Int J Adv Manuf Technol (2000) 16:765–771

The International Journal of
Advanced
Manufacturing
Technology

A Tabu Search Algorithm for Job Shop Scheduling

S. G. Ponnambalam[1], P. Aravindan[1] and S. V. Rajesh[2]

[1]Department of Production Engineering, Regional Engineering College, Tiruchirapalli, 620 015 India; and [2]Tata Consultancy Services, Chennai

In this paper the tabu search technique is considered for solving job shop scheduling problems. The performance measure considered is makespan time. The adjacent pairwise interchange method is used to generate neighbourhoods. The results of tabu search are compared with simulated annealing and genetic algorithms. It is concluded that the performance of tabu search is comparable to that of genetic algorithm and simulated annealing. The future research directions are also discussed.

Keywords: Heuristics; Job shop scheduling; Tabu search

1. Introduction

As a context for solving sequencing and scheduling problems, the job shop model is highly complicated. Compared to the basic single-machine model, which in its own right yields complex combinatorial problems, the job shop has the added features of several machines and logical constraints. Complexity grows remarkably fast as problem size increases. For small problems, enumerative procedures such as branch and bound can be applied to obtain the optimum schedule [1].

In the case of an enumerative procedure, the effort, time, and mathematical sophistication for finding and understanding a suitable method may be judged not worth the possible return. There may be no method which is suitable for the complex constraints of the real situation. The methods may be inapplicable in practice, owing to inappropriate computing time, deceptive solution quality, or lack of robustness. As a consequence, optimisation methods may be ignored and some rules known as heuristics may be implemented. These are capable of obtaining good solutions very quickly but do not necessarily yield optimal solutions [2].

There are combinatorial optimisation problems in all sectors of management, e.g. sequencing and scheduling problems in production management, routeing and transportation, plant location, and time tabling. In recent years, much attention has been devoted to four general heuristics: simulated annealing (SA); tabu search (TS); genetic algorithm (GA); and neural network (NN). They are applicable, in particular, for solving combinatorial optimisation problems. These methods are capable of providing high-quality solutions with reasonable computational effort. Scheduling provides one of the most fruitful areas for these four modern heuristic techniques.

Genetic Algorithm (GA). A genetic algorithm is a set of procedures which, when repeated, enables solutions to be found to specific problems. In order to achieve this objective, GAs generate successive populations until a solution is obtained that yields acceptable results.

Simulated Annealing (SA). A simulated annealing algorithm is a step-by-step procedure which can be considered as an improvement of the local optimisation algorithm. The SA starts with an initial solution and a relatively high value of temperature to avoid being prematurely entrapped in a local optimum.

Tabu Search (TS). Tabu search is a metaheuristic designed for finding a near optimal solution of combinatorial optimisation problems. It consists of several elements called the move, neighbourhood, initial solution, search strategy, memory, aspiration function and stopping rules.

Effective use of resources is crucial in any manufacturing enterprise. In a majority of industrial settings, different types of job compete with one another for the available scarce resources. A schedule is a feasible resolution of the resource constraints when no operation ever occupies the same machine simultaneously. Proper scheduling of jobs is very important for the successful operation of a shop. Much research has been carried out in the scheduling area where approaches have ranged from the use of a Gantt chart to optimisation methods such as dynamic programming, integer programming and branch and bound methods. However, most scheduling problems are classified, as NP-complete and optimum seeking methods have been found to be impractical for a real-size manufacturing shop. As mentioned earlier, this leads to the development of heuristic techniques such as neighbourhood search and random sampling [3].

Correspondence and offprint requests to: Dr S. G. Ponnambalam, Department of Production Engineering, Regional Engineering College, Tiruchirapalli, 620 015 Tamil Nadu, India. E-mail: pons@rect.ernet.in

In this paper, tabu search, one of the four modern heuristic techniques, is used to solve the job shop scheduling problems with the makespan objective. In a job shop, the jobs can have quite different flows and numbers of operations. In a classical job shop, job j has m_j operations to be performed in sequence; each job/operation has been preassigned to a particular machine, which may perform that operation; a job does not visit the same machine twice.

