International Journal of Computer Applications (0975 – 888)
Volume 48– No.5, June 2012

Integrating Genetic Algorithm, Tabu Search and Simulated Annealing For Job Shop Scheduling Proble

R. Thamilselvan
Associate Professor
Kongu Engineering College
Perundi, Erode – 638052
Tamilnadu, India

P. Balasubramanie
Professor
Kongu Engineering College
Perundurai, Erode - 638 052
Tamilnadu, India

ABSTRACT
Job Shop Scheduling Problem (JSSP) is an optimization problem in which ideal jobs are assigned to resources at particular times. In recent years many attempts have been made at the solution of JSSP using a various range of tools and techniques such as Branch and Bound and Heuristics algorithms. This paper proposed a new algorithm based on Genetic Algorithm (GA), Tabu Search (TS) and Simulated Annealing (SA) algorithms to solve JSSP. The proposed algorithm is mainly based on the genetic algorithm. The reproduction phase of the genetic algorithm uses the tabu search to generate new population. Simulated annealing algorithm is used to speed up the genetic algorithm to get the solution by applying the simulated annealing test for all the population members. The proposed algorithm used many small but important features such as chromosome representation, effective genetic operators, and restricted neighbourhood strategies. The above features are used in the hybrid algorithm to solve several bench mark problems.

Keywords
Simulated Annealing, Tabu Search, Genetic Algorithm and Job Shop Scheduling

1. INTRODUCTION
Scheduling in the manufacturing systems is one of the most important issues in the planning and operation. Many scheduling problems are difficult to solve due to complex in nature. The JSSP can be described as follows. There is a set of jobs and each job consists of set of operations. The operations have to processed uninterrupted on a given machine for a specified length of time. A schedule is an allocation of operation to time intervals on the machine. Proficient algorithms are used to solve JSSP, it will increase the production efficiency, cost reduction in the manufacturing system. JSSP is one of the most difficult NP-hard problems [1] and there is exact algorithm to solve. Due to the complexity of the problem, techniques such as branch and bound [2, 3] and dynamic programming [4, 5] are used only for the moderate problems. But most of them failed to get the solution because it required huge amount of memory and lengthy computational time. With the development of new techniques from the field of artificial intelligence, more importance has been given to metaheuristics. The tabu search [6, 7, 8] and simulated annealing [9, 10] are the type of metaheuristics and it is the construction and improvement heuristic. Genetic algorithm (GA) [11, 12, 13], particle swarm optimization (PSO) [14, 15] is the population based algorithms.

Genetic Algorithm proposed by John Holland [16] and Goldberg [17], is regarded as problem independent approach and is well suited to dealing with hard combinational problems. GAs uses the basic Darwinian mechanism of "survival of the fittest" and repeatedly utilizes the information contained in the solution population to generate new solutions with better performance. The goal of the scheduling algorithms is to find a solution that satisfies the constraints.

Tabu Search was developed by Glover [18, 19, 20]. TS is a search procedure that limits the searching and negotiates a local minimum, while keeping the history of searching in memory. According to Brucker [21], TS is an intelligent search technique that uses a memory function in order to avoid being trapped at a local minimum and hierarchically canalizes one or more local search procedure in order to search quickly the global optimality.

2. THE JOB SHOP SCHEDULING PROBLEM
The nxm Job Shop Scheduling problem is labeled by the symbol n, m, J, O, G and C_{max}. It can be described by the finite set of n jobs $J=\{J_0, J_1, J_2, J_3, ...J_n, J_{n+1}\}$ (the operation 0 and $n+1$ has duration and represents the initial an final operations), each job consist of a chain of operations $O=\{O_1,O_2,O_3,....O_m\}$, each operation has processing time $\{\lambda_{i1}, \lambda_{i2}, \lambda_{i3}, ... \lambda_{im}\}$, finite set of m machines $M=\{M_1, M_2, M_3....M_m\}$, G is the matrix that represents the processing order of job in different machines and C_{max} is the makespan that represents the completion time of the last operation in job shop. On O define A, a binary relation representing precedence between operations. If $(v,u) \in A$ then u has to be performed before v. A schedule is a function $S: O \rightarrow IN \cup \{O\}$ that for each operation u defines a start time $S(u)$. A schedule S is feasible if

$$\forall u \in O: \quad S(u) \geq 0 \qquad (1)$$
$$\forall u, v \in O, (u,v) \in A: S(u) + \lambda(u) \leq S(v) \qquad (2)$$
$$\forall u, v \in O, u \neq v, M(u) = M(v):$$
$$S(u) + \lambda(u) \leq S(v) \text{ or } S(v) + \lambda(v) \leq S(u) \qquad (3)$$

The length of a schedule S is

$$len(S) = max_{v \in O} (S(u) + \lambda(u)). \qquad (4)$$

The goal is to find an optimal schedule, a feasible schedule of minimum length, *min(len(S))*.

An instance of the JSS problem can be represented by means of a disjunctive graph G=(O, A, E). Here *O* is the vertex which represents the operations and *A* represents the conjunctive arc which represents the priority between the operations and the edge in $E = \{(u,v) | u, v \in O, u \neq v, M(u) = M(v)\}$ represent the machine capacity constraints. Each vertex *u* has a weight, equal to the processing time *λ(u)*. Let us consider the bench mark problem of the JSSP with four jobs, each has three different operations and there are three different machines. Operation sequence, machine assignment and processing time are given in Table 1.

Based on the above bench mark problem, we create a matrix *G*, in which rows represent the processing order of operation and the column represents the processing order of jobs. Also we create a matrix *P*, in which row *i* represents the processing time of J_i for different operations.

Table 1. Processing Time and Sequence for 4X3 problem instance

Job	Operation Number and Processing Sequence	Machine Assigned	Processing Time
Start Operation	0	--	0
J_1	O_{11}	M_1	2
	O_{12}	M_2	3
	O_{13}	M_3	4
J_2	O_{21}	M_3	4
	O_{22}	M_2	4
	O_{23}	M_1	1
J_3	O_{31}	M_2	2
	O_{32}	M_3	2
	O_{33}	M_1	3
J_4	O_{41}	M_1	3
	O_{42}	M_3	3
	O_{43}	M_2	1
End Operation	0	--	0

$$G = \begin{bmatrix} M_1 & M_2 & M_3 \\ M_3 & M_2 & M_1 \\ M_2 & M_3 & M_1 \\ M_1 & M_3 & M_2 \end{bmatrix} \qquad P = \begin{bmatrix} 2 & 3 & 4 \\ 4 & 4 & 1 \\ 2 & 2 & 3 \\ 3 & 3 & 1 \end{bmatrix}$$

