

The metaheuristic GRASP as an upper bound for a branch and bound algorithm in a scheduling problem with non-related parallel machines and sequence-dependent setup times

Pedro Leite Rocha¹, Martín Gómez Ravetti¹, Geraldo Robson Mateus¹

¹ Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627 – 31270-901 Belo Horizonte, MG

{djavan, martin, mateus}@dcc.ufmg.br

Abstract. *This paper approaches a scheduling problem with parallel non-related machines, sequence-dependent setup times, due dates, weighted jobs and eligibility constraints. A branch and bound algorithm (B&B) is developed and a solution provided by the metaheuristic GRASP is used as an upper bound. Instances are generated and solved by the B&B. The results are compared to the solutions found by the MILP solver XPRESS. The algorithm is shown experimentally to perform very well for problems with up to 17 jobs. A conclusion is shown, and new tests for the full article are proposed.*

1. Introduction

In the industrial context, scheduling problems are related to Manufacturing Resource Planning (MRP). Programming is made according to the planning horizon. Long term decisions have strategic characteristics and are, therefore, taken by high administration. The short term is known as the tactical level. In this level, the objective is to plan production in order to minimize the manufacturing costs. The demand and due dates are, usually, already defined. The main objective is to define the processing sequence in order to meet both demands and due dates.

A scheduling problem with sequence-dependent setup time is one of the most complex scheduling problems. Considering one machine and one stage, the problem is equivalent to the traveling salesman [Luh et al., 1998]. [Garey and Johnson, 1979] showed that minimizing the makespan considering two identical machines is an NP-hard problem. Certainly, a problem considering non-related machines, due dates, eligibility constraints and sequence dependent setups is also NP-hard.

There is much research work considering parallel machines, but few consider parallel non-related machines and sequence-dependent setup times. [Acero and Delgado, 2000] use a heuristic based on tabu search [Luh et al., 1998] considering parallel non-related machines at each stage, but it doesn't consider sequence-dependent setup times. [Luh et al., 1998] and [Liu and Liao, 2000] use dynamic programming for a problem with sequence-dependent setup times. [Rabadi et al., 2004] use a branch and bound for an early/tardy scheduling problem considering one machine with sequence-dependent setup times. There is more research work using branch and bound [Roundy et al., 1999]. [Hans-Joachim and John, 1996] approach a problem of job shop with no sequence-dependent setup times is using *constraint programming*. [Koulamas and Kyparisis, 2004] consider a makespan minimization problem on uniform parallel machines with release times. For a problem with similar characteristics, [Pinedo, 1995] suggests several heuristics.

An adaptation of the metaheuristic GRASP (Greedy Random Adaptative Search Procedure) [Resende and Feo, 1995] is used to find an upper bound for our problem. We have also referred to [Feo et al., 1991] and [Feo et al., 1996], where applications of GRASP for one-machine scheduling problems are shown, and [Resende et al., 2002], who show an interesting application of the metaheuristic for job shop problems.

In this paper we consider a particular problem. There are two groups of machines: one with identical characteristics and another composed by non-related machines. We also consider two kinds of jobs: any machine can process the type A jobs, but only the group of non-related machines can process the type B jobs. We also consider sequence-dependent setup times. As known, problems involving sequence-dependent setup times are specially difficult. In addition, we consider due dates and weighted jobs.

The objective of this research is to hybridize the metaheuristic GRASP as an upper bound for a branch and bound procedure. Much work has been done about the application of GRASP to our problem [Gómez Ravetti, M. et al., 2003]. In this research we intent to find not only a good solution for our problem, but also to find and prove the optimal solution.

This extended abstract is organized as follows: Section 2 gives the definition of a branch and bound algorithm and the customization for our problem. Section 3 explains how the metaheuristic GRASP works. Section 4 shows the

MILP model used to compare our B&B. In Section 5, the instances for the problem are defined, tests are detailed and results are shown. Section 6 gives a conclusion for this extended abstract and states what should be expected from the full article.

2. The Branch and Bound Algorithm (B&B)

A B&B is a specific enumeration tree strategy. In a B&B, there are three main procedures: *initialization*, *branching* and *bounding*. During the *initialization*, a fast heuristic is used to find a good initial solution, which serves as an upper bound (*UB*).

Branching divides the problem into smaller sub-problems. Each sub-problem represents a partial solution and is represented by a node in the tree. A search strategy must be associated with the branching. It decides which node should be branched next. The *UB* helps to prune nodes from the search tree that have a lower bound (*LB*) greater than *UB* (minimization problem).