2. Literature Review

Tabu search is still in an early stage of development, with the substantial majority of its applications occurring since 1989. The modern form of tabu search derives from Glover [4]. One of the early applications of the tabu search in scheduling is due to Widmer and Hertz [5], who developed a tabu search method for the solution of the permutation flow shop problems.

Ju-Seog Song and Tae-Eog Lee [6] considered job shops where an identical mixture of items are produced repetitively. They discussed the sequencing problem that finds the processing order at each machine that minimises the cycle time using the tabu search procedure for that problem. They characterised the neighbourhood structure that generates feasible solutions by reversing the order of two operations on a critical circuit in the associated graph.

Almeida and Centeno [7] proposed a new heuristic for the single machine early/tardy job shop scheduling problem (SMETP) that alternates search techniques performed at different neighbourhood ranges, using its own results to guide the alternation dynamically. This new composite heuristic for the SMETP combines tabu search, simulated annealing and steepest descent techniques to generate near optimal schedules.

Chu et al. [8], proposed a heuristic algorithm for job shop scheduling problem, which gradually improves a given schedule by reversing the order in which some tasks are performed on machines, which is equivalent to reversing the direction of a critical disjunctive arc.

Barnes and Chambers [9] proposed an effective tabu search approach to the job shop scheduling problem. This procedure starts from the best solution rendered by a set of 14 heuristic dispatching solutions. It then makes use of the classical disjunctive network representation of the problem and iteratively moves to another feasible solution by reversing the order of two adjacent critical path operations performed by the same machine. The concepts of historical generators and search restarts are used in conjunction with a continuous spectrum of short-term memory values to enhance the overall exploration strategy.

Vancheeswaran and Townsend [10] developed a heuristic for achieving a minimum makespan schedule for the job shop. This heuristic considers both the jobs in a given queue and those yet to arrive at any machine, and sequences the operations to be processed on all machines. It is a single-pass procedure and creates a feasible, near-optimum makespan schedule. A modification stage is then employed to further improve the schedule. This looks only at a subset of the set of feasible schedules. The sequence of operations processed on each machine is interchanged for the job with the maximum makespan so that each interchange introduces minimal disturbance to the existing schedule.

Lourenco [11] presented a computational study of different local search and large-step optimisation methods to solve the job shop scheduling problem. She revised local optimisation methods and proposed a two-phase optimisation method, known as large-step optimisation. The first phase of this new method consists of a large optimised transition in the current solution, whereas the second phase is a local search method.

James [12] used tabu search to solve the restricted, common-due-date, early/tardy machine scheduling problem. Different forms of tabu search were tested, including one based on a sequence of jobs solution space and another based on an early/tardy solution space. The conclusion was that a search which uses an early/tardy solution space with a neighbourhood scheme, which eliminates infeasible areas of the solution space, is the most efficient and effective solution method.

Marett and Wright [13] compared the performance of simulated annealing and tabu search when both are applied to a large, complex multi-objective flow-shop problem. Repeated descent, an older neighbourhood search technique, is used as a basis for comparison. The best parameters needed for the methods and their subsequent performance are compared on two further levels: as the total number of perturbations allowed increases, and as the number of objectives increases.

The various areas in which tabu search is applied are reported by Glover and Laguna [4] and shown in Table 1.

The aims of this paper are to develop an algorithm using the tabu search procedure to solve the job shop scheduling problem and to evaluate the performance of the proposed algorithm by comparison with genetic algorithm and simulated annealing.

