Iraqi Journal of Statistical Sciences (24) 2013
PP[45-58]

An Improved particle swarm algorithm to find optimal scheduling in two-stage hybrid flow shop problem

MANAL A . ZEIDAN*

Abstract

This paper deals with the two-stage hybrid flow shop problem , in which the first stage consists of three machines , the second stage consists of two machines . The aim is to find out the optimal scheduling for n jobs when processing in this environment when the makespan is minimum. Therefore we propose a particle swarm algorithm which consists of a new procedure to calculate the makespan and a new stopping criteria .Also, we added improvement to the proposed algorithm , by using one of the components of the genetic algorithm (crossover operation) in order to obtain initial swarm particles instead of random obtaining. After applying the two algorithms on several problems which were generated randomly by uniform distribution , the results showed that the improved proposed particle swarm algorithm was the best in finding out the optimal scheduling for jobs and in cpu time to reach the solution.

خوارزمية محسنة لأسراب الطيور لإيجاد أمثل جدولة في مسألة الورشة الانسيابية ذات المرحلتين

الملخص

تناول هذا البحث مسألة الورشة الانسيابية الهجينة ذات المرحلتين ، والتي تتكون المرحلة الاولى فيها من ثلاث ماكنات و المرحلة الثانية من ماكنتين. الهدف هو ايجاد افضل جدولة ل n من الاعمال عند معالجتها في هذه البيئة بحيث يكون وقت التنفيذ اقل ما يمكن، لذلك اقترحت خوارزمية لأسراب الطيور تضمنت اسلوبا جديدا لحساب وقت التنفيذ واقتراح معيار توقف جديد، كذلك اضيفت تحسين الى الخوارزمية المقترحة ، اذ استخدمت أحد مكونات الخوارزمية الجينية وهي عملية التداخل من أجل تكوين افراد السرب الابتدائي بدلاً من تكوينها بشكل عشوائي.بعد تطبيق الخوارزميتين على عدة مسائل التي تم ولدت عشوائياً من التوزيع المنتظم وقد أظهرت النتائج ان خوارزمية اسراب الطيور المقترحة المحسنة كانت الأفضل في ايجاد امثل جدولة للأعمال وفي الوقت المستغرق للوصول الى الحل.

* Assit. Lecturer / Dept. operational research & intelligent techniques / *Computer Sciences and Mathematics College / Mosul* University /Email:manal_8m@yahoo.com.

Received Date 18/3/2013——————Accept Date 23/5/2013

1- Introduction

The scheduling of n jobs through an S stage flow shop where at any stage ,there exists one or more identical processors , has been termed a Flow Shop with Multiple Processors (FSMP) or Hybrid Flow Shop (HFS) .

The HFS has recently been the focus of numerous research efforts aimed at developing efficient heuristics to solve the minimum makespan problem, this is primarily due to the fact that,even with only one machine at each stage,the problem is NP-complete[15].

Because of the HFS problem is difficult even in case of the number of stages equal to two as proved by Gupta at 1988[5] "The two-stage hybrid flow shop problem is NP-complete even if the number of machines at one of the two stages is one",we have studied a special case of a two-stage hybrid flow shop in which stage1 consists of three machines and the stage2 consists of two machines.

And because the heuristic algorithms are commonly used for obtaining approximate solutions and decreasing cpu time in NP-hard combinatorial optimization problem[7], we proposed a particle swarm algorithm to this special case, which consists of a new procedure to calculate the makespan and a new stopping criteria depend on the percentage difference of the solution from the lower bound of the makespan.Also,we add improvement to the proposed particle swarm algorithm,this improvement has been add to initial swarm by entering one of the componented of the genetic algorithm (crossover operation) in order to obtain initial swarm particles instead of random obtaining in order to improve the performance of the algorithm and reducing the cpu time to reach the solution.

This paper is organized as follows,In section 2, the two-stage hybrid flow shop scheduling problems will first be described.In section 3, the background of particle swarm optimization will be described and in section 4, we applied it to the two-stage hybrid flow shop problems.In section 5,we compare the performance of the proposed particle swarm algorithm and improved proposed particle swarm algorithm . Finally,in section 6 conclusions are discussed.

