DOI: 10.24996/ijs.2020.SI.1.17

ISSN: 0067-2904

Minimizing the Total Completion Time and Total Earliness Time Functions for a Machine Scheduling Problem Using Local Search Methods

Faez Hassan Ali*, Aseel Aboud Jawad
Deparment of Mathematics, College of Science, Mustansiryah University, Baghdad, Iraq

Received: 2/12/ 2019 Accepted: 15/ 3/2020

Abstract
In this paper we investigate the use of two types of local search methods (LSM), the Simulated Annealing (SA) and Particle Swarm Optimization (PSO), to solve the problems $1//(\sum C_j, \sum E_j)$ and $1//\sum C_j + \sum E_j$. The results of the two LSMs are compared with the Branch and Bound method and good heuristic methods. This work shows the good performance of SA and PSO compared with the exact and heuristic methods in terms of best solutions and CPU time.

Keywords: Machine Scheduling Problem, Multiple Objective Functions, Simulated Annealing, Particle Swarm Optimization.

تصغير دالة وقت الاتمام الكلي ووقت التبكير الكلي لمسألة جدولة الماكنة باستخدام طرق البحث المحلية

فائز حسن علي* ، اسيل عبود جواد
كلية العلوم, قسم الرياضيات, كلية العلوم ، الجامعه المستنصريه بغداد, العراق.

الخلاصه
في هذا البحث تم تطبيق نموذجين من طرق البحث المحلية (LSM) وهي محاكاة التلدين (SA) وامثلية السرب الجزيئي (PSO) لحل مسألة $1//(\sum C_j, \sum E_j)$ ومسألة $1//\sum C_j + \sum E_j$. تم مقارنة النتائج لطرق البحث المحلية مع طريقة التفرع والتقييد (BAB) وبعض الطرق التقريبية الجيدة. هذا العمل اثبت كفاءة طريقتي SA وPSO بالمقارنة مع طرق الحل التام والطرق التقريبية من حيث الحلول الجيدة ووقت التنفيذ.

1. Introduction

Scheduling, generally speaking, means to assign machines to jobs in order to complete all jobs under the imposed constraints. The problem is to find the optimal processing order of these jobs on each machine to minimize the given objective function. There are two general constraints in the classical scheduling theory [1]. Each job is to be processed by, at most, one machine at a time and each machine is capable of processing at most one job at a time. A schedule is feasible if it satisfies the two general constraints, and also if it satisfies the various requirements relating to the specific problem type. The problem type is specified by the machine environment, the job characteristics and an optimality criterion.
For the simultaneous objective function, there are two types; the first one is to find the sum of these objectives, while the second typically generates all efficient schedules (set of Pareto optimal solutions) and selects the one that yields the best composite objective function value of the criteria [2].

*Email: faezhassan@uomustansiriyah.edu.iq

There are several solving methods for the Machine Scheduling Problem (MSP) which are classified into the Complete Enumeration Method (CEM), Branch and Bound (BAB) method, Dynamic Programming (DP) method, heuristic methods, and metaheuristic (local search) methods.

Several studies on the local search methods proved that they led to significantly better results than those obtained from the traditional heuristics if they are implemented carefully [3]. Local search heuristics are built upon observations of processes in the physical and biological sciences [4]. In 2012, Abdul-Razaq et al. [5] suggested a new development to the methods of the scheduling in flow shop to minimize makespan problems. Also, they applied two local search methods, namely the GA and PSO algorithms, on flow shop problems. In 2015, Ali [6] used some types of LSM (GA, PSO and Bees Algorithm (BA)) to solve some types of combinatorial optimization problems, such as multicriteria MSP. Section two introduces two important local search methods (SA and PSO). In section three, the mathematical formulations of $1//(\sum C_j, \sum E_j)$ and $1//\sum C_j + \sum E_j$ problems are discussed. The practical and comparative results are introduced in section four. Lastly, in section five, most important conclusions and some recommendations were presented.