3. Local Search Strategies

Tabu search (TS), simulated annealing (SA) and genetic algorithms (GAs) can be considered to some extent as local search strategies; this is quite clear for SA and TS, and in a more elaborate sense for GAs. Essentially, local search consists in moving from a solution to another one in its neighbourhood, according to some well-defined rules. For definiteness, the problem of minimising a function $F(x)$ on a finite set of points X, can be considered as a general statement of a combinatorial optimisation problem. A local search strategy starts from an arbitrary solution $x_1 \in X$ and at each step n, a new solution x_{n+1} is chosen in the neighbourhood $V(x_n)$ of the current solution x_n. This presupposes the definition of a neighbourhood structure on X; with each $x \in X$ is associated a subset $V(x) \subseteq X$ called the neighbourhood of x. The most common criterion for selecting the next solution x_{n+1} is to pick up the best one in the neighbourhood of x_n [2], i.e. a solution $x_{n+1} \in V(x_n)$ with

$$F(x_{n+1}) \leq F(x) \quad \forall x \in V(x_n)$$

Then, x_{n+1} becomes the next current solution, provided it is not worse than x_n, i.e. $F(x_{n+1})$. Otherwise, the search is stopped. This strategy is usually called a descent or a steepest descent strategy.

Table 1. Some applications of tabu search.

Area	Brief description
Scheduling	Employee scheduling Flow shop scheduling Job shop scheduling with tooling constraints Convoy scheduling Single machine scheduling Just-in-time scheduling Multiple-machine weighted flow time problem Flexible-resource job shop scheduling Job shop scheduling Single machine scheduling (target analysis) Resource scheduling Sequencing jobs with deadlines and set-up times
Transportation	Travelling salesman problem Vehicle routeing problem
Layout and circuit design	Quadratic assignment problem Electronic circuit design
Telecommunications	Path assignment Bandwidth packing
Graphs	Clustering Graph colouring Stable sets in large graph maximum clique problem
Probabilistic logic and expert system	Maximum satisfiability problem Probabilistic logic Probabilistic logic/expert systems
Neural networks	Learning in an associative memory Non-convex optimisation problems
Others	Multiconstraint 0–1 knapsack problem Large-scale controlled routeing General fixed-charge problem

The choice of a good neighbourhood structure is generally important for the effectiveness of the process. The main weakness of the descent algorithm is its inability to escape from local minima. This is symbolically illustrated in Fig. 1. All solutions in the neighbourhood $V(x_n)$ are worse than x_n although, further away, there exists a global minimum of F which cannot be reached under the descent rule. Simulated annealing and tabu search are local search strategies explicitly designed for avoiding such a situation. This implies temporary deterioration of the objective function.

4. Choice Criterion Considered

4.1 The Choice of the Number of Solutions in the Subneighbourhood

The neighbourhood structure used is adjacent pairwise interchange. The number of solutions in the neighbourhood is $(n - 1)$, where n is the problem size.

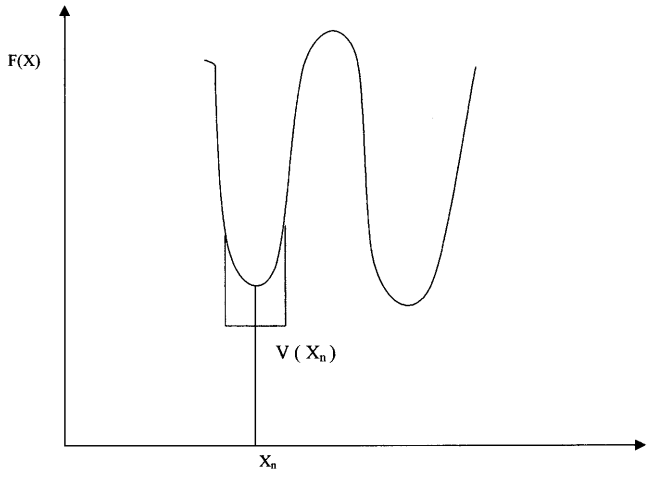

Fig. 1. Trapped in local minima.

4.2 The Choice of the Tabu Tenure Period

The tabu tenure period is the number of subsequent moves during which the last pair of solutions is to be forbidden. The rule considered for tabu tenure is a static rule in which tabu tenure $t = \sqrt{n}$, where n is a measure of the problem dimension.

