)	U(76,89)	U(87,94)	U(58,62)	U(65,70)	U(70,76)	U(68,74)	U(72,77)	U(62,67)	U(80,86)	U(90,96)	U(63,68)	
J7	U(84,98)	U(95,110)	U(88,104)	U(60,71)	U(82,97)	U(78,85)	U(71,78)	U(76,83)	U(54,58)	U(54,58)	U(91,100)	U(64,70)	U(51,56)	U(57,62)	U(69,75)	
J8	U(79,93)	U(100,119)	U(57,68)	U(51,60)	U(54,63)	U(73,79)	U(68,74)	U(55,60)	U(51,56)	U(55,60)	U(84,91)	U(90,98)	U(85,93)	U(63,68)	U(95,103)	
J9	U(76,90)	U(62,74)	U(86,103)	U(76,90)	U(94,109)	U(71,77)	U(92,99)	U(53,57)	U(61,66)	U(78,84)	U(96,100)	U(69,75)	U(51,55)	U(66,72)	U(57,60)	
J10	U(84,99)	U(55,63)	U(90,105)	U(87,104)	U(61,67)	U(74,81)	U(94,102)	U(84,92)	U(74,80)	U(83,90)	U(59,64)	U(56,61)	U(85,93)	U(66,72)	U(85,92)	
J11	U(68,79)	U(67,79)	U(89,105)	U(75,87)	U(96,115)	U(81,88)	U(67,73)	U(89,97)	U(74,80)	U(81,87)	U(81,87)	U(83,91)	U(96,105)	U(64,69)	U(64,69)	
J12	U(86,99)	U(82,95)	U(85,99)	U(61,72)	U(72,86)	U(67,80)	U(54,59)	U(85,92)	U(61,66)	U(63,68)	U(88,96)	U(79,85)	U(51,55)	U(51,55)	U(94,102)	
J13	U(95,113)	U(95,112)	U(96,112)	U(90,102)	U(97,115)	U(63,75)	U(90,98)	U(81,86)	U(57,62)	U(51,55)	U(92,100)	U(70,75)	U(86,93)	U(63,68)	U(56,60)	
J14	U(77,88)	U(52,61)	U(64,75)	U(57,67)	U(77,90)	U(88,104)	U(83,90)	U(59,64)	U(77,84)	U(68,74)	U(70,76)	U(85,90)	U(53,58)	U(94,101)	U(58,63)	
J15	U(52,62)	U(51,60)	U(64,75)	U(86,101)	U(80,94)	U(77,90)	U(59,64)	U(98,107)	U(63,69)	U(94,103)	U(51,55)	U(89,94)	U(89,97)	U(51,56)	U(99,108)	
	Machine Capacity (machine-min)		M1-400 M2-420 M3-450 M4-440 M5-450 M6-460 M7-450 M8-440 M9-450 M10-420 M11-450 M12-455 M13-470 M14-460 M15-470													
	Order Arrival Time (job-min)		J1-0 J2-0 J3-0 J4-0 J5-0 J6-20 J7-40 J8-0 J9-23 J10-156 J11-165 J12-170 J13-182 J14-210 J15-250													
	Dynamic Events	Occurence Time (min)	Event Types	Information about dynamic events												
	1	300	1	J4-Order 16												
	2	450	1	J5-Order 17												
	3	500	1	J6-Order 18												
	4	650	1	J7-Order 19												
	5	800	1	J12-Order 20												
	6	120	2	M1-repair time 30 min												
	7	240	2	M2-repair time 42 min												
	8	310	2	M3-repair time 51 min												
	9	210	2	M4-repair time 22 min												
	10	300	2	M5-repair time 43 min												
	11	150	2	M6-repair time 42 min												
	12	210	2	M7-repair time 36 min												
	13	420	2	M8-repair time 65 min												
	14	320	2	M9-repair time 80 min												
	15	520	2	M10-repair time 45 min												
	16	120	2	M11-repair time 52 min												
	17	210	2	M12-repair time 46 min												
	18	250	2	M13-repair time 35 min												
	19	400	2	M14-repair time 65 min												
	20	500	2	M15-repair time 53 min												
	21	750	3	Order 17												
	22	455	4	Order 12												
	23	520	5	Order 10- Due date 3900 min												

Appendix B

See Tables B1–B3

Table B1
The results of GRASP and dispatching rules for 5 × 5 problems.

Jobs	Machines	Utilization	Due date tightness	Methods	Flexibility level[a]	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
5	5	–	T	GRASP	0	81	5103	814	557	7	6	90.01
					1	34	5302	844	622	7	6	114.12
					2	0	4018	719	513	7	5	129.45
					3	0	3583	747	545	7	5	130.15
5	5	–	T	FIFO	0	581	2391	1025	604	7	4	0.05
					1	274	2287	1264	835	7	5	0.06
					2	256	821	990	513	7	5	0.08
					3	225	6223	1025	618	7	5	0.08
5	5	–	T	EDD	0	124	5687	1051	600	7	4	0.06
					1	437	5632	1661	928	7	6	0.07
					2	152	4561	1052	605	7	5	0.08
					3	22	4471	922	579	7	5	0.08
5	5	–	T	SPT	0	183	3036	874	641	7	5	0.05
					1	242	1156	1052	684	7	6	0.06
					2	185	1261	1095	651	7	6	0.07
					3	133	13288	867	596	7	6	0.08
5	5	–	T	LPT	0	262	3986	888	607	7	4	0.05
					1	221	13397	1026	690	7	6	0.06
					2	428	11181	1155	860	7	6	0.07
					3	155	8344	847	645	7	6	0.08
5	5	–	T	SL	0	102	5230	893	593	7	4	0.06
					1	271	7197	1362	818	7	6	0.07
					2	269	3278	926	608	7	6	0.07
					3	193	4490	990	593	7	5	0.08
5	5	–	T	CR	0	131	694	702	530	7	4	0.07
					1	88	3241	921	722	7	5	0.08
					2	200	4859	837	594	7	6	0.08
					3	1	2081	806	571	7	5	0.08
5	5	–	L	GRASP	0	47	3098	814	651	7	6	90.05
					1	19	2758	825	650	7	6	113.20
					2	0	8300	845	516	7	6	129.45
					3	0	4672	845	563	7	6	130.15
5	5	–	L	FIFO	0	360	1861	871	600	7	3	0.05
					1	640	8684	1692	960	7	6	0.06
					2	31	2059	831	516	7	5	0.08
					3	0	2486	749	491	7	5	0.08
5	5	–	L	EDD	0	111	2724	1125	694	7	5	0.06
					1	161	9239	1152	679	7	6	0.07
					2	0	1651	856	554	7	5	0.08
					3	0	1207	755	549	7	5	0.08
5	5	–	L	SPT	0	126	4175	1051	672	7	5	0.05
					1	415	4635	1467	948	7	6	0.06
					2	174	10066	1097	629	7	6	0.07
					3	100	9467	871	692	7	6	0.08
5	5	–	L	LPT	0	394	2499	888	661	7	4	0.05
					1	250	8376	920	740	7	6	0.06
					2	186	4826	1053	712	7	6	0.07
					3	30	5039	843	589	7	6	0.08
5	5	–	L	SL	0	99	1872	1054	714	7	5	0.06
					1	185	12143	1151	765	7	6	0.07
					2	211	9521	1121	674	7	6	0.07
					3	101	4277	887	652	7	5	0.08
5	5	–	L	CR	0	86	3847	768	562	7	5	0.07
					1	35	3351	853	676	7	5	0.08
					2	133	8536	884	605	7	6	0.08
					3	71	1991	862	690	7	6	0.08