The *bounding* procedure calculates the *LB* for each node in order to decide which node should be pruned and which should be branched next. In the following sections, the three procedures are customized for this scheduling problem and described in more detail.

2.1. Initialization

During the initialization, a complete initial solution is found to serve as an *UB*. Any node in the enumeration tree with a *LB* greater than *UB* can be pruned. In this paper, we use the a simple greedy heuristic to find our first solution. This heuristic consists of two phases: ordering the jobs and assigning them to the machines.

Initially, the jobs are ordered in a non-decreasing order using the due date as sorting rule. Following this order, each job is assigned to the machine capable of finishing the job first. This greedy function was chosen arbitrarily.

2.2. Branching

The branching procedure develops the enumeration tree. Our branching scheme is divided into 2 phases. In the first phase, the jobs are assigned to the machines, but no processing sequence is defined. This sequence is decided only in the second phase.

The first branching scheme for our B&B starts at the root of the tree (level 0), where no jobs are assigned. At level 1, m nodes are created to m machines. For each node in this level a certain job is assigned to a specific machine. For example, with a number of machines $m = 2$ and a number of jobs $n = 6$, at level 0 there is one node with no jobs assigned. At level 1, 2 nodes are created, and a certain job is assigned to each machine. This goes on until level 6, where all 6 jobs are assigned. The first branching scheme is illustrated at Fig. 1(a).

Now, the job sequence must be decided. Suppose a particular node at level 6 where 3 jobs are assigned to each machine. The second branching scheme begins at the first machine. For this particular node at level 6, the branching scheme creates 3 nodes at level 7, where a certain job from the first machine is assigned to the first position of the sequence

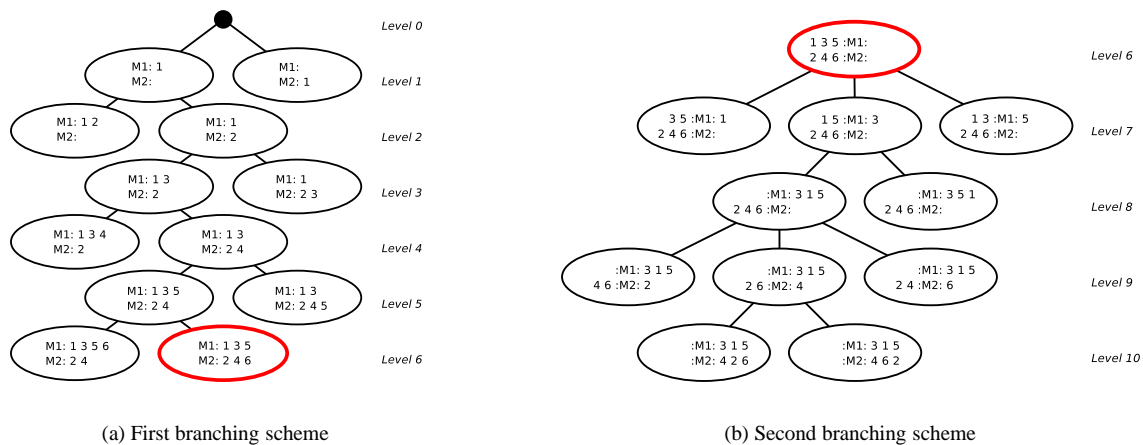


Figure 1: Illustration of the branching schemes

in this machine. For each node at level 7, 2 nodes are created at level 8, where each remaining job is assigned to the second position. The earliest due date is the most common way of assigning jobs and also used in this B&B. However, this is not necessarily the best job-assigning method when you have sequence-dependent setup times. The second branching scheme is illustrated at Fig. 1(b).

To avoid a full enumeration of the tree, a bounding procedure is applied to find a LB for each node. For a certain node i at level k , if $LB_{ik} \geq UB$ this node is pruned. Obtaining a LB at each node is not only useful to prune nodes, but also to guide the search. In this B&B, at each level in the first phase, the node with the smallest LB is followed. The derivation of the LB is discussed in the following section.

2.3. Bounding

In this section, a lower bound algorithm for a given partial solution is presented. Since this B&B is elaborated with two branching schemes, two different bounding algorithms are developed.

In the first branching scheme, the sequence is not defined yet. The processing times for the assigned jobs and the smallest processing times for the non-assigned jobs are added considering the smallest setup time. The result is divided by the number of machines to give the LB for the makespan. The processing times for the jobs assigned to a certain machine considering the smallest setup time is also a LB for the makespan. Since the due date for a job may be smaller than its processing time, the LB for the weighted tardiness is also calculated. Finally, the LB for the makespan is added to the LB for the sum of the tardiness penalties for all jobs, giving us the LB for the partial solution.

