RAIRO-Oper. Res. 55 (2021) S1165–S1193
https://doi.org/10.1051/ro/2020082

**RAIRO Operations Research**
www.rairo-ro.org

# A SUPERVISED METHOD FOR SCHEDULING MULTI-OBJECTIVE JOB SHOP SYSTEMS IN THE PRESENCE OF MARKET UNCERTAINTIES

Aidin Delgoshaei[1,*], Aisa Khoshniat Aram[2], Saeed Ehsani[3], Alireza Rezanoori[1], Sepehr Esmaeili Hanjani[3], Golnaz Hooshmand Pakdel[4] and Fatemeh Shirmohamdi[1]

**Abstract.** In real industries, managers usually consider more than one objective in scheduling process. Minimizing completion time, operational costs and average of machine loads are amongst the main concerns of managers during production scheduling in practice. The purpose of this research is to develop a new scheduling method for job-shop systems in the presence of uncertain demands while optimizing completion time, operational costs and machine load average are taken into account simultaneously. In this research a new multi-objective nonlinear mixed integer programming method is developed for job-shop scheduling in the presence of product demand uncertainty. The objectives of the proposed method are minimizing cost, production time and average of machine loads index. To solve the model, a hybrid NSGA-II and Simulated Annealing algorithms is proposed where the core of the solving algorithm is set based on weighting method. In continue a Taguchi method is set for design of experiments and also estimate the best initial parameters for small, medium and large scale case studies. Then comprehensive computational experiments have been carried out to verify the effectiveness of the proposed solution approaches in terms of the quality of the solutions and the solving times. The outcomes are then compared with a classic Genetic Algorithm. The outcomes indicated that the proposed algorithm could successfully solve large-scale experiments less than 2 min (123 s) that is noticeable. While performance of the solving algorithm are taken into consideration, the proposed algorithm could improve the outcomes in a range between 9.07% and 64.96% depending on the input data. The results also showed that considering multi-objective simultaneously more reasonable results would be reached in practice. The results showed that the market demand uncertainty can significantly affect to the process of job shop scheduling and impose harms in manufacturing systems both in terms of completion time and machine load variation. Operational costs, however, did not reflect significantly to market demand changes. The algorithm is then applied for a manufacturing firm. The outcomes showed that the proposed algorithm is flexible enough to be used easily in real industries.

**Mathematics Subject Classification.** 93A30.

Received December 1, 2019. Accepted July 24, 2020.

*Keywords.* Job-shop, NSGA-II, Simulate Annealing, multi-objective scheduling.

[1] Department of Mechanical and Manufacturing Engineering, University of Putra Malaysia, Serdang, Sl43400, Malaysia.
[*]Corresponding author:delgoshaei.aidin@gmail.com

[2] Department of Management, The Social Science University of Ankara, Ankara, Turkey.

[3] Department of Management, Islamic Azad University, Tehran, Iran.

[4] Department of Industrial Engineering, Kharazmi University, Tehran, Iran.

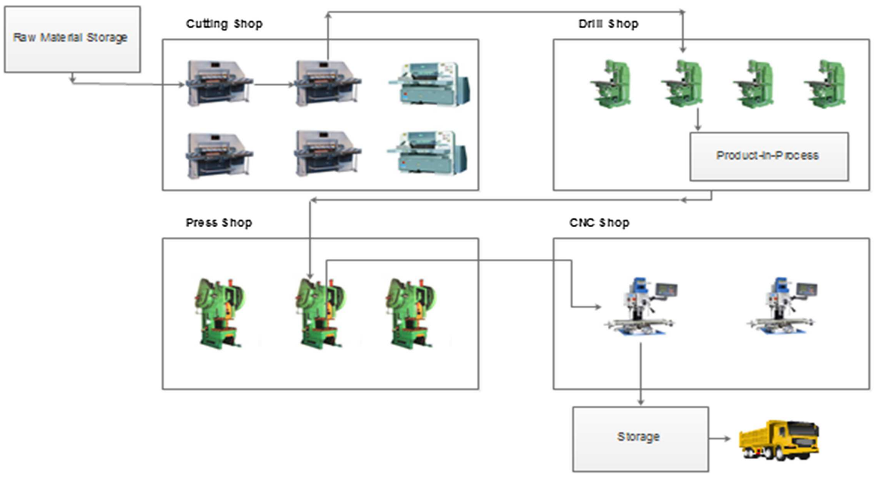


FIGURE 1. A graphical view of job-shop based layout.

## 1. Introduction

Job-shop layouts cover a wide range of manufacturing systems in industries. Classic job-shop scheduling problem (JSSP) aims to find the best part routes of completing products respect to operational process chart of products. Although the philosophy of classic JSSP problems is delivering the products on-time but reviewing the literature shows that scientists dealt with various objective functions such as completion time (makespan), cost, quality, job tardiness, etc. In JSSP problem, each of the processes of a product can be served by one machine or set of parallel machines which are located in a shop and can serve similar services [19]. In modern production systems, new scheduling methods are always a vital to achieve the fastest response to customer demands considering market changes (Fig. 1).

In most real cases, part demands are different from one planning horizon to another. Such a phenomena is known as dynamic part demands. Market changes, changes in product designs, pandemic calamities and designing new products are considered as the main reasons for the changes in part demands through different periods. One outcome of such conditions is emerging of imbalances in part routings and bottleneck machines.

The target of this research is to find the best production schedule of job shop systems in the dynamic condition of product demands in order to minimize the makespan, cost of product completion and average of machine loads simultaneously. Therefore, the objectives of the research will be minimizing makespan, cost of completion and average of machine loads. For this purpose, a mathematical programing model will be developed in the first step. In continue, a hybrid NSGA-II and Simulated Annealing (SA) algorithms will be proposed to solve the model. The results will be then compared with Standard Genetic Algorithm.

In this paper a new scheduling method is proposed to optimize system costs (including setup, operating, and material transferring); manufacturing time and machine load variance simultaneously. The proposed algorithm is flexible enough to be used easily in real industries while the market demands are not fixed and can be varied from a period to another.

## 2. Literature review

Job-shop researches are wide and include many objectives and conditions. During the last three decades, scientists addressed various problems to fill the gaps in Job-shop issue. Therefore, a number of novel research studies from 2009 to 2019, which are more related to the problem statement of this research, will be reviewed.

## 2.1. Completion time/makespan in job-shop scheduling problem (JSSP)

Perhaps minimizing the completion time of producing products (also named makespan) is developed more than other researches in job-shop scheduling during the last 2 decades. The aim is to find best series of machines in shops to finalize the product demands. Heuristics and Metaheuristics have been widely applied for Job-shop scheduling problems. Hasan *et al.* [17] argued that the primary objective of this research is to solve the job-shop scheduling problems, by minimizing the makespan, with and without process interruptions. Hence, they developed a GA for solving the job-shop problem, and then improved the algorithm by integrating it with two simple priority rules and a hybrid rule. Y. Wang [41] used GA for job-shop scheduling problem. In order to increase the diversity of the population, a mixed selection operator based on the fitness value and the concentration value was given. To make full use of the characteristics of the problem itself, new crossover operator based on the machine and mutation operator based on the critical path were specifically designed. In the same year, Hamidinia *et al.* [16], used genetic algorithm to sort machines and operators into a flexible process system. Lei [21] addressed a fuzzy flexible job-shop scheduling problem that was solved by an effective co-evolutionary genetic algorithm where the minimization of fuzzy makespan was taken into account. Demir and İşleyen [10] addressed four frequently used formulations of the FJSSP and a time-indexed model for FJSSP considering sequencing operations on machines. Baykasoğlu *et al.* [4] used teaching learning-based optimization algorithm (TLBO) for Job-shop scheduling problem. The performance of the TLBO algorithm is tested on some combinatorial optimization problems, namely flow-shop and job-shop scheduling problems. Saidi-Mehrabad *et al.* [35] focused on transportation times of the jobs between machines in a job-shop scheduling problem. Therefore, a mathematical model that is composed of job-shop scheduling and conflict-free routing problem. Then a two stage Ant Colony Algorithm is coded to solve the problem. Delgoshaei *et al.* [9] proposed a hybrid Greedy and Genetic Algorithm for minimizing makespan. Nouiri *et al.* [30] proposed a particle swarm optimization algorithm for flexible job-shop scheduling problem to minimize makespan. Dao *et al.* [6] parallel Bat Algorithm for minimizing Makespan in JSSP. Garg [14] used a hybrid Gravitational Search and Genetic Algorithms for solving non-linear mixed integer models.

## 2.2. Flexible job-shop scheduling problem (FJSSP)

Flexible job-shop problem is referred to a special type of job-shop scheduling problem where operations are allowed to be processed by any machine from a given set. Rahmati and Zandieh [33] proposed a Biogeography-based optimization algorithm (BBO) for flexible job-shop scheduling problem. To assess the performance of BBO, it is also compared with a genetic algorithm that has the most similarity with the proposed BBO in terms of three different objective functions that were makespan, critical machine workload, and total workload of machines. Y. Wang [41] proposed an effective artificial bee colony (ABC) algorithm for the flexible job-shop scheduling problem to minimize the maximum completion time. Yuan *et al.* [45] developed a flexible job-shop scheduling problem with the criterion to minimize makespan. In continue a novel hybrid harmony search algorithm based on the integrated approach is proposed to solve the problems. Rossi [34] used a swarm intelligence approach based on a disjunctive graph model in order to schedule a manufacturing system with resource flexibility and separable setup times in order to minimize the makespan. Garg [12] applied a hybrid PSO-GA algorithm for constrained optimization problems. They showed superiority of the proposed hybrid algorithm comparing to other existing methods. Xu *et al.* [43] presented an effective teaching learning-based optimization algorithm to solve the flexible job-shop problem with fuzzy processing time. In the first phase of their method, a special encoding scheme is used to represent solutions, and a decoding method is employed to transfer a solution to a feasible schedule in the fuzzy sense. Then, a bi-phase crossover scheme based on the teaching–learning mechanism and special local search operators are incorporated into the search framework of the TLBO to balance the exploration and exploitation capabilities. Yuan and Xu [44] proposed a new memetic algorithm for solving a multi-objective flexible job-shop scheduling problem while minimizing the makespan, total workload, and critical workload are considered. In continue, Shah *et al.* [37] proposed a Global Artificial

Bee Colony Search, use the foraging behavior of the global best and guided best honeybees for solving complex optimization tasks that worked based on maximizing fitness values instead of cycle numbers.

## 2.3. Multi objective/hybrid meta-heuristics for job-shop scheduling

In the real world, decision makers are often used single objective for scheduling. During the last decade many scientists tried to formulate models by considering more than one objective functions. Although developing multi-objective functions increases the complexity of the job-shop models, but at the same time it will results in more realistic solutions. Jinsong *et al.* [18] developed a new integrated model and an approach based on a modified genetic algorithm to facilitate the integration and optimization of the process of scheduling and timing for components in a sequence. The novel feature of their study was considering the functions of the planning process and scheduling simultaneously. Meeran and Morshed [28] presented a hybrid of Genetic and Tabu Search Algorithms. The rationale behind using such a hybrid method as in the case of other systems which use GA and TS is to use the diversified global search and intensified local search capabilities of GA and TS respectively. Chen *et al.* [5] proposed a Genetic and Grouping Genetic Algorithm, for scheduling job-shop in the presence of parallel machines and reentrant process. They considered total tardiness, total machine idle time and makespan are important performance measures used in this study. Banharnsakun *et al.* [3] proposed an ABC algorithm for Job-shop scheduling. The solution quality is measured based on best, average, standard deviation, and relative percent error of the objective value. Y. Wang [41], the hierarchical chromosome code of the genetic algorithm were designed to solve the multiparty project scheduling problem and several constraints. In the first layer, chromosomes were used to select the sequence of operations, in the second layer chromosomes were used for decision making in the combination of activity states. Li *et al.* [23] proposed a hybrid shuffled frog-leaping algorithm for solving the multi-objective flexible job-shop scheduling problem where the objectives were completion time (makespan), total workload of all machines and workload of the critical machine. Shao *et al.* [38] addresses a multi-objective flexible job-shop scheduling problem while makespan, maximal machine workload, and total workload are taken into consideration. In order to solve the problem, a hybrid discrete particle swarm optimization and simulated annealing algorithm was proposed. Garg and Sharma [15] proposed a multi-objective reliability redundancy allocation problem of a series system where the reliability of the system and the corresponding designing cost are targeted to be optimized simultaneously.

Li *et al.* [22] proposed a discrete ABC algorithm for solving the multi-objective flexible job-shop scheduling problem while maintenance activities are taken into consideration. In order to evaluate the performance of the algorithm maximum completion time, total workload of machines and workload of the critical machine are taken into account. Gao *et al.* [11] used a Pareto-based grouping discrete harmony search algorithm to solve the multi-objective flexible job-shop scheduling problem where maximum completion time and the mean of earliness and tardiness are considered as objective functions. In order to measure the performance of the proposed algorithm, number of non-dominated solutions, diversification metric and quality metric were used. Nguyen *et al.* [29] argued that scheduling policy strongly influences the performance of a manufacturing system. Therefore, they developed four new multi-objective genetic programming-based hyper-heuristic methods for automatic design of scheduling policies, including dispatching rules and due-date assignment rules in job-shop environments.

