

Improved Genetic Algorithm Approach Based on New Virtual Crossover Operators for Dynamic Job Shop Scheduling

KAOUTHER BEN ALI¹, ACHRAF JABEUR TELMOUDI², (Member, IEEE), AND SAID GATTOUFI¹

¹SMART Lab, ISG, University of Tunis, Tunis 2000, Tunisia

²LISIER Laboratory, The National Higher Engineering School of Tunis, University of Tunis, Tunis 1008, Tunisia

Corresponding author: Achraf Jabeur Telmoudi (achraf-j.telmoudi@ieee.org)


ABSTRACT The realtime manufacturing system is subject to different kinds of disruptions such as new job arrivals, machine breakdowns, and jobs cancellation. These different disruptions affect the original schedule that should be updated to maintain the system's performance. An effective re-scheduling is required in this situation to make better utilization of the system resources. This paper studies the dynamic job shop scheduling problem. The problem is known as strongly NP Hard optimization problem where new jobs are unconditionally arrived at the system. Hence, to deal with system changes and performing hard tasks scheduling, we propose an evolutionary genetic algorithm based on virtual crossover operators. Experimental results are compared with state-of-the-art heuristics and metaheuristics dedicated for evaluating large scale instances. Simulation results show the efficiency of the proposed virtual crossover operators integrated into the genetic algorithm approach.

INDEX TERMS Dynamic job shop scheduling problem, genetic algorithm, crossover operators, makespan, dispatching rules, metaheuristics.

I. INTRODUCTION

Scheduling is a scientific domain concerning the allocation of tasks (e.g., jobs, services) to resources (e.g., machines). The objective of a scheduling problem is to maximize or minimize a desired objective in an optimal manner to perform measures such as the makespan, the total tardiness, the cost, etc. Operation research and artificial intelligence fields are the most referred to the literature of scheduling problems. Their goal is to solve the question of how to move forward in the production process that is considered the basic step of the process. Scheduling can be viewed as two types: static and dynamic. Static scheduling (called as deterministic) constructs a complete schedule of tasks on candidate machines through a well known data. However, dynamic scheduling considers tasks arrival one by one during the system execution.

The production scheduling encounters many challenges due to production environment changes that are considered

The associate editor coordinating the review of this manuscript and approving it for publication was Md Asaduzzaman .

dynamic in nature (job arrivals, job cancellation, machine breakdowns, etc.) and cause a high degree of complexity during the scheduling. Hence, the industrial process must be optimized in terms of different performance criteria. An illustrative workflow of scheduling in the manufacturing system is presented in Fig. 1. The manufacturing system starts processing after the arrival of production orders. An effective schedule is required to maximize production effectiveness and gain profits. The most frequent industrial objective is the minimization of the makespan C_{max} ; well known as the completion time of the last job at the system.

Different scheduling problems are classified based on their complexity in different production lines: flow shop, job shop, and open shop, etc. Obviously, the job shop is one of the most complicated NP-Hard combinatorial optimization problems. The goal of the job system is to find a processing order of a set of textitn jobs on textitn machines where different jobs may have a separate processing sequence. Each operation will be processed on a candidate machine during a fixed processing time [1], [2]. The interest in a real-time system with unexpected events is of great importance where the

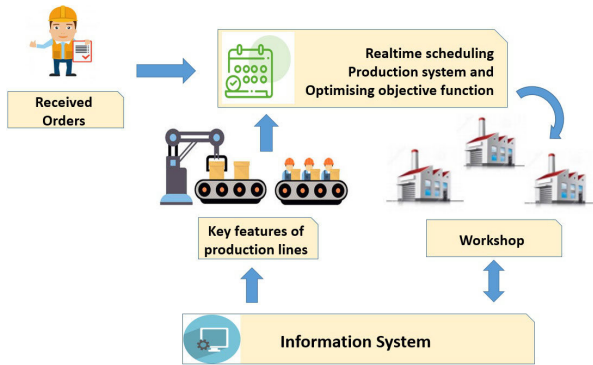


FIGURE 1. An illustrative workflow of the manufacturing systems.

system should be able to manage dynamic events with different perturbations. Unexpected events such as machine failure, new job arrivals are hard tasks that must be well processed to perform the final schedule. The frequency of dynamic events makes the Job Shop Scheduling Problem (JSSP) subjects of several studies and researches in the production management and the combinatorial optimization areas. Among different categories of JSSPs, Dynamic JSSP is classified as the most complicated subclass. It is considered a strongly NP-Hard problem [3]–[5].

During the scheduling of a set of tasks with the presence of frequent changes, the original schedule may become infeasible and needs to be updated. This kind of problem is known as dynamic scheduling. Dynamic JSSP has attracted much attention in the field of computer science, system engineering, and operations research. Well-known algorithms and metaheuristics such as genetic algorithm, particle swarm optimization, tabu search, artificial bee colony algorithm, and ant colony optimization are dedicated to solving it. It has taken the interest of a considered number of researchers in the literature. The first study in the Dynamic JSSP was published by Holloway and Nelson in 1974 where they developed a multipass heuristic scheduling procedure for the JSSP with due dates and variable processing time. A survey of the Dynamic JSSP has been developed in [6]. Many different methods have been developed to solve the Dynamic JSSP including heuristics, metaheuristics, knowledge-based system, fuzzy logic, neural network, Petri nets, hybrid techniques, and multi-agent systems, etc. In the state of the art, diverse techniques are developed through different publications mentioned in the following:

- Artificial intelligence algorithms: [7]–[11].
- Artificial immune system: [12]
- Evolutionary algorithm approach and Genetic programming: [4], [13]–[15], [35]
- Memetic algorithm: [16]–[18]
- Petri nets: [19], [20]
- Tabu search: [17], [21], [22]
- Hybrid approaches: [4], [9]
- Dispatching rules: [34], [36]

Several approaches inspired by the above-presented methods are developed to maximize the performance of scheduling

problems in considered systems. In the following, we will focus our interest on recent optimization techniques applied in the literature to solve the Dynamic JSSP. The contributed work in [23] has studied the problem of Dynamic JSSP by re-scheduling randomly job arrivals with the performance of three sub-objectives: the discontinuity rate of job arrivals, the deviation of the makespan from its initial solution, and the deviation of an updated sequence of machines. To solve the problem, the authors have proposed an updated version of the PSO (Particle Swarm Optimization) algorithm. The proposed algorithm is developed based on the following improvements: (i) modification of decoding scheme, (ii) adaptation of a new approach for population initialization, and, (iii) proposition of a novel particle movement method. Through extensive simulations with generated re-scheduling instances, the proposed modified PSO has shown its efficiency. It has given significantly better results than the three compared metaheuristics selected from the state-of-the-art.

Moreover, in the work of Zhou *et al.* cited in [24], they have proposed effective scheduling policies (SPs) generated through off-line learning and implemented on online evolved SPs for fast application. They have proposed three types of hyper-heuristic methods for the co-evolution of the machine assignment and job sequencing rules to solve the multi-objective Dynamic JSSP. The used methods are genetic programming with two sub-populations, genetic programming with two sub-trees, and genetic expression programming with two chromosomes. Experimental results showed the superiority of the proposed method called (CCGP-NSGA II) in terms of efficiency and competitiveness; comparing against other evolutionary approaches. Other studies have presented different inspired hyper-heuristics approaches such as the work of Park *et al.* [3] that have proposed a new genetic programming based hyper-heuristic. The proposed approach is based on a systematic analysis of diverse popular combination schemes and decision making of different elements to be evolved by genetic programming. The analysis method has shown a significant influence on the behaviors of generated ensembles through different combinations.

To minimize the makespan in a Dynamic Job System, the authors of the work cited in [18] have developed a hybrid genetic algorithm called the GAKK algorithm. This method consists of combining the new KK heuristic with the GA. The main idea of this approach is to combine the population generated by the KK heuristic with 25% of the initial generated population for the execution of the hybrid algorithm. The provided experimentations have shown efficient results for large sizing problem instances in terms of Cmax minimization and running time. In addition to the previous works, the research cited in [25] has suggested a hybrid genetic algorithm with a tabu search approach for solving the Dynamic JSSP. The problem is investigated under two sub-constraints: machine breakdowns and random job arrival. Both of schedule efficiency and schedule stability are measured to validate the

performance of the proposed method. Compared to well-known heuristics and metaheuristics being introduced in the state-of-the-art, experimental results have demonstrated the effectiveness of the algorithm under the defined sub-constraints based on different shop-floor conditions.