2. Local Search Methods

Local search methods (LSMs) form a very general class of heuristic methods to treat discrete optimization problems (DOP). Such problems are given by a finite set *S* of feasible solutions and an objective function $f{:}S \to R$. The goal is to find a solution with minimal objective value, i.e., we look for a solution $s^* \in S$ with

$$f(s^*) = \underset{s \in S}{Min}\{ f(s) \}$$

Generally speaking, LSMs move iteratively through the solution set *S* of a DOP. Based on the current, and may be on the previous visited solutions, a new solution is chosen. The basic structure of the local search algorithm is as follows:

Choose an initial solution;
REPEAT
Choose a solution from the neighborhood of the current solution and move to this solution;
UNTIL stopping criteria are met.

Evolutionary Algorithms (EAs) or **Local Search Methods** have been shown to be successful for a wide range of optimization problems [1].

2.1 Simulated Annealing (SA)

Simulated annealing is an algorithmic method that is able to escape from local minima. It is a randomized local search method for two reasons: First, from the neighborhood of a solution a neighbor is randomly selected. Second, in addition to better-cost neighbors, which are always accepted if they are selected, worse-cost neighbors are also accepted, although with a probability that is gradually decreased in the course of the algorithm's execution. The randomized nature enables asymptotic convergence to optimum solutions under certain mild conditions. Nevertheless, the energy landscape, which is determined by the objective function and the neighborhood structure, may admit many and/or "deep" local minimum. Therefore, avoiding local minima is a crucial part of the performance of the algorithm [7, 8].

SA algorithm starts to work by generating random initial solution (s), then the difference $\Delta = F(s') - F(s)$ and neighbor (s') in the objective function are calculated. If $\Delta < 0$, the neighbor (s') will be accepted to be the new solution in the next iteration since it has a better function value. If the objective function value does not decrease (i.e.$\Delta \geq 0$), the generated neighbor may also be accepted with a probability $\exp(-\Delta/T)$, where T is a control parameter called temperature. This temperature is always reduced by a cooling technique in every iteration. As a stopping criteria, one may use e.g. a given number of iteration, a time limit or a given number of iterations without an improvement of the best objective function value. In the first two cases, one must adjust the cooling scheme in such a way that SA stops with a small temperature [3]. Let *B* be an integer s.t. $B \in [2,5]$, $N^*(s)$ be the neighborhood of s, $p(\Delta, t_k)$ be the probability that depends on the exponential function.

Algorithm (1): **Simulated Annealing (SA)**

```
Step(1): Select an initial solution $s \in S$, $s^*=s$; select an initial temperature $t_0>0$; $K=0$, $G=0$;
Step(2): Define $B$; choose $s' \in N^*(s)$; $\Delta = f(s') - f(s)$;
$p(\Delta, t_k) = \exp(-\Delta / t_k)$;
```

If $\Delta \leq 0$, then $s=s'$, and if $f(s)<f(s^*)$,then $s^* = s$; else ($\Delta>0$) ;
If a random number of $[0, 1] \leq p\,(\Delta, t_k)$,then $s=s'$; $G=G+1$,
Step(3): If $G \leq B$, **then** return to step (2),
Step(4): Update temperature **;**$k=k+1$**;** return to step (2**)** until some stopping criteria are met.

2.2 Particle Swarm Optimization (PSO)

PSO has found applications in a lot of areas. In general, all the application areas that the other evolutionary techniques are good at are good application areas for PSO [9].
PSO was originally developed by two specialist; first, the social-psychologist J. Kennedy and, second, the electrical engineer R. Eberhart in 1995 [10]. It emerged from earlier good experiments with algorithms that modeled the "flocking behavior" seen in many swarms of birds. The suggested algorithm is an optimization algorithm which falls under the evolving algorithms umbrella that covers many algorithms as well.
PSO is a very simple concept which can be implemented without complex data structures. No costly or complex mathematical functions are used, and it doesn't require a great amount of memory [11]. PSO possesses a fast convergence, only a small number of control parameters, very simple computations, and good performance, with the lack of derivative computations made it an attractive option for solving the problems.
The **PSO algorithm** depends on the following two relations:

$$v_{id} = w^* v_{id} + c_1 * r_1 * (p_{id} - x_{id}) + c_2 * r_2 * (p_{gd} - x_{id}) \quad \ldots(1.a)$$
$$x_{id} = x_{id} + v_{id} \quad \ldots(1.b)$$