For the second branching scheme, the LB is calculated in a different way. After choosing the next job, the LB for the makespan and for the tardiness are updated. The LB for the tardiness is updated considering the earliest possible start time for each unscheduled job.

3. GRASP (Greedy Random Adaptative Search Procedure)

Besides the greedy function described in Section 2.1, we use the metaheuristic GRASP with different configurations to find our first solution to evaluate it as an UB for our problem. This metaheuristic consists of two phases: the construction of a feasible solution and a local search. These two phases are repeated for each iteration.

Initially, the jobs are ordered in a non-decreasing order using the due date as sorting rule. During the construction phase, the algorithm randomly reorders the array by means of a specific probability distribution. In this paper, the probability function is $f(x) = 1/x$, where $f(x)$ represents the probability for the x^{th} job to be chosen next. Finally, a greedy function is used to schedule the jobs. The local search, arbitrarily chosen, works switching all the existing pairs of jobs assigned to different machines in the first phase.

We also use path relinking (PR) at the end of each iteration to intensify the local search. Even so, there are several ways to implement this technique. In this paper, PR works as follows: first, a pool of good solutions is retained. Every time PR is used, a solution is randomly chosen from this pool. Then, all solutions in the path from the solution found in the local search to the selected solution from the pool are analyzed. If a better solution is found, it is added to the pool.

4. MILP (Mixed Integer and Linear Programming) Model

To verify that the B&B produces optimal solutions, a set of problems are solved using MILP. We use a previously proposed MILP model [Gómez Ravetti, M, 2003]:

- M : set of machines
- L : set of identical machines
- N : set of jobs
- d_i : due date for job i
- w_i : weight for job i
- p_{im} : processing time for job i at machine m
- $s_{ii'm}$: setup time between jobs i and i' at machine m
- G : a big integer

Decision variables:

- α_{im} : 1 if job i is processed at machine m , 0 otherwise
- t_i : processing start time for job i

- $\beta_{ii'm}$: 1 if job i is processed at machine m before job i'
- ρ_i : tardiness for job i
- Z : makespan

The problem can be formulated as:

$$\text{Min}(Z + \sum_{i \in N} \rho_i \times w_i)$$

s.t.:

$$\sum_{m \in M} \alpha_{im} = 1, \quad \forall i \in N \quad (1)$$

Each job i will be processed at one machine.

$$d_i + \rho_i \geq t_i + p_{im} - (1 - \alpha_{im}) \times G, \quad \forall i \in N, \quad \forall m \in M \quad (2)$$

The start time for job i added to its processing time at machine m must be equal to or less than the due date added to a possible tardiness ρ_i .

$$Z \geq t_i + p_{im} - (1 - \alpha_{im}) \times G, \quad \forall i \in N, \quad \forall m \in M \quad (3)$$

The makespan is either greater than or equal to the start time of every job i added to the processing time for job i at the machine where it is processed.

$$(1 - \alpha_{im}) \times G + (1 - \alpha_{i'm}) \times G + (1 - \beta_{ii'm}) \times G + t_{i'} \geq t_i + p_{im} + s_{ii'm} \quad (4)$$

$$(1 - \alpha_{im}) \times G + (1 - \alpha_{i'm}) \times G + \beta_{ii'm} \times G + t_i \geq t_{i'} + p_{i'm} + s_{i'im} \quad (5)$$

$$\forall i \neq i', \{i, i'\} \in N, \forall m \in M$$

The first equation states that job i' will be processed at machine m after the start of job i added to the processing time for job i at machine m and to the setup from i to i' at machine m , if job i' is processed at machine m after job i . The second equation states the reciprocal. A job i will be processed at machine m after the start of job i' added to the processing time for job i' at machine m and to the setup from i' to i at machine m .

$$\sum_{i \in N} \alpha_{im} \leq \sum_{i \in N} \alpha_{im'}, \quad \forall m < m', \quad \{m, m'\} \in L \quad (6)$$

The last constraint determines that a machine cannot process more jobs than a machine of the same type with greater index, i.e. a machine m cannot process more jobs than m' if $m < m'$ and m and m' are of the same type. This constraint eliminates simetry among the solutions.

5. Tests and Results

Instances are generated following uniform discrete distributions. The processing and setup times are generated in a triangular form, where the processing times p_{im} range from 5 to 15 and the setup times $s_{ii'm}$ range from 1 to 7. The due dates are related to the number of jobs N and the job priority is a random number, where the due dates d_i range from 1 to $N \times 3.5$ and the priority w_i ranges from 1 to 3. The machine configuration has 6 machines, where 4 are identical and 2 are non-related. It is also set that a job has a probability of 50% for being type A, and 50% for being type B. Data for problems with number of jobs ranging from 7 to 17 is generated. For each problem size, 10 instances are generated (a total of 110 instances).