Among many metaheuristics used for solving the Dynamic JSSP, the Greedy Randomised Adaptive Search Procedure (GRASP) has been studied by baykasoglu with different optimization objectives. In [26], Baykasoglu and Karaslan addressed a Dynamic JSSP with machine capacity constraints. Moreover, the GRASP inspired approach is proposed under constraints related to order due dates and sequenes of dependent setup times. The problem is evaluated based on four performance criteria (mean tardiness, makespan, mean flow time, and schedule instability). Simulation results have shown the efficiency and the feasibility of the proposed method to solve the problem in realtime as well as the rescheduling problem. Similar to the studied research works being published in [27] and [28], the authors have proposed a multi-start and constructive search algorithm applied to solve an interesting industrial dynamic job shop floor case called parallel heat treatment furnaces; likewise in [29]. The obtained results have proved the high performance of the proposed method in terms of energy consumption and incomes.

The present study considers a dynamic job shop environment to minimize the makespan. Comparing to the studied works presented in the literature, this paper proposes an approach based on a genetic algorithm with new virtual crossover operators. The new proposal is dedicated to enabling a fast reaction of the genetic operators facing dynamic changes. The remaining of this work is organized as follows: Section 2 describes the used notation and formulation of the developed objective function. Section 3 describes the proposed GA approach based on virtual crossover operators for solving sequencing operations. Computational results and their interpretations are discussed in Section 4. Finally, Section 5 presents the conclusion and lines for future research.

II. NOTATIONS AND PROBLEM FORMULATION

Due to the occurrence of external and/or internal events illustrated in Fig.2, the dynamic scheduling is required to absorb the impact of disruptions(arrival of new jobs, machine breakdowns, etc.) that affect the original schedule. In this context, the high frequency of environment changes involves extremely complicated processing during the system’s execution that needs powerful algorithmic techniques. The current study is addressed to find an effective schedule with the minimum makespan within a dynamic job shop system. Clearly, the dynamic job shop is an extension of the well-known job shop problem where during the execution of n jobs on m machines, new job arrivals may appear randomly at a time t. Each job has one or more operations with a fixed processing time where their relationships can be modeled with precedence constraints. If the actual operations have been processed or being processed, their schedule could be

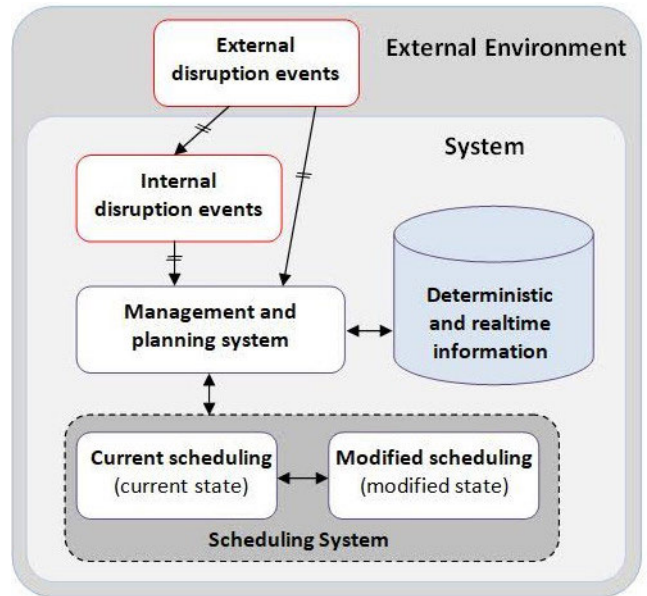


FIGURE 2. Description of the dynamic scheduling environment.

changed with the fact of perturbations. That is why rescheduling is required sometimes to update the schedule. A further explanation of the dynamic job shop is given as a sequence of static job-shop problems. Consequently, the completion time of the whole system is the makespan of the last static sub-problem presented at the system.

The considered assumptions are considered as follows:

- (i) Each machine can perform only one operation of any job at a time.
- (ii) An operation of a job could be performed by only one machine at a time.
- (iii) An operation of a job cannot be performed until its preceding operation is completed.
- (iv) Machines are considered preemptive and activities are interrupted (if any dynamic event occurs).
- (v) A job cannot be cancelled.
- (vi) Machines are available at time zero.

We consider a Dynamic JSSP being formulated as a sequence of static sub-problems. Table 1 defines the used notations in the mathematical model.

Objective Function:

$$\text{Minimize } C_{max} \tag{1}$$

Subject to:

- 1) Constraint on the start time of each operation $O_{i,j}$
- 2) Makespan calculation

$$C_{max} = \max (st_{i,j} + p_{i,j}), \quad \forall i \leq n, j \leq \lambda_i \tag{2}$$

- 3) Precedence constraint between operations

$$st_{i,j} + p_{i,j} \leq st_{i',j'} + p_{i',j'} \tag{3}$$

- 4) Machine capacity constraints (inspired from [30])

$$\zeta[i, j, i', j'] + \zeta[i, j, i', j'] \geq \varepsilon_{ijk} + \varepsilon_{i'jk} - 1 \tag{4}$$

TABLE 1. Description of the used notations.

Notation	Definition
(i) Indices	
J_i	i^{th} job, $1 \leq i \leq n$
O_{ij}	j^{th} operation of the i^{th} job, $1 \leq i \leq n, 1 \leq k \leq \lambda_i$
$O_{i'j'}$	Precedence constraint with O_{ij}
M_k	k^{th} machine, $1 \leq k \leq m$
(ii) Parameters and variables	
λ	denotes the number of operations/Job.
n	Total number of jobs
m	Total number of machines
λ_i	Number of operations at job J_i
Ω_{M_k}	A set of operation proceeded on machine M_k
C_{max}	The makespan value
p_{ij}	The processing time of O_{ij}
$p_{i'j'}$	The processing time of $O_{i'j'}$
(iii) Decision variables	
$\varepsilon_{i,j,k}$	$\xi[i, j; i', j'] = 1; \varepsilon_{i,j,k}=1$; If operation O_{ij} is processed on machine k ; 0 Otherwise
st_{ij}	The starting time of operation O_{ij}
$st_{i'j'}$	The starting time of operation $O_{i'j'}$
$\xi[i, j; i', j']$	$\xi[i, j; i', j'] = 1$; If O_{ij} is processed before $O_{i'j'}$ on k ; 0 Otherwise

with

$$O_{i,j} \cap O_{i',j'} \in \Omega M_k \tag{5}$$

and

$$st_{ij} + p_{ij} - (1 - \zeta[i, j; i', j']).L \leq st_{i'j'} \tag{6}$$

III. GENETIC ALGORITHM BASED VIRTUAL CROSSOVER OPERATORS

To solve the Dynamic JSSP, we present in Fig3 a flow chart that describes the global approach. It starts with an initial job shop problem following the initial data and the parameters. The GA is applied step by step following the processing in Fig.4. Once the schedule is planned for application, the decision process is activated. If no real-time job(s) arrive unconditionally at the system during system execution, the current schedule is considered as the final output. Otherwise, the process of the current state is pre-empted. Therefore, new job(s) are scheduled by respecting the precedence constraints. The current schedule is continued performing until (tx) is finished. Indeed, the new job (s) are added to the rest of the previous schedule. Hence, a new JSSP is initialized based on updated data and, then, the application of the adopted GA is activated. For a finite number of iteration, the same process is applied for achieving the best schedule with a minimum Cmax value.

The efficiency of the GA is based on the recombination process that plays the main role to obtain results with high performance. Genetic operators (selection, crossover, and mutation) are leading the intensification and diversification goals. Each operation that represents a gene of a chromosome is allocated on a candidate machine. The studied objective function is the minimization of the completion time of the last processed operation. Applied to Dynamic JSSP, the methodology consists of creating chromosomes with a number of genes equals to $\sum_{i=1}^n \lambda_i$. After that, each operation is assigned

randomly to a machine and placed on a gene of the chromosome. The objective is to minimize the maximum completion time of all operations.