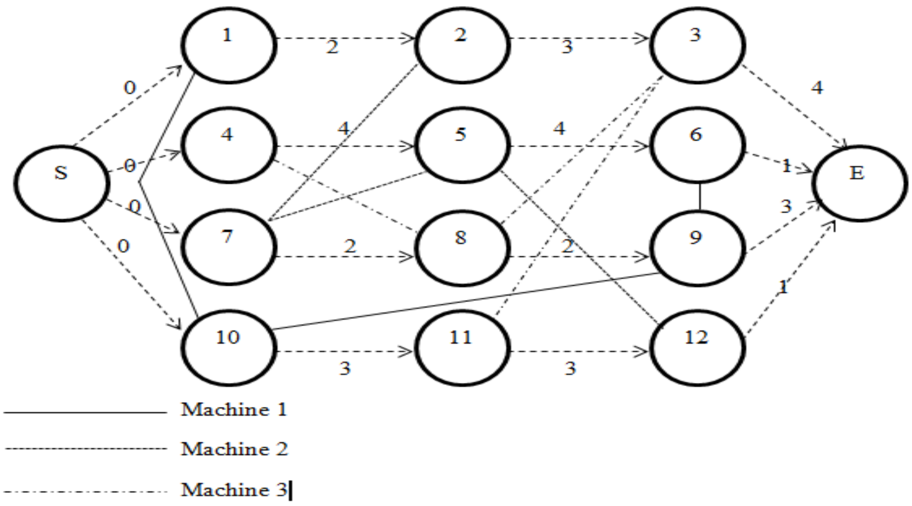

Figure 1. Illustration of disjunctive graph

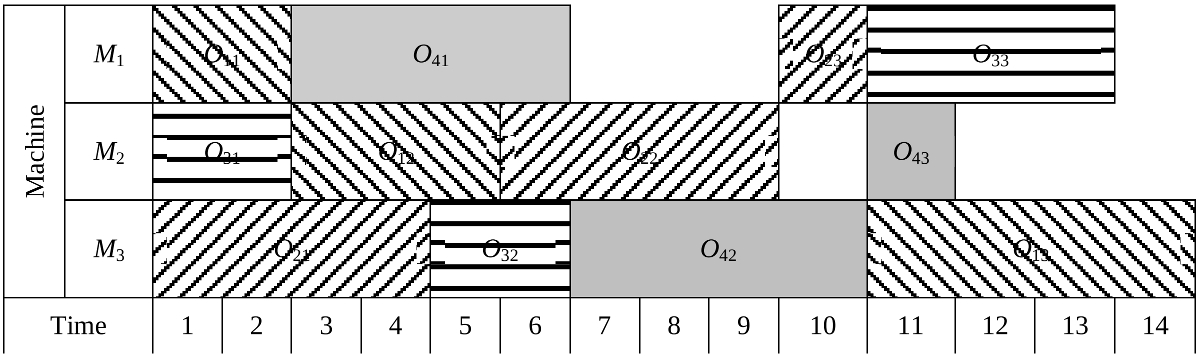

Figure 2. A Schedule of Gantt Chart for 4X3 problem Instance

The processing time of operation i on machine j is represented by O_{ij}. Let λ_{ij} be the processing time of O_{ij} in the relation $O_{ij} \rightarrow O_{ij}$. C_{ij} represents the completion of the operation O_{ij}. So that the value $C_{ij} = C_{ik} + \lambda_{ij}$ represents the completion time of O_{ij}. The main objective is to minimize of C_{max}. It can be calculated as

$$C_{max} = \max_{\text{all } O_{ij} \in O} (C_{ij}) \qquad (5)$$

The distinctive graph of the above bench mark job scheduling problem is shown in Figure 1, in which vertices represents the operation. Precedence among the operation of the same job is represented by Conjunctive arc, which are doted directed lines. Precedence among the operation of different job is represented by Disjunctive arc, which are undirected solid lines. Two additional vertices *S* and *E* represent the start and end of the schedule.

The Gantt Chart of the above bench mark job scheduling problem is shown in Figure 2. Gantt Chart is the simple graphical representation technique for job scheduling. It simply represents a graphical chart for display schedule; evaluate makespan, idle time, waiting time and machine utilization etc.

3. LITERATURE REVIEW

There are many local search algorithms have been proposed by various researchers. Local search algorithms such as Genetic Algorithms (GA) [22-35], Tabu Search (TS) [17, 19,25, 31, 36, 37], ant optimization and genetic local search (GLS) [39, 41,42, 43], scatter search and path relinking (SS and PR) and Simulated Annealing (SA). The majority of the GA methods gave a poor result due to the difficulty in crossover operation and schedule representation. TS algorithms are able to generate good schedule with in the reasonable computing time. TS algorithm has to maintain many parameters and these parameters can carefully adjusted for each problem. It is therefore apparent that if the current obstacles within job shop scheduling problems are to be overcome, hybrid approaches are worth considering.

There are many metaheuristic algorithms has been integrated to improve the solution of JSSP Guohui Zhang et.al [28, 44, 45], Wang and Zheng (GA and SA); Park et al. (parallel GA (PGA)). Hybridization of the meta-heuristic algorithms improves the performance of the JSSP. But it requires huge computing time. And there is no proper method to hybrid the algorithms; hence there is a need for exploring various combinations of search techniques. There are number of algorithms proposed with the combination of GA and TS. Meeran and Morshed [46] have used GA as the base search mechanism and TS to improve their search. They have measured the effectiveness of hybrid GA and TS which is called GTA against GA and TS. González et al.[47] presented a hybrid GA and TS system as in the case of Meeran and Morshed [46], however Gonzalez et.al proposed method is for the job shop scheduling problem with set-up times. Although they have obtained some very good results, but they have tested only the limited number of bench mark problems. Thamilselvan et.al [48, 49] has used the GA with TS and GA with parallel SA for JSSP. Here GA is used as a base algorithm and other two algorithms are used to improve the performance of the algorithm. Both of the algorithms are very efficient for the small size problems. The system being presented here is tested on a substantial number of bench mark problems including hard instances from FT, LA, ABZ and ORB, attaining optimum solutions.

4. PROPOSED ALGORITHM

4.1. Hybridization of GA, TS and SA (HGATSSA)

The proposed algorithm hybrids the important features of genetic algorithm, tabu search and simulated annealing. The proposed hybrid algorithm is implemented on JSSP. Genetic algorithm integrates the TS algorithm in the reproduction phase to generate a new schedule. To escape the local minimum and to prevent the early convergence of the GA, insert the new members in GA. Simulated annealing is used to improve the convergence of the GA testing each scheduling members after each generation.

The proposed algorithm runs on a group of networked machines. One machine act as a coordinating machine and others are the client machines. Initially GA generates a number of initial solutions from coordinating machine for distributing among *n* client machines. GA uses the Unordered Subsequence Exchange Crossover (USXX) for generating initial solutions. The *n* machines in the network run the SA and TS algorithms by using different initial solutions. After a fixed number of iterations the best solution is selected. Each machine in the network can exchange the partial solutions after a fixed number of iterations. Client machine in the network use the TS algorithm to generate a neighborhood of the initial solution and SA is used to improve the convergence of the solution.

Procedure: HGATSSA()

Step 1: Initialize the parameter of GA, SA and TS.

n (number of client machines); $g_{n=}$ 1(iteration number), t_i = m (number of iterations)

Step 2: Generate a n number of initial schedule S[i] $(i=1..n)$ using GA.

Step 3: Compute the cost $C_{S[i]}$ of initial schedule S[i].

Step 4: If the stopping criterion is satisfied. Stop the process.

Step 5: Distribute each initial schedule S[i] to the client machines.

Step 6: Each client node use the TS to generate a neighbor S[j] of S[i].

Step 7: Apply the USXX to the current schedule to complete the new set of schedules.

Step 8: Apply the mutation operator to the new schedules.

Step7: Calculate the new temperature of the SA algorithm cooling schedule. Apply the SA test to accept or reject the members of the new population (one by one) according to the SA current solution.

Step 8: Calculate the objective function of new schedule S[j].

Step 9: if (g_n<t_i) go to Step 4 otherwise go to Step 6.

