Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management
Kuala Lumpur, Malaysia, January 22 – 24, 2011

A tabu search approach for makespan minimization in a permutation flow shop scheduling problems

Sawat Pararach
Department of Industrial Engineering, Faculty of Engineering,
Thammasat University, Pathumthani 12121. THAILAND
E-mail: psawat@engr.tu.ac.th

Abstract

On a permutation flowshop scheduling problem(PFSP), *n* jobs are arranged on *m* machines in a series with the same routing for each machine. The objective of scheduling is to find a sequence of jobs under minimization of makespan. From many researches, the upperbound values from the benchmark problems have been shown. In this paper, a tabu search approach for improving the upperbound values from the previous works is investigated. The results show that the proposed procedure yields new better upperbound values for the benchmark problems.

Keywords
Permutation flowshop, flowshop scheduling, tabu search.

1. Introduction

Flow shop scheduling problems (FSP) have been addressed extensively many years ago. The general characterizations of FSP are as follows: there are *n* jobs; each job comprises *m* operations; each operation requires a different machine; all jobs are processed in the same processing sequence to minimize (or maximize) a given objective function. The makespan is denoted by C_{max} and is defined as the time the last job leaves the system; that is, $C_{max}= max(C_1,...,C)$, where C_j is the completion of job *j*. Makespan is related to machine utilization for a static scheduling problem, minimizing makespan thus implies minimizing machine idle time. The FSP on at least three machines is computationally intensive tasks [1]. Most studies further restrict the FSP by assuming an identical processing order on all machines called a permutation flow shop problem (PFSP).

The *n*-job, *m*-machines permutation PFSP with minimized makespan (C_{max}) generally has the following assumptions [2]:

- The number of jobs on each machine and their processing times are known in advance.
- The number of machines is known, and all machines are always available.
- A job must be processed once on each machine and in the same machine order.
- The setup times of the operations are included in the processing times, and are sequence-independent.
- No pre-emption (i.e. process interruption) of operations is allowed.
- All the jobs are available at time zero.
- Each job is processed on a maximum of one machine at a time.
- No machine can process more than one operation at a time.
- Machines may be idle within the schedule period.

The total number of possible schedules in permutation FSP is $n!$, where *n* is the number of jobs to be processed. In many researches, PFSP can be solved in an appropriate time and obtained a near optimal solution by meta-heuristic approaches as follows:

A particle swarm optimization algorithm (PSO) [3] employs the neighborhood structure by interchange two jobs between *i* and *j* positions or remove the job in *i* position and insert it in the *j* position. Population in this method is the set of particles in the swarm. The permutation rule is considered as creating a new population in the next iteration. The results from a PSO show that in the case of $n \geq 50$, the PSO outperforms the existing procedures. For an ant colony optimization algorithm ((MHD-ACS) [4], each iteration initializes the pheromone trail to construct a solution and modifies them. An iterated greedy method with local search [5] iteratively applies two phases, destruction and

construction. For destruction, some jobs are eliminated from the incumbent solution. For construction, the eliminated jobs are reinserted in the sequence. Cooperative meta-heuristics [6], a task is divided into several parts and solved simultaneously using multi-processors. Each part is needed in order to run a meta-heuristic procedure, such as the genetic algorithm, tabu search and simulate annealing. A discrete firefly algorithm [7] creates the motivation from a firefly *i* attracted to another more attractive firefly *j*. For the benchmark problems, there are many generated test problems for the PFSP, such as Applegate and Cook[8], Storer et.al[9], Tailard[10] and Demirkol et.al[11]. For the real world instances, the generated test problems are available by Demirkol et al.[11].

The aim of this paper is to propose a method that yields a new upperbound value for the Demirkol's test problems while performs within a suitable time. The experimented result of 40 benchmark problems shows that the upperbound value of the proposed approach is better performance than the previous research.

The remainder of this paper is organized as follows: Section 2 contains a review of tabu search approach, Section 3 considers the proposed approach while Section 4 shows the results of solving benchmark problems by the proposed approach. Discussion and conclusion are presented in Section 5 and 6.

2. Methodology Review

Tabu Search is a neighborhood search based meta-heuristic aimed to escape the local minima. The best solution in the neighborhood of the current solution should be chosen to be a new current solution, even if this implies increasing the cost. Moving from a local minimum by choosing the best value in its neighborhood becomes a better solution in the new neighborhood [12].

