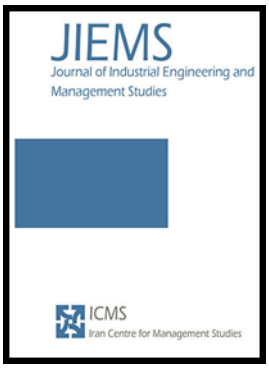
**JIEMS**

**Journal of Industrial Engineering and Management Studies**

**Vol. 7, No. 2, 2020, pp. 223-239**

DOI: 10.22116/JIEMS.2020.206474.1310

www.jiems.icms.ac.ir




# Ant colony optimization, genetic algorithm and hybrid metaheuristics: A new solution for parallel machines scheduling with sequence-dependent set-up times

Mahdi Nakhaeinejad[1,*]

**Abstract**

The parallel machine scheduling problem (PMSP) is one of the most difficult classes of problem. Due to the complexity of the problem, obtaining optimal solution for the problems with large size is very time consuming and sometimes, computationally infeasible. So, heuristic algorithms that provide near-optimal solutions are more practical and useful. The present study aims to propose a hybrid metaheuristic approach for solving the problem of unrelated parallel machine scheduling, in which, the machine and the job sequence dependent setup times are considered. A Mixed-Integer Programming (MIP) model is formulated for the unrelated PMSP with sequence dependent setup times. The solution approach is robust, fast, and simply structured. The hybridization of Genetic Algorithm (GA) with Ant Colony Optimization (ACO) algorithm is the key innovative aspect of the approach. This hybridization is made in order to accelerate the search process to near-optimal solution. After computational and statistical analysis, the two proposed algorithms are used to compare with the proposed hybrid algorithm to highlight its advantages in terms of generality and quality for short and large instances. The results show that the proposed hybrid algorithm has a very good performance as regards the instance size and provides the acceptable results.

**Keywords:** scheduling; parallel machines; ant colony optimization; genetic algorithm; machine scheduling.

*Received: October 2019-25*

*Revised: September 2020-21*

*Accepted: September 2020-26*

## 1. Introduction

The PMSP is one of the most difficult classes of problem, and is widely applied in various commercial and industrial fields (Ying and Cheng, 2010). Having multiple machines with different capabilities in parallel is commonly seen in practice. In addition, setup time between the jobs is a very common phenomenon in many industries. The unrelated PMSP addressed in this paper is the scheduling of $n$ jobs that are available at time zero on $m$ unrelated machines ($R_m$) to minimize the makespan, $C_{max}$, without preemption. Each job is made up of one single task that requires a given processing time. Machines are considered unrelated and jobs' processing times depend on the machine to which they are assigned to. There is no relationship

* Corresponding author; m.nakhaeinejad@yazd.ac.ir
[1] Department of Industrial Engineering, Yazd University, Yazd, Iran.

between machine speeds. Machine-dependent sequence-dependent setup times $S_{ijk}$ are also considered. In fact, the setup times, considered in this paper, are both sequence and machine dependent in a sense that each machine has its own matrix of setup times.

Due to the complexity of the problem, finding optimal solutions for the problems with large size is very time consuming and sometimes, computationally infeasible. Developing heuristic algorithms to reach near-optimal solutions are more practical and useful. This research deals with the problem, denoted as $R_m|S_{ijk}|C_{max}$. The basic identical PMSP ($P_m||C_{max}$) is NP-hard, even when $m = 2$. Since, $R_m|S_{ijk}|C_{max}$ is a generalization of the former problem, then it is also NP-hard (Rabadi et al., 2006).

Since the majority of PMSPs are known to be NP-hard, these problems present a great challenge to industrial practitioners and academic researchers (Ying and Cheng, 2010). In Radhakrishnan and Ventura's work (2000), the SA algorithm and the capacity of its application has been revealed for the problem of parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. A Meta-RaPS (Meta-heuristic for Randomized Priority Search) algorithm has been introduced by Rabadi et al. (2006) for the problem of non-preemptive unrelated parallel machine scheduling problem with the objective of minimizing the make-span. Logendran et al. (2007) presented a methodology for minimizing the weighted tardiness of jobs in unrelated parallel machining scheduling with sequence-dependent setups. They assumed the dynamic release of jobs and dynamic availability of machines. They developed six different search algorithms based on tabu search to identify the best schedule that gives the minimum weighted tardiness. Behnamian et al. (2009) introduced an effective hybrid algorithm based on hybridizing the population-based evolutionary searching ability of ACO that introduced with the local improvement ability of some Variable Neighborhood Search (VNS) and Simulated Annealing (SA) to balance exploration and exploitation. An iterated greedy heuristic for dynamic parallel machine scheduling problem with sequence-dependent setup times has also been presented by Ying and Cheng (2010). Fanjul-Peyro and Ruiz (2010) proposed a set of simple iterated greedy local search based on metaheuristics that produce very good quality solutions in a very short amount of time. Fanjul-Peyro and Ruiz (2011) studied the unrelated parallel machines problem where $n$ independent jobs must be assigned to one of $m$ parallel machines and the processing time of each job differs from machine to machine. They deal with the minimization objective of the maximum completion time of the jobs. They proposed a set of metaheuristics based on a size-reduction of the original assignment problem that produce very good quality solutions in a short amount of time. Vallada and Ruiz (2011) presented GA for the unrelated parallel machine scheduling problem with the objective of minimizing the make-span, in which, machine and job sequence dependent setup times are considered. Sung et al. (2013) proposed a method for Mobile Harbor scheduling. They formulated the problem with Mixed-Integer Programming, based on an m-parallel machine problem and developed a heuristic approach using a GA to obtain a near optimal solution with reduced computation time. Arnaout et al. (2014) introduced an enhanced ACO, which referred to it as ACO II for minimizing the schedule's makespan on unrelated parallel machines with sequence-dependent setup times and compared its performance to other existing and new algorithms including ACO I, MetaRaPS, and SA. They showed the superiority of the enhanced ACO II by extensive and expanded experiments. Tsai and Wang, (2015) proposed mixed-integer programming formulations for the unrelated parallel-machine scheduling problem considering machine idle times with the objective of minimizing the total earliness and tardiness. Sels et al. (2015) developed three heuristic approaches, i.e., a genetic algorithm, a tabu search algorithm and a hybridization of these heuristics with a truncated branch-and-bound procedure. They compared the performances of these heuristics on a standard dataset available in the literature. In Gedik et al. (2016), the authors proposed constraint programming model and logic- based Benders algorithm in order to make the best decision for scheduling

