Experimental Analysis of a Neighborhood Generation Mechanism Applied to Scheduling Problems

Marco Antonio Cruz-Chávez[1], Juan Frausto-Solis[2], Jesús Roberto Cora-Mora[1]

[1]Engineering and Applied Science Research Center, UAEM
Av. Universidad 1001, Col. Chamilpa, 62270, Cuernavaca Morelos, MEXICO
mcruz@uaem.mx, jrcmricky@hotmail.com
[2]Department of Computer Science, ITESM, Campus Cuernavaca
Paseo de la Reforma 182-A, 62589, Temixco Morelos, MEXICO
Juan.frausto@itesm.mx

Abstract

This paper presents a neighborhood generation mechanism for the job shop scheduling problems (JSSP). In order to obtain a feasible neighbor with the generation mechanism, it is only necessary to generate a permutation of an adjacent pair of operations in a scheduling of the JSSP. If there is no slack-time between the adjacent pair of operations that is permuted, then it is proven, through experimentation that the new neighbor (schedule) generated is feasible.

1. Introduction

The job shop scheduling problem is classified in the Theory of Complexity as one of the most difficult problems inside the NP-complete class [1]. For this reason, Meta heuristics are frequently applied to search for the solution in instances of JSSP [2,3,4]. These algorithms have the characteristic of approaching the solution of the problem through searches in neighborhoods. Because of this, the development of new neighborhood generation mechanisms that are more efficient is important. This development will enable an increase in the efficiency of the Meta heuristic when the new mechanisms are used to search for the solution of the JSSP.

The Meta heuristics, such as simulated annealing and some procedures of genetic algorithms [5], take the longest amount of execution time in the local searches. In JSSP, the local search requires a generation mechanism for new schedules. In each iteration that the Meta Heuristic carries out, a local search improves the schedule. This is done using a neighborhood generation mechanism (local search) and the Meta heuristic specifies criteria that direct the search toward good solutions.

In this article, an efficient neighborhood generation mechanism for the JSSP is proposed. The mechanism requires only a permutation in a pair of operations that belong to the same machine. The approach used to select the pair of operations to be permuted in order to generate a new schedule is both simple and efficient.

After this introduction, section two explains the model of the disjunctive graph of the JSSP, which is used in the proposed neighborhood generation mechanism in order to generate neighbors (schedules). Section three describes the neighborhood generation mechanism and its origins. Section four presents the experimental tests that demonstrate the efficiency and feasibility of the proposed neighborhood generation mechanism. The final section states the conclusions that can be drawn from the presented work.

2. Disjunctive Graph Model

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 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 2, 6, 8 and 3, 4, 9 respectively. Each machine forms a clique (a subset of E which is 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 cannot execute more than one operation in the same interval of time.

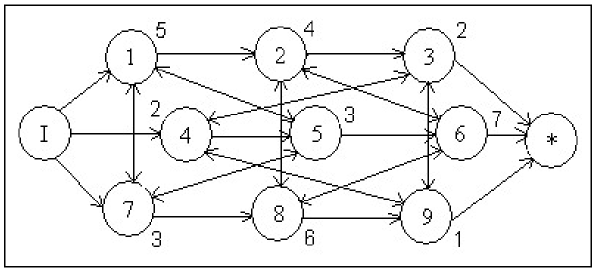

Figure 1. Representation of a JSSP with three jobs and three machines using a disjunctive graph

3. Neighborhood Generation Mechanism

In the development of the proposed neighborhood generation mechanism, the following previous knowledge and restrictions were considered.

a) The generation mechanism is applied in schedules generated from a disjunctive graph, for example, the schedule (digraph) that is presented in Figure 2. This schedule has Hamilton type routes [6]; there is one Hamilton route in each machine. The schedule is generated with the scheduling algorithm [7].

b) Only pairs of operations that are adjacent in the schedule and belong to the same machine can be permuted. For example, according to Figure 2, it is possible to permute the adjacent pairs of operations (1,7), (7,5), (8,2), (2,6), (4,9) and (9,3).

