11

Hybrid Tabu Search for Re-Entrant Permutation Flow-Shop Scheduling Problem

Jen-Shiang Chen[1], Jason Chao-Hsien Pan[2] and Chien-Kuang Wu[2]
[1]*Far East University,*
[2]*National Taiwan University of Science and Technology,*
Taiwan,
R.O.C

1. Introduction

The assumption of classical shop scheduling problems that each job visits each machine only once (Baker, 1974) is often violated in practice. A new type of manufacturing shop, the re-entrant shop has recently attracted attention. The basic characteristic of a re-entrant shop is that a job visits certain machines more than once. For example, in semiconductor manufacturing, consequently, each wafer re-visits the same machines for multiple processing steps (Vargas-Villamil & Rivera, 2001). The wafer traverses flow lines several times to produce the different layer on each circuit (Bispo & Tayur, 2001). A re-entrant flow-shop (RFS) refers to situations in which every job must be processed on machines in the order, M_1, M_2, …, M_m, M_1, M_2, …, M_m, …, and M_1, M_2, …, M_m. Every job can be decomposed into several layers each of which starts on M_1 and finishes on M_m. In the RFS case, if the job ordering is the same on any machine at each layer, then no passing is said to be allowed, since no job is allowed to pass any former job. The RFS scheduling problem in which no passing is allowed, is called a re-entrant permutation flow-shop (RPFS) problem.

The assumptions made for the RPFS scheduling problems are summarized here. Every job may visit certain machines more than once. Machine order is the same for each of the n jobs. Job order is the same for each of the m machines at each layer. The classical permutation flow-shop scheduling problem can be modified to suit the RPFS scheduling problem by relaxing the assumption that each job visits each machine no more than once. This study considers the RPFS scheduling problems with the objective of minimizing makespan of jobs.

Hwang & Sum (1998) addressed a two-machine flow-shop problem with re-entrant workflows and sequence dependent setup times, which have a special structure, to minimize makespan. Demirkol & Uzsoy (2000) proposed a decomposition method to minimize maximum lateness for a RFS with sequence-dependent setup times. Graves et al. (1983) modeled a wafer fabrication as a RFS, where the objective is to minimize average throughput time subject to meeting a given production rate. Drobouchevitch & Strusevich (1999) developed a heuristic algorithm for the two-machine re-entrant shop problem to minimize the makespan. Kubiak et al. (1996) considered a class of re-entrant shops in which jobs followed the route of M_1, M_2, M_1, M_3,…, M_1, M_m, M_1 with the objective of minimizing the mean flow time. They showed that the shortest-processing-time (SPT) rule was optimal

Source: Local Search Techniques: Focus on Tabu Search, Book edited by: Wassim Jaziri, ISBN 978-3-902613-34-9, pp. 278, October 2008, I-Tech, Vienna, Austria

provided certain restrictive conditions held. Wang et al. (1997) proposed the scheduling of a chain-reentrant shop in which each job is first processed on a machine called the primary machine, then on other machines in a fixed sequence, and finally back to the primary machine for last operation. The objective of the problem is to minimize the makespan.

Tabu search (TS) is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality. The local procedure is a search that uses an operation called move to define the neighborhood of any given solution. One of the main components of TS is its use of adaptive memory, which creates a more flexible search behavior. Memory-based strategies are the hallmark of TS approaches (Glover & Languna, 1997). It has been shown to be a remarkably effective approach in a wide spectrum of problem areas from general integer and nonlinear programming to sequencing and production scheduling problems. Tabu search is a local search based optimization method that has been successfully used to solve many difficult combinatorial optimization problems, particularly in the scheduling area. These methods suggested by Glover (1989) can be sketched as follows: starting from an initial feasible solution, at each step we choose a move to a neighboring solution in such a way that we move stepwise towards a solution giving hopefully the minimum value of some objective function. Nowicki & Smutnicki (1996, 1998) developed effective TS methods for job-shop, flow-shop, and flow-shop with parallel machines problems to optimize the makespan criterion. These algorithms employ a classical insertion neighborhood, which is significantly reduced by a candidate list strategy for removing useless moves, in order to concentrate on "the most promising part" of the neighborhood.

As to the $n/m/J/C_{\max}$ problem which has been studied for a long time and is known to be NP-hard (Garey et al., 1976), the algorithm given by Adams et al. (1988), called shifting bottleneck uses the iterative solutions of a single bottleneck machine problem to build up and improve a schedule. Better solutions than the ones given by deterministic algorithms were found using simulated annealing but at the cost of longer computations. Tabu search was the first applied to job-shop by Taillard (1989), who proposed a sequential and a parallel algorithm. Dell'Amico & Trubian (1993) applied TS to the notoriously difficult job-shop scheduling problem.