Karthikeyan *et al.* [20] used a hybrid discrete firefly algorithm for a multi-objective flexible job-shop scheduling problem where the possibility of performing an operation by any machine from a given set along different routes are allowed. Peng *et al.* [32] used a hybrid Tabu Search algorithm and path relinking for the job-shop scheduling problem where the proposed algorithm comprised several distinguishing features such as a specific relinking procedure to effectively construct a path linking the initiating solution and the guiding solution, and a reference solution determination mechanism based on two kinds of improvement methods. Shen and Yao [39] developed a multi-objective proactive–reactive evolutionary algorithm to deal with dynamic flexible job-shop scheduling, and provide different trade-offs among objectives. Delgoshaei and Gomes [7] focused on machine-load variation as a major shortcoming in manufacturing systems. For this purpose, a Multi-layer Perceptron method is proposed for scheduling dynamic manufacturing systems in the presence of bottleneck and parallel machines. They showed that the condition of dynamic costs affects the routing of materials in process and may induce machine-load variation. Zhang and Chiong [46] dealt with the objective of minimizing energy consumption into a typical production scheduling model. To solve this bi-objective optimization problem, a multi-objective genetic algorithm is proposed.

### 2.4. Uncertain job-shop scheduling

Luo *et al.* [24] designed a two-stage hybrid flow-shop scheduling problem in the condition of deterministic and uncertain modes for minimizing makespan. A genetic algorithm is used to obtain a near-optimal solution. The method is then applied in a metal-working company.

Zhang *et al.* [47] developed a method to address job-shop scheduling problem under an uncertain environment. Another contribution of this paper is to put forward a generalized Intuitionistic Fuzzy Sets (IFS) to process the additive operation and to compare the operation between two IFSs. Garg [13] used fuzzy membership function to reduce the uncertainty level in the industrial systems. Delgoshaei *et al.* [8] presented a new method for short-term period scheduling using simulated annealing where the aim was maximizing the profit of the system. Patwal *et al.* [31] focused on impact of renewable energy sources on the allocation of optimal power generation schedule. For this purpose, a time varying acceleration coefficient particle swarm optimization with mutation strategies is offered. Sajadi *et al.* [36] addressed a two-stage GA for solving job-shop scheduling problem where machine breakdown may occur during manufacturing process.

Table 1 compares opted researches which are close to the scope of the current research. An in depth survey in literature of job-shop scheduling shows that the idea dynamic job-shop scheduling (D-JSSP) for minimizing completion time, operational costs and average of machine loads has not been developed yet.

This paper alternates various material processing in the presence of parallel machines in plant shops and find the best part routes. This research is in continue of the Saidi-Mehrabad *et al.* [35] by considering makespan, system costs and average of machine loads simultaneously.

## 3. Research methodology

Saidi-Mehrabad *et al.* [35] focused on transportation times of the jobs between machines in a job-shop scheduling problem to provide conflict-free part routes. In this model, the same idea is expanded by considering completion time, system costs and average of machine loads simultaneously in an alternative part routing model.

### 3.1. Developing an appropriate model for planning

It should be noted that the demand for products in different planning periods is not constant and is estimated from the normal distribution function that has already been considered by other researchers in the available investigations in the literature section and hence, it is reliable in a mathematical model as a default. It should be noted that in the real world, the estimation of demand is estimated based on the mathematical formulas and the sales unit information of each plant and there is no need to use the normal function.

Briefly, the features of the model presented in this study can be explained as follows:

– Consider multi-objective decision making.
– Consider the potential demand for production planning.
– Consider vertical and horizontal movements.
– Presentation of the new method of intra-shop and inter-shop movements.

Model assumptions are as follows:

(1) The demand for various time periods is not constant and follows the normal distribution function.
(2) Each shop has a number of parallel machines that are located in specified locations.
(3) The distance between machines is measured as vertical and horizontal units (there is no diagonal and diametric movement between devices).
(4) The initial inventory of products is zero.
(5) The inventory at the end of the planning shall be zero.

TABLE 1. Comparing opted researches in the literature.

| Reference | Problem | Objective type | Objective(s) | Fuz./ Crsp. | Method | Demand | Contribution |
|---|---|---|---|---|---|---|---|
| Malve and Uzsoy [27] | JSSP | Single | Makespan | Crsp. | GA | Fixed | Using GA for JSSP with arranging processes and resources |
| Hasan *et al.* [17] | JSSP | Single | Makespan | Crsp. | GA | Fixed | Minimizing the makespan, with and without process interruptions |
| Y. Wang [41] | JSSP | Single | Makespan | Crsp. | GA | Fixed | Using a mixed selection operator based on the fitness value |
| Hamidinia *et al.* [16] | JSSP | Single | Makespan | Crsp. | GA | Fixed | Sorting machines and operators into a flexible process system |
| Lei [21] | Fuz-JSSP | Single | Makespan | Fuz. | GA | Fixed | Fuzzy flexible job-shop scheduling problem |
| Baykasoğlu *et al.* [4] | JSSP | Single | – | Fuz. | TLBO | Fixed | Using teaching learning-based optimization algorithm for JSSP |
| Saidi-Mehrabad *et al.* [35] | JSSP | Single | Material transferring time | Crsp. | ACO | Fixed | Dealing with material transferring time between machines |
| Y. Wang [41] | F-JSSP | Single | Makespan | Crsp. | ABC | Fixed | Using ABC algorithm for F-JSSP |
| Yuan *et al.* [45] | F-JSSP | Single | Makespan | Crsp. | Harmony search | Fixed | |
| Rossi [34] | F-JSSP | Single | Makespan | Crsp. | PSO | Fixed | Considering resource flexibility and separable setup times |
| Garg [12] | Scheduling problem | Single | Makespan | Crsp. | A hybrid PSO-GA algorithm | Fixed | Proposing A hybrid PSO-GA algorithm for constrained optimization problems |
| Yuan and Xu [44] | F-JSSP | Multi-objective | Makespan, total workload, critical workload | Crsp. | Memetic algorithm | Fixed | Developing a multi-objective decision making model for F-JSSP |
| Jinsong *et al.* [18] | JSSP | Multi-objective | Planning process, scheduling | Crsp. | GA | Fixed | Considering the functions of the planning process and scheduling simultaneously |
| Chen *et al.* [5] | JSSP | Multi-objective | Total tardiness, total machine idle time, makespan | Crsp. | GA-GGA | Fixed | Considering parallel machines and reentrant process |
| Shao *et al.* [38] | F-JSSP | Multi-objective | Makespan, machine workload, total workload | Crsp. | PSO-SA | Fixed | Developing multi-objective decision making model |
| Garg [13] | Fuzzy-scheduling | Single | | Fuzzy | Fuzzy Programming | Fixed | |
| Zhang *et al.* [47] | Fuz-JSSP | Single | Makespan | Fuzzy | IFS | Fixed | Using IFS for JSSP |
| Luo *et al.* [24] | JSSP | Single | Makespan | Crsp. | GA | Fixed | Considering deterministic and uncertain modes for minimizing makespan |

(6) If the loading rate of a machine is higher than the average of other machines in the same shop, this issue is considered as the loading variance. One of the goals of the model is to reduce shop-load variance as far as possible.
(7) The cost of intra-shop and inter-shop movements is specified for each unit.
(8) The sequence of operations needed to complete each product unit is already specified and is expressed by the OPC matrix.

*Inputs*

The range of variables and parameters of the model are defined as follows:

$i$ : number of products.
$j$ : number of machines.
$s$ : machine types.
$k$ : number of shops.
$t$ : time periods.
$e$ : number of machine-loads.

*Model parameters*

| | |
|---|---|
| $D(it) \sim N(\mu_i, \sigma_i^2)$ | The amount of $i$th product demand during the period $t$. |
| $L$ | Service level (used to estimate demand for products). |
| OPC($is$) | Operational Processes that needed to complete product $i$. |
| SC($i,j$) | If the product $i$ needs machine type $j$ for production. |
| IMC($i$) | The cost of loading product $i$. |
| EMC($i$) | The cost of each unit intra-shop movement of $i$th product. |
| IMT($i$) | The cost of each unit inter-shop movement of $i$th product. |
| EMT($i$) | Time of each unit intra-shop movement of product $i$ (min). |
| OT($sj$) | The time of each intermediate shop movement of $i$th product (min). |
| SC($i$) | Time to perform machine $j$ operations of type $s$ (min). |
| OC($sj$) | The cost of each unit of machine $j$ operations of type $s$. |
| IDC($jj'k$) | The distance of machine $j$ to machine $j'$ in the shop. |
| EDC($kk'$) | The shop distance $k$ to $k'$. |
| PDC($jk$) | The distance between each machine in the shop $k$ from the shop exit door. |
| QDC($jk$) | The distance between each machine in the shop $k$ from the shop entrance door. |
| CP($jk$) | The capacity of the machine $j$ in the shop $k$. |

*Model variables*

In this mathematical model, three groups of variables will be considered:

$X(it)$ : number of product $i$ which is produced in period $t$ (Integer).
$Y(ijkt)$ : if the product $i$ gets services from the machine $j$ in the shop $k$ during the period $t$ (Binary).
$\omega(j,k,t,e)$ : the number of times which the machine $j$ is loaded in the shop $k$ during period $t$ (Integer).

*Presentation of the mathematical model*

$$\begin{aligned}
\text{Min } Z_1 : W_1 \cdot & \sum_t^T \sum_i^I \sum_s^S \sum_k^K \sum_j^J \mathrm{SC}(i,j) \cdot \mathrm{OC}(sj) \cdot Y(ijkt) \cdot \mathrm{OPC}(is) \\
& + \sum_t^T \sum_i^I \sum_j^{J\sim J'} \sum_k^K \mathrm{IMC}(i) \cdot \mathrm{IDC}(jj'k) \cdot (Y(ijkt) - (1 - Y(ij'kt)) \cdot Y(ijkt)) \\
& + \sum_t^T \sum_i^I \sum_j^J \sum_k^{K\sim K'} \mathrm{EMC}(i) \cdot (\mathrm{EDC}(kk') + \mathrm{PDC}(jk) + \mathrm{QDC}(jk)) \cdot (Y(ijkt) \\
& - (1 - Y(ijk't)) \cdot Y(ijkt))
\end{aligned} \tag{3.1}$$

$$\begin{aligned}\text{Min } Z_2 : W_2 \cdot &\sum_{t}^{T}\sum_{i}^{I}\sum_{s}^{S}\sum_{k}^{K}\sum_{j}^{J}\text{OT}(sj)\cdot Y(ijkt)\cdot \text{OPC}(is)\\ &+\sum_{t}^{T}\sum_{i}^{I}\sum_{j}^{J\sim J'}\sum_{k}^{K}\text{IMT}(i)\cdot \text{IDC}(jj'k)\cdot (Y(ijkt)-(1-Y(ij'kt))\cdot Y(ijkt)) \qquad (3.2)\\ &+\sum_{t}^{T}\sum_{i}^{I}\sum_{j}^{J}\sum_{k}^{K\sim K'}\text{EMT}(i)\cdot (\text{EDC}(kk')+\text{PDC}(jk)+\text{QDC}(jk))\\ &+(Y(ijkt)-(1-Y(ijk't))\cdot Y(ijkt))\end{aligned}$$

$$\text{Min } Z_3 : W_3\sum_{t}^{T}\sum_{j}^{j}\sum_{k}^{K}\left|\omega(j,k,t,e)-\overline{\omega(j,k,t,e)}\right| \qquad (3.3)$$

s.t.

$$\sum_{k}^{K}\sum_{j}^{J}Y(ijkt)=\text{OPC}(is)\quad \forall X(it)\ \&\ i\ \&\ t \qquad (3.4)$$

$$\sum_{k}^{K}Y(ijkt)=X(it)\quad \forall t\ \&\ i, \max j\,\{j\,|\text{OPC}_{ij}=1\} \qquad (3.5)$$

$$\sum_{e=1}^{E}\omega(j,k,t,e)\leq \text{CP}(jk)\quad \forall\ t\ \&\ k\ \&\ j \qquad (3.6)$$

$$\omega(j,k,t,e)=\omega(j,k,t,(e-1))+1\,|Y(ijkt)=1\quad \forall\ t\ \&\ k\ \&\ j \qquad (3.7)$$

$$X(it)\leq D(it)\quad \forall\ t\ \&\ i \qquad (3.8)$$

$$X(it)\text{ is integer} \qquad (3.9)$$

$$Y_{ijkt}\text{ is binary.} \qquad (3.10)$$

The mathematical model presented in this study is a nonlinear mixed integer programming model. On the other hand, the multiplicity of objective functions in this model causes the model to be considered as a MODM model and therefore, the type of mathematical model presented in this study can be considered as Multi-objective Nonlinear mixed integer programming. This makes this mathematical model to be even more complicated. The first objective function of this model is for calculating system various costs. The first part of the objective function indicates the costs of loading and various operations which is done in different shops by machines. The second part of the first objective function represents the cost of intra-shop movements. This section will be explained in more detail below. The third part of the first objective function represents the external shop movement, which will be subsequently explained in the next section.

The second objective function is similar to the first objective function and represents the time of construction and transportation so that the first part indicates the operating time, the second part indicates the time of intra-shop movements and the third part indicates the time of the inter-shop movements. The third objective function also shows the degree of machines shop-load variance. As can be seen, this objective function is designed in such a way to increase the variance of a shop if the machine loading be greater than the average load in a shop.