c) It is known that if a pair of operations that belongs to the critical path of a schedule is permuted, the result will be a feasible schedule [8]. In Figure 2, the critical path of the schedule is shown with a thicker line. In each adjacent pair of operations (*i*,*j*) that belongs to the critical path, slack-time does not exist [9]. That means that when the machine finishes operation *i*, it immediately begins operation *j*.

d) If a pair of adjacent operations that belong to the critical path is permuted, the result of this permutation can originate a better schedule if the objective function is to search for the minimization of the makespan of the problem. If the permutation is applied to a pair of operations that does not belong to the critical path of the schedule, the result cannot be a better schedule than the previous one [8]

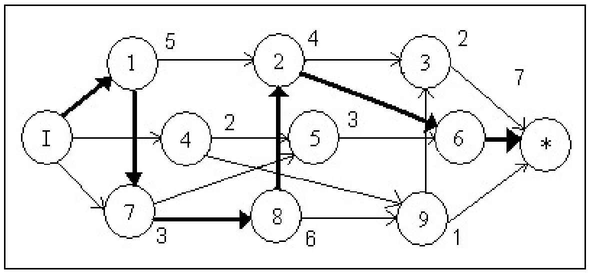

Figure 2. A schedule of 3x3 where the operations that form the critical path are shown by a thicker line

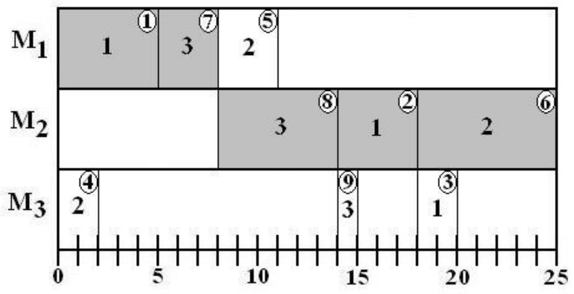

Figure 3. Gantt chart. Scheduling generated from Figure 2, starting with the fictitious operation I.

The generation mechanism starts with a feasible schedule, from which scheduling is generated and start times for each of the operations are found. The scheduling of the schedule in Figure 2 is shown in Figure 3 with a Gantt chart. In Figure 3, it can be seen which pairs of adjacent operations belong to the same machine. The number of the operation is shown enclosed in a circle.

The selection of the pair of operations to permute in the proposed generation mechanism was defined based on the points (c) and (d). What are not considered of (c) and (d), is that the pair of operations to be permuted must belong to the critical path. This is necessary in order to avoid the calculation of the critical path every time a search is conducted for a neighbor (schedule) of the neighborhood. With this considered, the only pairs of operations that are allowed to permute in the schedule are the pairs that do not contain slack-time, how happen with the pairs that belong to the critical path. But it is not necessary that the pair of operations belongs to the critical path. This selection forms a group of possible pairs of operations to permute. This group includes all the pairs of operations that pertain to the critical path, plus a few others. For example, the pairs of operations (without slack-time) that one could permute based on the scheduling in Figure 3 are (1,7), (7,5), (8,2) and (2,6). One can see, from examination of Figure 2, that all the pairs of operations that are in the critical path are also part of the group of pairs of operations that can be permuted. From the Figure 2 it can be seen that the adjacent pairs of operations that form part of the critical path and that one could permute are (1,7), (8,2) and (2,6). This is the principle that is used in the proposed neighborhood generation mechanism.

Hypothesis: In a feasible schedule, the permutation of an adjacent pair of operations that belong to the same machine, where these operations have no slack-time between them, results in a new feasible schedule.

Summary of the generation mechanism:

1. Generate a random schedule, using the disjunctive graph model.
2. Generate the scheduling with the scheduling algorithm [7].
3. Randomly choose the pair of operations (i,j) to permute in the scheduling.
    a. The pair of operations should belong to the same machine.
    b. The pair of operations should not have slack-time.
4. To permute the pair of operations.
5. Apply the algorithm of scheduling in the new feasible schedule
6. Return to 3, if desired, to generate another neighbor (schedule)
7. Finish

The following section presents the results of the test of feasibility (in experimental form) on the neighbors generated with the proposed generation mechanism. The stated hypothesis is proven experimentally.