4.3 The Choice of the Aspiration Criterion

An aspiration criterion is introduced in tabu search to determine when tabu restriction can be overridden, thus removing a tabu classification otherwise applied to a move.

A solution is above the current aspiration level if it is better than any solution met before. If all available moves are classified tabu, and are not rendered admissible by some other aspiration criteria, then a "least tabu" move is selected.

4.4 The Choice of a Termination Rule

The termination condition used in this paper is the number of iterations, and the number of iterations considered is 100.

4.5 Tabu status

The condition of being tabu-active or tabu-inactive is the tabu status of an attribute.

5. Tabu Search Algorithm

The working of the tabu search (short-term) algorithm [2] is explained in this section.

Notation

x_1 initial solution
x_n current solution
x^* best solution
$\bar{x}$ best solution in the neighbourhood of x_n

x_d dummy solution with $f(x_d) = \infty$

Step 1

Select an initial solution x_1 in $X(x_1 \in X)$
Initialise the best value F^* of F and the corresponding solution x^*

$F^* \leftarrow F(x_1)$
$x^* \leftarrow x_1$
F = objective function
Tabu list (TL) is empty.

Step 2

Iteration $n = 1, 2, 3, \ldots$; x_n denotes the current solution.

$\bar{F}$ = best accessible value of F met during the exploration of the subneighbourhood $V^1(X_n)$
$\bar{x}$ = solution in $V^1(x_n)$ for which $F(\bar{x}) = \bar{F}$
Assign to $\bar{F}$ to "∞" at the beginning of the each step.
For all x in $V^1(x_n)$
If $F(x) < \bar{F}$ (if the move $(x_n \leftarrow x)$ is not tabu or if the move is tabu but passes the aspiration criterion)
then $\bar{F} \leftarrow F(x)$
and $\bar{x} \leftarrow x$
Reset $x_{n+1} \leftarrow x$
If $\bar{F} < F^*$ then,
$x^* \leftarrow \bar{x}$ and $F^* \leftarrow \bar{F}$

The appropriate characteristic of the move $(x_n \rightarrow x_{n+1})$ enters the tabu list once the first entered characteristic has been removed if the list was full.

Step 3

If the stopping criterion is fulfilled, then STOP.

6. Representation of Solution Seed

An operation based representation is used to represent the solution seed. The method proposed by Gen and Cheng [14] is used in this paper. They name all operations for a job with the same symbol and then interpret them according to the order of occurrence in the sequence for a given solution seed. Each job appears in the seed exactly m times, and each repeating number does not indicate a concrete operation of a job but refers to an operation which is context-dependent. It is easy to see that any permutation of the seed always yields a feasible schedule.

Table 2. Example of a three-job three-machine problem.

Processing time Operations				Machine sequence Operations			
Job	1	2	3	Job	1	2	3
j1	3	3	2	j1	m1	m2	m3
j2	1	5	3	j2	m1	m3	m2
j3	3	2	3	j3	m2	m1	m3

Consider the three-job three-machine problem given in Table 2. Suppose a seed is given as [3 2 2 1 1 2 3 1 3], where 1 stands for job j1, 2 for job j2, and 3 for job j3. Because each job has three operations, it occurs three times in the string. For example, there are three 2s in the seed, which stands for the three operations of job j2. The first 2 corresponds to the first operation of job j2 which will be processed on machine 1, the second 2 corresponds to the second operation of job j2 which will be processed on machine 3, and the third 2 corresponds to the third operation of job j2 which will be processed on machine 2.

We can see that all operations for job j2 are given the same symbol 2 and then interpreted according to their orders of occurrence in the sequence of this seed. The corresponding relationships of the operations of jobs and processing machines are shown in Fig. 2. According to these relationships, we can obtain a corresponding machine list as [2 1 3 1 2 2 1 3 3], which is shown in Fig 2. From Fig. 2 we can see that the job processing order for machine 1 is 2-1-3, for machine 2 it is 3-1-2, and for machine 3 it is 2-1-3. We then summarise them to form a feasible schedule, as shown in Fig. 3.

