Journal of Kerbala University , Vol. 13 No.4 Scientific . 2015

Scheduling jobs with families setups on identical parallel machines to minimize makespan function

جدولة الاعمال مع عوائل الاعداد على مكائن متوازية متماثلة لتصغير دالة *makespan*

Hussam Abid Ali Mohammed
Department of Mathematics, College of Education, University of Kerbala, Kerbala, Iraq, (hussammath@yahoo.com).

Mohammed Hassan Saloomi
Department of Mathematics, College of Education, University of Kerbala, Kerbala, Iraq, (mohammedsalomi@yahoo.com).

Abstract:

This paper considers the problem of scheduling n independent jobs on m identical parallel machines with family setup times. The preemption of jobs is forbidden. The aim is to minimize makespan. We develop compare and test different local search methods such as Memetic algorithm approach (MA), Threshold acceptance algorithm (TH) and Tabu search (TS). Computational experience is found that these local search algorithms solve problem to 5000 jobs with reasonable time.

Key words: Machine scheduling, parallel machines, Meta-heuristic, Tabu search, Memetic algorithm, Threshold acceptance algorithm, family setup times.

المستخلص:

تناولنا في البحث مسألة جدولة n من الأعمال المستقلة على m من المكائن المتوازية المتماثلة بوجود عوائل من وقت الأعداد. والأسبقية بين الأعمال غير موجودة. الهدف من البحث هو تقليل قيمة دالة الهدف وهي القيمة العظمى من وقت التمام. وقد أظهرنا مقارنة واختبار بين طرق بحث محلية مختلفة مثل (MA), (TH) و(TS) . طرائق البحث المحلي استخدمت لتصغير الزمن المستخدم لإيجاد الحل يصل إلى 5000 عمل في زمن معقول.

1. Introduction

The parallel machines scheduling problem is widely studied optimization problem. This problem consider several available identical machines to execute a set of jobs $N = \{1, \dots n\}$ is consider, where the jobs are divided into F families. Each family $f, f = 1, \dots F$, contains n_f jobs are numbered $1, \dots, n$, sometimes it is more convenient to refer to job(i, f)which is the ith job in family f,for $1 \leq i \leq n_f$.

Indeed it can be described as a special hybrid flowshop scheduling problem which has only one stage. Every job j is considered with a processing time p_{if} and sequencing independed setup times S_f, the objective is to find optimal sequence which give minimize makespan function the studied problem is defined $P_m|S_f|C_{max}$.

The most studied criteria for scheduling problems is the makespan. It is the completion time of the job which is finished at last (maximum completion time of jobs). Cheng and Sin [4] have proved that the problem of minimizing the makespan on two identical parallel machines is NP-hard. Brucker [2] has proved that if the number of machines is greater than two, then the problem is even strongly NP-hard.

In the literature many methods are proposed to solve it Gendreau [6] have proposed heuristic and a lower bound for the $P_m|S_{ij}|C_{max}$ problem. Neronet al. [10] have used two branching schemes for the parallel machines scheduling problem with release dates and tails. Zouba et al. [12] present heuristics algorithms to solve a parallel machines scheduling problem with a period based changing mode operators to minimize the maksepan, Fanjul-peyro and Ruiz [5] have proposed the size reduction heuristics for the unrelated parallel machines scheduling problem with the minimization of makespan.

In section (2) details of the given problem. Section (3) presents a description the local search method, computational results obtained by the proposed local search methods in section (4).

2. Problem Description and Mathematical Formulation

In the parallel machines scheduling problem, a set of n independent jobs should be scheduled on m identical machines without preemption. Each job has a processing time p_{if}. Suppose the processing order $\sigma = (\sigma(1), \ldots, \sigma(n_k))$, $k = 1, \ldots, m$. A vector $S = (S_{\sigma(1)}, \ldots, S_{\sigma(n_k)})$, of corresponding setup times is easily constructed:-

The setup time required immediately before the processing of job $\delta(i), (i = 1, \ldots, n)$ is given by:

$S_{\delta(1)}$: is the setup time of the first job (positive integer constant)

$$S_{\delta(i)} = \begin{Bmatrix} \alpha_{fg} & i > 1, \sigma(i-1) \in f \ and \ \sigma(i) \in g, f \neq g, f, g \in F \\ o & o.w \end{Bmatrix}$$

Where α_{fg} is a positive integer constant.