4. Experimental results

In order to experimentally prove the hypothesis on which the proposed mechanism is based, eight problems of different sizes were taken from the OR-Library [10]. The problems used were: YN1, YN2, YN3, YN4 of 20 jobs and 20 machines; the problems LA40, LA38 with a size of 15 x 15; and the problems FT10 and FT06 with a size of 10x10 and 6 X 6 respectively. Table 1 presents the obtained results. Ten thousand tests for each problem were performed. Each test consisted of generating a schedule S randomly. The neighborhood of this schedule was revised to make sure the pair of adjacent operations did not contain slack-time. The configuration mechanism chose a pair of operations without slack-time. The pair was then permuted, and the resulting schedule S’ was checked for feasibility with the scheduling algorithm [7]. This was done for each pair of operations without slack-time in the schedule S. For example, for the problem YN1, the size of the neighborhood is 380 adjacent pairs of operations, some pairs with slack-time and others without slack-time between them. Each pair of operations without slack-time was evaluated. It was found that in every case, a feasible schedule was generated by the permutation of these adjacent pairs of operations.

For all the evaluated problems shown in Table 1, in no case was an unfeasible schedule generated. This indicates that for these problems, the hypothesis is proven. The result of this investigation presents a very efficient neighborhood generation mechanism because the only requirement is the permutation of an adjacent pair of operations that does not contain slack-time. Finding a pair of operations without slack-time in the schedule is not difficult. One can see in Table 1 that a high number of adjacent pairs of operations without slack exist on the average, compared with the size of the neighborhood of the problem. For the problems of 20x20 and 15x15, the percentage on the average is between 48 and 49% from the size of the neighborhood. For the problems of 10x10 and 6x6 presented in this paper, the percentage on the average is 54 and 53% respectively. This indicates that for approximately half of the neighbors belonging to the neighborhoods in this study, permuting an adjacent pair of operations without slack-time can generate feasible schedules.

Table 1. Experimental result from the generation mechanism of neighborhoods

Problem (Neighbor hood Size) (Number of Trials)	% of pairs of operations without slack that, if permutated, obtain a feasible schedule.	% of pairs of operations without slack that, if permutated, obtain an unfeasible schedule.	Average number of pairs of operations without slack-time
YN1 20x20 (380) (10000)	100	0	181
YN2 20x20 (380) (10000)	100	0	185
YN3 20x20 (380) (10000)	100	0	184
YN4 20x20 (380) (10000)	100	0	187
LA40 (210) (10000)	100	0	103
LA38 (210) (10000)	100	0	100
FT10 (90) (10000)	100	0	49
FT06 (90) (10000)	100	0	16

5. Conclusions

The search for more efficient neighborhood mechanisms will enable Meta heuristics to work in a more efficient manner. The proposed neighborhood generation mechanism requires only the permutation of an adjacent pair of operations without slack-time in order to obtain a neighbor (feasible schedule) of the neighborhood in study.

6. References

[1] C.H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, Dover Publications Inc., USA. ISBN 0-486-40258-4, 496 pp., 1998.

[2] 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.

[3] Aydin, M.E. and Fogarty, T. C. (2004), “A distributed evolutionary simulated annealing algorithm for combinatorial optimization problems”, *Journal of Heuristics*, 10 (3): 269-292, May 2004.

[4] M.A. Cruz-Chavez and J. Frausto-Solís, “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.

[5] T. Yamada and R. Nakano, “A Genetic Algorithm with Multi-Step Crossover for Job-Shop Scheduling Problems”, *Genetic Algorithms in Engineering Systems: Innovations and Applications*, No. 414, IEE, 1995.

[6] M.R. Garey and D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, ISBN 0-7167-1045-5, W.H. Freeman and Company, USA, 340 pp., 1991.

[7] R. Nakano and T. Yamada, “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.

[8] P. J. M. Van Laarhooven, E. H. L. Aarts, and J. K. Lenstra, “Job Shop Scheduling by Simulated Annealing”, *Operations Research*, Jan-Feb, 40(1), 113-125, 1992.

[9] F. S. Hiller, and G. J. Lieberman, *Introduction to Operations Research*, ISBN: 0-07-113989-3, International Editions, 1995.

[10] 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.