non- identical jobs with availability intervals and sequence dependent setup times on unrelated parallel machines in a fixed planning horizon. A robust schedule by min- max regret criterion has been identified in Hu et al. (2016) for parallel machine scheduling problem in plastic production that the processing time and arrival time are uncertain and each job must be processed together with a mold while jobs which belong to one family can share the same mold. They proposed an exact and a modified artificial bee colony algorithm to solve large- scale problems. Santos et al. (2016) investigated the performance of four different stochastic local search methods, simulated annealing, iterated local search, late acceptance hill-climbing, and step counting hill-climbing, designed for solving the unrelated parallel machine scheduling problem with sequence-dependent setup times. All their proposed methods performed significantly better than two state-of-the-art hybrid heuristics, especially for larger instances. Strohhecker et al. (2016) analyzed different heuristic strategies for the scheduling of production orders on two unrelated parallel machines with sequence-dependent set-up times. They examined the combined effects of loading and sequencing heuristics on several performance indicators such as total busy time (total set-up and production time), customer service level and packaging cycle time. Sayyah et al. (2016) proposed an effective ant colony optimization (EACO) which is different with common ant colony optimization and includes insert, swap and 2-Opt moves for solving vehicle routing problem with simultaneous pickup and delivery. They showed that EACO not only presented a very satisfying scalability, but also is competitive with other meta-heuristic algorithms. In Fanjul-Peyro et al. (2017), the authors analyzed a parallel machine scheduling problem in which the processing of jobs on the machines requires a number of units of a scarce resource when the number depends both on the job and on the machine. They modeled the problem by means of two integer linear programming problems and proposed three matheuristic strategies for each of these two models. Ezugwu et al. (2018) developed an improved symbiotic organism search (SOS) algorithm to solve the parallel machine scheduling problem. They evaluated the performances of their proposed methods by comparing the solutions with other existing techniques from the literature. Lee et al. (2018) considered a scheduling problem on parallel machines for the objective of minimizing total tardiness. Each of the machines needs preventive maintenance tasks that should be started within a given cumulative working time limit after the previous maintenance. They developed dominance properties and lower bounds for the problem, and presented a branch and bound algorithm. Fanjul-Peyró et al. (2019) proposed new mixed integer linear programs and a mathematical programming-based algorithm for unrelated parallel machine scheduling problem with machine and job sequence dependent setup times with makespan minimization criterion. They showed that the proposed methods significantly improve on existing methods and are able to obtain solutions for extremely large instances. Ezugwu (2019) studied on the non-preemptive unrelated parallel machine scheduling problem with the objective of minimizing makespan. He proposed three different approaches to solve the problem including: An Enhanced Symbiotic Organisms Search (ESOS) algorithm, a Hybrid Symbiotic Organisms Search with Simulated Annealing (HSOSSA) algorithm, and an Enhanced Simulated Annealing (ESA) algorithm. He incorporated a local search procedure into each of the three algorithms as an improvement strategy to enhance their solution qualities. The computational experiments showed ESOS and HSOSSA performed better than other methods on large problem instances with 12 machines and 120 jobs. Yepes-Borrero et al. (2019) studied the unrelated parallel machine scheduling problem with setup times and additional limited resources in the setups (UPMSR-S), with makespan minimization criterion. They proposed three metaheuristics following two approaches: a first approach that ignores the information about additional resources in the constructive phase, and a second approach that takes into account this information about the resources. They carried out computational experiments over a benchmark of small and large instances. After the computational analysis they concluded that

the second approach shows an excellent performance, overcoming the first approach. Hosseini (2019) studied a kind of two-stage assembly flowshop scheduling problem. He considered two objective functions: (1) minimizing the completion time of all products (makespan) as a classic objective, and (2) minimizing the cost of energy consumption as a new objective. He modeled the problem as a mixed integer linear programming (MILP) and applied GAMS software to solve small problems. To solve the bi-objective model, he used Epsilon Constraint algorithm on some test problems obtained standard references. Kim et al. (2020) presented a novel mathematical model with meta-heuristic approaches to solve the problem of identical parallel machine scheduling problem with minimizing total tardiness. They proposed two encoding schemes for meta-heuristic solutions and three decoding methods for obtaining a schedule from the meta-heuristic solutions. They performed computational experiments to find the best combination from those encoding schemes and decoding methods. Arnaout (2020) introduced a Worm Optimization (WO) algorithm and applied to the unrelated parallel machine scheduling problem with setup times, with an objective of minimizing the makespan. They evaluated WO's performance by comparing its solutions to six other known metaheuristics. Bastos and Resendo (2020) investigated the problem of manufacturing processes by bar-turning machine as a problem of scheduling jobs to non-related parallel machines with sequence dependent setup times applying batch splitting. They proposed a mixed-integer linear programming model (MILP) and they presented a two-step approach to solve real-size instances. They presented a heuristic algorithm to adjust the solutions. Babaee Tirkolaee et al., (2020) proposed a novel bi-objective Mixed-Integer Linear Programming (MILP) model for flowshop scheduling with an outsourcing option and Just-in-Time (JIT) delivery in order to simultaneously minimize total cost of the production system and total energy consumption. They considered that each job to be either scheduled in-house or to be outsourced to one of the possible subcontractors. They proposed a hybrid technique based on an interactive fuzzy solution technique and a Self-Adaptive Artificial Fish Swarm Algorithm (SAAFSA). Their proposed model is treated as a single objective MILP using a Multi-Objective Fuzzy Mathematical Programming (MOFMP) technique based on the ε-constraint, and SAAFSA is then applied to provide Pareto optimal solutions.

Scheduling problems with make-span measures of performance in the presence of sequence-dependent set-up times are mathematically complex to solve, and optimal solutions to such problems are highly intractable. Therefore, optimal solutions for moderately sized problems of this type cannot be obtained in a reasonable amount of time and the need exists for efficient heuristics to achieve near optimal solutions. This paper considers the unrelated PMSP with machine and job sequence dependent setup times by proposing algorithms that the results show excellent performance for a large benchmark of instances. The aim of the present study is to propose and evaluate the Genetic Algorithm (GA), Ant Colony Optimization (ACO), and also a hybrid algorithm based on GA and ACO. To the best of our knowledge, there are no research article that consider the hybridization of GA with ACO algorithms that is the key innovative aspect of this paper.

The remaining of this paper is organized as follows. The problem formulation is explained briefly in the next section. In Sect. 3, the proposed GA and ACO algorithm are presented as solution methodology for the problem. In Sect. 4, the hybrid metaheuristic approach combines ACO algorithm and GA is proposed. The computational results are studied in section 5 to illustrate the process of the proposed methods. Finally, the last section is devoted to conclusion.

