A New Approach Based on Matching Theory and Simulated Annealing for Parallel Machine Scheduling Problems

Maissa Moussa (maisaamosa910@gmail.com)
Tarbiat Modares University
Adel Azar
Tarbiat Modares University Faculty of Management and Economics
Ali Rajabzadeh Ghatari
Tarbiat Modares University Faculty of Management and Economics

Research Article

Keywords: Meta-heuristic algorithms, Parallel machine scheduling problem , Matching theory, Simulated annealing.

Posted Date: September 27th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-919404/v1

A New Approach Based on Matching Theory and Simulated Annealing for Parallel Machine Scheduling Problems

Maissa Moussa[*1], *Adel Azar* [*2], Ali Rajabzadeh Ghatari [3]

[1*] *Ph.D of Management and Economics, Department of Industrial Management, Tarbiat Modares University, Iran* *maisaamosa910@gmail.com*

[2*] *Professor at the School of Management and Economics, Department of Industrial Management, Tarbiat Modares University, Iran* *azara@modares.ac.ir*

[3] *Professor at the School of Management and Economics, Department of Industrial Management, Tarbiat Modares University, Iran* *alirajabzadeh@modares.ac.ir*

ABSTRACT

As an extension of the classical Parallel Machine Scheduling Problem (PMSP), Unrelated Parallel Machine Scheduling Problem (UPMSP) is a much substantial issue in the modern manufacturing environment. It has been demonstrated to be a NP-hard problem. This research suggests a hybrid algorithm that combines Matching Theory (MT) and Simulated Annealing (SA) for solving an UPMSP with sequence-dependent setup time aimed at minimizing the total completion time. The hybrid algorithm is based on allocation of works to the best machine that can do it, and the determination of the order in which jobs have to be handled on the machines. The hybridization of MT and SA that integrates the features of these two individual parts is the main innovation aspect of the strategy. MT encourages the convergence, while SA promotes the diversity. Therefore, the designed algorithm can balance the intensification and diversification very well. Some tests were conducted using 16 tests for two problems to assess the efficiency of the suggested algorithm. Furthermore, the execution of the suggested algorithm with that of other meta-heuristic methods was contrasted. The outcome revealed that the performance dimensions of the suggested algorithm overrated those of other techniques.

Keywords:
Meta-heuristic algorithms, Parallel machine scheduling problem , Matching theory, Simulated annealing.

1. INTRODUCTION

Production scheduling is one of the main factors in manufacturing and production planning. A highly organized scheduling system can reduce cost, increase productivity, and maximize customer satisfaction. In addition, it has a significant impact on competitive advantages. In parallel machines environment, the same type of machines, set up in parallel, can produce every product [1]. Parallel machines environment has been used in a variety of different fields such as drilling mechanisms in the printed circuit board facility, scheduled textile industry activities, dicing of semiconductor wafers producing computers, and ship docking systems [2].

The parallel machines facilities are made up of similar or non-identical kinds of machines. The non-identical types can be classified into two types: unrelated and uniform. Unrelated parallel machine scheduling problem supposes that the processing times of a job on various machines are unrelated and different [3]. Machine setup time is a significant aspect for production scheduling, rather than whether the parallel machines are unrelated or uniform, and if they are non-identical or identical. Sequence independent setup occurs when the setup time is just based on the work that is being handled. The contrary case, i.e. sequence dependent setup, occurs if the setup time depends on the work being handled; it also relies on the previous work carried out on that machine [4]. The issue is called NP-hard since an UPMSP contains both job sequencing decisions and job allocation to machines [5]. This problem can be represented by using the triplet $\boldsymbol{R/s_{ij}/C_{max}}$, which informs us that the machine environment has *m* unrelated machines (R), with setup times in the machines that are based on the work (sequence dependent setup times $\boldsymbol{s_{ij}}$) and where the makespan

($\boldsymbol{C_{max}}$) is the objective function. Therefore, a significant challenge in this process is to coordinate between machines to execute the scheduling of resource-task assignments' optimality.

Considering the machines and works as two various agent groups, the options can be studied as a two-sided relation. Game theory (GT) is a very important tool to deal with the interaction of some players [6]. It has been widely applied to handle with several situations in a wide number of fields like economics, management, politics, engineering, sociology, philosophy and so on [7]. In complex situations, conventional game theoretical models can hardly be used. Hence, when the problem becomes more complex, a new perspective called "matching theory" (MT) should be used due to the difficulty of utilizing a conventional game theoretically [8]. Therefore, the machine agents can apparent their desires over the job agents, and conversely. Each machine ranks jobs in the other side based on the processing time; on the other hand, each job ranks all machines according to the setup time. Thus, the matching, which take into consideration machine assignment and job sequencing, can be performed simultaneously, not sequentially. Parallel machine scheduling problem is of NP-hard optimization problems, which could be solved using algorithmic approaches. For this purpose, we propose an approach based on MT and SA. Therefore, a representation encoding has been constructed based on two distinct alleles. The first generated allele would be regarded as our proposed algorithm. This avoids the machines assigning their resources to the jobs that are not very suitable for them and vice versa, because the scheduling and process planning were performed simultaneously, and not consecutively. The second allele will be generated by SA.