For $n/m/F/C_{\max}$ problems, Palmer (1965) developed a quick method of obtaining a near optimum and Campbell et al. (1970) presented a heuristic algorithm as well. Widmer & Hertz (1989) used a simple insertion heuristic based on an analogy with the traveling salesman to the flow-shop problem to generate the starting order of the jobs and tried to improve this solution using TS techniques. In direct competition with the heuristic developed by Nawaz et al. (1983), TS method performed superiorly for 58% of the problems and matched the best solutions found for 92% of the problems.

Pan & Chen (2003) presented three extended mixed binary integer programming formulations and six extended effective heuristics for solving RPFS scheduling problems to minimize makespan. The TS method has been used to solve classical flow-shop problems and has performed well. This study considers RPFS scheduling, and applies hybrid tabu search (HTS) to minimize the makespan of jobs. The hybridization method is used to improve pure TS performance. The HTS is compared to the optimal solutions generated using the integer programming technique (Pan & Chen, 2003), and to the near optimal solutions generated by pure TS and other heuristics proposed by Pan & Chen (2003).

2. The optimization model

A classical (permutation) flow-shop problem assumes that all operations of each job visit every machine exactly once in the order of M_1, M_2, …, and M_n. Define this order of processing to be a level, then the routing requirement of a job in a RPFS problem can be decomposed into several levels. Hence, a classical permutation flow-shop is a special case of a RPFS with a single level and some of its formulations can be extended to solve the RPFS. To illustrate the concept of level decomposition, consider job i consisting of six operations to be processed on two machines, where (i, j, k) denotes that operation j of job i must be processed on M_k and thus its routing is (i, 1, 1)→ (i, 2, 2)→ (i, 3, 1)→ (i, 4, 2)→ (i, 5, 1)→ (i, 6, 2) and the corresponding processing time of each operation is orderly 8, 2, 7, 4, 5, and 1. The processing requirement of job i can be decomposed into three levels, where (i, 1, 1)→ (i, 2, 2) is the first level, (i, 3, 1)→ (i, 4, 2) is the second, and (i, 5, 1)→ (i, 6, 2) is the third. Let O^i_{lk} be the operation of job i on machine k at level l, p^i_{lk} be the processing time of the operation of job i on machine k at level l. Consequently, $O^i_{11} = (i, 1, 1)$, $O^i_{12} = (i, 2, 2)$, $O^i_{21} = (i, 3, 1)$, $O^i_{22} = (i, 4, 2)$, $O^i_{31} = (i, 5, 1)$, $O^i_{32} = (i, 6, 2)$, $p^i_{11} = 8$, $p^i_{12} = 2$, $p^i_{21} = 7$, $p^i_{22} = 4$, $p^i_{31} = 5$, and $p^i_{32} = 1$.

2.1 Notations

M = a very large positive number;
m = number of machines in the shop;
n = number of jobs for processing at time zero;
L = number of levels of job i;
p^i_{lk} = the processing time of the operation of job i on machine k at level l;
x_{ij} = 1, if job i is scheduled in the jth position at each level; 0, otherwise;
h_{klj} = the starting time of the operation scheduled at jth position of level l on machine k;
$C_{\max}$ = the maximum completion time or makespan;

2.2 Formulation

Pan & Chen (2003) were the first authors to present the integer programming model for solving the reentrant permutation flow-shop problem. The binary variable x_{ij} that the model uses is restricted by a single permutation of the numbers 1, 2, …, n that specifies the order in which jobs are processed on any machine at each level. The model is as follows.

Minimize

$$C_{\max} \tag{1}$$

Subject to

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, 2, ..., n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, 2, ..., n \tag{3}$$

$$h_{111} = 0 \tag{4}$$

$$h_{1,1,j+1} = h_{11j} + \sum_{i=1}^{n} x_{ij} p^i_{11} \quad j = 1, 2, \ldots, n-1 \tag{5}$$

$$h_{1,l,j+1} \geq h_{1lj} + \sum_{i=1}^{n} x_{ij} p_{l1}^{i} \quad l = 2, 3, \ldots, L; j = 1, 2, \ldots, n-1 \tag{6}$$

$$h_{1,l+1,1} \geq h_{1ln} + \sum_{i=1}^{n} x_{in} p_{l1}^{i} \quad l = 1, 2, \ldots, L-1 \tag{7}$$

$$h_{1,l+1,j} \geq h_{mlj} + \sum_{i=1}^{n} x_{ij} p_{lm}^{i} \quad l = 1, 2, \ldots, L-1; j = 1, 2, \ldots, n \tag{8}$$

$$h_{k,l,j+1} \geq h_{klj} + \sum_{i=1}^{n} x_{ij} p_{lk}^{i} \quad k = 2, 3, \ldots, m; l = 1, 2, \ldots, L; j = 1, 2, \ldots, n-1 \tag{9}$$

