Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (10): 1943-1949, 2009
ISSN 1812-5654

A Novel Simulated Annealing Algorithm to Hybrid Flow Shops Scheduling with Sequence-Dependent Setup Times

A. Alem Tabriz, M. Zandieh and Z. Vaziri
Department of Industrial Management, Faculty of Management and Accounting,
Shahid Beheshti University, GC, Tehran, Iran

Abstract: This study deals with the hybrid flow shop scheduling problems in which there are sequence-dependent setup times. This type of production system is found in industries such as chemical, textile, metallurgical, printed circuit board and automobile manufacture. This study describes a simulated annealing algorithm to the scheduling of a hybrid flow shop with sequence-dependent setup times. The obtained results are compared with those computed by RKGA presented previously. The superiority and effectiveness of our novel simulated annealing algorithm is inferred from all the results obtained in various situations.

Key words: Scheduling, hybrid flow shops, sequence-dependent setup times, makespan, heuristics, simulated annealing

INTRODUCTION

A hybrid flow shop model, commonly known as flexible flow line, allows us to represent most of the production systems. The process industry such as chemical, pharmaceutical, oil, food, tobacco, textile, study and metallurgical industry can be modeled as a hybrid flow shop. A hybrid flow shop consists of a series of production stages, each of which has several facilities in parallel (Zandieh *et al*., 2006). Some stages may have only one facility, but for the plant to be qualified as a hybrid flow shop, at least one stage must have several facilities. The flow of products in the plant is unidirectional. Each product is processed at only one facility in each stage and at one or more stages before it exits the plant. Each stage may have multiple parallel identical machines. These machines can be identical, uniform, or unrelated. Each job is processed by at most one machine at each stage.

Pinedo (2002) cited that machine setup time is a significant factor for production scheduling in all flow patterns and it may easily consume more than 20% of available machine capacity if not handled well. Also the completion time of production and machine setups are influenced by production mix and production sequence. On the one hand, processing in large batches may increase machine utilization and reduce the total setup time. On the other hand, large batch processing increases the flow time. Scheduling problems with Sequence-Sependent Setup Times (SDST) are among the most difficult classes of scheduling problems. A single-machine sequence-dependent setup scheduling problem is equivalent to a Traveling Salesman Problem (TSP) and is NP-hard (Pinedo, 2002). Even for a small system, the complexity of this problem is beyond the reach of existing theories (Luh *et al*., 1998).

Moreover sequence-dependent setup scheduling of a hybrid flow shop system is even more challenging. Although there has been some progress reported, but the understanding of sequence-dependent setups, however, is still believed to be far from being complete (Luh *et al*., 1998). Robust local search improvement techniques for flexible flow-line scheduling were considered by Leon and Ramamoorthy (1997).

Hung and Ching (2003) addressed a scheduling problem taken from a label sticker manufacturing company which is a two-stage hybrid flow shop with the characteristics of sequence-dependent setup time at stage 1, dedicated machines at stage 2 and two due dates. The objective was to schedule one days mix of label stickers through the shop such that the weighted maximal tardiness is minimized. They proposed a heuristic to find the near-optimal schedule for the problem. The performance of the heuristic was evaluated comparing its solution with both the optimal solution for small-sized problems and the solution obtained by the scheduling method used in the shop.

While many studies have been written in the area of scheduling hybrid and flexible flow lines, many of them are restricted to special cases of two stages, specific configurations of machines at stages and to simplify the

