Relaxation of Job Shop Scheduling Problem using a Bipartite Graph

Marco Antonio Cruz-Chávez[1], Alina Martínez-Oropeza[1], E. Y. Ávila Melgar, Rafael Rivera López[2]

[1]*CIICAp, Universidad Autónoma del Estado de Morelos*
Avenida Universidad 1001. Col. Chamilpa, C.P. 62209. Cuernavaca, Morelos.
MÉXICO
{mcruz, alinam,erikay}@uaem.mx
[2]*Departamento de Sistemas y Computación, Instituto Tecnológico de Veracruz*
Calzada Miguel Ángel de Quevedo 2779, Veracruz, México
rrivera@itver.edu.mx

Abstract

This paper addresses the Job Shop Scheduling Problem (JSSP). Basic constraints are established and it is modeled by a disjunctive graph. The model was mapped to Unrelated Parallel Machines Problem through a bipartite graph. An analysis of constraints is made in both problems to perform a relaxation of the manufacturing problem. Conducting the relaxation of JSSP, it is that any operation can be assigned to any machine, in addition, the precedence constraint between two operations longer applies. The application of approximated bipartite graph model is shown as a new alternative model to represent the JSSP problem, it may be an option to work with this type of problems instead of the disjunctive graph model.

1. Introduction

The Job Shop Scheduling Problem, well-known as JSSP has been considered as one of the most important problems in manufacturing and optimization area, as well as the efficient resources management is vital in businesses. Due to its complexity, JSSP has been one of the most studied problems during the last four decades; therefore, it is the problem that has achieved more progress in the Scheduling area, and serves as reference to compare the techniques used in several problems classified as difficult, such as Traveling Salesman Problem, among others. In addition, it has been classified as NP-Hard by the Computational Complexity Theory [1], also, it is known as one of the most difficult problems to solve in this classification. The class NP-Complete can be defined alternatively as the NP and NP-Hard intersection. A feature of NP-Hard is that there is unknown a deterministic algorithm in polynomial time to solve problems within this classification [23]; hence, many researchers have been attracted trying to solve them using several optimization methods as Branch and Bound[11], Genetic Algorithms [12], Simulated Annealing [13], Ant Colony [3], among others. Some of these algorithms have been designed specifically to solve certain instances of the problem.

Several approaches to solve JSSP have been proposed using different models. Disjunctive graphs model is used in most of cases [14], constraints satisfaction [15], and integer linear programming [14]. Small, medium and large instances of JSSP can be solved or find approximated solutions to the global optimum being modeled by a disjunctive graph using metaheuristics, which apply local search methods. Small instances of JSSP modeled with integer linear programming can be solved using exact algorithms. The search methods of heuristic type can be very fast to find a feasible solution to an instance of JSSP, but unfortunately there is no assurance that the optimal solution was found. However, search methods can provide an starting point. Methods based in satisfiability [9, 15, 17] and priority rules [20] are some examples of search methods. The Bottleneck method [18, 19] is a special type of search method, it has better performance than others, but only for small intances. Traditionally, the integer linear programming, better known as ILP, and packages of Binary Decision Diagrams (BDD) [10], are used to obtain exact solutions for many automation problems. The Scheduling problem has been addessed to any of these methods by researches. Formulations based on ILP are popular for scheduling although the schedulers based on BDD also offer attractive solutions.

A deficiency of solutions based on BDD is the binary diagram size, which could become too big for large instances [21]. It is noteworthy that most of these methods require a feasible initial solution at the begining of the process [9].

The researching reach is to develop an algorithm with the Ant Colony approach, which will be parallelized using message passing in C language and the Message Passing Library (MPI), in this way it can be implemented in a cluster structure. The proposed algorithm can solve the JSSP relaxed. Efficiency and

effectiveness can be determined using test problems randomly generated. The performance of the algorithm proposed will be tested using these benchmarks, which will be compared with solutions of local search algorithms previously tested, such as simulated annealing, and an exact method (Simplex).

Tests will be realized according to stadistic area proposal for its complete evaluation and analysis. That is, the same benchmarks will be used to test the three algorithms above mencioned, thus, calculate effectiveness, efficacy and relative error of the algorithm proposed and make a direct comparison among them.

