A New Algorithm That Obtains an Approximation of the Critical Path in the Job Shop Scheduling Problem

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

[1] CIICAp, Autonomous University of Morelos State
Av. Universidad 1001, Chamilpa, 62209, Cuernavaca Morelos, México
mcruz@uaem.mx
[2] Department of Computer Science, ITESM, Campus Cuernavaca
Paseo de la Reforma 182-A, Lomas de Cuernavaca, 62589, Temixco, Morelos, México
juan.frausto@itesm.mx

Abstract. This paper presents a new algorithm that obtains an approximation of the Critical Path in schedules generated using the disjunctive graph model that represents the Job Shop Scheduling Problem (JSSP). This algorithm selects a set of operations in the JSSP, where on the average ninety nine percent of the total operations that belong to the set are part of the critical path. A comparison is made of cost and performance between the proposed algorithm, CPA (Critical Path Approximation), and the classic algorithm, CPM (Critical Path Method). With the obtained results, it is demonstrated that the proposed algorithm is very efficient and effective at generating neighborhoods in the simulated annealing algorithm for the JSSP.

Keywords: Critical path, metaheuristic, schedule, slack time, neighborhood.

1 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 [1]. Due to this, the importance of finding new algorithms that help in the solution of this problem is understandable. The search for more efficient algorithms has been focused in the area of non-deterministic algorithms, given the characteristics of JSSP.

Given the good results obtained in JSSP for some non-deterministic algorithms of local search, such as simulated annealing (SA) [2, 3, 4, 5, 6, 7, 8], great interest has been taken in improving the operation of these metaheuristics. Most work has been done in searching for better neighborhood structures [2, 5, 6, 7, 9] that permit more efficient selection of neighbors.

At the moment, a very effective neighborhood structure exists, N_1 [4], that allows for the selection of neighbors in a schedule such that the search space of the JSSP decreases considerably. This allows more rapid advancement in the search for the global optimum. In order to use the structure of neighborhood N_1, it is necessary find the critical path (CP) of the schedule, which is formed by a set of operations of the JSSP. These operations are called critical operations. In this way, the only neighbors with

A. Gelbukh and C.A. Reyes-Garcia (Eds.): MICAI 2006, LNAI 4293, pp. 450 – 460, 2006.

the possibility of being chosen are those that are generated by a permutation of a pair of critical operations.

A metaheuristic has the characteristic of generating repeated local searches. Therefore, when N_1 is used, it is necessary to calculate the CP every time that a neighbor in the neighborhood [4] of the schedule is chosen. Because of this, the execution of the metaheuristic tends to be slower in large problems of job shop because of the repeated calculation of the CP. This is true even though [10] the algorithms that are used in order to calculate the CP are polynomial [11].

In JSSP, it has been proven [4] that only neighbors that are obtained through the permutation of pair of operations that belong to the CP of a schedule could have a makespan[1] less than that of the original schedule. Due to this, when working with metaheuristics in JSSP where the objective function is obtaining the minimum makespan, the structure of neighborhood N_1 or one of its derivatives [2, 3, 4, 5, 6, 7, 8] is used.

Several options exist that could improve the efficiency of the metaheuristics that use the neighborhood structure N_1. One option is to generate algorithms that calculate the critical path in a more efficient form. Another option is to generate algorithms that do not calculate the critical path. In place of this calculation, these algorithms would have a high probability of generating neighbors by a permutation of a pair of critical operations. The second option is presented in this paper. This option involves generating an algorithm that, based on certain heuristics approaches, selects a set Ω of operations from all the existent operations in the job shop. This set is selected such that it contains all of the operations that form the CP and a few others operations as well. Within the set Ω, the largest percentages of operations are critical operations, that is, Ω possesses a minimum number of non critical operations. In order to explore this second option, an algorithm was generated called CPA (Critical Path Approximation), which selects an operations set from a schedule. This operations set has the characteristic of containing all the operations belonging to the CP. These critical operations constitute 99 percent of all the operations in the set Ω.

This research contributes to the effort to find more efficient ways to use metaheuristics for JSSP with the neighborhood structure N_1 by proposing an efficient algorithm for calculating the CP.