The goal of this research is to present a novel method, which aims to (i) reduce the complexity of UPMSP, (ii) avoid the machines that assign their resources to the jobs that are not very suitable for them and vice versa, and (iii) allow an efficient workload balance among the machines (i.e. to avoid overloading certain machines with tasks while others remain free). To the best of our knowledge, this is the initial attempt in this direction**.**

The remainder of this article is arranged as follows: The related works concerning PMSP and MT are introduced in Section 2. The preliminaries for UPMSP, SA, and MT are shown in Section 3. Section 4 presents the proposed hybrid algorithm SA - matching to find solution for the UPMSP with dependent setup times. Section 5 discusses the results of the experiments and evaluates the efficiency of the suggested method by comparing it with three other algorithms. In Section 6, the conclusion and future relevant works are discussed.

2. *Methodological background*

A brief literature review of UPMSP and matching theory is provided in this section. Finally, we identify the research gap.

2.1 UPMSP

UPMSP was first proposed by McNaughton (1959) [9]. It has been a field of study for more than six decades. PMSP was shown to be an NP-hard problem. There are several PMSP reviews, including Sin and Cheng [10], Mokotoff [11], Kravchenko and Werner [12], and Edis, Oguz and Ozkarahan [13].

Exacts, heuristics and metaheuristics methods have been proposed along this direction. Few exact methods to solve PMSP have been established [14], [15], [16] and [17]. To achieve optimal solutions to small problem cases, exact approaches are convenient. For large scale problems, exact approaches are often incapable of optimizing PMSP in the reasonable time. In order to address this problem efficiently, metaheuristic and heuristic methods have been suggested to handle the well-known NP-hard UPMSP. It has been shown that metaheuristic techniques and hybrid algorithms are the most efficient and powerful alternative strategies to solve UPMSPs**.**

Morales et al. [18] suggested an Iterated Greedy Algorithm (IGA) to minimize the overall completion time with setup times in a parallel machine scheduling problem. Bozorgirad and Logendran [19] developed Tabu-search-based algorithms for solving the problem of sequence-dependent group-scheduling using UPMSPs. Also Anagnostopoulos et al. [20] addressed UPMSPs to minimize makespan. To reach a near-optimum solution for this NP-hard with sequence-dependent setup times, SA algorithm was applied.

However, the number of solutions created from the neighborhoods has increased dramatically by raising the number of works (increasing the size of the problem), which may make the algorithm (SA, for example) experience a number of weaknesses (e.g. time, efficient solution and local optimum). Therefore, interest in the development of hybrid metaheuristic algorithms has been growing over the years to solve complex problems. Similar to as a number of ressearchers such as Behnamian et al. [4], Xu et al.[21], Haddad et al. [22], Schaller [23] and Joo et al. [24],have utilized hybrid metaheuristic algorithms to outperform pure metaheuristic algorithms for various applications.

Báez et al. [25] proposed a hybrid algorithm that integrates Greedy Randomized Adaptive Search Procedure (GRASP) and Variable Neighborhood Search (VNS) to reduce the total completion time in UPMSP with dependent setup time. Schaller [16] studied PMSPs with family setup time to minimize lateness. He proposed some techniques to solve the desired problem by rely on GA and Tabu-search.

In the literature, there exist many research studies regarding the application of SA combined with another algorithm to solve PMSPs. Jouhari et al. [2] proposed an integrated approach based on SA and Sine Cosine Algorithm (SCA) to

solve UPMSP with sequence-dependent setup times. It aims to get solutions near to optimal by updating the solutions according to their properties.
Abdeljaoued et al. [26] presented two new heuristics to achieve a suitable solution and one SA metaheuristic for the purpose of improving the solution's quality for PMSPs under resource reservation limitations. A new hybridization of the symbiotic organism search algorithm with SA has been established in another study [5] to address UPMSP with sequence-based setup times.

2.2 Matching theory

Another body of research that inspired this work is collaboration between different machines in PMSPs in order to deal with the high dynamics of problem.
Matching theory (MT) can be summed up in finding the stable matching between different agents by organizing the relations among them. Recently, the application of MT in various fields (e.g. economics and communication) has received increasing attention to handle resource allocation problems [8], [27], [28].
Gu, Y. et al. [29] presented the first comprehensive tutorial to provide an integrated behavior of MT towards engineering problems through defining several classes of matching scenarios and applying them in several applications drawn from the state-of-the-art research in resource allocation in wireless networks.
With the aim of enhancing evolutionary algorithms, as the first work in this direction, Li, K. et al. [30] applied MT and stable matching model for organizing the selection process between sub-problems and solutions for multi-objective evolutionary algorithm based on decomposition, where sub-problems and solutions (two factors) would be able to convey their inclinations over each other.
Despite the importance of this tool to develop low complexity, high performance, and decentralized protocols complex networks [8], to the best of our knowledge, there is no related research in the literature on PMSPs by MT.

2.3 Research gap

The aforementioned literature sections proved the importance of hybrid algorithms to solve UPMSP. Although metaheuristic algorithms are one of the most attractive methodologies in optimization, they cannot guarantee the optimal solution [31]. In this instance, obtaining optimal solutions seems to be very hard, particularly in the case of large-scale problems. Also, the literature showed that the hybridization of a metaheuristic with optimization techniques would greatly increase consistency of the solutions and computational performance. Diversity and convergence are critical for metaheuristic optimization in order to achieve a near-optimum solution. The proposed hybrid metaheuristic benefits from the advantages of MT and SA to deal more effectively with the UPMSP. The SA algorithm is chosen primarily because of its capability to achieve high-quality solutions in combination with its ability to search around the solution area in order not to become trapped in a local optimum [5], [32]. A further advantage of the hybridization method with SA is that it is simple to identify the neighborhood structures [32]. In contrast, to avoid the machines assigning their resources to jobs that are not very suitable for them and vice versa, MT had been applied. Therefore, MT encourages the convergence.
The basic principle behind integrating these two algorithms is to balance the convergence and diversity of the algorithmic search, thereby finding a better optimum schedule for UPMSP.

