Proceedings of the 6th WSEAS Int. Conf. on EVOLUTIONARY COMPUTING, Lisbon, Portugal, June 16-18, 2005 (pp86-91)

Experimental Analysis in Simulated Annealing to Scheduling Problems when Upper Bounds are used

Marco Antonio Cruz-Chávez[1] and Juan Frausto-Solís[2] and David Juárez-Romero[1]

[1]Center of Investigation in Engineering and Applied Science, UAEM
Av. Universidad 1001, Col. Chamilpa, 62270, Cuernavaca, Morelos, MÉXICO

[2] Department of Computer Science, ITESM, Campus Cuernavaca
Paseo de la Reforma 182-A, 62589, Temixco, Morelos, MÉXICO

Abstract- An algorithm of simulated annealing for the job shop scheduling problem is presented. The proposed algorithm restarts with a new value every time the previous algorithm finishes. To begin the process of annealing, the starting point is a randomly generated schedule with the condition that the initial value of the makespan of the schedule does not surpass a previously established upper bound. The experimental results show the importance of using upper bounds in simulated annealing in order to more quickly approach good solutions.

Key-Words: - Job shop, upper bound, scheduling, makespan and simulated annealing.

Introduction

The job shop scheduling problem (JSSP) is considered to be one of the most difficult to solve in combinatorial optimization. It is also one of the most difficult problems in the NP-hard class [7]. For this reason, Muth and Thompson [9] took over 20 years to solve the problem of ten machines and ten jobs [5].

The job shop scheduling problem consists of a set of machines that each carry out the execution of a set of jobs. Each job consists of a certain number of operations, which must be carried out in a specific order. Each operation is carried out by a specific machine and has a specific time of execution. Each machine can execute a maximum of one operation at any given point in time. A single machine is unable to carry out more than one operation of the same job. The objective of the problem is to find the makespan. The makespan is defined as the time it takes to complete the last operation in the system. In a solution to the JSSP, the sequence of operations for each machine as well as start times for each operation are obtained.

An immense number of models exist that represent the JSSP, but the two most important and influential models are those of disjunctive formulation [6] and disjunctive graph [6]. From these two models many others have emerged.

The disjunctive formulation model considers several sets: a set J of *n* jobs, where $\boldsymbol{J} = \{J_1, J_2, ...J_n\}$; a set $\boldsymbol{M}$ of *m* machines where $\boldsymbol{M} = \{M_1, M_2, ..., M_m\}$; and a set $\boldsymbol{O}$ of operations where $\boldsymbol{O} = \{1,2,3,...\}$. These operations form *k* subsets of operations for each one of the jobs ($J_k \subseteq \boldsymbol{O}$) and machines ($M_k \subseteq \boldsymbol{O}$).

Each operation *j* has a processing time of p_j. In a job J_k, each pair of operations *i, j* possess a relationship of precedence represented $(i \prec j)$. Only one operation performed by a machine M_k, can be executed at any given point in time. Given the previously mentioned problem restrictions, the function of the starting time, $\boldsymbol{s}$ of each operation can be represented in the following manner:

$$\forall_{\ j \in \boldsymbol{O}} \qquad s_j \geq 0 \qquad (1)$$

$$\forall_{\ i,j \in \boldsymbol{O},\ (i \prec j) \in J_k} \qquad s_i + p_i \leq s_j \qquad (2)$$

$$\forall_{\ i,j \in \boldsymbol{O},\ (i,j \in M_k)} \qquad s_i + p_i \leq s_j \vee s_j + p_j \leq s_i \qquad (3)$$

The constraint in (1) indicates that the starting time of the operation *j* must be greater than or equal to zero; meaning only positive values are accepted. The constraint in (2) is a precedence constraint. It indicates that within one job which contains operations *i* and *j*, in order for *j* to begin, *i* must be completed. The constraints in (3) are disjunctive. These constraints ensure that two operations, *i* and *j*, which are performed by the same machine are not carried out simultaneously. The objective is to minimize the makespan, which is defined based on starting times, and can be expressed as (4):