where w is the inertia weight for convergence, c_1 and c_2 are positive constants, r_1 and r_2 are random functions in the range [0,1], $X_i=(x_{i1},x_{i2},\ldots,x_{id})$ represents the i^{th} particle, $P_i=(p_{i1},p_{i2},\ldots,p_{id})$ represents the best previous position (pbest; the position giving the best fitness value) of the i^{th} particle; the symbol g represents the index of the best particle among all the particles in the population, $V_i=(v_{i1},v_{i2},\ldots,v_{id})$ represents the rate of the position change (velocity) for particle i [9]. The PSO algorithm is as follows.

Algorithm (2): **Particle Swarm Optimization (PSO) algorithm**

Step(1): Initialize a population of particles with random positions and velocities on d-dimensions in the problem space.
Step(2): For each particle, evaluate the desired optimization fitness function in d variables.
Step(3): Compare particle's fitness evaluation with its pbest. If the current value is better than pbest, then set pbest equal to the current value, and p_i equals to the current location x_i.
Step(4): Identify the particle in the neighborhood with the best success so far, and assign its index to the variable g.
Step(5): Change the velocity and position of the particle according to equations (1.a) and (1.b).
Step(6): Loop to step (2) until a criterion is met.

The main parameters which affect the good performance of PSO are as follows [10]:
1. The number of particles in a specified swarm affects the run-time of algorithm, thus there is a balance between the number of particles and the speed of the algorithm.
2. The parameter maximum velocity (v_{max}) limits the maximum jump of a particle in the swarm.
3. In equation (1.a), the inertia weight (w) is considered critical for PSO's convergence behavior. This parameter controls the impact of the previous velocities on the current one.
4. In equation (1.a), the parameters c_1 and c_2, considered as fine-tuning, may result in faster convergence and alleviation of local minima. It is better to choose a larger cognitive parameter c_1 than a social parameter c_2 but with $c_1 + c_2 = 4$.
5. In equation (1.a), r_1 and r_2 are used to maintain the diversity of the population, which are uniformly distributed in the interval [0,1].

3. Total Completion Time and Total Earliness Time ($\sum C_j, \sum E_j$)

The object can be described as a set of n jobs $N=\{1,2,\ldots,n\}$ on a single machine to find $\sigma \in S$ (where S is the set of all feasible schedules), so they can be useful to specify whether that minimizes the multi-criteria $(\sum C_j, \sum E_j)$. The $1//(\sum C_j, \sum E_j)$ problem can be written as [12]:

$$\left.\begin{array}{ll} Min\,\{\sum C_j, \sum E_j\} & \\ \text{Subject to,} & \\ C_j \geq\ p_j, & j = 1,2,\dots,n. \\ C_j = C_{(j-1)} + p_j, & j = 2,3,\dots,n. \\ E_j \geq d_j - C_{j,} & j = 1,2,\dots,n. \\ E_j \geq 0, & j = 1,2,\dots,n. \end{array}\right\} \quad \dots(P)$$

For the P-problem, we can deduce a sum of the two objectives to obtain the $1//\sum C_j + \sum E_j$ problem which is formulated as follows:

$$\left.\begin{array}{ll} Min\{\sum C_j + \sum E_j\} & \\ \text{Subject to,} & \\ C_j \geq p_j, & j = 1,2,\dots,n. \\ C_j = C_{(j-1)} + p_j, & j = 1,2,\dots,n. \\ E_j \geq d_j - C_j, & j = 1,2,\dots,n. \\ E_j \geq 0, & j = 1,2,\dots,n. \end{array}\right\} \quad \dots(P_1)$$

4. Practical Results of the Implementing Local Search Methods (LSMs)

In this section, we will apply the two proposed LSMs; these methods are SA and PSO. The following parameters are applied for both SA and PSO:

1. For SA, cooling rate is 0.95, temperature is 10000, and final temperature is 0, R as a uniform random and some hundreds number of generations.
2. While for PSO, which is used for the first time for multicriteria MSP, from our experience, the following parameters are preferred to be used: number of particles (N_par=20,30), maximum velocity (V_{max}=Number of jobs (n)), minimum velocity (V_{min}=1), inertia weight ($w \in [0.4,0.9]$), first acceleration parameter ($c_1 \in [0.5,2]$), second acceleration parameter (c_2=c_1), diversity of the population maintenance (random r_1, $r_2 \in [0,1]$) and some hundreds number of generations.