For any solution *s*, a subset of moves to *s* is applicable. This subset of moves generates a subset of solution *NH(s)*, called the neighborhood of *s*. Starting from an initial solution, TS iteratively moves from the current solution *s* to the best solution *s** in *NH(s)*, even if *s** is worse than the current solution *s*, until reaching a maximum iteration number.

In order to prevent cycling, moves which bring it back to a recently visited solution, that should be forbidden or declared tabu for a certain number of iteration. This is accomplished by keeping the attributes of the forbidden moves in a list, called a tabu list. The size of the tabu list must be large enough to prevent cycling. There are several issues and refinements to consider in order implementing an effective and efficient algorithm, such as tabu move, an aspiration criterion and tabu tenure. Tabu moves are the forbidden swapping of some attributes to prevent the cycling of job sequences. An aspiration criterion is the relaxation of a tabu move to restrict the large search process. Tabu tenure are the length of tabu move or the number of iterations at which job *i* is allowed to return to the j^{th} position of the scheduling sequence. The termination rule is the condition that the maximum iterations are reached [13].

The basic tabu navigation algorithm can be described as follows:

Step 1: choose an initial current solution x_{cur}, a tabu list size, let k=1, $f^*=f(x_{cur})$, $x^*= x_{cur}$.

Step 2: evaluate the neighborhood, $NH(x_{cur})$, update the current solution with the best non tabu solution in the neighborhood, if necessary update $x^*$and f^*,

Step 3: add some attribute of the new solution or apply perturbation to the tabu list,

Step 4: if the maximum iteration number has been reached, stop, otherwise set $k=k+1$, and go to step 2.

2.1 **Neighborhood**

In an FSP, two kinds of move are alternately used as follows: (1) Insertion—an insertion move consists of moving a job from its current position and inserting it into another position, and (2) Swapping—a swapped move consists of exchanging positions of two jobs[14]. Both inserted and swapped moves are defined an $O(n^2)$ neighborhood.

2.2 **Tabu move**

Tabu move is used to determine whether a solution with a characteristic attribute has been visited or not. If TS finds a candidate solution that possesses the attributes of a recently visited solution within tabu tenure, the move is forbidden and the next candidate move is entertained [15].

2.3 **An aspiration criterion**

An aspiration criterion is defined to deal with the case in which an interesting move is tabu. If a current tabu move satisfies the aspiration criterion, its tabu status is cancelled and it becomes an allowable move. In order to improve the performance of TS, intensification strategies can be used to accentuate the search in a promising region of the solution

space. An intensification strategy consists of storing the elite solutions and restarts the search from them so as to explore neighborhoods with potentially good solutions. The recovery of the elite solutions is deferred until the last stage of the search. The elite solutions, a set of best solutions found so far, are recovered in the list of the worst solution to the best solution [16].

2.4 **Tabu tenure**

Tabu tenure is used to force the search process to visit the search regions that are not yet explored. For each swapped move which job i in position l and job k in position r are exchanged, the tabu tenure is penalized according to the frequency of job i in position r and the frequency of job k in position l. For each inserted move which the job in position l is inserted into position r, $r>l$, the tabu tenure is penalized according to the frequency of job i in position k where $r \leq k \leq l$ [17].

3. Proposed Methodology

The concept of the proposed algorithm is based on improving the solution from the constructive algorithm by a local search procedure. The selected constructive algorithm should be polynomial and solvable. Therefore, the proposed approach generates the initial solution(s) by obtaining job sequence(s) from the $O(n^4 m)$ polynomial time algorithm. This method is discussed by Laha and Sarin [18].

3.1 **An $O(n^4m)$ polynomial time algorithm.**

The objective of this algorithm is for generating the initial solution of permutation FSP. Steps are as follows:

Step 1: For each job i and machine j, given the processing time p_{ij}, then find the total processing times, P_i on all machines, as in Equation (1),

$$P_i = \sum_{j=1}^{m} p_{ij} \quad \text{for all } i=1,2,\ldots,n. \tag{1}$$

Step 2: Sort jobs in ascending order of their total processing time.

Step 3: Set k=2. Select the first two jobs from the sorted list and select the better between the two possible sequences.

Step 4: For k=3 to n do a following:

Insert the k^{th} job on the sorted list into k possible positions of the (k-1)-job current sequence and select a sequence from these a k-job partial sequence with the best permutation makespan value. Designate it as a k-job current sequence. Place each job (except the k^{th} job in the sorted list) of this sequence into the (k-1) position and select the best k-job sequence having the best makespan value from those generated sequences. This becomes the next k-job current sequence.

