Scheduling to Minimize the Makespan on Identical Parallel Machines: An LP-Based Algorithm

Ethel Mokotoff

University of Alcalá
Plaza Victoria, 3 - 28802 - Alcalá de Henares - Spain
Tel: 34 91 8854202, Fax: 34 91 8854239
ehemf@alcala.es

Abstract

Polyhedral Combinatorics approaches have turned out to be successful computational tools in many hard Combinatorial Optimization problems. We present an approximation algorithm based on linear programming formulations with binary decision variables which are a kind of assignment variables for the classical deterministic scheduling problem of minimizing the makespan on identical machines. The problem is known to be NP-hard in the strong sense. The structure of the corresponding polytope is analyzed, and the strong cutting planes are identified. Computational results show that, in all tested cases, the problem can be solved exactly.

Keywords: Optimization, Polyhedral Combinatorics, Scheduling.

1 Introduction

In the classical Parallel Machines Scheduling (PMS) problem, there are n jobs and m machines. Each job needs to be executed on one of the machines during a fixed processing time, without pre-emption.. So, the aim is to find the schedule that optimizes certain performance measure.

Many real life problems can be modeled as PMS ones. On production lines, it is common to find more than one machine of each kind carrying out the production tasks. Other examples are docks - ships, teachers - students groups, hospital assistance - patients, etc. The PMS also constitutes an important issue within the field of Computer Science, due to the increments in use of share time systems, or multiprocessor computers, which require efficient procedures for assigning tasks. Many other

problems could be mentioned when we speak about scarce resources, or machines, dedicated to the production of some goods, or jobs.

The aim of this article is to present a new approximation algorithm based on Linear Programming (LP) formulations with binary decision variables that are a kind of assignment variables for the PMS problem. We have given to polyhedral approaches a try-out because Polyhedral Combinatorics [15] have turned out to be practically successful computational tools in many hard Combinatorial Optimization (CO) problems like the classical Travelling Salesman Problem [14] , [2], the problem of scheduling to Minimize the Total Weighted Completion Time [8], etc. Using an algorithm that combines heuristics and cutting planes techniques, we analyze the structure of the corresponding polytope and identify the strong cutting planes. The core of the algorithm is a cutting plane method, using the simplex method to solve the LP relaxations. Computational results showed that, in many cases, the problem can be solved exactly.

In the next section, the problem statement is presented and its complexity is analyzed. Section 3 describes the proposed algorithm. Section 4 discusses the computational experience. We conclude, in Section 5, with a summary discussion on research directions.

2 Problem Statement and Complexity Results

2.1 Notation, Formulation and Assumptions

We will use this notation in what follows:

J_i : job i, $i \in N$=$\{1, ..., n\}$

M_j : machine j, $j \in M$=$\{1, ..., m\}$

p_i: processing time of job J_i

p_{ij}: processing time of job J_i on machine M_j

C_i: completion time of job J_i

C_{max}: makespan, the maximum completion time of all jobs J_i,
$C_{max} = max\{C_1, C_2, ..., C_n\}$

x_{ij}: assignment variable

Consider a set J of n jobs J_i (i=1,...,n) to be processed, each of them on one machine, on a set M of m machines M_j (j=1,...,m). All the jobs can be processed on any of the m machines. We consider identical machines models, for which the processing times of each job, p_i, are independent of the machine processing it.

The objective is to find an appropriate allocation of jobs from set J to machines from set M, that would optimize a performance criterion. We are interested in the maximum completion time criterion, C_{max}, called *makespan.* However, there are many other performance criteria to be considered when solving scheduling problems ([4] presents a complete classification).

We will use the three parameters notation $\alpha/\beta/\gamma$, introduced by [6], where the first field specifies the machine environment, the second, the job characteristics, and the third refers to the chosen optimality criterion. The problem at hand, is then denoted by $P//C_{max}$, where P represents identical parallel machines, the jobs are not constrained, and the objective is to produce a minimum length schedule.

A Mixed Integer Programming (MIP) formulation of the minimum makespan problem follows:

$$\begin{aligned} & Min.\ y \\ s.t.\ & \sum_{j=1}^{m} x_{ij} = 1,\ \ 1 \leqslant i \leqslant n \\ y\text{-}& \sum_{i=1}^{n} p_i x_{ij} \geqslant 0,\ \ 1 \leqslant j \leqslant m \end{aligned}$$

where the optimal value of y is the $C_{\max}$ and $x_{ij} = \begin{cases} 1 \text{ if the job } i \text{ is assigned to machine } j \\ 0 \text{ if the job } i \text{ is not assigned to machine } j \end{cases}$

It is enough to know just the optimal assignment, because the C_{max} will be the same for any permutation of the jobs assigned to each machine. This formulation has $m+n$ restrictions and $mn+1$ variables, being the x_{ij} variables binary and the C_{max} variable integer.