7. Numerical Illustration

Instance number: orb 03 (problem considered in OR-Library)
Problem size: 10×10 (number of jobs = 10, number of machines = 10)

The algorithm presented in Section 5 is explained with an example problem given in Table 3.

Tabu tenure: 10 iterations ($\sqrt{(10 \times 10)}$)

Step 1

The initial schedule is randomly generated using an operation based representation. Makespan time for this schedule is calculated. Initially, the schedule and the makespan time are taken as best schedule and best makespan.

Initial solution (X_1):
6 0 2 0 6 7 5 5 8 6 4 8 1 9 2 0 2 1 3 7 9 1 0 5 4
9 2 1 9 6 1 1 5 3 4 0 9 5 1 2 9 8 6 4 2 3 6 4 9 1
7 0 5 2 8 6 7 2 4 9 9 3 6 1 0 8 5 9 0 3 3 5 2 3 7
6 8 0 8 6 0 2 7 5 8 8 1 7 5 7 3 7 3 3 4 8 4 4 4 7

Makespan time $F(x_1) = 1757$
F^* (best makespan) $\leftarrow F(x_1)$
x^* (best schedule) $\leftarrow x_1$

Step 2

Using pairwise exchange, the neighbourhoods $V^*(x_1)$ are generated for this schedule. For each of these schedules $[x]$ makespan times $F(x)$ are calculated and schedule $(\bar{x})$ which has minimum makespan time is taken as the initial schedule (x_{n+1}) for the next iteration. For illustration purposes, only the top five candidates which have minimum makespan time are shown in Table 4. Among these moves, move (5,8) is taken as the best move and, to avoid cycling, this move is classified as tabu. This move is forbidden for the next 10 iterations.

Table 3. The example problem (orb 03). *i*: machine number; *k*: operation sequence number; *j*: machine number; T_{ij}: processing time of job *i* on machine *j*

i / *k*	1		2		3		4		5		6		7		8		9		10	
	j	T_{ij}	*j*	T_{ij}	*j*	T_{ij}	*j*	T_{ij}	*j*	T_{ij}	*j*	T_{ij}	*j*	T_{ij}	*j*	T_{ij}	*j*	T_{ij}	*j*	T_{ij}
1	0	96	1	69	2	25	3	5	4	55	5	15	6	88	7	11	8	17	9	82
2	0	11	1	48	2	67	3	38	4	18	7	24	6	62	5	92	9	96	8	81
3	2	67	1	63	0	93	4	85	3	25	5	72	6	51	7	81	8	58	9	15
4	2	30	1	35	0	27	4	82	3	44	7	92	6	25	5	49	9	28	8	77
5	1	53	0	83	4	73	3	26	2	77	6	93	5	92	9	99	8	38	7	38
6	1	20	0	44	4	81	3	88	2	66	6	70	5	91	9	37	8	55	7	96
7	1	21	2	93	4	22	0	56	3	34	6	40	7	53	9	46	5	29	8	63
8	1	32	2	63	4	36	0	26	3	17	5	85	7	15	8	55	9	16	6	82
9	0	73	2	46	3	89	4	24	1	99	6	92	7	7	9	51	5	19	8	14
10	0	52	2	20	3	70	4	98	1	23	5	15	7	81	8	71	9	24	6	81

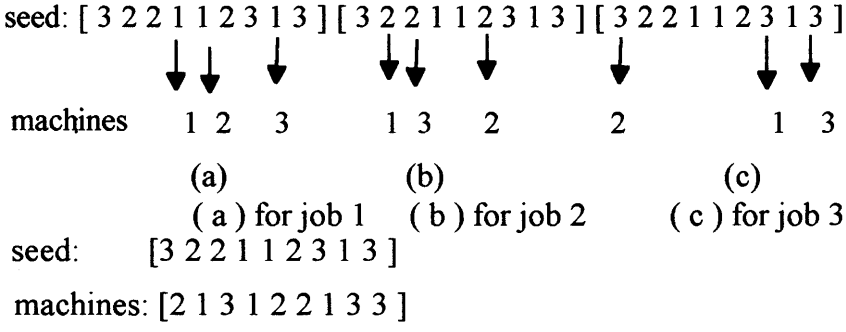

Fig. 2. Relations of the operations of jobs and processing times.