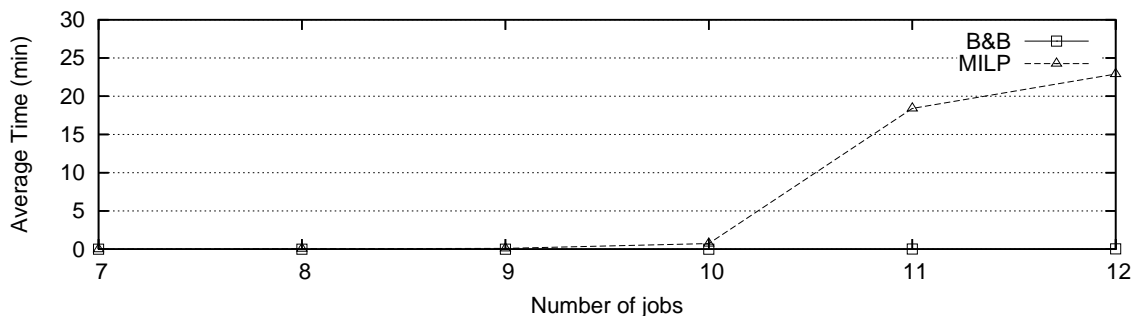


Figure 2: Comparison between the performance of the XPRESS MILP solver and the B&B

5.1. Comparison between the B&B and the MILP

The MILP presented in Section 4 is modeled as an LP, and solved using XPRESS 14.27. XPRESS, as well as other MILP solvers, uses built-in algorithms that find *LBs* by linear programming relaxation. For more information on XPRESS, please see [Dash Optimization, 2004].

The time needed to solve some problems of 12 jobs using XPRESS is more than 1.5 hour, which makes the MILP impractical for solving larger problems. The set of problems with size from 7 to 12 jobs is solved by XPRESS. The solver is unable to solve two of these instances due time constraints. These two instances are not considered here. The same set of problems is solved using the B&B developed in this research, which obtains the same optimal solutions as the MILP solver with CPU of seven seconds or less per instance. Fig 2 compares the performance of the solver to the B&B developed in this research. It is clear that the B&B outperforms the MILP solver. The significant difference is due to the fact that the B&B developed here is tailored to this problem, while the MILP solver uses generic algorithms.

5.2. Upper bound (GRASP) effectiveness test results

The improvement on the number of nodes branched is an interesting method to measure the efficiency of GRASP as an *UB* for our problem. We calculate the improvement by subtracting the number of nodes branched with GRASP from the number of nodes branched without GRASP and dividing the result by the number of nodes branched without GRASP. This way, we can obtain the percentage of nodes that didn't need to be branched thanks to the *UB*. These results are shown in Table 1.

All configurations of GRASP had very similar results with the smaller numbers of jobs. As the number of jobs grows, the differences among them become clear. With 9 jobs there is some difference between GRASP at 100 and 1000 iterations, but none between 1000 and 20000 iterations. With 13 jobs is the first time a difference between 1000 and 20000 interactions appears. This difference is greater with 16 and 17 jobs. From these results, we can see how important an efficient upper bound is to our problem. When the improvement at 100 iterations is the same as at 1000, it means GRASP finds the same solution at 100 iterations as it does at 1000. In the worst cases there is no improvement at all. In these cases the first solution found by the B&B is either the same found by GRASP or better. Therefore, there is not much to be improved. The average improvement is between 20% and 25%. In the best cases the improvement is higher than 95%.

6. Conclusions

The main contribution of this paper is that a B&B was developed to solve instances of a scheduling problem with parallel non-related machines, sequence-dependent setup times, due dates and eligibility constraints up to 17 jobs within a reasonable time. The metaheuristic GRASP was tested as an upper bound and showed to be effective. The quality of the upper