2.2 $NP-hardness$ of $P//C_{max}$

Even though the $P//C_{max}$ problem has been intensively investigated by a number of researchers over the past 40 years, exact polynomial algorithms have not been found yet. Furthermore, it is possible to verify, by reduction to the partition set problem, that, even for $n = 2$, this problem is *NP-hard* [12]. The seminal papers [3] and [11] lead one to suspect that it is unlikely that there exist efficient optimization algorithms to solve it (unless $P = NP$).

3 Approximate Algorithm Proposed

3.1 Background of the Algorithm

The proposed algorithm presents a similar cutting plane scheme followed by [7], [14], [1], [9] and [17]. The main part of these algorithms is a separation procedure that checks whether or not a given point, the optimum solution of the last LP relaxation solved, is feasible for the CO problem. In the latter, at least one new inequality is

added.

Since, even for moderately sized PMS problems, the number of binary variables and inequalities is rather large, we have to be as economical as possible in order to keep the problem of manageable size. We have implemented a special preprocessing procedure to accomplish it.

The above mixed integer programming formulation can be represented by the region

$$F = \left\{ (x,y) : x \in B^{n\times m}, y \in \Re_+ : \sum_{j=1}^{m} x_{ij} = 1,\ \forall i; y - \sum_{i=1}^{n} p_i x_{ij} \geqslant 0,\ \forall j \right\},$$

where $B^{n\times m}$ is the set of nm-dimensional binary vectors.

Consider the polytope P related to F,

$$P = \{(x,y) : x \in \Re_+^{n\times m}, y \in \Re_+ : \sum_{j=1}^{m} x_{ij} = 1, \forall i; y - \sum_{i=1}^{n} p_i x_{ij} \geqslant 0,\ \forall j\},$$

The set P satisfies

$$F = P \cap \{(x,y) : x \in B^{n\times m}, y \in \Re\}.$$

Following the classical Polyhedral Combinatorics (see e.g. [10] or [15]), there exists a finite set of inequalities $Ax + Dy \leq \overline{b}$, such that

$$\min\{y : (x,y) \in F\} = \min\left\{y : x \in \Re_+^{n\times m}, y \in \Re_+, Ax + Dy \leq \overline{b}\right\}.$$

The system of inequalities $Ax + Dy \leq \overline{b}$, called the *linear description* of the CO problem, is too large and only a very small part of it could be known. Nevertheless, even a partial linear description provides a rather powerful tool for the solution of the CO problems. An explicit list of the constraints is not required. A method for identifying valid inequalities for the original problem, starting from the solution to the LP relaxation is the core of the Cutting Planes techniques. Iteratively, valid inequalities are added and LP relaxations are solved, until a feasible solution to the CO problem is obtained.

If no more cuts can generated, yet the solution is not feasible, we have to resort to B&B to solve the problem to optimality. The PA hardly ever makes use of this "escape".

3.2 Valid Inequalities

For a given LP solution $(x^0, y^0) \in P$, we have to find valid inequalities that exclude (x^0, y^0) if it is not an integer vector, that is to say, if $(x^0, y^0) \notin F$. To manage it we

have drawn inspiration from the ideas applied to the upper-bound flow model [16].

Let (x^0, y^0) the current LP solution. For each machine j, we call S_j the set of jobs processed, at least in part, by that machine,

$$S_j = \left\{ i \in N : x_{ij}^0 > 0 \right\}$$

Let $\Delta_j = \sum_{i \in S_j} p_i - y^0$ be called the *excess charge* for the machine j, and let $S_j^{'} = \{k \in S_j : p_k > \Delta_j\}$.

If x^0 is not a binary vector, there is some machine j such that

$$\sum_{i \in S_j} p_i > y^0.$$

The following proposition shows that for every machine j with positive excess charge Δ_j, and such that $p_k > \Delta_j$ for some job k in S_j, a valid inequality is generated (such inequality is strong when $0 < x_{kj}^0 < 1$).

Proposition 1:

If $\Delta_j > 0$ and $S_j^{'} \neq \phi$, the following inequality

$$\sum_{i \in S_j} p_i x_{ij} \leqslant y^0 - \sum_{i \in S_j} (p_i - \Delta_j)^+ (1 - x_{ij}) \qquad [1]$$

1. is not satisfied by the current LP solution (x^0, y^0), if there is a $k \in S_j^{'}$ such that $0 < x_{kj}^0 < 1$.
2. is satisfied by every integer solution (x^*, y^*) such that $y^* = y^0$.