## 2. Problem formulation

In this section, a Mixed- Integer Programming (MIP) mathematical model is formulated for the unrelated PMSP with sequence dependent setup times. The problem is to schedule $N$ jobs with sequence-dependent set-up times and varying processing times on $M$ identical parallel

machines. All machines are continuously available and each machine can handle one job at a time without preemption. Also, each arriving job can be processed on any of the machines, one at a given time. The notations for the scheduling problem are given as follows:

**Sets**

*J* set of jobs
*K* set of machines

**Parameters**

$p_{jk}$ processing time of job $j \in J$ at machine $k \in K$.
$st_{ijk}$ set-up time for job $j \in J$ immediately following job $i \in J$ ($i \neq j$) in sequence at machine $k \in K$.
*M* a large positive number.

**Decision variables**

$C_{max}$ Maximum completion time
$C_i$ Completion time of jobs at machine $i$

$$x_{ijk} = \begin{cases} 1, & if\ job\ i\ precedes\ job\ j\ on\ machine\ k \\ 0, & otherwise \end{cases}$$

The variable $C_{max}$ and variables $C_i$ are positive variables, and the variables $x_{ijk}$ are 0- 1 integer variables.
Based on the definitions and notation, the problem can be formulated as follows (Note that this model is an adapted version of the one, proposed by Guinet (1993)):

$$min\ C_{max} \quad (1)$$

*Subject to*

$$C_{max} \geq C_i, \qquad i = 1, 2, \dots, n. \quad (2)$$

$$\sum_{i=0, i\neq j}^{n} \sum_{k=1}^{m} x_{ijk} = 1, \qquad j = 1, 2, \dots, n. \quad (3)$$

$$\sum_{j=0}^{n} x_{0jk} = 1, \qquad k = 1, 2, \dots, m. \quad (4)$$

$$\sum_{i=0, i\neq h}^{n} x_{ihk} - \sum_{j=0, j\neq h}^{n} x_{hjk} = 0, \qquad h = 1, 2, \dots, n; k = 1,2, \dots, m. \quad (5)$$

$$C_i + \sum_{k=1}^{m} x_{ijk} \left(st_{ijk} + p_{jk}\right) + M\left(\sum_{k=1}^{m} x_{ijk} - 1\right) \leq C_j, \qquad i \neq j; i = 0, 1, 2, \dots, n;\ j = 1, 2, \dots, n. \quad (6)$$

$$C_i + x_{ijk} * \left(st_{ijk} + p_{jk}\right) \geq C_j + M * \left(x_{ijk} - 1\right) \qquad i \neq j; i = 0, 1, \dots, n;\ j = 1, 2, \dots, n; k = 1,2, \dots, m. \quad (7)$$

$$C_0 = 0. \quad (8)$$

$$C_j \geq 0, \qquad j = 1, 2, \dots, n. \quad (9)$$

$$x_{ijk} \in \{0,1\}, \qquad i \neq j; i, j = 0, 1, 2, \dots., n; k = 1, 2, \dots, m. \quad (10)$$

In order to assign a real job to the first and last position in the sequence on each machine, a dummy job 0 are introduced. So, $x_{0jk}$ =1, when job *j* is the first job to be processed on machine *k*, and $x_{0jk}$ =0, otherwise. If $x_{i0k}$ = 1, job *i* is the last job to be processed on machine *k*, and is 0 otherwise. As the same, $st_{0jk}$ is the setup time for job *j* at the first sequence position, and $st_{i0k}$ is the setup time for job *i* at the last sequence position at machine *k*.
Based on the MIP mathematical formulated model, objective function (1) illustrates minimizing the maximum completion time or make-span. Constraint (2) states that completion time of job *i* is lower than or equal to maximum completion time of jobs. Constraint (3) ensures that each job is processed only once and by one machine. Constraint (4) ensures that at most, one job could be scheduled first at each machine. Constraint (5) makes sure that each job is

neither be preceded nor succeeded by more than one job. Constraints (6) and (7) calculate the completion times and make sure that no job precedes and succeeds the same job. Constraint (8) sets the completion time of the dummy job 0 to be zero. Constraint (9) ensures that completion time is not negative. Constraint (10) specifies that the decision variables $x_{ijk}$ are binary variables.

## 3. Proposed genetic and ant colony optimization algorithm

The Genetic algorithm starts with a set of randomly selected chromosomes as the initial population that encodes a set of possible solutions. In GA, variables of a problem are represented as genes in a chromosome, and the chromosomes are evaluated according to their fitness values using some measures of profit or desired optimizable utility. Two operators of crossover and mutation are applied for composition of genes to create new chromosomes, called offspring. The selection operator is an artificial version of natural selection, to create populations from generation to generation. The chromosomes with better fitness values have higher probabilities of being selected in the next generation. After several generations, GA can converge to the best solution (Lee et al., 2008). The GA structure is designed as described in Algorithm 1.

**Algorithm 1**: Genetic Algorithm
1: **Inputs**: Initial population of individuals
2: Evaluate the individuals' fitness
3: **While** some stop criteria not satisfied **do**
4: Select two individuals (parents) in the population
5: Generate new offspring by crossover operation between two parents
6: Generate new offspring by mutating some older individuals
7: Evaluate the fitness of new offspring
8: Update the population based on population size for the next generation
9: **End while**
10: **End**

As algorithm 1 shows, the GA input is a set of solutions called population of individuals. Parents are selected based on individuals' evaluation and a crossover mechanism is used to create a new generation of individuals (offspring). Moreover, a mutation scheme is also applied in order to create diverse populations.

The GA of this paper is formed by a population of *P* size individuals; each of them is a solution for the problem. The solution representation in this algorithm is an array that represents the processing order of jobs, assigned to each machine. Each array contains the following number of variables:

$$nvar = n + m - 1 \tag{11}$$

Where *nvar* is the number of variables, *n* is the number of jobs and *m* is the number of machines. For example, when there are 6 jobs and 2 machines, one individual could be as shown in Figure 1 (a) that *nvar*= 7. Grey blocks represent the processing order of jobs assigned to each machine (setup times between jobs are not shown for clarity).

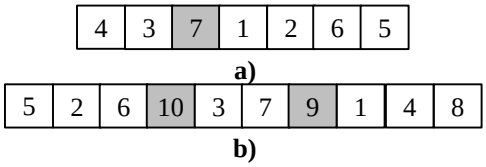



**Figure 1. The solution representation: a) 6 jobs and 2 machines; and b) 8 jobs and 3 machines**

Figure 1 (a) represents that jobs sequence on machine 1 is {4, 3} and on machine 2 is {1, 2, 6, 5}. When there are 8 jobs and 3 machines, one individual could be as shown in Figure 1 (b), having *nvar*= 10. Figure 1 (b) represents jobs sequences {3, 7} on machine 1, {5, 2, 6} on machine 2, and {1, 4, 8} on machine 3. This representation is complete because all feasible solutions of the MIP model could be illustrated.