$$\text{Min}\left[\max_{j\in O}\left(s_j + p_j\right)\right] \qquad (4)$$

The disjunctive graph model is shown in Figure 1 for a JSSP of 3x3. From Figure 1, it can be seen that the nodes of the graph represent the operations performed in the problem. In each operation (node) of the graph, the first number represents which job the operation pertains to, and the second number represents the machine that performs that operation. It can be observed that the group of operations that form a job are united with a conjunctive arc, which represents the precedence constraints for each pair of operations (e.g., operations 1,1 and 1,2). In the group of operations that a machine executes, each pair of operations is united with a disjunctive arc (e.g., operations 1,1 and 2,1). These arcs represent the resource capacity constraints and correspond to the constraints of the equations in (3) of the disjunctive formulation model. In addition, there are two operations, I and *, which represent the beginning and end of the problem respectively. These operations are actually fictitious and have a processing time of zero. The processing time of each other operation is written beside the node and corresponds to the times pi of the disjunctive formulation model (equations 2 and 3). For example, the operation 1,1 has a processing time of P1,1.

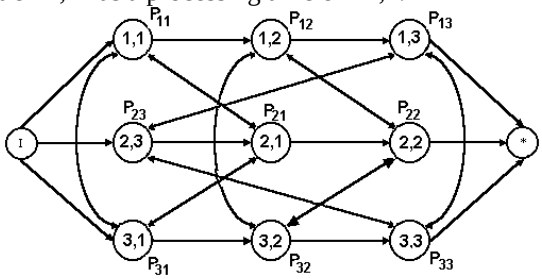

Fig. 1. Disjunctive graph for a JSSP of 3x3

In order to find a solution using the disjunctive graph model, it is necessary to arrange the arrows of the disjunctive arcs in such a way that the obtained sequence does not contain any cycles [1]. Once the sequence is established, it is common to obtain the scheduling of the operations, which is expressed as starting times for each operation. The model of the disjunctive graph was used to develop the SAR algorithm presented here.

The simulated annealing algorithm introduced by Kirkpatrick et al. [8] is an analogy between the annealing process of solids and the problem of solving combinatorial optimization problems. This algorithm has been used with high rates of success for JSSP by several researchers [1],[2],[10],[12],[13], and [14]. This simulated annealing algorithm is shown in Figure 2.

1. Given an initial configuration $S = S_0$ and an initial temperature $T = T_0$
2. While the final temperature Tf is not reached,
3. While equilibrium is not reached,
- *generate a state S′ by means of a perturbation in S*
- *if f(S')-F(S)<=0, the state is accepted as the current state, S =S'*
- *if f(S')-F(S)>0, the state is accepted with the probability:*

$$P_{accept} = e^{-\left(\frac{f(S')-f(S)}{T}\right)} \qquad (5)$$

- *with a randomly generated number α evenly distributed between (0,1)*
- *if $\alpha < P_{accept}$ the state is accepted like the current, S = S'*

If the equilibrium does not exist, return to 3
$T=T*\beta$
If T >= Tf, return to 2
4. The best obtained configuration is the solution

Fig. 2. Simulated annealing algorithm

In Figure 2 for the JSSP, *S* is a schedule obtained by using a randomly generated initial point. *S′* is in the neighborhood of *S*, which is obtained by a small perturbation of *S*. T_o and T_f are the intial and final temperatures of the process. β is the coefficient of temperature that controls the cooling of the system. f(S) is the energy of the configuration S, which is generally the makespan. The equation (5) is the Boltzmann distribution function [11].

The simulated annealing algorithm, represented in the Figure 2, allows for a search for the global optimum when the temperature is high because it accepts good and bad configurations in a similar percentage. As the temperature begins to diminish, the algorithm accepts more good configurations than bad. Due to this behavior, if in each cycle equilibrium is reached, there is a high probability that the optimal, or very close to the optimal, solution will be reached.

One of the ways of perturbing the neighborhood of S is proposed by Balas [3], and involves exchanging a pair of adjacent operations that are within critical blocks of operations. This form of altering the neighborhood is known as N_1. The critical blocks of operations are the operations that form the longest path of the schedule. Each critical block of operations that form this path are performed by a common machine. According to Balas, the first operation of the path becomes the last, and the last becomes the fictitious first operation. Changes in the neighborhood of this type, N_1, have been used previously in simulated annealing with good results by [1], [2], [10], [12], and [14]. This type of change is what is used in this work due to ease of implementation.

Other researchers have developed variations of N_1. The algorithm of Matsuo et al. 1[0] is a derivation of N_1, called N_{1a}. This type of change to the neighborhood involves changing the placement of three pairs of adjacent operations simultaneously, where each operation is performed by a different machine. The algorithm of Aart et al. [1], also a derivation of N_1, called N_{1b}, involves reversing three adjacent pairs of operations that are all performed by the same machine, and with the condition that one of the pairs does not form the longest path.