Following this brief introduction, section two explains the disjunctive graph model of the job shop scheduling problem that is used to generate schedules where the CP and Ω are obtained. Section three presents the neighborhood structure N_1, section four introduces the algorithm proposed that generates the configuration generation mechanism for neighborhoods, called Critical Path Approximation (CPA). Section five presents the experimental results. The final section draws conclusions about the information presented in the previous sections.

2 The Disjunctive Graph Model of the JSSP

Figure 1 shows the disjunctive graph model $G = (A, E, O)$ for a JSSP of 3x3 (three machines and three jobs). This disjunctive graph is formed by three sets. The operations

[1] Maximum completion time of the jobs.

set, *O*, is made up of the nodes *G*, numbered one to nine. The processing time appears next to each operation. The beginning and ending operations (I and * respectively) are fictitious, with processing times equal to zero. The set *A* is composed of conjunctive arcs, each one of these arcs unites a pair of operations that belong to the same job. The operations 1, 2, and 3 are connected by one of these arcs and therefore form job one. Jobs two and three are made up of the operations 4, 5, 6 and 7, 8, 9 respectively. Each arc of *A* represents a precedence constraint. For example, in job one, operation two must finish before operation three begins. Set *E* is composed of disjunctive arcs. Each arc that belongs to *E* unites a pair of operations that belong to the same machine. It can be seen that operations 1, 5 and 7 are executed by machine one and united by these arcs. Likewise, machines two and three execute the operations 3, 4, 9 and 2, 6, 8 respectively. Each machine forms a clique (a subset of *E* completely connected). Each arc of *E* represents a resources capacity constraint between a pair of operations that belong to the same machine. This type of constraint indicates that the machine cannot execute more than one operation in the same interval of time.

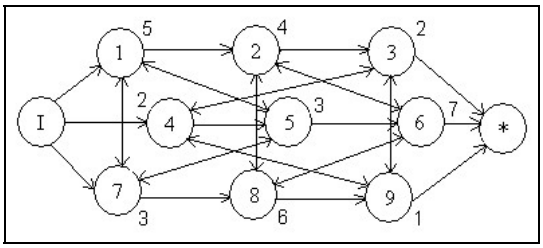

Fig. 1. Representation of a JSSP with three jobs and three machines using a disjunctive graph

3 The N_1 Neighborhood Function

The selection of the neighborhood structure strongly influences the performance of the metaheuristics [12] because the neighborhood has to be evaluated constantly. Consequently, this evaluation is the most critical one in the metaheuristics. In JSSP, the neighborhood N_1, introduced by Van Laarhoven et al. [4], has been used with great success to minimize the makespan. The evaluation of this neighborhood is made based on the set of solutions that are generated by the disjunctive graph G. Each solution (schedule) represents a digraph that does not contain cycles. In order to evaluate the neighborhood of the schedule, S, using N_1, it is necessary to find the CP of S. The neighbors in an N_1 neighborhood are generated by a permutation in a pair of adjacent operations that belong to the set of operations that form the CP of S. Figure 2 presents a schedule, S, for the JSSP shown in Figure 1, where the CP of S is demonstrated by a thicker line. The only pairs of adjacent operations that could swap in the CP are the compound pairs of operations that are executed by the same machine. For S in Figure 2, the pair of operations 1 and 7, which are executed by machine one, can be swapped. Likewise, the pair of operations 8 and 2 executed by machine two can be swapped. As one can see, in order to obtain a neighbor of S (permutation of a pair of

operations of S), it is necessary to calculate the CP of S. If N_1 is used in a local search, every time that a new S is formed, it is necessary to recalculate the CP to continue evaluating N_1.

Great advantages are gained by using N_1 in local searches [4]. For example, any permutation carried out in order to find a new schedule, S', obtains a feasible schedule, as long as $S' = f(S, N_1)$. It is also possible to obtain an S' with a makespan which is less than S, although this does not happen if the swap is made with a pair of operations that do not belong to the CP.