The flow chart of Fig.4 describes the proposed GA approach used to solve the Dynamic JSSP with new job arrival. The algorithm starts with the initialization of input data and GA parameters for the system execution. The initial population is generated randomly to provide good initial solutions. As well as, to get various chromosomes with the best genes information. The contributed GA approach is applied based on the following main steps (selection, crossover, mutation, and evaluation). For the selection process, the roulette wheel selection operator is applied to choose the best parents for reproduction. For intensification purposes, the crossover operator is applied. The selected parents are combined with random chromosomes to create new offsprings. The proposed crossover operators are called virtual operators; a new shape of crossing between genes information which does not consider the physical nature of crossing by regards to the dynamic characteristic of the job environment. Two kinds of virtual crossing over are proposed based on two points and three-point crossing over. Then, the chosen swap operator is applying on a selected chromosome to perform diversification purposes. The principle of the swap operator consists of switching randomly two genes from the selected chromosomes. After that, the provided solutions will be evaluated to select the best chromosome(s) at the current iteration.

The last step consists of testing the performance of the selected chromosome(s), provided by the current iteration (based on its (their) makespan), vs. the chromosomes with the less makespan value of all previews iterations. If one of the termination criteria is reached, the algorithm will be stopped by sending the best solutions to the output. The termination criteria are: (criterion 1) the makespan remains constant for five successive iterations, or (criterion 2) a finite number C of iterations are reached. Otherwise, the iteration solutions will be stored for trying to improve in the next generations by comparing previous results to the next ones. Hence, to enhance solutions results of the next iteration, 20% of initial the population will be constituted by stored solutions in previous iterations and the rest (80%) by randomly generated population.

The recombination process plays a crucial role in the performance of the evolutionary GA. According to a widely accepted representation for crossover operators, information already existing in the parent solutions should be recombined without introducing new information. For the Dynamic JSSP problem, the precedence relations of operation to other operations projected for the same machine is of great importance. This information should be passed to produce competitive offspring. In this process, crossover operator plays an important role for intensification purpose. To comprehend the GA process, it is necessary to understand the role of the crossover operator.

In the following subsections, four new crossover operators will be described. We are going to propose new virtual

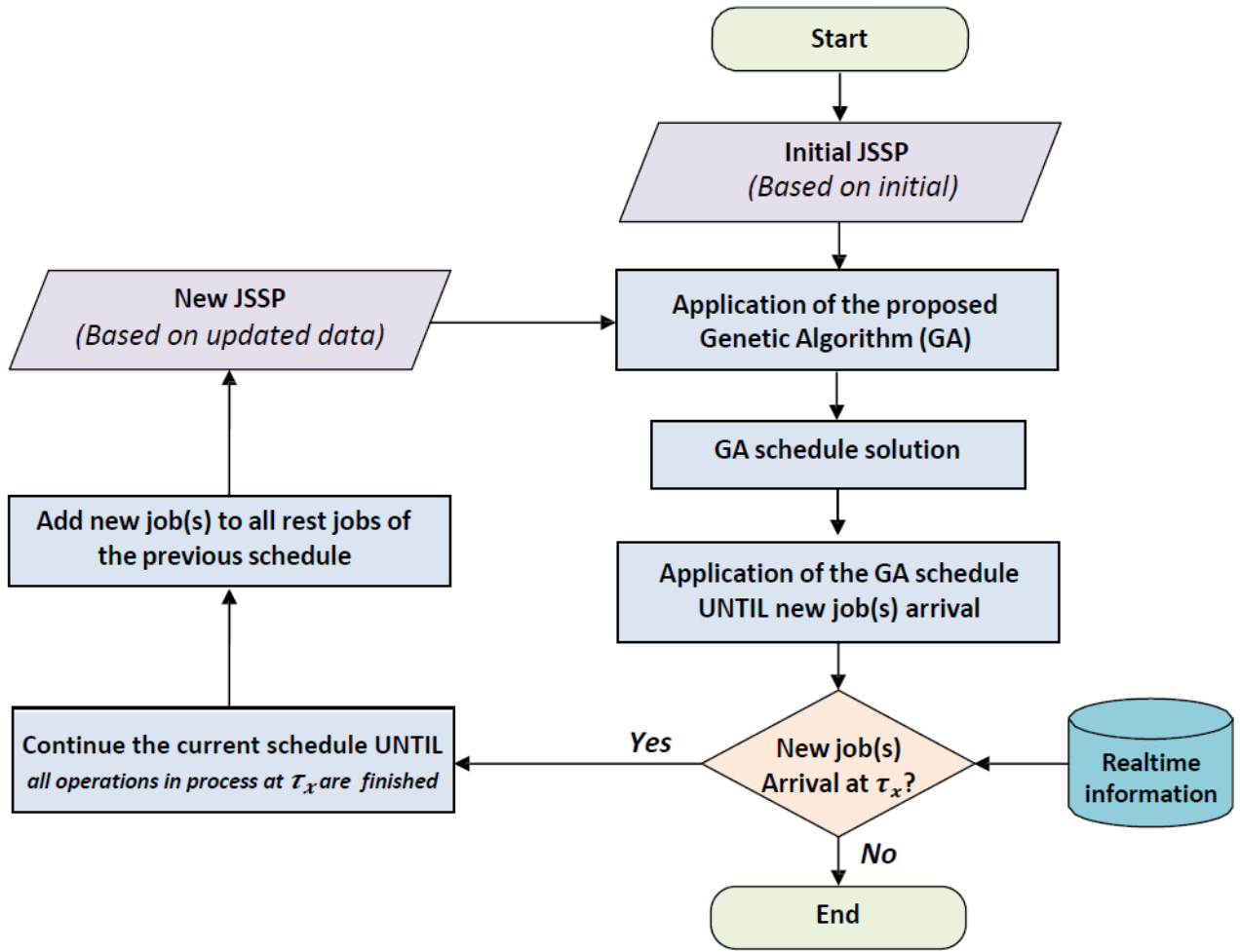


FIGURE 3. Flow chart of the proposed approach for dynamical job sequencing.

crossover operators based on the behavior inheritance of best selected parents by their offsprings. The aim is to test their influence on the performance of the GA and implicitly on makespan minimization. All along the next subsections, we will use the same chromosome presented in Fig.5 to explain step by step the principle of the two proposed crossover operators applied with two points and three points crossover.

Starting by the given example: considering a job shop problem with two jobs J_1, J_2 and three machines M_1, M_2, M_3 :

$$\begin{matrix}
 J_1 : & \left[\begin{array}{ccccc}
 \overbrace{(1, 3)}^{O_{1,1}} & \overbrace{(3, 2)}^{O_{1,2}} & \overbrace{(2, 1)}^{O_{1,3}} & \overbrace{(3, 4)}^{O_{1,4}} & \overbrace{(2, 3)}^{O_{1,5}} \\
 \end{array} \right] \\
 J_2 : & \left[\begin{array}{ccccc}
 \overbrace{(2, 2)}^{O_{2,1}} & \overbrace{(1, 2)}^{O_{2,2}} & \overbrace{(3, 5)}^{O_{2,3}} & \overbrace{(2, 3)}^{O_{2,4}} & \overbrace{(3, 3)}^{O_{2,5}} \\
 \end{array} \right]
 \end{matrix}$$

Each operation $O_{i,j}$ is presented by a pair of numbers (k, p) ; where k is the index of the allocated machine for processing an operation and p is the processing time of that operation. In the above example, the size of the chromosome is equals to 10 genes.

A. PROPOSED MIN-MAX CROSSOVER OPERATORS

The process of reproduction starts by selecting two parents' chromosomes. The principle of the two first proposed crossover operators, called respectively, Min-Max 2 points (Fig.6) and Min- Max 3 points (Fig.7) is described as follows:

- For the Min-Max 2 points: Random selection of 2 points crossover.
- For the Min-Max 3 points: A random selection of 3 points crossover.
- Calculate the absolute processing time $Pa_{c,s}$ of each segment of the two parents chromosomes:

$$Pa_{c,s} = \sum_{i=1}^{P_s} P_{(i^{th}, s^{th})} \quad (7)$$

where

P_s denotes the length of the segment s . $P_{i,s}$ is the processing time of the i^{th} gene of the s^{th} segment of the chromosome c .