This paper presents an alternative to represent the JSSP relaxed using a bipartite graph, which can be solved as a P-type problem, it gives a very attractive posibility to test exact algorithms and metaheuristics applied in literature to solve bipartite graphs [22], and realize them an effectiveness and efficacy evaluation for JSSP. The challenge to find a feasible solution rapidly is mapping a JSSP instance as an unrelated parallel machines problem, where problem constraints are relaxed in order to represent it by means of a bipartite graph.

This paper is divided into the following sections: JSSP definition, where it is explained the general definition of the problem, as well as the aim and basic constraints. In the second section, it is explained the JSSP model by a disjunctive graph giving a solution to a 3x3 instance. Third section approaches the bipartite graph definition and its features; in addition, a representation of the instance used is realized for JSSP by a bipartite-type graph. In fourth section it is explained the problem relaxation to be represented by a bipartite graph. Finally, conclusions are mentioned as well as future work.

2. JSSP Definition

The problem is defined as a set of machines, and a set of jobs, where each job has certain number of operations that must be processed during a determined time in a given machine without interruptions. Each machine can process only one operation in an instant of time. The Classic problem of JSSP can be defined as a finite set J of n jobs $\{J_j\}_{1 \leq j \leq n}$, which must be processed in a finite set M of m machines $\{M_k\}_{1 \leq k \leq m}$. Each job is seen as a machines sequence where it can be processed. The processing of a job J_j within a machine M_k is known as O_{jk}. The operation O_{jk} requires the exclusive use of a machine M_k, where the processing time of each operation does not allow interruptions and is known as t_{jk} [3]. A problem solution for an instance of 3 x 3 is presented in *Table 1*. It is noteworthy that the set of solutions for this problem is given by $(n!)^m$; therefore, for an instance of 3 x 3 we would have 216 possible solutions, where, it is important to say that not all of them are feasible solutions, due to some of them violate precedence constraints.

Table 1. JSSP solution for a problem with 3 machines, and 3 jobs with 3 operations each one.

Jobs	Operations Machines (Processing Time)		
	1	**2**	**3**
1	1(1)	2(2)	3(1)
2	1(3)	3(1)	2(3)
3	2(2)	1(2)	3(3)

The solution proposed for this problem (*Table 1*), can be represented by a Gantt Graph, which represents both the scheduling of operations, and the units of time consumed in process (*makespan*).

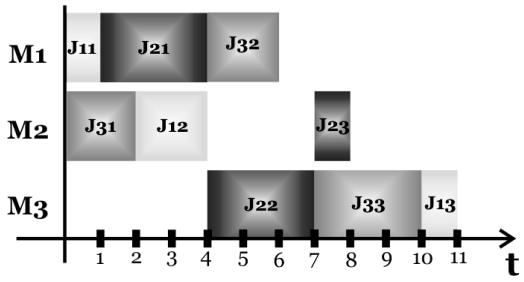

Fig. 1. One of the 216 possible solutions for a problem of 3 machines and 3 jobs.

An important feature of JSSP is that, it is no necesary that the starting machine begins with the first job, but any machine can starts or ends, because the processing order of the jobs is a problem constraint [6].

2.1. Constraints

To perform the jobs scheduling in a manufacturing workshop it is necessary to define certain constraints, which ensure the operations integrity, and the process effectiveness. Constraints considered in this problem are shown below.

- Only a job can be processed by a machine in a period of time.
- Each machine can handle at most one operation at a time.

- A job must not be processed twice.
- It does not exist constraints among operations of different jobs.
- An operation assigned can not be interrupted.
- A job is composed of *i* operations.
- There is a precedence constraint between operations within the same job, which as processing times, are data known.
- There is a resource capacity constraint, which is known as the operations sequence within a machine, and they are data known.
- The jobs operations have the same priority
- It is not specify neither release nor due dates.

The Scheduling of feasible jobs can be gotten permuting the order of the operations in each machine, taking care not to violate the constraints specified [6].

3. Disjunctive Graph Model

The JSSP problem can be described by a disjunctive graph $G = (V, C \cup D)$, where V is the set of operations in each job join two special nodes *source* and *destination*, which indicate the beginning, and the end of the scheduling. C is the set of conjunctive edges representing the operations sequence (precedence constraint). D is the set of disjunctive edges (cliques) representing the set of operation that must be processed in the same machine [2]. The processing time for each operation is defined; in addition, the processing time of the last job (C_{max}) depends directly of the operations sequence within each machine.

An instance representation of 3 x 3 (*table 1*) is shown in figure 2 by a disjunctive graph.