Step 5: If $k=n$, then go to Step 6, else go back to Step 4.

Step 6: Swap jobs in position i and j for all i,j ,$1 \leq i < n$, $i < j \leq n$. Select the best sequence that has minimum makespan obtained from the $n(n-1)/2$ sequences.

Steps 1-5 of algorithm are obtained from Laha and Sarin[18], except step 6 is added to improve the current solution, however the algorithm is solvable in polynomial time.

3.2 **The proposed TS approach**

The proposed approach is the tabu search procedure to determine the new neighborhood by setting tabu tenure to a constant value and stops the procedure when the maximum iteration has been reached. Two types of position range of jobs in a sequence are randomly generated, and considered for solving by the proposed procedure. Type 1 is generated from1 to n-1 whereas Type 2 is generated from 1 to n. The steps of proposed TS approach are as follows:

Step 0: Set the maximum number of iterations, *Iter*, the fixing tabu tenure, H, and the maximum value, obj_{best} . Select a set of seed number randomly with four digits, and set type of position range, *type=1*.

Step1: Generate an initial solution, X, from $O(n^4m)$ polynomial time algorithm that mentioned above, $X_{initial}$ and $obj_{initial}$ are the obtained sequence and makespan value, respectively.

Step2: Set $X=X_{initial}$ and $obj(x)=obj_{initial}$. Tabu move can be presented as $tabu(i,j) = 0$ for all i,j, $i=1,..n$, $j=1,..,n$;

Step3: Set number of iteration $I=0$; obj_{local} = the maximum objective value.

Step4: Swapping two jobs in all of job sequences of solution X. If the swapping between job i and j has $tabu(i,j) > 0$, this swapping is discarded, set $tabu(i,j)=tabu(i,j)+1$ and a new swapping of jobs is generated. If the $tabu(i,j)$ equals to H, setting $tabu(i,j)$ to zero. Solution Y is a sequence that obtained the minimum makespan value from all of swapping of job sequence of these sets.

Step5: $I=I+1$; If $obj(y)<obj_{local}$ then $X_{local}=Y$ and $obj_{local}=obj(y)$; replacing solution X with solution Y; $obj(x)=obj(y)$, otherwise randomly generate a position w_1 of job sequence at machine *1* and *2*, a position w_2 of job sequence at machine *3* to *m-2* , and a position w_3 of job sequence at machine *m-1* and *m*. If $type=1$, the positions w_1, w_2, and w_3 are randomly generated between 1 and n-1, otherwise the positions w_1, w_2, and w_3 are randomly generated between 1 and n ($type=2$). Setting $u_1=w_1+1$,$u_2=w_2+1$ and $u_3=w_3+1$. If u_1 or u_2 or u_3 are greater than n, set u_1 or u_2 or u_3 to 1. Solution Y_1 is a sequence from swapping positions w_1 and u_1. Solution Y_2 is a sequence from swapping positions w_2 and u_2. Solution Y_3 is a sequence from swapping positions w_3 and u_3. Let solution Z be the best solution from solution Y_1, Y_2, and Y_3 that obtained minimum value of makespan. The w_x and u_x are the swapping positions of changing solution X to solution Z. Replacing solution X with solution Z and set $tabu(w_x,u_x)=1$.

Step6: If $I=Iter$ then goto step 7, else go to step 4.

Step7: If $obj_{local} < obj_{best}$ then $obj_{best} = obj_{local}$; $X_{best}=X_{local}$.

Step8: If $type=2$ then stop the procedure, otherwise set $type=1$ and go to step 2.

For the repeated step 2 -8 of setting $type=1$ and $type=2$, the cooperative computation of Vallada and Ruiz(2009) could be implemented. It has been reduced a half of the computational times.

The results of performing by the proposed approach are in Table 1.