The first constraint of this model represents the performance of products various operations according to OPC. The second constraint represents the relationship between the number of operations performed with the total number of completed products so that whenever final operations are performed according to the OPC of each product, that is, a product has been completed and therefore, semi-fabricated products will not be

Table 2. The method of calculating intra-shop movement.

| Possible state | Current machine | Next machine | $(Y(ijkt) - (1 - Y(ij'kt))Y(ijkt))$ | Result |
|---|---|---|---|---|
| The semi-finished product is located in machine $j$, $(j = 1)$ and then goes to machine $j'$, $(j' = 1)$ | $J = 1$ | $J' = 1$ | $(1-(1-1)\cdot 1) = 1$ | An intra-shop movement unit will be computed from $j$ to $j'$. |
| The semi-finished product is in machine $j$, $(j = 1)$ but will not go to machine $j'$, $(j' = 0)$ | $J = 1$ | $J' = 0$ | $(1-(1-0)\cdot 1) = 0$ | Intra-shop movement will not be calculated. |
| The semi-finished product is not in machine $j$, $(j = 0)$ and will not go to machine $j'$, $(j' = 0)$ | $J = 0$ | $J' = 0$ | $(0-(1-0)\cdot 0) = 0$ | Intra-shop movement will not be calculated. |
| The semi-finished product is not in machine $j$, $(j = 0)$ but it will be sent to machine $j'$, $(j' = 1)$ | $J = 0$ | $J' = 1$ | $(0-(1-1)\cdot 0) = 0$ | Intra-shop movement will not be calculated. |

computed to satisfy the demand for the period. The third constraint indicates that the service level of each machine in each shop should not be greater than the capacity of the machine. The fourth constraint is specified for determining the loading number of each machine in each shop, so that if one operation is assigned to a machine, the machine loading number becomes one unit greater. Fifth constraint shows that the amount of products which are produced in each period should not exceed the demand for the course. Sixth constraint determines the number of times each machine is loaded. The seventh constraint is to determine the range of variables. In this research, the variables of type $X$ and $\omega$ are in the form of integer variables and type $Y$ variables are zero and one.

## 3.2. The way of intra-shop movements calculation

The way of intra-shop movements calculation is based on the vertical and horizontal movements from a machine to another one. As regards, intra-shop movement calculation from a machine to itself is not possible, this machine is included in the term sigma of the domain machine $j \sim j'$. In this statement, the amount of movement from one machine to another machine will be presented by $\mathrm{IDC}(jj'k)$ and the cost of movement of each half-finished of product $i$ will be represented by $\mathrm{IMC}(i)$. Therefore, the $\mathrm{IMC}(i) \cdot \mathrm{IDC}(jj'k)$ indicates the cost of moving from the machine $j$ to the machine $j'$. In the following, if the product of the machine $j$ moves to $j'$, the expression $Y(ijkt) - (1 - Y(ij'kt))$ can show it because the range of movements is specified in the Table 2.

As can be seen, the following statement will only be able to calculate the possible intra-shop movements and, on that basis, calculate the cost of intra-shop movements.

$$\sum_{t}^{T}\sum_{i}^{I}\sum_{j}^{J\sim J'}\sum_{k}^{K}\mathrm{IMC}(i)\cdot\mathrm{IDC}(jj'k)\cdot(Y(ijkt)-(1-Y(ij'kt))\cdot Y(ijkt)). \tag{3.11}$$

It should be noted that calculating the time of intra-shop material movement is also calculated in this way.

TABLE 3. The method of calculating inter-shop movement.

| Possible state | Current shop | Next shop | $(Y(ijkt) - (1 - Y(ijk't)) \cdot Y(ijkt))$ | Result |
|---|---|---|---|---|
| The half-built product is located in the shop $K$, $(K = 1)$ and then goes to the shop $K'$, $(K' = 1)$ | $K = 1$ | $K' = 1$ | $(1-(1-1)\cdot 1) = 1$ | An inter-shop movement unit will be calculated from $K$ to $K'$. |
| The half-built product is located in the shop $K$, $(K = 1)$ but will not go to the shop $K'$, $(K' = 0)$ | $K = 1$ | $K' = 0$ | $(1-(1-0)\cdot 1) = 0$ | The inter-shop movement will not be calculated. |
| The half-built product is not located in shop $K$, $(K = 0)$ and will not go to the shop $K'$, $(K' = 0)$ | $K = 0$ | $K' = 0$ | $(0-(1-0)\cdot 0) = 0$ | The inter-shop movement will not be calculated. |
| The half-built product is not in shop $K$, $(K = 0)$ but will be sent to the shop $K'$, $(K' = 1)$ | $K = 0$ | $K' = 1$ | $(0-(1-1)\cdot 0) = 0$ | The inter-shop movement will not be calculated. |

## 3.3. The way of calculating inter-shop movements

In some cases that the existent machines are not enough in a shop to carry out the next service, (*e.g.*, the next required machine doesn't exist in the shop) or the existing machines be in full capacity, but the adjacent shop machines be empty, inter-shop motions can be used. For this purpose, the term EMC$(i)$ represents the cost of each unit of inter-shop movement. The way of the inter-shop motion calculation is the movement summation from a machine to the shop port, moving from one shop to another, and moving from a new shop port to the location of the new shop, which is presented by $\mathrm{EDC}(kk') + \mathrm{PDC}(jk) + \mathrm{QDC}(jk)$ is shown. In the following, if the circumstances of an inter-shop movement is shown by Table 3.

Thus, the following expression represents the inter-shop motion of the material flow.

$$\sum_{t}^{T}\sum_{i}^{I}\sum_{j}^{J}\sum_{k}^{K\sim K'} \mathrm{EMC}(i)\cdot(\mathrm{EDC}(kk') + \mathrm{PDC}(jk) + +\mathrm{QDC}(jk))\cdot(Y(ijkt) - (1 - Y(ijk't))\cdot Y(ijkt)). \quad (3.12)$$

Note that these movements will only take place when the capacity of the machines is completed in a shop and a parallel machine exists in other shops because in the normal state, the cost of intra-shop movements is much less than the inter-shop movement, and according to the type of target function, intra-shop movements will be selected at first.

## 3.4. Complexity evaluation of the developed mathematical model

After developing the mathematical model, the complexity of the model will be examined to check which algorithm is suitable for it. For this purpose the method that is applied by Delgoshaei and Gomes [7] is used. In this method, number of the feasible and infeasible basic points of the model will be calculated for a small size case study. On this basis, a company which produces 20 products, 10 machine types, 15 cells (each cell has an average of 7 parallel machines), and 5 scheduling periods is considered. Tables 4 and 5 show the number of variables and constraints for this model respectively.

TABLE 4. The number of variables in the mathematical model presented for a medium-sized problem.

| Variable | Domain | Number of variables | Result |
|---|---|---|---|
| $X(i,t)$ | $i*t$ | $20*5$ | 100 |
| $Y(i,j,k,t)$ | $i*j*k*t$ | $20*10*15*5$ | 15 000 |
| Total number | | | 25 100 |

TABLE 5. Number of constraints in the mathematical model presented for a medium-sized problem.

| Constraint | Domain | Number of constraints | Results |
|---|---|---|---|
| Constraint 1 | $t, I$ | $5*20$ | 100 |
| Constraint 2 | $t, I$ | $5*20$ | 100 |
| Constraint 3 | $t*k*j$ | $5*15*10$ | 750 |
| Constraint 4 | $t*k*j$ | $5*15*10$ | 750 |
| Constraint 5 | $t*i$ | $5*20$ | 100 |
| Constraint 6 | $i*t$ | $20*5$ | 100 |
| Constraint 7 | $i*j*k*t$ | $20*10*15*5$ | 15 000 |
| Total number | | | 16 900 |

TABLE 6. Contribution and novelties of this research.

| Reference | Mathematical model | Product demand | Objective function | Objectives | | | Solving algorithm |
|---|---|---|---|---|---|---|---|
| This research | NL-MIP | Dynamic | Multi-objective | Completion Time | System cost | Average of machine loads | NSGAII |
| Saidi-Mehrabad *et al.* [35] | MIP | Crisp | Single | Makespan | – | – | ACO |
| Li *et al.* [22] | MIP | Crisp | Multi-objective | Completion time | Total workload | Workload of the critical machine | ABC |
| Hasan *et al.* [17] | IP | Crisp | Single | Makespan | – | – | GA |

$$C_m^{n+m} = \frac{(n+m)!}{n!m!} = \frac{(25\,100 + 16\,900)!}{25\,100!16\,900!}. \tag{3.13}$$

The amount of this number is so high that even computational software such as Excel cannot calculate it. This suggests that optimization algorithms will not be able to solve this model in a medium and large dimension. Therefore, like most of the similar presented models, the heuristic and metaheuristic algorithms will be used.

## 3.5. Novelties and contributions of the presented mathematical model

As already mentioned, the mathematical model presented in this study is followed by mathematical models that have already been presented by other researchers. In Table 6, contribution and novelties of this research are compared with some of the similar researches in literature:

Table 7. Comparing the abilities of the well-known metaheuristics.

| Name | Abbreviation | Local optimum escaping | Memory | Learning | Speed | Single point | Multi point | Multi agent |
|---|---|---|---|---|---|---|---|---|
| Simulated annealing | SA | + | | | F | | + | |
| Tabu search | TS | | + | | S | | + | |
| Genetic algorithm | GA | + | + | | F | | + | + |
| Memtic algorithm | MA | + | + | | F | | + | + |
| Non dominated sorting genetic algorithm | NSGAII | + | | | F | | + | + |
| Ant colony optimization | ACO | + | | | M | | + | + |
| Bacterial forging algorithm | BEA | + | | | M | | + | + |
| Particle swarm method | PSO | | | | M | | + | + |
| Single-layer perceptron | SLP | | + | + | F | + | | |
| Multi-layer perceptron | MLP | | + | + | M | | + | + |
| Hopfield networks | HF | | | + | M | | + | + |
| Self-organized map | SOM | | | | M | + | | |
| Ranked order clustering | ROC | | | | M | + | | |

## 4. Solving algorithm: A hybrid NSSGAII and Simulated Annealing algorithm

Results of reviewing the literature show that most of the production scheduling models are NP-hard. As a result, the solving time for them increases exponentially by increasing the size of the problem. Therefore, the use of a multi objective metaheuristic algorithm is essential. In order to choose appropriate metaheuristics for solving the proposed models, Table 7 is provided which shows and compares the advantages and disadvantages of famous metaheuristics. In continue, NSGAII will be proposed for the developed model. The method which is inspired from Genetic Algorithm is developed for solving multi-objective decision making models. NSGAII is now considered as a promising way to solve multi-objective models.

The reasons to choose NSGAII is that there are many similar mathematical models in the literature that solved by the NSGAII and therefore it is a promising way for solving multi-objective models. Moreover, NSGAII uses Genetic Algorithm operators which are frequently used in the similar Job-shop Scheduling problems.

Figure 2 shows the flowchart of the proposed NSGAII-SA. Table 8 shows the pseudo code of the NSGAII-SA.

Therefore, according to the previous research, NSGAII-SA is proposed to solve the developed model. The steps in this method are as follows:

**Step 1.** This method works based on the principles of the genetic algorithm. Therefore, in the first step, the number of inputs should be adjusted based on the decision making items, as follows:

**Algorithm parameters.** Number of generations; Population; Mutation Rate.

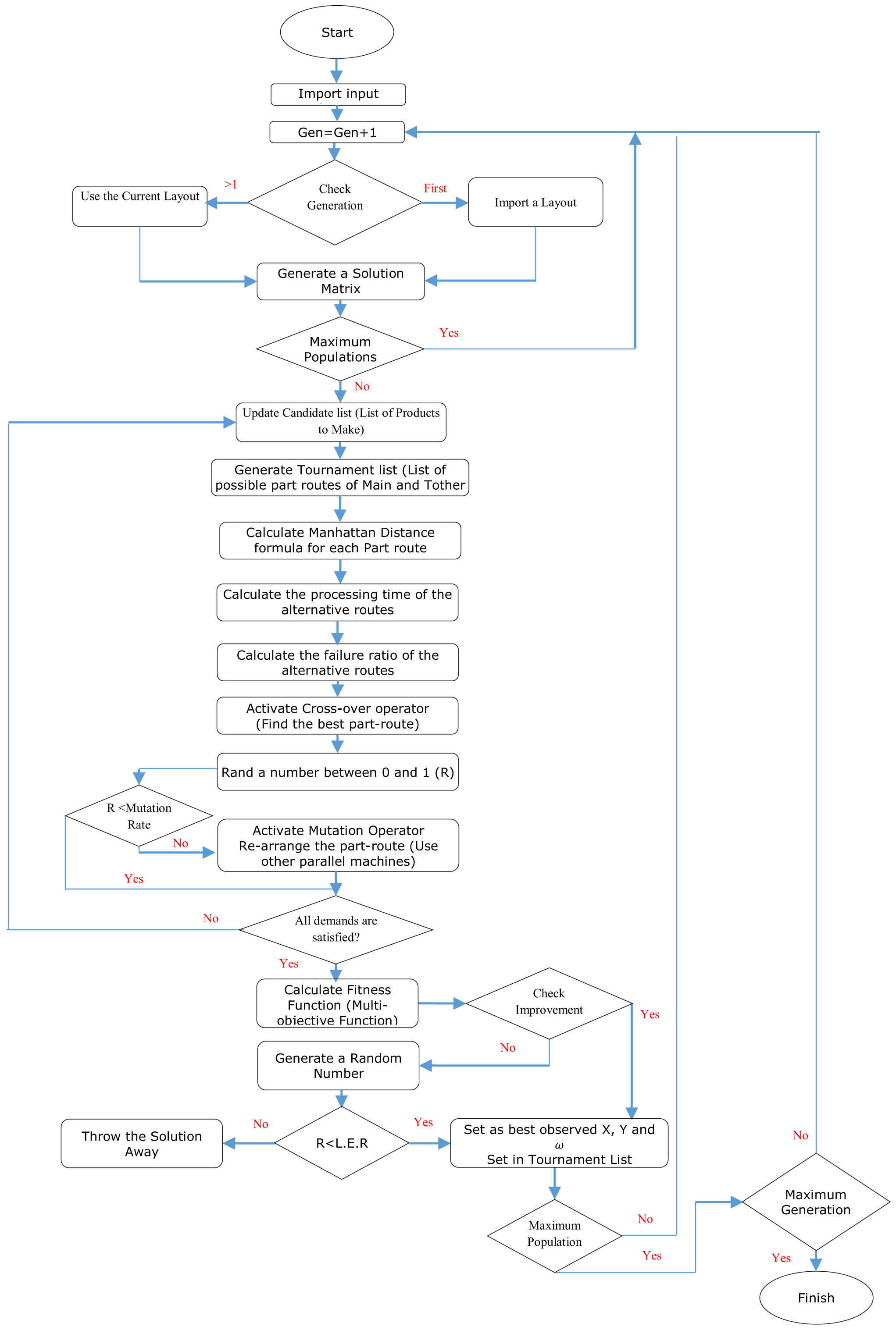
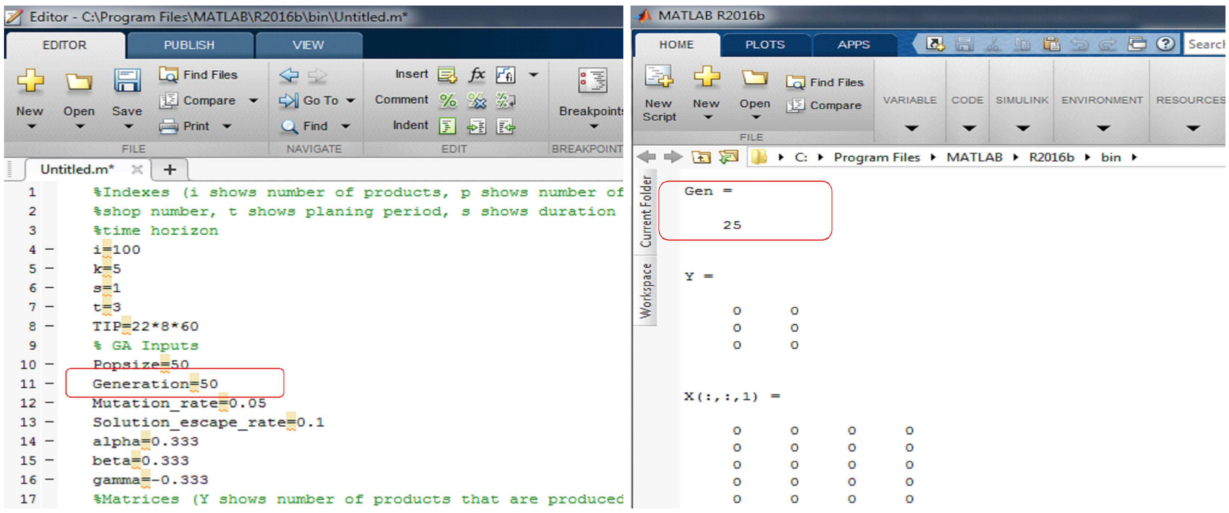