Corresponding Author: Prof. M. Zandieh, Department of Industrial Management, Faculty of Management and Accounting, Shahid Beheshti University, GC, Tehran, Iran Tel/Fax: (009821) 44330749

problem, setups are seldom considered in the scheduling. For those ones addressing setups, the setup times are fixed and included in processing times. However, in most real world cases, the length of the setup time depends on both jobs, which is separable from processing. There seems to be published only three works addressing heuristics for flexible flow lines with sequence-dependent setup times. Kurz and Askin (2003) examined scheduling rules for SDST flexible flow lines. They explored three classes of heuristics. The first class of heuristics (cyclic heuristics) is based on simplistic assignment of jobs to machines with little or no regard for the setup times. The second class of heuristics is based on the insertion heuristic for the TSP. The third class of heuristics is based on Johnson's Rule. Note that the second class caters to setup aspects of the problem while the third class derives from standard flow shops. They proposed eight heuristic rules and compared the performance of those on a set of test problems. Moreover, Kurz and Askin (2004) formulated the SDST flexible flow lines as an integer programming model. Because of the difficulty in solving the Integer Programming (IP) model directly, they developed a Random Keys Genetic Algorithm (RKGA). Problem data was generated to evaluate the RKGA with other scheduling heuristics rules, which they proposed aforetime. They created a lower bound to evaluate the heuristics. Zandieh *et al.* (2006) proposed an immune algorithm and showed that this algorithm outperforms the RKGA of Kurz and Askin (2004).

PROBLEM DEFINITION

Let g be the number of workshops in series. Let n be the number of jobs to be processed and m^t be the number of machines in parallel at each stage t. We assume that machines are initially setup for a nominal job 0 at every stage. Job n+1 exists at every stage only to indicate the end of the process, if needed. We have the following definitions.

- p_i^t = Processing time for job i at stage t (i = 1,2....,n; t = 1,2,...,g)
- s_{ij}^t = Sequence-dependent setup time from job i to job j at stage t (i = 1,2....,n; j = 1,2,...,n; t = 1,2,...,g)
- $\tilde{p}_i^t$ = Modified processing time for job i at stage t ($\tilde{p}_i^t = p_i^t + \min_j s_{ij}^t$) (i = 1,2....,n; j = 1,2,...,n; t = 1,2,...,g)
- S^t = Set of jobs that visit workshop stage t (t = 1,2,...,g)
- m^t = No. of machines at stage t (t = 1,2,...,g)

The processing time of job 0 is set at 0. The setup time from job 0 indicates the time to move from the nominal set solution state. We assume that all jobs currently in the system must be completed at each stage before the jobs under consideration may begin setup. The completion times of job 0 at each stage are set to the earliest setup time may begin at that stage. The setup time for job n+1 is set at 0; this job only exists to indicate the end of the schedule. We also include the restriction that every stage must be visited by at least as many jobs as there are machines in that stage.

THE PROPOSED SIMULATED ANNEALING (SA) ALGORITHM

Standard simulated annealing algorithm: SA was first proposed by Kirkpatrick *et al.* (1983) as a method for solving combinatorial optimization problems. The name of the algorithm derives from an analogy between the simulation of the annealing of solids first proposed by Metropolis *et al.* (1953) and the strategy of solving combinatorial optimization problems. Annealing refers to a process of cooling material slowly until it reaches a stable state.

SA (Metropolis *et al.*, 1953) is a generalization of a Monte Carlo method for statistically finding the global optimum for multivariate functions. In SA, a system is initialized at a temperature T with some configuration whose energy is evaluated to be E. A new configuration is constructed by applying a random change and the change in energy ΔE is computed. The new configuration is unconditionally accepted if it lowers the energy of the system. If the energy of the system is increased by the change, the new configuration is accepted with some random probability. In the original Metropolis scheme (Metropolis *et al.*, 1953), the probability is given by the Boltzmann factor

$$\exp\left(\frac{-\Delta E}{k.T}\right)$$

This process is repeated sufficient times at the current temperature to sample the search space and then the temperature is decreased. The process is repeated at the successively lower temperatures until a frozen state is achieved.

SA has been used in operations research to successfully solve a large number of optimization problems (Kirkpatrick, 1984) such as the TSP and various scheduling problems. Here, it is applied to the problem of application scheduling in hybrid flow shops with sequence-dependent setup times.