(The notation $(...)^+$ indicates that the content enclosed in the parentheses is taken into account only if it is greater than zero)

Proof

1. It is obvious that, for any LP solution, every machine j uses the same time interval in operation. This implies that

$$\sum_{i \in S_j} p_i x_{ij}^0 = y^0.$$

Since there is a job $k \in S_j$ such that $p_k > \Delta_j$ and

$$0 < x_{kj}^0 < 1, \sum_{i \in S_j} (p_i - \Delta_j)^+ \left(1 - x_{ij}^0\right) > 0.$$

Thus, the LP solution (x^0, y^0) does not satisfy the inequality [1].

2. Let (x^*, y^*) be a solution in what x^* is binary and $y^* = y^0$.

Let $B_j = \{i \in S_j / x^*_{ij} = 0\}$,

$$\sum_{i \in S_j} p_i x^*_{ij} = \sum_{i \in S_j \setminus B_j} p_i = \sum_{i \in S_j} p_i - \sum_{i \in B_j} p_i = y^0 + \Delta_j - \sum_{i \in B_j} p_i = y^0 - (\sum_{i \in B_j} p_i - \Delta_j).$$

Since $\sum_{i \in S_j} p_i x^*_{ij}$ can not be larger than $y^* = y^0$, it is necessary that $\sum_{i \in B_j} p_i - \Delta_j \geq 0$.

Thus, $\sum_{i \in S_j} p_i x^*_{ij} = y^0 - (\sum_{i \in B_j} p_i - \Delta_j)^+ \leq y^0 - \sum_{i \in B_j} (p_i - \Delta_j)^+$.

On the other hand, since $1 - x^*_{ij} = 1$ if $i \in B_j$, and $1 - x^*_{ij} = 0$ if $i \in S_j - B_j$, we have

$$y^0 - \sum_{i \in B_j} (p_i - \Delta_j)^+ = y^0 - \sum_{i \in B_j} (p_i - \Delta_j)^+ (1 - x^*_{ij}) - \sum_{i \in S_j - B_j} (p_i - \Delta_j)^+ (1 - x^*_{ij}) = y^0 - \sum_{i \in S_j} (p_i - \Delta_j)^+ (1 - x^*_{ij}).$$

Thus, we can conclude

$$\sum_{i \in S_j} p_i x^*_{ij} \leq y^0 - \sum_{i \in S_j} (p_i - \Delta_j)^+ (1 - x^*_{ij}).$$

⊡

3.3 Transitory Inequalities

For a given not integer LP solution $\overline{x_0} \in P$, new restrictions can be generated by considering the maximum number of jobs to be processed in each machine.

By using the same notation defined in subsection 3.2, the number of jobs from S_j will be termed as s_j. Then, considering the corresponding set of processing times

$$P_j = \{p_{j1}, p_{j2}, \ldots p_{js_j}\}$$

ordered so that

$$p_{j1} \geq p_{j2} \geq \ldots \geq p_{js_j},$$

The following succession is defined

$S_{j1} = p_{js_j}$

$S_{j2} = p_{js_j} + p_{j(s_j-1)}$

.....

$S_{j1} = p_{js_j} + p_{j(s_j-1)} + \ldots + p_{j1} = \sum_{i \in S_j} p_i$

The upper bound on the number of elements belonging to S_j, L_j, is then defined as

$$L_j = h - 1 \Leftrightarrow S_{jh} > lb, \text{ and } S_{jh-1} \leq lb$$

Therefore, the constraint

$$\sum_{i \in S_j} x_{ij} \leqslant L_j$$

can be added. The constraint cut off the non feasible $\overline{x_0}$.

3.4 Preprocessing

The problem is reformulated so as to make as small as possible the difference between the objective function value in the LP relaxation and that corresponding to the integer program. As in cutting planes algorithms, the preprocessing phase is an essential part. It consists of analyzing the given problem instance in order to discover some structure that helps to decompose the instance, to reduce its size, or to tighten the Integer Programming formulation by turning some inequalities into equations, fixing certain variables, etc.

For the proposed algorithm we have implemented: identification of infeasibilities and redundancies, improving bounds and coefficients, and fixing variables. We have only employed specially developed techniques for the PMS problem, exploiting the features of the problem and the actual meaning of the variables.

The jobs are sorted according to

$$p_1 \geq p_2 \geq \ldots \geq p_n.$$

The preprocessing task presents three stages that are described in the following.