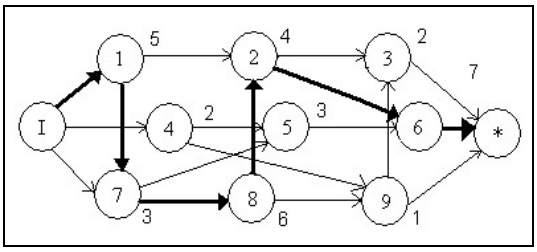

Fig. 2. A schedule of 3x3 where the operations that form the critical path are shown

An easily noted disadvantage of using N_1 is the necessity of calculating the CP constantly when N_1 is used in local searches.

The following section presents an alternate proposal to the use of N_1, which avoids repeatedly calculating the critical path of the job shop scheduling problem, consequently allowing for a reduction in the calculation time.

4 Critical Path Approximation Algorithm

The Critical Path Approximation (CPA) algorithm is based on three main lines of reasoning. The first line of reasoning is that for a defined schedule such as the one in Figure 2, the CP that is generated beginning from the fictitious operation I and ending with the fictitious operation *, is the same CP that is generated beginning with the fictitious operation * and ending with the fictitious operation I. That is, both critical paths generated in opposite directions are formed by the same operations due to being the same schedule. The second line of reasoning involves Equation 1, which indicates that upon generating the scheduling[2] starting with the fictitious operation I the start time s_i^I is obtained from the operation i that belongs to the critical path of the schedule. When this start time is added to the completion time c_i* (for the same operation), which is obtained by the scheduling beginning with the fictitious operation *, the sum is equal to the makespan (MS), where the MS is equal to the value of the CP [4] of the schedule. For $c_i^* = s_i^* + t$, s_i^* is the start time that is obtained from operation i when the scheduling starts with the fictitious operation * and t is the processing time of the operation i. The final line of reasoning that is considered for the generation of the CPA algorithm is that slack time between a pair of operations (i, j) that is part of the CP does not exist [4]. Considering the above-mentioned, it can be seen that the

[2] Start times of the operations of a schedule.

operations that fulfill both Equation 1 and the conditions of the last line of reasoning will have a high probability of belonging to the critical path of the defined schedule because they fulfill the conditions necessary for the pair (*i, j*) to belong to the CP. This means that there is neither slack between the pair (*i, j*) nor in either operation. More details are presented in CPM in [14].

$$MS = s_i^{\mathrm{I}} + c_i^{*} \tag{1}$$

The steps of the CPA algorithm are the following:

1. Take a schedule S as initial data.
2. Generate the scheduling S^{I} of S, beginning with the fictitious operation I.
3. Generate the S^{*} scheduling of S, beginning with the fictitious operation *.
4. Find the operations set Ω' that satisfies Equation 1.
5. In each machine of the JSSP, look for pair of subsequent[3] operations of Ω' that do not have slack time between them. The operations pair that satisfies this requirement forms the set Ω.

In order to obtain the scheduling of a schedule in CPA, the scheduling algorithm is used [13]. The time function of CPA is shown in Equation 2, where η is the number of operations obtained from *mxn*, *m* is the number of machines, and *n* is the number of jobs in the problem. The complexity of the algorithm is $O(\eta^{3/2})$.

$$f(\eta) = 2\eta^{3/2} + 2\eta^{1/2} \tag{2}$$

According to the three lines of reasoning, one could affirm that the set Ω includes all the operations that belong to the CP. As can be seen in the study done in [16], it is known that a permutation carried out in a pair of subsequent operations that do not have slack time between them results in a feasible schedule. Therefore, any permutation of a pair of subsequent operations that belong to the set Ω, will result in a feasible schedule.

The following is an example of how the set Ω is obtained, using the schedule of the 3x3 JSSP presented in Figure 2.

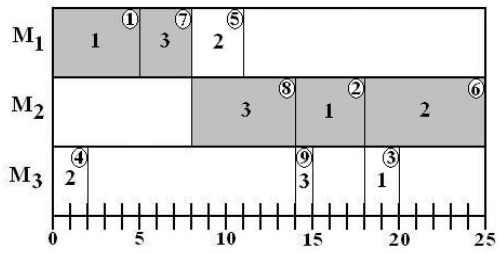

Fig. 3. Scheduling generated from Figure 2, starting with the fictitious operation I

[3] For the same machine, when operation *i* immediately precedes operation *j*, then *i*, *j* is a pair of subsequent operations.