Step 10: Coordinator node receive the current solution from each client machine.

Step 11: Select the best schedule among the set of current solutions.

Step 12: Go to step 3.

Step 13. Stop.

Stopping Criteria: There are many stopping criteria for job scheduling. In this proposed algorithm, we stop the search if one of the following conditions is satisfied.

- The number of iterations performed since the best solution last changed is greater than a prespecified maximum number of iterations, or
- Maximum allowable number of iterations (generations) is reached.

Schedule Representation: The main idea is how to represent the jobs in terms of sequence. In the relationship between the job scheduling and the chromosomes to represent the schedule. So that we can use the GA to find better job scheduling. For the above 4X3 job shop scheduling the chromosome such as [3 4 1 2 1 4 3 4 1 2 3 2] may be formed and then change the order for the better schedule. In the given chromosome the genes "1" stands for J_1, "2" stands for J_2 and so on. The order of the operation corresponds to the relative position of the gene. For example the first gene "3" stands for first operation of J_3, seventh gene "3" stands for the second operation of J_3, second gene "4" stands for first operation of J_4 and so on. The above scheduling chromosome is also represented as $[O_{31}, O_{41}, O_{11}, O_{21}, O_{12}, O_{42}, O_{32}, O_{43}, O_{13}, O_{22}, O_{33}, O_{23}]$. O_{ij} stands for the j^{th} operation of the job Ji. For example O_{31} stands for the first operation of J_3.

Reproduction strategies: The crossover operator involves the swapping of genetic material (bit-values) between the two parent strings. Two parents produce two offspring. There is a chance that the chromosomes of the two parents are copied unmodified as offspring. There is a chance that the chromosomes of the two parents are randomly recombined (crossover) to form offspring. Generally the chance of crossover is between 0.6 and 1.0 [6]. The following sections propose the new crossover algorithms for job shop scheduling.

The second genetic operator, mutation, can help GA to get a better solution in a faster time. In this model, relocation is used as a key mechanism for mutation. Operations of a particular job that is chosen randomly are shifted to the left or to the right of the string. Hence the mutation can introduce diversity without disturbing the sequence of jobs operations. When applying mutation one has to be aware that if the diversity of the population is not sufficiently maintained, early convergence could occur and the crossover cannot work well.

4.2. TS implementation of the proposed algorithm

In the proposed algorithm TS is used to generate new neighbors to randomly selected members of the GA populations. TS algorithm is generally simple for JSSP. The algorithm begins with initial solution and stored it as the current seed and the best solution. The neighbors of the current schedule are produced by neighborhood algorithm. They are evaluated for an objective function and a candidate which is not in tabu list and this is selected as a new seed solution. This selection is added to tabu list and this is compared with current best solution. If it is better, it is stored as a best solution. Iterations are repeated until the stopping criteria are satisfied. The following is the TS part of the proposed algorithm.

Procedure: TS(JSSP)
Initialize the parameter of TS.
S (schedule); $N(S)$ (neighbor of schedule S);S[i] (initial schedule); TL (tabu list); B_c (Best Cost); B_s (Best schedule
$S \leftarrow S$[i]
$B_C \leftarrow C_{S[i]}$
$B_S \leftarrow S$
$TS \leftarrow \emptyset$
Do $N(S) \leftarrow \{S[j] \in N(S) | Move(S, S[j] \neq TL\}$
if $N(S) \neq \emptyset$
then $S' \leftarrow x \in N(S) | \forall y \in N(S) C_x \leq C_y$
Update the tabu list for S'
$if (C_{move\,(S,\ S')}) < B_c\ then$
$Bs \leftarrow S'$
$B_C \leftarrow C_{S'}$
$S \leftarrow S'$
*Return B*s

Aspiration Criteria: Different forms of aspiration criteria are used in the literature. The one we used in this work is to override the tabu status if the current solution associated with tabu status has a better objective function than the one obtained before, for the same move.

Variable tabu list size: The basic role of the tabu list is to prevent cycling. The fixed length tabu cannot prevent cycling. We can observe that if the length of the list is too short, cycling cannot be prevented, and long-size tabu creates many restrictions so as to increase the mean value of the visited solutions. An effective way of removing this difficulty is to use a tabu list with variable size according to the current iteration number. The length of the tabu list is initially assigned according to the size of the problem and it will be decreased and increased during the construction of the solution so as to achieve better exploration of the search space.

4.3. SA implementation of the proposed algorithm

Simulated annealing gives new chances to commence new valuable hill climbing processes in which the considered particular solution may have chances to change to better situation. Therefore, the more time to see a particular solution for SA, the better to reach global optimum. SA algorithm generates an initial solution randomly. A neighbor of this solution is then generated by a suitable mechanism and the change in the cost function is calculated. If a decrease in the cost function is obtained, the current solution is replaced by the generated neighbour. If the cost function fun of the neighbour is greater, the newly generated neighbour replaces the current solution with an acceptance probability function given in equation (6)

$$P(d,T) = \exp(-\frac{d}{T}) \quad (6)$$

Where $d = C_{S[j]} - C_{S[i]}$

Procedure: SA(JSSP)
Input:
T: Temperature; *T*s : Starting temperature; *T*e : Ending temperature; *N* :Number of iteration.
Begin
generate initial schedule *S*[i] .
compute the cost $C_{S[i]}$ of initial schedule *S*[i].
*n=1, T=T*s.
while *T<T*e
while *n<N*
select neighbourhood *S*[j] of *S*[i].
compute the cost $C_{S[j]}$ of the new schedule *S*[j].
compute $= C_{S[j]} - C_{S[i]}$.
if$d \leq 0$ then
S[i]=*S*[j].
$C_{S[i]}=C_{S[j]}$.
else
generate a random variable $R \sim (0,1)$.
if$\exp(-d/T) > R$
S[i]=*S*[j].
$C_{S[i]}=C_{S[j]}$.
end if
end if
n=n+1.
end while
*T= T*0.995.*
end while
if $C_{S[i]}<B_c$
$B_c=C_{S[i]}$.
$B_s=S[i]$.
end if
End

5. RESULTS AND DISCUSSIONS

The efficiency of the proposed algorithm is tested with standard bench marks problems of Lawrence instances from LA30 to LA40 ,Storer *et al.* instances SWV11-SWV20 and Yamada and Nakano instances from YN01-YN04. The output of this algorithm is compared against the Genetic Algoritm, parallel simulated annealing and hybrid algorithm of Genetic algorithm with parallel simulated annealing. Twenty five bench mark problems were tested with proposed algorithm and other algorithms. Table 2 shows that the proposed algorithm produces better results than the other algorithm. Several measures, which gain some statistics relating to implementation of these methods, are created. They are the mean relative improvement (MRI%), the mean relative error (MRE%) shown in equation (7) and (8) respectively.

$$MRI\% = \frac{(MS_C - MS_{HGATSSA}\ )}{(MS_C)} X100 \quad (7)$$

$$MRE\% = \frac{(MS_C - MS_O)}{(MS_O)} X100 \quad (8)$$

Where MS_C is the makespan of the algorithm being compared to, $MS_{HGATSSA}$ is the makespan of the proposed algorithm, MS_O is the optimal makespan of the given problem.