It is important to mention that the results of applying all solving methods are tested for five experiments.

The values of p_j and d_j for all experiments are generated randomly s.t. $p_j \in [1,10]$ and

$$d_j \in \begin{cases} [1,30], & 1 \leq n \leq 29 \ , \\ [1,40], & 30 \leq n \leq 99 \ , \\ [1,50], & 100 \ \leq n \leq 999\,, \\ [1,70], & \text{otherewise.} \end{cases}$$

under the condition that $d_j \geq p_j$, for j=1,..,n.

Before showing all the results, we introduce some important abbreviations:

Av	:	Average.
AT/S	:	Average Time per second.
AAE	:	Average Absolute Error.
ASOF	:	Average Single Objective Function.
AMOF	:	Average Multi Objective Function.
R	:	0 < Real < 1.
F	:	Objective Function of P-problem.
F_1	:	Objective Function of P_1-problem.
ACT	:	Average of Complete Time.
ABT	:	Average of Best Time.

In this section, we apply SA and PSO for some chosen n with the following initial solutions:

1. Start with **a** random initial solution for SA and PSO.
2. Start with initial solutions for SA and PSO, obtained from the two heuristics, namely the SPT-MST-SCSE for P-problem and DR-SCSE for P_1-problem.[12].

We use the notations LSM (ob,i) to specify the type of LSM (SA or PSO) which be used, ob for the objective function (F or F_1) for the problem (P or P_1), respectively, and i=1,2 for the kind of initial solution (random or chosen initial) from the heuristic method. For example, PSO(F_1,2) means that LSM is PSO, the problem is P_1 and the initial solution is obtained from SPT-MST-SCSE. While SA(F,1) means that LSM is SA, the problem is P and the initial solution is random.

4.1 Comparison Results of P-problem.

Tables- (1 and 2) show the comparison results of applying SA(F,1) with SA(F,2) and PSO(F,1) with PSO(F,2) for P-problem for chosen *n*.

Table 1-Comparison results of applying SA(F,1) and SA(F,2) for P-problem for different *n*.

n	SA(F,1)		SA(F,2)	
	AMOF	ACT/S	AMOF	ACT/S
6	(93.4,20.4)	R	(93.4,19.9)	R
10	(240.9,34.6)	R	(237.2,33.7)	R
30	(2112.0,29.3)	R	(2018.0,50.8)	R
70	(10148.4,29.1)	2	(9916.0,98.1)	2
100	(20537.4,33.8)	3	(19788.8,156.8)	5
300	(243111.6,34.9)	5	(180486.7,340.0)	14

Table 2-Comparison results of applying PSO (F, 1) and PSO (F, 2) for P-problem for different *n*.

n	PSO(F,1)			PSO(F,2)		
	AMOF	ABT/S	ACT/S	AMOF	ABT/S	ACT/S
6	(93.4,19.8)	R	R	(93.3,19.8)	R	R
10	(236.6,33.8)	R	1	(236.4,34.0)	R	1
30	(2107.5,33.9)	2	3	(2114.9,36.1)	R	2
70	(10934.9,52.2)	5	6	(10922.4,43.6)	R	4
100	(22182.1,75.8)	7	8	(21989.9,62.5)	R	6
300	(215956.0,66.1)	15	20	(201780.8,65.7)	R	17

From Tables -(1 and 2), we notice the good performance of LSM (F, 2) for P-problem. Therefore, it will be used for the next comparison tables. For simplicity we use LSM (ob) instead of LSM (ob, 2). Table -3 shows the comparison results between the LSM methods (SA(F) and PSO(F)) compared with the CEM(F) method to solve P-problem, *n*=4:10.