Start
Import input
Gen=Gen+1
Check Generation
>1
First
Use the Current Layout
Import a Layout
Generate a Solution Matrix
Maximum Populations
Yes
No
Update Candidate list (List of Products to Make)
Generate Tournament list (List of possible part routes of Main and Tother
Calculate Manhattan Distance formula for each Part route
Calculate the processing time of the alternative routes
Calculate the failure ratio of the alternative routes
Activate Cross-over operator (Find the best part-route)
Rand a number between 0 and 1 (R)
R <Mutation Rate
No
Yes
Activate Mutation Operator Re-arrange the part-route (Use other parallel machines)
All demands are satisfied?
No
Yes
Calculate Fitness Function (Multi-objective Function)
Check Improvement
Yes
No
Generate a Random Number
R<L.E.R
No
Yes
Throw the Solution Away
Set as best observed X, Y and ω Set in Tournament List
Maximum Population
No
Yes
Maximum Generation
No
Yes
Finish

FIGURE 2. Flowchart of the proposed hybrid method for solving the model.

Table 8. Pseudo code of the proposed NSGAII-SA algorithm.

```
Step 1) Insert Dataset
    ▪ Input data sets ($TH$; $D(it)$; $L$; $OPC(is)$; $SC(i,j)$; $IMC(i)$; $EMC(i)$; $IMT(i)$; $EMT(i)$; $OT(sj)$; $SC(i)$;
      $OC(sj)$; $IDC(jj'k)$; $EDC(kk')$; $PDC(jk)$; $QDC(jk)$; $CP(jk)$)
    ▪ Initialize pop.size, k, G, Mu & LEP (pop.size: population size; k= neighbourhood radius; Mu: mutation rate; G: number of
      generations; LEP=local escape probability)
While n<G & for m=1: pop.size
Step 2)
    ▪ if n=1
    ▪ Generate/Import a layout
    ▪ Else
Step 3)
    ▪ Choose a member from candidate list
    ▪ Find the required operations from $OPC_{is}$
    ▪ find $tournament\ list_i$ (list of the parallel machines that can serve the required operation)
    ▪ Activate Cross over Operator
        o Calculate cost, time and average of machine loads for the set of consecutive machines that can produce the selected
          product
        o Choose the best part route
    ▪ Call Mutation Operator
        o Rand a number ($R$: $Rand(1)$_
        o if $Y \leq Mu$
        o Re-arrange the part-route (Use other parallel machines)
        o If $p \,\&\, p' \in Tournament.list_i\ then\ w_{ipkft} = 1\ and\ w_{ip'kft} = 0$; DO $w_{ipkft} = 0\ and\ w_{ip'kft} = 1$
Step 4)
    ▪ Calculate Remained $D_{it}$
    ▪ if Remained $D_{it} > 0$
    ▪ Go Step 2
Step 5)
    ▪ Calculate Objective Functions
    ▪ $MinZ_1: W_1.\sum_t^T \sum_i^I \sum_s^S \sum_k^K \sum_j^J SC(i,j).OC(sj).Y(ijkt).OPC(is) + \sum_t^T \sum_i^I \sum_j^{J\sim J'} \sum_k^K IMC(i).IDC(jj'k).\Big(Y(ijkt) -$
      $\big(1 - Y(ij'kt)\big).Y(ijkt)\Big) + \sum_t^T \sum_i^I \sum_j^J \sum_k^{K\sim K'} EMC(i).\big(EDC(kk') + PDC(jk) + QDC(jk)\big).\Big(Y(ijkt) -$
      $\big(1 - Y(ijk't)\big).Y(ijkt)\Big)$
    ▪ $MinZ_2: W_2.\sum_t^T \sum_i^I \sum_s^S \sum_k^K \sum_j^J OT(sj).Y(ijkt).OPC(is) + \sum_t^T \sum_i^I \sum_j^{J\sim J'} \sum_k^K IMT(i).IDC(jj'k).\Big(Y(ijkt) -$
      $\big(1 - Y(ij'kt)\big).Y(ijkt)\Big) +$
              $\sum_t^T \sum_i^I \sum_j^J \sum_k^{K\sim K'} EMT(i).(EDC(kk') + PDC(jk) + QDC(jk)).\Big(Y(ijkt) - \big(1 - Y(ijk't)\big).Y(ijkt)\Big)$
  $\mathrm{MinZ_3: W_3} \sum_t^T \sum_j^j \sum_k^K \left|\omega(j,k,t,e) - \overline{\omega(j,k,t,e)}\right|$
    ▪ $Normalize\ Each\ Objective\ Function$
    ▪ Calculate Multi Weighting Objective Function ($MOFV = \omega_1.Z_1 + \omega_2.Z_2 + \omega_3.Z_3$)
Step 6) If MOFV>mean (MOFV)
        o List Solution in the Best Solution
        o $OFV^{best}$ =$OFV_i$
        o $STR^{best} = {STR_k}^{itr}*$
        o $Tournament\ list_i$=$STR_i$
    ▪ else
    ▪ Call Simulated Annealing Local Escape Operator
        o Rand a Number (R)
        o if R<L.E.R
        o List Solution in the Best Solution
        o $Tournament\ list_i$=$STR_i$
    ▪ Calculate $OFV_i$
    ▪ if $OFV_i \leq min_{itr}(OFV)$
        o $Tournament\ list_i$=$STR_i$
        o $OFV^{best}$ =$OFV_i$ & $STR^{best} = {STR_k}^{itr}$
        o save $Candidate.list({STR_k}^{itr})$
Step 7) Check Stopping Criteria
```



```
%Indexes (i shows number of products, p shows number of
%shop number, t shows planing period, s shows duration
%time horizon
i=100
k=5
s=1
t=3
TIP=22*8*60
% GA Inputs
Popsize=50
Generation=50
Mutation_rate=0.05
Solution_escape_rate=0.1
alpha=0.333
beta=0.333
gamma=-0.333
%Matrices (Y shows number of products that are produced
```

```
Gen =

    25


Y =

     0     0
     0     0
     0     0


X(:,:,1) =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0
```

Figure 3. Number of generations in NSGAII (*left image*)/results of running in MATLAB (*right image*).

**Model parameters.** Number of shops; Number of machines in each shop; Number of products; Operational Process Chart; Machines capacity; Production time and Cost of operation.

**Step 2.** In this step, the algorithm will generate or imports a layout.

**Step 3.** In this step of the algorithm, a product for the scheduling of successive processes must be selected. In the proposed algorithm, this choice is based on the time, cost, and load of a machine. In other words, those products that are less costly and time consuming should be selected first. It should also be noted that this is in accordance with the multi-objective model.

**Step 4.** Next, the algorithm will use the intersection operator to select the best set of machines for the related process operations. To achieve this goal, the NSGA will select a set of machines that have the highest capacity and the lowest cost and time of operation.

**Step 5.** While different types of machines are selected in different shops to complete a product, the algorithm will calculate the suitability function. If the new solution is better or equal to the previous one, then the algorithm considers the generated solution as a new member of this generation.

**Step 6.** Even if the new solution does not improve the amount of target observed, there should be some chance according to a solution for the next generation. This issue can be expressed by the mutation operator, which is also necessary, to get rid of local optimal solutions in the future.

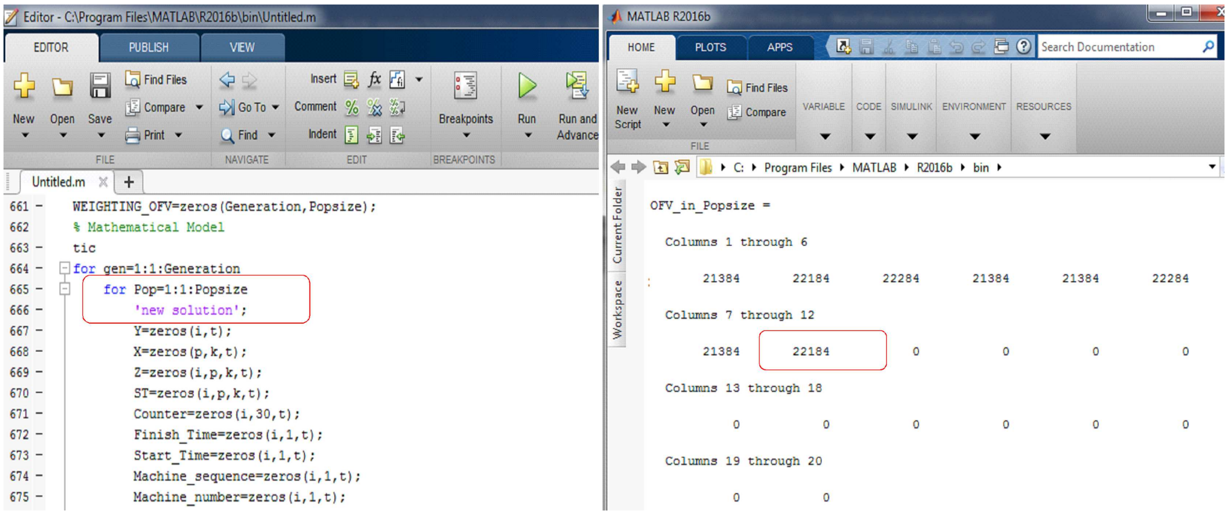
**Step 7.** Finally, at each step, the algorithm will check the stop criteria (number of generations, population size and runtime). The proposed NSGAII-SA components will be further elaborated.

## 4.1. Number of generations

The number of generations is the first and most important factor in the genetic algorithm, which indicates the times when the promoted categories of populations to reach the optimal or near optimal solution. The number of generations varies from one problem to another (Fig. 3). However, input values can play a key role in choosing the number of generations. In this stage, a layout will be generated (or imported) by the algorithm.