1.1- Previous works

Many different approaches have been proposed to solve the HFS problem,such as exact solution,heuristic and metaheuristic . One of the exact solution methods mostly used for the HFS problem is the branch and bound approach. Haouari et al [6] present an exact branch-and-bound algorithm for the two-stage hybrid flow shop problem with multiple identical machines in each stage. In recent years,metaheuristics have become a popular approach to solve the HFS scheduling problem. Alayky'ram et al [1] present an improved Ant System (AS) algorithm which uses the same formula as classical AS , but with different starting solution procedure. Qiao and Sun [14] proposed a method based on improved immune particle swarm optimization algorithm to solve the HFS scheduling problem . Yue-wen et al [19] present a Multi-agent Particle Swarm Optimization (MPSO) based on multi-agent system (MAS) and pso for hybrid flow shop scheduling problem.Liao et al [9] present an approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem.Wang and Liu [12] proposed a heuristic method for two-stage hybrid flow shop with dedicated machines,they consider the first stage contains a single common critical machine and the second stage contains several dedicated machines.Wang and Liu [17] present a genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem.

2- Two-stage hybrid flow shop scheduling problem

The two-stage hybrid flow shop scheduling problem may be formulated as follows :A set J of n jobs has to be scheduled in a manufacturing system with two stages (machining centers) S_1 and S_2.Each stage S_i (i=1,2) has m_i identical machines in parallel. Each job j (j=1,2,….,n) has to be processed first for a_j units of time by one machine of S_1 , and then for b_j units of time by one machine of S_2 . These operations must be processed without preemption .Moreover , a job cannot be processed by more than one machine at the same time and each machine processes at most one job at one time . All processing times are assumed to be deterministic and integer and all machines are ready from time zero onwards[6].

In this paper, a special case of two-stage hybrid flow shop has been studied , in which stage1 consists of three machines and stage2 consists of two machines.

In order to clarify this case , we will take the following simple example as shown in Figure (1).

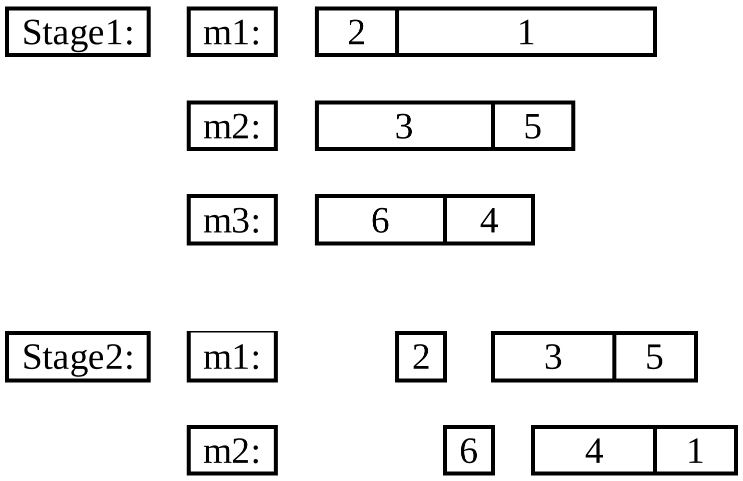

Figure (1) a simple example for the special case of two-stage hybrid flow shop problem

As depicted in Fig.(1) , the solution (2,3,6,1,4,5) gives the schedule for the three machines in stage1 , where jobs are arranged to in machines by the sequence to the first available machine. The schedule in stage2 is arranged as soon as jobs are completed by the preceding stage.

3- The background of particle swarm optimization

Particle Swarm Optimization (PSO) is a population-based swarm intelligence algorithm that was first presented by Kennedy, J. and Eberhart, R. in 1995 [8] .The PSO concept is based on observations of the social behavior of animals.