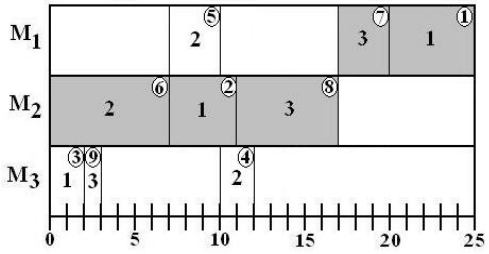

Fig. 4. Scheduling generated from Figure 2, starting with the fictitious operation*

Figure 3 presents the scheduling that is obtained starting with the fictitious operation I. Figure 4 presents the scheduling that is obtained starting with the fictitious operation*.

Table 1. Results obtained from the scheduling beginning with operation I, with operation*, and with the evaluation of the Equation 1

Operation	Scheduling		Evaluation of Equation 1
	s_i^I	s_i^*	$MS = s_i^I + c_i^*$
1	0	20	25
2	14	7	25
3	18	11	32
4	0	10	12
5	8	7	18
6	18	0	25
7	5	17	25
8	8	11	25
9	14	2	17

In Figures 3 and 4, the number of each operation is enclosed in a circle and the other number corresponds to the job where this operation is required. In these figures, the shaded gray areas correspond to the operations that fulfill Equation 1. The makespan of both schedulings is the same and equal to 25. Table 1 presents the start times of each operation obtained from Figures 3 and 4. In the table, the shaded regions represent the operations that fulfill Equation 1. As one can observe in Table 1, all the operations that form the set Ω' = {1, 2, 6, 7, 8}, belong to the CP (see Figure 2). One can see that all the operations that form the critical path are found in the set Ω'. These operations fulfill Equation 1, due to the fact that the MS obtained upon evaluating Equation 1 is the same as when evaluating for the CP (MS= 25). It can be observed in Figure 3 and 4 that the pairs of subsequent operations presented in the set Ω', do not have slack time; the pairs are (1, 7), (8, 2) and (2, 6). These pairs form the set Ω.

The following section presents a comparison of cost/performance of the CPA algorithm with a classical algorithm of polynomial time called CPM [11] (Critical Path

Method) which is used frequently [14] in the area of operations research for the planning and control of projects due to the high performance and low cost with which CPM works.

5 Computational Results

The proposed algorithm was proven with seven job shop scheduling problems benchmarks registered in the OR library [13]. Two problems of small size were used from this library, the FT06 with 6 machines, 6 jobs and 36 operations, and the FT10 with 10 machines, 10 jobs and 100 operations, both were proposed by Muth and Thompson. The problem of medium size LA40 with 15 machines, 15 jobs and 225 operations, proposed by Lawrence was used as well. Finally, four problems of larger size, proposed by Nakano and Yamada, were used which include YN1, YN2, YN3, and YN4, each one with 20 machines, 20 jobs and 400 operations.

In order to carry out the tests, a personal computer with a processor of 2.4 GHz and 640 MB in RAM was used.

The comparison of cost/performance of the CPA algorithm in the calculation of the operations pair that forms the set Ω (a set for each schedule generated randomly) was carried out with the CPM algorithm (Critical Path Method) [14].

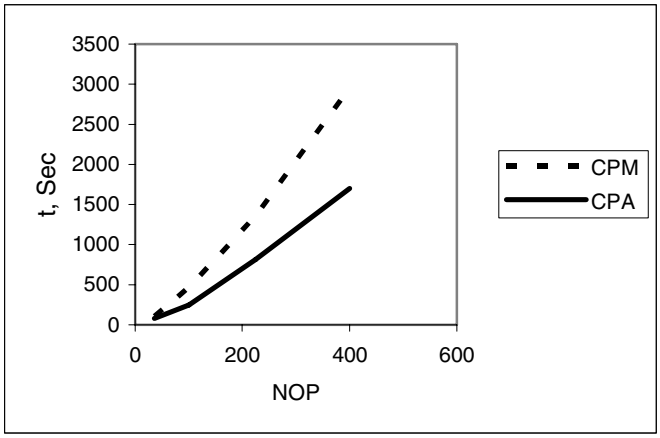

Fig. 5. Cost generated by the CPM and CPA algorithms upon calculating 800,000 critical paths (sets Ω, respectively) as the JSSP increases in size