3. Preliminaries

3.1 Problem formulation

The UPMSP with sequence–dependent setup time can be defined as follows [24].
Suppose that $N = \{1, \ldots, n\}$ indicates the jobs that will be handled on $M = \{1, \ldots, m\}$ unrelated machines. The UPMSP consists of assigning n obtained jobs to m available machines, and to decide the arrangement of the jobs on the machines that can process them in manner that keeps the total completion time is minimized. Any machine can handle any of the jobs exactly once. Additionally, each job $j \in N$ has an associated processing time p_{ji} that relies on the machine $i \in M$ where it will be assigned. Furthermore, if job j is processed just after job k, the machine setup time s_{jki} is needed, where i represents the machine on which jobs are processed.
Considering the above features of the issue, we discuss the MIP formulation for the unrelated parallel machine scheduling with sequence–dependent setup time [24]. The MIP model can be obtained as follows:

$$Min\ Z = C_{max} \qquad (1)$$

S.T:

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j = 1, \ldots, N \qquad (2)$$

$$\sum_{j=0}^{n} y_{jki} = 1 \quad \forall\, k = 1, \ldots, N \tag{3}$$

$$\sum_{k=1}^{n} y_{jki} \le 1 \quad \forall\, j = 1, \ldots, N \tag{4}$$

$$x_{ij} \le a_{ij} \quad \forall\, j, k = 1, \ldots, N \tag{5}$$

$$S_k \ \ge \ \sum_{i=1}^{m} s_{jki}.x_{ik} \ + C_j - V.(1 - y_{jki})$$

$$\forall\, i = 1, \ldots, M, j, k = 1, \ldots, N \tag{6}$$

$$C_j = S_j + \sum_{i=1}^{m} s_{jki}.x_{ij} + \sum_{i=1}^{m} p_j.x_{ij} \quad \forall\, j = 1, \ldots, N \tag{7}$$

$$C_{max} \ge C_j \tag{8}$$

$$y_{jki}, x_{ij} = 0 \ or \ 1 \quad \forall\, i = 1, \ldots, M, j, k = 1, \ldots, N \tag{9}$$

Where, C_j indicates completion time for job *j*, S_j is the starting time of job *j*, s_{jki} is sequence-dependent setup time to handle job *j* after job *k* on machine i , and p_{ji} represents processing time for job *j* on machine i . In addition, $a_{ij} = 1$ if job *j* is available to producing at machine *k*; 0, otherwise. In addition, N represents the number of jobs, M indicates the number of machines, and V is a large positive number. x_{ij} *and* y_{jki} are two binary variables. x_{ij} is equal to 1, if job *j* is assigned to machine *i*; 0, otherwise,y_{jki} is equal to 1, if job k is handled after job *j* on machine *i*; 0, otherwise. The objective function (1) minimizes Makespan. Constraint (2) forces each *j* to be performed only by one machine. Constraint (3) allows that each job is processed after another job. Constraint (4) indicates that there is one job at most to be produced after each job on machine *i*. Constraint (5) specifies that a machine can process the available job. Constraint (6) specifies the time to start the job *j*. Constraint (7) specifies the time for completion of job *j*. Constraint (8) confirms that C_{max} is bigger than the completion time of the work.

3.2 Simulated Annealing (SA) algorithm:

First, we introduce a SA algorithm. Our approach is based upon combining the SA algorithm with MT. The SA algorithm has been inspired from the similarity of the annealing process in metals [33]. It also uses the objective function of an optimization problem as an alternative for the energy of a material. SA has been regarded as improvement heuristic because the initial solution obtained by SA is iteratively improved upon.
Sometimes, worse neighboring solutions are also accepted according to the Boltzmann's probability distribution. This procedure prevents the algorithm to stuck in any local optimum it finds in its early execution. SA performs this in a probabilistic manner. In the initial stages of optimization, worse solutions are adopted with a comparatively high probability, which decreases over time in order to reach convergence. The dynamic threshold for accepting a worse solution is defined by using probability function. By using cooling schedule scheme, which is the main parameter of the SA algorithm, the SA temperature regularly decreases from a moderately high value to near zero. When the algorithm temperature decreases to a pre-determined final value, the SA phase stops. One key weaknesses of SA is that it can require much tuning of the initial temperature and annealing schedule. Additionally, there is a clear gap between the efficiency of the solutions and the required time for calculating them [2], [26].