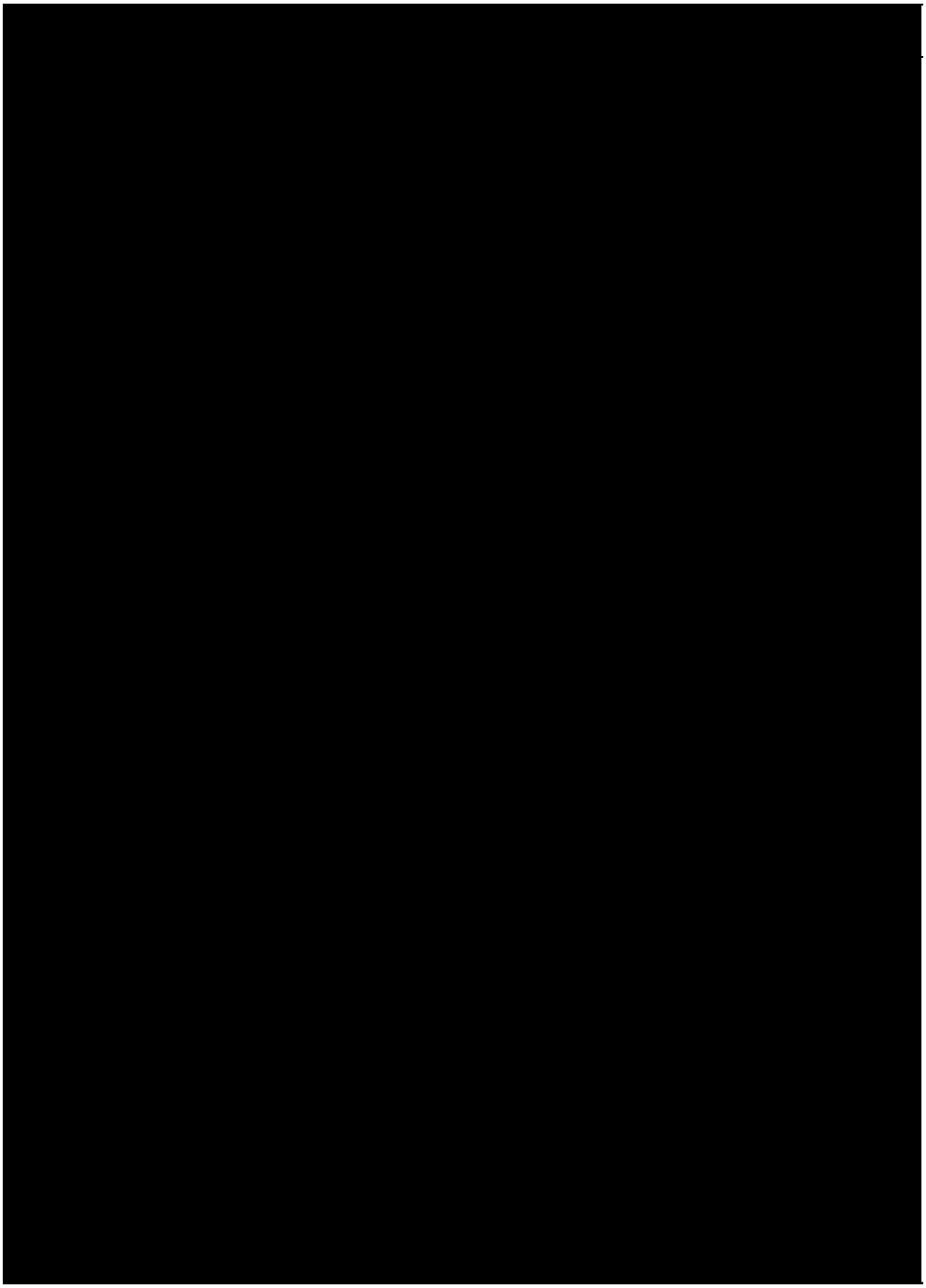
4. Computational results

The performance of the proposed algorithm is examined and compared with the best result of the other existing approach, in the 40 benchmark problems that given by Demirkol et al. [11] in Table 1. The available benchmark problems can be downloaded from www.ecn.purdue.edu/labs/uzsoy. The test instances contain two machine number values (m=15, 20) and four job number values (n=20, 30, 40, 50), eight combinations of m and n are generated. The total number of operations ranges from 300 to 1000. The *flcmax_n_m_r* is the name of an instance from the downloaded data. The code was implemented using C++ language and run on a personal computer, Intel Pentium Dual-Core (2.70 GHz.) CPU and RAM 2 GB. The parameters, $Iter$ = 20,000 and *tabu tenure* =5 are used in the proposed approach. A four digits of seed number, 1234, is set for generating a random values. From the obtained results in Table 1, the best makespan value was bolded. Type 1 is the TS approach that is randomly generated a position of job sequence between 1 to n-1. Type 2 is the TS approach that randomly that is generated a position of job sequence between 1 to n. The results shown that one of the obtained value from two types is better than the previous best known value and given the new upperbounds for all benchmark problems. The computation times of the proposed approach are the sum of running of two types, whereas the discrete-firefly algorithm is run on a personal computers with Intel Pentium(R) (3GHz) CPU 1GB RAM, it could not accountable that the proposed approach is better performance than the discrete-firefly algorithm, however they show that the proposed method can solve in suitable time.

