International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:1, 2007

The Problem of Using the Calculation of the Critical Path to Solver Instances of the Job Shop Scheduling Problem

Marco Antonio Cruz-Chávez, Juan Frausto-Solís, and Fernando Ramos-Quintana

***Abstract*— A procedure commonly used in Job Shop Scheduling Problem (JSSP) to evaluate the neighborhoods functions that use the non-deterministic algorithms is the calculation of the critical path in a digraph. This paper presents an experimental study of the cost of computation that exists when the calculation of the critical path in the solution for instances in which a JSSP of large size is involved. The results indicate that if the critical path is use in order to generate neighborhoods in the meta-heuristics that are used in JSSP, an elevated cost of computation exists in spite of the fact that the calculation of the critical path in any digraph is of polynomial complexity.

***Keywords*— Job Shop, CPM, critical path, neighborhood, meta-heuristic.

I. Introduction

THE Job Shop Scheduling Problem (JSSP) is currently considered to be one of the most difficult problems in the computer sciences, because it is [1] in the NP-complete class. The study of this problem is of supreme importance for the computer sciences because it is a practical problem in the manufacturing industry [2]. JSSP consists of a set m of machines and a set n of jobs, where each machine carries out an operation of each job. This problem attempts to find the optimum sequence of operations in each machine that allows for the minimization of the maximum completion time of the jobs (makespan).

The models that have been applied to JSSP are the disjunctive graph model [3], the disjunctive formulation model [3] and the satisfiability model [4], [5]. The disjunctive graph model is used most frequently due to the fact that it is possible to easily manipulate the meta-heuristics [6] applied to JSSP in which minimizing the makespan is desired. Generally, the meta-heuristics [7], [8], [9], [10], [11], [12] that have given the best results in JSSP involve neighborhood functions of the type N_1 [13]. The use of this type of neighborhood requires the calculation of the critical path (CP) and is one which works in a more efficient form because it considerably limits the search space in JSSP, while including the optimum solution (the minimum makespan) [13]. It is important to limit the solution space because JSSP contains a large number of solutions, which increase exponentially[1] when the size of the problem increases.

It is clear that the most time consuming calculation in a meta-heuristic which involves repeated local searches is, in fact, the evaluation of the neighborhood function which is executed in each local search. This evaluation involves a higher computational cost if neighborhoods of the type N_1 are used, because each evaluation of the neighborhood function requires the calculation of the CP. In this paper, experimental results are presented related to the computational cost generated when neighborhoods of the type N_1 are used in order to solve instances of the job shop scheduling problem. The concentration of this analysis uses the N_1 type of neighborhood because it has contributed a high efficiency to the meta-heuristics that have been applied to JSSP, but in turn present certain deficiencies, which should be considered.

The remainder of the paper is divided into the following sections. Section two presents the model of the disjunctive graph that is used for the calculation of the CP of a JSSP. Section three presents a brief description of the type of neighborhood N_1 that involves the calculation of the CP and possible deficiencies found. Section four presents the method used in this paper for calculation of the CP. Section five presents the experimental results, which show that the use of the CP as a main element of the local search in meta-heuristics requires a high computational cost.

II. The Disjunctive Graph Model

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

Manuscript received October 15, 2004. This work was supported by the Combinatorial Optimization Chair of the Department of Computer Science (DCS), ITESM, Campus Cuernavaca.

Marco Antonio Cruz-Chávez is with the Center of Investigation in Engineering and Applied Sciences, Autonomous University of Morelos State, CO 62270 Mexico (phone: 52-01-7773297084; fax: 52-01-7773297084; e-mail: mcruz@uaem.mx).

Juan Frausto-Solís is with the DSC, ITESM, Campus Cuernavaca, CO 62589 México (e-mail: juan.frausto@itesm.mx).

Fernando Ramos-Quintana is with the DSC, ITESM, Campus Cuernavaca, CO 62589 México, (e-mail: fernando.ramos@itesm.mx).

[1] For a JSSP where $m = n$ an upper bound of solutions exists = $(m!)^m$

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 arch 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 restriction indicates that the machine can not execute more than one operation in the same interval of time.

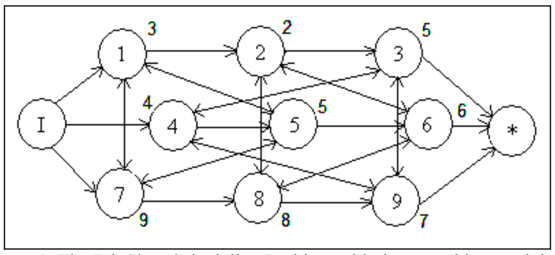

Figure 1. The Job Shop Scheduling Problem with three machines and three jobs.

III. The N_1 Neighborhood Function

The selection of the neighborhood function strongly influences the [14] performance of the meta-heuristics because the neighborhood has to be evaluated constantly. Consequentially, this evaluation is the most critical one in the algorithm. In JSSP the neighborhood N_1 introduced by Van Laarhoven et al. [13], 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 S schedule 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 an S schedule 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 9 and 4 executed by machine three 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 [15] by using N_1 in local searches. 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 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.