A simulated annealing algorithm to SDST hybrid flow shop scheduling: SA consists of four phases including initialization of the algorithm at a temperature T_0, evaluate the objective function for each solution, application of neighborhood search to the next solution by hill climbing movements, decrease of temperature and repeat phases 2 through 4 until a lower temperature is achieved.

ALGORITHM SA

Initialize the parameters (T_0 : Initial temperature; N: Number of temperature decrement ; NITER: Number of iterations at each temperature; CS: Cooling schedule):

Choose an initial solution s ∈ S
Evaluate the objective function for solution S
s* = s
best = objective value of solution s
Repeat
Generate a neighborhood solution s' ∈ N(s)
If random [0,1] ≤ $\exp(\frac{-\Delta f}{T})$ Then s = s'
If best > objective value of solution s' Then s* = s';
best = objective value of solution s'
Reduced temperature based on cooling schedule (CS)
Until stop criterion

Initialization: In this step we generate a solution randomly from the feasible domain. Usually, initial solution is randomly generated in the feasible space, but initial solution can influence the convergence time. Thus, random numbers serve as sort keys in order to decode the solution. The decoded solution is evaluated with a objective function for the problem at hand. For example, Norman and Bean (1999) suggest using the following solution representation for an identical multiple machine problem. Each job is assigned a real number between [1, m^1] whose integer part is the machine number to which the job is assigned and whose fractional part is used to sort the jobs assigned to each machine. Once the job assignments and order on each machine are found through the decoding, a schedule can be built incorporating additional factors such as non-zero ready times and sequence-dependent setup times. The desired performance measure can then be found using the schedule.

Kurz and Askin (2003) proposed three heuristics based on greedy methods, flow line methods and the insertion heuristic for the TSP. These heuristics were named Shortest Processing Time Cyclic Heuristic (SPTCH), Flow-Time Multiple Insertion Heuristic (FTMIH) and Johnson's rule based heuristics (g/2, g/2).

In SPTCH, the jobs are ordered at stage 1 in increasing order of the modified processing times $\tilde{p}_i^t$. At subsequent stages, jobs are assigned in earliest ready time order. Jobs are assigned to the machine in every stage that allows it to complete at the earliest time.

The FTMIH is a multiple insertion heuristic to minimize the sum of flow times (completion-ready times) at each stage. It is a multiple machine, multiple stage adaptation of the Insertion Heuristic for the TSP. Setup times are accounted for by integrating their values into the processing times using $\tilde{p}_i^t$. The FTMIH can then be performed using these modified processing times at each stage. Once jobs have been assigned to machines, the true processing and setup times can be used.

Johnsons Rule (Gupta and Tunc, 1994) finds the optimal makespan solution for $n/2/F/C_{max}$ (C_{max} is makespan). The g/2,g/2 Johnson's rule is an extension of Johnson's rule to take into account the setup times for the flow shop with more than two stages. The aggregated first half of the stages and the aggregated last half of the stages are considered to create the order for assignment in stage 1. The value $\tilde{p}_i^t$ is the sum of modified processing times for stages 1 to g/2 and $\tilde{p}_i^t$ is the sum over stages [g/2]+1 to g.

In this research, the representation of Norman and Bean (1999) is used for the jobs in the first stage. The assignment of jobs to machines in subsequent stages follows the method used in SPTCH and the Johnson's rule based heuristics (g/2, g/2), where each job is assigned to the machine that allows it to complete at the earliest time as measured in a greedy fashion.

In this study, we adapted a heuristic making use of NEH (Nawaz, Enscore and Ham) rules in SDST hybrid flow shops. The NEH procedure is based on the idea that jobs with high processing times on all machines should be scheduled as early as possible. Modified processing time is used in this adapted algorithm.