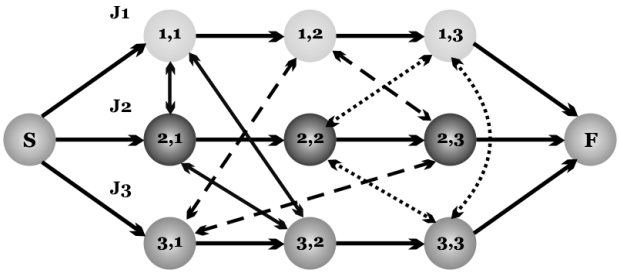

Fig. 2. JSSP representation using an instance of 3 x 3 by a disjunctive graph.

A disjunctive edge can be directed by any of its two endings. The direction of disjunctive edges is fixed when the scheduling is realized; in this way, the operations sequence in each machine is gotten. When the operations sequence is obtained, the disjunctive edges change into conjunctive edges (*Figure 3*).

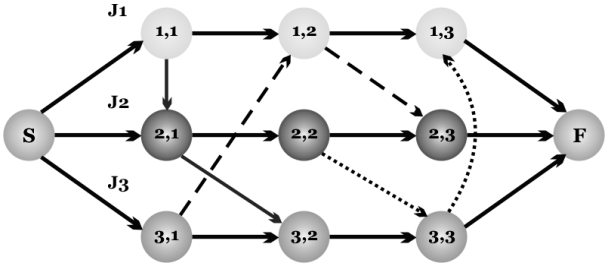

Fig. 3. JSSP problem solution with 3 jobs and 3 machines.

The operations scheduling as mentioned above, can be represented by a Gantt graph. *Figure 1* shown the final scheduling for this solution.

4. Bipartite Graph Model

A bipartite graph is an undirected graph with the feature of a set of vertex V, which can be divided into two disjoint subsets $G = \{V_1 \cup V_2, A\}$, so that each vertex of each subset is conected by an egde; that is to say, an edge connects an element of subset V_1 and another of V_2, taking into account that there should be no adjacencies between elements in the same subset [5]. The basic features of a bipartite graph are listed below.

- Subsets V_1 and V_2 are disjoint and not empty
- Each edge of A joins a vertex of V_1 with one of V_2
- There are no edges joining two elements of V_1; similarly to V_2.

Taking into account the features mentioned above, JSSP problem was represented by a bipartite-type graph (*Figure 4*), it was realized because it already exist linear programming algorithms that solve bipartite graphs, such as Simplex method.

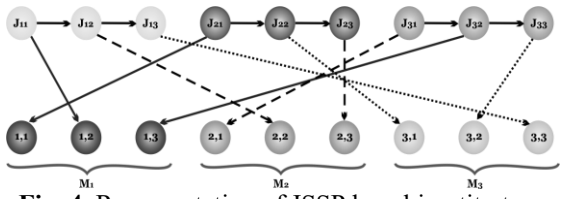

Fig. 4. Representation of JSSP by a bipartite-type graph.

According to this representation (*Figure 4*), is said to be a bipartite-type graph, due to the prevailing precedence constraint represented between operations of each job. For this representation, it is necessary to relax the problem constraints in order to represent a genuine bipartite graph, and in this way, trying to solve it by linear programming algorithms.

5. JSSP Relaxation

To realize the relaxation of JSSP, each constraint was analyzed, concluding that some of basic constrints can be implicitly taken, so that, not necessarily have to be represented graphically; therefore, in general, the bipartite graph representation for an instance of JSSP is as follow (*Figure 5*).

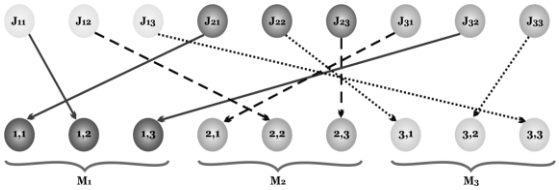

Fig. 5. Representation of JSSP relaxed by a bipartite graph for an instance of 3 x 3

Taking into account this representation where the relaxation was realized, a mapping to unrelated parallel machines problem (UPMP) is obtained. According to literature, the UPMP problem can be defined as a set of *n* independet jobs that need to be schedule in *m* unrelated parallel machines, so that, the objetive function that minimize the total completion time of all jobs is accomplished; therefore, it must be considered that a machine can not process more than a job at a period time, and jobs assigned can not be interrupted.