- Arrange of three absolute processing times as the following: -For Min-Max 2 points: minimum (min), medium (med), and maximum (max).

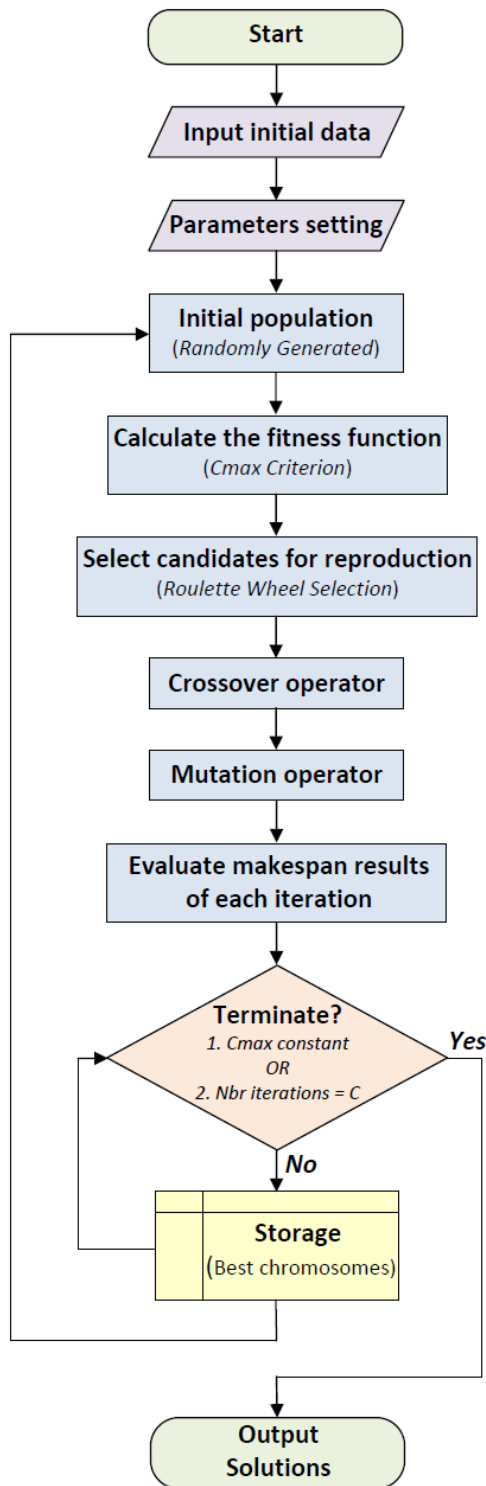


FIGURE 4. An illustrative representation of the GA based approach

- For Min-Max 3 points: minimum (min), medium 1 (med 1) medium 2 (med 2), and maximum (max).
- Switching each segment of each parent chromosome between both of them based on their positions:
 - For Min-Max 2 points: min by min, med by med, and max by max.

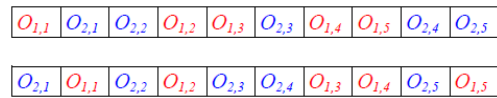


FIGURE 5. Selected chromosomes for reproduction process.

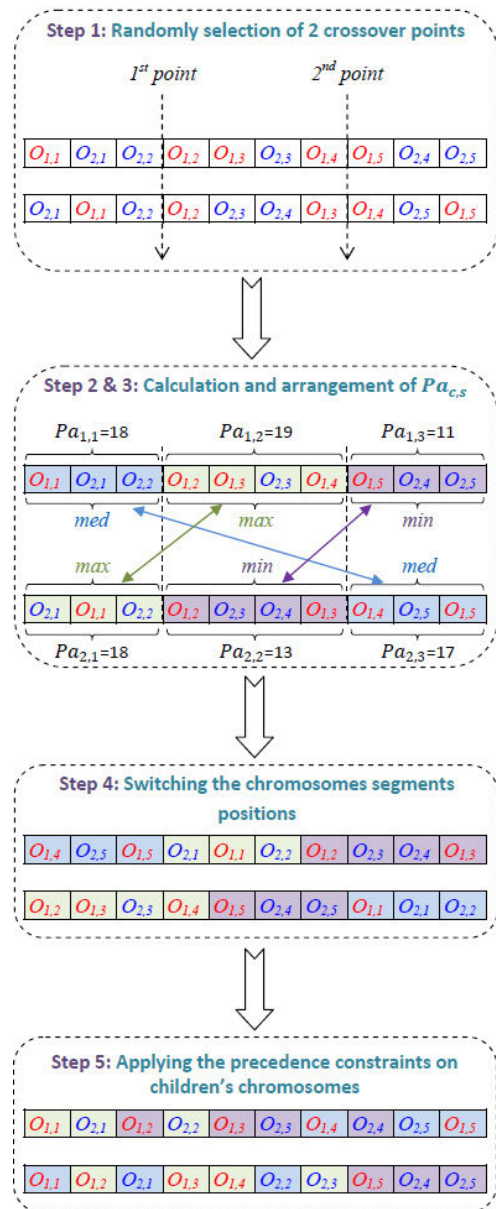


FIGURE 6. Min-Max 2 points crossover operator.

-For Min-Max 3 points: min by min, med 1 by med 1, med 2 by med 2, and max by max.

- Sequencing of the new offsprings with respect to precedence constraint relationships.

Performance Analysis of the Min-Max crossover operators:

Unlike all crossover operators established in the literature where a physical exchange is performed, the significant advantage of the present new method is the behavioral

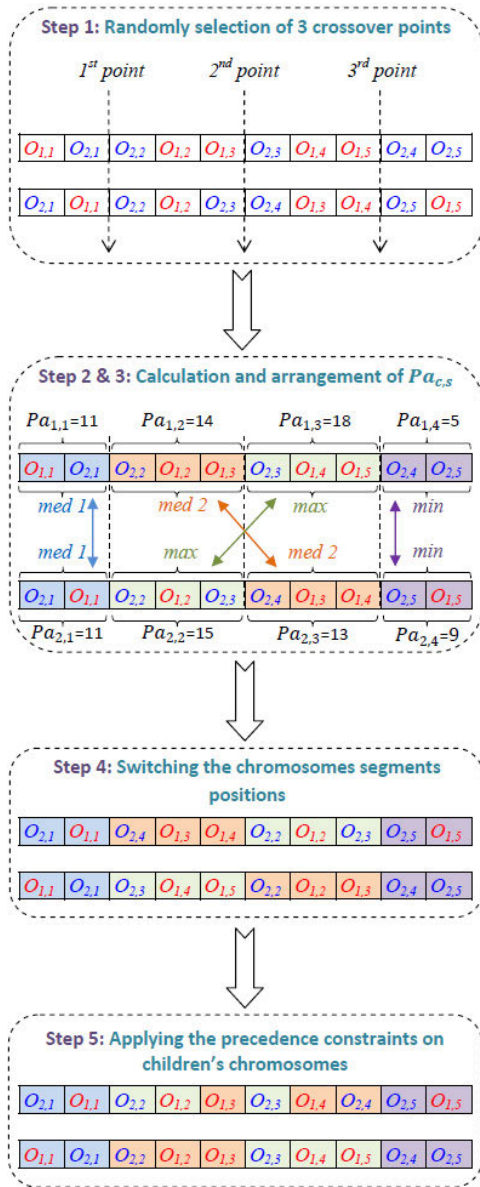


FIGURE 7. Min-Max 3 points crossover operator.

crossing aspect of genes that do not considers the physical crossing between the selected parents. The proposed operator can be viewed or called a virtual crossover. Indeed, it consists of applying local permutations of the positioning of the chromosome segments. Consequently, the new offsprings will be reconstructed based on the excellent quality behavior of the opposite parent. The application of the virtual crossover permits the combination of the behavior of crossed chromosomes to estimate the possible best solutions. One more important aspect is to perform the internal performance analysis of the scheduling parameters vs. the local reproducibility evolution of fixed segments. The major disadvantage of this approach is that identical reproduction genes can occur when parents are with the same quantitative arrangement of segments. This inconvenience becomes negligible by increasing the number of crossover points.