Mathematically, assume that the search space is D – dimensional and there are m particles in the swarm. Each particle is located at position $x_i=\{x_{i1},x_{i2},\ldots\ldots,x_{id}\}$ with velocity $v_i=\{v_{i1},v_{i2},\ldots\ldots,v_{id}\}$, where i=1,2,…..,m . In the pso algorithm , each particle moves toward its own best position (pbest) denoted as $pbest_i=\{pbest_{i1},pbest_{i2},\ldots,\ pbest_{id}\}$.The best position of whole swarm (gbest)is denoted as $Gbest=\{gbest_1,gbest_2,\ldots\ldots,gbest_d\}$ with each iteration [9].

The basic procedure for implementing PSO is described as follows:- [7][9]

<u>Step1:</u> generate the initial swarm of particles with random positions and velocities on D dimensions in the search space.

<u>Step2:</u> evaluate its fitness function.

<u>Step3:</u> compare particle's fitness values with their pbest , if the current particle is better than pbest ,then set pbest to the current particle.

<u>Step4:</u> update particle velocities according to the following equation:

$$v_{id}^t = w * v_{id}^{t-1} + c_1 * rand_1 * \left(pbest_{id}^{t-1} - x_{id}^{t-1}\right) + c_2 * rand_2 * \left(gbest_d^{t-1} - x_{id}^{t-1}\right) \qquad (1)$$

Where t is the iteration number,w is inertia weight which used to balance between global and local searches,c_1 is the cognition learning factor,c_2 is the social learning factor, $rand_1$ and $rand_2$ are random numbers uniformly distributed between 0 and 1.

<u>Step5:</u> particles are changed to their new positions according to the following equation:

$$x_{id}^t = x_{id}^{t-1} + v_{id}^t \qquad (2)$$

<u>Step6:</u> stop the algorithm if the stopping criterion is satisfied ; else,return to step2 .

4- A proposed PSO algorithm for the two-stage hybrid flow shop problem

When applying PSO to a scheduling problem there is an obvious practical difficulty.We need a different string representation,an approach for converting the continuous postion values in to permutation of the job sequence and other component of the PSO algorithm,including swarm size,fitness function and stopping criterion.These are shown below in detail.

Solution representation

A solution is simply represented by a string of numbers consisting of a permutation of n jobs denoted by (1,2,...,n). To decode the solution for a specific problem, the jobs are arranged to in machine by priority rules to the first available machine [9].To illustrate the decoding , see the example in paragraph (2). As depicted in Fig.(1) ,the string (2,3,6,1,4,5) represents the decoding of the scheduling.

Initial swarm and swarm size

Randomly arrange the jobs to each particle. These particles will generate the initial population. The swarm population size ranging from 20 to 50 are the most common one for lower dimensional problems , and for higher dimensional problems it is better to select swarm size equal to 50 [11] , therefore we use swarm population size equal to 50.At the initialization of algorithm we set the velocities of the particles equal to zero.

Fitness function

We use the makespan criteria as the fitness function. In this paper ,we proposed a new approach to calculate the makespan for n jobs when processed at two-stage hybrid flow shop in which the stage1 consists of three machine and stage2 consists of two machines, this approach depends on the processing of all jobs in stage1 one by one within the available machine firstly, then we will obtain the completion time of each job processing within stage1.Then we start of jobs processing within stage2 for the jobs which finished recently from stage1 within the available machine too. We will obtain the time for each machine in stage2, then we will choose the longest time as a makespan.