ALGORITHM ADAPTED NEH

(1) Find the total modified processing times of all jobs:
$\tilde{p}_i = \sum_{t=1}^{g} \tilde{p}_i^t$
(2) Order the jobs in descending order of $\tilde{p}_i$
(3) Let π = Ø % empty initial sequence
(4) For [i]: = 1 to n :
(a) J: = job[i]
(b) In order to test job [i] in each possible position of π, call **Procedure A**
(c) Insert job [i] in π at position resulting in lowest makespan
End for

Procedure "A"

(1) At each stage t = 1; . . . ; g, assign job [0] to each machine in that stage.

(2) At stage 1:

```
Let bestmc = 1
For [I]: = 1 to n do
  For mc: = 1 to m^t do
    Place job [i] last on machine mc.
    Find the completion time of job [i]. If this
    time is less on mc than on bestmc,
    Let bestmc = mc.
    Assign job [i] to the last position on
  machine bestmc.
  End for
End for
```

(3) For t: = 2 to g do

```
    Update the ready times in stage t to be the
    completion times in stage (t-1).
    Arrange jobs in increasing order of ready
  times.
    Let bestmc = 1.
    For [i]: = 1 to n do
      For mc: = 1 to m^t do
        Place job [i] last on machine mc.
        Find the completion time of job [i]. If this
        time is less on mc than on bestmc,
        Let bestmc = mc.
        Assign job [i] to the last position on
      machine bestmc.
      End for
  End for
End for
```

In the proposed SA, first the problem is solved by four heuristics SPTCH, FTMIH, g/2, g/2 Johnson's Rule and NEH. Then the better solution is selected as initial solution.

Local search: In this step, randomly two solutions in the neighborhood of the initial solution (current solution) are generated, their makespans are calculated and the better solution is selected as new one. If the new solution is better than the current solution, it is accepted. Otherwise, it will be accepted with a probability:

$$p = \exp(-\frac{\Delta f}{T}) \quad (1)$$

where, Δf is the increase in cost function f and T is a control parameter.

For generating new solutions, two methods were used as follows. In the first method, two numbers within the range of 1 to m^t are generated randomly and then they are exchanged with two numbers which themselves have been chosen randomly from current sequence (Fig. 1a). In the second one, two numbers are selected randomly from the current sequence and then the new solution is generated as shown in Fig. 1b.

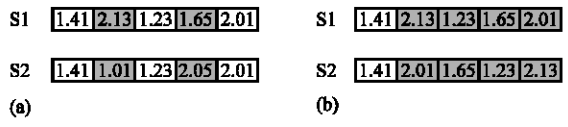

Fig. 1: Generation of new solution

SA's major advantage over other methods is an ability to avoid becoming trapped at local minima. The algorithm employs a random search, which not only accepts changes that decrease objective function, f, but also some changes that increase it.

In each step of Metropolis algorithm, a particle is given a small random displacement and the resulting change, Δf, in the energy of the system is computed. If $\Delta f \leq 0$, the displacement is accepted. The case $\Delta f > 0$ is treated probabilistically. The probability that the configuration is accepted is given in Eq. 1. A certain number of iterations are carried out at each temperature and then the temperature is decreased. This is repeated until the system freezes into a steady state.

The probability of accepting a worse state is given by the equation:

$$p = \exp(-\frac{\Delta f}{T}) > r \quad (2)$$

Where:

Δf = The change in cost (objective) function
T = The current temperature
r = A random number uniformly distributed between 0 and 1

The probability of accepting a worse move is a function of both the temperature of the system and of the change in the objective function. As the temperature of the system decreases, the probability of accepting a worse move is decreased. If the temperature is lower temperature, then only better moves will be accepted.

This loop is repeated number of iterations times at each temperature, then temperature is decreased by one of these cooling schedule:

$$T_i = \frac{T_0 - i}{N}(T_0 - 1) \quad (3)$$

$$T_i = \frac{A}{i+1} + B,\ A = \frac{(T_0 - 1)(N+1)}{N},\ B = T_0 - A \quad (4)$$

$$T_i = T_0 - i^A,\ A = \frac{\log(T_0 - 1)}{\log(N)} \quad (5)$$

where, i is index for stage, T_0 is initial temperature, T_i is the temperature of stage i and N is the number of temperature decrease.