Table 2. Makespan value comparison

Algorithm	No. of Problems reached optimal makespan
GA	3
PSA	3
HGAPSA	12
HGATSSA	23

Table 3 shows comparison of makespan value produced from different algorithms for problem instances LA30-LA40 (Lawrence, 1984) Column 1 specifies the problem instances, Column 2 specifies the number of jobs, Column 3 shows the number of machines, Column 4 specify the optimal value for each problem. Column 5, 6, 7 and 8 specify results from GA, PSA, HGATSSA and HGAPSA respectively. It shows that the proposed hybrid algorithm has succeeded in getting the optimal solutions for all the problems. The average makespan value of the proposed algorithm is comparetively lower than the other algorithms.

Table 4 shows comparison of relative error and relative improvement of different algorithms for problem instances LA30-LA40 (Lawrence, 1984). The relative error for all the problem instances becomes 0 for the proposed algorithm, but other algirhtms there is a relative error value that shows that the problem does not reach the optimal makespan. The comparision average markspan and relative error are also shown in Figure 3 and 4 respectively. The relative improvement is also compared with other algorithms. There is a 0.13% improvement compared to HGAPSA, 0.95% imporvement compare to PSA and 2.09% improvement compare to genetic algorithm.

Table 3. Results for instances by Lawrence (1984)

Problem Name	Problem Size		Makespan				
	Jobs (n)	Machines (m)	Optimal	GA	PSA	HGATSSA	HGAPSA
LA30	20	10	1355	1398	1360	1355	1355
LA31	30	10	1784	1829	1800	1784	1790
LA32	30	10	1850	1877	1875	1850	1860
LA33	30	10	1719	1820	1740	1719	1719
LA34	30	10	1721	1810	1742	1721	1725
LA35	30	10	1888	1950	1953	1888	1895
LA36	15	15	1279	1279	1285	1279	1279
LA37	15	15	1408	1441	1423	1408	1408
LA38	15	15	1219	1220	1219	1219	1219
LA39	15	15	1246	1246	1250	1246	1246
LA40	15	15	1241	1241	1245	1241	1241
Average			**1519.09**	**1555.55**	**1535.64**	**1519.09**	**1521.55**

Table 4. Results for instances by Lawrence (1984)

Problem Name	Problem Size		Relative Error				Relative Improvement (%)		
	Jobs (n)	Machines (m)	GA	PSA	HGATSSA	HGAPSA	With GA	With PSA	With HGAPSA
LA30	20	10	3.17	0.37	0.00	0.00	3.08	0.37	0.00
LA31	30	10	2.52	0.90	0.00	0.34	2.46	0.89	0.34
LA32	30	10	1.46	1.35	0.00	0.54	1.44	1.33	0.54
LA33	30	10	5.88	1.22	0.00	0.00	5.55	1.21	0.00
LA34	30	10	5.17	1.22	0.00	0.23	4.92	1.21	0.23
LA35	30	10	3.28	3.44	0.00	0.37	3.18	3.33	0.37
LA36	15	15	0.00	0.47	0.00	0.00	0.00	0.47	0.00
LA37	15	15	2.34	1.07	0.00	0.00	2.29	1.05	0.00
LA38	15	15	0.08	0.00	0.00	0.00	0.08	0.00	0.00
LA39	15	15	0.00	0.32	0.00	0.00	0.00	0.32	0.00
LA40	15	15	0.00	0.32	0.00	0.00	0.00	0.32	0.00
Average			**2.17**	**0.97**	**0.00**	**0.13**	**2.09**	**0.95**	**0.13**

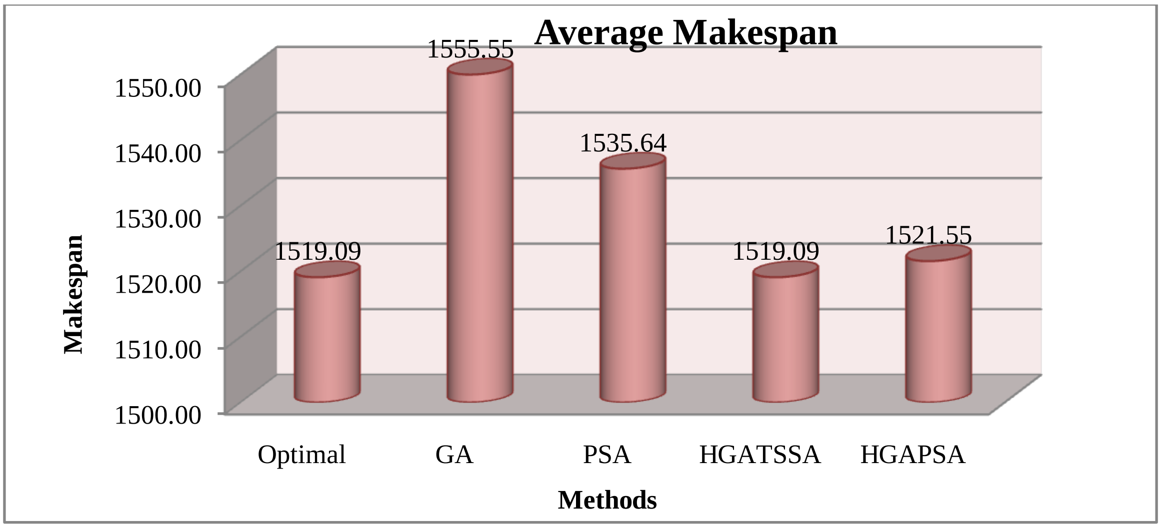

Figure 3. Average Makespan values obtained by Different algorithms for LA30-LA40

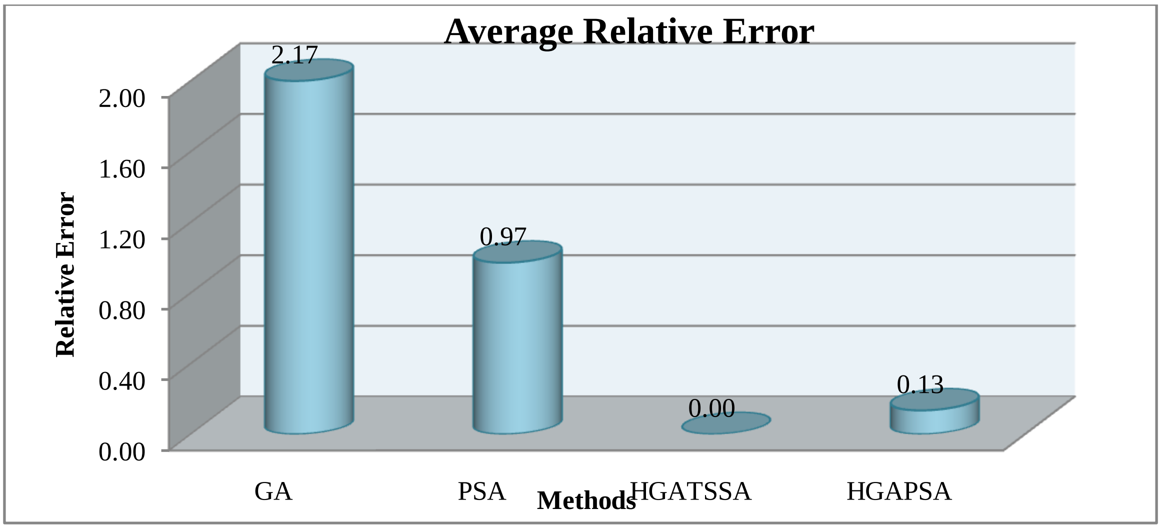

Figure 4. Average Relative Error obtained by Different algorithms for LA30-LA40

Table 5. Results for instances by Storer, Wu and Vaccari (1992)