(continued on next page)

Table B1 (*continued*)

Jobs	Machines	Utilization	Due date tightness	Methods	Flexibility level[a]	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
5	5	+	T	GRASP	0	135	7557	802	695	7	5	90.01
					1	71	4012	777	642	7	6	114.12
					2	6	5858	787	553	7	5	129.45
					3	2	5884	802	537	7	5	130.15
5	5	+	T	FIFO	0	681	2635	1125	740	7	3	0.05
					1	151	41	1051	664	7	3	0.06
					2	151	0	1051	559	7	4	0.08
					3	0	1383	834	519	7	4	0.08
5	5	+	T	EDD	0	201	6274	1153	651	7	4	0.06
					1	151	41	1051	664	7	3	0.07
					2	112	2814	1053	607	7	3	0.08
					3	358	3185	1258	633	7	4	0.08
5	5	+	T	SPT	0	167	4065	1051	847	7	3	0.05
					1	252	8977	1219	780	7	5	0.06
					2	677	2735	1116	647	7	4	0.07
					3	672	2437	1121	648	7	4	0.08
5	5	+	T	LPT	0	270	12513	832	603	7	4	0.05
					1	510	36205	1506	980	7	5	0.06
					2	135	1466	1051	755	7	5	0.07
					3	472	4558	1247	872	7	5	0.08
5	5	+	T	SL	0	146	2745	1054	516	7	3	0.06
					1	151	811	1051	664	7	3	0.07
					2	111	2814	1052	607	7	4	0.07
					3	116	1032	799	624	7	4	0.08
5	5	+	T	CR	0	167	1405	755	539	7	5	0.07
					1	134	7920	850	725	7	4	0.08
					2	308	8886	775	582	7	5	0.08
					3	177	7151	1053	564	7	5	0.08
5	5	+	L	GRASP	0	106	8619	802	695	7	5	90.05
					1	66	4761	829	719	7	6	113.20
					2	33	517	571	462	7	3	129.45
					3	11	1341	817	505	7	4	130.15
5	5	+	L	FIFO	0	225	2	736	546	7	3	0.05
					1	439	2399	1504	960	7	5	0.06
					2	247	87	1092	598	7	4	0.08
					3	0	1383	834	519	7	4	0.08
5	5	+	L	EDD	0	190	2846	1127	599	7	4	0.06
					1	499	9492	1532	882	7	5	0.07
					2	377	58	114	779	7	4	0.08
					3	253	1419	1153	618	7	4	0.08
5	5	+	L	SPT	0	244	1821	1151	980	7	3	0.05
					1	224	9046	1095	708	7	5	0.06
					2	173	1300	811	585	7	4	0.07
					3	479	5968	1212	799	7	6	0.08
5	5	+	L	LPT	0	415	5048	1174	829	7	4	0.05
					1	451	36205	1506	990	7	6	0.06
					2	129	1845	833	647	7	5	0.07
					3	359	13954	1252	829	7	6	0.08
5	5	+	L	SL	0	253	5658	1231	875	7	5	0.06
					1	268	11643	1157	850	7	5	0.07
					2	0	918	832	552	7	4	0.07
					3	192	3264	834	753	7	5	0.08
5	5	+	L	CR	0	234	4394	1248	814	7	5	0.07
					1	230	10705	1151	945	7	5	0.08
					2	131	1481	762	581	7	5	0.08
					3	272	6010	1247	946	7	6	0.08

[a] 0: Static problem, 1: Small flexibility level, 2: Medium flexibility level, 3: Large flexibility level.

Table B2
The results of GRASP and dispatching rules for 10 × 10 problems.