Therefore, the solution of the problem is the best of all temperature current solutions.

EXPERIMENTAL DESIGN

Data generation and settings: An experiment was conducted to test the performance of the electromagnetism algorithm. Following Kurz and Askin (2003), data required for a problem consists of the number of jobs, number of stages, number of machines in each stage, range of processing times and the range of sequence-dependent setup times. The ready times for stage 1 are set to 0 for all jobs. The ready times at stage t + 1 are the completion times at stage t, so there is no need this data to be generated. Processing times are distributed uniformly over two ranges with a mean of 60: [50-70] and [20-100]. Flexible flow lines are considered by allowing some jobs to skip some stages. Following Leon and Ramamoorthy (1997), the probability of skipping a stage is set at 0, 0.05, or 0.40. The setup times are uniformly distributed from 12 to 24 which are 20 to 40% of the mean of the processing time. The setup time matrices are asymmetric and satisfy the triangle inequality. The setup time characteristics follow Rios-Mercado and Bard (1998).

The problem data can be characterized by six factors and each of these factors can have at least two levels. These levels are shown in Table 1.

In general, all combinations of these levels will be tested. However, some further restrictions are introduced. The variable machine distribution factor requires that at least one stage have a different number of machines than the others. Also, the largest number of machines in a stage must be less than the number of jobs. Thus, the combination with 10 machines at each stage and 6 jobs will be skipped and the combination of 1-10 machines per stage with 6 jobs will be changed to 1-6 machines per stage with 6 jobs. There are 252 test scenarios and five data sets are generated for each one.

Simulated annealing algorithm parameters tuning: It is known that the different levels of the parameters clearly affect the quality of the solutions obtained by a simulated annealing algorithm. A number of different simulated annealing algorithms can be obtained with the different combinations f the parameters. We have applied

Table 1: Factor levels

Factor	Levels
No. of jobs	6-30-100 Constant:1-2-10
No. of machines	Variable: Uniform[1-4] - Uniform[1-10]
No. of stages	2-4-8
Processing times	Uniform[50-70]- Uniform[20-100]
Skipping probability	0.00-0.05-0.40

Table 2: Parameters tuning

	Problems		
Parameters	Small	Medium	Large
Initial temperature (T_0)	15	20	15
No. of temperature decrement (N)	20	40	30
No. of iterations at each temperature (NITER)	3	60	100
Cooling schedule (CS)	I	II	II

parameters tuning only for the initial temperature (T_0), number of iterations executed at each temperature (NITER), number of temperature decrease (N) and Cooling Schedule (CS), considering the following ranges:

- Initial temperature (T_0): two levels (15, 20)
- No. of temperature decrement (N): two levels for each state small (20, 30), medium (40, 30), large (30, 50)
- No. of iterations executed at each temperature (NITER): two levels for each state small (3, 2), medium (60, 80), large (100, 60)
- Cooling schedule (CS): three levels (I, II, III)

Thirty six different SA are obtained by these levels. We generate 14 instances, 5 small, 5 medium and 4 large, for each combination of n, m, SDST. All 14 instances are solved by 36 different SA algorithms.

The results have been indicated in three states small, medium and large problem in Table 2.

As the results of the SA algorithm with the mentioned parameters were not better than the RKGA one, Number of temperature decrement and also iterations at each temperature were increased and as a result best final parameters of SA algorithm versus RKGA was estimated for all size problems:

Initial temperature (T_0): 15
No. of temperature decrement (N): 80
No. of iterations at each temperature (NITER): 80
Cooling schedule (CS): Cooling schedule II

Experimental results: Here, we are going to compare the proposed simulated annealing algorithm with the RKGA which proposed by Kurz and Askin (2003) for the SDST flexible flow lines. The heuristics were implemented in MATLAB 7.01 and run on a PC with a Pentium IV 2600 MHz processor with 256 MB of RAM. When the C_{max} of each algorithm has been obtained for its instances, the