Problem Name	Problem Size		Makespan					
	Jobs (n)	Machines (m)	Optimal		GA	PSA	HGATSSA	HGAPSA
			UB	LB				
SWV11	50	10	2991	2983	3200	3012	2983	3048
SWV12	50	10	3003	2972	3250	3120	2972	3012
SWV13	50	10	3104		3754	3250	3104	3108
SWV14	50	10	2968		3487	3212	2968	2968
SWV15	50	10	2904	2885	4235	3225	2885	2904
SWV16	50	10	2924		3547	3332	2950	3025
SWV17	50	10	2794		3269	3002	2794	2800
SWV18	50	10	2852		3156	2962	2860	2875
SWV19	50	10	2843		3169	2930	2843	2850
SWV20	50	10	2823		3231	2963	2823	2823
Average			**2920.60**	**2946.67**	**3429.80**	**3100.80**	**2918.20**	**2941.30**

Table 5 and Table 7 shows comparison of makespan value produced from different algorithms for problem instances SWV11-SWV20 and YN01-YN04 respectively. Column 1 specifies the problem instances, Column 2 specifies the number of jobs, Column 3 shows the number of machines, Column 4 specify the optimal value for each problem. Column 5, 6, 7 and 8 specify results from GA, PSA, HGATSSA and HGAPSA respectively. It shows that the proposed hybrid algorithm has succeeded in getting the optimal solutions for all the problems. The average makespan value of the proposed algorithm is comparetively lower than the other algorithms.

Table 6 shows comparison of relative error and relative improvement of different algorithms for problem instances SWV11-SWV20. There are 10 bench mark problems were testing and 9 problems reached the optimal makespan using proposed algorithm. The average relative error is also very less for the proposed algorithm. The comparision average markspan and relative error are also shown in Figure 5 and 6 respectively. The relative improvement is also compared with other algorithms. There is a 0.77% improvement compared to HGAPSA, 5.79% imporvement compare to PSA and 14.31% improvement compare to genetic algorithm.

Table 6. Results for instances by Storer, Wu and Vaccari (1992)

Problem Name	Problem Size		Makespan Optimal		Relative Error				Relative Improvement (%)		
	Jobs (n)	Machines (m)	UB	LB	GA	PSA	HGATSSA	HGAPSA	With GA	With PSA	With HGAPSA
SWV11	50	10	2991	2983	6.99	0.70	0.00	1.91	6.78	0.96	2.13
SWV12	50	10	3003	2972	8.23	3.90	0.00	0.30	8.55	4.74	1.33
SWV13	50	10	3104		20.94	4.70	0.00	0.13	17.31	4.49	0.13
SWV14	50	10	2968		17.49	8.22	0.00	0.00	14.88	7.60	0.00
SWV15	50	10	2904	2885	45.83	11.05	0.00	0.65	31.88	10.54	0.65
SWV16	50	10	2924		21.31	13.95	0.89	3.45	16.83	11.46	2.48
SWV17	50	10	2794		17.00	7.44	0.00	0.21	14.53	6.93	0.21
SWV18	50	10	2852		10.66	3.86	0.28	0.81	9.38	3.44	0.52
SWV19	50	10	2843		11.47	3.06	0.00	0.25	10.29	2.97	0.25
SWV20	50	10	2823		14.45	4.96	0.00	0.00	12.63	4.72	0.00
Average			**2920.60**	**2946.67**	**17.44**	**6.19**	**0.12**	**0.77**	**14.31**	**5.79**	**0.77**

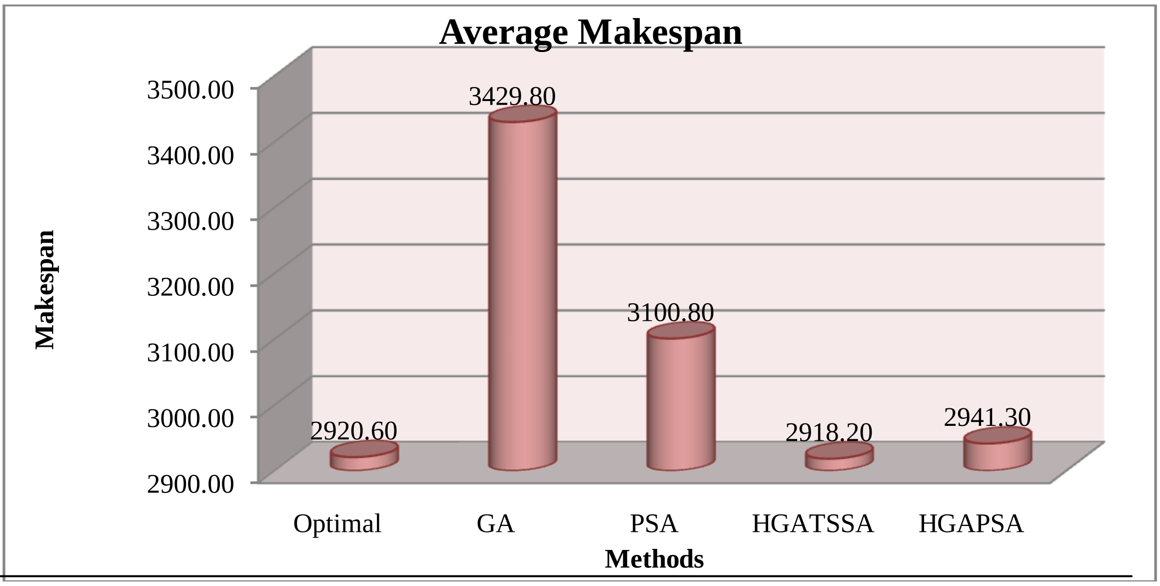

Figure 5. Average Makespan values obtaind by Different algorithms for SWV11-SWV20

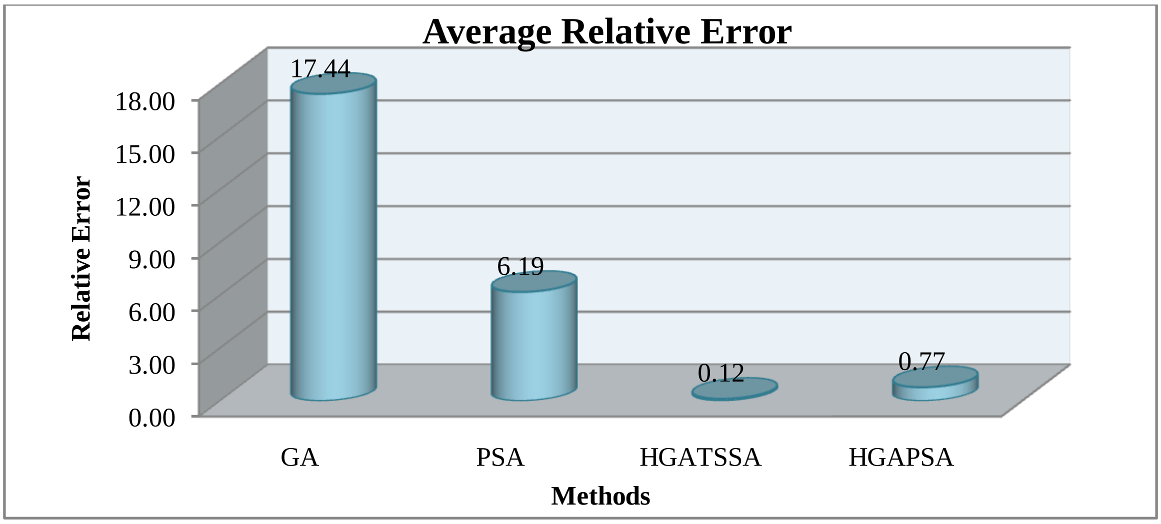

Figure 6. Average Relative error obtained by Different algorithms for SWV11-SWV20