Jobs	Machines	Utilization	Due date tightness	Methods	Flexibility level	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
10	10	–	T	GRASP	0	399	65973	2981	2006	13	12	191.48
					1	256	43663	2719	1867	13	11	201.24
					2	111	54287	2501	1799	13	12	205.63
					3	81	78965	2356	1640	13	12	208.42
10	10	–	T	FIFO	0	856	1	2645	1790	13	3	0.21
					1	2785	14040	7514	4566	13	10	0.22
					2	1810	26815	5590	3430	13	10	0.25
					3	1540	163550	4560	2690	13	12	0.28
10	10	–	T	EDD	0	1140	51497	4214	2670	13	7	0.25
					1	2120	44108	6924	3901	13	11	0.24
					2	2840	285190	7990	4130	13	12	0.29
					3	1920	18814	6820	3330	13	12	0.29
10	10	–	T	SPT	0	1014	66023	3755	2288	13	8	0.25
					1	2510	156570	6800	4030	13	12	0.26
					2	1460	251950	4740	2980	13	12	0.27
					3	1410	251500	4490	2850	13	12	0.28
10	10	–	T	LPT	0	1142	38106	3033	2574	13	5	0.25
					1	1590	201040	4660	2940	13	12	0.26
					2	1220	194790	4530	2610	13	11	0.27
					3	1500	202430	4010	2740	13	12	0.28
10	10	–	T	SL	0	574	33080	3466	2310	13	7	0.30
					1	1760	130650	5600	3500	13	11	0.32
					2	970	15592	4630	2670	13	12	0.32
					3	270	123720	3290	1990	13	12	0.35
10	10	–	T	CR	0	603	33906	2773	2247	13	11	0.29
					1	2400	189830	5160	3900	13	11	0.32
					2	1100	164420	3440	2880	13	12	0.35
					3	606	91817	3160	2385	13	12	0.36
10	10	–	L	GRASP	0	309	65347	3195	2126	13	12	191.48
					1	188	35071	2701	2109	13	11	201.24
					2	177	38290	2612	2008	13	12	205.63
					3	128	57343	2480	1917	13	12	208.42
10	10	–	L	FIFO	0	553	2	2484	1718	13	3	0.21
					1	2703	27927	8090	4688	13	10	0.22
					2	2550	118290	8370	4320	13	11	0.25
					3	1970	213220	6180	3050	13	12	0.28
10	10	–	L	EDD	0	1650	122340	6330	3460	13	8	0.25
					1	1916	47935	6924	3901	13	11	0.24
					2	1950	150540	6340	3050	13	12	0.29
					3	1900	23046	7610	3720	13	12	0.29
10	10	–	L	SPT	0	2019	78847	5186	3776	13	9	0.25
					1	1490	202830	5110	3430	13	11	0.26
					2	1610	170110	5350	3310	13	12	0.27
					3	1090	184490	4730	2980	13	12	0.28
10	10	–	L	LPT	0	1261	54493	3450	2954	13	6	0.25
					1	2200	265140	5700	4000	13	12	0.26
					2	1940	174960	4680	3410	13	12	0.27
					3	1580	202430	4010	2740	13	12	0.28
10	10	–	L	SL	0	482	34341	3466	2401	13	9	0.30
					1	2168	58321	6656	3910	13	11	0.32
					2	1130	117390	4520	3120	13	12	0.32
					3	310	57374	3096	2193	13	12	0.35
10	10	–	L	CR	0	891	58638	3567	3752	13	11	0.29
					1	1430	127120	4720	3420	13	12	0.32
					2	837	59020	3451	2825	13	11	0.35
					3	685	76858	3450	2610	13	12	0.36

(*continued on next page*)

Table B2 (*continued*)

Jobs	Machines	Utilization	Due date tightness	Methods	Flexibility level	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
10	10	+	T	GRASP	0	693	47168	3065	2493	13	8	191.48
					1	455	67567	2981	2288	13	11	201.24
					2	301	84278	2720	2092	13	11	205.63
					3	202	85685	2981	1924	13	11	208.42
10	10	+	T	FIFO	0	1828	9	6397	3610	13	6	0.21
					1	2753	29885	6900	4258	13	9	0.22
					2	3160	237490	8260	4680	13	11	0.25
					3	3210	241230	8850	4390	13	11	0.28
10	10	+	T	EDD	0	742	19295	3208	2283	13	4	0.25
					1	2330	125170	7030	4140	13	9	0.24
					2	3040	288920	9430	4870	13	11	0.29
					3	2420	29760	7030	3750	13	11	0.29
10	10	+	T	SPT	0	1245	47336	3922	2910	13	5	0.25
					1	2590	27930	6320	3880	13	11	0.26
					2	2140	243550	5830	3720	13	11	0.27
					3	2120	278930	6480	3970	13	11	0.28
10	10	+	T	LPT	0	1990	84743	5381	3862	13	6	0.25
					1	3370	189080	7040	4840	13	10	0.26
					2	2210	387480	5540	4040	13	11	0.27
					3	1920	318700	4630	3580	13	10	0.28
10	10	+	T	SL	0	673	28424	2646	2308	13	5	0.30
					1	1983	94133	6343	3368	13	9	0.32
					2	1850	101880	6340	3680	13	11	0.32
					3	1200	202550	4520	3030	13	11	0.35
10	10	+	T	CR	0	373	22287	2504	2150	13	6	0.29
					1	2502	95103	5129	4326	13	9	0.32
					2	1250	106820	3920	3070	13	11	0.35
					3	1238	97060	3683	3070	13	11	0.36
10	10	+	L	GRASP	0	641	67223	3442	2662	13	10	191.48
					1	435	62862	3189	2495	13	11	201.24
					2	337	60368	3192	2302	13	11	205.63
					3	216	68223	2714	2195	13	11	208.42
10	10	+	L	FIFO	0	1566	3	3605	2464	13	3	0.21
					1	3303	3771	8887	5026	13	11	0.22
					2	2480	191700	8260	4540	13	11	0.25
					3	2210	215400	6542	3520	13	12	0.28
10	10	+	L	EDD	0	1352	47752	5584	3199	13	5	0.25
					1	3180	150350	10180	5240	13	11	0.24
					2	3370	299050	10660	5430	13	11	0.29
					3	2890	306020	9700	4950	13	11	0.29
10	10	+	L	SPT	0	1503	59144	4882	2887	13	8	0.25
					1	2750	184780	7410	4810	13	11	0.26
					2	3020	302190	7990	4800	13	11	0.27
					3	2230	284340	6450	4290	13	11	0.28
10	10	+	L	LPT	0	1632	44822	3678	3208	13	5	0.25
					1	3610	228390	8250	5670	13	11	0.26
					2	2240	259690	5520	4030	13	11	0.27
					3	1970	259690	5520	4030	13	11	0.28
10	10	+	L	SL	0	898	36812	3545	2853	13	5	0.30
					1	2460	12700	7510	4520	13	11	0.32
					2	1540	136880	4900	3600	13	11	0.32
					3	1210	150770	4630	3270	13	11	0.35
10	10	+	L	CR	0	1604	71501	4421	3664	13	11	0.29
					1	2480	188760	6400	4540	13	11	0.32
					2	1530	165480	4660	3580	13	11	0.35
					3	1070	128420	3920	3130	13	11	0.36

Table B3
The results of GRASP and dispatching rules for 15 × 15 problems.