B. PROPOSED MEAN/Min-MAX CROSSOVER OPERATORS

The Mean/Min-Max crossover consists of an exchange of the behavior between two parents based on the mean of absolute processing time of chromosomes segments. In Fig.8 and Fig.9, we illustrate through an example the following steps, one by one, of the two proposed crossover operators, called Mean/Min-Max 2 points and Mean/Min-Max 3 points.

- For Mean/Min-Max 2 points: Random selection of 2 points crossover.
- For Mean/Min-Max 3 points: Random selection of 3 points crossover.
- Calculate the mean of the absolute processing time $Pa_{c,s}$ of each segment of the two parent chromosomes:

$$\mu(Pa_{c,s}) = \frac{\sum_{i=1}^{Ps} P_{i,s}}{n_{c,s}} \tag{8}$$

where: $n_{c,s}$ is the number of genes of the s^{th} segment of the chromosome c.

- Arrangement of the three means of the absolute processing times as the following: -For Mean/Min-Max 2 points: minimum (min), medium (med), and maximum (max).
- For Mean/Min-Max 3 points: minimum (min), medium 1 (med 1), medium 2 (med 2), and maximum (max).
- Switching the chromosome segment position between the two parents: -For Mean/Min-Max 2 points: min by min, med by med, and max by max.
- For Mean/Min-Max 3 points: min by min, med 1 by med 1, med 2 by med 2, and max by max.
- Sequencing of the new offsprings with respect to precedence constraint relationships.

Performance Analysis of the Mean/Min-Max crossover operators:

The Mean/Min-Max 2 points crossover keeps the same advantages as described above. Nevertheless, we consider the Mean/Min-Max method is better due to its ability to reduce identical reproductions, considering the more stringent and detailed assessment of segments. We note that the two used proposed crossover operators are used based on the changing of genes information for 2 and 3 points crossover operators whereas the total processing time is calculated and exchanged between parent1 and its corresponding in parent2. For the second operator, we calculate the mean processing time and exchanging genes information between the two parents. For both methods, new offsprings are created.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

A. INSTANCES AND PARAMETERS SETTING

To evaluate the performance of the GA, 26 benchmarking random instances with different sizing problems (small, medium and large) are generated. We denote by n^*m the problem size; where n^* is the total number of jobs (including new job arrivals) executed on m machines. The number of operations

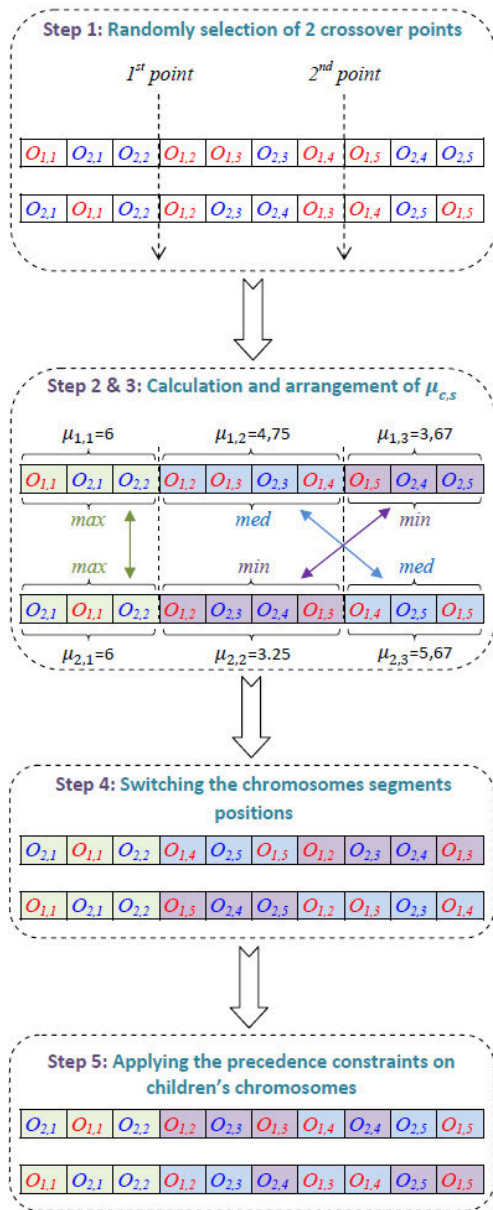


FIGURE 8. Mean/Min-Max 2 points crossover operator.

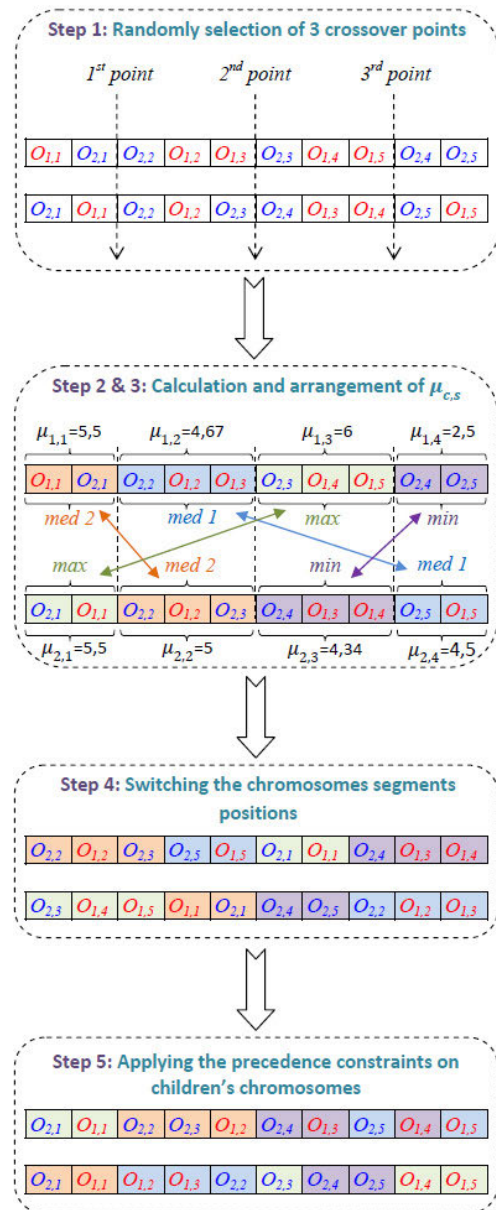


FIGURE 9. Mean/Min-Max 3 points crossover operator.

for a job follows the uniform discrete distribution within the interval [1, 10] and [1, 20]. The processing time of each job follows the uniform distribution of [1, 100]. While increasing the size, problems become more complex. The first one is of 25 jobs and 5 machines. The last instance contains 300 jobs treated by 25 machines. The size of the first and the last generated problems are denoted respectively by (25*5) and (300*25).

To evaluate the approach, the 26 generated instances are structured in the following three subsets:

- Small problem-size (Small P-size) where the number of jobs is strictly less than 50: 25*5, 25*10, 30*5, 35*10, 35*15, 40*5, 40*10, 45*5, and 45*10.
- Medium problem-size (Medium P-size) where the number of jobs is between 50 and 100: 50*10, 55*10, 60*5,

70*5, 70*10, 75*5, 75*10, 80*5, 80*10, 90*5, and 95*10,

- Large problem-size (Large P-size) where the number of jobs is strictly superior to 100: 100*10, 115*5, 120*10, 130*10, 220*25, and 300*25.

With the goal of optimizing a job shop system with frequent dynamic changes, the makespan value of each static sub problem is calculated and, then, the whole completion time is performed at the end of the process. In this study, the proposed GA approach is applied to different sizing generated random instances. The obtained results are compared against the following state of the art heuristics and metaheuristics:

- GA1: Simple GA with one-point crossover operator (used in [31]).
- GA2: Simple GA with two-point crossover operator.

TABLE 2. GA parameters.

Parameters	values
Chromosome size	Number of operations
Crossover probability	0.99
Mutation probability	0.01
Number of iteration	100
Population size	From 60 to 500(According to P-size)
Termination criterion	min makespan or max NBr iteration

- GRASP: Greedy Randomised Adaptive Search Procedure (used in [29]).
- HGAT: Hybrid GA and Tabu search approach (used in [32] and [33]).
- GAKK: Genetic Algorithm based on KK heuristic proposed in [18].
- MPSO: Modified PSO proposed in [23].
- SPT: Shortest Processing Time heuristic.
- LPT: Longest Processing Time heuristic.
- EDD: Earliest Due Date heuristic.