Figure 5 presents the results obtained in the calculation of Ω and the critical path using the CPA and CPM algorithms respectively, for the problems FT06, FT10, LA40 and YN1. This figure shows the time of execution that is obtained for each algorithm when the number of operations in JSSP is increased. The time *t* is equal to the time that it takes the algorithm to calculate 800,000 critical paths (with CPM) or 800,000 sets Ω (with CPA). As can be observed, the cost of CPA is less than that of CPM. This figure also shows that when the number of operations (NOP) increases, the difference

in times (CPM vs. CPA) needed to obtain the 800,000 critical paths (or sets Ω) is more significant. This indicates that CPA works more efficiently than CPM as the problems of job shop increase in size.

Figure 6 presents the results obtained in the calculation of critical paths using the CPM algorithms and the results obtained in the calculation of the sets Ω using the CPA algorithms for the problem YN1. This figure shows the time of execution that is obtained when the Number of Critical Paths, NCP, (Number of sets Ω, NSΩ, respectively) increases for each algorithm. The *t* time is equal to the time that it takes the CPM algorithm to calculate a determined number of critical paths (for CPA, number of sets Ω). As shown, the cost of CPA is less than that of CPM. Also in this figure it can be observed that when NCP or NSΩ increases, the difference in times (CPM vs. CPA) in order to obtain these critical paths (sets Ω, respectively), is made more noticeable. This indicates that CPA will work better than CPM when NCP increases. For larger problems of JSSP, the metaheuristics that use the N_1 structure need a large NCP in order to be able to execute an acceptable search within the large solution space that these problems have[4].

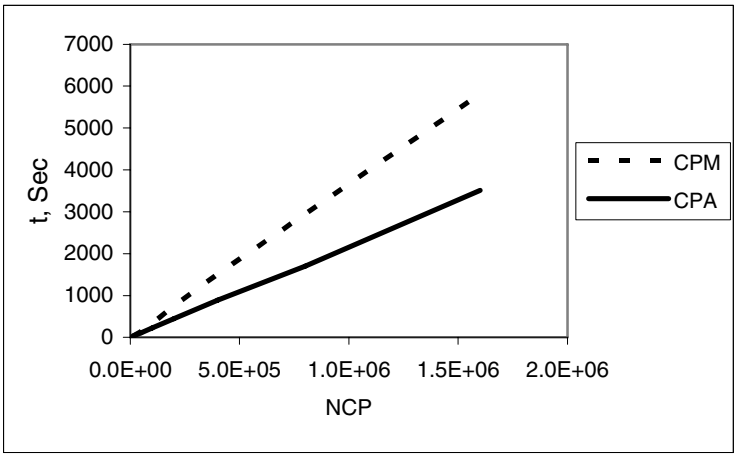

Fig. 6. Cost generated by the CPM and CPA algorithms in the problem YN1, with increasing NCP (NSΩ, respectively)

Table 2 presents a measure of the performance of the CPM and CPA algorithms. In order to evaluate this performance, the problems YN1, YN2, YN3, and YN4 were used. This table shows the average result for the calculation of 15,000 critical paths (15,000 sets Ω respectively). As can be seen upon studying the table, on the average, the NPCP (number of pairs of operations that belong to the critical path) obtained by CPM (obtained also by CPA in Ω), is between 34 and 38, depending on the problem. The MNP[5] (maximum number of pairs) that a symmetrical problem of JSSP is able to

[4] In a JSSP, the solution space is bound by $(m!)^n$.
[5] Maximum number of permutations, better known as the neighborhood size.