Jobs	Machines	Utilization	Due date tightness	Methods	Flexibility level	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
15	15	–	T	GRASP	0	810	384590	4730	3520	20	18	325.05
					1	340	211260	4010	3060	20	19	351.49
					2	300	272860	4170	3040	20	19	362.14
					3	260	236910	4250	3060	20	19	365.28
15	15	–	T	FIFO	0	1101	8	3926	2748	20	3	2.11
					1	6700	1075800	16400	8700	20	19	2.21
					2	5500	1346200	14600	7400	20	19	2.35
					3	5500	1863100	15500	7100	20	19	2.48
15	15	–	T	EDD	0	2160	153930	7280	4780	20	6	2.22
					1	5800	1331100	16900	8600	20	19	2.22
					2	5100	1600400	14500	7100	20	19	2.31
					3	5000	1786800	15000	7700	20	19	2.35
15	15	–	T	SPT	0	3450	593410	9310	6330	20	13	2.15
					1	4900	2671700	12300	7500	20	19	2.25
					2	5600	2596200	14100	8400	20	19	2.21
					3	3400	1672900	10300	5900	20	19	2.38
15	15	–	T	LPT	0	1894	52871	5931	4211	20	7	2.11
					1	3900	1604800	9700	6800	20	19	2.31
					2	2800	1440300	7300	5400	20	19	2.45
					3	2300	1526900	7000	5200	20	19	2.48
15	15	–	T	SL	0	980	139540	5370	3690	20	14	2.21
					1	3030	955430	10480	5910	20	19	2.25
					2	730	503320	5060	3610	20	19	2.37
					3	690	338650	4650	3570	20	19	2.48
15	15	–	T	CR	0	940	110340	3990	3690	20	16	2.24
					1	3900	1021300	8600	6700	20	19	2.38
					2	1480	361940	5110	4760	20	19	2.41
					3	1080	515090	4750	3960	20	19	2.49
15	15	–	L	GRASP	0	650	412050	4760	3710	20	20	325.05
					1	370	260710	4590	3500	20	19	351.49
					2	350	226870	4200	3270	20	19	362.14
					3	320	245340	4250	3410	20	19	365.28
15	15	–	L	FIFO	0	2019	18742	7071	3801	20	6	2.11
					1	8000	1137600	18200	9400	20	19	2.21
					2	5200	1723800	13800	6900	20	19	2.35
					3	4700	1874800	13700	6900	20	19	2.48
15	15	–	L	EDD	0	2890	336040	9750	5850	20	8	2.22
					1	5300	117300	16700	8600	20	19	2.22
					2	5000	1413700	15000	7600	20	19	2.31
					3	4900	1732600	14600	7200	20	19	2.35
15	15	–	L	SPT	0	5640	918840	12560	8500	20	17	2.15
					1	3700	1810200	10800	6500	20	19	2.25
					2	5400	2198400	15100	8000	20	19	2.21
					3	4100	2553100	12800	6900	20	19	2.38
15	15	–	L	LPT	0	2540	204990	7090	5390	20	7	2.11
					1	3300	1395100	8600	6400	20	19	2.31
					2	3100	1785200	8100	6400	20	19	2.45
					3	3000	1796700	9200	6300	20	19	2.48
15	15	–	L	SL	0	1240	158510	5620	4390	20	15	2.21
					1	2850	652160	10970	6130	20	19	2.25
					2	1270	368890	5930	4560	20	19	2.37
					3	1040	338830	5590	4320	20	19	2.48
15	15	–	L	CR	0	820	211000	4390	4030	20	18	2.24
					1	3880	786810	8880	6760	20	19	2.38
					2	1340	505560	4740	4220	20	19	2.41
					3	1010	44509	4860	4290	20	19	2.49

(*continued on next page*)

Table B3 (*continued*)