Another type of derivation of N_1 is the neighborhood of critical block (CB), which is called N_2. In this type of neighborhood, one operation in the block is changed for either the initial or final operation of the block. It is not required that the operations that change places be adjacent. The algorithms that use N_2 are the CBSA of Yamada et al. [14] and the CSBA+CB of Yamada and Nakano [13]. This last algorithm uses a deterministic local search called shifting bottleneck [15]. It uses shifting bottleneck when the schedule S', which is a perturbation of S, is rejected in the simulated annealing. When the rejection occurs, the shifting bottleneck is applied to S' in order to improve it. Once S' is improved, if $f(S')$ greater than $f(S)$, the solution is accepted. In this type of neighborhood, the shifting bottleneck must be used whenever S' is rejected in the algorithm. The implementation of the shifting bottleneck is not easy because it requires that the release time and due dates are calculated. Here, an algorithm is proposed which is easy to implement and produces high quality solutions; it is a simulated annealing algorithm with restart [16].

Simulated Annealing using Upper Bounds

The proposed algorithm of simulated annealing with restart (SAR) consists of executing a set of simulated annealing algorithms, as can be seen in Figure 2. Each simulated annealing that is executed involves the iteration of the SAR algorithm. Each repetition begins using a different schedule. The idea of beginning with different schedules for each repetition of SAR came about because of experimental tests that were carried out. In the experimental tests, it was detected that even though simulated annealing allows the search for a global optimum, at some point low temperatures eliminate this possibility and the search finds a local optimum. By beginning with different schedules, more variability of solutions is obtained than by beginning with only a single schedule. It is seen that the restart of the simulated annealing algorithm leaves a different point in the solution space each time it is repeated.

The restarting of the algorithm allows for the search of global optimums by using a new schedule in each repetition of SAR. This allows for a different part of the solution space to be explored when SAR is at a local optimum. This not only increases the probability of finding a global optimum, but also increases the time of the search.

In the SAR algorithm, at the beginning of each simulated annealing, an UB (upper bound) is established in order to randomly obtain the schedule with which to start the process. The value of the makespan of this schedule may not be greater than the UB, or the schedule is not accepted as the initial configuration of the annealing. An UB is used due to the fact that in SAR, a great number of simulated annealings are executed. The UB is used to limit the solution space of the problem in order to decrease the time it takes for the SAR to arrive at a good solution. Based on the tests carried out, it was found that by using an UB, the algorithm improved the quality of the solution it was able to attain. For the SAR algorithm, the UB was established by trial and error, so that the algorithm took no longer than 15 seconds to find an I-SCHED (initial schedule) that could be used to begin the repetitions of the SAR algorithm. In order to obtain an I-SCHED that did not surpass the UB, a random procedure was used. First, a random sequence of operations was formed for each machine. Next, a scheduling algorithm [17] was used to eliminate global cycles. Finally, this schedule is improved by using the Giffler and Thompson algorithm [18] that obtains active schedules [19]. With the proposed procedure, obtaining good initial solutions that they don't surpass an UB in a short period of time could be assured.

In each iteration of the SAR algorithm, simulated annealing is fully completed. The best configuration after all the repetitions are completed is the final solution. The number of annealings carried out, that is, the number of repetitions of the algorithm, is a function of time of execution and depends on the problem. The simulated annealing algorithm with restart using upper bound can be seen in Figure 3.

Other forms of choosing new schedules have been proven that also permit escape from the local optimum in simulated annealing. One of them is proposed by Yamada and Nakano [13] and involves carrying out a procedure called re-intensification. In this procedure, when the annealing checks the whole neighborhood and will not accept a new schedule because of a low temperature, a function of probability is used. Yamada and Nakano use the probability obtained from the function of Boltzmann, where each neighbor S' of the schedule S is evaluated. Based on the probabilities of the neighbors of S, one is chosen and the process of annealing is continued. In this way it is possible to look more broadly than the local optimum to find the global optimum.

Another type of selection of new schedules is proposed by Yamada et al. [14]. In this procedure, if the simulated annealing has not improved the solution after a great number of accepted configurations, it is assumed that one is in a local optimum. In this case, the best solution found up to the current point in time replaces the current solution, and the temperature is calculated in an adaptive form. As long

as the calculated temperature is greater than the current temperature, the algorithm is restarted with the new values. In this way, it is possible to continue searching for the global optimum.