3.4.1 Fixing Variables

Due to the fact that the m machines are identical, it is possible to fix, at first, a job to a certain machine. This procedure does not imply a lost of different feasible solutions. In this way a saving of m binary variables is achieved. Furthermore, one constraint is eliminated, and m restrictions are improved by tightening their right hand sides.

3.4.2 Assigning Variables

Taking into account the value of the lower bound of $C_{\max}$, lb, infeasibilities and redundancies could be identified. For a determined value of lb, the possibility that two jobs, with processing times p_k and p_l, could be assigned to the same machine is analyzed. In other words, it is necessary to check if

$$p_k + p_l \leq lb.$$

Beginning with the first two jobs, $p_1 + p_2$ is compared with lb. If the sum is longer than lb, the binary variable x_{21} is fixed to 0, and following the same idea described in the above paragraph, job J_2 is assigned to the machine M_2, and the x_{2j} variables, for $j = 1...m$ are fixed. This procedure is applied to $J_2, J_3, ..., J_m$. The constraints from the first group, formed by the fixed binary variables, disappear. The constraints from the second group are also improved by tightening their bounds.

3.4.3 Adding Constraints

Once the two first stages are finished, the actual lb for each machine could be different. If in the previous procedures the job J_i has been assigned to the machine M_j, then the lower bound lb for this machine has decreased in p_i. Then, analyzing the new values of capacity for each machine, it is possible that a machine may not be able to process a certain job, J_k, because of its processing time. In this case, the restriction

$$x_{kj} = 0$$

is added. This analysis is extended to the rest of jobs and machines.

3.5 Description of the Algorithm

The procedures described in the previous sections have been imbedded into a cutting planes algorithm for the $P//C_{\max}$ problem. We shall now describe its general functionality. The algorithm essentially consists in the iterative computation of lower bounds of $C_{\max}$, starting from the successive lineal programming relaxations. The computation of lower and upper bounds are also crucial for the fast convergence of the algorithm.

The algorithm starts computing the lower bound by means of an improved McNaughton's lower bound. The early McNaughton's algorithm [13] solves the preemptive problem $P/pmtn/C_{max}$ by

$$L = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i; \max_i \{p_i\}\right\}.$$

This lower bound can be improved considering that

$$C_{\max} \geq p_m + p_{m+1}.$$

Then, it is possible to tighten the bound as follows:

$$L_1 = \max\left\{\frac{1}{m}\sum_{i=1}^{n} p_i; \max_i \{p_i\}; p_m + p_{m+1}\right\}$$

Later, this bound can only be increased when the linear program is infeasible.

The upper bound is obtained by the Longest Processing Time first (LPT) heuristic. This procedure consists on making a list according to LPT order and then, as

with any list scheduling algorithm, assigning the uppermost job from the list to a free or to the least loaded machine, until the list is exhausted [5].

If the lower and upper bounds coincide, the algorithm finishes, since the LPT heuristic has found the optimal solution. Otherwise, the iterative process of convergence starts after the preprocessing has been performed.

In each iteration a linear relaxation program is solved in which $C_{\max}$ is required to be equal to the current lower bound. If the obtained solution is integer, hence feasible, the algorithm stops and the current solution is optimal. Otherwise, the separation procedure is executed, so new inequalities, as described above, are added to the linear relaxation.

The developed separation procedure generates two kinds of inequalities, the valid and transitory inequalities, described before (if new inequalities can not be generated, we resort to the B&B algorithm). The new LP is solved and the algorithm stops if the solution is integer. On the other hand, transitory inequalities are removed before to solve further LP relaxations. If the LP relaxation is not feasible, the lower bound is increased in one unit and the process, including the preprocessing phase, restarts.

4 Computational Results

To know how the algorithm described in Section 3 works, we have employed the following algorithms:

1. The proposed algorithm has been coded in Visual Basic 4.0, using the *simplex* method, included in the CPLEX Callable Library v4.0, to solve the linear relaxations.
2. The B&B procedure imbedded into the CPLEX Mixed Integer Solver 4.0. (To avoid the CPLEX routine building a huge tree, because of memory space constraints, we have fixed at 500,000 the parameter determining the maximum number of nodes solved before the B&B terminates. Given the high computational complexity of the problem at hand, the CPLEX routine stops, sometimes, without reaching optimality. In such cases, we have discarded the instances to evaluate the $C_{\max}$ deviation, and computed the time employed solving the 500,000 nodes as if the B&B had solved the problem.)