Table 7. Results for instances by Yamada and Nakano (1992)

Problem Name	Problem Size		Makespan					
	Jobs (n)	Machines (m)	Optimal		GA	PSA	HGAPSA	HGATSSA
			UB	LB				
YN01	20	20	888	826	890	888	888	826
YN02	20	20	909	861	910	909	909	861
YN03	20	20	893	827	924	900	893	827
YN04	20	20	968	918	1098	1012	942	918
Average			**914.50**	**858.00**	**955.50**	**927.25**	**908.00**	**858.00**

Table 8. Results for instances by Yamada and Nakano (1992)

Problem Name	Problem Size		Optimal		Relative Error				Relative Improvement		
	Jobs (n)	Machines (m)	UB	LB	GA	PSA	HGAPSA	HGATSSA	With GA	With PSA	With HGAPSA
YN01	20	20	888	826	0.23	0.00	0.00	0.00	7.19	6.98	6.98
YN02	20	20	909	861	0.11	0.00	0.00	0.00	5.38	5.28	5.28
YN03	20	20	893	827	3.47	0.78	0.00	0.00	10.50	8.11	7.39
YN04	20	20	968	918	13.43	4.55	2.48	0.00	16.39	9.29	2.55
Average			**914.50**	**858.00**	**4.31**	**1.33**	**0.62**	**0.00**	**9.87**	**7.42**	**5.55**

Table 8 shows comparison of relative error and relative improvement of different algorithms for problem instances YN01-YN04. The result in the table shows that the proposed algorithm produced better result compare to the other algorithms. The average relative error is also very less for the proposed algorithm. The comparision average markspan and relative error are also shown in Figure 7 and 8 respectively. The relative improvement is also compared with other algorithms. There is a 5.55% improvement compared to HGAPSA, 7.42% imporvement compare to

PSA and 9.87% improvement compare to genetic algorithm.

Typical runs of problem instances LA30 and SWV15 are illustrated in Figure 9 and 10 respectively by the GA, PSA, HGAPSA and HGATSSA. The graph shows that the proposed HGATSSA reach the optimal solution faster than other two methods. In the graph x axis represnt the number of iterations and y axis represent the makespan value. For both the problems, proposed algorithm takes less number of iterations to reach the optimal value.

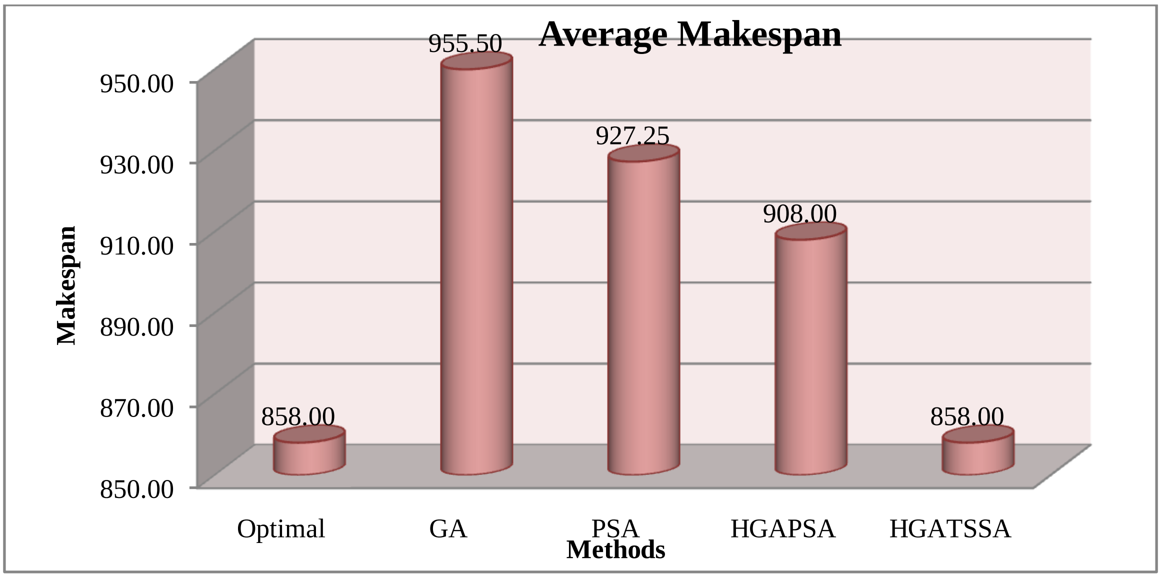

Figure. 7. Average Makespan values obtained by Different algorithms for YN01-YN04