3.3 Matching theory

MT was first developed in a Nobel Prize winning paper, [30], and found several applications in many areas of search over time (e.g., [34]-[35]). Proposed in economics, stable matching can be an effective way to solve conflicts of interests between selfish agents of the market. The benefits of MT include: (i) an appropriate model for characterizing heterogeneous agent interactions, each having its own category, goal, and information, (ii) the capability to identify common "preferences" in order to deal with the problem complexity, (iii) a suitable solution that accurately represents various stable and optimal goals of the problem, and (iv) an effective algorithmic implementation that is essentially self-organizing and fast-moving. The stable marriage problem (SMP) proposed in [30], considers matching two groups of agents, i.e. men and women. According to his desires, each man ranks the women, and vice versa. If a match involves a woman and a man who are not paired jointly but prefer each other to their allocated partners, it is definitely undesirable. The matching that contains such pair is defined as an unstable matching because they have reasonable motivation to break up; instead they pair each other from their real marriages. Figure 1 shows an example of simple marriage problem, where m1, m2, m3 are men, and, w1, w2, w3 are women, we have listed their order of preference over the other sort around. For example, for m1, [w1, w2, w3] means that m1 ranks w1 first, w2 second, and w3 last. Some couples are linked by robust lines, e.g., (w3, m1), the other couples is linked by dashed lines, e.g., (w2, m1). Obviously, the matching {(w1, m2), (w2, m3), (w3, m1)} is stable whereas the matching {(w1, m2), (w2, m1), (w3, m3)} is unstable. The reason is that m2 prefers w3 to w1, and w3 prefers m2 to her spouse m3.

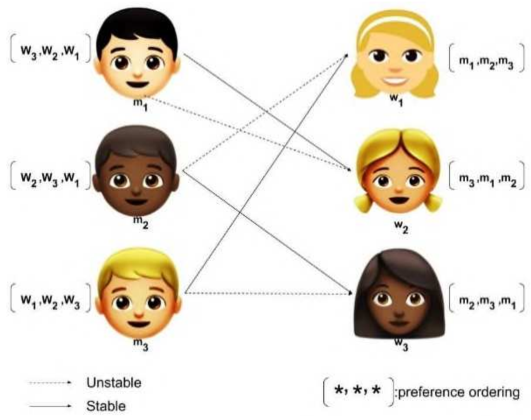

Figure 1. A simple marriage problem

4. Solution Approach

This section presents the proposed approach for solving the UPMSP with sequence-dependent setup times. The proposed approach is based on combining the SA algorithm with MT to create a more efficient local search capability for the SA algorithm. Generally, it begins by creating a random solution for UPMSP. The SA then creates a new solution v_i from the neighbor($N(x_i)$) of the original solution x_i. The objective function (minimizing Cmax) is calculated for these two solutions. Then the difference ΔE between the fitness (v_i) and fitness (x_i)is calculated. $\Delta E < 0$ implies that v_i is better than x_i ; then v_i is the SA current solution, and the temperature T will be reduced after a new iteration is added. If $\Delta E > 0$, it implies v_i is worse than x_i; then x_i would be acceptable with the probability $e^{\frac{-\Delta E}{T}}$. MT is then used to improve x_i by using its technique. The above-mentioned procedures are described in more detail below.

4.1 Initial Solution

We consider to create a random integer solution $x_i = [x_1, x_2, \ldots, x_{N_j}]$ where N_j is the number of jobs and its value takes from 1 to N_j+N_i-1. We have, for example, 3 machines and10 jobs; thereafter, the solution representation x_i could be presented as [x_1,x_2, . . . , x_{N_j}] = [7 6 12 3 10 8 11 5 4 1 2 9]. This indicates that machine number 1 processes jobs 7 and 6, machine number 2 process jobs 3, 10 and 8, and machine number 3 processes jobs 5, 4, 1, 2 and 9, where 12 and 11 are delimiters. The next step is to evaluate the fitness function (Cmax) for solution x_i by employing Eq. (1), and choose the best one.

4.2 Solution Update

The solution update begins by choosing solution v_i from the solution neighbor x_i and calculates fitness (v_i) (the fitness function). Then the difference ΔE between the fitness (v_i) and fitness (x_i) is calculated. $\Delta E < 0$ implies that v_i is better thanx_i; then v_i is the SA current solution. If $\Delta E > 0$, it implies that v_i is worse than x_i then x_i would be acceptable with the probability $e^{\frac{-\Delta E}{T}}$. The MT is using to allocate the job to the best machine that can do it. We assume that each machine has list of preferences. To make it easier, the only factor that is taken into account is the monotonic preference connection; this assumes that each machine (agent) is able to put all the jobs on the alternate side sequentially. Let $t^i \succ s^j t^k$ denotes that machine j prefers job t^i to t^k, and $s^i \succ t^j s^k$ indicates that job t^j prefers machine s^i to s^k. A matching is considered to be stable under the following two circumstances: If job *t* could not match any machine, then no machine *s* tends to choose *t* over its current paired job.

If job *t* matches a machine other than *s*, then *t* chooses its current match over *s*, so that s goes for its present pair, and not for *t*. In other words, pairing t with s would not offer them a better situation than when they are with their current given pairs.

To make it simple, the current research defines the preference list in the following way:

1) A machine selects the job that uses its max capability, considering the cost of every capability in the system.
2) A job prefers the machine that completes it with high performance.

The mentioned preference values enable us to simply come up with two preference arranging matrices of machines and jobs ∇_t, ∇_s . The parts of the i^{th} row in ∇_t are the preference arranging of t^i on all the tasks in t, and the parts of the j^{th} row in ∇_s are the preference arranging of s^j on all the machines in s. In such an ascending order of choice preferences, all of them arising. The pseudo-code of calculating∇_t, ∇_s is presented in Algorithm 1.

```
Algorithm 1: Computing the preference list
Initialization:
1:        $T = \{t_i\}_{i=1}^{m}, S = \{s_j\}_{j=1}^{n}, C = \{c_k\}_{k=1}^{l}$
2:        Boolean matrix; $B_t = [t, c_k]_{i=1}^{m}, B_s = [s, c_k]_{i=1}^{n}$
3: Performance matrix $PM_{ji} = [s, t]_{j=1,i=1}^{n,m}$
4: Cost matrix $CO_{ji} = [s, t]_{j=1,i=1}^{n,m}$
Dynamic proposals:
5:                 For i: 1 to m do
6:                   For j: 1 to n do
7:                           if $B_{ti} \cap B_{sj} = B_{ti}$ then
8:                   $\Delta s = sum((B_{sj} - B_{ti}) * CO_{ji})$
9:                           End If
10:                          $\nabla_t = PM_{ji}$
11:                  End for
12:                End for
13: Sort each row of $\Delta s$ in ascending order and each row of $PM$ in descending order, and keep the sorted indices in $\nabla_t, \nabla_s$.
```