All these jobs data are generated randomly. Some assumption must be respected: each machine can execute only one job at once, each job can be processed only once, some notation are notations are defined below:

n: the number of jobs.

m: the number of machines.

j: the index of jobs $j = 1,2, \ldots, n$.

k: the index of machines $k = 1,2, \ldots, m$.

r: the order of job in the machine.

p_{jf}: the processing time of job j, $j = 1,2, \ldots, n$, $f = 1, \ldots F$.

C_{jf}: the completion time of job j, $j = 1,2, \ldots, n$, $f = 1, \ldots F$.

S_{fg}: the sequence indepened setup times if family g is the immediate successor of the family f on the same machine.

n_{kf}: the number of jobs assigned to machine k on same family f.

The problem can be formulated as follows:

Minimize C_{max} ... (1)

Subject to

$$\textstyle\sum_{j=1}^{n} x_{jkr} = 1, \ \ k = 1, 2, \ldots, m, \ \ r = 1, 2, \ldots, n_{kf} \quad \ldots (2)$$

$$\textstyle\sum_{k=1}^{m} \sum_{r=1}^{n_{kf}} x_{jkr} = 1\ , \ \ j = 1, 2, \ldots, n \quad \ldots (3)$$

$$p_{[kr]} = \textstyle\sum_{j=1}^{n} x_{jkr}\, p_j, \ \ k = 1, 2, \ldots, m, \ \ r = 1, 2, \ldots, n_{kf} \quad \ldots (4)$$

$$C_{[kr]} = C_{[kr-1]} + p_{[kr]}, \ \ k > 1, k = k-1, \ \ r = 1, 2, \ldots, n_{kf} \quad \ldots (5)$$

$$C_{[kr]} = C_{[kr-1]} + p_{[kr]} + S_f, \ \ k > 1, k \neq k-1, \ \ r = 1, 2, \ldots, n_{kf} \quad \ldots (6)$$

$$C_{max} = max_{k=1}^{m} max_{r=1}^{n_{kf}} C_{[kr]} \quad \ldots (7)$$

$$x_{jkr} = 0 \ \ or \ \ 1, \ \ k = 1, 2, \ldots, m, \ \ r = 1, 2, \ldots, n_{kf} \ \ \ j = 1, 2, \ldots, n_{kf} \quad \ldots (8)$$

$$y_{ij} = 0 \ \ or \ \ 1, \ \ i = 1, 2, \ldots, n, \ \ j = 1, 2, \ldots, m \quad \ldots (9)$$

Equation (1) represents the objective function and the goal of our work is to minimize the maximum completion time. Constraint (2) ensures that only one job can be scheduled at the rth job position. Constraint (3) means that each job can be scheduled only once. Constraints (4)-(6) denote the data of jobs which are scheduled at the rth job position of kth machine position jobs, such as processing times and completion time. Constrain (7) explain how to compute the makespan. Constraint (8) is a decision variable, if job j is scheduled on machine i in position r, then $x_{jkr} = 1$, otherwise 0. Constraint (9) shows that if job j is the immediate successor of the job i on the some machine, then $y_{ij} = 1$, otherwise, y_{ij} =0.

3. Local search heuristics

Research a local search in scheduling is quite extensive, but application to parallel machine scheduling are scarce. There are few computational studies that compare different local search methods on the same scheduling problem [1]. Three local search algorithms are implemented.

3.1 Neighborhood generating Mechanisms

We develop a local search method here where five operations are used to generate local search neighborhoods these operations are the

- **Move operation:**

Reassigning one job from a machine with minimum makespan to another machine.