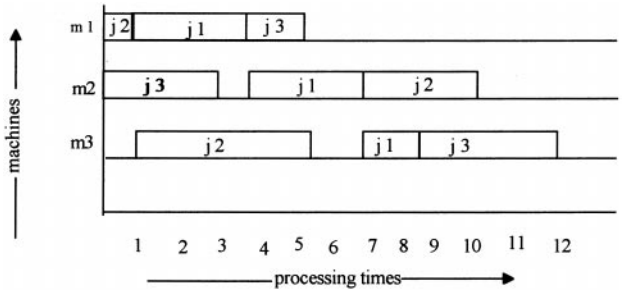

Fig. 3. Feasible schedule.

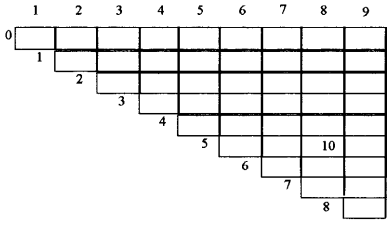

Fig. 4. Tabu structure.

$\bar{F}$ best makespan obtained in neighbourhood of $F(x_1) = 1617$

x corresponding solution seed:

6 0 2 0 6 7 5 8 5 6 4 8 1 9 2 0 2 1 3 7 9 1 0 5 4
9 2 1 9 6 1 1 5 3 4 0 9 5 1 2 9 8 6 4 2 3 6 4 9 1
7 0 5 2 8 6 7 2 4 9 9 3 6 1 0 8 5 9 0 3 3 5 2 3 7
6 8 0 8 6 0 2 7 5 8 8 1 7 5 7 3 7 3 3 4 8 4 4 4 7

Table 4. Top five candidate solutions (first iteration).

Swap		Makespan
8	5	1617*
6	8	1699
5	7	1725
9	7	1754
7	6	1757

The tabu structure for this iteration is shown in Fig. 4. In this structure each cell contains the number of iterations remaining until the corresponding moves are allowed to exchange position again.

Update

Since $F^* > \bar{F}$

F^* (best makespan) = 1617

x^* (best schedule):

6 0 2 0 6 7 5 8 5 6 4 8 1 9 2 0 2 1 3 7 9 1 0 5 4
9 2 1 9 6 1 1 5 3 4 0 9 5 1 2 9 8 6 4 2 3 6 4 9 1
7 0 5 2 8 6 7 2 4 9 9 3 6 1 0 8 5 9 0 3 3 5 2 3 7
6 8 0 8 6 0 2 7 5 8 8 1 7 5 7 3 7 3 3 4 8 4 4 4

The above schedule is taken as the initial schedule (x_{n+1}) for the next iteration. The results obtained after 50 iterations is shown in Fig. 5 and Table 5. Moves which are classified as tabu are shown in a tabu structure (Fig. 5). In this, move (6,5) is the best move, but it is classified as tabu. Because of this, (7,6) is taken as the best move.

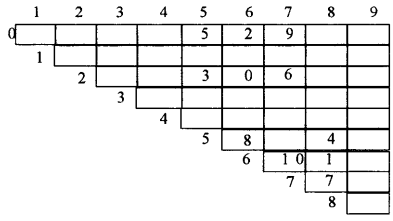

Fig. 5. Tabu structure (50 iterations).

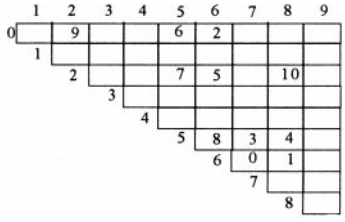

Fig. 6. Tabu structure (100 iterations).

Table 6. Top five candidate solutions (after 100 iterations). Best makespan time: 1551.