Figure 2. Pseudo-priority list code

The proposed method in the Stable Marriage Algorithm (SMA) is applied in Algorithm 2 for consistent matching between machines and jobs. Algorithm 2 starts by releasing all machines and jobs (i.e., machines and jobs are not matched to any one on the other side). In Algorithm 2, $\overline{T}_{matchlist} = \emptyset$ indicates that the job is set free, and 1 indicates that it is connected. In a comparable scenario, $\overline{T}_{matchlist}[j]$ represents that t^j is free, and $\emptyset(i,j) = 0$ indicates that s^i has not offered to choose t^j formerly as a partner. On the other hand, 1 indicates that it has carried out so. Line 6 requests Algorithm 1 to calculate ∇_s, ∇_t. The algorithm randomly selects machine s^i and figures out the highest−ranking jobt^j , to which s^i has still not suggested; this happens in the main while-loop when several machines are still set free (line 8 to line 9). If t^j is also free, then s^i and tj would be connected with each other (line 11 to line 13). Under the condition that t^j is not free, t^j separates from its present pair s^k and pairs with s^i provided that $s^i \succ t^j\, s^k$ (line 15 to line 17). The main while-loop finishes exactly when N machine-job couples have been created; putting it in another way, it gives no free machine.

```
Algorithm 2: Stable matching game
Initialization:
1:        $T = \{t_i\}_{i=1}^{m}, S = \{s_j\}_{j=1}^{n}, C = \{c_k\}_{k=1}^{l};$
2: $T_{matchlist} = \emptyset,$ all tasks are free;
3: $S_{matchlist} = \emptyset,$ all systems are free;
4:$\emptyset(i,j) = 0;$
5: Cost matrix$CO_{ji} = [s, t]_{j=1,i=1}^{n,m};$
6:        $[\nabla_s, \nabla_t]$ computing preference list
7:        While some tasks are still not matched, do
8:                  Randomly choose task from $\overline{T}_{matchlist}$;
9:                  Find $t^i$'s most preferred system $s^j$ with $\emptyset(i,j) = 0;$
10:            $\emptyset(i,j) \rightarrow 1;$
11:                  If $\overline{T}_{matchlist}[j] = 1$ ,do
12:                    $t^i$ and $s^j$ are set to be paired;
13:                    $T_{matchlist} = T_{matchlist} \cup \{t^i\}, S_{matchlist} = S_{matchlist} \cup \{s^j\};$
14:       else
15:            if $t^i \succ s^j\, t^k$ (the current partner of $\boldsymbol{s}^j$ ), then
```

```
16:                 $t^i$ and $s^j$ are set to be paired;
17:                 $\bar{T}_{matchlist}[i] = 1$ , $\bar{T}_{matchlist}[k] = 1$
18:             End
19:         End
20:     End
```
Figure 3. Pseudo-code of the stable marriage algorithm

5. Computational Results

In this part, we introduce a set of tests to show that the suggested meta-heuristic overrates the other meta-heuristics discussed in this paper. Therefore, the outcome of the suggested algorithm is contrasted with the other three algorithms: GA, SA, and Population-based Simulated Annealing (SA_P).

5.1 Cases of tests

Since we did not found any prominent benchmark in the literature for the problem studied, we generated 16 test instances for two problems, and also allocated to each problem its own machine and job group. Numbers of machines for two problems are 4, 6, 8 and 16. The first problem has 16, 30, 40 and 80 jobs, and the second problem has 40, 80, 100 and 200 jobs. The processing times were uniformly generated ($p_{ij} = U[10,60]$) for 16 instances.
Each proposed meta-heuristic was addressed with the MATLAB software by a computer with 3 GHz processor and 1 GB of RAM. To determine the parameter settings of meta-heuristic, we considered common parameter settings in the literature. For the suggested GA, the mutation probabilities and crossover were set as 0.15 and 0.85, respectively. We initially set the temperature and the temperature reduction rate of the proposed SA_MATCH, SA and SA_P as 10 and 0.8, respectively.

Table 1. Average of Cmax values of the issues for each algorithm

Machines	Jobs	*SA_MATCH*	*SA*	SA_P	*GA*
4	16	115.4	116.4	116.8	124.4
	30	181.8	184.6	184	197.8
	40	319.2	324.6	323.6	330.6
	80	526.4	536	526.6	556.2
6	16	60.6	62	64.4	70.2
	30	164.2	164.8	168.4	182.8
	40	232.4	232.6	237	247.4
	80	338	349.2	358.4	413.4
8	40	160	160.2	172.6	178.6
	80	241.2	257.6	257.4	292
	100	305	347	336	367.4
	200	651.8	750.8	666	791.9
16	40	58	69.2	58.8	81.6
	80	121.2	142.4	148	160.4
	100	153.4	191.8	188	215.4
	200	304.2	420.6	364.6	433.4

5.2 Comparative analysis of algorithms

To evaluate the quality of the results of the applied algorithms, we considered the Relative Percentage Deviation (RPD) [36], which considers the most common performance measure and is computed as follows:

$$\text{Relative percentage deviation } (RPD) = \frac{algorithm_{SOL} - Best_{SOL}}{Best_{SOL}} * 100\%$$

Where, $algorithm_{SOL}$ is the average value of the solution, which is obtained by an assigned method, and $Best_{SOL}$ is the best solution obtained amongst all the other ways or the best known solution. RPD value closer to zero produces the best results. $Best_{SOL}$ is obtained by running a particular algorithm five times for a specific problem, and $algorithm_{SOL}$ is the final average solution for all the five runs given by the algorithm.