So, we programmazied " Function " by using matlab, where the Function's inputs are the number of jobs and matrix of processing time for jobs at the two-stage the steps of this function are as follows:-

```
1- Define the following variables:
   $m1 = p_{11}$ , $m2 = p_{12}$ , $m3 = p_{13}$
   FT(1) = m1 , FT(2) = m2 , FT(3) = m3
2- For i= 4 to n
   2.1- If (m1<= m2) and (m1<= m3)
            Then $m1 = m1 + p_{1i}$
                 FT(i) = m1
        else if (m2<= m1) and (m2<= m3)
            Then $m2 = m2 + p_{1i}$
                 FT(i) = m2
        else
```

```
              $m3 = m3 + p_{1i}$
              $FT(i) = m3$
          endif
   2.2 endfor
3- Define the variable s to represent vector consisting completion times of processing
   each job in stage1 increasingly.
4- Define the variable I to represent vector consisting the index of completing times of
   processing each job in stage1 FT(i) in the vector FT.
5- Define the following variables:
   $mm1 = s(1) + p_{2I(1)}$
   $mm2 = s(2) + p_{2I(2)}$
6- FOR j = 3 to n
   6.1 If $s(j) = mm1$ or ($s(j) = mm1$ and $s(j) = mm2$)
           Then $mm1 = mm1 + p_{2I(j)}$
       elseIf $s(j) = mm2$
           Then $mm2 = mm2 + p_{2I(j)}$
       elseIf $s(j) > mm1$ and $s(j) < mm2$
          Then $mm1 = mm1 + p_{2I(j)} + (s(j) - mm1)$
       elseIf $s(j) > mm2$ and $s(j) < mm1$
          Then $mm2 = mm2 + p_{2I(j)} + (s(j) - mm2)$
       else
          $w = \min\{mm1, mm2\}$
          If $w = mm1$
             Then $mm1 = mm1 + p_{2I(j)}$
          else
             $mm2 = mm2 + p_{2I(j)}$
          endif
       endif
   6.2 endfor
7- makespan = $\max\{mm1, mm2\}$
```

SPV rules

We use a heuristic approach called Smallest Position Value (SPV)[2] for converting the continuous position values to in permutations of the job sequence .This is the key to enable the continuous pso algorithm be applied to sequencing issues. The SPV rule is effectively used for the flow shop sequence problem(PFSP) [7] , following the successful applications of the above , we use this rule. Table (1) illustrates the process of decoding a particle using the SPV rule.

Table (1) example for decoding a particle using the SPV rule

position	position value	ranking of value	Job order
1	-0.41	-2.90	3
2	4.14	-1.54	5
3	-2.90	-0.41	1
4	3.61	0.25	6
5	-1.54	3.61	4
6	0.25	4.14	2

Proposed stopping criteria

In this paper ,we proposed a new stopping criteria based on the lower bound of the makespan , because of the lower bound consider as "effective tool for estimating the optimal makespan and for evaluating the quality of sub-optimal heuristics"[15].Because of the optimal makespan unknown , but we know that $0 < LB \leq C^*_{\max}$,where $C^*_{\max}$ is the optimal makespan [18] , so the proposed stopping criteria tries to obtain optimal makespan $(C^*_{\max} = LB)$ .Otherwise, it will try to obtain the best makespan when it's deviation about LB is minimum ,this criteria is :

$$\left(\frac{f_{gbest} - LB}{LB}\right) * 100 \leq g$$

Where f_{gbest} is the fitness value for gbest , LB is the lower bound of the makespan and we depended on the LB presented by Santos,D. et al [16] because of this LB is strong .

The aim is to find out the optimal makespan $C^*_{\max}$ by tring to get $\left(\frac{f_{gbest}- LB}{LB}\right) * 100$ equal to zero (g=0) ,if we get it before to completing of 50 repetition ,pso will stop in this case $C_{\max}$ will be optimal $C^*_{\max}$ because of $(C_{\max} = LB)$.

Otherwise, we try out to get $\left(\frac{f_{gbest}- LB}{LB}\right) * 100$ less than or equal to 0.5 (g=0.5) to find the makespan $C_{\max}$ where it's difference about LB is minimum. In case of 50 repetition is completed and we couldn't get $\left(\frac{f_{gbest}- LB}{LB}\right) * 100 \leq 0.5$, we increase g by 0.5 in order to get $\left(\frac{f_{gbest}- LB}{LB}\right) * 100 \leq 1$ and continue with the same steps until g reaching 5 , where 5 is the allowed percentage difference of the solution from the lower bound of the makespan.

Proposed PSO algorithm

Now,we are at the position to present the steps of our proposed PSO algorithm for two-stage hybrid flow shop problem.

Step1:(Initialization)
Determine the initial swarm population as described in an earlier section.The size of the swarm population is 50.

Step2:(Fitness)
Evaluate the fitness of each particle solution in the population using the proposed approach to calculate the makespan.

Step3: (Find)
Find pbest and gbest from the solutions.