Experimentation results were performed using Matlab on a computer with an Intel Core i5 (2.7 GHz) processor, 4 GB of RAM, and running under a Windows 10. All the used parameters of the proposed GA are selected and presented in Table 2.

B. EXPERIMENTATION RESULTS OF THE DYNAMIC JSSP

Table 3 and Fig.10 represent the different results of the makespan minimization for 26 Dynamic JSSP. In the following list, we give all compared approaches used for simulations:

- $GA1_{Cmax}$: MinMax GA with 2 points crossover combined with the swap operator mutation.
- $GA2_{Cmax}$: MinMax GA with 3 points crossover combined with the swap operator mutation.
- $GA3_{Cmax}$: Mean/MinMax with 2 points crossover combined with the swap operator mutation.
- $GA4_{Cmax}$: Mean/MinMax with 3 points crossover combined with the swap operator mutation.
- Mean 1: The average makespan of the set of the Small problem-size.
- Mean 2: The average makespan of the set of the Medium problem-size.
- Mean 3: The average makespan of the set of the Large problem-size.
- G-Mean: The average makespan of the set of the 26 generated instances and simulated results.

C. DISCUSSION AND INTERPRETATION

The simulation results show the efficiency of the proposed GA approach based virtual crossover operators for solving the Dynamic JSSP. Each one of the proposed virtual crossover operators (four types) is applied while executing the GA approach. The generated results are compared vs. all selected

state of the art metaheuristics and heuristics methods in terms of minimum makespan values.

The best selected Cmax values marked by gray color in Table 3 is the result of the proposed combination with the principle of virtual crossovers seen in Section 3: $GA1_{Cmax}$, $GA2_{Cmax}$, $GA3_{Cmax}$ and $GA4_{Cmax}$. Only for six problems (23% of generated problems), the minimum value of the makespan is equally shared by the proposed GA approach and one of the used metaheuristics: GRASP for the two instances (40*5) and (70*5); HGAT for (45*5) and (70*5) cases; MPSO for the (25*5) and (70*10) problems; finally, only one with the GAKK metaheuristic for the instance (30*5).

Hence, it is clear that the marginal efficiency of GRASP, HGAT, MPSO, and GAKK is validated occasionally only with Small and Medium problem-sizes. Moreover, it is shown that the $GA1_{Cmax}$ method is much better than the $GA2_{Cmax}$ method in terms of Cmax minimization for all 26 cases (small, medium and large instances) by calculating the mean of each problem instance. We saw the same remark when we calculate the G-mean. According to Table 3, by applying the Mean/Min-Max 2 points, the performance of our proposed method seen in $GA3_{Cmax}$ results is much better for small problem size than the $GA2_{Cmax}$, $GA1_{Cmax}$ and $GA4_{Cmax}$. On one hand, the addressed results have seen interesting for small problem size based on G-mean value of $GA3_{Cmax}$.

We show an average of makespan value equal to 204.11(s) in mean 1. On the other hand, for medium and large problem size instances, the computational results were clearly improved using the $GA4_{Cmax}$ method with a slight performance of $GA2_{Cmax}$ and $GA3_{Cmax}$. The mean 2 of $GA4_{Cmax}$ was equal to 341.27(s) much better than $GA2_{Cmax}$ and $GA3_{Cmax}$ results. Same remarks, for mean 3 that is equal to 396.33(s) as well as the obtained G-mean result that reached 309.31(s).

Based on different computed values of mean2, mean 3 and G-mean, we affirm the efficiency of the proposed crossover operators in terms of minimal Cmax using the $GA4_{Cmax}$ method. Fig. 10 confirms our opinion as well. Unlike the results of the first set of problems seen in Table 3 and Figure 10, we can easily deduce the aptitude of the Mean/Min-Max 2 points virtual crossover $GA3_{Cmax}$ to provide minimal Cmax values. Based on $GA4_{Cmax}$ results, we show a marginal quality with 7 optimal solutions from 11 for medium size (in gray color). It is also mentioned that the $GA2_{Cmax}$ gives 3 optimal solution (in gray color) and one solution provided by $GA3_{Cmax}$ for the instance (60*5). For large size problems, it has been seen the same remark with only 4 best solution among 6 for $GA4_{Cmax}$ method (in gray color) and 2 best solution for $GA3_{Cmax}$ with the instances (100*10) and (220*25). One solution for the $GA3_{Cmax}$ using the instance(60*5) and three solutions for the $GA2_{Cmax}$ with (50*10), (70*5), and (80*5) instances.

These results show the performance of the 3-point virtual crossover operator to enhance the performance of the GA to solve the Dynamic JSSP with random arrival of new jobs.

TABLE 3. Experimentation results of the proposed GA based virtual crossover (VC) operators VS. metaheuristics and dispatching rules in terms of minimum Cmax.

Problem size	Data $n \times m$	GA-VC				Meta-heuristics					P-Dispatching Rules			
		$GA1_{Cmax}$	$GA2_{Cmax}$	$GA3_{Cmax}$	$GA4_{Cmax}$	GA1	GA2	GRASP	HGAT	GAKK	MPOSO	SPT	LPT	EDD
SMALL P-SIZE	25*5	280	277	268	270	371	308	299	292	297	268	384	380	389
	25*10	198	192	187	194	252	212	205	207	207	205	269	262	265
	30*5	242	243	233	237	349	273	254	270	233	258	362	354	359
	35*10	179	178	171	183	228	216	201	199	203	193	237	233	239
	35*15	80	78	89	94	166	116	103	103	106	106	183	179	173
	40*5	232	235	241	244	391	274	232	270	276	244	394	389	402
	40*10	105	91	98	115	207	136	122	127	131	115	210	215	202
	45*5	371	384	380	392	409	422	392	371	388	399	451	446	442
	45*10	174	171	170	181	231	209	195	185	185	191	238	232	229
	mean 1		206,78	205,44	204,11	212,22	289,33	240,67	222,56	224,89	225,11	219,89	303,11	298,89
MEDIUM P-SIZE	50*10	127	125	132	141	229	178	145	138	142	155	228	226	237
	55*10	252	258	248	244	339	286	274	271	269	270	352	348	349
	60*5	462	460	457	459	577	498	487	469	462	492	592	587	584
	70*5	439	436	441	439	519	462	436	436	459	466	516	510	528
	70*10	291	289	276	262	344	311	295	299	287	295	354	369	352
	75*5	500	489	492	486	530	510	503	501	503	503	548	532	539
	75*10	283	289	284	280	335	316	305	299	301	313	348	341	338
	80*5	452	445	450	448	548	471	466	473	471	473	554	549	559
	80*10	312	317	312	301	348	334	331	326	331	326	368	356	376
	90*5	380	377	378	374	464	415	402	409	392	402	472	476	469
95*10	328	322	325	320	454	349	336	344	333	350	463	469	456	
mean 2		347,82	346,27	345,00	341,27	426,09	375,45	361,82	360,45	359,09	367,73	435,91	433	435,18
LARGE P-SIZE	100*10	265	274	264	272	379	309	292	294	294	307	386	382	378
	115*5	410	400	406	394	449	417	412	407	412	421	460	455	458
	120*10	366	355	360	354	452	375	388	390	377	392	466	462	457
	130*10	349	342	344	340	403	379	381	384	381	394	433	430	438
	220*25	584	592	579	586	692	627	627	627	632	632	682	679	719
	300*25	436	439	464	432	525	491	473	469	464	468	594	682	688
	mean 3		401,67	400,33	402,83	396,33	483,33	433	428,83	428,50	426,67	435,67	503,50	515
G-Mean		312,19	310,77	309,50	309,31	391,96	342,08	328,27	329,23	328,31	332,23	405,54	405,50	414,40