Aydin and Fogarty [2] use an initial population of individuals (schedules) in their simulated annealing logarithm parallel. In each population, the individuals are taken one by one randomly in order to improve the population. Each one goes through the process of simulated annealing. Each population is improved until a maximum number of repetitions are reached. With time, the initial populations are substituted with the better obtained individuals.

1. Given initial iteration k = 0, initial values of S_f, T_f, β
2. Beginning of annealing k=k+1:
3. S=So<=upper bound, T=T_o, initial Sc.
4. While the final temperature Tf is not reached,
5. While equilibrium is not reached:
- *generate a state S′ by means of a perturbation in S*
- *if f(S')-F(S)<=0 the state is accepted as the current state, S= S'*
- *if f(S')-F(S)>0 the state is accepted with the probability*
- $P_{accept} = e^{-\left(\frac{f(S')-f(S)}{T}\right)}$
- *with a randomly generated number β evenly distributed between (0,1)*
- *if $\beta < P_{accept}$ the state is accepted like the current, S = S'*
- *if $S < S_c$ then $S_c = S$*
- *If the equilibrium does not exist, return to 5*

T = T β.*
The best configuration is stored, if $S_c < S_f$ then $S_f = S_c$
If T >= Tf, return to 4
If k < maxiter, return to 2 to begin a new annealing
The solution is S_f

Fig. 3. The simulated annealing algorithm with restart to JSSP using an upper bound

Computational Results

The proposed algorithm was proven with two problems registered in the OR library [4] and for which the optimum solution is known. The first benchmark is the FT10 of Muth and Thompson, which is the 10x10 they proposed in 1963. The second is the benchmark LA40 of S. Lawrence that is a 15x15, proposed in 1984. For the problem FT10, ten trials were done with an UB = 1570 (Makespan), generating the initial schedules with the procedure I-SCHED, and ten trials were done without an UB, but also using the procedure of I-SCHED. The parameters of T_0 = 32*(Makespan of the initial solution), T_f = 1.0, □ = 0.98, and N_1 as the type of neighborhood were equal in each test performed, whether there was an UB or not. Table 1 shows the results obtained by using an UB of 1570 in the makespan. In the same table, the results are shown when an UB was not used. In both cases, a maximum time of four hours was used to obtain the results. As demonstrated by the data in the table, when the UB is used, the optimum is obtained in 80% of trials, the quickest being obtained in 44 minutes, 55 seconds. The standard deviation is 2.95 with an average makespan of 931.4.

We compared the results obtained in Table 1 with the algorithms of re-intensification ASSA (Adjacent Swapping) and CBSA, of Yamada et al. [14] because they have obtained some of the best results for the problem FT10. They also carried out ten trials. ASSA found solutions using the neighborhood N_1 and CBSA using the neighborhood N_2. It can be seen that ASSA presents a greater standard deviation of 5.10 and a higher average makespan of 939.5. The worst result obtained by the SAR algorithm for the makespan was 937 and by ASSA 951. Through this comparison, it is obvious that the SAR algorithm is more effective than ASSA in obtaining the optimum for the problem FT10. For the CBSA algorithm, in ten trials, and average makespan of 930.8 was obtained and a standard deviation of 2.4. Only in one trial was CBSA unable to obtain the optimum result, this trial gave a makespan of 938. These results indicate that CBSA, by a small margin, is better able to find accurate solutions than SAR. It is believed that the CBSA obtains better results due to the neighborhood it uses, because the only difference between ASSA and CBSA is the type of neighborhood used.

The average time it took for the SAR algorithm to arrive at the solution was 2 hours, 15 minutes, and 13 seconds, for ASSA it was 35 minutes, 43 seconds, and for CBSA it was 44 minutes, 36 seconds. The great difference in times can be explained by the fact that the SAR algorithm restarts with a new annealing in each repetition which causes it to need a great deal of time to be executed. It is important to emphasize that a great number of tests were generated, with sequences of slower cooling in both cases, with and without upper bounds. Tests were also generated with the basic algorithm, as seen in Figure 2. The results of all the performed tests slower cooling (with β of 0.981 to 0.999) were farther from the global optimum. In addition, there was a considerable increase in the time of execution of the algorithm in order to find these solutions. In the results reported in the table 1, the values of β was fixed at β = 0.98, because it is the value at which the best results were obtained. Other parameters were also fixed with the values mentioned previously, because they were the values found to improve the quality of results in the performed tests.