We executed a series of computational experiments on an IBM compatible PC, by considering three classes of test problems obtained by randomly generating the p_i values according to a uniform distributions in the ranges $[1, 100]$, $[10, 100]$ and $[50, 100]$. For each class, and for different values of n and m, the entries in the Table I give the average number of iterations and CPU time required by each of the three algorithms, computed over 10 problem instances.

The objective function values, computed by the algorithm have been compared with the values corresponding to the optimal solution (when this has been reached by the B&B procedure). In all tested cases, the PA has reached the same objective value than the B&B.

To evaluate the computational performance, we have compared the number of LP relaxations solved by the PA with the number of nodes solved by the B&B, and the CPU times, for the same instances. The computational performance of the PA was, in general, satisfactory for all tested cases. It is important to notice that, while the B&B was not capable of optimizing all tested problems, the PA always was able to converge to an integer solution. Furthermore, the average computational effort of the PA shows an important saving.

5 Concluding Remarks

In this paper we propose a new approximation algorithm based on cutting planes techniques for a hard scheduling problem. We have followed the recent trend of the CO literature, which consists in applying Polyhedral Theory in order to optimize NP-hard combinatorial problems (which gave birth to the called Polyhedral Combinatorics Theory).

The results obtained show that this kind of approach is a quite powerful tool for effectively producing good feasible solutions. They also give support to the hypothesis stating that specially developed algorithms for specified combinatorial problems, work better than general methods like Branch and Bound or Dynamic Programming.

References

[1] Ascheuer, N., Escudero, L., Grötschel, M., and Stoer, M., "A Cutting Plane approach to the sequential Ordering Problem (with aplications to Job Scheduling in manufacturing)", SIAM, Journal of Optimization 1/3, 1993, pp. 25-42.

[2] Christof, T., and Reinelt, G., "Combinatorial Optimization and small polytopes", Top 1/4, 1996, pp. 1-64.

[3] Cook, S.A., "The complexity of theorem-proving procedures", Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151-158.

[4] French, S., "Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop", Ellis Horwood, Chichester, 1982.

[5] Graham, R.L., "Bounds on the performance of scheduling algorithms", SIAM Journal on Applied Mathematics 17, 1969, 263-269.

[6] Graham, R.L., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G., "Optimization and approximation in deterministic sequencing and scheduling. A survey", Annals of Discrete Mathematics 5, 1979, pp. 287-326.

[7] Grötschel, M., Jünger, M., and Reinelt, G., "A Cutting Plane Algorithm for the Linear Ordering Problem", Operations Research 32, 1984, pp.1195-1220.

[8] Hall, L., Schulz, A., Shmoys, D., and Wein, J., "Scheduling to minimize average completion time: off-line and on-line approximation algorithms", Mathematics of Operations Research 3/22, 1997, pp. 513-544.

[9] Hoffman, K.L., and Padberg, M., "Solving airline crew scheduling problems by branch and cut", Management Science 39, 1993, pp. 657-682.

[10] Jünger, M., Reinelt, G., and Thienel, S., "Practical Problem Solving with Cutting Plane Algorithms in Combinatorial Optimization", en: W. Cook, L. Lovász, and P. Seymour (eds.), Combinatorial Optimization, Dimacs 20, 1995, pp. 111-152.

[11] Karp, R.M., "Reducibility among combinatorial problems", in: R.E. Miller and J.W. Tatcher (eds.), Complexity of Computer Computations, Plenum Press, 1972, pp. 85-103.

[12] Lenstra, J.K., and Rinnooy Kan, A.H.G., "Computational complexity of discrete optimization", in: J.K.Lenstra, A.H.G. Rinnooy Kan and P. Van Emde Boas (eds.), Interfaces Between Computer Science and Operations Research, Proceedings of a Symposium held at the Matematisch Centrum, Amsterdam, 1979, pp. 64-85.

[13] McNaughton, R., "Scheduling with deadlines and loss function", Management Science 6, 1959, 1-12.

[14] Padberg, M., and Rinaldi, G., "A Branch and Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problem", SIAM Review 33, 1991, pp. 60-100.

[15] Pulleyblank, W.R., "Polyhedral Combinatorics", in: J.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd (eds.), Handbooks in Operations Research and Management Science 1, Optimization, 1989, pp. 371-446.

[16] Roy, T., Van, and Wolsey, L., "Valid Inequalities for Mixed 0-1 Programs", Discrete Applied Mathematics 4, 1986, 199-213.

[17] Schulz, A., "Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP-Based Heuristics and Lower Bounds", in: W. Cunningham, S. T. McCornick, and M. Queyranne (eds.), Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science 1084, Proceedings of the 5th International IPCO Conference, 1996, pp. 301-315.