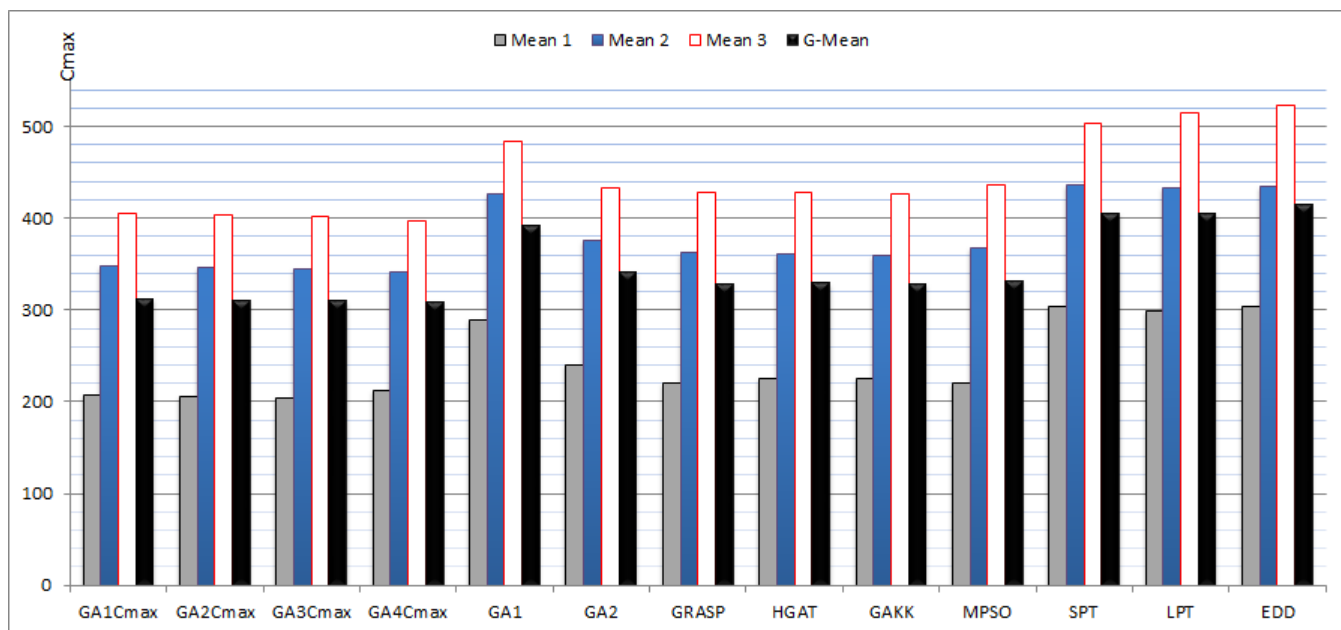


FIGURE 10. Comparisons in terms of average makespan results using various problems size.

Indeed, for 10 problems among 11 (90,9% of total cases), best results were obtained when we combine the proposed GA approach with one of the virtual crossover operators based 3 points crossover. This is due to the performance of the 3-point crossover in the reduction of producing identical genes through the virtualization of crossing over. These results confirm our opinion when we have discussed the proposed GA based new crossover operators in previous subsections. Also, we can interpret that the GA approach combined with virtual 2 points crossover operators do not

give the best makespan values for the medium class of problems (only for small P-size). In Figure 10, the given values of Mean 2 and mean 3 confirms the efficiency of the proposed virtual crossover operators vs. methods developed in the literature (seen in Table 3). Obviously, the obtained means values (for medium and large P-size) of the $GA4_{Cmax}$ method demonstrate the global performance of the proposed Mean/Min-Max with 3 points crossover operator to reach an acceptable solution results. For the set of problems with large size, the Min-Max 2 points methods give only

33.33% best solutions of total cases and about 66,66% based Mean/Min-Max 3 points.

The G-mean values demonstrate the efficiency of the four proposed virtual crossover operators vs. all compared methods. By comparing our proposal to the best metaheuristics cited in the literature, it is shown near results but globally there is a huge difference this is due to the variation of the job environment with dynamic perturbations. As well as for the used Priority Dispatching rules (P-dispatching rules) that have been seen very far from both the used metaheuristics and our proposal in terms of Cmax values. It is meaningful due to the role of metaheuristics in giving optimal solutions (marked with gray color in Table 3).

The global G-mean values of the proposed approach indicate that the experimentation outputs are appeared in an interval of small uncertainty due to the definition of the Dynamic JSSPs as a sequence of static JSSPs. However, the performance of each virtual crossover operator is proportional to the problem size and the best solution is given based on Mean/Min-Max 3 points crossover virtual operator.

V. CONCLUSION AND FUTURE DIRECTIONS

In this paper, a Dynamic JSSP is addressed with the objective to minimize the makespan with continuous jobs arriving in a manufacturing system while considering the sequencing of operations. A GA based metaheuristic approach is used to manage the job operating processing through the proposed new virtual crossover operators. Important formulations are presented in the context of developing the dynamic JSP with precedence constraints and a preemptive model during scheduling. Experimental results demonstrate that our proposal GA has a shorter makespan than the other compared algorithms when scheduling is performed with mean/min-max 3 points crossover operator. The proposed algorithm gives promoted solutions for small, medium, and especially large size problems according to the obtained global average makespan. To the best of our knowledge, there are few works that deal with this kind of optimization shop floor by regards to its hardness. Hence, we aim to continue working on extended methods capable to ensure system robustness in terms of schedule quality and time-consuming.

REFERENCES

- [1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.
- [2] A. Sangaiah, M. Suraki, M. Sadeghilalimi, S. Bozorgi, A. Hosseinabadi, and J. Wang, "A new meta-heuristic algorithm for solving the flexible dynamic job-shop problem with parallel machines," *Symmetry*, vol. 11, no. 2, p. 165, Feb. 2019.
- [3] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling," *Appl. Soft Comput.*, vol. 63, pp. 72–86, Feb. 2018, doi: [10.1016/j.asoc.2017.11.020](https://doi.org/10.1016/j.asoc.2017.11.020).
- [4] F. Zhao, J. Zhang, C. Zhang, and J. Wang, "An improved shuffled complex evolution algorithm with sequence mapping mechanism for job shop scheduling problems," *Expert Syst. Appl.*, vol. 42, no. 8, pp. 3953–3966, May 2015.
- [5] Q. Meng, L. Zhang, and Y. Fan, "A hybrid particle swarm optimization algorithm for solving job shop scheduling problems," in *Proc. Asian Simul. Conf.*, 2016, pp. 71–78.
- [6] Ramasesh, R., "Dynamic job shop scheduling: A survey of simulation research," *Omega Int. J. Manage. Sci.*, vol. 18, pp. 43–57, Aug. 1990.
- [7] J.-Q. Li and M. F. Pan Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job shop scheduling problem with maintenance activities," *Appl. Math. Model.*, vol. 38, no. 3, pp. 1111–1132, 2014.
- [8] M. A and R. R., "A comparison of artificial bee colony algorithm and genetic algorithm to minimize the makespan for job shop scheduling," *Procedia Eng.*, vol. 97, pp. 1745–1754, Dec. 2014, doi: [10.1016/j.proeng.2014.12.326](https://doi.org/10.1016/j.proeng.2014.12.326).
- [9] S. Sundar and P. N. Suganthan, C. T. Jin, C. T. Xiang, and C. C. Soon, "A hybrid artificial bee colony algorithm for the job shop scheduling problem with no-wait constraint," *Soft Comput.*, vol. 21, no. 5, pp. 1193–1202, 2017, doi: [10.1007/s00500-015-1852-9](https://doi.org/10.1007/s00500-015-1852-9).
- [10] B. Çalı and S. Bulkan, "A research survey: Review of AI solution strategies of job shop scheduling problem," *J. Intell. Manuf.*, vol. 26, no. 5, pp. 961–973, Oct. 2015.
- [11] D. Tk, "Parallel bat algorithm for optimizing makespan in job shop scheduling problems," *J. Intell. Manuf.*, vol. 29, p. 451, Dec. 2018, doi: [10.1007/s10845-015-1121-x](https://doi.org/10.1007/s10845-015-1121-x).
- [12] A. Bagheri, M. Zandieh, I. Mahdavi, and M. Yazdani, "An artificial immune algorithm for the flexible job-shop scheduling problem," *Future Gener. Comput. Syst.*, vol. 26, no. 4, pp. 533–541, Apr. 2010.
- [13] M.-C. Wu, C.-S. Lin, C.-H. Lin, and C.-F. Chen, "Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems," *Comput. Oper. Res.*, vol. 80, pp. 101–112, Apr. 2017.
- [14] F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling," in *Advances in Artificial Intelligence*, vol. 11320. Cham, Switzerland: Springer, 2018.
- [15] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: A survey with a unified framework," *Complex Intell. Syst.*, vol. 3, no. 1, pp. 41–66, Mar. 2017, doi: [10.1007/s40747-017-0036-x](https://doi.org/10.1007/s40747-017-0036-x).
- [16] M. R. Raeesi N. and Z. Kobti, "A memetic algorithm for job shop scheduling using a critical-path-based local search heuristic," *Memetic Comput.*, vol. 4, no. 3, pp. 231–245, Sep. 2012.
- [17] G. Vilcot and J.-C. Billaut, "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem," *Eur. J. Oper. Res.*, vol. 190, no. 2, pp. 398–411, Oct. 2008.
- [18] N. Kundakc and O. Kulak, "Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem," *Comput. Ind. Eng.*, vol. 96, pp. 31–51, Jun. 2016.
- [19] Maaoui, Amel, "Petri nets-based approach for scheduling and rescheduling in multi-site manufacturing system," *Int. J. Digit. Signals Smart Syst.* vol. 2, no. 4, pp. 317–329, 2018.
- [20] A. Maaoui, A. Abdellatif, A. J. Telmoudi, S. Gattoufi, and L. Nabli, "Multi-site manufacturing system scheduling using Petri nets," in *Proc. 15th Int. Multi-Conference Syst., Signals Devices (SSD)*, Mar. 2018, pp. 469–474.
- [21] J. W. Barnes and J. B. Chambers, "Solving the job shop scheduling problem with tabu search," *IIE Trans.*, vol. 27, no. 2, pp. 257–263, Apr. 1995, doi: [10.1080/07408179508936739](https://doi.org/10.1080/07408179508936739).
- [22] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Manage. Sci.*, vol. 42, no. 6, pp. 797–813, Jun. 1996.
- [23] Z. Wang, J. Zhang, and S. Yang, "An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals," *Swarm Evol. Comput.*, vol. 51, Dec. 2019, Art. no. 100594.
- [24] Y. Zhou, J.-J. Yang, and L.-Y. Zheng, "Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling," *IEEE Access*, vol. 7, pp. 68–88, 2019.
- [25] L. Zhang, L. Gao, and X. Li, "A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3516–3531, Jun. 2013.
- [26] A. Baykasoğlu and F. S. Karaslan, "Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3308–3325, Jun. 2017.
- [27] A. Baykasoğlu and F. B. Ozsoydan, "Dynamic scheduling of parallel heat treatment furnaces: A case study at a manufacturing system," *J. Manuf. Syst.*, vol. 46, pp. 152–162, Jan. 2018.