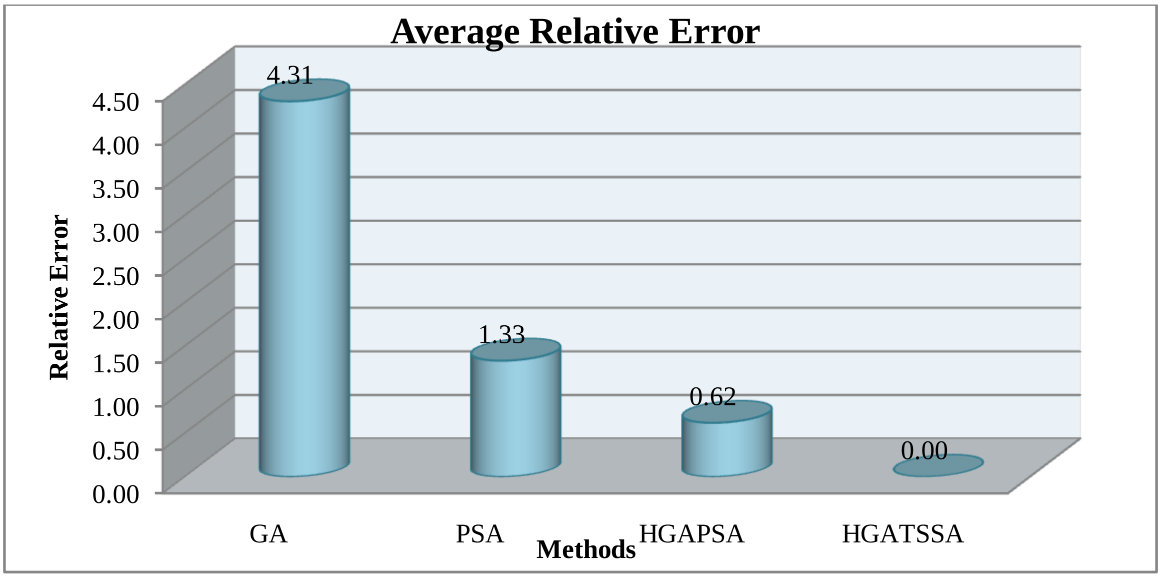

Figure 8. Average Relative error obtained by Different algorithms for YN01-YN04

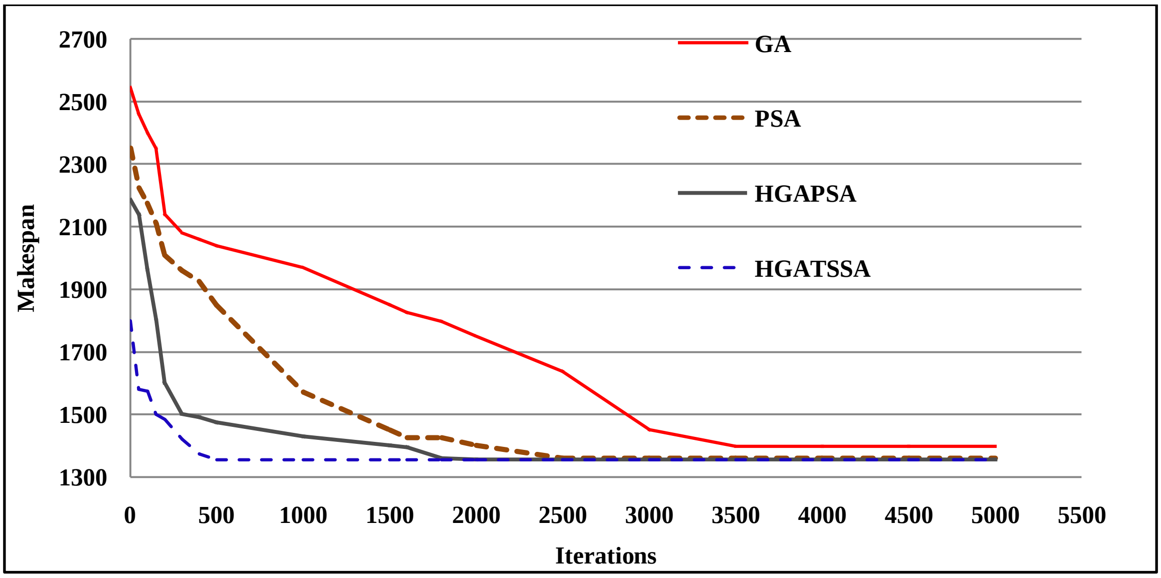

Figure 9. The time evolutions of makespans for the LA30 (20 jobs and 10 machines)

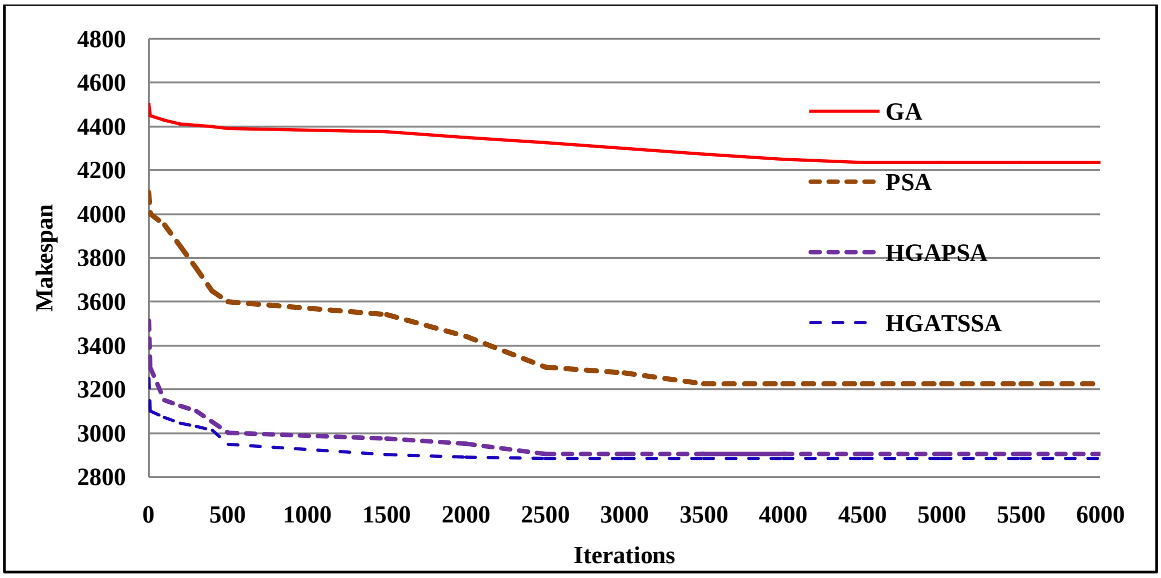

Figure 10. The time evolutions of makespans for the SWV15 (50 jobs and 10 machines)

6. CONCLUSIONS

In this paper we have proposed a new hybrid algorithm for job shop scheduling. The algorithm incorporates the main features of the meta-heuristic algorithm GA, TS and SA. The algorithm is based mainly on the GA, while the TS method is used to generate new members in the GA population. The SA algorithm is used to accelerate the convergence of the GA by testing all the GA members after each reproduction of a new population. This algorithm is implemented in a group of machine. GA is working on the coordinator node and other two algorithms are working in the client nodes. A TS implement of the proposed algorithm is used to generate a neighbor schedule and the SA part is used to simplify and speed up the calculations. The main advantage of the proposed algorithm is speed up the convergence of the optimal schedule of job shop scheduling compare to GA, PSA and HGAPSA.

REFERENCES

[1]. Garey.E.L, Johnson. D. S and Sethi. R, “The complexity of flow shop and job shop scheduling”, Mathematics of Operations Research, Vol. 1, pp.117-129, 1976.

[2]. Brucker. P, Jurisch. B and Sievers. B, “A branch and bound algorithm for job shop scheduling problem”, Discrete Applied Math, Vol. 49, pp. 105-127, 1994.

[3]. Artigues. C, Feillet. D, “A branch and bound method for the job shop problem with sequence dependent setup times”, Annals of Operations Research, Vol. 159, No. 1, pp. 135-159, 2008.

[4]. Lorigeon. T, “A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint”, Journal of the Operational Research Society, Vol. 53, No. 11, pp. 1239-1246, 2002.

[5]. Potts. C. N and Van Wassonhove. L. N, “Dynamic programming and decomposition approaches for the single machine total tardiness problem”, European Journal of Operational Research, Vol. 32, pp, 405-414, 1987.

[6]. Zhang. C.Y, Li. P. G, Rao. Y. Q, “A very fast TS/SA algorithm for the job shop scheduling problem”, Computers and Operations Research, Vol. 35, pp. 282-294, 2008.

[7]. Nowicki. E andSmutnicki. C, “A fast taboo search algorithm for the job shop scheduling problem”, Management Science, Vol. 41, No. 6, pp. 113-125, 1996.

[8]. Dell. A. M, Trubian. M, “Applying tabu-search to job shop scheduling problem,” Annals of Operations Research, Vol. 41, No. 3, pp. 231-252, 1993.

[9]. Wang. T. Y, Wu. K. B, “A revised simulated annealing algorithm for obtaining the minimum total tardiness in job shop scheduling problems”, International Journal of Systems Science, vol. 31, no. 4, pp. 537-542, 2000.

[10]. Kolonko. M, “Some new results on simulated annealing applied to the job shop scheduling problem”, European Journal of Operational Research, vol. 113, no. 1, pp. 123-136, 1999.

[11]. Moon. I, Lee. S and Bae. H, “Genetic algorithms for job shop scheduling problems with alternative routings”, International Journal of Production Research, Vol. 46, No. 10, pp. 2695-2705, 2008.