Step4: (Update)
Update the velocity of each particle according to eq.(1) , and update the position of each particle according to eq.(2).

Step5:(SPV rules)
Apply the SPV rules for converting the continuous position values to in permutation of job sequence.

Step6: (Fitness)
Evaluate the fitness of the new particle position.

Step7: (Find)
Compare particle's fitness values with their pbest , if the current value is better than pbest ,replace the pbest value ,then find gbest (particle which has better fitness value in the swarm).

Step8:(Termination)
Stop the algorithm if the proposed stopping criterion is satisfied. else, return to step4.

Improvement of proposed pso algorithm

We add improvement on the proposed algorithm,by adding it to the initial swarm. Instead of the jobs arrangement in each particle randomly,we arrange the jobs in the first particle according to the processing time in stage1 increasingly and arrange the jobs in the second particle according to the processing time in stage1 decreasingly,then we generated 24 particles by job arrangement in each particles randomly , then we entered one of the genetic algorithm component (crossover) in order to obtain variable particles.We made crossover operation between the 24 particles in order to produce another 24 particles by using linear order crossover (LOX),which was presented by Flalkenauer,E. and Bouffouix,S.[4] because (LOX) has the best performance between other crossover operations. Steps of (LOX)[3] are :-

1- Choose an interval by selecting two random cut points at each parent structure .In our example cut points are at the third and fifth position of the parent structures.
2- Determine the elements of the selecting interval in each structure. Selected interval in parent1 is '3,4,5' and in parent is 2 '6,2,7' . **Parent1 : 1 2 | 3 4 5 | 6 7**

Parent2 : 3 4 | 6 2 7 | 1 5

3- Exchange the element which is the same as selecting interval of the other parent structure for 'H' . In parent1 '2' at the left part and '6,7' at the right part are exchanged with 'H' . In parent2 , '3,4' at the left part and '5' at the right part are exchanged with 'H' . **1 H | 3 4 5 | H H**

H H | 6 2 7 | 1 H

4- Collect the 'H', in the selecting interval by shifting them inside of the cut points . Now we prevent the job repeat.

1 3 | H H H | 4 5

6 2 | H H H | 7 1

5- Swap the selecting intervals in the two parent structure . In our example parent1 and parent2 exchanged their '3,4,5' and '6,2,7' parts. Two children are constructed at the end of the crossover procedure.

Child1 : 1 3 | 6 2 7 | 4 5

Child2 : 6 2 | 3 4 5 | 7 1

5- Computational results

In order to evaluate the success of the two algorithms on HFS problems , it was run on different problems with wide range of jobs. Five different problem were examined , they correspond to the different job number 10,30,50,100 and 200, respectively. For each problem , 10 tests are generated by setting different processing time randomly and at the same interval , so all together 50 tests are performed.

The job processing time p_{ij} of job j on stage i $(i = 1,2; 1 \leq j \leq n)$ are uniformly distributed integers in the range between 1 and 30 , we have generated the values of p_{ij} by the following way[13]:

```
For i = 1 to 2
    For j = 1 to n
        $P_{ij} = U[1,30]$
```

The proposed pso can be compared to the improved proposed pso on the basis of cpu time because the computational environment of the two algorithms is the same. Also a comparison is made based on the solution quality , measured by the percentage difference between the solution and the lower bound as follows:

$$\% \text{ difference} = \frac{C_{max} - LB}{LB} * 100$$

The two algorithms were programmed in MATLAB (R2011a) and run on a PC with Intel(R) core (TM) i5-2430M cpu 2.40 GHZ and RAM 4GB . The algorithm parameters were set as follows: We chose $c_1=c_2=2$ depending on the previous experiment[10][14] ,also we chose w= 0.9 because when w=0.9 , the pso takes the least average number of iterations to find the global optimum [16].

The computational results are summarized in Table(2) , in which the "% difference " columns show the performance comparison among two algorithms.