## 4.2. Population size

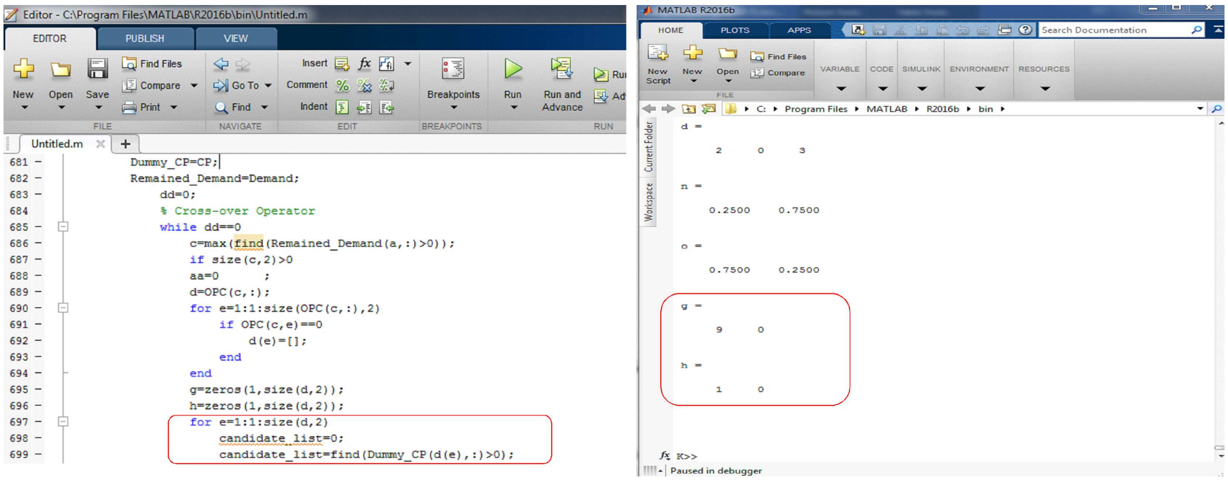
Population size is an important factor in reaching or approaching optimal solutions. Population size reflects the number of solutions that will be generated in each generations. It is obvious that the more population size,



```
WEIGHTING_OFV=zeros(Generation,Popsize);
% Mathematical Model
tic
for gen=1:1:Generation
    for Pop=1:1:Popsize
        'new solution';
        Y=zeros(i,t);
        X=zeros(p,k,t);
        Z=zeros(i,p,k,t);
        ST=zeros(i,p,k,t);
        Counter=zeros(i,30,t);
        Finish_Time=zeros(i,1,t);
        Start_Time=zeros(i,1,t);
        Machine_sequence=zeros(i,1,t);
        Machine_number=zeros(i,1,t);
```

```
OFV_in_Popsize =

  Columns 1 through 6

       21384       22184       22284       21384       21384       22284

  Columns 7 through 12

       21384       22184           0           0           0           0

  Columns 13 through 18

           0           0           0           0           0           0

  Columns 19 through 20

           0           0
```

FIGURE 4. Number of population in NSGAII (*left image*)/Results of running in MATLAB (*right image*).



```
Dummy_CP=CP;
Remained_Demand=Demand;
    dd=0;
    % Cross-over Operator
    while dd==0
        c=max(find(Remained_Demand(a,:)>0));
        if size(c,2)>0
        aa=0       ;
        d=OPC(c,:);
        for e=1:1:size(OPC(c,:),2)
            if OPC(c,e)==0
                d(e)=[];
            end
        end
        g=zeros(1,size(d,2));
        h=zeros(1,size(d,2));
        for e=1:1:size(d,2)
            candidate_list=0;
            candidate_list=find(Dummy_CP(d(e),:)>0);
```

```
d =

     2     0     3


n =

    0.2500    0.7500


o =

    0.7500    0.2500


g =

     9     0


h =

     1     0


K>>
```

FIGURE 5. Cross-over operator in NSGAII (*left image*)/Results of running in MATLAB (*right image*).

the better results will be achieved in each generations but at the same time, choosing a large population size will cause increasing the computing time as well (Fig. 4). Hence, appropriate values should be considered for population size.

## 4.3. Cross over operator

Cross-over operator is an operator for improving the selected strings of genes among the solution chromosomes. Cross-over operator can be a mathematical function or a series of scripts to do a logic function. In this study, a mathematical equation is used for the cross-over operator as shown by Figure 5.

(a) Choose a member from candidate list ($P \epsilon \text{OPC}_{ik}$).

(b) Find the required operations from $\mathrm{OPC}_{ik}$. Find Tournament.list$_i$ (list of the parallel machines that can serve the required operation):

$$\{M_{pk} | C_{pk} > 0 \ \& \ \mathrm{OPC}_{ik} = 1\}. \tag{4.1}$$

(c) Calculate cost, time and load ratio for the set of consecutive machines that are required for producing the selected product.

(d) Choose the best part route:

$$\{X_{p,k,t} = 1 | P \ \epsilon \ \mathrm{OPC}_{ik} \ \& \ \text{Minimum Cost, Quality and Time}\}. \tag{4.2}$$

For example, suppose that there is a mode that machines of type 2, 3 and 4 should be used to complete a product of type A. If the capacity of the mentioned machines be 20, 30 and 40, then only 20 products of type A can be completed in the use of this series of machines.

## 4.4. Fitness function

Fitness function is an operator to evaluate the quality of the generated solutions in each generation. It is also a criteria to pass or reject a generated solution. In most of the JSSP models in the literature, the objective function can be considered as fitness function operator. Since the developed model in this research is multi-objective, the MCDM goals will be considered as fitness function (Fig. 6). Weighting is one of the most important and valid methods for solving MCDM problems. In this way, each objective function acquires a weight (which can be put by the decision maker), and then the algorithm obtains the best solutions according to these weights. In the following, the mathematical equation will be proposed for calculating the objective function based on the weighting method.

## 4.5. Mutation operator

In metaheuristic algorithms, it is very common that solutions in a generations become more and more close to each other that means the meta-heuristic algorithm is focusing on a specific area of the solution space. Although it shows the accuracy of the algorithm but at the same time it prevents algorithm to search other parts of the solution space where optimum (or near optimum solutions) may be hided. This phenomena is called early solution convergence. To avoid such shortcoming, in the genetic algorithm, the use of the mutation operator is a promising that allows algorithm to jump from a side of solution algorithm to another by changing a part of solution string (genes). Mechanism of action is shown by Figure 7:

(a) Generate a random number ($R$).

(b) If $R \leq \mathrm{Mu}$. (4.3)

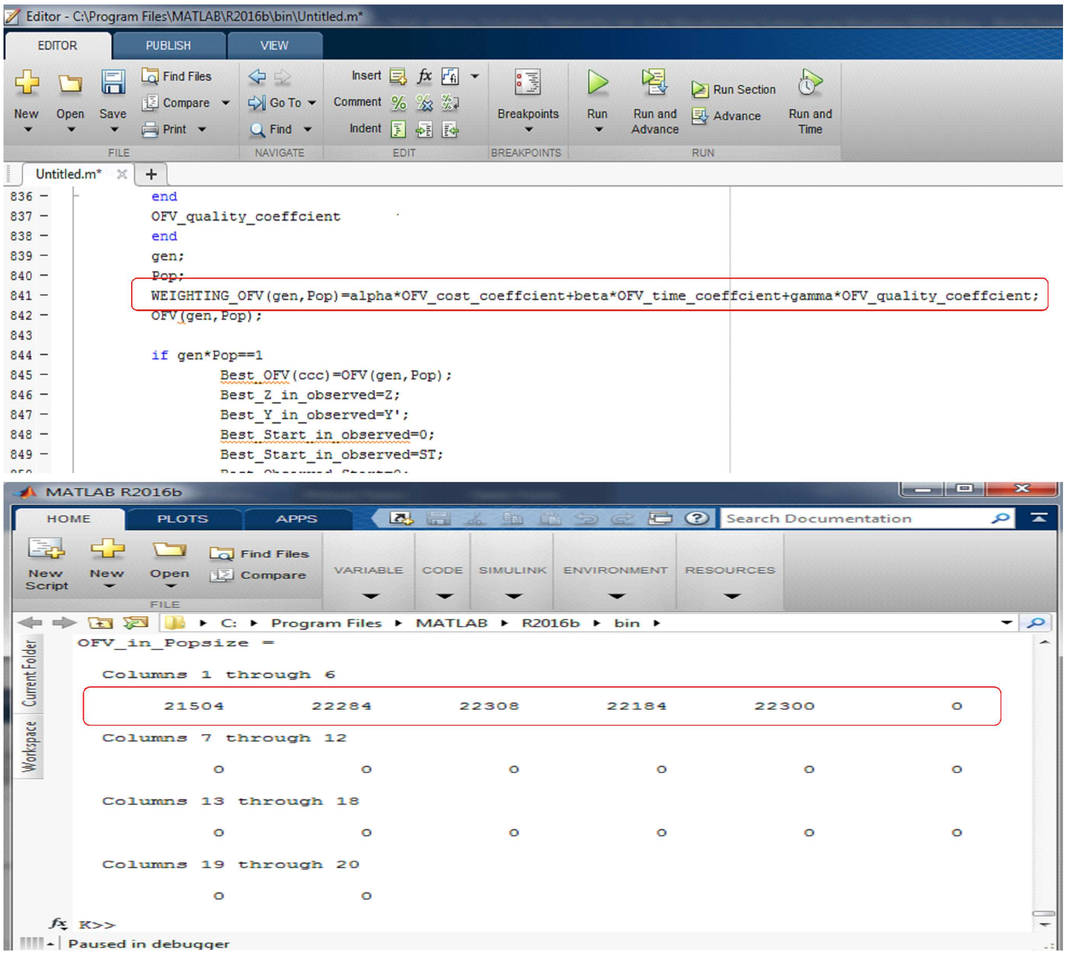
(c) Re-arrange the part-route (use other parallel machines).

(d) If $p$ & $p' \in$ Tournament.list$_i$ then $w_{ipkft} = 1$ and $w_{ip'kft} = 0$; (4.4)

DO $w_{ipkft} = 0$ and $w_{ip'kft} = 1$. (4.5)

## 4.6. Local optimum escaping operator

Falling into local optimum traps is a big concern in optimizing problems. In such situations, since a found solution is better than its neighbors, it is considered as a optimum (or near optimum) solution. However, such solution might be a local optimum solution. To overcome this shortcoming, the proposed NSGAII is using the ability of Simulated Annealing algorithm to escape from local optimum traps (Fig. 8). After calculating the fitness function for the developed solution in iterations (say $X_k^{\mathrm{itr}}$), the NSGAII-SA algorithm checks them with the best observed value achieved so far ($F^{\mathrm{best;itr}-1}$). If the fitness function value ($F_k^{\mathrm{itr}}$) is less than the $F^{\mathrm{best;itr}-1}$



```
    end
    OFV_quality_coeffcient
    end
    gen;
    Pop;
    WEIGHTING_OFV(gen,Pop)=alpha*OFV_cost_coeffcient+beta*OFV_time_coeffcient+gamma*OFV_quality_coeffcient;
    OFV(gen,Pop);

    if gen*Pop==1
            Best_OFV(ccc)=OFV(gen,Pop);
            Best_Z_in_observed=Z;
            Best_Y_in_observed=Y';
            Best_Start_in_observed=0;
            Best_Start_in_observed=ST;
```

```
OFV_in_Popsize =

  Columns 1 through 6

       21504        22284        22308        22184        22300            0

  Columns 7 through 12

           0            0            0            0            0            0

  Columns 13 through 18

           0            0            0            0            0            0

  Columns 19 through 20

           0            0

K>>
```

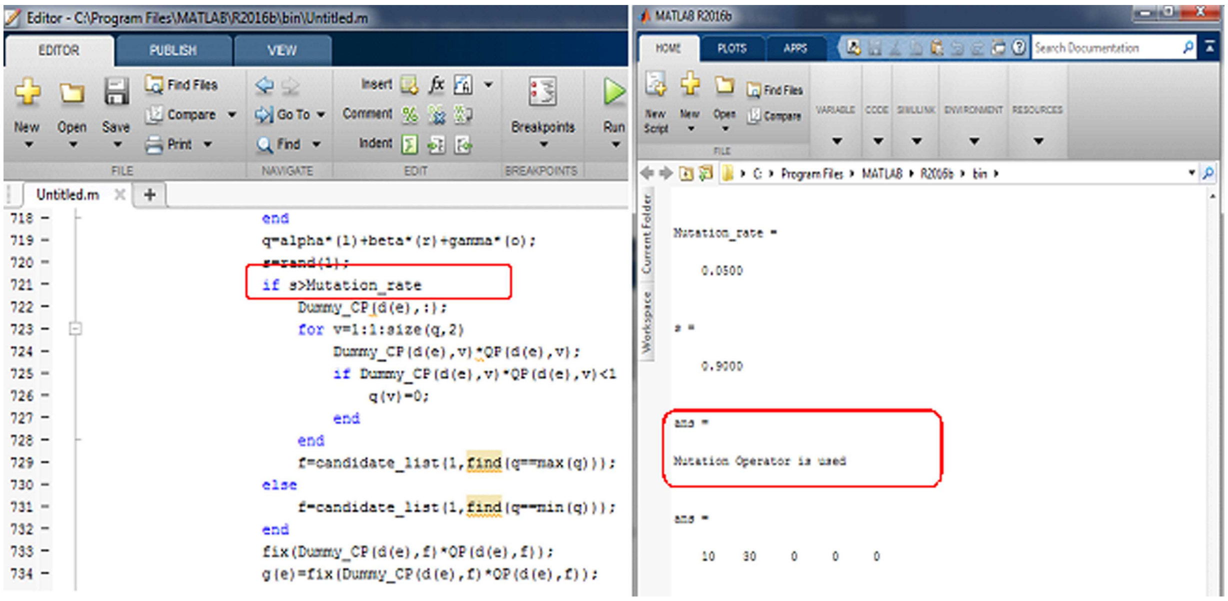
FIGURE 6. Calculating multi-objective weighting method in NSGAII (*above image*)/Results of running in MATLAB (*below image*).

NSGAII-SA replaces the $X_k^{\text{itr}}$ with $X^{\text{best;itr}-1}$. But at the same time if the value of $F_k^{\text{itr}}$ is more than $F^{\text{best;itr}-1}$ it will be withdrawn immediately.