From Table 1, for the problem of FT10, it can be observed that if an upper bound is not used, the obtained results are of poor quality, with a standard deviation of 4.97 and an average makespan of 941.9. This is because when

an upper bound is used, SAR uses better schedules, which allow it to reach better solutions through the changes in the neighborhood. This indicates higher probability that the solution space is nearer the global optimum, and a greater number of good schedules exist, while only a small number of bad schedules are present.

Table 1. Results of the simulated annealing algorithm with restart, with and without upper bound for the problem FT10

FT10 , 10 x 10, optimum = 930			
UB = 1570		without UB	
Makespan	t = sec	Makespan	t = sec
930	4584.18	943	14400
937	14250.25	944	14400
930	3185.29	944	14400
930	14137.15	938	14400
930	6472.21	937	14400
937	14000.00	949	14400
930	5070.34	949	14400
930	6876.12	943	14400
930	9310.64	935	14400
930	2695.30	937	14400

Table 2 shows the results of several algorithms of simulated annealing for the problem FT10. In the table, the type of neighborhood each author used is specified. The makespan presented by Aart et al. [1], using N_1 and N_{1b}, is an average of five trials. The makespan presented by Van Laarhoven [12], N_1, is the best of five trials. The makespan presented by Matsuo et al. [10], N_{1a}, was obtained in one trial. From Table 2 it can be observed that it was not possible for any algorithm to find the global optimum. None of these algorithms involve restarting the annealing so their time of execution is small. It can also be observed that most of the results are poorer than those obtained by the SAR when an upper bound is not applied (Table 1). This shows that a simulated annealing with restart and without upper bounds could improve the solution obtained for FT10.

Table 2. Results of several simulated annealing algorithms for the problem FT10

FT10 , 10 x 10, optimum = 930		
Authors	t = seg.	Makespan
Aart et al. (N_1)	99	969
Aart et al. (N_{1b})	99	977
Van Laarhoven (N_1)	3895	951
Matsuo et al. (N_{1a})	987	946

Table 3 shows the best performance of several algorithms of simulated annealing, including the SAR algorithm, for the benchmark LA40 of JSSP. The parameters in SAR were fixed to: UB = 2300, T_0 = 25, T_f = 5.0 and □ = 0.99. The table presents the type of neighborhood that each author used in their algorithm. The neighborhoods are the following: algorithm of Van Laarhoven et al. [12], N_1, which represents the best of five trials; algorithm of Aart et al [1], N_1, which represents the average of five trials; algorithm of Aart et al [1], N_{1b}, which represents the average of five trials; algorithm of Matsuo et al. [10], N_{1a}, which represents a single trial; CBSA algorithm, N_2, of Yamada et al. [14] which represents the best of five trials, CBSA+SB algorithm, N_2, of Yamada and Nakano, which represents the best result of ten trials; and SAR, N_1, which represents the best of five trials.

Table 3. Results of several simulated annealing algorithm for the problem LA40

LA40, 15 X 15, optimum = 1222		
Algorithm	Makespan	%ER
CBSA+SB (N_2)	1228	0.49
SAR (N_1)	1233	0.90
Van Laarhoven (N_1)	1234	0.98
Matsuo et al. (N_{1a})	1235	1.06
CBSA (N_2)	1235	1.06
Aart et al. (N_{1b})	1254	2.62
Aart et al. (N_1)	1256	2.78

In Table 3 it can be observed that the result obtained by SAR with a 0.90% relative error is better than all of the other algorithms that use N_1 and derivatives of N_1. SAR also surpasses the CBSA, which uses the N_2 type of neighborhood. SAR is surpassed only by CBSA+SB. The CBSA+SB algorithm uses the neighborhood N_2 and implements the procedure of deterministic local search, called shifting bottleneck, for the re-optimization of schedules obtained in each repetition of the algorithm.

Conclusion

The use of upper bounds in the algorithm allows the solution of the problem FT10 to be found in almost all trials. This indicates that for this problem, in the simulated annealing algorithm with restart, starting with the good schedules that do not surpass an upper bound, improves the solution considerably. Also, it is recommended that the simulated annealing be restarted in several points with good schedules. By doing this, better solutions are obtained which are nearer to the global optimum. This could help avoid the situation where a local optimum is found which is far from the global optimum.