have is MNP= *n (n-1)*, where *n* is the number of jobs that the problem has; for problems YN1 to YN4, MNP= 380 pair of operations. As can be seen by studying the table, only a small part of the total pairs of operations belong to the CP (see NPCP in Table 2). In the same table, it is observed that there are a greater number of operations in the set Ω than those that exist in the critical path. These operations that are not part of the critical path make up an average of only 1% of the pairs of operations that are in Ω. The percentage of COP (Critical Operations Pairs) in the set is very high and almost constant, making up around 99% of the total number of operations pairs in Ω. It is important to clarify that in Ω, all the operations pairs that form the critical path are always present, plus a few pairs that are not part of the critical path. This means that the CPA algorithm has a very high performance in the approximation of the CP. This means that if CPA is used in metaheuristics that apply structures N_1, a probability of 99% exists that any given pair of operations chosen from Ω will pertain to the critical path. CPA allows full advantage to be taken of this neighborhood structure, but with a lower cost in generation time of the approximation of CP in order to evaluate N_1.. Table 2 also shows MinCP (Minimum number of pairs of critical operations found with CPM) generated in 15,000 tests and the MaxCP (maximum number of critical pairs found with CPM) generated in 15,000 tests for each problem of JSSP. This indicates that the number of pairs of operations that form the critical path will always be much lower that the MNP of a JSSP.

Table 2. Results obtained in 15,000 schedules generated randomly

Problem	Average			CPM	
	NPCP (CPM)	NPCP (CPA)	%COP Ω	MinCP	MaxCP
YN1	34	34	99	18	54
YN2	35	35	99.1	17	56
YN3	37	37	99	16	59
YN4	38	38	99	20	62

As a final test, in order to check the efficiency of the proposed configuration generation mechanism, the mechanism was implemented in the simulated annealing (SA) algorithm with restart presented in [8]. The cooling sequence of the simulated annealing is shown in Table 3. T_o is the initial temperature, t_f is the final temperature, C are the Metropolis cycles, and f is the cooling factor.

Table 3. Cooling sequence of the simulated annealing algorithm

Problem	T_0	T_f	C	F
FT10	64000	1	1000	0.98
LA40	200	1	750	0.99
YN1	64000	1	40000	0.98
YN2	64000	1	40000	0.98

Four problems were used to test the proposed mechanism. The results are presented in Table 4. The average and standard deviation, σ, reported in the table are the average of five executions of the SA algorithm. The SA algorithm is executed until a RE (Relative Error) of less than 2% is obtained. The longest execution time using the proposed mechanism is for the problem YN2, which was approximately 5 hours and 9 minutes. In this case, there is a RE of 1.98% when comparing the result with the best upper-bound found to this date, which is reported by Der and Steinhöfel [17]. The SA parallel algorithm of Der and Steinhöfel, which requires the calculation of the critical path by using the neighborhood N_1, took 16 hours and 30 minutes to obtain the upper-bound using a PC-cluster of 12 processors, each one of 550 MHz. In Table 4, for YN1, a RE of 1.7% was obtained in approximately 2 hours. Der and Steinhöfel [17] report a RE of 0.68% obtained in 16 hours for the same problem. In Table 4, for the FT10 problem, the optimum is obtained in less than 27 minutes. This result is obtained very quickly with respect to the time of 44 minutes, 55 seconds, reported in [8] when using the same algorithm that requires the calculation of the critical path. With the results shown in Table 4, it is proven that the configuration generation mechanism for neighborhoods proposed in this work, has a low cost, due to the short generation times needed to obtain good results for the evaluated problems with SA. It shows very good performance because it obtains results with low RE.

SA works with the neighborhood N_1 because pairs of operations that belong to the CP are permuted (99% of the time, see Table 2) using the CPA algorithm.

Table 4. Results obtained when the proposed configuration generation mechanism for neighborhoods is implemented in the SA algorithm with restart

Problem	t* sec	CS	Better MS*	Bad MS	Average MS	%RE*	σ
FT10	1585	930	930	937	931.4	0	2.8
LA40	1024	1222	1229	1234	1230.0	0.57	2.0
YN1	7659	885	900	909	904.2	1.70	2.9
YN2	18542	909	927	933	929.0	1.98	2.1

6 Conclusion

With the experimental results presented here, one can draw the conclusion that the CPA algorithm works more efficiently than the CPM algorithm with respect to cost because it obtains results more quickly. With respect to performance, CPA is competitive with CPM, because on the average, 99% of the total of pairs obtained in the set Ω belongs to the critical path. It is important to clarifying that Ω will always contain all the operation pairs that form the CP.