The initial population for the proposed GA is generated randomly. Two individuals with lowest makespan values among the randomly generated individuals are selected for crossover operator. The *Cmax* value can be directly used as a fitness value. The crossover operator is applied according to a probability (*Pc*). In general, the goal of the crossover operator is to generate two good offsprings from the two selected progenitors.

After testing different crossover operators, the best performance was obtained by one-point order crossover. In this operator, one-point *p* randomly selected from parent 1 and jobs from the first to the *p* position are copied to the first offspring. Jobs from the point *p* + 1 position to the end are copied to the second offspring.

Figure 2 shows an example of 10 jobs and 2 machines. Two parents and a point *p* have been selected. Specifically, point *p* is set to 5. Therefore, the first offspring is formed with the jobs of parent 1 from positions 1 to 5 and jobs from positions 6 to 11 of parent 2. The second offspring contained jobs of parent 2 from positions 1 to 5 and jobs from positions 6 to the last of parent 1 (Figure 2). Then, the jobs in generated offspring return in the same order and jobs which are not already in the offspring are inserted randomly in the last positions.

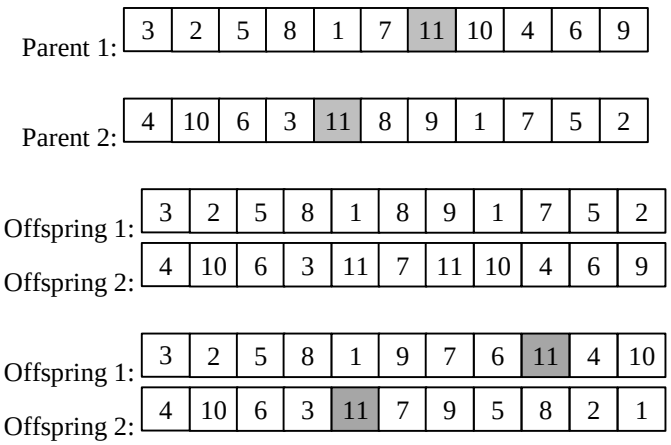



**Figure 2. Example of the crossover operator**

Mutation is another operator could be applied according to a probability ($P_m$). After short experiments testing different mutation operators, the best performance has been obtained as follows: one individual is randomly selected and a job also is randomly picked out and reinserted to a different position, randomly chosen in the same individual. According to Figure 3, points 3 and 9 has been selected and related jobs (jobs 6 and 8) were interchanged.

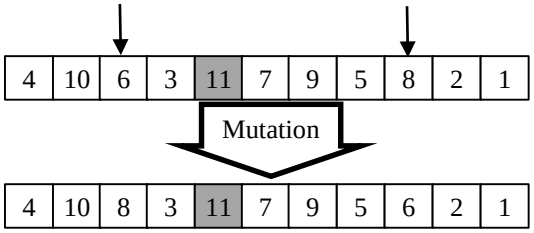



**Figure 3. Example of the mutation operator**

### 3.2. The proposed ACO algorithm

ACO algorithms are a class of algorithms inspired by the observation of real ants. Ants are capable of exploring and exploiting pheromone information, which have been left on the ground when they traversed. They then can choose routes based on the amount of pheromone. The pheromone trail information is changed during problem solution to reflect the experience acquired by ants during problem solving. Ants deposit an amount of pheromone proportional based on the quality of the solutions they produced. This choice helps to direct search towards good solutions (Lee et al., 2008; Babaee Tirkolaee et al., 2019). The larger amount of pheromone is left on a route, the greater is the probability of selecting the route by artificial ants.

In ACO, the artificial ants find solutions by starting from a start node and moving to feasible neighbor nodes. During the process, information collected by artificial ants is stored in the so-called pheromone trails. In the process, artificial ants can release pheromone while building the solution. An ant-decision rule, made up of the pheromone and heuristic information, governs artificial ants to search towards neighbor nodes stochastically. Pheromone evaporation is a process of decreasing the intensities of pheromone trails over time. This process is used to avoid locally convergence and to explore more search space. The algorithmic framework of ACO is described in algorithm 2 (Lee et al., 2008).

**Algorithm 2**: Ant colony optimization
1: **inputs**: population, a set of individuals
2: **while** termination conditions not satisfied **do**
3: Position each ant in a starting job
4: **Repeat**
5: **For** each ant **do**
6: Choose next job by applying the state transition rule
7: Apply step by step pheromone update
8: **End for**
9: **Until** every ant has built a solution
10: Update best solution
11. Apply offline pheromone update
12. **End While**
13. **End**

The solution representation in this algorithm is an array of the variables, exactly the same as solution representation for GA that represents the processing order of the jobs assigned to each machine (Figure 1). The proposed ACO algorithm initializes the pheromone matrix $\tau(i,j)\ \forall\ i,j\epsilon\ nvar$ according to (12):

$$\tau_{ij} = \begin{cases} \tau_0 * ones(nvar, nvar) & i \neq j \\ 0 \qquad otherwise & (i = j) \end{cases} \tag{12}$$

Where $\tau_0$ is a fixed number, $ones(nvar, nvar)$ is a unique matrix and *nvar* is the number of variables according to Eq. (11). Once an individual is produced (or a sequence is completed by ant), the pheromone $\tau_{ij}$, associated with the variable *j* immediately following variable *i* in the sequence, is updated according to (13):

$$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{13}$$

Where $\rho\epsilon(0,1]$ is the evaporation rate, $m$ is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid by ant *k* on rout (sequence) between variable *j* immediately following variable *i* according to (14):

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & if\ ant\ k\ used\ rout\ variable\ j\ follows\ variable\ i\ in\ the\ sequence \\ 0 & otherwise \end{cases} \quad (14)$$

Where $Q$ is a constant and $L_k$ is the fitness function related to the rout constructed by ant $k$. In this paper $L_k$ is the objective function based on Eq. (1) related to the sequence constructed by ant $k$. In the construction of a solution, ants select the following variable to be processed through a stochastic mechanism. The probability of going to variable $j$ when ant $k$ is in variable $i$ and has so far constructed the partial solution $S^P$, is given by:

$$p_{ij}^{k} = \begin{cases} \tau_{ij}^{\alpha}.\eta_{ij}^{\beta} & if\ (i,j)\epsilon\ S^P, \\ 0 & otherwise \end{cases} \quad (15)$$

Where $\alpha$ and $\beta$ are the parameters control the relative importance of the pheromone versus the heuristic information $\eta_{ij}$, which is given by (16) as follows:

$$\eta_{ij} = \begin{cases} 1/\sum_{k=1}^{m}\left(st_{ijk} + p_{jk}\right) & i \neq j \\ 0 \quad otherwise \end{cases} \quad (16)$$

Where, as described in the problem formulation section, $st_{ijk}$ is the set-up time for job $j$ immediately following job $i$ in sequence at machine $k$, $p_{jk}$ is the processing time of job $j$ at machine $k$, and $m$ is the number of machines.