Jobs	Machines	Utilization	Due date tightness	Methods	Flexibility level	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
15	15	+	T	GRASP	0	1180	291040	5070	3860	20	14	325.05
					1	730	676920	4570	3540	20	18	351.49
					2	610	620390	4170	3420	20	19	362.14
					3	550	547830	4150	3420	20	19	365.28
15	15	+	T	FIFO	0	2983	1744	8333	5263	20	5	2.11
					1	8480	967010	19460	10810	20	16	2.21
					2	8400	1766900	20400	98000	20	19	2.35
					3	7500	2980700	1900	8600	20	19	2.48
15	15	+	T	EDD	0	2780	161380	8390	5470	20	5	2.22
					1	9700	1993900	22300	12000	20	19	2.22
					2	8300	3097600	21900	11200	20	19	2.31
					3	6600	2697300	18900	8700	20	19	2.35
15	15	+	T	SPT	0	2250	184110	8240	5190	20	8	2.15
					1	8600	4137100	18800	11000	20	19	2.25
					2	5400	3526100	15600	8300	20	19	2.21
					3	3100	329500	9900	5600	20	19	2.38
15	15	+	T	LPT	0	2380	105430	6570	5320	20	6	2.11
					1	5600	2416300	12700	8500	20	19	2.31
					2	5000	2683200	11300	7600	20	19	2.45
					3	2500	2856400	8300	5400	20	19	2.48
15	15	+	T	SL	0	192	33755	3132	2839	20	8	2.21
					1	3700	1165000	13100	6600	20	18	2.25
					2	1820	731660	6570	4700	20	19	2.37
					3	1120	645570	5540	4000	20	19	2.48
15	15	+	T	CR	0	145	7363	2754	2685	20	6	2.24
					1	5700	1643900	11900	8600	20	18	2.38
					2	2000	645870	5610	5290	20	19	2.41
					3	1820	904240	5830	5100	20	19	2.49
15	15	+	L	GRASP	0	990	255240	5010	4070	20	15	325.05
					1	700	430340	4660	3880	20	19	351.49
					2	630	438730	4900	3760	20	19	362.14
					3	540	533430	4900	3700	20	19	365.28
15	15	+	L	FIFO	0	2673	1735	8333	5263	20	5	2.11
					1	9100	1712200	21400	11400	20	19	2.21
					2	10400	2424700	24600	11900	20	19	2.35
					3	7000	3414500	18500	9500	20	19	2.48
15	15	+	L	EDD	0	2390	162790	8390	5100	20	6	2.22
					1	8700	1857400	20500	11400	20	19	2.22
					2	6800	2352000	17900	9300	20	19	2.31
					3	6600	2697300	18900	8700	20	19	2.35
15	15	+	L	SPT	0	2690	443400	9290	6000	20	9	2.15
					1	7100	3443200	16400	10400	20	19	2.25
					2	5000	3827500	15700	8000	20	19	2.21
					3	4300	3409500	11900	6900	20	19	2.38
15	15	+	L	LPT	0	2540	108930	7040	5340	20	5	2.11
					1	5200	1796600	11900	8200	20	19	2.31
					2	4800	2462200	10700	8100	20	19	2.45
					3	4400	2349100	10400	7700	20	19	2.48
15	15	+	L	SL	0	876	88836	5138	4037	20	12	2.21
					1	5000	1170100	14100	8200	20	19	2.25
					2	1840	570560	6820	5120	20	19	2.37
					3	1400	805480	6480	4680	20	19	2.48
15	15	+	L	CR	0	1040	129190	4480	4210	20	12	2.24
					1	5100	152300	10700	8000	20	18	2.38
					2	1980	804960	5440	4860	20	19	2.41
					3	1580	636320	5100	4460	20	19	2.49

Table B4
The results of different rescheduling policy for 5 × 5 problems.

Jobs	Machines	Utilization	Due date tightness	GRASP(event driven, periodic:120, 180,240)	Flexibility level[a]	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
5	5	–	T	Event driven	0	81	5103	814	557	7	6	90.01
					1	34	5302	844	622	7	6	114.12
					2	0	4018	719	513	7	5	129.45
					3	0	3583	747	545	7	5	130.15
5	5	–	T	120	0	115	7850	866	677	7	6	30.48
					1	110	4259	774	576	7	5	35.49
					2	90	1708	773	592	7	5	37.12
					3	90	474	835	536	7	6	39.85
5	5	–	T	180	0	132	10781	848	658	7	6	25.15
					1	73	60	774	517	7	5	26.25
					2	71	47	772	445	7	5	27.31
					3	103	1032	899	728	7	6	28.19
5	5	–	T	240	0	40	17412	906	588	7	6	20.21
					1	65	14525	827	609	7	6	21.58
					2	50	6164	911	607	7	5	23.93
					3	42	9053	908	600	7	5	24.47
5	5	–	L	Event driven	0	47	3098	814	651	7	6	90.01
					1	19	2758	825	650	7	6	114.12
					2	0	8300	845	516	7	6	129.45
					3	0	4672	845	563	7	6	130.15
5	5	–	L	120	0	145	1753	926	790	7	6	30.48
					1	86	1868	864	669	7	5	35.49
					2	58	600	838	669	7	6	37.12
					3	42	1163	884	606	7	6	39.85
5	5	–	L	180	0	126	3589	1052	726	7	6	25.15
					1	64	6519	895	606	7	6	26.25
					2	63	55	895	558	7	6	27.31
					3	60	5210	899	600	7	6	28.19
5	5	–	L	240	0	93	8223	880	633	7	6	20.21
					1	72	47	878	547	7	5	21.58
					2	100	548	885	674	7	6	23.93
					3	18	999	873	600	7	6	24.47
5	5	+	T	Event driven	0	135	7557	802	695	7	5	90.01
					1	71	4012	777	642	7	6	114.12
					2	6	5858	787	553	7	5	129.45
					3	2	5884	802	537	7	5	130.15
5	5	+	T	120	0	129	9225	802	635	7	5	30.48
					1	86	8430	914	586	7	6	35.49
					2	97	1324	672	540	7	6	37.12
					3	79	208	780	550	7	6	39.85
5	5	+	T	180	0	177	20751	1051	722	7	5	25.15
					1	100	3961	676	577	7	6	26.25
					2	78	9863	837	656	7	3	27.31
					3	70	37837	813	593	7	5	28.19
5	5	+	T	240	0	97	10943	1051	650	7	5	20.21
					1	94	3272	788	667	7	5	21.58
					2	28	9375	828	572	7	5	23.93
					3	22	4096	781	565	7	6	24.47
5	5	+	L	Event driven	0	106	8619	802	695	7	5	90.01
					1	66	4761	829	719	7	6	114.12
					2	33	517	571	462	7	3	129.45
					3	11	1341	817	505	7	4	130.15
5	5	+	L	120	0	251	10064	1051	883	7	5	30.48
					1	120	3125	831	703	7	5	35.49
					2	119	32769	1024	763	7	6	37.12
					3	105	6756	809	653	7	6	39.85
5	5	+	L	180	0	154	25803	816	641	7	5	25.15
					1	114	648	1052	738	7	6	26.25
					2	127	0	987	666	7	6	27.31
					3	65	2079	988	665	7	6	28.19
5	5	+	L	240	0	133	11051	1153	754	7	5	20.21
					1	89	47	988	654	7	6	21.58
					2	81	25	990	600	7	6	23.93
					3	0	4785	800	592	7	6	24.47

[a] 0: Static problem, 1: Small flexibility level, 2: Medium flexibility level, 3: Large flexibility level.

Table B5
The results of different rescheduling policy for 10 × 10 problems.