The use of the proposed Configuration Generation Mechanism for Neighborhoods in SA, when searching for a solution to instances of varying sizes of JSSP, enables results to be obtained efficiently with respect to cost/performance. This suggests that the proposed mechanism could work efficiently for any of the existing benchmarks of JSSP.

References

1. M.R. Garey, D.S. Johnson and R. Sethi, The complexity of Flow shop and Job shop Scheduling. Mathematics of Operations Research, Vol. I, No 2, USA, 117-129, May, 1976.
2. E.H.L. Aarts, P.J.M. Van Laarhoven, J.K. Lenstra, and N.L.J. Ulder, A computational study of local search algorithms for job shop scheduling, ORSA Journal on Computing 6, 118-125, 1994.
3. M.E. Aydin, M.E. and T. C. Fogarty, A distributed evolutionary simulated annealing algorithm for combinatorial optimisation problems, accepted for publication in Journal of Heuristics, 10 (3): 269-292, May 2004.
4. P.J.M. Van Laarhoven, E.H.L. Aarts and J.K. Lenstra. Job shop scheduling by simulated annealing. Oper. Res., 40(1):113-125, 1992.
5. T. Yamada and R. Nakano, 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.
6. T. Yamada, B. E. Rosen and R. Nakano, A simulated annealing approach to job shop scheduling using critical block transition operators, IEEE, 0-7803-1901-X/94, 1994.
7. K. Steinhöfel, , A. Albrecht , C.K. Wong, An Experimental Analysis of Local Minima to Improve Neighborhood Search Computers & Operations Research, 30(14):2157-2173, 2003.
8. M. A. Cruz-Chávez, J. Frausto-Solís, Simulated Annealing with Restart to Job Shop Scheduling Problem Using Upper Bounds, LNAI, ICAISC 2004, Vol. 3070, pp. 860 – 865, Springer-Verlag Pub., ISSN: 0302-9743, 2004.
9. S. Knust, Optimal conditions and exact neighborhoods for sequencing problems, Universität Osnabrück Fachbereich Mathematik/Informatik, D-49069 Osnabruck, Germany, January 1997.
10. M. A. Cruz-Chávez, J. Frausto-Solís and F. Ramos-Quintana, The Problem of Using the Calculation of the Critical Path to Solver Instances of the Job Shop Scheduling Problem, International Journal of Computational Intelligence, ENFORMATIKA, ISSN: 1304-2386, Vol. 1, No. 4, pp. 334-337, 2004.
11. S. Chanas, and P. Zielinski,., The Computational Complexity of the Critical Problems in a Network with Interval Activity Times, European Journal of Operational Research 136, 541-550, 2002.
12. H. Yildiz, Simulated Annealing & Applications to Scheduling Problems, Department of Industrial Engineering, Bilkent University, TR-06533, yildiz@ug.bcc.bilkent.edu.tr, 2000.
13. P. J. Zalzala, 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.
14. F. S. Hiller, and G. J. Lieberman, Introduction to Operations Research, ISBN: 0-07-113989-3, International Editions, 1995.
15. J. E. Beasley. OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society, Vol. 41. No. 11, 1069-1072, 1990. Last update 2003.
16. M. A. Cruz-Chávez, J. Frausto-Solís, J. R. Cora-Mora, Experimental Analysis of a Neighborhood Generation Mechanism Applied to Scheduling Problems, Proceedings of CERMA2006, IEEE-Computer Society, 26-29 Sep, México, 2006.
17. U. Der, K. Steinhöfel, A Parallel Implementation of Job Shop Scheduling Heuristics In Sørevik, T., Manne, F., Moe, R., Gebremedhin, A.H. (eds.), Proc. 5th International Workshop on Applied Parallel Computing, Springer-Verlag (LNCS 1947), pp. 215 - 222, 2001.