	Swap	Makespan
6	5	1551 T
0	6	1551 T
8	2	1551*
8	6	1592
1	9	1601

Table 5. Top five candidate solutions (after 50 iterations). Makespan time: 1556.

	Swap	Makespane
6	5	1556 T
7	6	1556*
8	6	1592
1	9	1601
8	0	1629

The number of iterations for termination is taken as 100. After 100 iterations, since moves (6,5) and (0,6) are classified as tabu, move (8,2) is taken as the best move with a makespan time of 1551. The tabu structure and top five moves are shown in Fig. 6 and Table 6.

The corresponding best schedule is shown below.

5 6 0 8 2 7 4 0 6 5 6 8 1 9 2 0 2 1 3 7 9 1 0 5 4
9 2 1 9 6 1 1 5 3 4 0 9 5 1 2 9 6 8 4 2 3 6 4 9 1
7 0 5 2 8 6 7 2 4 9 9 3 6 1 0 8 5 9 0 3 3 5 2 3 7
6 8 0 8 6 0 2 7 5 8 8 1 7 5 7 3 7 3 3 4 8 4 4 4 7

8. Results

The tabu search algorithm is coded in the C language. The computing system used is an HCL-HP Pentium, 16 MB RAM–100 MHz. The performance of the tabu search algorithm is compared with GA [15] and SA [16]. Twenty-five problems available in the open literature are used for the evaluation. The first 5 problems available in the OR-Library [17] and the next 20 problems used by Jawahar et al. [15] are used for the evaluation purpose. Comparison of results for these problems are shown in Table 7 and Fig. 7.

Out of 25 problems considered, for 6 problems the tabu search performs better. For the remaining problems the results of tabu search are very close to that of genetic algorithm and simulated annealing.

9. Conclusions

In this paper, a tabu search algorithm is proposed for job shop scheduling and it is implemented using the C language. The objective function considered is the minimisation of the makespan time. An adjacent pairwise interchange method is used to generate the neighbourhoods. The initial schedule is generated randomly. Schedules are represented using an operation-based representation.

The performance of the tabu search algorithm is compared with GA and SA methods. They are tested using problems published in literature. The results obtained from tabu search are encouraging and are comparable with those of GA and

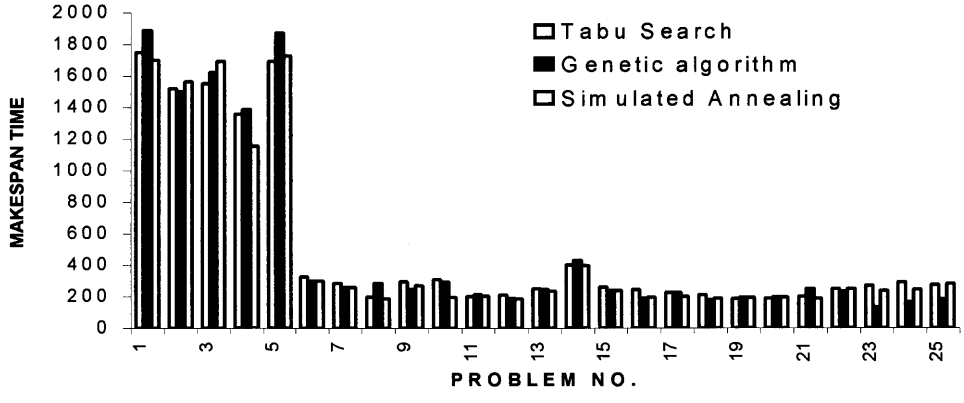

Fig. 7. Performance of TS, GA and SA.

Table 7. Comparison of TS, GA, and SA results.