5. Discussion

The proposed method is an effective constructive algorithm for solving the PFSP. The algorithm distributes the searching spaces widely than the discrete-firefly procedure. In this paper, the sequential processes are considered by implementing the a position of job sequence, $type=1$ and $type=2$ procedures. The best value of two types gives the new best-known value from the benchmark problems. The computational times are up to 653 seconds for the large size of test problem. It has been shown that the proposed algorithm performs in a suitable time. The parameters of the proposed procedure are investigated like Lin and Ying[18] and set them to constant values for solving each test problem.

Table 1 Results obtained by the discrete-firefly method and the proposed TS approaches



Note: Type 1 is the TS approach with randomly generated a position of job sequence between 1 to n-1.
Type 2 is the TS approach with randomly generated a position of job sequence between 1 to n.

6. Conclusion and remarks

The proposed tabu search procedure yields better upperbounds of the 40 benchmark problems from Demirkol et.al.[11] in suitable times. It implies that the searching steps of the proposed approach is better than the discrete-firefly method of Sayadi et.al[7] and the previous works. The future work, computation time could be improved by using the parallel computing for distributing the searching spaces of the swapping in the proposed algorithm.

References

1. Garey, M., Johnson, D., & Sethi, R., 1976, "The complexity of flowshop and jobshop scheduling," Mathematics of Operations Research, 1 (2), 117–129.
2. Brandimarte, P. and Villa, A., 1995, Advanced Models for Manufacturing Systems Management, New York, U.S.A; CRC Press.
3. Tasgetiren, M. F., Liang, Y-C., Sevkli, M., & Gencyilmaz, G., 2007, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," European Journal of Operational Research, 177, 1930–1947.
4. Ying, K.-C., & Lin, S.-W., 2007, "Multi-heuristic desirability ant colony system heuristic for non-permutation flowshop scheduling problems," International Journal of Advanced Manufacturing Technology, 33, 793– 802.
5. Ruiz, R., & Stützle, T., 2007, " A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," European Journal of Operational Research, 177, 2033–2049.
6. Vallada, E. & Ruiz, R., 2009, "Cooperative metaheuristics for the permutation flowshop scheduling problem," European Journal of Operational Research, 193- 365–376.
7. Sayadi, M. K., Ramezanian, R. and Ghaffari-Nasab, N., 2010, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems," International Journal of Industrial Engineering Computations, 1, 1–10.
8. Applegate, D. and Cook, W., 1991, "A computational study of the job shop scheduling problem," ORSA Journal on Computing, 3, 149-156.
9. Storer, R.H., Wu, SD. and Vaccari, R., 1992, "New search spaces for sequencing problems with application to job shop scheduling," Management Science , 38, 1495-1509.
10. Taillard, E.D., 1994, "Parallel taboo search techniques for the job shop scheduling problem," ORSA Journal on Computing, 6, 108-117.
11. Demirkol, E., Mehta, S., & Uzsoy, R., 1998, "Benchmarks for shop scheduling problems, " European Journal of Operational Research, 109, 137–141.
12. Glover, F.,1995, "Tabu Thresholding: Improving Search by Nonmonotonic Trajectories,"ORSA Journal of Computing ,7(4),426-442.
13. Lin, S.W. and Ying, K.C., 2009, "Applying a hybrid simulated annealing and tabu search approach to non-permutation flowshop scheduling problems," International Journal of Production Research, 47(5), 1411-1424.
14. Liaw, C., 2003, "An efficient tabu search approach for the two-machine preemptive open shop scheduling problem," Computer& Operations Research, 30, 2081-2095.
15. Chen, J.S., Pan, J.C.H., and Wu, C.K., 2007, "Minimizing makespan in reentrant flow-shops using hybrid tabu search" Int. J. Adv. Manuf. Technol., 34, 353-361.
16. Ekşioğlu, B., S.D. and Jain, P., 2008, "A tabu search algorithm for the flow shop scheduling problem with changing neighborhoods," Computer & Industrial Engineering, 54,1-11.
17. Abdullah, S., Ahmadi, S, Burke E.K., Dror, M., and McCollum, B., 2007, "A tabu-based neighborhood search methodology for the capacitated examination timetabling problem," Journal of the Operational Research Society,58,1494-1502.
18. Laha D. and Sarin S.C. 2009, "A heuristic to minimize total flow time in permutation flow shop," Omega, 37:734-739.