## 4. Proposed hybrid metaheuristic approach

The hybrid algorithm proposed in this section combines of ACO algorithm and GA. The search starts from some initial solution based on ACO and iterative moves are performed among neighboring solutions by crossover and mutation operators. Every individual (solution), generated by ACO is at the same time a chromosome for GA. The basic proposed hybrid algorithm structure is designed as shown in Algorithm 3.

**Algorithm 3**: hybrid metaheuristic approach

1: **while** time < *Tmax* **do**
2: Initialize: For every variable (*i, j*) set an initial $\tau_0 = c$ in Eq. (12) for trail density and $\Delta\tau_{ij} = 0$.
3: **while** *iter* < *maxiter*
4: **For** *k*=1 to *nant* **do** {*nant* is the number of ants}
5: Set *v*=0 {*v* is variable counter}
6: Place ant *k* on a variable randomly. Put the variable in *visited*$_k$.
7: Repeat until $v \leq nvar$
8: Set *v*=*v* + 1
9: Choose the next variable from the unvisited variables to be visited according to the probability $p_{ij}^{k}$ given by Eqs. (15) and (16).
10: Move the ant *k* to the selected variable.
11: Put the selected variable in *visited*$_k$.
12: Compute the objective function for ant *k*.
13: Update the best scheduling found.
14: For every variable (*i, j*) **do**
15: Update the pheromone trail density $\tau_{ij}$ according to Eqs. (13) and (14).
16: **End For**
17: save the best scheduling found in the *BPOP* {*BPOP* is the best population (scheduling) found}
18: Empty *visited*$_k$
19: **End while**

20: **For** *k*=1 to *ncross* {*ncroos* is the number of crossovers} **do**
21: Select parent solutions *P1* and *P2* randomly from *BPOP*
22: Create offspring *C* from *P1* and *P2* based on crossover operator
23: Insert *C* into *Crosspop* {*Crosspop* is the population generated by crossover} population
24: **End For**
25: **For** *k*=1 to *nmut* {*nmut* is the number of mutations} **do**
26: Randomly select a population *P* from *BPOP*
27: Randomly select a point in the selected population
28: Perform offspring *M* based on mutation operator
29: Insert *M* into *Mutpop* {*Mutpop* is the population generated by mutation} population
30: **End For**
31: Update BPOP by selecting *sizeBPOP* {*sizeBPOP* is population size of *BPOP*} number of best individuals from *BPOP*, *Crosspop* and *Mutpop*
32: **End while**
33: Return best feasible solution

## 5. Computational results

To check the accuracy of the proposed algorithms and MIP formulated model, a small-scale example with six jobs and two unrelated parallel machines was taken into account. Table 1 shows job processing times and Tables 2 and 3 show job set up times at machines 1 and 2, respectively. The CPLEX software and also three proposed algorithms were used for solving the MIP model. The output of CPLEX software and proposed algorithms show the same results, as shown in Figure 4.

**Table 1. Job processing times for six jobs and two machines**

| **Job** | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $\mathbf{p_{j1}}$ | 73 | 60 | 19 | 71 | 86 | 45 |
| $\mathbf{P_{j2}}$ | 29 | 25 | 51 | 76 | 27 | 14 |

**Table 2. Job set up times at machine 1**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 0 | 1 | 3 | 3 | 6 | 3 | 1 |
| 1 | 0 | 1 | 3 | 5 | 7 | 3 |
| 2 | 3 | 0 | 6 | 3 | 1 | 2 |
| 3 | 6 | 2 | 0 | 3 | 1 | 9 |
| 4 | 5 | 7 | 4 | 0 | 4 | 5 |
| 5 | 7 | 5 | 3 | 1 | 0 | 7 |
| 6 | 6 | 9 | 3 | 6 | 1 | 0 |

**Table 3. Job set up times at machine 2**

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 2 | 4 | 2 | 5 |
| 1 | 0 | 1 | 2 | 2 | 2 | 4 |
| 2 | 4 | 0 | 4 | 2 | 5 | 3 |
| 3 | 4 | 1 | 0 | 2 | 1 | 6 |
| 4 | 6 | 2 | 9 | 0 | 1 | 5 |
| 5 | 6 | 9 | 2 | 7 | 0 | 2 |
| 6 | 8 | 3 | 5 | 7 | 1 | 0 |

The Gantt chart of solution in Figure 1(a) which has been shown in Figure 4 is the optimal jobs schedule for the above example, when the model solved by the CPLEX software and also by

the proposed algorithms. As figure 4 shows jobs 4 and 3 are processed in sequence on machine 1, and are finished at times 77 and 100, respectively. Jobs 1, 2, 6 and 5 are completed on machine 2 at times 31, 57, 74 and 102, respectively. So, the make-span of the proposed example is 102.

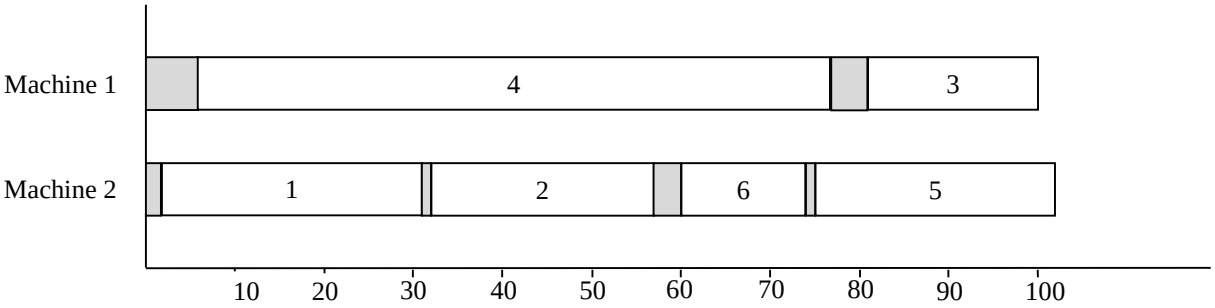



**Figure 4. The Gantt chart schedule of six jobs on two machines**

The behavior of the proposed algorithms to obtain solution plotted in Figures 5 and 6 for the above example of six jobs and two machines. This Figures show the trend of GA and ACO respectively.