As shown by Figure 8, in proposed method, even after achieving the worse fitness function, the algorithm provides a base to keep them with a small probability. Such strategy lets the algorithm to keep searching to find better solutions as shown by Figure 8. Such local escaping operator is added to the algorithm by using a function which described:

$$F^{\text{best;itr}} = \begin{cases} F_k^{\text{itr}}; & \text{if } F_k^{\text{itr}} \leq \min\{F_i^{\text{itr}}, F^{\text{best;itr}-1}\} \quad \forall i \in \text{itr} \\ F_k^{\text{itr}}; & \text{if } R \leq \text{LER} \\ F^{\text{best;itr}-1}; & \text{otherwise} \end{cases} \tag{4.6}$$

where $R$ is a normal random number between $(0, 1)$ and LER is a local escaping rate which is defined by decision maker. Note that the exact amount of LER cannot be determined and may be different from case to case but it can be approximately estimated using design of experiments.



```
        end
        q=alpha*(l)+beta*(r)+gamma*(o);
        s=rand(1);
        if s>Mutation_rate
            Dummy_CP(d(e),:);
            for v=1:1:size(q,2)
                Dummy_CP(d(e),v)*OP(d(e),v);
                if Dummy_CP(d(e),v)*OP(d(e),v)<1
                    q(v)=0;
                end
            end
            f=candidate_list(1,find(q==max(q)));
        else
            f=candidate_list(1,find(q==min(q)));
        end
        fix(Dummy_CP(d(e),f)*OP(d(e),f));
        g(e)=fix(Dummy_CP(d(e),f)*OP(d(e),f));
```

```
Mutation_rate =

    0.0500


s =

    0.9000


ans =

Mutation Operator is used


ans =

    10    30     0     0     0
```

FIGURE 7. Mutation operator in NSGAII (*left image*)/Results of running in MATLAB (*right image*).

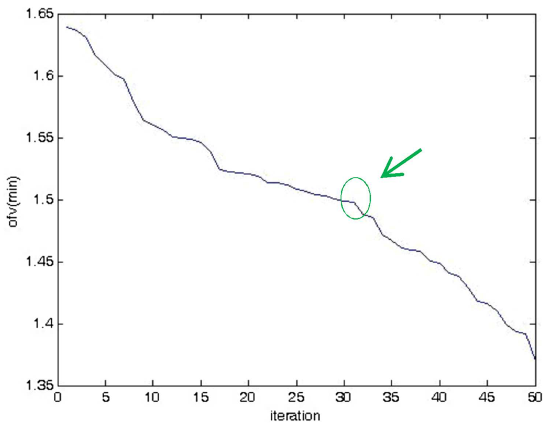


FIGURE 8. Sample of escaping from local optimum traps by using local optimum escaping operator.

## 4.7. Stopping criteria

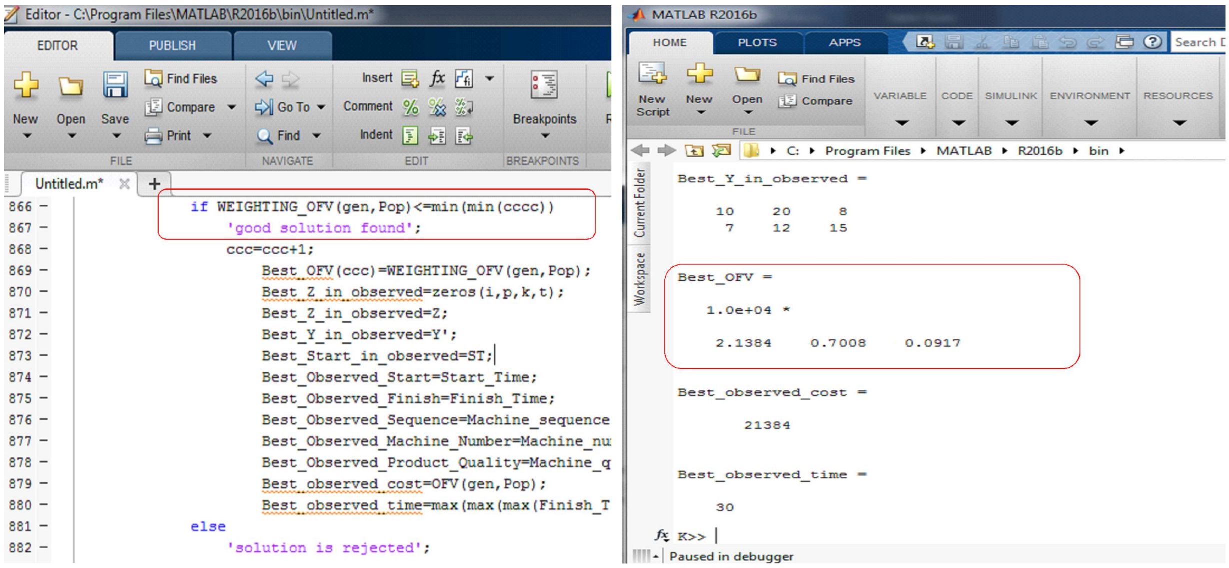
The stopping criteria in the proposed NSGAII-SA algorithm set as (Fig. 9):

(1) Reaching to maximum number of pre-defined iterations.
(2) If there is no choice in tournament list in iteration which means none of the solutions in the iteration can improve the fitness function so there will be no choice for improving the algorithm.

   (a) Suppose $X_k^{\text{itr}}$ is the $k$th solution in itrth iteration.

   (b) If $F(X_k^{\text{itr}}) > \min(F(X_1^{\text{itr}}),\ F(X_2^{\text{itr}}), \ldots, F(X_{k-1}^{\text{itr}}); F^*(X_k^{\text{itr}-1}));\ \forall\ k \in i.$ (4.7)

   Then $\text{Tournament.list}^{\text{itr}} = \emptyset$ for next iteration. (4.8)



```
            if WEIGHTING_OFV(gen,Pop)<=min(min(cccc))
                'good solution found';
                ccc=ccc+1;
                    Best_OFV(ccc)=WEIGHTING_OFV(gen,Pop);
                    Best_Z_in_observed=zeros(i,p,k,t);
                    Best_Z_in_observed=Z;
                    Best_Y_in_observed=Y';
                    Best_Start_in_observed=ST;
                    Best_Observed_Start=Start_Time;
                    Best_Observed_Finish=Finish_Time;
                    Best_Observed_Sequence=Machine_sequence
                    Best_Observed_Machine_Number=Machine_nu
                    Best_Observed_Product_Quality=Machine_q
                    Best_observed_cost=OFV(gen,Pop);
                    Best_observed_time=max(max(max(Finish_T
            else
                'solution is rejected';
```

```
Best_Y_in_observed =

    10    20     8
     7    12    15


Best_OFV =

   1.0e+04 *

    2.1384    0.7008    0.0917


Best_observed_cost =

       21384


Best_observed_time =

    30

K>>
```

FIGURE 9. Termination operator in NSGAII (*left image*)/Results of running in MATLAB (*right image*).

TABLE 9. Initial estimation for levels of factors.

| Algorithm | Factor | Small scale | | | Medium scale | | | Large scale | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $L_1$ | $L_2$ | $L_3$ | $L_1$ | $L_2$ | $L_3$ | $L_1$ | $L_2$ | $L_3$ |
| GA | Number of generations | 30 | 50 | 80 | 50 | 100 | 150 | 50 | 100 | 200 |
| | Population size | 20 | 30 | 50 | 30 | 50 | 80 | 50 | 80 | 100 |
| | Mutation rate | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 |
| | Local escaping rate | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 | 0.1 | 0.2 | 0.3 |

## 5. DESIGN OF EXPERIMENTS

In this section design of experiments is used to estimate appropriate values for the elements of the NSGAII-SA. The propose NSGAII-SA has 4 main elements which are number of generations, population size, mutation rate and local escaping rate. Hence, an $L_9(4\hat{}3)$ orthogonal optimization in Taguchi method is taken into account. In this research due to complexity of the mathematical model and the solving algorithm it is very hard to estimate a comprehensive mathematical formula for each of the input parameters. Moreover, the results of using such formulas in complex conditions may not be trustful. Therefore, as used by many other scientists, for each parameter, the models are run for a range of values while other parameters are fixed at maximum point and then the most reasonable value will be chosen by comparing the observed objective function. The best estimated values for each parameter of metaheuristics are shown by the Table 9.

The levels of factors, which are shown by Table 10 are then used for DOE in order to find the best estimating value for each parameter, significant parameters and interactions between them.

Upon implementing the experiments designed for the Taguchi method (Tab. 10), the obtained results show that the algorithm is sensitive to the number of generations, population size, local escaping rate and mutation rate respectively. Hence, the appropriate ranges for parameters can be set as what shown by Table 11.

Table 10. Results of implementing the $L_9(4\hat{}3)$ experiments for Taguchi method for NSGAII-SA.

| Experiment number | Factor | | | | OFV |
|---|---|---|---|---|---|
| | Generations (A) | Population size (B) | Mutation rate (C) | Local escaping rate (D) | |
| 1 | 1 | 1 | 1 | 1 | 469 |
| 2 | 1 | 2 | 2 | 2 | 5239 |
| 3 | 1 | 3 | 3 | 3 | 3334 |
| 4 | 2 | 1 | 2 | 2 | 4763 |
| 5 | 2 | 2 | 3 | 3 | 5137 |
| 6 | 2 | 3 | 1 | 1 | 533 |
| 7 | 3 | 1 | 3 | 3 | 5070 |
| 8 | 3 | 2 | 1 | 1 | 1297 |
| 9 | 3 | 3 | 2 | 2 | 4869 |

Table 11. An estimation for the inputs of the proposed metaheuristic.

| Algorithm | Factor | Small scale | Medium scale | Large scale |
|---|---|---|---|---|
| NSGAII-SA | Number of generations | 80 | 100 | 150 |
| | Population size | 50 | 80 | 80 |
| | Mutation rate | 0.1 | 0.3 | 0.3 |
| | Local escaping rate | 0.2 | 0.2 | 0.3 |

## 6. Results and discussion

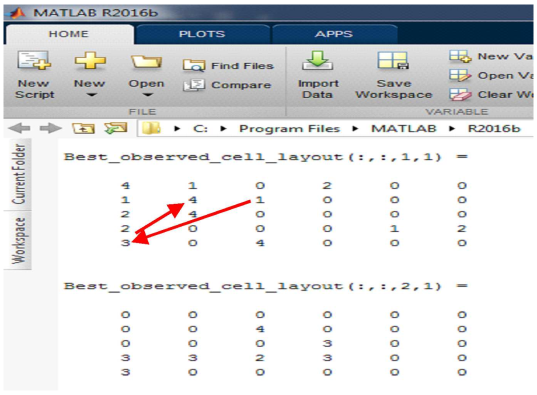
### 6.1. Solving small size problem using NSGAII-SA

In order to confirm the operation of the mathematical algorithm, data from a heavy vehicles parts production factory located on the Tehran-Qazvin highway were used. In part of the plant, 4 types of the products related to the heavy vehicles chassis in 2 shops are produced. In these two large shops there are 20 machines of cutting, bending, welding and coloring, which are placed in the factory traditionally. These machines are coded in numbers 1 to 4 in the mathematical algorithm.

Initial information obtained from problem in the mathematical algorithm was coded as follows:

```
i=4
j=4
k=2
t=4
length=5
width=6
CL=length*width
NOM=[4 5 6 5]
neighbour_radius=2.
```

The data of the integrated algorithm of the ant colony and the simulated cooling which was obtained from the experimental design part were considered as follows:



```
Best_observed_cell_layout(:,:,1,1) =

     4     1     0     2     0     0
     1     4     1     0     0     0
     2     4     0     0     0     0
     2     0     0     0     1     2
     3     0     4     0     0     0


Best_observed_cell_layout(:,:,2,1) =

     0     0     0     0     0     0
     0     0     4     0     0     0
     0     0     0     3     0     0
     3     3     2     3     0     0
     3     0     0     0     0     0
```

FIGURE 10. The layout of company.

```
Generations=80
Popsize=50
Local_escape=0.1
Mutation Rate=0.2.
```

In addition, other information needed for problem solving was extracted from the organization and coded as follows:

```
OPC=[1 1 0 1;
     1 0 1 1;
     0 1 1 1;
     1 1 0 1].
machine_capacity=[900 900 800 800]
op=[14 12 16 14]
setup=[220 150 320 180]
inter-shop_cost=[6 8 12 9]
intra-shop_cost=[2 3 4 6]
inter-shop_time=[2 3 2 2]
intra-shop_time=[1 2 2 3]
Lost_sale_cost=[100 200 50 80]
SETUP_TIME=[5 4 6 5]
OP_TIME=[12 5 7 5].
```

In this sample, the demand for products was estimated by the average parameters and variance by the production planning unit and based on sales data.

```
d_mean = [221 165 185 192;      d_sigma = [2 5 5 9;       d =[219 161 190 189
           200 130 170 155;                 2 3 7 5;          198 135 175 155
           300 150 160 140;                 3 5 6 1;          300 152 158 142
           130 140 150 135]                 1 4 1 5]          131 131 150 140].
```

Figure 10 represents the layout of the company. The first issue after the problem solving by the algorithm is the time calculation. In this case, the time calculation is about 78 s, which is evaluated appropriately.

```
Elapsed time is 78.799 s.
```

In the following, the best values of the produced products inside the shops were calculated by the "Best_observed_Inhouse_manufacturing" matrix as follows. Also, due to the impossibility of producing products

TABLE 12. The obtained part route.

| Selected machines order | | | | Planning period | Product type | Production volume |
|---|---|---|---|---|---|---|
| Machine 1 | Machine 2 | Machine 3 | Machine 4 | | | |
| 12 | 5 | 7 | 0 | 1 | 2 | 32 |
| 6 | 3 | 7 | 0 | 1 | 4 | 32 |
| 12 | 4 | 15 | 0 | 1 | 4 | 32 |
| 16 | 5 | 15 | 0 | 1 | 3 | 26 |
| 4 | 5 | 8 | 0 | 1 | 3 | 26 |
| 12 | 4 | 15 | 0 | 1 | 1 | 30 |
| 2 | 5 | 8 | 0 | 1 | 2 | 32 |
| 3 | 5 | 15 | 0 | 1 | 3 | 26 |
| 6 | 16 | 42 | 0 | 1 | 1 | 30 |
| 2 | 3 | 8 | 0 | 1 | 4 | 32 |
| 4 | 35 | 7 | 0 | 1 | 3 | 26 |
| 44 | 48 | 8 | 0 | 1 | 3 | 26 |
| 6 | 16 | 42 | 0 | 1 | 1 | 30 |
| 2 | 3 | 8 | 0 | 1 | 1 | 30 |
| 2 | 3 | 1 | 0 | 1 | 4 | 32 |
| 12 | 44 | 15 | 0 | 1 | 4 | 32 |
| 24 | 4 | 7 | 0 | 1 | 1 | 30 |
| 2 | 4 | 7 | 0 | 1 | 1 | 30 |
| 6 | 48 | 42 | 0 | 1 | 2 | 32 |
| 12 | 44 | 15 | 0 | 1 | 4 | 29 |

by external suppliers, these values are calculated and equaled to zero according to the below matrix.

```
Best_observed_Inhouse_manufacturing = Best_observed_Out_source =
219 161 190 189                       0 0 0 0
198 135 175 155                       0 0 0 0
300 152 158 142                       0 0 0 0
131 131 150 140                       0 0 0 0.
```

In the following, the best sequencing paths of the product operations (flow of materials) were also shown (Tab. 12). In each case, the amount of loading on a machine is equal to the minimum remaining capacity of the machine and the amount of half-finished products reached by the machine (from the previous machine).