- **Swap operation:**

Swap one job from a machine with minimum makespan with one job from another machine.

- **Insert $[\boldsymbol{i}, \boldsymbol{j}]$ operation:**

Represent a move where job i is remove from machine $m(i)$ such that [let $m(i)$ denote the machines that job i is currently processed on] and inserted right before j. In machine $m(j)$, where j is the first job on $m(j)$ with the property $i < j$.

- **Insert $[\boldsymbol{j}, \boldsymbol{p}(\boldsymbol{l})]$ operation:**

Denote the move where job j is scheduled to be processed at the end of machine l.

- **$\boldsymbol{k}$-insert operation:**

We construct a restricted version of the k-insert neighborhood by only allowing moves insert$[i_1, j_1]$, inserts$[i_2, j_2]$, …, insert$[i_k, j_k]$, where$i_l < i_{l+1}$ for $l = 1$ to $k - 1$ with $j_l < j_{l+1}$, for $l = 1$ to $k - 1$.
Now we introduce algorithm (1) which is applied initial solution to use in local search method.

Algorithm (1):

Swaps as kick moves for parallel machines scheduling.
Procedure kick move (s)
For M time
Randomly select two machines k and $l \ni k \neq l$.
Randomly select two jobs $m_k(i)$ and $m_l(j)$
Apply swap$[m_k(i), m_l(j)]$
End

Where M dependent on the number of machines m. In our experiments, we discovered that choosing M randomly from interval (0.3m, 0.8m).

Initial solution (Ini)

When we start with the initial solution before using local search means we start with good solution may be gives the optimal solution.

Initial solution:

$J_i^k(l)$ denotes job j_i which is placed in the lth position on machine k. The initial solution is generated as follows:

<u>Step(1):</u>

Arrange all jobs by SST (shorted setup times)

And obtain a sequence $\{J_i(f), f = 1, 2, \dots, n\}$, $J_i(f)$ means that job J_i is placed in the fth position on the SST sequence.

Step(2):
$k \leftarrow 1, f \leftarrow 1, l \leftarrow 1$
Step(3):
$J_i^k(l) \leftarrow J_i(f)$
Step(4):
$k \leftarrow k + 1, f \leftarrow f + 1$
Step(5):
If $k > m$, then $k \leftarrow 1$, and $l \leftarrow l + 1$ if $f > n$ stop otherwise go to step(3).

l^k denote the number of jobs assigned to machine k.
After algorithm (1) the jobs assigned to each machine are ordered by NEH algorithm [11].

3.2 Threshold acceptance method (TH)

A variant of simulated annealing is the **threshold acceptance method** (Brucker 2007). It differs from simulated annealing only by the acceptance rule for the randomly generated solution $s' \in N$. s' is accepted if the difference $F(s') - F(s)$ is smaller than some non-negative threshold t. t is a positive control parameter which is gradually reduced. Figure (1) shows the generic implementation of threshold acceptance structure.

```
While (termination condition in not satisfied)
    New solution ←neighbors(best solution);
    If new solution is better than actual solution
        Best solution ← actual solution
    Else difference between old and new solution less than control
        parameter t then
        Best solution ← actual solution
    End
End
```

Figure 1: Threshold acceptance structure

The threshold acceptance method has the advantage that they can leave a local minimum. They have the disadvantage that it is possible to get back to solutions already visited. Therefore oscillation around local minima is possible and this may lead to a situation where much computational time is spent on a small part of the solution set.

3.3 Tabu search

The use of the tabu search was pioneered by Glover [7] who from 1985 onwards has published many articles discussing its numerous applications. Others were quick to adopt the technique which has been used for such purposes as sequencing, scheduling, oil exploration and routing.

The properties of the tabu search can be used to enhance other procedure by preventing them becoming stuck in the regions of local minima. The tabu search utilizes memory to prevent the search from returning to a previously explored region of the solution space too quickly. This is achieved by retaining a list of possible solutions that have been previously encountered. These solutions are considered tabu-hence the name of the technique. The size of the tabu list is one of the parameters of the tabu search.