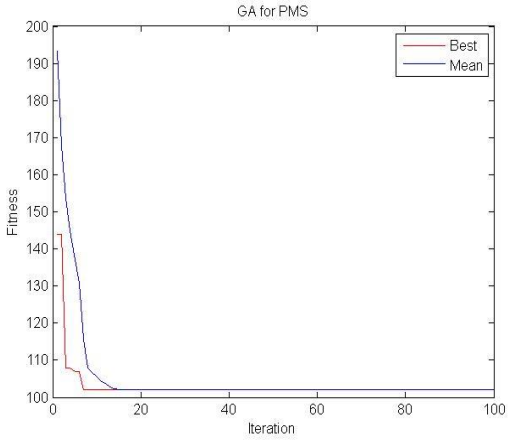



**Figure 5. The trend of proposed GA for six jobs on two machines**

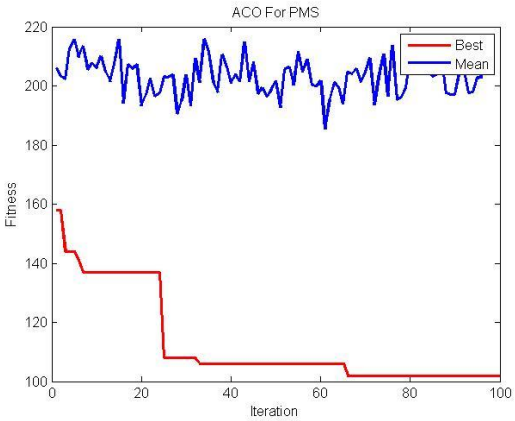



**Figure 6. The trend of the proposed ACO for the example with six jobs and two machines**

According to the fact that the problem under study is NP-hard, optimal solutions cannot be obtained in polynomial time. Hence, only small-sized problems like the example of Figure 4, can be solved to optimality using CPLEX Linear Optimizer, which has a built-in mixed integer

solver. So, for small-sized problems, the results from the proposed algorithms were compared with the best integer solutions, yielded by the CPLEX software. The measure used for performance evaluation of the problems is the Relative Percentage Deviation (RPD) following expression (17).

$$RPD = \frac{Method_{sol} - Best_{sol}}{Best_{sol}} \quad (17)$$

Where $Best_{sol}$ is the best solution obtained after all the experiments carried out through the paper, and $Method_{sol}$ is the solution obtained with a proposed algorithm.
Based on the fact that calibration of the algorithms is a very important step that could significantly improve the results; a Design of Experiments carried out on benchmark of instances for calibration. The parameters considered in calibration experiments, are summarized in Tables 4 and 5 for GA and ACO, respectively. The best combination of values is in bold face.

**Table 4. Test values for the parameters in GA calibration experiments (best combination in bold face)**

| Parameter | Calibration |
|---|---|
| Population size ($P_{size}$) | 20; 40; **60** |
| Probability of crossover ($P_c$) | 0; 0.2; **0.5** |
| Probability of mutation ($P_m$) | 0; **0.2**; 0.5 |

**Table 5. Test values the parameters in ACO calibration experiments (best combination in bold face)**

| Parameter | Calibration |
|---|---|
| $\alpha$ | 0.2; 0.5; **1**; 2 |
| $\beta$ | 0.2; 0.5; 1; **2** |
| $\rho$ | 0.5; 0.6; 0.7; **0.8** |
| $Q$ | 0.5; **0.6**; 0.7; 0.8 |

## 5.1. Comparative evaluation for instances

After the calibration experiments, the comparative evaluation was carried out under the set of small and large instances. All metaheuristic methods were run five independent times over each instance to get an accurate picture of the results. The processing times were uniformly distributed between 1 and 99 and for the setup times, 4 subsets were generated, in which, the setup times were uniformly distributed between 1–9, 1–49, 1–99 and 1–124, respectively. In order to test the sensitivity of the algorithms to the size of the instance, two sets of instances are generated. The first one, small instances, which the number of jobs set to $n = \{6, 8, 10, 12\}$, and the number of machines set to $m = \{2, 3, 4, 5\}$ and the second one, large instances, where the following values are tested to $n = \{50, 100, 150, 200, 250\}$ and $m = \{10, 15, 20, 25, 30\}$.
The algorithms were coded in *Matlab* software and run on computer with a Dual- Core CPU running at 2.6 Ghz with 4 GB of main memory. The RPD was computed for each instance, according to the expression (17). The maximum CPU time for the MIP model was set to 1h, that is, if after 1h no optimal solution obtained, the best current integer solution is returned. Table 6 shows the results of small instances for proposed methods. Obviously, if CPLEX obtained the optimal solution, the RPD value computed over the optimal make-span value.

**Table 6. Average RPD for the proposed algorithms (small instances)**

| Instance | GA | ACO | Proposed hybrid metaheuristic approach (HMA) |
|---|---|---|---|
| 6*2 | 0.00 | 0.00 | 0.00 |
| 6*3 | 0.07 | 0.05 | 0.02 |
| 6*4 | 0.20 | 0.22 | 0.17 |
| 6*5 | 0.13 | 0.16 | 0.10 |
| 8*2 | 1.21 | 1.09 | 0.07 |
| 8*3 | 1.67 | 0.37 | 0.023 |
| 8*4 | 2.47 | 2.70 | 0.19 |
| 8*5 | 4.33 | 2.54 | 1.02 |
| 10*2 | 4.05 | 3.98 | 1.71 |
| 10*3 | 3.67 | 3.75 | 1.54 |
| 10*4 | 7.17 | 6.81 | 1.15 |
| 10*5 | 3.85 | 3.33 | 0.98 |
| 12*2 | 9.29 | 9.53 | 1.12 |
| 12*3 | 9.84 | 8.77 | 1.30 |
| 12*4 | 19.63 | 15.42 | 1.49 |
| 12*5 | 23.02 | 20.10 | 1.36 |
| Average | 5.66 | 4.93 | 0.77 |

Table 6 displays the best results of proposed HMA. In order to validate the results, an analysis of variance (ANOVA) was applied in order to check the significance of the statistical analyses. The mean plot shown in Figure 7, with HSD Tukey intervals (a= 0.05) for small instances. This figure shows that the proposed hybrid methodology is better than other proposed methodologies.

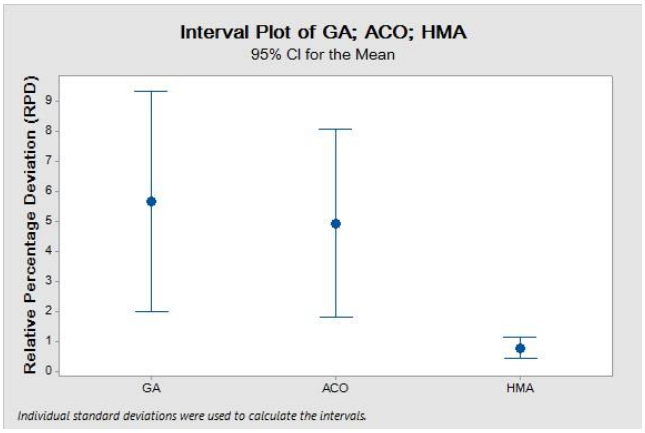



**Figure 7. Mean plot and Tukey HSD intervals at the 95% confidence level for the evaluated methods (small instances)**

Also, for analyzing the algorithms when the size of the instances increased, the experiments carried out using the large set of instances. In Table 7, the results of proposed algorithms for large instances has been shown, where 40 instances of each *n* * *m* group have been averaged.