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

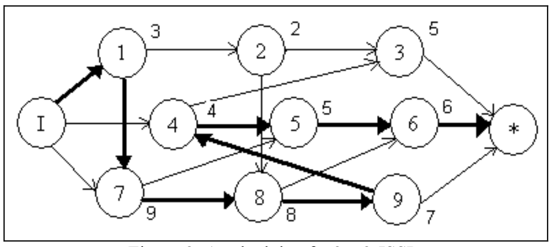

Figure 2. A schedule of a 3 x 3 JSSP.

IV. Critical Path Evaluation

In order to obtain a measure of the efficiency of evaluating N_1, the CPM [16] (critical path method) algorithm is used for the calculation of the CP. This is used throughout [17] the field of operations research for the planning and control of projects.

The CPM algorithm obtains initial data from a digraph, such as the digraph shown in Figure 2. Each arc of the digraph represents the activity carried out by the operation that precedes it. Each one of the activities is executed in a time, t, which is equal to the processing time of the operation that precedes it.

The CPM algorithm passes through each node of the graph twice, once forward (from the operation I to the operation *), and once backward (from the operation * to the operation I).

In the forward pass through the graph, each operation is assigned a Maximum Earliest Time ($MET = ET + t_{activity}$), where the operation's Earliest Time (ET) is the time at which the operation will begin if the preceding activities are carried out as quickly as possible.

In the backward pass through the graph, each operation is assigned a Minimum Latest Time ($MLT = LT - t_{activity}$), where the Latest Time (LT) of the operation is equal to the last moment in which the operation could begin without delaying the termination of any another operation.

After calculating the Maximum Earliest Time and Minimum Latest Time for each operation, the difference is calculated and the result is known as the slack. The CP is made up of all the operations whose slack have a value of zero.

V. Experimental results

The tests used in this paper to evaluate the computational cost generated in instances of JSSP when the CP is used were constructed in problems of small, medium and large size. The benchmarks used were taken from the OR library [18]. For small instances, the problem FT6 of 6 x 6 and the problem FT10 were used, both of Fisher and Thompson. For medium instances, the problem LA40 of Lawrence et al. was used, and for large instances the problem yn1 of Yamada and Nakano was used. The computer used for these tests was a Personal Computer with a processor of 550 MHz and memory of 125 MB. The tests consisted of generating feasible schedules randomly for each problem, and the CP for each schedule was calculated using the CPM algorithm. Every 5 minutes the number of critical paths (NCP) calculates was measured. The generation time of feasible schedules was subtracted, so that only the real time was counted in order to calculate the CP.

The results obtained in the generation of the NCP in several instances of JSSP in an interval of time of 5 to 20 minutes are shown in Figure 3. In this figure it can be observed that for small instances, NCP is very big. For example, for the problem FT10, around 179,800 CP were calculated in 20 minutes. As the problem increases in size, the number of calculated CP diminishes drastically. This is illustrated by the problem LA40, which calculated 28,185 CP in 20 minutes and the problem yn1 which calculated only 7,910 CP in 20 minutes.

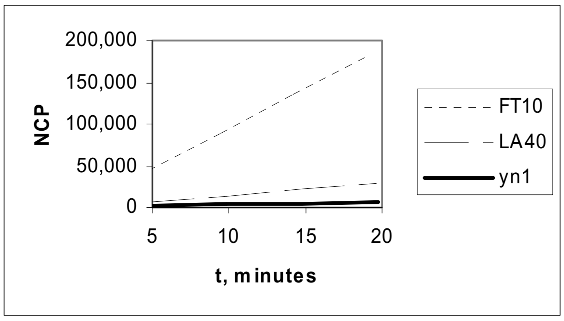

Figure 3. Cost of computation in the calculation of the CP for JSSP.

If it is taken into account that the search space of JSSP grows exponentially according to the size of the problem, only a small solution space can be evaluated when using N_1 type neighborhoods within a meta-heuristic. This is because the meta-heuristic has to recalculate the CP each time, which causes long delays. Consequently, in spite of the fact that the neighborhoods that evaluate the CP limit the solution space in an efficient way, these neighborhoods will cause the meta-heuristics to work slowly in order to obtain a solution near to the global optimum in instances of large JSSP. This is due to the fact that the calculation of the CP requires a much longer computation time as the size of the instance of the problem increases, as shown by the results in Figure 3.

Table 1 presents the slope of the straight line obtained upon evaluating the amount of NCP obtained in an interval of time *t*. One can note that the slope diminishes drastically upon increasing the instance size of JSSP. This indicates that for problems larger than 20 in size, the neighborhood that is required for evaluating the CP has very poor performance and for that reason tends to drastically slow the algorithms used to evaluate large instances of JSSP.