Overall, we tested the algorithms for 320 runs (four algorithms * 16 examples * five runs per instance). Table 1 provides the results of this study. Each row indicates example test, and its columns indicate the values of mean RPD and MAD for the corresponding algorithm.

Table 2. Mean RPD and MAD (%) for SA_MATCH, SA, SA_P and GA algorithms

Machines	Jobs	SA_MATCH		SA		SA_P		GA	
		RPD	MAD	RPD	MAD	RPD	MAD	RPD	MAD
4	16	2.12	1.28	3.01	2.72	3.36	0.48	10.08	7.92
	30	1	1.04	2.43	3.12	2.22	2.4	9.84	6.56
	40	3.73	2.31	0.96	4.24	4.4	4.04	5.96	15.36
	80	1.42	4.48	3.27	10	1.46	2.4	7.16	10.64
6	16	2.71	1.28	5.08	2	9.15	1.28	18.98	3.52
	30	1.98	5.84	1.86	4.4	4.35	2.32	13.54	12.48
	40	0.17	1.92	0.26	1.28	2.15	2	6.63	8.88
	80	0.29	0.8	3.6	5.36	3.41	8.08	22,67	10.08
8	40	3.15	1.44	3.22	4	11.35	1.76	15.22	7.12
	80	2.2	4.24	9.15	6.56	9.07	2.32	23.73	25.2
	100	1.66	5.2	15.66	6	12	1.2	29.48	14.33
	200	1.84	8.56	17.31	15.36	4.66	3.6	34.96	13.95
16	40	2.34	2.17	10.75	3.76	15.68	0.82	31.34	7.16
	80	1	3.04	18.66	1.12	23.33	3.2	33.66	7.68
	100	1.07	3.12	31.36	9	28.76	1.6	47.53	11.68
	200	1.06	1.36	39.73	8.48	21.13	2.72	49.93	13.43

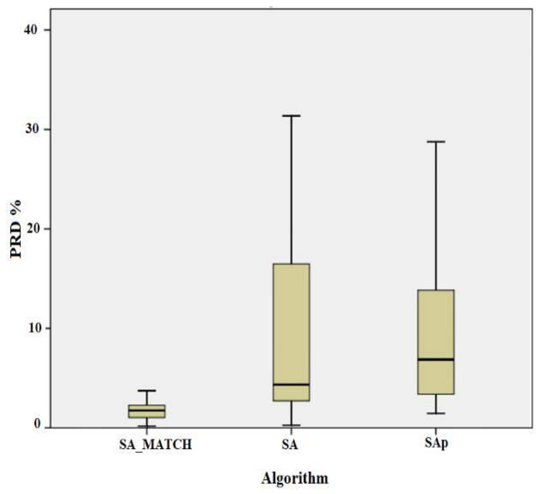

Figure 4. Mean plots and 95% confidence level Tukey HSD intervals for the SA_MATCH, SA and SA_P algorithms.

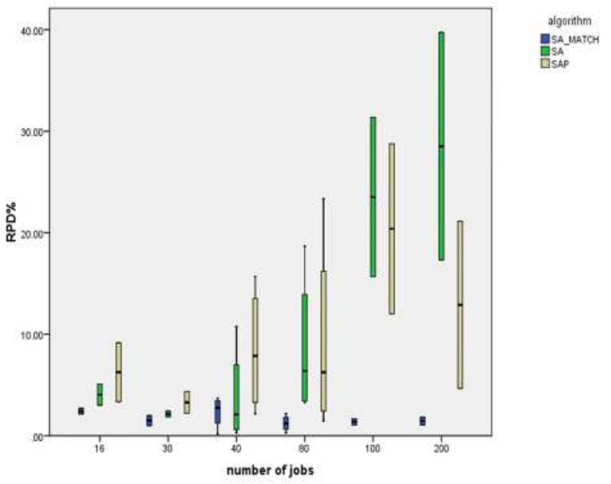

Figure 5. Mean plots and 95% confidence level Tukey HSD intervals for the SA_MATCH, SA and SA_P algorithms with different number of jobs

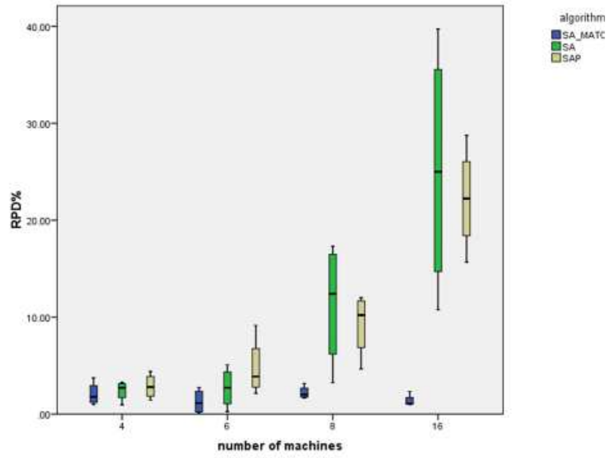

Figure 6. Mean plots and 95% confidence level Tukey HSD intervals for the SA_MATCH, SA and SA_P algorithms with different number of machines.