its.		7	8	9	10	11	12	13	14	15	16	17	Average
100	Min	4.43	0.02	0.00	0.05	0.00	0.00	0.97	0.06	0.00	0.00	0.00	0.50
	Avg	40.36	27.83	15.90	22.80	16.75	8.08	23.90	29.72	25.76	12.87	13.69	21.60
	Max	85.51	81.87	59.29	43.83	45.92	29.36	56.65	96.62	63.37	35.14	41.37	58.08
1000	Min	4.43	0.02	0.00	0.05	0.00	0.00	1.47	0.06	0.00	0.00	0.05	0.55
	Avg	40.36	28.34	17.07	22.80	16.78	9.82	24.10	36.85	26.68	15.51	21.73	23.64
	Max	85.51	81.87	70.93	43.83	45.92	29.36	56.65	96.62	63.37	59.44	66.14	63.60
20000	Min	4.43	0.02	0.00	0.05	0.00	0.00	1.47	0.06	0.00	0.00	4.85	0.99
	Avg	40.36	28.34	17.07	22.80	16.78	9.82	24.12	37.09	26.68	16.62	28.89	24.41
	Max	85.51	81.87	70.93	43.83	45.92	29.36	56.65	96.62	63.37	70.52	66.14	64.61

Table 1: Number of nodes branched by the B&B

bound greatly affects the performance of the B&B, and a better upper bound means better performance, although it may cost more to calculate the upper bound at the beginning.

GRASP proved to be an efficient upper bound, but it doesn't guarantee the optimal or a maximum *gap* to the optimal. Using a branch and bound with GRASP we found a way to guarantee the optimal, even if only for small problems.

The full version of this article includes a deeper analysis of the effectiveness of GRASP as the upper bound, as well as results with more general set of tests. There is also a full factorial experimental design to test the overall performance of the B&B.

References

- Acero, R. D. and Delgado, J. T. (2000). Aplicación De Una Heurística De Búsqueda Tabú En Un Problema De Programación De Tareas En Línea Flexible De Manufactura. *COLOMBIA*.
- Dash Optimization (2004). Xpress-MP - the fast track from formulation to solution. <http://www.dashoptimization.com/products.html?locale=english>.
- Feo, T., Sarathy, K., and McGahan, J. (1996). A grasp for single machine scheduling with sequence dependent setup costs and linear delay penalties. *Computers Operations Research*, (23):881–895.
- Feo, T., Venkatraman, K., and Bard, J. (1991). A grasp for a difficult single machine scheduling problem. *Computers & Operations Research*, (18):635–643.
- Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W H Freeman & Co., Brasil.
- Gómez Ravetti, M (Dez, 2003). Problemas de seqüenciamento com máquinas paralelas e tempos de preparação dependentes da seqüência. Master's thesis, DCC, Universidade Federal de Minas Gerais.
- Gómez Ravetti, M., Mateus, G. R., and Rocha, P. L. (2003). A scheduling problem with unrelated parallel machines and sequence dependent setups. *XII CLAIO (Congreso Latino-Iberoamericano de Investigación de Operaciones y Sistemas)*.
- Hans-Joachim, G. and John, U. (April, 1996). Methods for Solving Practical Problems of Job-Shop Scheduling Modelled in CLP(FD). In *Conf. on Practical Application of Constraint Technology*.
- Koulamas, C. and Kyparisis, G. J. (2004). Makespan minimization on uniform parallel machines with release times. *European Journal of Operational Research*, pages 262–266.
- Liu, C.-Y. and Liao, D.-Y. (2000). Scheduling Flexible Flow Shops with Sequence-Dependent Setup Effects. In *IEEE Transactions on Robotics and Automation*, volume 16.
- Luh, P. B., Gou, L., Nagahora, T., Tsuji, M., Yoneda, K., Hasegawa, T., Kyoya, Y., and Kano, T. (1998). Job Shop Scheduling With Group-Dependent Setups, Finite Buffers, And Long Time Horizon. *Annals of Operations Research*, 76.
- Pinedo, M. (1995). *Scheduling Theory, Algorithm and System*. Technical report, Englewood Cliffs, NJ. Prentice Hall.
- Rabadi, G., Mollaghasemi, M., and Anagnostopoulos, G. C. (2004). A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers & Operation Research*, 31:1727–1751.
- Resende, M. and Feo, T. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, (6):109–133.
- Resende, M. G. C., Binato, S., Hery, W. J., and Loewenstern, D. M. (2002). A Greedy Randomize Search Procedure For Job Shop Scheduling. In *Essays and Surveys in Metaheuristics*, pages 59–80. C.C. Ribeiro and P. Hansen, editors. Kluwer Academic.
- Roundy, R., Cakanyldirim, M., Chen, D., Freimer, M., Jackson, P. L., and Melkonian, V. (February, 1999). Capacity-Driven Acceptance of Customer Orders for a Multi-Stage Batch Manufacturing System: Models and Algorithms. Technical report, School of Operations Research and Industrial Engineering, Cornell University, Technical Report No. 1233.