**Table 7. Average RPD for the proposed algorithms (large instances)**

| Instance | GA | ACO | Proposed HMA |
|---|---|---|---|
| 50*10 | 27.28 | 23.042 | 6.93 |
| 50*15 | 29.06 | 28.31 | 11.77 |
| 50*20 | 35.16 | 37.95 | 13.24 |
| 50*25 | 47.21 | 43.92 | 13.71 |
| 50*30 | 53.81 | 50.02 | 15.27 |
| 100*10 | 34.51 | 31.60 | 10.42 |
| 100*15 | 40.61 | 35.13 | 12.19 |
| 100*20 | 44.08 | 41.26 | 14.34 |
| 100*25 | 49.67 | 49.33 | 15.67 |
| 100*30 | 53.26 | 51.27 | 16.91 |
| 150*10 | 37.54 | 36.51 | 18.26 |
| 150*15 | 43.72 | 44.61 | 18.32 |
| 150*20 | 51.29 | 52.64 | 20.57 |
| 150*25 | 51.99 | 52.91 | 22.19 |
| 150*30 | 53.49 | 51.97 | 23.00 |
| 200*10 | 39.15 | 32.84 | 17.91 |
| 200*15 | 45.54 | 43.14 | 19.87 |
| 200*20 | 46.37 | 45.44 | 22.98 |
| 200*25 | 49.62 | 50.19 | 25.57 |
| 200*30 | 52.37 | 53.44 | 27.50 |
| 250*10 | 40.50 | 39.51 | 18.51 |
| 250*15 | 47.87 | 48.05 | 20.61 |
| 250*20 | 52.35 | 53.6 | 21.81 |
| 250*25 | 55.48 | 54.26 | 27.91 |
| 250*30 | 57.62 | 56.81 | 28.07 |
| Average | 45.58 | 44.31 | 18.54 |

The interesting outcome is the great performance in the behavior of proposed hybrid method. The proposed hybrid algorithm shown a very good performance and provided the best results, which proves that this algorithm is robust as regards the instance size.
In order to validate the results, a statistical analysis (ANOVA) are again applied for large instances such as small instances. Figure 8 demonstrates the mean plot of all proposed methods, on average, with HSD Tukey intervals (a = 0.05).

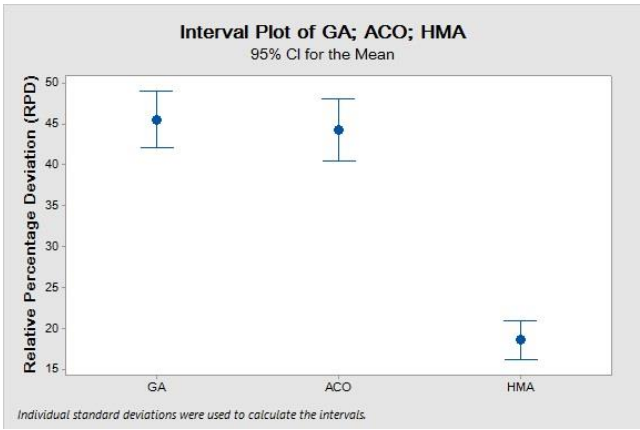



**Figure 8. Means plot and Tukey HSD intervals at the 95% confidence level for the evaluated methods (large instances)**

In this case, the differences between proposed hybrid method and other proposed methods are statistically significant (confidence intervals are not overlapped). Specifically, the proposed hybrid methodology is almost 10% better than other proposed algorithms. Also, the total

amount of CPU time was significantly reduced. So, it can be concluded that the proposed hybrid method is better than other proposed methods, on average.
Based on the results, it is clear that the proposed hybrid algorithm performs better both in small and large instances. In the proposed algorithms, the average RPD for a certain number of jobs increased as the number of machines increased and also, it increased as the number of jobs increased for a certain number of machines in both small and large instances (see Tables 6 and 7). The average RPD of the proposed hybrid algorithm for the all small instances is 0.77 which is much better comparing to 5.66 for GA and 4.93 for ACO, and for large instances it is 18.54 which is also much better comparing to 45.58 for GA and 44.31 for ACO. The results show the decision maker the proposed hybrid method is appropriate for being used in practical situations. The proposed hybrid metaheuristic has made a step towards reducing make-span in industrial applications that have the environment of unrelated parallel machine scheduling, in which, the machine and the job have sequence dependent setup times.

## 6. Conclusions

The parallel machine scheduling problem is one of the most difficult classes of problem, that is widely applied in various industries. This paper studies PMSP with machine-dependent and sequence-dependent setup times to minimize the make-span. Due to the complexity of the problem, finding optimal solutions for the problems with large size is very time consuming and sometimes, computationally infeasible. In this paper a MIP mathematical model formulated for the problem. Three proposed metahuristic approaches, including; GA, ACO, and hybrid approach have been developed to find solutions with better quality. The application of a very fast procedure, based on the hybrid method of GA and ACO, is innovative characteristics of the proposed hybrid method. The proposed hybrid algorithm, combines ACO algorithm and GA, starts from some initial solution based on ACO and iterative moves are performed among neighboring solutions by crossover and mutation operators. In the numerical analysis by calibration, the best combination of parameters for each proposed method was obtained. Then, comparison of the proposed hybrid method against GA and ACO was carried out for the same problem under a comprehensive benchmark of instances. For all the evaluated proposed methods, the stopping criterion was set to a maximum elapsed CPU time (1h). The results show that the proposed hybrid method can obtain the best results for small and large instances. The procedure used in the paper is capable to use in any real cases in the production systems that confront the problem of PMSP with machine-dependent and sequence-dependent setup times to minimize the make-span that is widely applied in various commercial and industrial fields. Further research could be investigated for combining of two algorithms, GA and ACO, in different ways for PMSP and comparing the results with this study. Also, studying the proposed hybrid method for other scheduling problem is other area that could be considered for future research.

## References

Arnaout, J., (2020). "A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times", *Annals of Operations Research*, Vol. 285, pp. 273–293.