Table 3-Comparison results of SA (F), PSO (F) with CEM (F) for P-problem, *n*=4:10.

n	CEM(F)		SA(F)			PSO(F)		
	AMOF	AT/S	AMOF	AT/S	AAE	AMOF	AT/S	AAE
4	(58.6,16.3)	R	(58.6,16.3)	R	(0,0)	(58.6,16.3)	R	(0,0)
5	(72.8,24.4)	R	(73.0,24.1)	R	(0.003,0.012)	(72.8,24.4)	R	(0,0)
6	(93.4,19.8)	R	(93.4,19.9)	R	(0,0.005)	(93.3,19.8)	R	(0.001,0)
7	(134.6,24.5)	R	(135.2,25.6)	R	(0.004,0.045)	(134.8,24.3)	R	(0.001,0.008)
8	(143.3,27.5)	R	(145.1,28.7)	R	(0.013,0.043)	(144.1,26.7)	R	(0.006,0.029)
9	(199.3,28.2)	4.4	(205.0,26.9)	R	(0.029,0.046)	(200.8,26.5)	R	(0.008,0.06)
10	(235.8,34.1)	43.1	(237.2,33.7)	R	(0.006,0.012)	(236.4,34.0)	1	(0.003,0.003)
AAE					(0.008,0.023)			(0.003,0.014)

Comparison results between SA(F) and PSO(F) with efficient results of BAB(F) for P-problem are shown in table 4, *n*=11:20.

Table 4- Applying SA (F) and PSO (F) for P-problem compared with BAB for *n*=11:20.

n	BAB(F)		SA(F)			PSO(F)		
	AMOF	AT/S	AMOF	AT/S	AAE	AMOF	AT/S	AAE
11	(286.0,28.2)	R	(281.1,32.3)	R	(0.017,0.145)	(285.6,28.2)	1	(0.001,0)
12	(366.2,18.1)	R	(359.3,26.5)	R	(0.019,0.464)	(365.2,18.9)	1	(0.003,0.044)
13	(441.3,28.9)	1	(436.1,36.7)	R	(0.012,0.27)	(443.5,28.5)	1	(0.005,0.4)
14	(452.0,29.9)	R	(451.5,30.9)	R	(0.001,0.033)	(455.9,29.1)	1	(0.009,0.027)
15	(531.7,33.2)	6.3	(529.0,35.3)	R	(0.005,0.063)	(538.1,29.3)	1	(0.012,0.117)
16	(660.4,24.9)	11.4	(655.0,30.5)	R	(0.008,0.225)	(668.3,22.4)	1	(0.012,0.1)
17	(657.0,37.7)	21.5	(654.3,42.6)	R	(0.004,0.13)	(671.8,33.9)	1	(0.023,0.101)
18	(729.0,36.8)	56.4	(724.9,41.9)	R	(0.006,0.139)	(740.7,31.7)	1	(0.016,0.139)
19	(855.5,30.6)	128.5	(847.5,35.2)	R	(0.009,0.15)	(869.7,30.4)	1	(0.017,0.007)
20	(919.9,31.0)	211.5	(915.3,33.4)	R	(0.005,0.077)	(941.8,27.0)	1	(0.024,0.129)
AAE					(0.009,0.17)			(0.012,0.11)

In table 5, we compare the results obtained from SA(F), PSO(F) and SPT-MST-SCSE(F) for P-problem, *n*=30,70,100,300,700,1000.

Table 5-Results of comparison of SPT-MST-SCSE (F), SA (F) PSO (F) for (P), for different *n*

n	SPT-MST-SCSE(F)		SA(F)		PSO(F)	
	AMOF	AT/S	AMOF	AT/S	AMOF	AT/S
30	(2343.8,56.2)	R	(2018.0,50.8)	R	(2114.9,36.1)	2
70	(11985.8,96.5)	R	(9916.0,98.1)	2	(10922.4,43.6)	4
100	(24316.2,155.0)	R	(19788.8,156.8)	5	(21989.9,62.5)	6
300	(241361.9,123.1)	1	(180486.7,340.0)	14	(201780.8,65.7)	17
700	(1286395.7,98.9)	6.7	(920961.4,314.6)	34	(1050475.5,62.6)	41
1000	(2681881.1,160.8)	16.8	(1934909.8,611.6)	50	(2204431.3,119.2)	60

4.2 Comparison Results of P_1-problem.

Table-6 shows the comparison results of applying SA (F_1, 1) with SA (F_1, 2) and PSO (F_1, 1) with PSO (F_1, 2) for P_1-problem for a chosen *n*.