$$h_{k,l+1,1} \geq h_{kln} + \sum_{i=1}^{n} x_{in} p_{lk}^{i} \quad k = 2, 3, \ldots, m; l = 1, 2, \ldots, L-1 \tag{10}$$

$$h_{k+1,1,1} = h_{k11} + \sum_{i=1}^{n} x_{i1} p_{1k}^{i} \quad k = 1, 2, \ldots, m-1 \tag{11}$$

$$h_{k+1,l,j} \geq h_{klj} + \sum_{i=1}^{n} x_{ij} p_{lk}^{i} \quad k = 1, 2, \ldots, m-1; l = 1, 2, \ldots, L; j = 1, 2, \ldots, n;$$

$$(l, j) \notin \{(1, 1)\} \tag{12}$$

$$h_{k+1,l+1,1} \geq h_{kln} + \sum_{i=1}^{n} x_{in} p_{lk}^{i} \quad k = 1, 2, \ldots, m-1; l = 1, 2, \ldots, L-1 \tag{13}$$

$$C_{\max} = h_{mLn} + \sum_{i=1}^{n} x_{in} p_{Lm}^{i} \tag{14}$$

$$C_{\max} \geq 0, h_{klj} \geq 0 \quad k = 1, 2, \ldots, m; l = 1, 2, \ldots, L; j = 1, 2, \ldots, n;$$

$$x_{ij} = 0 \text{ or } 1 \quad i = 1, 2, ..., n; j = 1, 2, ..., n \tag{15}$$

Constraint (1) describes the objective function. Constraints (2) to (5) and (11) are essentially definitional, while constraints (6) to (10), (12) and (13) enforce the precedence relationships. Constraint (14) defines $C_{\max}$ to be the finish time of the last job processed on M_m at the last level. The non-negativity and binary restrictions on h_{klj} and x_{ij}, respectively, are specified in (15).

3. Hybrid tabu search

The HTS method differs from pure TS that it is not likely to trap in local optimum. The main idea of HTS is that when neighboring solutions are not able to update the current best solution for a period of time, a good problem-specific heuristic or dispatching rule is combined in pure TS to explore new solution region. With this feature, HTS is able to avoid falling into local optimum and move toward a better solution.

3.1 Initial solution

The classical (permutation) flow-shop problem has been proved to be NP-complete (Coffman, 1976; Rinnooy Kan, 1976). Hence, many heuristics have been proposed to provide a quick and good solution. Some of the well-known heuristics include the methods proposed by Campbell et al. (1970), Dennenbring (1977), Johnson (1954), Nawaz et al. (1983), and Palmer (1965). Pan & Chen (2003) made appropriate modifications to these six heuristics to solve the RPFS scheduling problems by taking the reentry property into account. The results showed that heuristic NEH (Nawaz et al., 1983) outperform the other algorithms in the set of problems with unknown optimal solutions. Hence, NEH is used to generate initial solution for RPFS problems.

3.2 Neighborhood search

Neighborhood search starts from current solution and seeks to find feasible, hopefully better, solutions in its neighborhood. If the neighboring solution is better than current one, this current solution is replaced by the neighboring solution until stopping rules satisfied. When dealing with RPFS problems, we have to focus on the jobs. The main reason is that once the processing sequence is determined, every machine follows the same order for all jobs. The problem will be simpler when we focus on jobs instead of operations. The neighborhood solutions are produced by interchanging the job order of the initial solution.

3.3 Choosing a move

First, the makespan for each neighborhood solution is calculated. Second, the solution that has the minimal makespan among others and outside tabu list or meets aspiration criterion is selected as a move.

3.4 Recording in tabu list

Nowicki & Smutnicki (1996) suggested recording the number of jobs exchanges of the move in tabu list. By doing so, whether two jobs had performed exchange or not can be held in the tabu list. A move $v = (x, y)$ is added to tabu list T in the following standard way. The tabu list T is shifted one position forward and put v in the last position in the list, that is, $T_j = T_{j+1}$, $j = 1, 2, \ldots, maxt - 1$, and $T_{maxt} = v$. In this study, the length of tabu list is set to seven and *first-in-first-out* (FIFO) rule is adopted; that is, when the tabu list is full, the new move replaces the earliest one entering tabu list and adds the maximal searching times by one.

3.5 Recording the best-so-far solution

If the solution after the move is better than the current best-so-far solution, replace the best-so-far solution and reset the non-improvement times to zero; otherwise, add non-improvement times by one.