Problem number	Instance	Problem size (jobs × machines)	Makespan time		
			Tabu search	Genetic algorithm	Simulated annealing
1	abz5	10 × 10	1748	1887	1699
2	abz6	10 × 10	1521	1500	1562
3	orb03	10 × 10	1551	1620	1691
4	1a12	20 × 5	1359	1387	1153
5	1a25	15 × 10	1692	1869	1728
6	jap1	10 × 6	326	299	299
7	jap2	10 × 6	285	255	255
8	iap3	10 × 6	197	185	185
9	jap4	10 × 6	293	245	270
10	jap5	10 × 6	310	295	295
11	jap6	10 × 6	201	213	203
12	jap7	10 × 6	210	190	185
13	jap8	10 × 6	250	245	235
14	jap9	10 × 6	400	430	395
15	jap10	10 × 6	260	240	240
16	jap11	10 × 6	247	189	196
17	jap12	10 × 6	223	226	200
18	jap13	10 × 6	210	180	191
19	jap14	10 × 6	185	194	194
20	jap15	10 × 6	191	198	196
21	jap16	10 × 6	200	250	190
22	jap17	10 × 6	250	232	249
23	jap18	10 × 6	271	234	239
24	jap19	10 × 6	290	265	245
25	jap20	10 × 6	272	283	279

SA. Comparisons of results are presented in the form of tables and charts.

10. Scope for Future Work

In this paper we assumed that each operation of a job is to be completed on a single machine. However, an investigation of how to extend the TS method to flexible job shop problems, where more than one machine can perform each operation, would be of practical interest. A hybrid technique can be developed by combining the features of TS, GA, and SA.

References

1. K. R. Baker, Introduction to Sequencing and Scheduling, John Wiley, New York, 1974.
2. M. Pirlot, "General local search methods", European Journal of Operational Research, 92, pp. 493–511, 1996.
3. S. K. Khator and K. Panpaliya, "Tabu search based scheduling of single batch processing machines", Working paper, University of South Florida, pp. 1–7, 1998.
4. F. Glover and M. Laguna, "Tabu search", in Colin R. Reeves (ed.), Modern Heuristic Technique for Combinatorial Problems, Blackwell, Oxford, pp. 70–150, 1993.
5. M. Widmer and A. Hertz, "A new method for the flow sequencing problem", European Journal of Operational Research, 41, pp. 186–193, 1989.
6. Ju-Seog Song and Tae-Eog Lee, "A tabu search procedure for periodic jobshop scheduling", Computers and Industrial Engineering, 30(3), pp. 433–447, 1996.
7. Maria Teresa Almedia and Mario Centeno, "A composite heuristic for the single machine early/tardy job scheduling problem", Computers and Operations Research, 25(7/8), pp. 625–635, 1998.
8. C. Chu, J. M. Proth and C. Wang, "Improving job-shop scheduling through critical pairwise exchanges", International Journal of Production Research, 36(3), pp. 683–694, 1998.
9. W. Barnes and J. B. Chambers, "Solving jobshop scheduling problem with tabu search", IIE Transactions, 27, pp. 257–263, 1995.
10. R. Vancheeswaran and M. A. Townsend, "Two-stage heuristic procedure for scheduling job shops", Journal of Manufacturing Systems, 12(4), pp. 315–325, 1995.
11. Helena Ramalhinho Lourenco, "Job-shop scheduling: Computational study of local search and large-step optimisation methods", European Journal of Operational Research, 83, pp. 347–364, 1995.
12. R. J. W. James, "Using tabu search to solve the common due date early/tardy machine scheduling problem", Computers and Operations Research, 24(3), pp. 199–208, 1997.
13. Richard Marett and Mike Wright, "A comparison of neighbourhood search techniques for multi-objective combinatorial problems", Computers and Operations Research, 23(5), pp. 465–483, 1996.
14. M. Gen and R. Cheng, "Genetic algorithm and engineering design", John Wiley, New York, 1997.
15. N. Jawahar, P. Aravindan and S. G. Ponnambalam, "A genetic algorithm for scheduling flexible manufacturing systems", International Journal of Advanced Manufacturing Technology, 14, pp. 588–607, 1998.
16. S. G. Ponnambalam, N. Jawahar and P. Aravindan, "A simulated annealing algorithm for job shop scheduling", Production Planning and Control, 10(8), pp. 767–777, 1999.
17. Internet address: www.mscmga.ms.ic.ac.uk.