Table(2) Solutions of the HFS problems

No. Of jobs	generated problems	LB	Proposed pso		Improved proposed pso		% difference	
			C_{max}	CPU time	C_{max}	CPU time	Proposed pso	Improved Proposed pso
10	Prob.1,1	84	84	0.1092	84	0.0312	0%	0%
	Prob.1,2	77	77	0.2028	77	0.1248	0%	0%
	Prob.1,3	97	97	0.0312	97	0.0156	0%	0%
	Prob.1,4	65	65	0.0468	65	0.0156	0%	0%
	Prob.1,5	84	85	0.6708	85	0.6240	1.19%	1.19%
	Prob.1,6	89	89	0.0312	89	0.0156	0%	0%
	Prob.1,7	102	103	0.4368	102	0.2028	0.98%	0%
	Prob.1,8	93	93	0.0312	93	0.0312	0%	0%
	Prob.1,9	47	48	1.0452	47	0.9048	2.13%	0%
	Prob.1,10	93	94	0.6552	94	0.5928	1.08%	1.08%
30	Prob.2,1	221	222	0.2808	221	0.2652	0.45%	0%
	Prob.2,2	215	217	0.5616	216	0.5304	0.93%	0.47%
	Prob.2,3	216	216	0.1404	216	0.0624	0%	0%
	Prob.2,4	197	199	0.8268	198	0.7332	1.02%	0.51%
	Prob.2,5	259	259	0.1872	259	0.1248	0%	0%
	Prob.2,6	230	231	0.2808	230	0.0936	0.43%	0%
	Prob.2,7	224	224	0.1092	224	0.0624	0%	0%
	Prob.2,8	216	216	0.0156	216	0.0624	0%	0%
	Prob.2,9	245	246	0.2808	245	0.1716	0.41%	0%
	Prob.2,10	237	238	0.2964	237	0.1404	0.42%	0%
50	Prob.3,1	364	365	0.5304	364	0.2184	0.27%	0%
	Prob.3,2	364	366	0.9048	365	0.4524	0.55%	0.27%
	Prob.3,3	308	309	0.4680	308	0.3744	0.32%	0%
	Prob.3,4	419	421	0.5616	419	0.3744	0.48%	0%
	Prob.3,5	422	422	0.3120	422	0.2340	0%	0%
	Prob.3,6	426	427	0.4524	426	0.0312	0.23%	0%
	Prob.3,7	447	447	0.4056	447	0.2964	0%	0%
	Prob.3,8	377	379	0.8580	377	0.0156	0.53%	0%
	Prob.3,9	401	402	0.4680	401	0.4368	0.25%	0%
	Prob.3,10	361	361	0.0780	361	0.0312	0%	0%

Table(2) (Continued)

No. Of jobs	generated problems	LB	Proposed pso		Improved proposed pso		% difference	
			C_{max}	CPU time	C_{max}	CPU time	Proposed pso	Improved Proposed pso
100	Prob.4,1	786	788	0.9048	787	0.8892	0.25%	0.13%
	Prob.4,2	812	813	0.9048	813	0.9048	0.12%	0.12%
	Prob.4,3	812	812	0.4680	812	0.3588	0%	0%
	Prob.4,4	808	808	0.5148	808	0.2808	0%	0%
	Prob.4,5	813	814	0.9048	813	0.6396	0.12%	0%
	Prob.4,6	776	776	0.5928	776	0.1092	0%	0%
	Prob.4,7	710	711	0.9828	711	0.8892	0.14%	0.14%
	Prob.4,8	789	791	0.9204	790	0.9048	0.25%	0.13%
	Prob.4,9	838	839	0.9048	838	0.2496	0.12%	0%
	Prob.4,10	756	756	0.2184	756	0.0156	0%	0%
200	Prob.5,1	1523	1524	2.9796	1523	1.7004	0.07%	0%
	Prob.5,2	1509	1511	2.8704	1510	2.7924	0.13%	0.07%
	Prob.5,3	1606	1607	2.8548	1607	2.7456	0.06%	0.06%
	Prob.5,4	1462	1463	2.8704	1462	2.1996	0.07%	0%
	Prob.5,5	1589	1591	2.8392	1590	2.8080	0.13%	0.06%
	Prob.5,6	1527	1527	0.7488	1527	0.7488	0%	0%
	Prob.5,7	1587	1587	2.1372	1587	0.4524	0%	0%
	Prob.5,8	1519	1520	2.8548	1520	2.7612	0.07%	0.07%
	Prob.5,9	1578	1580	2.7612	1579	2.7300	0.13%	0.06%
	Prob.5,10	1594	1595	2.9328	1594	0.0624	0.06%	0%

The general performance of the two compared algorithms is summarized in Table(3) , in which the (% solved) columns show the percentage of problems(with respect to all 50 problems) which are solved to their LBs and (% difference) columns show the average percentage differences from the LBs.