Table 3: Average RPD of makespan and solution time for the SA and GA algorithms by n and g

	RPD of makespan		Solution time	
Instance	SA	GA	SA	GA
6×2	1.89	8.30	0.43	0.74
6×4	3.56	8.73	0.61	1.12
6×8	1.69	5.64	0.69	2.28
6 Job	2.38	7.56	0.57	1.38
30×2	3.67	15.75	32.72	36.07
30×4	5.32	16.96	52.79	80.23
30×8	1.52	7.47	107.71	152.24
30 Job	3.50	13.39	64.40	89.51
100×2	0.97	12.15	356.10	210.43
100×4	4.76	25.88	697.43	522.56
100×8	1.74	21.41	1250.73	984.85
100 Job	2.49	19.81	768.22	572.61
Average	2.79	13.59	277.74	221.16

best solution obtained for each instance (which is named Min_{sol}) by any of the two algorithms is calculated. Relative percentage deviation (RPD) is obtained by given formula below:

$$RPD = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \quad (6)$$

where, Alg_{sol} is the C_{max} obtained for a given algorithm and instance. RPD of 4% for a given algorithm means that this algorithm is 4% over the best obtained solution on average. Clearly, lower values of RPD are preferred.

Analysis of makespan and solution time: The results of the experiments for two subsets, averaged for each one of the n and g configurations (5 runs in each experiment) are shown in Table 3. As it can be seen, SA algorithm provides better results than RKGA.

To identify the best algorithm, we have performed a design of experiments and an analysis of results (Table 3).

The results demonstrate that there is a clear difference between performances of the algorithms. The means plot for two algorithms are shown in Fig. 2.

Analysis of problem size factor (number of jobs): In order to see the effects of number of jobs on two algorithms, means of makespan RPD is applied. Means plot for the interaction between the factors type of method and number of jobs are shown in Fig. 3. As we can see, the all size problems (3, 30, 100), SA works better than RKGA.

Analysis of g factor (number of stages): In this section, Means of RPD are applied to see the effect of magnitude

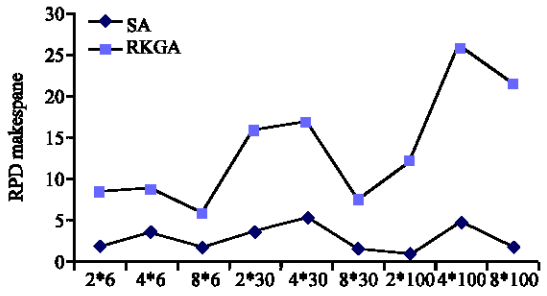

Fig. 2: Means plot and for SA and GA algorithms

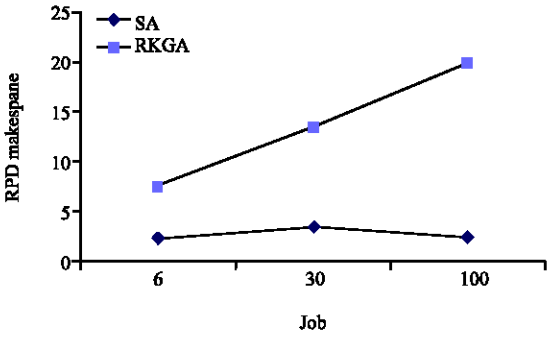

Fig. 3: Means plot for the interaction between the factors type of algorithm and number of jobs

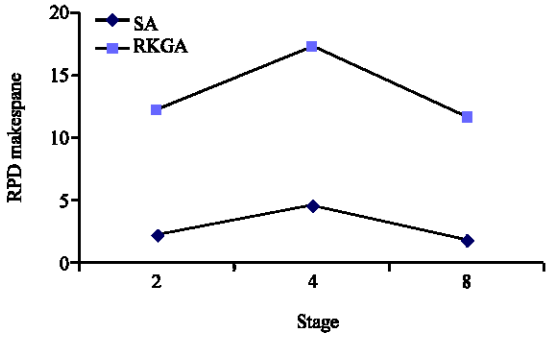

Fig. 4: Means plot for the interaction between the factors type of algorithm and magnitude of stages

of stages on quality of the algorithms. The results are shown in Fig. 4. As we can see, in the majority of the cases/trials SA works better than RKGA.