As shown by Figure 11, the algorithm starts the value of the target functions from 1.2 and then reaches to 0.78 after 80 repetitions. The downside slope of the algorithm represents the high power of the algorithm in reducing the target functions. And as previously stated, the existence of the relative extreme points (in this mathematical model, in particular, relative minimum) indicates the existence of local optimal points. According to the problem data, the proposed algorithm can solve a real-world problem in a short time.

## 6.2. Solving numerical case studies

In this section, a set of the numerical examples that are extract from the literature is solved for validating the performance of the proposed algorithm. The 12 numerical examples, where data are mostly extracted from subject literature, have been selected in small, medium and large sizes (Tab. 13. The type of processor which is used has a dramatic effect on the output speed of the algorithm. So in this research, a personal computer system with Intel®Core$^{TM}$ i5 that has the CPU 2.0 GH and 4 GB RAM will be used.The results are shown by Table 14. The outcomes show that the algorithm can solve all experiments with different dimensions that have been extracted from the literature. Table 14 also compares the outcomes of the SGAII-SA with Standard

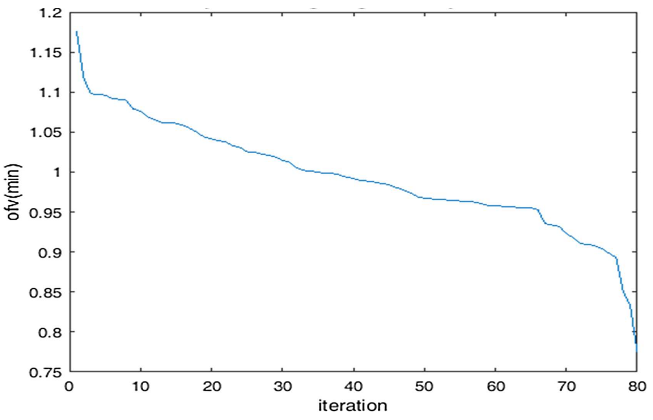


FIGURE 11. Multi-objectives reduction diagram.

TABLE 13. Input data collected from the literature for solving experiments.

| Size | No. | Problem source | K | P | M | O.P | P.P | A.S | N.O.M |
|---|---|---|---|---|---|---|---|---|---|
| Small | 1 | Askin and Huang [2] | 2 | 2 | 2 | 2 | 4 | 8 | [2 4] |
| | 2 | Askin and Huang [2] | 2 | 2 | 4 | 2 | 4 | 20 | [3 6] |
| | 3 | Suer and Cedeño [40] | 1 | 4 | 4 | 4 | 4 | 15 | [3 2 3 2] |
| | 4 | Askin and Huang [2] | 8 | 2 | 2 | 2 | 4 | 4 | [14 6] |
| Medium | 5 | Askin and Huang [2] | 8 | 2 | 2 | 2 | 4 | 4 | [8 8] |
| | 6 | Mahdavi *et al.* [25] | 2 | 4 | 4 | 4 | 2 | 20 | [4 5 3 6] |
| | 7 | Mahdavi *et al.* [26] | 2 | 4 | 4 | 4 | 4 | 16 | [4 3 6 5] |
| | 8 | Aryanezhad *et al.* [1] | 3 | 3 | 3 | 3 | 3 | 20 | [5 4 5] |
| Large | 9 | Aryanezhad *et al.* [1] | 5 | 4 | 5 | 5 | 4 | 8 | [4 3 2 3 3] |
| | 10 | Aryanezhad *et al.* [1] | 4 | 5 | 5 | 5 | 4 | 12 | [2 4 3 4 4] |
| | 11 | Li *et al.* [23] | 2 | 5 | 5 | 5 | 4 | 15 | [3 5 4 2 3] |
| | 12 | Mahdavi *et al.* [25] | 2 | 6 | 6 | 6 | 3 | 20 | [5 4 3 6 5 4] |

**Notes.** K: Shops; P: Product; M: Machine; O.P: Operations; P.P: Planning Period; A.S: Available Space; N.O.M: Number of Machines.

Genetic Algorithm. The results show that for small size problems, both algorithms reported the same results but for the medium and large scale problems NSGAII-SA always reported solutions with better quality.

The outcomes showed that the proposed algorithm is strong enough to solve all studied cases without failures. In addition, the algorithm could solve the large scale problems in less than 2 min (123 s). In order to check the performance of the algorithm the objective function value (fitness function) of each problem in the 1st generation is compared with ones that observed in the last generation. The results indicated that the algorithm can improve small size cases up to 65%, 31.2% for the medium size cases and 17% for large scale ones. In addition, in order to check the capability of the proposed algorithm, all studied cases are solved with the standard GA once again and the solutions is compared with the NSGAII-SA. The outcomes revealed that in all studied cases, NSGAII-SA reported better solutions (best OFV NSGAII-SA column in the Table 14) than classic GA (standard GA column in the Table 14).

TABLE 14. Results of solving experiments using NSGAII-SA.

| No. | Reference | Initial OFV NSGAII-SA | Best OFV NSGAII-SA | %Improvement | Standard GA | CPU time |
|---|---|---|---|---|---|---|
| 1 | Askin and Huang [2] | 2.1370 | 0.7487 | 64.96 | 0.749 | 16.399 |
| 2 | Askin and Huang [2] | 0.89 | 0.78 | 12.360 | 0.780 | 7.71 |
| 3 | Suer and Cedeño [40] | 0.747 | 0.702 | 16.024 | 0.688 | 11.915 |
| 4 | Askin and Huang [2] | 0.703 | 0.579 | 17.639 | 0.579 | 18.599 |
| 5 | Askin and Huang [2] | 0.869 | 0.728 | 16.226 | 0.725 | 18.367 |
| 6 | Mahdavi *et al.* [25] | 0.869 | 0.841 | 31.222 | 0.841 | 36.293 |
| 7 | Mahdavi *et al.* [26] | 0.922 | 0.835 | 9.436 | 0.743 | 64.632 |
| 8 | Aryanezhad *et al.* [1] | 0.823 | 0.696 | 15.431 | 0.564 | 24.723 |
| 9 | Aryanezhad *et al.* [1] | 0.827 | 0.752 | 9.069 | 0.654 | 71.691 |
| 10 | Aryanezhad *et al.* [1] | 0.79 | 0.701 | 11.266 | 0.652 | 72.143 |
| 11 | Li *et al.* [23] | 0.848 | 0.704 | 16.981 | 0.584 | 123.898 |
| 12 | Mahdavi *et al.* [25] | 0.81 | 0.694 | 14.321 | 0.548 | 120.473 |

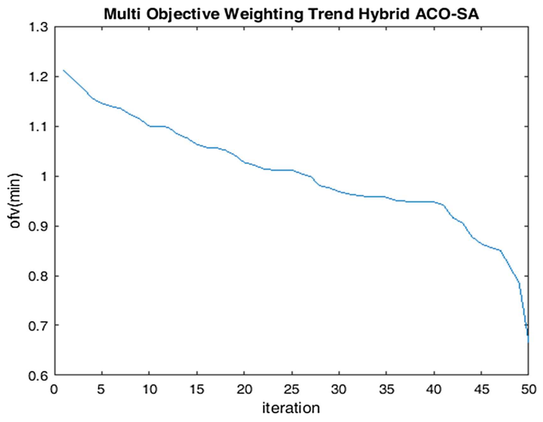


FIGURE 12. Weighting target function reduction diagram problem number 1.

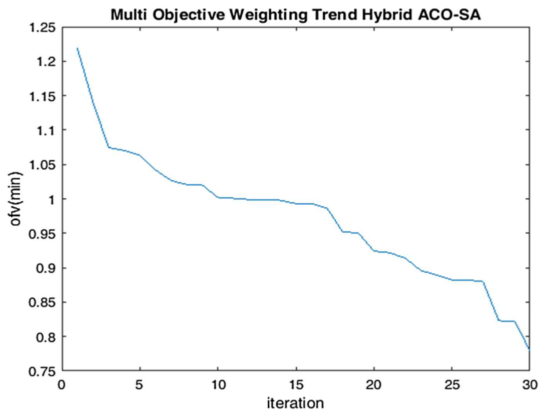


FIGURE 13. Weighting target function reduction diagram problem number 2.

### 6.3. Analysis of the reduction power of weighting target functions

Figures 12–23 show the optimization process in each of the solved experiments in the Table 14. The figures shows that the proposed hybrid NSGAII-SA can successfully reduce objective function value during searching process.

As the slope of the graphs yields, the algorithm is able to reduce the amount of weight target functions to a large extent with the appropriate slope during the determined repetitions.

## 7. CONCLUSIONS

In this paper, a mathematical model is developed for multi-objective scheduling of products in a job-shop based manufacturing system. For this purpose, a new mathematical programming model (multi-objective non-linear mixed integer programming) is developed which contains 3 opposing objectives: minimizing production costs, completion time and average of machine load. The outcome of the model can help decision makers to find best material sequences in each manufacturing period which best match to all mentioned objective functions.

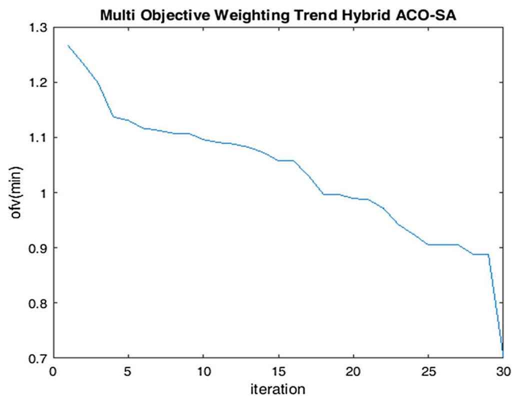


FIGURE 14. Weighting target function reduction diagram problem number 3.

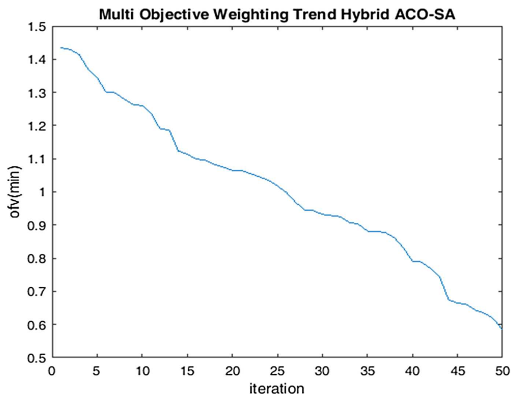


FIGURE 15. Weighting target function reduction diagram problem number 4.

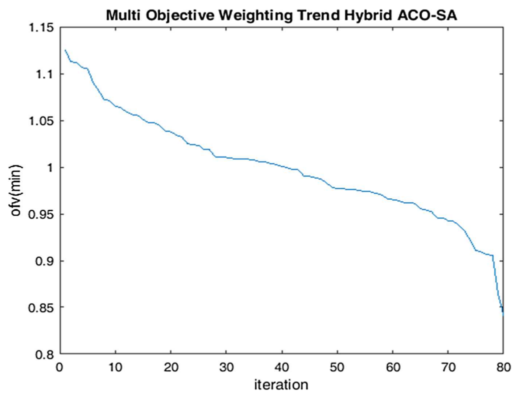


FIGURE 16. Weighting target function reduction diagram problem number 5.

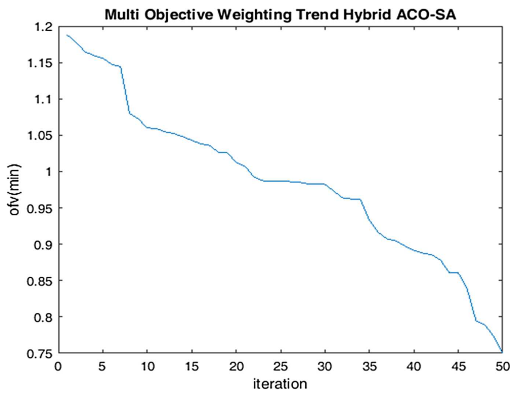


FIGURE 17. Weighting target function reduction diagram problem number 6.

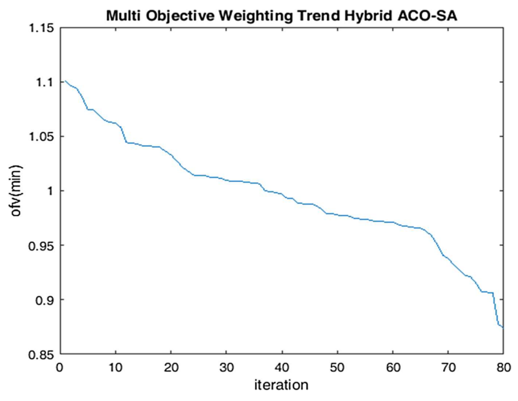


FIGURE 18. Weighting target function reduction diagram problem number 7.