Table 6- Comparison results of applying SA (F_1, 1) and SA (F_1, 2) as well as PSO (F_1, 1) and PSO (F_1, 2) for P_1-problem for different *n*.

n	SA(F_1,1)		SA(F_1,2)		PSO(F_1,1)			PSO(F_1,2)		
	ASOF	ACT/S	ASOF	ACT/S	ASOF	ACT/S	ABT/S	ASOF	ACT/S	ABT/S
6	111.2	R	110.4	R	110.4	1	R	110.4	1	R
10	269.6	R	268.6	R	267.4	1	R	267.4	1	R
30	2098.8	R	2063.6	R	2073.4	6	5	2081.2	2	R
70	10050.6	R	10000.4	R	10369.6	13	12	10038.6	12	R
100	20208	R	19912.6	R	21121.4	29	28	19995.4	27	R
300	242042.4	2	180727.2	2	206199	82	79	180842.8	79	R

For simplicity, we use LSM (F_1) instead of LSM (F, 2), which is used to compare with other methods.

In Figure-1, the comparison results of SA (F1, 1), SA (F1, 2), PSO (F1, 1), and PSO (F1, 2) for n=10:10:100 are calculated.

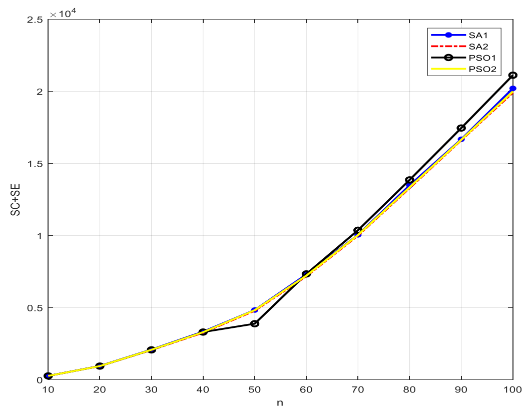

Figure 1-Comparison results of SA (F_1, 1), SA (F_1, 2), PSO (F_1, 1), and PSO (F_1, 2) for *n*=10:10:100.

In Table -7, we show a comparison between the optimal results of CEM(F_1) and the results of the local search methods SA(F_1) and PSO(F_1), *n*=4:10 for P_1-problem.

Table 7-Comparison between CEM (F_1) and SA (F_1) and PSO (F_1) for P_1-problem, *n*=4:10.

n	CEM(F_1)		SA(F_1)		PSO(F_1)	
	ASOF	AT/S	ASOF	AT/S	ASOF	AT/S
4	73	R	73	R	73	R
5	95.6	R	95.8	R	95.6	1
6	110.4	R	110.4	R	110.4	1
7	156	R	156.4	R	156	1
8	166.8	R	169	R	166.8	1
9	224.2	4.5	225.2	R	224.2	1
10	267.4	46.2	268.6	R	267.4	1

Table-8 shows the results of BAB(F_1) or P_1-problem compared with results of SA(F_1) and PSO(F_1) methods for P_1-problem, *n*=11:15.

Table 8- Comparison results of SA (F_1), PSO (F_1) with BAB (F_1) for (P_1), *n*=11:15.

n	BAB(F_1)		SA(F_1)		PSA(F_1)	
	ASOF	AT/S	ASOF	AT/S	ASOF	AT/S
11	309.4	R	312	R	309.4	2
12	378.8	1.2	381.4	R	378.8	3
13	465.4	6.6	467.6	R	465.6	3
14	476.6	45.4	481.2	R	477	3
15	559	66.2	562.8	R	559.4	3

Table-9 describes the efficient solution for P_1-problem for *n*=30:70,100, 300, 700, 1000, using DR-SCSE(F_1) compared with SA(F_1) and PSO(F_1) methods.