Final analysis of makespan and solution time: Table 4 shows the required computational time for SA versus RKGA. Based on this data, we can see the solution time for SA will significantly increase as problem size increases. This is reasonable because makespan will

Table 4: SA Solution time versus GA

	Solution time				
	Decrease		Similar	Increase	
Problem size	Percent of problem	Average decrease	Percent of problem	Percent of problem	Average increase
Small	95.8	0.88		4.2	0.84
Medium	65.5	49.11		34.5	20.57
Large	12.2	195.74		87.8	250.10

Table 5: SA Makespan value versus GA

	Makespan value				
	Decrease		Similar	Increase	
Problem size	Percent of problem	Average decrease	Percent of problem	Percent of problem	Average increase
Small	64	37.15	2.7	33.3	8.55
Medium	68	91.73	-	32.0	21.20
Large	70	558.45	-	30.0	49.63

significantly increase as the problem size increases. Table 5 shows the makespan value for SA versus RKGA.

CONCLUSION

A simulated annealing approach for the scheduling of a hybrid flow shop has been successfully developed. SA's major advantage over other methods is an ability to avoid becoming trapped at local minima. The algorithm employs a random search, which not only accepts changes that decrease objective function, but also some changes that increase it.

A set of experiments were carried out to illustrate the effectiveness of simulated annealing algorithm in SDST hybrid flow shop scheduling. Compared to past RKGA, the lower makespan values in many test problems, specially in medium and large problems can be attributed to the fact that the SA tends to converge prematurely due to the absence of a method to restraint the 'over-dominance of good solution.

REFERENCES

Gupta, J.N.D. and E.A. Tunc, 1994. Scheduling a two-stage hybrid flowshop with separable setup and removal times. Eur. J. Operat. Res., 77: 415-428.

Hung, T.S.L. and J.L. Ching, 2003. A case study in a two-stage hybrid flow shop with setup time and dedicated machines. Int. J. Prod. Econ., 86: 133-143.

Kirkpatrick, S., 1984. Optimization by simulated annealing: Quantitative studies. J. Statist. Phys., 34: 975-986.

Kurz, M.E. and R.G. Askin, 2003. Comparing scheduling rules for flexible flow lines. Int. J. Prod. Econ., 85: 371-388.

Kurz, M.E. and R.G. Askin, 2004. Scheduling flexible flow lines with sequence-dependent setup times. Eur. J. Operat. Res., 159: 66-82.

Leon, V.J. and B. Ramamoorthy, 1997. An adaptable problem-space-based search method for flexible flow line scheduling. IIE Trans., 29: 115-125.

Luh, P.B., L. Gou, Y. Zhang, T. Nagahora, M. Tsuji and K. Yoneda *et al*., 1998. Job shop scheduling with group-dependent setups, finite buffers and long time horizon. Ann. Operat. Res., 76: 233-259.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, 1953. Equation of state calculations by fast computing machines. J. Chem. Phys., 21: 1087-1092.

Norman, B.A. and J.C. Bean, 1999. A genetic algorithm methodology for complex scheduling problems. Naval Res. Logist., 46: 199-211.

Pinedo, M., 2002. Scheduling Theory, Algorithms and Systems. 2nd Edn., Prentice-Hall, Englewood Cliffs, New Jersey,.

Rios-Mercado, R.Z. and J.F. Bard, 1988. Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. Comput. Operat. Res., 25: 351-366.

Zandieh, M., S.M.T. Fatemi Ghomi and S.M. Moattar Husseini, 2006. An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. J. Applied Math. Comput., 180: 111-127.