The tabu search also contains mechanism for controlling the search. The tabu list ensures that some solution will be unacceptable; however, the restriction provided by the tabu list may become too limiting in some cases causing the algorithm to become trapped at a locally optimum solution.

The tabu search introduces the notion of aspiration criteria in order to overcome this problem. The aspiration criteria over-ride the tabu restrictions making it possible to broaden the search for the global optimum.

An initial solution is generated (usually randomly). The tabu list is initialized with the initial solution. A number of iterations are performed which attempt to update the current solution with a better one, subject to the restriction of the tabu list. A list of candidate solution is proposed in every iteration. The most admissible solution is selected from the candidate list. The current solution is updated with the most admissible one and the new current solutions added to the tabu list. The algorithm stops after a fixed number of iterations or when a better solution has been found for a number of iterations. Figure (2) shows the generic implementation of tabu search.

S = Generate Initial Solution()
Initialize Tabu List (TL_1, …, TL_r)
K = 0
While (termination condition in not satisfied)
    Allowed Set (S,K) = $\{z \in N(s) |$ no tabu condition is violated or at least one Aspiration criterion is satisfied$\}$
    S = Best Improvement (S, Allowed Set(S,K))
    Update Tabu List and Aspiration Condition()
    K = K+1
End

Figure 2: A generic tabu search

3.4 Memetic algorithm

The memetic algorithms [9] can be viewed as a marriage between a population-based global technique and a local search made by each of the individuals. They are a special kind of genetic algorithm with a local hill climbing. Like genetic algorithm, memetic algorithms are a population based approach. They have shown that they are orders of magnitude faster than traditional *genetic algorithm* for some problem domains. In a memetic algorithm the population is initialized at random or using a heuristic. Then, each individual makes local search to improve its fitness. To form a new population for the next generation, higher quality individuals are selected. The selection phase is identical inform *to* that used in the classical tabu search selection phase. Once two parents have been selected, their chromosomes are combined and the classical operators of crossover are applied to generate new individuals. The latter are enhanced using a local search technique. The role of local search in memetic algorithms is to locate the local optimum more efficiently then the tabu search. Figure 3 explains the generic implementation of memetic algorithm.

Encode solution space
Set pop_size, max_gen, gen=0
setcross_rate, mutate_rate;
initialize population
While(gen <gensize)
    Apply generic GA
    Apply local search
End
Apply final local search to best chromosome

Figure 3: The memetic algorithm

3.4.1 Hill climbing local search algorithm

The hill climbing search algorithm is a local search and is shown in figure 4. It is simply a loop that continuously moves in the direction of increasing quality value [9]

```
While (termination condition in not satisfied)
        New solution ←neighbors(best solution);
        If new solution is better than actual solution
                Best solution ← actual solution
        End
End
```

Figure 4: The Hill climbing local search algorithm

4. Computational experience

This section reports the results of computational test to assess the effectiveness heuristics algorithms. These algorithms are coded in Matlab R2009b and runs on a Pentium IV at 2.00 GHz, 2.92 GB computer.

Test problems with (10, 30, 50, 100, 200, 500, 1000, 2000, 5000) jobs and with (2,4,6) families were generated as follows: jobs are distributed uniformly across families so that each family contains $\lceil n/f \rceil$ or $\lfloor n/f \rfloor$ jobs.

The processing time has been observed in the literature (e.g. [3]). The setup times are randomly generated integers from uniform distribution defined on $[1,10]$. Since the size of setup time's relation to processing times may affect problem "hardness", we generated problems with small (S), medium (M) and large (L) setup times. Medium setup times are randomly generated integers from the uniform distribution defined on $[1,10]$. Having generated an instance with small setup times $\left(S_f/2\right)$ and with large setup times $\left(2S_f\right)$ were constructed.