3.6 A hybrid method

When a new best solution cannot be found for longer than a predetermined number of iterations, that is, count > threshold, the search switches to heuristic phase. Normal TS in this situation usually calls for intensification or diversification strategies to get out of a local optimum. Typical intensification or diversification strategies keep memory structures for storing rather a rather long history of recent search activities and use these structures to

guide future search directions (Hwang et al., 2002). The idea of Hwang et al. (2002) is cited to find the hybrid occasion of TS and heuristic. When the non-improvement times increase continuously, it means that best-so-far solution is not replaced by neighborhood solutions for a period of time, which is a signal that TS is likely to entrap in local optimum. In this situation, a hybrid method is introduced to explore new solution region. A threshold is set to twenty, which means that once the non-improvement times were cumulated to twenty, NEH is hybridized into TS to find a new solution. After that, non-improvement times are reset to zero and the searching process proceeds based on the new solution until stopping rules are satisfied. The overall procedure for the HTS algorithm is as follows.

Pseudo-code for the HTS algorithm
Find an initial solution x
Define tabu structure and set $Count = 0$
Repeat until stopping condition is met
Generate neighborhood sets of x: S_1, S_2, ..., S_k
Select the best non-tabu solution x' from $S_1 \cup S_2 \cup \ldots \cup S_k$
$x \leftarrow x'$
if x is better than the current best-so-far solution **then**
  $Count = 0$
else
  $Count = Count + 1$
end if
if $Count >$ threshold **then**
  Update x by calling NEH heuristic and set $Count = 0$
end if

This hybrid method is illustrated in Fig. 1. Suppose the sequence of a schedule is (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), two position C_1, C_2 are selected randomly from 1 to 12, representing the starting and end point of the substring (C_1, C_2). This substring is then treated as a sub-problem and solved by NEH heuristic. The new sequence then replaces the original substring (C_1, C_2). By doing this, a new solution is generated and serves as a new starting point of TS in order to get rid of local optimum.

3.7 Stopping rules

There are two stopping rules considered in this study and they are stated below.

(1) Non-improvement times:
This rule counts the number of non-improvement moves for TS. When the best-so-far solution cannot be replaced after one iteration, this counter adds by one.

(2) Max iteration:
This is the maximal iteration number that a TS takes. Once this number is reached, the TS is terminated.

4. An illustrative example

A small size problem of RPFS is given in this section. In the example, it is assumed that there are five jobs ($n = 5$), three machines ($m = 5$), and each job reenter twice ($L = 2$) in the shop. In a RPFS problem, the job sequence on each machine is the same and any sequence change on one machine will result in the sequence change in the rest machines.

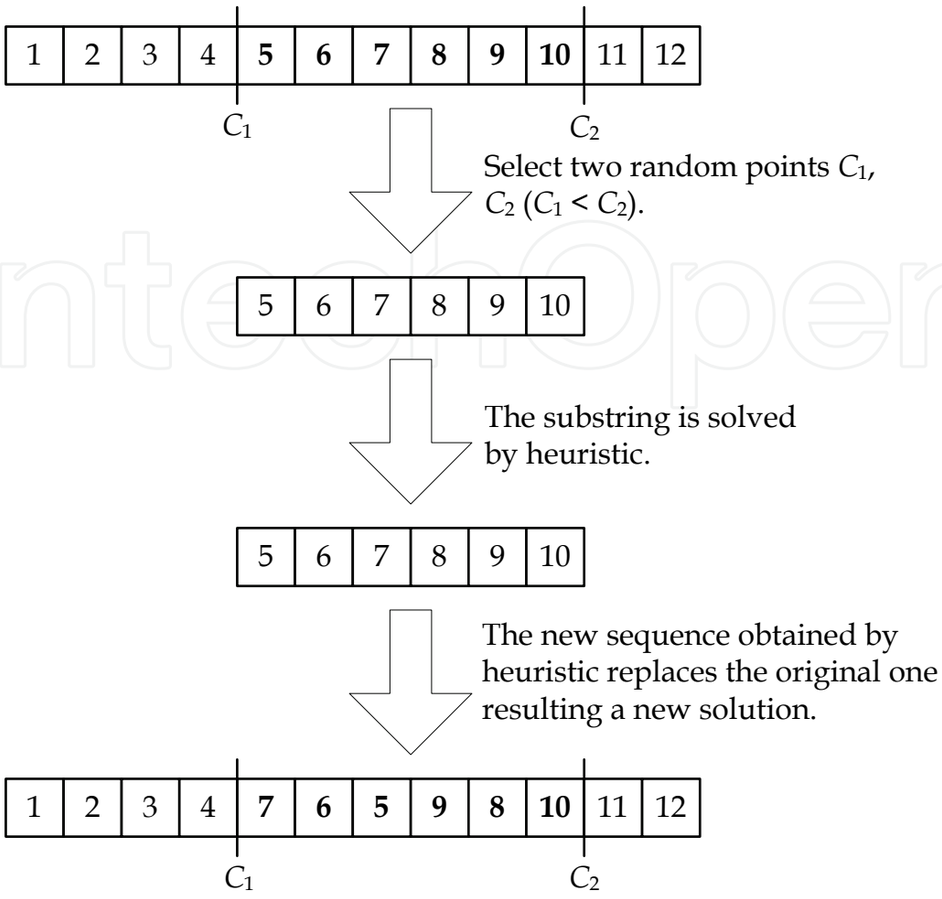


Fig. 1. A hybrid tabu search

4.1 Initial solution

In RPFS problems, NEH heuristic is used to generate an initial solution. For example, the schedule by NEH is (3, 5, 1, 4, 2) which represents the job sequence on each machine in the shop. If this sequence is changed, the processing order of jobs on each machine changes accordingly.