Table 1 illustrates the results of the average of Cmax. As clearly shown, the proposed algorithm can obtain the smallest Cmax among all the test problems, especially for big size problems.

Table 2 presents the mean RPD and MAD of the five replications for 4, 6, 8 and 16 machines for SA_MATCH, GA, SA and SA_P. The results of the proposed algorithm give the best performance with a lower value of RPD and MAD. On average, it gives a better result in comparison with its opponents.

In order to justify the outcomes, a set of statistical significance tests of the perceived variances in the RPD values of each applied algorithm were carried out. Since SA_MATCH, SA and SA_P give very lower RPD with smaller variance than GA, it is much demanding to point out the variances between SA_MATCH, SA andSA_P. Therefore, we decided to eliminate GA from the statistical tests. Figure 3 displays the mean plots and Tukey HSD intervals at 95% confidence level for the all problems given in Table 2. Figure 3 shows the mean plots and 95% confidence level Tukey HSD intervals for all issues given in Table 2. Figure 3 illustrates that there are variations statistically observable between the RPD values in all the applied algorithms.

The plots show that the RPD in SA_MATCH is remarkably different with that in SA_MATCH, SA and SA_P, because the number of jobs per machine and the number of machines increase, as illustrated in Figs 4 and 5. These results are due to the fact that the proposed hybrid algorithm does not check for the filled spaces. In addition, it has both intensification and diversification mechanisms and works well at the same time.

Hence, SA_MATCH proves to be an effective approach for solving parallel machine scheduling for any problem size comparing to GA, SA and SA_P.

The proposed algorithm has the following advantages: speed of execution, ease of implementation and better response. The outcomes of the implementation of the suggested algorithm for several problem instances showed that it works better than the SA, GA and SA_P algorithms, especially in large size problems.

Among the benefits of the proposed meta-heuristic are the following:

• simple algorithm structure
• ability to evade local optimizations
• extensive searchable answer space
• algorithm convergence mechanism in problem generation method.

6. Conclusion

The presented work presents a new brand approach to improve the efficacy of scheduling in a parallel machine in manufacturing systems. This algorithm is grounded upon integrating the SA algorithm with the matching theory (MT). The most important part of this hybrid algorithm is to produce new answer formulas from the best current answer. Another feature is allocation of jobs to the best machine that can do it. We assumed that each machine has a list of preferences, meaning that one machine can rank all of its jobs.
For simplicity, this article defines priority in the following way. A machine prefers a job with small processing time. Conversely, a job prefers a system that has less setup time.
One of the main strength points of the algorithm is that it has both diversification and convergence mechanisms and works well at the same time.
The results of the suggested algorithm, with low RPD values, produced the best efficiency and provided better results than the average of its opponents. It was also found that the proposed algorithm has a significant difference with other algorithms in terms of providing near optimal solution, especially for large size problems.
In conclusion, SA_MATCH proved to be an efficient method for solving parallel machine scheduling for any problem size comparing to GA, SA and SA_P.
As future works, this algorithm can be expanded to be applied to various problems, as well as One-to-Many Matching or Many-to-Many Matching.

REFERENCES

1. Mir, M. S. S., Rezaeian, J., & Mohamadian, H. (2020). Scheduling parallel machine problem under general effects of deterioration and learning with past-sequence-dependent setup time: heuristic and meta-heuristic approaches. **Soft Computing**, 24(2), 1335-1355.
2. Jouhari, H., Lei, D., AA Al-qaness, M., Abd Elaziz, M., Ewees, A. A., & Farouk, O. (2019). Sine-Cosine Algorithm to Enhance Simulated Annealing for Unrelated Parallel Machine Scheduling with Setup Times. **Mathematics**, 7(11), 1120.
3. Naderi, B., Tavakkoli-Moghaddam, R., Sadati, A., & Mohammadi, M. (2017). Solving a new multi-objective unrelated parallel machines scheduling problem by hybrid teaching-learning based optimization. **International Journal of Engineering**, TRANSACTIONS B: Applications, 30(2), 224-233.
4. Bicakci, P. S., & Kara, İ. (2019). A New Formulation for the Single Machine Order Acceptance and Scheduling Problem with Sequence-Dependent Setup Times. **International Journal of Supply and Operations Management**, 6(2), 159-167.
5. Ezugwu, A. E. (2019). Enhanced symbiotic organisms search algorithm for unrelated parallel machines manufacturing scheduling with setup times. **Knowledge-Based Systems**, 172, 15-32.
6. Li, X., Gao, L., & Li, W. (2012). Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling. **Expert Systems with Applications**, 39(1), 288-297.
7. Dasilva, L. A., Bogucka, H., & Mackenzie, A. B. (2011). Game theory in wireless networks. **IEEE Communications Magazine**, 49(8), 110-111.
8. Bayat, S., Li, Y., Song, L., & Han, Z. (2016). Matching theory: Applications in wireless communications. **IEEE Signal Processing Magazin**e, 33(6), 103-122.
9. McNaughton, R. 1959. “Scheduling with Deadlines and Loss Functions.” **Management Science** 6: 1–12.
10. Cheng, T. C. E., & Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. **European Journal of Operational Research**, 47(3), 271-292.
11. Mokotoff, E. (2001). Parallel machine scheduling problems: A survey. **Asia-Pacific Journal of Operational Research**, 18(2), 193.
12. Kravchenko, S. A., & Werner, F. (2011). Parallel machine problems with equal processing times: a survey. **Journal of Scheduling**, 14(5), 435-444.

13. Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. **European Journal of Operational Research**, 230(3), 449-463.
14. Bülbül, K., & Şen, H. (2017). An exact extended formulation for the unrelated parallel machine total weighted completion time problem. **Journal of Scheduling**, 20(4), 373-389.
15. Wu, L., & Wang, S. (2018). Exact and heuristic methods to solve the parallel machine scheduling problem with multi-processor tasks. **International Journal of Production Economics**, 201, 26-40.
16. Mokotoff, E. (2004). An exact algorithm for the identical parallel machine scheduling problem. **European Journal of Operational Research**, 152(3), 758-769.
17. Li, X., Yalaoui, F., Amodeo, L., & Chehade, H. (2012). Metaheuristics and exact methods to solve a multiobjective parallel machines scheduling problem. **Journal of Intelligent Manufacturing**, 23(4), 1179-1194.
18. Morales, M. F., Angel-Bello, F., & Alvarez, A. (2016). An Iterated Greedy Algorithm for minimizing the total completion time in a parallel machine scheduling problem with dependent setup time. In Technical Report. **Graduate Program in Systems Engineering**, UANL San Nicolás de los Garza, México.
19. Bozorgirad, M. A., & Logendran, R. (2012). Sequence-dependent group scheduling problem on unrelated-parallel machines. **Expert Systems with Applications**, 39(10), 9021-9030.
20. Anagnostopoulos, G. C., & Rabadi, G. (2002, June). A simulated annealing algorithm for the unrelated parallel machine scheduling problem. **In Proceedings of the 5th Biannual world automation congress** (Vol. 14, pp. 115-120). IEEE.
21. Xu, Z., Zou, Y., & Kong, X. (2015). Meta-heuristic algorithms for parallel identical machines scheduling problem with weighted late work criterion and common due date. **SpringerPlus**, 4(1), 782.
22. Haddad, M. N., Cota, L. P., Souza, M. J. F., & Maculan, N. (2014, April). AIV: A Heuristic Algorithm based on Iterated Local Search and Variable Neighborhood Descent for Solving the Unrelated Parallel Machine Scheduling Problem with Setup Times. **In ICEIS** (1) (pp. 376-383).
23. Schaller, J. E. (2014). Minimizing total tardiness for scheduling identical parallel machines with family setups. Computers & Industrial Engineering, 72, 274-281.
24. Joo, C. M., & Kim, B. S. (2015). Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability. **Computers & Industrial Engineering**, 85, 102-109.
25. Báez, S., Angel-Bello, F., Alvarez, A., & Melián-Batista, B. (2019). A hybrid metaheuristic algorithm for a parallel machine scheduling problem with dependent setup times. **Computers & Industrial Engineering**, 131, 295-305.
26. Abdeljaoued, M. A., Saadani, N. E. H., & Bahroun, Z. (2018). Heuristic and metaheuristic approaches for parallel machine scheduling under resource constraints. **Operational Research**, 1-24.
27. Roth, A. E. (2008). Deferred acceptance algorithms: History, theory, practice, and open questions. **International Journal of game Theor**y, 36(3-4), 537-569.
28. Sng, C. T., & Manlove, D. F. (2010). Popular matchings in the weighted capacitated house allocation problem. **Journal of Discrete** Algorithms, 8(2), 102-116.
29. Gu, Y., Saad, W., Bennis, M., Debbah, M., & Han, Z. (2015). Matching theory for future wireless networks: Fundamentals and applications. **IEEE Communications Magazine**, 53(5), 52-59.
30. Li, K., Zhang, Q., Kwong, S., Li, M., & Wang, R. (2013). Stable matching-based selection in evolutionary multiobjective optimization. **IEEE Transactions on Evolutionary Computatio**n, 18(6), 909-923.
31. Villa, F., Vallada, E., & Fanjul-Peyro, L. (2018). Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. **Expert Systems with Applications,** 93, 28-38.
32. Hamzadayi, A., & Yildiz, G. (2017). Modeling and solving static m identical parallel machines scheduling problem with a common server and sequence dependent setup times. **Computers & Industrial Engineering**, 106, 287-298.
33. Maliki, F., Souier, M., Dahane, M., & Sari, Z. (2017). The Use of Metaheuristics for a Stochastic Supply Chain Design Problem's Resolution–A Comparison Study–. **International Journal of Supply and Operations Management**, 4(3), 193-201.
34. Xu, C., Gao, C., Zhou, Z., Chang, Z., & Jia, Y. (2016). Social network-based content delivery in device-to-device underlay cellular networks using matching theory. **IEEE Access**, 5, 924-937.
35. Jiao, Z., & Tian, G. (2017). The Blocking Lemma and strategy-proofness in many-to-many matchings. **Games and Economic Behavior**, 102, 44-55.

36. Al-Shayea, A. M., Saleh, M., Alatefi, M., & Ghaleb, M. (2020). Scheduling Two Identical Parallel Machines Subjected to Release Times, Delivery Times and Unavailability Constraints. **Processes**, *8*(9), 1025.