TABLE 1.
SLOPE OF EACH LINE OBTAINED UPON EVALUATING NCP VS. TIME

Problem	Slope
6	1093
10	149
15	23.4
20	5.75

VI. CONCLUSION

Although the complexity of CPM is polynomial [19] and methods that calculate the CP with lineal complexity also exist [20], it can be noted that for any method that calculates the CP, it is always necessary to pass through the digraph of JSSP at least once every time calculation of the CP is required. The consequences of this statement can be of catastrophic effect in meta-heuristics that apply neighborhood functions that require the calculation of the CP. This is even more so the case if the meta-heuristics used involve problems in which the solution space increases exponentially if the problem size increases, as in the case of JSSP. Although the neighborhood N_1 works efficiently in instances of JSSP classified as small size (problems of 6 x 6 and 10 x10) and medium size (problems 15 x 15), this is not the case for problems of large size (problems of 20 x 20). In these instances of large size, the algorithms work very slowly [7] and for that reason require very long periods of time (over 12 days) in order to find good solutions near the global optimum.

The previous conclusion opens an area of opportunity in the study of meta-heuristics that involve N_1 neighborhoods, in the sense of search for better algorithms that work from more efficient form in the construction of neighborhoods of type N_1 or in the search of improving the efficiency with which are built N_1. We took the second alternative (improving the efficiency of N_1). At the moment, tests of efficiency of JSSP for a new neighborhood are being carried out, which do not require calculation of the CP in problems of JSSP. Rather in these tests, the evaluation of any pair of operations is carried out in order to see if the pair of operations belongs to the CP. This procedure does not require passing through the digraph, it is only necessary to check the given pair of operations. It is thought that the neighborhood will work in a similar manner to N_1, by only swapping the pairs of operations that belong to the CP. This new procedure would take advantage of the benefits of using an N_1 neighborhood while eliminating the long calculation time.

REFERENCES

[1] Garey, M.R. and Johnson, D.S.: Computers and Intractability: A Guide of the Theory of NP-Completeness, W.H. Freeman and Co, New York, 1979.

[2] Schutten, J.M.J. Practical job shop scheduling, Annals of Operations Research, 83, 161-177, 1998.
[3] Conway, R.W., Maxwell, W.L. and Miller, L.W.: Theory of Scheduling. Addison-Wesley, Reading, Massachusetts 1967.
[4] Crawford, J.M. and Baker, A.B.: Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems, in Proc. of the 12th National Conf. on Artificial Intelligence, Austin, TX, 1092-1098, 1994.
[5] Frausto-Solís J. and Cruz-Chavez, M.A., A Reduced Codification for the Logical Representation of Job Shop Scheduling Problems, Lecture Notes in Computer Science, Springer-Verlag, ISSN: 0302-9743, Vol. 3046, 553 – 562 pp, May 14-17, 2004.
[6] Aarts E.H.L., Vaessens R.J.M. and Lenstra J.K., Job shop scheduling by local search. Memorandum COSOR 94-05, Eindhoven University of Technology, Department of Mathematics and Computing Science, P.O.Box 513, 5600 MB Eindhoven, 1994.
[7] Der, U. and Steinhöfel K., 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.
[8] Steinhöfel, K., Albrecht, A., Wong C.K. An Experimental Analysis of Local Minima to Improve Neighborhood Search Computers & Operations Research, 30(14):2157-2173, 2003.
[9] Taillard. E., Parallel tabu search technique for the job shop scheduling problem. ORSA Journal of Computing, (6):108-117, 1994.
[10] Yamada, T., Rosen B.E. and Nakano, R., A Simulated Annealing approach to job shop scheduling using critical block transition operators. In IEEE int. Conf. on Neural Networks (Orlando, Florida.) pp 4687-4692, 1994.
[11] Yamada, T., and Nakano, R., A Fusion Crossover and Local Search, Proceedings of the IEEE International Conference on Industrial Technology, 0-7803-3104-4, 1996.
[12] Cruz-Chavez, M.A., and Frausto-Solís, J., Simulated Annealing with Restart to Job Shop Scheduling Problem Using Upper Bounds, Lecture Notes in Computer Science, Springer-Verlag, ISSN: 0302-9743, Vol. 3070, 860 – 865 pp, June 7-11, 2004.
[13] Laarhoven, V., Aarts E.H., and Lenstra, J.K., Job Shop Scheduling by Simulated Annealing, Operational Research 40(1), 113-125. 1992.
[14] Yildiz, H., Simulated Annealing & Applications to Scheduling Problems, Department of Industrial Engineering, Bilkent University, TR-06533, yildiz@ug.bcc.bilkent.edu.tr, 2000.
[15] Wang, T. Y., and Wu, K. B., An Efficient Configuration Generation Mechanism to Solve Job Shop Scheduling Problems by the Simulated Annealing Algorithm, International Journal of Systems Science, Vol. 30, No. 5, 527-532, 1999.
[16] Kelley, J .E., Critical Path Planning and Scheduling Mathematical Basis, Operations Research, 9, 296 -320, 1961.
[17] Hiller, F. S. and Lieberman, G. J., Introduction to Operations Research, ISBN: 0-07-113989-3, International Editions, 1995.
[18] 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.
[19] Chanas, S. and Zielinski, P., The Computational Complexity of the Critical Problems in a Network with Interval Activity Times, European Journal of Operational Research 136, 541-550, 2002.
[20] Adams, J., Balas E., and Zawack, D., The Shifting Bottleneck Procedure for job shop scheduling, Management Science Vol. 34, No 3, 1988.