4.2 Neighborhood search

First, (3, 5, 1, 4, 2) is the starting point and the pair-wise exchange method is applied on it, as shown in Table 1. Next, the neighboring solution with the least makespan value and outside of the tabu list is selected as a move.

4.3 Record into tabu list

It is found that neighboring solution 2 has the minimal makespan among these neighborhood solutions (see Table 1), and this schedule is obtained by exchanging job 3 and job 1. Therefore, it is needed to check whether job 3 and job 1 are in tabu list. If they are not in tabu list, a move is made and iteration number is added by one; otherwise, the

neighboring solution with second least makespan is checked. If it is in tabu list, too, the third least one is checked. This procedure continues until none of the exchanged operations are in tabu list and a move can be made.

Number	Neighborhood solution					Makespan
1	5	3	1	4	2	340
2	1	5	3	4	2	335
3	4	5	1	3	2	347
4	2	5	1	4	3	337
5	3	1	5	4	2	348
6	3	4	1	5	2	355
7	3	2	1	4	5	340
8	3	5	4	1	2	348
9	3	5	2	4	1	358
10	3	5	1	2	4	345

Table 1. The sequence and makespan of neighborhood solutions

4.4 Record best-so-far solution

Compare the makespan (= 335) of neighboring solution 2 to that of the best-so-far solution (= 343). If it is better than the best-so-far solution, update the best-so-far solution and reset the non-improvement times to zero; otherwise, the best-so-far solution is kept and non-improvement times is added by one.

4.5 Hybrid method

The hybrid method for RPFS is described briefly. First, the new solution (3-5-1-4-2) is use to search better neighborhood solutions until stopping rule are satisfied. In the following, several iterations are omitted to describe the hybrid method directly. In RPFS example, the threshold value is also set to 3. After several iterations, the neighboring solution (3-2-1-5-4) is generated. Then, two points are selected randomly, say $C_1 = 2$, $C_2 = 4$ to define an interval (i.e., a substring). This substring is then treated as a RPFS subproblem and rescheduled by NEH heuristic. The new sequence then replaces the original substring to form a new solution, as shown in Fig. 2. Finally, we base on this new solution to search new neighborhood solutions and find a neighboring solution with makespan of 327 is better than the best-so-far solution (= 335). These neighborhood solutions based on above searching procedures repeats until the stopping rules are satisfied.

5. Computational results

The experimental environment and the meaning of each parameter are described as follows. n is the number of jobs, m is number of machines, and L is number of layers. The problem n×m×L is a RPFS problem with n jobs, m machines, L layers. The test problems are classified

into 3 categories: small problems, medium problems, and large problems. Types of small problems include 3×3×3, 4×4×4, 5×4×3, 5×5×4, 6×8×5, 7×8×4, 8×8×4, 9×7×4, 9×9×3, and 10×6×3. Types of medium problems include 11×17×5, 12×20×6, 13×19×7, 14×18×9, 15×17×6, 16×16×7, 17×15×8, 18×16×6, 19×12×10, and 20×15×8. Types of large problems include 25×25×10, 30×30×7, 40×40×6, 50×50×5, 60×60×3. The processing time of each operation for each type of problem is a random number generated from [1, 100] since the processing times of most library benchmark problems are generated in this range (Beasly, 1990).

In order to demonstrate the performance of HTS, it is compared to optimal solution obtained by integer programming (IP) for small problems. The IP model is proposed by Pan and Chen (2003) for solving RPFS scheduling problems. For medium and large problems, HTS is compared to its initial solution or to the pure TS solution. In this study, IP model is solved by ILOG CPLEX software. The programs for heuristics are coded in Visual C++ language and implemented on PC with Pentium IV 1.6 GHz.