We generate problem for each contribution of n and setup times. Ten test problems created this method of data generation follows the one given in Hariri and Potts [8].

5. Comparative computational results

This section will report the results of our computational test to show the effectiveness for the local search methods (Memetic algorithms (MAs), Threshold acceptance method (TH) and Tabu search (TS)), we present tables of results which shows the importance of each of the methods. In each tables the first column gives the number of jobs the second column gives the number of families. The third column describes the average solution initial solution (Ini) which describe in section 3.1. The fourth, fifth and sixth columns divided in two columns values and times describes the average computation for the local search MA, TH and TS respectively and we started with the initial solution (Ini).

Table (1) Comparative results values and times for local search for $P_3|\,S_f/2\,|C_{max}$ problem with small setup

n	S_f	Ini	MA		TH		TS	
			Val	Tim	Val	Tim	Val	Tim
10	2	7.499762	5.404484	0.137625	5.582381	0.031101	5.835317	0.048251
	4	9.317143	5.997063	0.144752	6.433849	0.03214	7.304286	0.050039
	6	9.687976	6.623968	0.141046	7.16377	0.03233	8.34504	0.048415
20	2	21.02417	13.78873	0.196954	14.98381	0.03321	14.70087	0.04858
	4	22.34544	15.72984	0.204438	16.42635	0.033888	18.93056	0.045176
	6	25.28147	16.44282	0.198034	18.18679	0.03284	18.49008	0.045758
30	2	23.02754	17.65825	0.248491	18.14353	0.035787	18.42774	0.052644
	4	30.26492	20.34135	0.251205	22.11476	0.034839	22.61512	0.052398
	6	32.58698	21.45905	0.259835	23.83841	0.035229	25.30048	0.052246
40	2	24.12476	21.32869	0.303784	21.4048	0.034853	21.51091	0.056554
	4	34.38254	24.8002	0.309595	27.03325	0.035845	27.43643	0.054895
	6	40.65802	27.03302	0.311028	31.2752	0.036063	31.31389	0.05228
50	2	30.59849	27.8519	0.361223	28.10528	0.036259	28.14599	0.05629
	4	43.6825	32.98591	0.362766	34.34377	0.0366	35.35048	0.054643
	6	48.04349	35.48611	0.365547	37.66933	0.037618	42.28845	0.050652
75	2	49.19119	44.57496	0.510873	44.86377	0.039904	47.76786	0.055497
	4	63.85556	50.13254	0.515885	51.5477	0.03928	51.56817	0.063269
	6	73.93671	56.99952	0.526756	58.28087	0.039823	58.95016	0.057035
100	2	71.90147	62.09333	0.661621	62.52282	0.041924	65.19774	0.065473
	4	92.21349	70.13246	0.671069	74.12488	0.043621	78.78976	0.066526
	6	104.5167	76.72183	0.677394	82.39528	0.043756	90.18849	0.067038
150	2	121.1949	101.8943	0.978098	104.8428	0.048139	110.5986	0.06459
	4	137.454	110.7662	1.026299	116.2101	0.04977	123.9929	0.073729
	6	159.1495	121.6353	0.999054	130.6042	0.049925	135.8589	0.080818
200	2	110.9557	107.2286	1.331226	107.4403	0.053336	107.3325	0.094986
	4	180.7709	141.4387	1.364877	148.7728	0.05694	155.8756	0.098767
	6	207.604	158.7535	1.35766	167.5166	0.057029	182.7675	0.084062
500	2	299.1984	289.2731	4.102536	292.0862	0.091931	293.9587	0.177064
	4	355.6124	321.7856	4.203978	336.8394	0.09586	341.4094	0.222831
	6	451.3882	384.4369	4.261365	419.4519	0.096741	422.4602	0.220283
1000	2	629.8014	627.9338	10.94377	629.0494	0.153073	629.0254	0.278649
	4	789.028	741.1156	11.76129	761.3333	0.155723	766.8788	0.531924
	6	927.3292	840.1833	10.86652	885.9419	0.159972	887.0525	0.659305
2000	2	1104.968	1101.448	32.18843	1104.364	0.273248	1103.901	0.721513
	4	1464.015	1453.484	32.73789	1458.094	0.273215	1456.3	0.885495
	6	1715.151	1669.612	34.14038	1692.589	0.292851	1697.551	0.988219
5000	2	*****	*****	*****	2761.648	0.685241	2760.717	1.693387
	4	*****	*****	*****	3051.662	0.700248	3050.984	1.616642
	6	*****	*****	*****	3394.877	0.729221	3393.74	1.498816