Jobs	Machines	Utilization	Due date tightness	GRASP(event driven, periodic:120, 180,240)	Flexibility level	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
10	10	–	T	Event driven	0	399	65973	2981	2006	13	12	191.48
					1	256	43663	2719	1867	13	11	201.24
					2	111	54287	2501	1799	13	12	205.63
					3	81	78965	2356	1640	13	12	208.42
10	10	–	T	120	0	374	57607	2723	1948	13	10	110.45
					1	149	382	1784	1511	13	12	115.21
					2	44	2939	1884	1529	13	12	125.36
					3	0	131	1756	1443	13	12	126.78
10	10	–	T	180	0	475	60764	6065	2109	13	12	105.62
					1	51	718	1784	1414	13	12	108.38
					2	23	34	1779	1395	13	12	110.24
					3	4	241	1786	1390	13	12	111.67
10	10	–	T	240	0	580	155590	3090	2220	13	12	101.69
					1	98	554	1786	1400	13	12	105.45
					2	79	1006	2009	1529	13	11	107.24
					3	61	699	1880	1451	13	11	108.31
10	10	–	L	Event driven	0	309	65347	3195	2126	13	12	191.48
					1	188	35071	2701	2109	13	11	201.24
					2	177	38290	2612	2008	13	12	205.63
					3	128	57343	2480	1917	13	12	208.42
10	10	–	L	120	0	429	86660	2985	2199	13	11	110.45
					1	156	39360	2162	1841	13	12	115.21
					2	17	5387	2159	1709	13	11	125.36
					3	16	1416	2108	1488	13	12	126.78
10	10	–	L	180	0	526	82639	3253	2288	13	12	105.62
					1	65	804	2164	1512	13	12	108.38
					2	20	337	2159	1482	13	12	110.24
					3	17	452	2159	1571	13	12	111.67
10	10	–	L	240	0	587	49166	3090	2402	13	12	101.69
					1	67	165	2269	1502	13	12	105.45
					2	66	207	2260	1513	13	12	107.24
					3	48	1237	2107	1550	13	12	108.31
10	10	+	T	Event driven	0	693	47168	3065	2493	13	11	191.48
					1	455	67567	2981	2288	13	11	201.24
					2	301	84278	2720	2092	13	11	205.63
					3	202	85685	2981	1924	13	11	208.42
10	10	+	T	120	0	747	63990	3465	2312	13	10	110.45
					1	117	16152	2026	1786	13	11	115.21
					2	118	12598	2021	1608	13	11	125.36
					3	94	57	2024	1806	13	11	126.78
10	10	+	T	180	0	850	125841	3850	2541	13	11	105.62
					1	128	6025	2087	1606	13	11	108.38
					2	110	6617	2021	1621	13	11	110.24
					3	96	216	2021	1661	13	11	111.67
10	10	+	T	240	0	990	148530	3680	2630	13	12	101.69
					1	281	3696	2130	1838	13	11	105.45
					2	204	3734	2024	1764	13	11	107.24
					3	156	2799	2088	1765	13	11	108.31
10	10	+	L	Event driven	0	641	67223	3442	2662	13	10	191.48
					1	435	62862	3189	2495	13	11	201.24
					2	337	60368	3192	2302	13	11	205.63
					3	216	68223	2714	2195	13	11	208.42
10	10	+	L	120	0	628	56291	3574	2419	13	11	110.45
					1	120	176290	2280	1930	13	11	115.21
					2	89	2699	2141	1808	13	11	125.36
					3	63	3059	2267	1782	13	11	126.78
10	10	+	L	180	0	910	136280	3920	2760	13	12	105.62
					1	107	389	2163	1782	13	11	108.38
					2	74	3351	2108	1753	13	11	110.24
					3	64	411	2087	1730	13	11	111.67
10	10	+	L	240	0	977	74929	3671	2824	13	12	101.69
					1	216	394	2244	1785	13	11	105.45
					2	116	6462	2271	1690	13	11	107.24
					3	85	6983	2276	1769	13	11	108.31

Table B6
The results of different rescheduling policy for 15 × 15 problems.

Jobs	Machines	Utilization	Due date tightness	GRASP(event driven, periodic:120, 180,240	Flexibility level	Mean Tardiness	Instability	Makespan	Mean Flow Time	Number of Orders	Number of accepted orders	CPU Time
15	15	–	T	Event driven	0	810	384590	4730	3520	20	18	325.05
					1	340	211260	4010	3060	20	19	351.49
					2	290	295930	4010	3050	20	19	362.14
					3	260	236910	4250	3060	20	19	365.28
15	15	–	T	120	0	1020	453650	4870	3630	20	19	251.31
					1	79	6572	2770	2289	20	19	265.38
					2	3	1571	2661	2191	20	18	272.52
					3	32	6241	2844	2259	20	19	281.96
15	15	–	T	180	0	960	155770	4660	3600	20	19	247.31
					1	73	14104	2825	2280	20	19	251.25
					2	72	20755	2746	2214	20	19	253.48
					3	71	29655	2843	2253	20	19	261.40
15	15	–	T	240	0	1220	187780	4880	3890	20	19	220.73
					1	57	4111	2775	2315	20	19	221.46
					2	28	3917	2771	2318	20	19	235.46
					3	27	29100	2760	2345	20	19	236.38
15	15	–	L	Event driven	0	650	412050	4760	3710	20	20	325.05
					1	370	260710	4590	3500	20	19	351.49
					2	350	226870	4200	3270	20	19	362.14
					3	320	245340	4250	3410	20	19	365.28
15	15	–	L	120	0	950	100720	4870	4000	20	18	251.31
					1	58	2944	3250	2490	20	19	265.38
					2	31	751	3056	2399	20	19	272.52
					3	31	782	3241	2445	20	19	281.96
15	15	–	L	180	0	640	100992	4560	3650	20	19	247.31
					1	104	33311	3251	2472	20	19	251.25
					2	89	46372	3336	2400	20	19	253.48
					3	62	45345	3332	2408	20	19	261.40
15	15	–	L	240	0	690	217660	4490	3720	20	19	220.73
					1	106	36613	3249	2542	20	19	221.46
					2	39	42734	3251	2505	20	19	235.46
					3	37	46881	3244	2501	20	19	236.38
15	15	+	T	Event driven	0	1180	291040	5070	3860	20	14	325.05
					1	730	676920	4570	3540	20	18	351.49
					2	610	620390	4170	3420	20	19	362.14
					3	550	547830	4150	3420	20	19	365.28
15	15	+	T	120	0	1590	249030	5550	4340	20	17	251.31
					1	113	2094	2765	2513	20	19	265.38
					2	92	5887	2762	2491	20	19	272.52
					3	76	1426	2770	2512	20	19	281.96
15	15	+	T	180	0	1540	481210	5360	4350	20	18	247.31
					1	87	8075	2765	2414	20	19	251.25
					2	81	8120	2946	2424	20	19	253.48
					3	78	8410	2945	2480	20	19	261.40
15	15	+	T	240	0	1810	198790	5940	4570	20	18	220.73
					1	111	3903	3029	2519	20	19	221.46
					2	105	2654	2986	2634	20	19	235.46
					3	86	4366	2752	2581	20	19	236.38
15	15	+	L	Event driven	0	990	255240	5010	4070	20	15	325.05
					1	700	430340	4660	3880	20	19	351.49
					2	630	438730	4900	3760	20	19	362.14
					3	540	533430	4900	3700	20	19	365.28
15	15	+	L	120	0	1480	212200	5920	4500	20	18	251.31
					1	134	4372	3429	2702	20	19	265.38
					2	98	2184	3244	2668	20	19	272.52
					3	97	1471	3245	2664	20	19	281.96
15	15	+	L	180	0	1310	163670	5610	4470	20	18	247.31
					1	149	30676	3429	2666	20	19	251.25
					2	121	41833	3251	2619	20	19	253.48
					3	118	14627	3240	2596	20	19	261.30
15	15	+	L	240	0	1400	198140	5910	4550	20	18	220.73
					1	194	37757	3435	2833	20	19	221.46
					2	191	22673	3255	2789	20	19	235.46
					3	140	22705	3246	2792	20	19	236.38