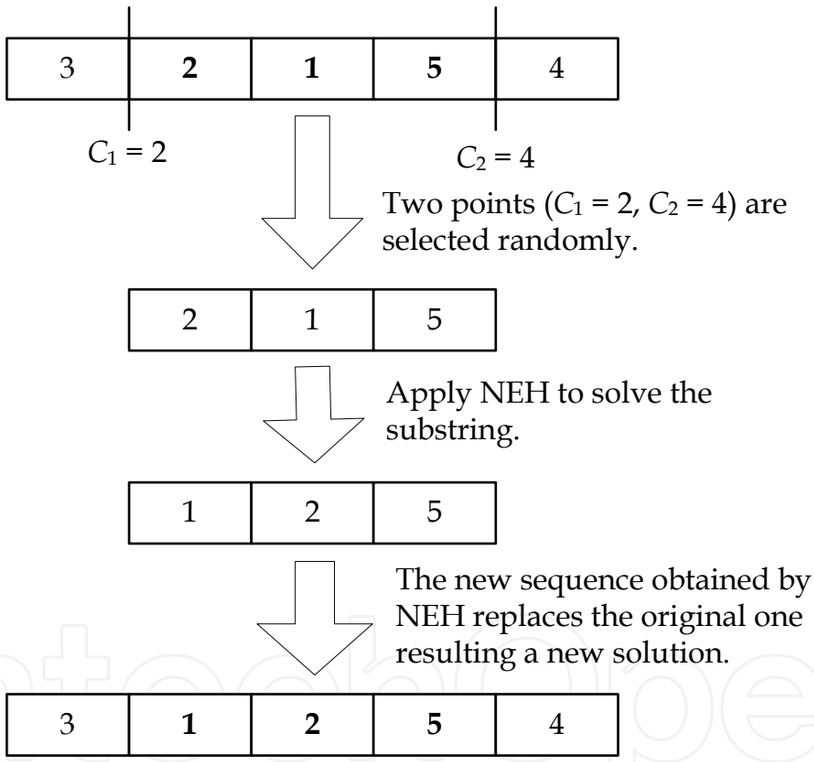


Fig. 2. The hybrid method of RPFS example

5.1 Small problems

In the experiment, ten instances are generated for each problem type and the average makespan is analyzed. For each problem type, the average makespan of HTS is compared to that of optimal makespan. The difference of these two average values is a measure of the efficiency of HTS. Similarly, HTS is also compared with its initial solution to obtain the improvement rate. The encoding scheme is based on jobs rather than on operations. The maximal number of iterations is set to 3,000 and non-improvement times are set to 1,500.

The results for the type of small problem are shown in Table 2. The solutions of HTS are also close to optimum (0.004% above optimum on the average). These results show that HTS is efficient and has good solution quality of less than 1% above optimal. Then the percentage error of HTS is defined as:

$$\text{Percentage error} = \frac{C_{\max}(\text{HTS}) - C_{\max}(\text{IP})}{C_{\max}(\text{IP})} \times 100\%$$

Where C_{max}(HTS) and C_{max}(IP) are the makepan obtained by HTS and IP, respectively.

5.2 The medium problems

For medium problems, the solutions obtained by HTS are compared to initial solutions and those obtained by pure TS, where the initial solutions are generated by heuristic NEH (Pan & Chen, 2003). The maximal iteration number is 2000 and non-improvement time is 1000. The comparison results of all medium problems are displayed in Table 3. Table 3 shows that the quality of solutions generated by HTS is 2.83% better than its initial solution obtained by NEH. Additionally, the performance of HTS is 0.57% better than pure TS.

5.3 Large problems

Large problems are tested with the same basis as those of medium problems and the types of large problems tested are shown in Table 4. The maximal iteration number is 1200 and non-improvement time is 600. The performance difference between HTS and NEH, HTS and pure TS is reported, respectively. It is shown that the solution quality of HTS is 2.53% better than its initial solutions generated by NEH. For comparison of the performance between HTS and pure TS, the efficiency of HTS is 0.81% better than that of the pure TS. It is noted that improvement rate increases as the number of jobs increases.

Types	IP	HTS	
	Time (s)	Time (s)	Avg. percentage error (%)
3×3×3	0.03	0	0
4×4×4	0.11	0.007	0.041
5×4×3	0.18	0.42	0
5×5×4	0.32	0.41	0
6×8×5	2.33	0.60	0
7×8×4	6.93	0.76	0
8×8×4	32.83	2.15	0
9×7×4	71.93	2.56	0
9×9×3	90.63	2.41	0
10×6×3	20.52	5.43	0

Table 2. Comparison of all small problems solved by IP and HTS

Types	CPU time(s)			Comparison	
	HTS	TS	NEH	The improvement rate of HTS over TS (%)	The improvement rate of HTS over NEH (%)
11×17×5	3.05	3.18	0.01	0.62	2.29
12×20×6	5.63	5.03	0.01	0.61	2.12
13×19×7	8.46	8.35	0.01	0.48	2.13
14×18×9	15.89	12.37	0.02	0.46	2.11
15×17×6	12.04	8.64	0.02	0.81	3.07
16×16×7	16.53	13.34	0.02	0.47	2.98
17×15×8	19.86	19.97	0.02	0.37	3.13
18×16×6	23.28	19.78	0.02	0.56	3.74
19×12×10	28.59	25.98	0.02	0.60	3.12
20×15×8	38.39	37.00	0.02	0.71	3.64