Table (3) General performance of the two algorithms

method	% solved	% difference
Proposed pso	0.38	0.268
Improved proposed pso	0.72	0.087

Table (3) shows that the improved proposed pso can solve 36 of the 50 problems (72%) with smallest average percentage difference among the two algorithms , while the proposed pso can solve only 19 of the 50 problems.

The average percentage differences (from LBs) due to the number of jobs are given in Table (4) and are represented in Fig.(2).

Table(4) The average percentage differences (from LBs) due to the no. of job

No. of job	**10**	**30**	**50**	**100**	**200**
The average percentage difference	**0.227**	**0.098**	**0.027**	**0.052**	**0.032**

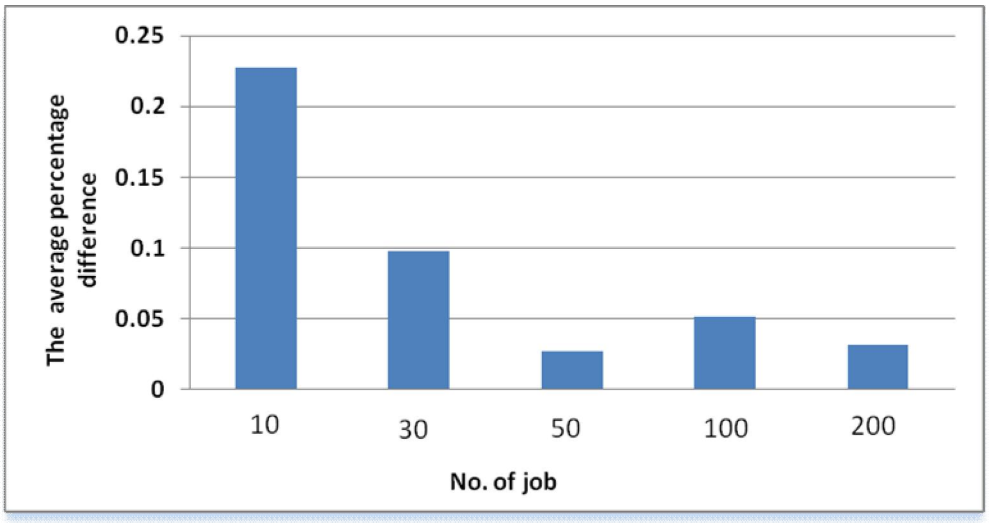

Fig.(2) The average percentage differences (from LBs) due to the no. of job

6-Conclusions

1- The results show that both of the algorithms applied successfully in this study on up the two-stage hybrid flow shop problem, also it shows that the improved proposed pso algorithm was the best in finding out the optimal makespan $(C^*_{max} = LB)$ .Even in case this algorithm camot achieve what is mentioned above, also it is the best in finding scheduling jobs which have the best makespan because the average percentage difference from LB is minimum.

2- Depending on computational results , the improved proposed pso algorithm takes less cpu time of the proposed pso algorithm for finding out the solution, except three problems where cpu time for both algorithms were equal and just one problem the cpu time was maximum when we used the improved proposed pso algorithm(Table(1)), this shows that the proposed improvement of the initial swarm helps widely in reducing of cpu time to find the solution.

3- By recognizing Table(4) ,the improved proposed pso algorithm it's performance will be better when the no. of jobs is high where the average percentage difference from LB will be less.