Arnaout, J., Musa, R., and Rabadi, G., (2014). "A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines—part II: enhancements and experimentations", *Journal of Intelligent Manufacturing*, Vol. 25, pp. 43–53.

Babaee T., E., Alinaghian, M., Rahmani Hosseinabadi, A. A., Bakhshi Sasi, M., and Kumar Sangaiah A., (2019). "An improved ant colony optimization for the multi-trip Capacitated Arc Routing Problem", *Computers and Electrical Engineering*, Vol. 77, pp. 457-470.

Babaee T., E., Goli, A. and Weber, G., (2020). "Fuzzy Mathematical Programming and Self-Adaptive Artificial Fish Swarm Algorithm for Just-in-Time Energy-Aware Flow Shop Scheduling Problem with Outsourcing Option", *IEEE Transactions on Fuzzy Systems*.

Bastos, C. E. N., and Resendo, L. C., (2020). "Two-step approach for scheduling jobs to non-related parallel machines with sequence dependent setup times applying job splitting", *Computers & Industrial Engineering*, Vol. 145.

Behnamian, J., Zandieh, M., and Fatemi Ghomi, S.M.T., (2009). "Parallel-machine scheduling problems with sequence-dependent setup times using an ACO, SA and VNS hybrid algorithm", *Expert Systems with Applications*, Vol. 36, pp. 9637–9644.

Ezugwu Absalom, E., Adeleke Olawale, J., and Viriri S., (2018). "Symbiotic organisms search algorithm for the unrelated parallel machines scheduling with sequence-dependent setup times", *PLoS ONE*, Vol. 13, No. 7.

Ezugwu Absalom, E., (2019). "Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times", *Knowledge-Based Systems*, Vol. 172, No. 15, pp. 15-32.

Fanjul-Peyro, L., Perea, F., and Ruiz, R., (2017). "Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources", *European Journal of Operational Research*, Vol. 260, No. 2, pp. 482-493.

Fanjul-Peyro, L., and Ruiz, R., (2010). "Iterated greedy local search methods for unrelated parallel machine scheduling", *European Journal of Operational Research*, Vol. 207, No. 1, pp. 55–69.

Fanjul-Peyro, L., and Ruiz, R., (2011). "Size-reduction heuristics for the unrelated parallel machines scheduling problem", *Computers & Operations Research*, Vol. 38, No. 1, pp. 301–309.

Fanjul-Peyro, L., Ruiz, R., and Perea, F., (2019). "Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times", *Computers and Operations Research*, Vol. 101, pp. 173-182.

Gedik, R., Rainwater, C., Nachtmann, H., and Pohl, E., (2016). "Analysis of a Parallel Machine Scheduling Problem with Sequence Dependent Setup Times and Job Availability Intervals", *European Journal of Operational Research*, Vol. 251, No. 2, pp. 640-650.

Guinet, A., (1993). "Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria", *International Journal of Production Research*, Vol. 31, No. 7, pp. 1579-1594.

Hosseini, Seyed Mohammad Hassan, (2019). "A bi-objective model for the assembly flow shop scheduling problem with sequence dependent setup times and considering energy consumption", *Journal of Industrial Engineering and Management Studies*, Vol. 6, No. 2, pp. 44-64

Hu, Hongtao, Ng K. K. H. and Qin, Y., (2016). "Robust Parallel Machine Scheduling Problem with Uncertainties and Sequence-Dependent Setup Time", *Scientific Programming*.

Kim, J., Song, S., and Jeong, B., (2020). "Minimising total tardiness for the identical parallel machine scheduling problem with splitting jobs and sequence-dependent setup times", *International Journal of Production Research*, Vol. 58, No. 6, pp. 1628-1643.

Lee, Ju-Y., Kim, Y., and Lee, T., (2018). "Minimizing Total Tardiness on Parallel Machines Subject to Flexible Maintenance", *International Journal of Industrial Engineering: Theory, Applications and Practice*, Vol. 25, No. 4, pp. 472-489.

Logendran, R., McDonell, B., and Smucker, B., (2007). "Scheduling unrelated parallel machines with sequence-dependent setups", *Computers & Operations Research*, Vol. 34, pp. 3420 – 3438.

Rabadi, G., Moraga, R. J., and Al-salem, A., (2006). "Heuristics for the unrelated parallel machine scheduling problem with setup times", *Journal of Intelligent Manufacturing*, Vol. 17, pp. 85–97.

Radhakrishnan, S., and Ventura, J. A., (2000). "Simulated annealing for parallel machine scheduling with earliness tardiness penalties and sequence-dependent set-up times", *International Journal of Production Research*, Vol. 38, No. 10, pp. 2233- 2252.

Santos Haroldo, G., Toffolo T´ulio, A.M., Silva Cristiano L.T.F. and Berghe G. V., (2016). "Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem", *International journal of Transaction in Operation Research*.

Sayyah M., Larki, H., and Yousefikhoshbakht, M., (2016). "Solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery by an Effective Ant Colony Optimization", *Journal of Industrial Engineering and Management Studies*, Vol. 3, No. 1, pp. 15 – 38.

Sels V., Coelho J., Dias António, M., and Vanhoucke, M., (2015). "Hybrid tabu search and a truncated branch-and-bound for the unrelated parallel machine scheduling problem", *Computers & Operations Research*, Vol. 53, pp. 107–117.

Sung, I., Nam, H., and Lee, T., (2013). "Scheduling Algorithms for Mobile Harbor: an Extended M-Parallel Machine Problem" ", *International Journal of Industrial Engineering: Theory, Applications and Practice*, Vol. 20, No. 1-2, pp. 211-224.

Tsai, C., and Wang, Y., (2015). "Efficient mixed integer programming formulations and dispatching rules for parallel machine scheduling with allowing machine idle times", *International Journal of Industrial and Systems Engineering*, Vol. 21, No. 3, pp. 320-333.

Vallada, E., and Ruiz, R., (2011). "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times", *European Journal of Operational Research*, Vol. 211, No. 3, pp. 612–622.

Ying, K., and Cheng, H., (2010). "Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic", *Expert Systems with Applications*, Vol. 37, No. 4, pp. 2848–2852.

Yepes-Borrero, J. C., Villa, F., Perea, F., and Caballero-Villalobos, J. P., (2019). "GRASP algorithm for the unrelated parallel machine scheduling problem with setup times and additional resources", *Expert Systems with Applications*, Vol. 141.

**This article can be cited:** Nakhaeinejad, M., (2020). "Ant colony optimization, genetic algorithm and hybrid metaheuristics: a new solution for parallel machines scheduling with sequence-dependent set-up times", *Journal of Industrial Engineering and Management Studies*, Vol. 7, No. 2, pp. 223-239.

✓ Copyright: Creative Commons Attribution 4.0 International License.