Table 3. The improvement results of all medium problems

Types	CPU time(s)			Comparison	
	HTS	TS	NEH	The improvement rate of HTS over TS (%)	The improvement rate of HTS over NEH (%)
25×25×10	155.22	132.67	0.02	0.78	2.53
30×30×7	242.60	179.28	0.21	0.79	2.63
40×40×6	612.24	481.94	0.31	0.81	2.43
50×50×5	1006.44	872.64	0.76	0.83	2.46
60×60×3	1195.09	1104.5	0.85	0.83	2.61

Table 4. The improvement results of all large problems

6. Conclusions and suggestions for future study

This study applies HTS to solve RPFS scheduling problems with objective to minimize makespan. In pure TS, if the solution cannot escape from local optimum, the improvement rate can hardly be increased even a great amount of computational time is spent. The proposed HTS is used to improve the efficiency of TS. The heuristic method is hybridized into pure TS to find better solution regions.

In RPFS, job-based encoding is adopted to deal with different types of problems. The results show that HTS obtains favorable solutions within reasonable time. For small problems, the percentage of HTS finding optimal solutions is near 100%. For medium problems,

comparisons are made between HTS and the initial solutions obtained by NEH. It is found that HTS improves the initial solution favorably. For large problems, HTS is superior to heuristic NEH. For the medium and large problems, HTS is compared to pure TS method. The results show that HTS is superior to pure TS. Moreover, it is found that the improvement rate of HTS over TS increases with the increase of problem size. Hence, it is clear that the incorporation of appropriate heuristic with pure TS is indeed effective.

Some future study suggestions are given as follows:

(1) A static tabu list is used in this study. A dynamic tabu list may be used in future study to investigate whether the solution quality can be improved.

(2) A thorough study of the effect of maximal iteration number and non-improvement times on solution quality may be carried out in future studies.

(3) Other exchanging method to obtain neighborhood solutions can be investigated and the techniques of experimental design may be applied to find out the best way of neighborhood search.

7. References

Adams, J., Balas, E., & Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management Science*, Vol. 34, No. 3, 391-401, ISSN: 0025-1909.

Baker, K. R., (1974). *Introduction to sequencing and scheduling*, John Wiley & Sons, ISBN: 0-471-04555-1, New York.

Beasly, J. E. (1990). OR-library: distribution test problems by electronic mail. *Journal of the Operational Research Society*, Vol. 41, No. 11, 1069-1072, ISSN: 0160-5682.

Bispo, C. F. & Tayur, S. (2001). Managing simple re-entrant flow lines: theoretical foundation and experimental results. *IIE Transactions*, Vol. 33, No. 8, 609-623, ISSN: 0740-817X.

Bowman, E. H. (1959). The scheduling-sequence problem. *Operations Research*, Vol. 7, 621-624, ISSN: 0030-364X.

Campbell, H. G., Dudek, R. A., & Smith, M. L. (1970).. *Management Science*, Vol. 16, No. 10, B630-B637, ISSN: 0025-1909.

Coffman, E. G. (1976). Computer and Job Shop Scheduling. Wiley, ISBN: 0471163198, New York.

Dell'Amico, M., & Trubian, M. (1993). Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, Vol. 41, No. 3, 231-252, ISSN: 0254-5330.

Dannenbring, D. G. (1977). An evaluation of flow shop sequencing heuristics. *Management Science*, Vol. 23, No. 11, 1174-1182, ISSN: 0025-1909.

Demirkol, E. & Uzsoy, R. (2000). Decomposition methods for re-entrant flow shops with sequence-dependent setup times. *Journal of Scheduling*, Vol. 3, No. 3, 155-177, ISSN: 1094-6136.

Drobouchevitch, I. G., & Strusevich, V. A. (1999). A heuristic algorithm for two-machine re-entrant shop scheduling. *Annals of Operations Research*, Vol. 86, 417-439, ISSN: 0254-5330.

Garey, M. R.; Johnson, D. S. & Sethi, R. (1976). The complexity of flow-shop and job-shop scheduling. *Mathematics of Operations Research*, Vol. 1, No. 2, 117-129, ISSN: 0364-765X.

Glover, F. (1989). Tabu search– Part I. *ORSA Journal on Computing*, Vol. 1, 190-206, ISSN: 0899-1499.

Glover, F., & Languna, M. (1993). Tabu Search– Modern heuristic techniques for combinatorial problems. Colin R. Reeves (Ed.), 70-150, Blackwell Scientific Publications, ISBN: 0470220791, Oxford.

Glover, F., & Languna, M. (1997). Tabu Search. Colin R. Reeves (Ed.), Blackwell Scientific Publications, ISBN: 079239965X, Oxford.