For the small setup times, the table (1) show that MA with using the initial solutions gives the best values better than TH and TS but it took more times therefore for the 5000 jobs the TS is better. And in the same table show that the TH using the initial solution gives the best times for all iterations and MA cannot calculate because it took big times.

Table (2) Comparative results values and times for local search for $P_3|S_f|C_{max}$ problem with medium setup

n	S_f	Ini	MA		TH		TS	
			Val	Tim	Val	Tim	Val	Tim
10	2	9.312976	6.500833	0.140159	6.97504	0.032136	6.816389	0.048377
	4	11.86571	6.618532	0.139533	7.591548	0.032334	8.207183	0.043735
	6	13.61206	8.391746	0.139822	8.62881	0.0317	9.372897	0.046484
20	2	25.53734	16.18079	0.199098	17.2752	0.033357	19.19603	0.047425
	4	28.98794	16.74056	0.198205	21.04746	0.033215	24.71548	0.042416
	6	33.57988	19.49171	0.195606	22.52175	0.034326	24.78194	0.050523
30	2	21.31234	19.21806	0.249715	19.51115	0.035206	20.45952	0.047295
	4	35.60813	24.1846	0.248191	28.03091	0.034198	26.97377	0.051636
	6	43.68448	26.42913	0.250804	31.60123	0.035495	32.27373	0.051757
40	2	25.80226	22.82456	0.306116	22.9294	0.035726	22.86226	0.060717
	4	36.1446	28.49698	0.30583	31.42329	0.035231	31.45198	0.05495
	6	47.04369	32.63325	0.30632	36.67675	0.036262	40.95198	0.048367
50	2	31.88976	29.1244	0.363136	29.21563	0.035967	29.81218	0.053345
	4	49.67282	37.45683	0.371073	37.74988	0.036843	39.17702	0.058559
	6	56.98786	40.63591	0.361014	43.75552	0.036768	45.88369	0.060304
75	2	47.75865	44.86302	0.512407	44.93381	0.038884	45.38841	0.067041
	4	67.45806	54.09917	0.520936	54.96782	0.039713	60.55821	0.06136
	6	85.97075	64.30111	0.510258	69.77734	0.039962	72.02988	0.060758
100	2	80.90369	63.09833	0.664757	66.96143	0.042159	68.53694	0.069853
	4	108.6078	78.38365	0.668523	82.02829	0.042572	89.35921	0.066382
	6	122.3565	86.10115	0.662964	90.16536	0.043626	90.79361	0.070896
150	2	127.1857	102.4189	0.99465	107.5727	0.049206	113.9699	0.068432
	4	135.2063	109.5323	0.985794	110.1465	0.049863	120.3557	0.06872
	6	177.4779	130.7313	0.984904	141.2892	0.051517	158.5815	0.070519
200	2	112.0277	108.3364	1.336498	108.5409	0.054925	108.4952	0.098899
	4	213.0858	154.9256	1.332325	174.9677	0.055976	194.2787	0.088204
	6	232.2787	172.8343	1.324519	189.7767	0.056856	198.2992	0.101605
500	2	314.8705	300.4474	4.126899	304.4371	0.090994	312.334	0.202334
	4	405.9867	352.3109	4.125117	370.9329	0.093501	375.2555	0.224531
	6	481.3981	393.3164	4.133696	441.9287	0.095458	461.1713	0.176562
1000	2	586.0598	582.4328	10.88111	583.5817	0.149576	584.7264	0.366864
	4	868.0785	793.9694	10.94862	838.2835	0.153689	832.1931	0.365399
	6	927.9655	806.597	10.95494	883.6323	0.156604	900.4122	0.484229
2000	2	1144.468	1140.877	31.73897	1143.694	0.272571	1143.566	0.699041
	4	1483.102	1457.739	32.2211	1466.779	0.276764	1469.209	0.875864
	6	1645.104	1567.77	31.97903	1609.014	0.272201	1620.97	1.120508
5000	2	*****	*****	*****	2800.193	0.696607	2799.291	1.700274
	4	*****	*****	*****	3168.877	0.695124	3168.305	1.988188
	6	*****	*****	*****	3502.673	0.720072	3501.559	1.551855