Although the algorithm for simulated annealing presented here is powerful, it is not perfect. For example, if another type of more powerful neighborhood were used (e.g., N_2), it is possible that the performance of SAR could be improved.

When a large numbers of annealing are executed, the algorithm takes a great deal of time to arrive at a solution. These aspects are currently being improved.

The quality of the solution of the SAR algorithm presented here for the problem LA40 is comparable to the CBSA+SB algorithm with N_2. One advantage that the SAR

algorithm has over the CBSA+SB is that for the CBSA+SB, it is necessary to find the machines with the shifting bottleneck in each repetition. To find all of this information, it is necessary to calculate the release times and due dates of each operation that is involved in the problem. Thus, because of the similar quality and simpler implementation, SAR appears to be of more interest from a scientific point of view. It is thought that SAR would have better performance with large problems than CBSA+SB due to the fact that SAR does not use deterministic local search procedures. Better performance would be possible if SAR were improved so it would not take so long. This work is currently being done.

References

1. Aarts, E.H.L., Van Laarhoven, P.J.M., Lenstra, J.K., and Ulder, N.L.J., A computational study of local search algorithms for job shop scheduling, ORSA Journal on Computing 6, 118-125, 1994.
2. Aydin, M. E., and Fogarty, T. C., Modular Simulated annealing algorithm for job shop scheduling running on distributed resource machin (DRM), South Bank University, SCISM, 103 Borough Road, London, SE1 0AA, UK, 2002.
3. Balas, E., Machine sequencing via disjunctive graphs: an implicit enumeration algorithm, Oper. Res., 17:941-957, 1969.
4. Beasley, J.E., OR Library, Imperial College, Management School, http://mscmga.ms.ic.ac.uk/info.html, 1990.
5. Carlier, J., and Pinson, E., An algorithm for solving the job-shop problem. Manage. Sci., 35(2): 164-176, 1989.
6. Conway, R. W., Maxwell, W.L., and Miller, L. W., Theory of Scheduling, Addison-Wesley, Reading, Massachusetts, 1967.
7. Garey, M.R., Johnson, D.S. and Sethi, R., The complexity of Flow shop and Job shop Scheduling. Mathematics of Operations Research, Vol. I, No 2, USA, 117-129, May, 1976.
8. Kirkpatrick, S., Gelatt S. D. Jr., and Vecchi, M. P., Optimization by simulated annealing. Science, 220(4598), 13 May, 671-680, 1983.
9. Muth, J. F., and Thompson, G. L., Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, Ch 15, pp. 225-251, 1963.
10. Matsuo, H. Suii, C.J. and Sullivan, R.S., A controlled search simulated annealing method for the general job shop scheduling problem, Working paper 03-04-88, Graduate School of Business, University of Texas, Austin, 1988.
11. Metrópolis, N., Rosenbluth, A. W., Rosenbluth, M. N. Teller, A. H. and Teller, E., Equation of state calculations by fast computing machines, The Journal of Chemical Physics, 21(6), 1087-1092, June 1953.
12. Van Laarhoven, P.J.M., Aarts E.H.L., and Lenstra, J.K., Job shop scheduling by simulated annealing. Oper. Res., 40(1):113-125, 1992.
13. Yamada T., and Nakano, R. Job-shop scheduling by simulated annealing combined with deterministic local search, Meta-heuristics: theory and applications, Kluwer academic publishers MA, USA, pp. 237-248, 1996.
14. Yamada, T., Rosen B. E., and Nakano, R., A simulated annealing approach to job shop scheduling using critical block transition operators, IEEE, 0-7803-1901-X/94, 1994.
15. Adams, J., Balas E., and Zawack, D., The shifting bottleneck procedure for job shop scheduling, Mgmt. Sci., 34, 1988.
16. Ingber, L., Simulated annealing: Practice versus theory, Mathematical Computer Modelling, 18(11), 29-57, 1993.
17. Nakano R. and Yamada, T., Conventional Genetic Algorithm for Job-Shop. Problems, in Kenneth, M. K. and Booker, L. B. (eds) Proceedings of the 4th International Conference on Genetic Algorithms and their Applications, San Diego, USA, pp. 474-479, 1991.
18. Zalzala, P. J., and Flemming., Zalsala, A.M.S. (Ali M.S.), ed., Genetic algorithms in engineering systems /Edited by A.M.S. Institution of Electrical Engineers, London, 1997.
19. Pinedo, M., Scheduling Theory, Algorithms, and Systems, Prentice Hall, U.S.A., 1995.