References

[1] Pinedo ML, Chao X. Operations scheduling with applications in manufacturing and service. Singapore: McGraw Hill; 1999.
[2] Ouelhadj D, Petrovic S. A survey of dynamic scheduling in manufacturing systems. J Sched 2009;12(4):417–31.
[3] Fang J, Xi Y. A rolling horizon job shop rescheduling strategy in the dynamic environment. Int J Adv Manuf Technol 1997;13:227–32.
[4] Ozguven C, Ozbakir L, Yavuz Y. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 2010;34(6):1539–48.
[5] Baykasoğlu A. Linguistic based meta-heuristic optimisation model for flexible job shop scheduling. Int J Prod Res 2002;40(17):4523–43.
[6] Sabuncuoglu I, Hommertzheim DL. Dynamic dispatching algorithm for scheduling

machines and automated guided vehicles in a flexible manufacturing system. Int J Prod Res 1992;30(5):1059–79.
[7] Melnyk SA, Ragatz GL. Order review/release and its impact on the shop floor. Production and Inventory Management 1988;29:13–7.
[8] Zäpfel G, Missbauer H. New concepts for production planning and control. Eur J Oper Res 1993;67:297–320.
[9] Baykasoğlu A, Ozsoydan FB. Evolutionary and population-based methods versus constructive search strategies in dynamic combinatorial optimization. Inf Sci (Ny) 2017;420:159–83.
[10] Feo TA, Resende MG. Greedy randomized adaptive search procedures. J Glob Optim 1995;6(2):109–33.
[11] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 1993;41(3):157–83.
[12] Baykasoğlu A, Ozbakir L, Sönmez AI. Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems. J Intell Manuf 2004;15(6):777–85.
[13] Pezzella F, Morganti G, Ciaschetti G. A genetic algorithm for the flexible job-shop scheduling problem. Comput Oper Res 2008;35(10):3202–12.
[14] Zhang G, Gao L, Shi Y. An effective genetic algorithm for the flexible job-shop scheduling problem. Expert Syst Appl 2011;38(4):3563–73.
[15] Shahsavari-Pour N, Ghasemishabankareh B. A novel hybrid meta-heuristic algorithm for solving multi objective flexible job shop scheduling. J Manuf Syst 2013;32(4):771–80.
[16] Bagheri A, Zandieh M. Bi-criteria flexible job-shop scheduling with sequence-dependent setup times – variable neighborhood search approach. J Manuf Syst 2011;30:8–15.
[17] Gao J, Sun L, Gen M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput Oper Res 2008;35(9):2892–907.
[18] Bagheri A, Zandieh M, Mahdavi I, Yazdani M. An artificial immune algorithm for the flexible job-shop scheduling problem. Future Gener Comput Syst 2010;26(4):533–41.
[19] Rajkumar M, Asokan P, Vamsikrishna VA. GRASP algorithm for flexible job-shop scheduling with maintenance constraints. Int J Prod Res 2010;48(22):6821–36.
[20] Rajkumar M, Asokan P, Anilkumar N, Page T. A GRASP algorithm for flexible job-shop scheduling problem with limited resource constraints. Int J Prod Res 2011;49(8):2409–23.
[21] Kemmoé-Tchomté S, Lamy D, Tchernev N. An effective multi-start multi-level evolutionary local search for the flexible job-shop problem. Engineering Applications of Artificial Intelligence 2017;62:80–95.
[22] Xu Y, Wang L, Wang SY, Liu M. An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time. Neurocomputing 2015;148:260–8.
[23] Gao KZ, Suganthan PN, Pan QK, Chua TJ, Cai TX, Chong CS. Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. J Intell Manuf 2016;27(2):363–74.
[24] Gao KZ, Suganthan PN, Pan QK, Tasgetiren MF, Sadollah A. Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion. Knowledge Based Syst 2016;109:1–16.
[25] Singh MR, Mahapatra SS. A quantum behaved particle swarm optimization for flexible job shop scheduling. Comput Ind Eng 2016;93:36–44.
[26] Vital-Soto A, Azab A, Baki MF. Mathematical modeling and a hybridized bacterial foraging optimization algorithm for the flexible job-shop scheduling problem with sequencing flexibility. J Manuf Syst 2020;54:74–93.
[27] Gao K, Cao Z, Zhang L, Chen Z, Han Y, Pan Q. A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. Ieee/caa J Autom Sin 2019;6(4):904–16.
[28] Kim MH, Kim YD. Simulation-based real-time scheduling in a flexible manufacturing system. J Manuf Syst 1994;13(2):85–93.
[29] Branke J, Mattfeld DC. Anticipation and flexibility in dynamic scheduling. Int J Prod Res 2005;43(15):3103–29.
[30] Buyurgan N, Saygin C. Application of the analytical hierarchy process for real-time scheduling and part routing in advanced manufacturing systems. J Manuf Syst 2008;27(3):101–10.
[31] Gholami M, Zandieh M. Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. J Intell Manuf 2009;20(4):481–98.
[32] Fattahi P, Fallahi A. Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability. CIRP J. of Manufacturing Science and Technology 2010;2(2):114–23.