For the small setup times, the table (2) show that MA with using the initial solutions gives the best values better than the other local search but it took more times therefore for the 5000 jobs the TS is better. And in the same table show that the TH using the initial solution gives the best times for all iterations and MA cannot calculate because it took big times.

Table (3) Comparative results values and times for local search for $P_3|2S_f\ |C_{max}$ problem with large setup

n	S_f	Ini	MA		TH		TS	
			Val	Tim	Val	Tim	Val	Tim
10	2	13.56298	10.06226	0.139081	10.14698	0.031582	10.33226	0.046005
	4	20.6248	8.795556	0.1395	12.38984	0.032056	11.68794	0.045766
	6	20.7471	12.96893	0.138782	13.16012	0.031903	14.55349	0.043236
20	2	38.33734	19.24909	0.199047	23.54464	0.032108	25.54921	0.049143
	4	45.76813	21.96028	0.19583	28.2556	0.033281	32.9923	0.05098
	6	56.42187	27.56595	0.19722	31.65821	0.034442	38.29988	0.044787
30	2	25.77143	23.58806	0.246823	23.62397	0.035035	23.92206	0.055043
	4	53.04631	31.83087	0.247774	36.94643	0.035657	48.24063	0.044578
	6	69.27488	38.74968	0.249234	45.49599	0.035144	49.38563	0.052763
40	2	29.26409	26.42417	0.30525	26.58341	0.034938	27.09881	0.052779
	4	51.35754	37.19167	0.311496	42.87238	0.036339	45.86119	0.055171
	6	78.78647	41.86841	0.308487	55.84111	0.036776	54.76655	0.057549
50	2	35.45365	32.33262	0.362927	32.60004	0.035765	32.27635	0.058969
	4	70.22194	47.07909	0.3628	50.3629	0.036705	49.83802	0.060908
	6	86.90159	54.03012	0.367237	62.53746	0.037702	65.81381	0.054826
75	2	51.11242	48.26996	0.508181	48.40401	0.039535	48.5706	0.069603
	4	95.74944	67.98131	0.506788	70.7602	0.03839	73.95591	0.049773
	6	128.9807	88.86313	0.512562	96.02389	0.039861	99.72698	0.062642
100	2	106.7069	78.72147	0.664927	82.98817	0.041898	80.08095	0.07379
	4	163.9833	93.73	0.660681	108.7509	0.041547	115.9073	0.075145
	6	188.844	113.7568	0.659717	132.3659	0.043957	131.0292	0.071718
150	2	165.9719	118.3101	0.98601	124.9194	0.048467	132.221	0.064908
	4	179.1923	138.3856	0.981861	149.9994	0.049863	167.1233	0.067764
	6	268.0869	177.5699	0.980935	196.1619	0.049643	221.1852	0.07339
200	2	115.0319	111.3361	1.328275	111.6644	0.055155	111.4904	0.101916
	4	320.5123	213.7615	1.325819	236.035	0.055507	250.7705	0.11028
	6	368.8299	237.7837	1.327307	268.3628	0.057368	284.736	0.119103
500	2	360.8062	333.0737	4.119293	338.101	0.089292	358.5635	0.20464
	4	539.5966	436.7151	4.091199	478.6222	0.092957	490.6478	0.21449
	6	703.0437	531.8573	4.127765	600.3911	0.093481	615.3435	0.190481
1000	2	632.9962	628.9369	10.92503	624.6407	0.150547	631.7449	0.361675
	4	1178.374	1047.863	10.94415	1130.863	0.154648	1169.185	0.35358
	6	1338.947	1087.764	10.96648	1250.037	0.155067	1284.244	0.562533
2000	2	1223.668	1219.527	32.03994	1223.34	0.267202	1222.5	0.766052
	4	1900.236	1837.859	31.73019	1872.795	0.275196	1876.756	0.774348
	6	2217.085	2067.16	31.9036	2154.162	0.278061	2140.359	1.292822
5000	2	*****	*****	*****	2903.108	0.67982	2902.63	1.802701
	4	*****	*****	*****	3640.604	0.676963	3640.184	1.819546
	6	*****	*****	*****	4310.177	0.688945	4307.38	1.736626