The difference between UPMP and JSSP is that in UPMP, a job can be processed in any machine and any position. The processing time P_{ij} of a job *j* within a machine *i* is a function of the machine where it was assigned, this is a fundamental feature of the problem being unrelated machines, the processing time of a same job varies from one machine to another one, due to its resources and capacities could be different.

According to this definition, for an instance used for JSSP, it would have an undirected bipartite graph (*Figure 6*), where each job can be processed in any machine; therefore, it would relax the precedence constraint and it could be represented only the resource capacity constraint.

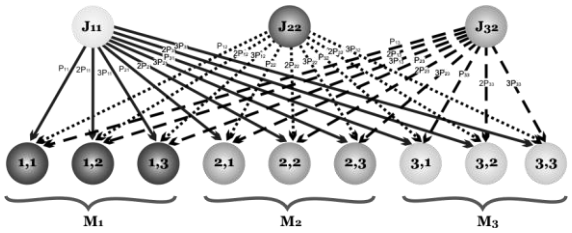

Fig. 6. Undirected bipartite graph for an instance of 3 x 3 applied to unrelated parallel machines problem.

In the bipartite graph shown above, it is used kP_{ij} to identify where each job will be processed. Where *k* is the position in machine *i* where job *j* will be processed. Therefore, in case of an undirected graph, the job can be assigned to any machine in any position

.To represent the solution used in JSSP as a bipartite graph, it is necessary to pass from the undirected graph (*Figure 6*) to a directed graph, that is to say, each job must be assigned to any machine in any position, taking into account both the problem constraints and the bipartite graph features, so that, a solution representation would be as follows (*Figure 7*).

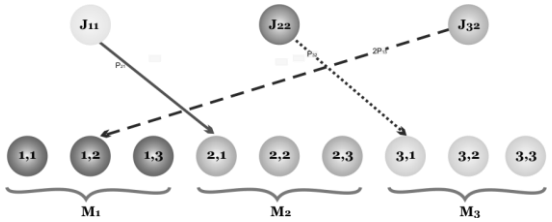

Fig. 7. Directed Bipartite Graph for an instance of 3 x 3, each job will be assigned to any machine in k-eth position.

Taking this solution for the instance proposed and according to processing times showed in *Table 1*, it would get the scheduling as follows (*Figure 8*) using a Gantt chart.

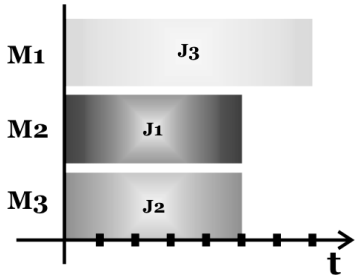

Fig. 8. An instance of 3 x 3 scheduled to the unrelated parallel machines problem.

6. Conclusion

The Job Shop Scheduling Problem (JSSP) is one of the most studied problems within Optimization area, which has attracted the attention of many researchers around the world due to its difficulty, and its classification as an intractable problem.

During the last years, researchers have focused on the use of hybrid algorithms, and heuristics trying to solve this problem.

Different models have been used to represent the problem, such as disjunctive graph model. A JSSP relaxation was realized in this paper, wherewith the mapping of the unrelated parallel machines problem was achieved by means of a bipartite graph.

7. Future Work

Develop an algorithm capable to solve a bipartite graph without relaxation for JSSP. Comparing results of the algorithm mentioned before against an algorithm based on disjunctive graph model and another of linear programming.

8. References

1 Garey M., Johnson, D. S. and SEIT, R.: The Complexity of Flow Shop and Job Shop Scheduling, in Mathematics of Operation Research, Vol. 1, No. 2. 1976. pp. 117-129.

2 Yamada Takeshi and Nakano Ryohei. Genetic Algorithms for Job-Shop Scheduling Problems. Proceedings of Modern Heuristic for Decision Support, pp. 67-81, UNICOM Seminar, London 1997.

3 Yamada Takeshi and Nakano Ryohei. Chapter 7: Job-Shop Scheduling. Genetic Algorithms in Engineering Systems. IEE Control Engineering Series 55. The Institution of Electrical Engineers. ISBN: 0 85296 902 3. pp. 134-160. 1997

4 Papadimitrious Christos H and Steiglitz Kenneth. Combinatorial Optimization. Algorithms and Complexity. ISBN 0-486-40258-4. pp. 163 – 166. New York. 1982.