[12]. Goncalves. J. F, Mendes. J. J. D. M and, Resende. M. G. C, “A hybrid genetic algorithm for the job shop scheduling problem”, European Journal of Operational Research, Vol. 167, No. 1, pp. 77-95, 2005.

[13]. Bierwirth. C and Mattfeld. D. C, “Production scheduling and rescheduling with genetic algorithms”, Evolutionary Computation, Vol. 7, No. 1, pp. 1-17, 1999.

[14]. Liu. B, Wang. L and Jin. Y. H, “An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers”, Computers and Operations Research, Vol. 35, No. 9, pp. 2791-2806, 2008.

[15]. Tasgetiren. M. F, Liang. Y. C and Sevkli. M, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem", European Journal of Operational Research, Vol. 177, No. 3, pp. 1930-1947, 2007.

[16]. Holland. J. H, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press, 1975.

[17]. Goldberg. G. E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.

[18]. Glover. F , "Future paths for integer programming and links to artificial intelligence", Computer and Operation Research, Vol.13, No. 5, pp. 533-549, 1986.

[19]. Glover. F , Tabu Search – Part I, ORSA Journal on Computing, Vol.1, No. 3, pp. 190-206, 1989.

[20]. Glover. F , Tabu Search – Part II, ORSA Journal on Computing, Vol.2, No. 3, pp. 4-32, 1990.

[21]. Brucker. P, Scheduling Algorithms, Springer Verlag, Berlin, 1995

[22]. Davis, L. Job-shop scheduling with genetic algorithm. In Proceedings of the 1st international conference on genetic algorithms and their applications, Pittsburgh, PA, pp. 136–140, 1985.

[23]. Della Croce. F., Tadei. R., andVolta. G. A genetic algorithm for the job shop problem. Computers and Operations Research, Vol.22, No.1,pp. 15–24, 1995.

[24]. Dorndorf. U., and Pesch. E. Evolution based learning in a job-shop scheduling environment. Computers and Operations Research, Vol.22, No.1, pp. 25–40, 1995.

[25]. Eswarmurthy. V., and Tamilarasi. A. Hybridizing tabu search with ant colony optimization for solving job shop scheduling problem. The International Journal of Advanced Manufacturing Technology, Vol.40, pp. 1004–1015, 2009.

[26]. Mattfeld. D. C. Evolutionary search and the job shop: Investigations on genetic algorithms for production scheduling. Heidelberg, Germany: Physica-Verlag, 1996.

[27]. Yamada. T. and Nakano. R. Scheduling by genetic local search with multi-step crossover. In PPSN'IV 4th international conference on parallel problem solving from nature, Berlin, Germany, pp. 960-969, 1996.

[28]. Park. B. J., Choi, H. R., and Kim. H. S. A hybrid genetic algorithm for the job shop scheduling problems. Computers and Industrial Engineering, Vol.45, pp.597–613, 2003

[29]. Aydin. M. E., and Fogarty. T. C. A simulated annealing algorithm for multi-agent systems: A job-shop scheduling application. Journal of Intelligent Manufacturing, Vol.15, No.6, pp.805–814, 2004.

[30]. Gao. J., Gen. M., and Sun. L. Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. Journal of Intelligent Manufacturing, Vol.17, No.4, pp.493–507, 2006.

[31]. Pezzella. F., and Merelli. E. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. European Journal of Operation Research, Vol.120, pp.297–310, 2000.

[32]. Pezzella. F., Morganti. G., and Ciaschetti. G. A genetic algorithm for the flexible job-shop scheduling problem. Computers and Operations Research, Vol.35, pp.3202–3212, 2008.

[33]. Gholami. M. and Zandieh. M. Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop. Journal of Intelligent Manufacturing, Vol.20, No.4, pp.481–498, 2009.

[34]. Gen. M., Gao. J., and Lin. L. Multistage-based genetic algorithm for flexible job-shop scheduling problem. Intelligent and Evolutionary Systems, Studies in Computational Intelligence, Vo. 187, pp.183–196, 2009.

[35]. Pérez. E., Posada. M., and Herrera. F. Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling. Journal of Intelligent Manufacturing, Online Firsttrademark, March 10, 2010.

[36]. Dell'Amico. M., and Trubian, M. Applying tabu search to the job-shop scheduling problem. Annals of Operations Research, Vol. 41, pp. 231–252, 1993.

[37]. Thomsen. S. Meta-heuristics combined with branch and bound. Technical Report. Copenhagen Business School, Copenhagen, Denmark, 1997.

[38]. Chen. L., Bostel. N., Dejax, P., Cai. J., and xi. L. A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. European Journal of Operational Research, Vol. 181, No.1, pp.40–58, 2007.

[39]. Yamada, T. and Nakano, R. Scheduling by genetic local search with multi-step crossover. In PPSN'IV 4th international conference on parallel problem solving from nature, Berlin, Germany, pp. 960–969, 1996

[40]. Zhou, R., Nee, A. Y. C., and Lee, H. P. Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems. International Journal of Production Research, Vol. 47, pp. 2903–2920, 2009.

[41]. Jain, A. S., andMeeran, S. A multi-level hybrid framework applied to the general flow-shop scheduling problem. Computers and Operations Research, Vol. 29, pp.1873–1901, 2002.

[42]. Satake, T., Morikawa, K., Takahashi, K., and Nakamura, N. Simulated annealing approach for minimizing the make-span of the general job shop. International Journal of Production Economics, Vol. 60, No. 61, pp.515–522, 1999.

[43]. Suresh, R. K., andMohanasundaram, K. M. Pareto archived simulated annealing for job shop scheduling with multiple objectives. The International Journal of Advanced Manufacturing Technology, Vol. 29, pp.184–196, 2005.

[44]. Guohui Zhang, Liang Gao and Yang Shi, "A Genetic Algorithm and Tabu Search for Multi Objective Flexible Job Shop Scheduling Problems", International Conference on Computing, Control and Industrial Engineering, 2010.

[45]. Wang. L., and Zheng, D. Z. An effective hybrid optimisation strategy for job shop scheduling problems. Computers and Operations Research, Vol. 28, pp.585–596, 2001.

[46]. Meeran. S. andMorshed. M. S. A hybrid configuration for solving job shop scheduling problems. In 8th Asia Pacific industrial engineering and mangement science Conference, Kaohsiung, Taiwan, 2007.

[47]. González, M. A., Vela, C. R., and Varela, R. Genetic algorithm combined with tabu search for the job shop scheduling problem with setup times' methods and models in artificial and natural computation. A homage to professor Mira's scientific legacy. Lecture Notes in Computer Science, 5601/2009, pp.265–274, 2009. doi:10.1007/978-3-642-02264-7_28.

[48]. Thamilselvan, R, P.Balasubramanie, Analysis of Various Alternate Crossover Strategies for Genetic Algorithm to Solve Job Shop Scheduling. European Journal of Scientific Research, 64(4): 538-554, 2011.www.europeanjournalofscientificresearch.com /ISSUES/EJSR_64_4_05.pdf.

[49]. Thamilselvan, R, P.Balasubramanie, Integration of Genetic Algorithm with Tabu Search for Job Shop Scheduling with Unordered Subsequence Exchange Crossover. Journal of Computer Science, 8(5): 681-693. DOI: 10.3844/jcssp.2012.681.693, 2012