For the small setup times, the table (3) show that MA with using the initial solutions gives the best values better than the other local search but it took more times therefore for the 5000 jobs the TS is better. And in the same table show that the TH using the initial solution gives the best times for all iterations and MA cannot calculate because it took big times.

6. Conclusion

In this paper, we have developed a number of solution procedures for the identical parallel machines scheduling problem:

Minimize the maximum completion time C_{max} taking into account sequence with setup times. The local search methods that are used to solve all of the large problems in this paper, the results show the robustness and flexibility of local search heuristics.

Future work Some suggestions for future research are described as follows:

First, the propose of extension the exact for $P_3|S_f|C_{max}$ problem by driving a good lower bound or using the dominance rule in branch and bound algorithm.

Second, using the local search heuristic should be explored finding an improvement potential of various polynomially bounded scheduling heuristic.

References

[1]Ahn, B.H, and Hyun, J.H., (1990), *Single facility multi class job scheduling*, Computers and Operation Research, 17:265-272.

[2]Brucker P. (2007), *Scheduling Algorithms*, Springer Berlin Heidelberg New York, Fifth Edition.

[3]Cheachan H.A., Mohammed H.A. and Khtan Q.A., (2010): *Scheduling flowshop machines to minimize the multi-objective functions*. Iraqi Journal for Administrative Sciences, 637-657.

[4]Cheng T.C.E., and, 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.

[5]Fanjul-peyro L. and, Ruiz R., (2011), *Size reduction heuristics for the unrelated parallel machines scheduling problem*, Computers and Operation Research, 38: (1) 301-309.

[6]Gendreau M., (2003), *An introduction to tabu search*, in Glover F. and Kochenberger G.A. editors handbook of meta-heuristic 37-54, Boston Kluwer academic publishers.

[7]Glover F., (1994) *A user's guide to tabu search*, Annals of Operations Research, 41:3-28.

[8]Hariri A.M.A. and Potts C.N., (1997), *Single Machine Scheduling with Batch Set-up Times to Minimize Maximum Lateness*, Annals of Operations Research, 70:75–92.

[9]Moscato P., (1989), *On evolution, search, optimization, genetic algorithm and martial arts: toward memetic algorithms*, Technical Report, California.

[10] Neron E., Tercinet F., and, Sourd F., (2008), *Search tree based approaches for parallel machine scheduling*.Computers and Operations Research, 35:(4)1127-1137.

[11] Nessah R., Chu, C. and Yalaoui F., (2007), *An exact method for problem denoted* $P_m|sds, r_j|\sum C_j$ *problem*, Computers and Operations Research, 34: 2840-2848.

[12] Zouba M., Baptiste P., and, Rebaine D., (2009), *Scheduling identical parallel machines and operations within a period based changing mode*, Computers and Operations Research, 36:(12)3231-3239.