5 Rosen Kenneth H. Discrete Mathematics and its Applications. Fifth Edition. Ed. Mc. Graw Hill. International Edition 2003. pp. 549.

6 Van Laarhoven Peter J. M., Aarts Emile H. L. and Lenstra Jan Karen. Job Shop Scheduling by Simulated Annealing. Operation Research, Vol. 40, No. 1. Published by INFORMS. 0030-364X/92/4001-0113. pp. 113-125. 1992.

7 Morikawa, K., Furuhashi, T. y Uchikawa, Y. Single Populated GA and its Application to Job Shop Scheduling. Proceedings of the 1992 International Conference on Industrial Electronics, Control, Instrumentation and Automation. Vol. 1-3, 1992, ch. 286_vl.003, pp. 1014-1019.

8 Peña Víctor y Zumelzu Lillo. Estado del Arte del Job Shop Scheduling Problem. Disponible on-line: http://www.alumnos.inf.utfsm.cl/~vpena/ramos/ili295/ia-jobshop.pdf. 2006.

9 Frausto-Solis J. and Cruz-Chávez M. A., A Reduced Codification for the Logical Representation of Job Shop Scheduling Problems, Lecture Notes in Computer Science, Springer Verlag Pub., Berlin Heidelberg, ISSN: 0302-9743, Vol. 3046 (4), pp. 553 - 562, 2004.

10 Ogrenci-Memik Seda and Fallah Farzan. Accelerated SAT-based scheduling of control/data flow graphs Computer Design: VLSI in Computers and Processors, pp395–400, Proc IEEE, 16-18 Sept. 2002.

11 Carlier, J. And Pinson, E.: An algorithm for Solving the Job-Shop Problem, in Management Sciences, Vol. 35, No.2. 1989. 164-176.

12 Zalzala,P.J. and Flemming: Genetic Algorithms in Engineering Systems, in A.M.S. Inst. of Electrical Engineers 1997.

13 Yamada, T. And Nakano, R.: Job-Shop Scheduling by Simulated Annealing Combined with Deterministic Local Search, In Metaheuristics Int. Conference, Colorado, USA, (1995) 344-349.

14 Conway, R.W., Maxwell, W.L and Miller, L.W.: Theory of Scheduling. Addison-Wesley, Reading, Massachusetts 1967.

15 Smith, S.F. and Cheng, C.C.: Slack-Based Heuristics for Constraint Satisfaction Scheduling, in Proc. Of the 11th National Conf. on Artificial Intelligence, Washington, D.C., (1993) 139-145.

16 Cruz-Chávez M. A., Rivera-López R., A Local Search Algorithm for a SAT Representation of Scheduling Problems, Lecture Notes in Computer Science, Springer-Verlag Pub., Berlin Heidelberg, ISSN: 0302-9743, Vol.4707, No. 3, pp. 697-709, 2007.

17 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, (1994) 1092-1098.

18 Adams, E., Balas, E. and Zawack, D.: The shifting Bottleneck Procedure for Job Shop Scheduling, in Management Science, Vol. 34, No. 3. 1988. 391-401

19 Schutten, M.J.: Practical Job Shop Scheduling, in Annals of Operations Research, Vol. 83, (1988) 161-177.

20 Yoshida T. and Touzaki, H., A Study on Association amount Dispatching Rules in Manufacturing Scheduling Problems, IEEE, 0-7803-5670, 1999.

21 Campailla Alexis, Chaki Sagar, Clarke Edmund, Jha Somesh and Veith Helmut. Efficient Filtering in Publish-Subscribe Systems udisng Binary Decision Diagrams. 23th International Conference on Software Engineering (ICSE'01). ISBN. 0-7695-1050-7. Canada. 2001

22 Cruz-Chávez M.A., Juárez-Pérez F., Ávila-Melgar E. Y., Martínez-Oropeza A., Simulated Annealing Algorithm for the Weighted Unrelated Parallel Machines Problem, Electronics, Robotics and Automotive Mechanics Conference, CERMA2009, IEEE-Computer Society, ISBN 978-0-7695-3799-3, pp 94-99, September 22 - 25, México, 2009.

23 Pinedo M., Scheduling Theory, Algorithms, and Systems, Third Edition. Springer Science-Business Media, LLC. ISBN: 978-0-387-78934-7, e-ISBN: 978-0-387-78935-4.