- [28] A. Baykasoğlu and F. B. Ozsoydan, "Evolutionary and population-based methods versus constructive search strategies in dynamic combinatorial optimization," *Inf. Sci.*, vol. 420, pp. 159–183, Dec. 2017.
- [29] Baykasoğlu, Adil, and Fehmi B. Ozsoydan, "A GRASP based approach to dynamic scheduling of parallel heat treatment furnaces in a manufacturing company," in *Proc. 7th Multidisciplinary Int. Conf. Scheduling, Theory Appl.*, Aug. 2015, Prague, Czech Republic, pp. 25–28.
- [30] Z. Cao, L. Zhou, B. Hu, and C. Lin, "An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem," *Bus. Inf. Syst. Eng.*, vol. 61, no. 3, pp. 299–309, Jun. 2019, doi: [10.1007/s12599-019-00590-7](https://doi.org/10.1007/s12599-019-00590-7).
- [31] K. Ben Ali, A. J. Telmoudi, and S. Gattoufi, "Stochastic cases of the dynamic job shop problem based on the genetic algorithm to minimize," in *Proc. 4th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Apr. 2017, pp. 760–765, doi: [10.1109/CoDIT.2017.8102686](https://doi.org/10.1109/CoDIT.2017.8102686).
- [32] K. B. Ali, A. J. Telmoudi, and S. Gattoufi, "An improved genetic algorithm with local search for solving the DJSSP with new dynamic events," in *Proc. IEEE 23rd Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2018, pp. 1137–1144.
- [33] K. B. Ali, A. J. Telmoudi, and S. Gattoufi, "Adopted rescheduling strategy for solving the dynamic job shop using GA based local search," in *Proc. Int. Conf. Adv. Syst. Emergent Technol.*, Mar. 2019, pp. 68–73, doi: [10.1109/ASET.2019.8871034](https://doi.org/10.1109/ASET.2019.8871034).
- [34] H. Xiong, H. Fan, G. Jiang, and G. Li, "A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints," *Eur. J. Oper. Res.*, vol. 257, no. 1, pp. 13–24, 2017, doi: [10.1016/j.ejor.2016.07.030](https://doi.org/10.1016/j.ejor.2016.07.030).
- [35] D. Karunakaran, Y. Mei, and G. C. M. Zhang, "Dynamic job shop scheduling under uncertainty using genetic programming," in *Proc. Intell. Evol. Syst.*, 2016, pp. 195–210.
- [36] D. Karunakaran, Y. Mei, G. Chen, and M. Zhang, "Evolving dispatching rules for dynamic job shop scheduling with uncertain processing times," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, San Sebastian, Spain, Jun. 2017, pp. 364–371, doi: [10.1109/CEC.2017.7969335](https://doi.org/10.1109/CEC.2017.7969335).



KAOUTHER BEN ALI received the M.Sc. degree in computer science from the Higher Institute of Management of Tunisia, in 2015, where she is currently pursuing the Ph.D. degree in computer science. Her research interests include almost the fields of operation research, combinatorial optimization problems and artificial intelligence for real-world applications.

Since 2017, she started working as a Researcher at the SMART Laboratory, Computer Science Department, Higher Institute of Management of Tunisia. The aim of her recent publications is to discuss new trends, review industrial needs, and present innovative solutions in production scheduling fields.



ACHRAF JABEUR TELMOUDI (Member, IEEE) received the M.Sc. degree in automation and systems engineering and the Ph.D. degree in electrical engineering from University of Tunis, Tunisia, in 2006 and 2011, respectively.

Since 2011, he has been an Associate Professor in automation and industrial informatics with the Higher Institute of Applied Science and Technology of Sousse, University of Sousse, Tunisia. He is currently a member of the LISIER Laboratory, The National Higher Engineering School of Tunis, University of Tunis, Tunisia. His current research interests include planning and scheduling, discrete event systems, Petri nets, prediction, and system identification. He serves as a Guest Editor for the *Journal of Systems and Control Engineering*, the *Transactions of the Institute of Measurement and Control*, and *Cybernetics and Systems*. He is also an Associate Editor of the *Journal of Systems and Control Engineering*, the *International Journal of Dynamics and Control*, and the *International Journal of Applied Metaheuristic Computing*. He is also a regular Reviewer for more than 20 journals, including IEEE SMC: Systems, IEEE TRANSACTIONS ON ROBOTICS, IEEE TRANSACTIONS ON AUTOMATIC CONTROL, IEEE ACCESS, *Neural Computing and Applications*, *Energy*, *Journal of Aerospace Engineering, Computers & Industrial Engineering*, *Journal of Systems and Control Engineering*, *Aircraft Engineering and Aerospace Technology*, *RAIRO—Operations Research*, and so on.



SAID GATTOUFI received the Ph.D. degree in management from Sabanci University, Turkey, in 2002. He was an Associate Professor with the Operations Management and Business Statistics Department, Sultan Qaboos University, Oman, from 2005 to 2014. He is currently a Professor with the Higher Institute of Management of Tunisia, where he is also a Researcher with the SMART Laboratory. He is an active Researcher. His publications are frequently cited in well-reputed international journals, including the *Journal of Operational Research Society*, *Socio-Economic Planning Science* journal, *International Journal of Flexible Manufacturing Systems*, *International Journal of Computer Mathematics*, *International Journal of Accounting and Finance*, *Journal of Risk Finance*, and *Journal of Business and Economics Research*. He participated as an organizer, a presenter, and a keynote speaker in well-known conferences around the world. His main areas of research are the performance assessment and the analysis in management, in general, using data envelopment analysis (DEA) and its derived approaches.

...