Experimental comparison of branch-and-bound algorithms for the $1 \mid r_j \mid L_{\max}$ problem

Ruslan Sadykov (Speaker) [*] Alexander Lazarev [†]

1 Introduction

We consider the problem $1 \mid r_j \mid L_{\max}$. A set $N = \{1, \ldots, n\}$ of jobs has to be scheduled on a single machine. Each job $j \in N$ has a processing time p_j, a release date r_j and a due date d_j. Lateness L_j of a job $j \in N$ in a schedule π is the difference between its completion time and due date: $L_j = c_j(\pi) - d_j$. The goal is to find a schedule that minimizes the maximum lateness $L_{\max}(\pi) = \max_{j \in N} L_j$.

This problem is one of classic scheduling problems. It is NP-hard in the strong sense [3]. The main interest is to develop exact algorithms for this problem. Such algorithms help to solve more complex and practical scheduling problems, for example, job-shop problem.

Several branch-and-bound algorithms have been developed for the problem. Two of them have been shown to be quite efficient when used in practice. These algorithms have been suggested by McMahon and Florian [4] and by Carlier [2]. Unfortunately, to the best of our knowledge, these two algorithm have never been compared computationally. The first result of this work is realization of this comparison. Second result is an improvement of the Carlier algorithm. This improvement is based on embedding "Edge-Finding" filtering algorithm [1, p.24] developed earlier for using in Constraint Programming. This modification of the Carlier algorithm allowed to increase its efficiency and robustness.

2 Branch-and-bound algorithms

We tested computationally the efficiency of four branch-and-bound algorithms for the $1 \mid r_j \mid L_{\max}$ problem. First one is the algorithm by McMahon and Florian (MMF). Second one is the algorithm by Carlier (CAR). Both algorithm were implemented using the recursive backtrack search strategy, whereas originally the "best bound" search strategy have been used. Moreover, in the algorithm CAR, lower bounds are obtained in the same way as in the algorithm MMF. This modification of lower bounding strategy allowed to increase efficiency of the algorithm CAR.

[*] `sadykov@core.ucl.ac.be`. Center of Operations Research and Econometrics, Université Catholique de Louvain, voie du Roman Pays 34, 1348 Louvain-la-Neuve, Belgium.

[†] `Alexandr.Lazarev@ksu.ru`. Chair of Economic Cybernetics, Kazan State University, Kremlevskaya Street 18, 420008 Kazan, Russia

Then, we tested Constraint Programming algorithm (CP). This algorithm solves a sequence of feasibility problems. Each feasibility problem tests whether there exists a schedule with maximum lateness not more than some value L'. For each job $j \in N$ we set a deadline $\bar{d}_j := d_j + L'$. Then the Constraint Satisfaction Problem with one disjunctive resource constraint is solved [1]. This algorithm is implemented with the Carlier branching strategy.

The fourth algorithm (HYB) is a modification of the algorithm CAR. At each node of the search tree the algorithm HYB uses the “Edge-Finding” filtering algorithm to tighten the time windows of jobs and to detect infeasibility if it is possible. To do this for each job $j \in N$ we set a deadline $\bar{d}_j = d_j + UB - 1$, where UB is the current upper bound on the optimal solution.

3 Numerical experiments

The experiments have been carried out on a computer with a 2 GHz processor and 512 Mb of memory. For the experiments we used two test instances which were generated using the following scheme. Processing times, release and due dates were uniformly generated in the intervals $[10, 100]$, $[1, K_r \cdot n]$ and $[1, K_d \cdot n]$, correspondingly. Parameters K_r, K_d take values from the set $\{20, 30, 40, 50, 60\}$. For each couple of parameters there are 100 instances in the test set. In total there are 2500 instances for each $n \in \{50, 100, 200, 300\}$.

In the experiments we were interested in the following statistics. P_{1m} is the percentage of instances solved to optimality in 1 minute. T_{av} is the average time in milliseconds, needed to solve an instance to optimality. N_{av} is the average number of nodes in the search tree, needed to solve an instance to optimality.

Notice that if the time limit is reached when solving an instance, the statistics take into account the current time and the current number of solved nodes. Therefore, if there exist unsolved instances for some algorithm, the given values of the statistics T_{av}, N_{av} are lower bounds of their real values.

Table 1: Experimental comparison of four algorithms

	MMF			CAR		
n	P_{1m}	T_{av}	N_{av}	P_{1m}	T_{av}	N_{av}
50	94.24%	3523.2	151842.5	99.84%	109.4	7192.1
100	96.32%	2213.3	31048.3	99.60%	247.9	5431.2
200	97.88%	1278.7	5438.1	99.76%	148.3	956.6
300	97.84%	1314.5	2853.4	99.64%	228.6	596.1
	CP			HYB		
n	P_{1m}	T_{av}	N_{av}	P_{1m}	T_{av}	N_{av}
50	100%	11.7	18.6	100%	3.4	5.6
100	100%	64.6	32.0	100%	18.2	10.0
200	100%	407.0	56.2	100%	119.0	18.7
300	100%	1179.6	75.6	100%	373.2	27.4

Results of the experiments are presented in Table 1. The algorithm MMF was the worst in all terms. The algorithm CAR was much more efficient and does not able to solve only less than 0.5% of instances in one minute. Therefore, the efficiency of the algorithm CAR is satisfactory to use it in the practice. Although, if it is needed to solve quickly all the given set of instances to optimality without exception, the algorithms CP or HYB are preferable. They are able to solve all the test instances. The probability of encountering instances which can not be solved by these algorithms in a reasonable time seems to be negligible.

Among the last two algorithms, CP is clearly worse. The reason is that it needs to solve several similar Constraint Satisfaction Problem instances (in the experiment it solved 3 such instances on average).

We tried to increase the time limit up to 30 minutes and apply the algorithm CAR to instances which were not solved in 1 minute. Only a small part (20-30%) of such instance were solved in more time.

The main conclusion of this work is that adding filtering (propagation) algorithms developed for Constraint Programming may really improve the efficiency and robustness of combinatorial algorithms for scheduling problems. However, one should use or develop a branching scheme that allows embedding of filtering algorithms.

References

[1] Ph. Baptiste, C. Le Pape, W. Nuijten (2001)*Constraint-based scheduling: applying constraint programming to scheduling problems.* Kluwer Academic Publishers.

[2] J. Carlier (1972)The one-machine sequencing problem.European Journal of Operations Research, **11**(1):42-47.

[3] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker (1975)Complexity of machine scheduling problems.*Annals of Operations Research* **1**:343-362.

[4] G. McMahon, M. Florian (1975)On scheduling with ready times and due dates to minimize maximum lateness.*Operations Research*, **23**:475-482.