References

1. Alayky'ran,K., Engin,O. and Döyen,A.,(2007),"Using ant colony optimization to solve hybrid flow shop scheduling problems",***Int J Adv Manuf Technol***,Vol.35,PP.541-550.
2. Bean,J.,(1994),"Genetic Algorithms and Random Keys For Sequencing and Optimization",***ORSA Journal on computing*** ,Vol.6,No.2,PP.154-160.
3. Etiler,O., Toklu,B., Atak,M. and Wilson,J.,(2004),"A Genetic Algorithm For Flow Shop Scheduling Problems", ***Journal of the Operational Research Society***,Vol.55, No.8,PP.830-835.
4. Falkenauer,E. and Bouffoix,S.,(1991),"A genetic algorithm for job shop",***IEEE,International Conference on Robotics and Automation***,PP.824-829.
5. Gupta,J.,(1988),"Two-Stage,Hybrid Flow Shop Scheduling Problem",***Journal of the Operational Research Society***, Vol.39,No.4,PP.359-364.
6. Haouari,M., Hidri,L. and Gharbi,A.,(2006),"Optimal scheduling of a two-stage hybrid flow shop",***Math.Meth. Oper.Res.***,Vol.64,PP.107-124.
7. Huang,K., Yang,C. and Tsai,C.,(2012),"A Two-Phase Hybrid Particle Swarm Optimization Algorithm For Solving Permutation Flow-Shop Scheduling Problem", ***International Journal of Computer Applications***,Vol.48, No.1,PP.11-18.
8. Kennedy,J. and Eberhart,R.,(1995),"Particle Swarm Optimization",***Proceedings of IEEE International Conference on Neural Network,Piscataway,NY***,PP.1942-1948.
9. Liao,C., Tjandradjaja,E. and Chung,T.,(2012),"An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem", ***Applied Soft Computing***,Vol.12,PP.1755-1764.
10. Liao,C., Tseng,C. and Luarn,P.,(2007),"A discrete version of particle swarm optimization for flow shop scheduling problems",***Computers & Operations Research***,Vol.34, PP.3099-3111.
11. Li-Ping,Z., Huan-jun,Y. and Shang-xu,H.,(2005), "Optimal choice of parameters for particle swarm optimization",***Journal of Zhejiang University Science***, Vol.6,PP.528-534.
12. Liu.M., and Wang,S.,(2013),"A heuristic method for two-stage hybrid flow shop with dedicated machines", ***Computers & Operations Research***,Vol.40,PP.438-450.
13. Oğuz,C., Fikret Ercan,M., Edwin Chang,T.C. and Fung, Y.F.,(2003),"Heuristic algorithms for multiprocessor task scheduling in

two-stage hybrid flow shop",***European Journal of Operational Research***,Vol.149,PP.309-403.

14. Qiao,p., and Sun,C.,(2011),"Research on Hybrid Flow- Sop Scheduling Problem Based on Improved Immune Particle Swarm Optimization", ***2nd International Conference on Artificial Intelligence,Management Science and Electronic Commerce(AIMSEC)***,PP.4240-4243.
15. Santos,D., Hunsucker,J., and Deal,D.,(1995),"Global lower bounds for flow shops with multiple processors", ***European Journal of Operational Research***, Vol.80, PP.112-120.
16. Shi,Y., and Eberhart,R.,(1998),"A modified Particle Swarm Optimizer",***IEEE,proceeding of the world congress on Computational Intelligence***,PP.69-73.
17. Wang,S. and Liu,M.,(2013),"A genetic algorithm for two-stage no-wait hybrid flow shop scheduling problem", ***Computer & Operations Research,***Vol.40, PP.1064-1075.
18. Xiao,W., Hao,P., Zhang,S. and Xu,X.,(2000),"Hybrid flow shop scheduling using genetic algorithms",***IEEE, proceeding of the 3rd world congress on intelligent control and automation,June 28- July 2 ,Hefei,P.R.china.***
19. Yue-Wen,F., Feng-Xing,Z., Xiao-hong,X., Qing-Zhu,C., and Jia-hua,W.,(2011),"Hybrid Flow-Shop Scheduling Method Based on Multi-agent particle Swarm Optimization ",***IEEE,International Conference on Information and Automation***,PP.755-759.