[33] Al-Hinai N, ElMekkawy TY. Robust and stable flexible job shop schedulingwith random machine breakdowns using a hybrid genetic algorithm. Int J Prod Econ 2011;132(2):279–91.
[34] Rajabinasab A, Mansour S. Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach. Int J Adv Manuf Technol 2011;54(9-12):1091–107.
[35] Lin L, Gen M, Liang Y, Ohno K. A hybrid EA for reactive flexible job-shop scheduling. Procedia Comput Sci 2012;12:110–5.
[36] He W, Sun DH. Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies. Int J Adv Manuf Technol 2013;66(1-4):501–14.
[37] Nie L, Gao L, Li P, Li X. A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. J Intell Manuf 2013;24(4):763–74.
[38] Negahban A, Smith JS. Simulation for manufacturing system design and operation: literature review and analysis. J Manuf Syst 2014;33(2):241–61.
[39] Shen XN, Yao X. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. Inf Sci (Ny) 2015;298:198–224.
[40] Ning T, Huang M, Liang X, Jin H. A novel dynamic scheduling strategy for solving flexible job-shop problems. J Ambient Intell Humaniz Comput 2016:1–9.
[41] Jin L, Zhang C, Shao X, Yang X. A study on the impact of periodic and event-driven rescheduling on a manufacturing system: an integrated process planning and scheduling case. Proc Inst Mech Eng Part B J Eng Manuf 2017;231(3):490–504.
[42] Gao K, Yang F, Zhou M, Pan Q, Suganthan PN. Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm. IEEE Trans Cybern 2018;49(5):1944–55.
[43] Gao K, Wang L, Luo J, Jiang H, Sadollah A, Pan Q. Discrete harmony search algorithm for scheduling and rescheduling the reprocessing problems in remanufacturing: a case study. Eng Optim 2018;50(6):965–81.
[44] Uhlmann IR, Frazzon EM. Production rescheduling review: opportunities for industrial integration and practical applications. J Manuf Syst 2018;49:186–93.
[45] Zadeh MS, Katebi Y, Doniavi A. A heuristic model for dynamic flexible job shop scheduling problem considering variable processing times. Int J Prod Res 2019;57(10):3020–35.
[46] Zhou Y, Yang JJ, Huang Z. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. Int J Prod Res 2019:1–20.
[47] Cao Z, Zhou L, Hu B, Lin C. An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem. Bus Inf Syst Eng 2019;61(3):299–309.
[48] Ozturk G, Bahadir O, Teymourifar A. Extracting priority rules for dynamic multi-objective flexible job shop scheduling problems using gene expression programming. Int J Prod Res 2019;57(10):3121–37.
[49] Hu L, Liu Z, Hu W, Wang Y, Tan J, Wu F. Petri-net-based dynamic scheduling of flexible manufacturing system via deep reinforcement learning with graph convolutional network. J Manuf Syst 2020;55:1–14.
[50] Luo S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. Appl Soft Comput 2020;91:106208.
[51] Giffler B, Thompson GL. Algorithms for solving production-scheduling problems. Oper Res 1960;8(4):487–503.
[52] Baykasoğlu A, Karaslan FS. Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach. Int J Prod Res 2017;55(11):3308–25.
[53] Baykasoğlu A, Göçken M. A simulation based approach to analyse the effects of job release on the performance of a multi-stage job-shop with processing flexibility. Int J Prod Res 2011;49(2):585–610.
[54] Behringer FA. Lexicographic quasiconcave multi-objective programming. Math Methods Oper Res 1977;21(3):103–16.
[55] Baykasoğlu A. Preemptive goal programming using simulated annealing. Eng Optim 2005;37(1):49–63.
[56] Amirthagadeswaran KS, Arunachalam VP. Improved solutions for job shop scheduling problems through genetic algorithm with a different method of schedule deduction. Int J Adv Manuf Technol 2006;28:532–40.
[57] Montgomery DC. Design and analysis of experiments. New York: John Wiley and Sons; 2000.
[58] Kacem I, Hammadi S, Borne P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. Ieee Trans Syst Man Cybern Part C 2002;32(1):1–13.
[59] Dauzère-Pérès S, Paulli J. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann Oper Res 1997;70:281–306.
[60] Xia W, Wu Z. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput Ind Eng 2005;48(2):409–25.
[61] Yazdani M, Amiri M, Zandieh M. Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Syst Appl 2010;37(1):678–87.
[62] Nouiri M, Bekrar A, Jemai A, Niar S, Ammari AC. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. J Intell Manuf 2015:1–13.
[63] Yuan Y, Xu H. An integrated search heuristic for large-scale flexible job shop scheduling problems. Comput Oper Res 2013;40(12):2864–77.
[64] González MA, Vela CR, Varela R. Scatter search with path relinking for the flexible job shop scheduling problem. Eur J Oper Res 2015;245(1):35–45.
[65] Derrac J, García S, Molina D, Herrera F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 2011;1(1):3–18.