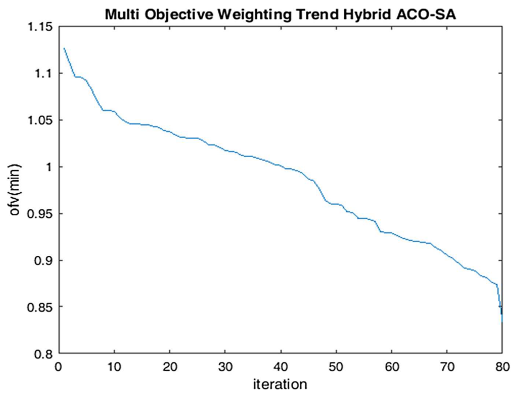


FIGURE 19. Weighting target function reduction diagram problem number 8.

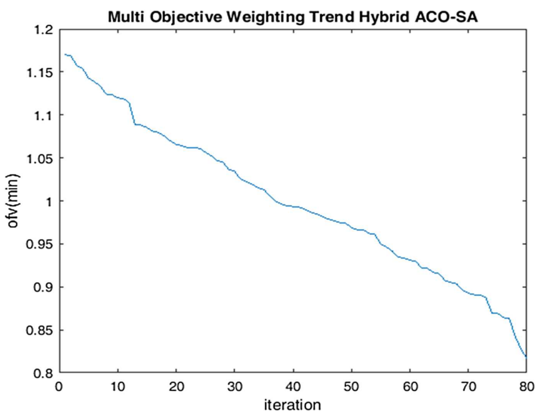


Figure 20. Weighting target function reduction diagram problem number 9.

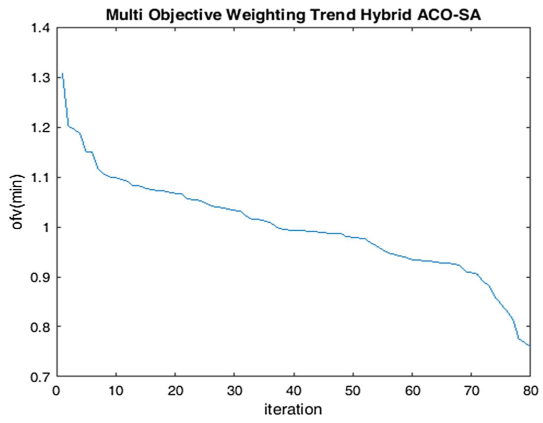


Figure 21. Weighting target function reduction diagram problem number 10.

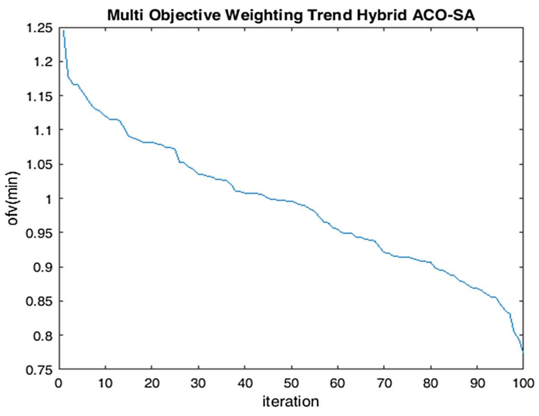


Figure 22. Weighting target function reduction diagram problem number 11.

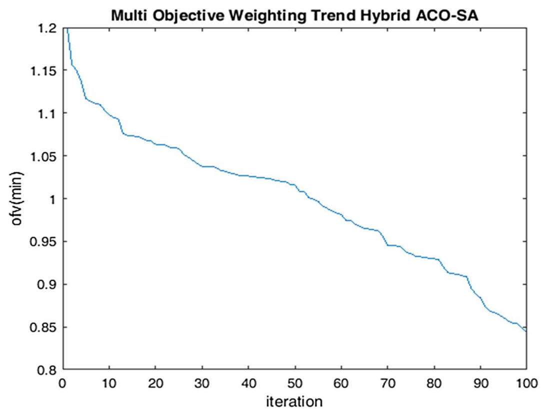


Figure 23. Weighting target function reduction diagram problem number 12.

Then, the complexity of the model is investigated that showed the model can be classified into NP-hard problems which cause using meta-heuristic algorithms accordingly. Hence, in order to solve the large-scale problem, a hybrid NSGAII and SA metaheuristic algorithm was used. In order to prove the quality of the optimal solution obtained by solving the model with the acquired algorithm, the outcomes are also compared with a classic GA. In addition, the performance of the solving algorithm for the initial and final solution strings are compared. The algorithm are then applied for small, medium and large scale problems from the literature. The results showed that the market demand uncertainty can significantly affect to the process of job shop scheduling and impose harms in manufacturing systems in terms of product completion and machine loads while operational cost are not too sensitive to product demands. In addition, the outcomes show the superiority of the NSGAII-SA in providing solutions with better quality than standard GA in a reasonable solving time.

Further expansion of the proposed model can be considered by using other metaheuristic algorithms. Moreover, integration of proposed model with production planning is suggested for future researches.

*Acknowledgements.* The authors would like to sincerely appreciate the editor and also anonymous reviewers for their constructive comments during publication process.

## Declaration of conflicting interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

## References

[1] Aryanezhad, M. B., Deljoo, V., & Mirzapour Al-e-Hashem, S. M. J., Dynamic cell formation and the worker assignment problem: a new model. *Int. J. Adv. Manuf. Tech.* **41** (2009) 329.

[2] Askin, R. G., & Huang, Y., Forming effective worker teams for cellular manufacturing. *Int. J. Prod. Res.* **39** (2001) 2431–2451.

[3] A. Banharnsakun, B. Sirinaovakul and T. Achalakul, Job shop scheduling with the best-so-far ABC. *Eng. App. Artif. Intell.* **25** (2012) 583–593.

[4] A. Baykasoğlu, A. Hamzadayi and S.Y. Köse, Testing the performance of teaching–learning based optimization (TLBO) algorithm on combinatorial problems: flow shop and job shop scheduling cases. *Inf. Sci.* **276** (2014) 204–218.

[5] J.C. Chen, C.-C. Wu, C.-W. Chen and K.-H. Chen, Flexible job shop scheduling with parallel machines using Genetic Algorithm and Grouping Genetic Algorithm. *Expert Syst. App.* **39** (2012) 10016–10021.

[6] T.-K. Dao, T.-S. Pan and J.-S. Pan, Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *J. Intell. Manuf.* **29** (2018) 451–462.

[7] A. Delgoshaei and C. Gomes, A multi-layer perceptron for scheduling cellular manufacturing systems in the presence of unreliable machines and uncertain cost. *Appl. Soft Comput.* **49** (2016) 27–55.

[8] A. Delgoshaei, M.K. Ariffin, B. Baharudin and Z. Leman, A backward approach for maximizing net present value of multi-mode pre-emptive resource-constrained project scheduling problem with discounted cash flows using simulated annealing algorithm. *Int. J. Ind. Eng. Manage.* **5** (2014) 151–158.

[9] A. Delgoshaei, M. Ariffin, B. Baharudin and Z. Leman, Minimizing makespan of a resource-constrained scheduling problem: a hybrid greedy and genetic algorithms. *Int. J. Ind. Eng. Comput.* **6** (2015) 503–520.

[10] Y. Demir and S.K. İşleyen, Evaluation of mathematical models for flexible job-shop scheduling problems. *Appl. Math. Modell.* **37** (2013) 977–988.

[11] K.-Z. Gao, P.N. Suganthan, Q.-K. Pan, T.J. Chua, T.X. Cai and C.-S. Chong, Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling. *Inf. Sci.* **289** (2014) 76–90.

[12] H. Garg, A hybrid PSO-GA algorithm for constrained optimization problems. *Appl. Math. Comput.* **274** (2016) 292–305.

[13] H. Garg, Performance analysis of an industrial system using soft computing based hybridized technique. *J. Braz. Soc. Mech. Sci. Eng.* **39** (2017) 1441–1451.

[14] H. Garg, A hybrid GSA-GA algorithm for constrained optimization problems. *Inf. Sci.* **478** (2019) 499–523.

[15] H. Garg and S. Sharma, Multi-objective reliability-redundancy allocation problem using particle swarm optimization. *Comput. Ind. Eng.* **64** (2013) 247–255.

[16] A. Hamidinia, S. Khakabimamaghani, M.M. Mazdeh and M. Jafari, A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Comput. Ind. Eng.* **62** (2012) 29–38.

[17] S.K. Hasan, R. Sarker, D. Essam and I. Kacem, A DSS for job scheduling under process interruptions. *Flexible Serv. Manuf. J.* **23** (2011) 137.

[18] B. Jinsong, H. Xiaofeng and J. Ye, A genetic algorithm for minimizing makespan of block erection in shipbuilding. *J. Manuf. Technol. Manage.* **20** (2009) 500–512.

[19] R.M. Karp, Reducibility among combinatorial problems. In: 50 Years of Integer Programming 1958–2008. Springer, Berlin-Heidelberg (2010) 219–241.

[20] S. Karthikeyan, P. Asokan, S. Nickolas and T. Page, A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems. *Int. J. Bio-Inspired Comput.* **7** (2015) 386–401.

[21] D. Lei, Co-evolutionary genetic algorithm for fuzzy flexible job shop scheduling. *Appl. Soft Comput.* **12** (2012) 2237–2245.

[22] J.-Q. Li, Q.-K. Pan and M.F. Tasgetiren, A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities. *Appl. Mathe. Modell.* **38** (2014) 1111–1132.

[23] J. Li, Q. Pan and S. Xie, An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems. *Appl. Math. Comput.* **218** (2012) 9353–9371.

[24] H. Luo, G.Q. Huang, Y. Zhang, Q. Dai and X. Chen, Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm. *Robotics and Computer-Integrated Manufacturing* **25** (2009) 962–971.

[25] S. Mahdavi, H. Kermanian and A. Varshoei, Comparison of mechanical properties of date palm fiber-polyethylene composite. *BioResources* **5** (2010) 2391–2403.

[26] I. Mahdavi, A. Aalaei, M.M. Paydar and M.A. Solimanpur, A new mathematical model for integrating all incidence matrices in multi-dimensional cellular manufacturing system. *J. Manuf. Syst.* **31** (2012) 214–223.

[27] S. Malve and R. Uzsoy, A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Comput. Oper. Res.* **34** (2007) 3016–3028.

[28] S. Meeran and M. Morshed, A hybrid genetic tabu search algorithm for solving job shop scheduling problems: a case study. *J. Intell. Manuf.* **23** (2012) 1063–1078.

[29] S. Nguyen, M. Zhang, M. Johnston and K.C. Tan, Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. *IEEE Trans. Evol. Comput.* **18** (2014) 193–208.

[30] M. Nouiri, A. Bekrar, A. Jemai, S. Niar and A.C. Ammari, An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *J. Intell. Manuf.* **29** (2018) 603–615.

[31] R.S. Patwal, N. Narang and H. Garg, A novel TVAC-PSO based mutation strategies algorithm for generation scheduling of pumped storage hydrothermal system incorporating solar units. *Energy* **142** (2018) 822–837.

[32] B. Peng, Z. Lü and T. Cheng, A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Comput. Oper. Res.* **53** (2015) 154–164.

[33] S.H.A. Rahmati and M. Zandieh, A new biogeography-based optimization (BBO) algorithm for the flexible job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **58** (2012) 1115–1129.

[34] A. Rossi, Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. *Int. J. Prod. Econ.* **153** (2014) 253–267.

[35] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian and V. Mahmoodian, An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Comput. Ind. Eng.* **86** (2015) 2–13.

[36] S.M. Sajadi, A. Alizadeh, M. Zandieh and F. Tavan, Robust and stable flexible job shop scheduling with random machine breakdowns: multi-objectives genetic algorithm approach. *Int. J. Math. Oper. Res.* **14** (2019) 268–289.

[37] H. Shah, N. Tairan, H. Garg and R. Ghazali, Global Gbest guided-artificial bee colony algorithm for numerical function optimization. *Computers* **7** (2018) 69.

[38] X. Shao, W. Liu, Q. Liu and C. Zhang, Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **67** (2013) 2885–2901.

[39] X.-N. Shen and X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. *Inf. Sci.* **298** (2015) 198–224.

[40] G.A. Suer and A.A. Cedeño, A configuration-based clustering algorithm for family formation. *Comput. Ind. Eng.* **31** (1996) 147–150.

[41] Y. Wang, A new hybrid genetic algorithm for job shop scheduling problem. *Comput. Oper. Res.* **39** (2012) 2291–2299.

[42] L. Wang, G. Zhou, Y. Xu, S. Wang and M. Liu, An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **60** (2012) 303–315.

[43] Y. Xu, L. Wang, S.-Y. Wang and M. Liu, An effective teaching–learning-based optimization algorithm for the flexible job-shop scheduling problem with fuzzy processing time. *Neurocomputing* **148** (2015) 260–268.

[44] Y. Yuan and H. Xu, Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Trans. Autom. Sci. Eng.* **12** (2015) 336–353.

[45] Y. Yuan, H. Xu and J. Yang, A hybrid harmony search algorithm for the flexible job shop scheduling problem. *Appl. Soft Comput.* **13** (2013) 3259–3272.

[46] R. Zhang and R. Chiong, Solving the energy-efficient job shop scheduling problem: a multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *J. Cleaner Prod.* **112** (2016) 3361–3375.

[47] X. Zhang, Y. Deng, F.T. Chan, P. Xu, S. Mahadevan and Y. Hu, IFSJSP: a novel methodology for the job-shop scheduling problem based on intuitionistic fuzzy sets. *Int. J. Prod. Res.* **51** (2013) 5100–5119.