Table 9-Comparison results of DR-SCSE (F_1) with SA (F1) and PSO (F_1) for (P_1), for different *n*.

n	DR-SCSE(F_1)		SA(F_1)		PSA(F_1)	
	ASOF	AT/S	ASOF	AT/S	ASOF	AT/S
30	2083.6	R	2063.6	R	2081.2	6
70	10038.6	R	10000.4	R	10038.6	12
100	19995.4	R	19912.6	R	19995.4	27
300	180842.2	R	180727.2	2	180842.8	79
700	921031.6	R	921031.6	3	921031.6	172
1000	1934981.8	2.3	1934981.8	5	1934981.8	248

6. Conclusions

1. In this paper, we demonstrated the good performance of the two suggested LSM (SA and PSO) in solving multicriteria MSP where their results are compared with CEM, BAB and good heuristic methods.
2. We noticed the effects of the starting solution to obtain optimal solutions for SA and PSO for different n (according to tables 1, 2, and 6).
3. For P-problem, $n \leq 10$, the performance of PSO is better than SA in accuracy (according to table 3), while SA is better than PSO for $n > 10$ in both accuracy and CPU time (according to table 4).
4. For P_1-problem, $n \leq 15$, the performance of PSO is better than SA in accuracy (according to tables 7 and 8).
5. For P_1-problem, as a comparison of heuristic methods, we notice that SA is the best method among the DR-SCSE and PSO, in terms of accuracy, while in terms of CPU-time, the DR-SCSE and SA are better than PSO.
6. To develop the performance of LSM, we suggest a hybrid between SA and PSO for the P and P_1-problems.

References

1. Abbas, I. T. **2009**. "The performance of multicriteria scheduling in one machine", M.Sc. thesis Univ. of Al-Mustansiriyah, College of Science, Dep. of Mathematics.

2. Ali F. H. and Abdul-Kareem S. B. **2017**. "Scheduling a Single Machine to Minimize Max Tardiness, Max Late Work and Total Late Work", *Mathematics and Statistics Journal*, **3**(1): 1-17.

3. Aarts E. H. L. and Lenstra J. K. **1997**. (eds.), "*Local Search in Combinatorial Optimization*", John Wiley and Sons, Chichest.

4. Tarasewich P. and McMullen P. R. **2002**. "Swarm Intelligence: Power in Numbers", *Communications of the ACM* , **45**: 62-67.

5. Abdul-Razaq T. S., Chachan H. A. and Ali F. H. **2012**. "Modified Heuristics For Scheduling in Flow Shop to Minimize Makespan", *Journal of Al-Rafidain University College for Sciences*, No#.30.

6. Ali, F. H. **2015**. "Improving Exact and Local Search Algorithms for Solving Some Combinatorial Optimization Problems", Ph. D. Thesis, University of Al-Mustansiriyah, College of Science, Dept. of Mathematics, (2015).

7. Wang, X.J., Zhang, C.Y., Gao, L., Li, P.G. **2008**. " A survey and future trend of study on multiobjective scheduling", *International Conference on Natural Computation*, **6**: 382-391 (2008).

8. David, M.A. **2007**. "A multi-objective genetic algorithm to solve single machine scheduling problems using a fuzzy fitness function", M.Sc. thesis College of Engineering and Technology of Ohio University of Mathematics.

9. Salman F. M. **2008**. "Exact and Local Search Methods for Single Machine Problems", M. Sc. thesis Al-Mustansiriyah University, College of Science, Department of Mathematics.

10. Kennedy J. and Eberhart R. C. **1995**. "Particle Swarm Optimization", Proceedings of IEEE International Conference on NN, Piscataway, pp. 1942-1948.

11. Ribeiro P. F. and Kyle W. S. **2003**. **"**A Hybrid Particle Swarm and Neural Network Approach for Reactive Power Contro***l*"**,** Member, http://engr.calvin.edu/…/Reactivepower-PSO-wks.pdf.

12. Aseel A. and Faez H. **2020**. "Using Heuristic and Branch and Bound Methods to Solve Multi-Criteria Machine Scheduling Problem", *Iraqi Journal of Science*, (Accepted in, Nov. 2019 and will published in Volume (61), Issue (8).