Graves, S. C.; Meal, H. C.; Stefek, D. & Zeghmi, A. H. (1983). Scheduling of re-entrant flow shops. *Journal of Operations Management*, Vol. 3, No. 4, 197-207, ISSN: 0272-6963.

Hwang, H. and Sun, J. U. (1998). Production sequencing problem with re-entrant work flows and sequence dependent setup times. *International Journal of Production Research,* Vol. 36, No. 9, 2435-2450, ISSN: 0020-7543.

Hwang, J., Kang, C. S., Ryu, K. R., Han, Y., & Choi, H. R. (2002). A hybrid of tabu search and integer programming for subway crew paring optimization. Proceedings of the Sixth IASTED International Conference on Artificial Intelligence and Soft Computing (ASC-2002), 72-77, Banff, Canada.

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with set up times included, *Naval Research Logistics Quarterly*, Vol. 1, No. 1, 61-68.

Kubiak, W.; Lou, S. X. C. & Wang, Y. (1996). Mean flow time minimization in re-entrant job-shops with a hub. *Operations Research*, Vol. 44, No. 5, 764-776, ISSN: 0030-364X.

Nawaz, M., Enscore, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine n-job flow-shop sequencing problem. *OMEGA*, Vol. 11, No. 1, 91-95, ISSN: 0305-0483.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science*, Vol. 42, No. 6, 797-813, ISSN: 0025-1909.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the permutation flow-shop problem. *European Journal of Operational Research*, Vol. 91, No. 1, 160-175, ISSN: 0377-2217.

Nowicki, E., & Smutnicki, C. (1998). Flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research*, Vol. 106, No. 2-3, 226-253, ISSN: 0377-2217.

Palmer, D. S. (1965). Sequencing jobs through a multi-stage process in the minimum total time– A quick method of obtaining a near optimum. *Operational Research Quarterly*, Vol. 16, No. 1, 101-107, ISSN: 0030-3623.

Pan, J. C. H. & Chen, J. S. (2003). Minimizing makespan in re-entrant permutation flow-shops. *Journal of the Operational Research Society*, Vol. 54, No. 6, 642-653, ISSN: 0160-5682.

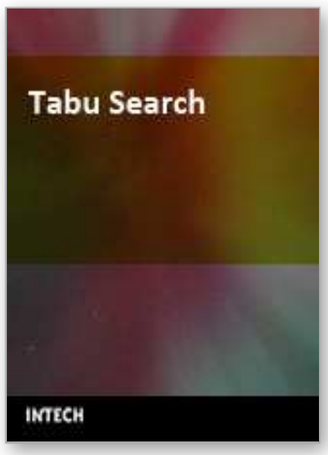
Rinnooy Kan, A. H. G. (1976). *Machine scheduling problems: classification, complexity and computations*, Martinus Nijhoff, ISBN: 90.247.1848.1, The Hague, Holland.

Taillard, E. (1989). Parallel taboo search technique for the job shop scheduling problem. Internal Report ORWP89/11, Department de Mathematiques, Ecole Polytechnique Federale de Lausanne, Lausanne.

Vargas-Villamil, F. D. & Rivera, D. E. (2001). A model predictive control approach for real-time optimization of re-entrant manufacturing lines. *Computers in Industry*, Vol. 45, No. 1, 45-57, ISSN: 0166-3615.

Wang, M. Y.; Sethi, S. P. & Van De Velde, S. L. (1997). Minimizing makespan in a class of re-entrant shops. *Operations Research*, Vol. 45, No. 5, 702-712, ISSN: 0030-364X.

Widmer, M., & Hertz, A. (1989). A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, Vol. 41, 186-193, ISSN: 0377-2217.



Tabu Search

Edited by Wassim Jaziri

ISBN 978-3-902613-34-9
Hard cover, 278 pages
Publisher I-Tech Education and Publishing
Published online 01, September, 2008
Published in print edition September, 2008

The goal of this book is to report original researches on algorithms and applications of Tabu Search to real-world problems as well as recent improvements and extensions on its concepts and algorithms. The book' Chapters identify useful new implementations and ways to integrate and apply the principles of Tabu Search, to hybrid it with others optimization methods, to prove new theoretical results, and to describe the successful application of optimization methods to real world problems. Chapters were selected after a careful review process by reviewers, based on the originality, relevance and their contribution to local search techniques and more precisely to Tabu Search.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Jen-Shiang Chen, Jason Chao-Hsien Pan and Chien-Kuang Wu (2008). Hybrid Tabu Search for Re-Entrant Permutation Flow-Shop Scheduling Problem, Tabu Search, Wassim Jaziri (Ed.), ISBN: 978-3-902613-34-9, InTech, Available from: http://www.intechopen.com/books/tabu_search/hybrid_tabu_search_for_re-entrant_permutation_flow-shop_scheduling_problem

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen