Combining AI/OR techniques for solving Open Shop problems

Christelle GUÉRET and Narendra JUSSIEN
École des Mines de Nantes
La Chantrerie – 4 Rue Alfred Kastler
BP20722
F–44307 Nantes Cedex 03, FRANCE
email: {Christelle.Gueret, Narendra.Jussien}@emn.fr

1 Introduction

In the Open-Shop problem, a set J of n jobs, consisting each of m operations (or tasks) must be processed on a set M of m machines. The operations of a job can be processed in any order, but only one at a time. We consider the construction of non-preemptive schedules of minimal makespan, which is NP-Hard for $m \geq 3$ [Gonzalez and Sahni, 1976]. In the following, the processing time of a task i is denoted by p_i, its head (least feasible starting time) by r_i and its tail (symmetric notion from the end of the scheduling) by q_i.

Few branch and bound methods for that problem have been published so far. One of the best [Brucker *et al.*, 1997] consists, in each node, in fixing disjunctions (adding new tentative precedence constraints) on the critical path of a heuristic solution. This branch and bound method uses *immediate selections* [Carlier and Pinson, 1989] in each node in order to fix additional disjunctions by propagation. Although that method is the best one as far as we know, some problems from size 7×7 remain unsolved.

In this paper, we present the integration of two techniques in that branch and bound method in order to improve the search. The first one is an AI intelligent backtracking technique. The second one is a new propagation rule developed thanks to OR algorithms. We tested those two techniques on Open-Shop problems from the literature. The search is definitely improved: on some square problems of size 10, the number of backtracks is reduced by more than 90%.

2 Using an AI technique: Intelligent Backtracking

The Branch-and-Bound method of Brucker *et al.* is a depth-first search algorithm based on the use of a chronological backtracking search mechanism to explore the search tree. Such an exploration is known to lead to useless branch tests and explorations. Works on improving chronological backtracking have led to the mechanism of intelligent backtracking in the AI community [Bruynooghe and Pereira, 1984].

We provide here an intelligent backtracking technique based on an explanation recording mechanism that is able to provide interesting information for determining relevant backtrack points.

The main idea [Jussien and Lhomme, 1998] is to record an *explanation* (a set of previously made decisions) for each domain reduction during the search (that is heads and tails adjustments due to the selection of disjunctions). In order to reduce the overall space complexity of the approach, we actually keep only explanations for the bounds of our variables.

Our variables are the starting time of the tasks in the problem. Therefore, recording explanations for the bounds of the variables provides, for any task i, explanation for its head r_i (the minimum feasible starting time – the lower bound of the variable value) as for its tail q_i (it is strongly related[1] with its maximum feasible starting time – the upper bound of the variable value).

Those explanations are used when contradiction occurs:

- when there exists a task i for which it is known that it will end after the current upper bound computed for the schedule[2]. In that case, the contradiction (leading to a backtrack to the last choice point in a classical approach) is due to $Expl(r_i) \cup Expl(q_i)$ where $Expl(x)$ is the explanation of the data x. The only relevant backtrack point that ensures that no solution will be missed is then the most recent decision point appearing in the computed explanation.

- all children of a given node r have been explored without success. Thus, node r is not valid and backtracking is needed. Since all the children have been rejected due to a computable reason (see above), a relevant backtracking point can be computed using those explanations in the same way as in the previous case.

This technique has non exponential space complexity. Indeed, each explanation are, by construction, limited to polynomial size and explanations are kept only for upper and lower bounds for the variables and for each node in the current path in the search tree (at most $n \times m \times (n + m)$, the worst case depth of the search tree).

As regards time complexity, recording explanations for the modification of a head or a tail requires to make the union of several explanations during constraint propagation. Those unions involve a polynomial number of sets of polynomial size and therefore take polynomial time.

3 Using OR algorithms: Forbidden Intervals

Forbidden Intervals are intervals in which, in an optimal solution, no task starts nor ends [Guéret and Prins, 1998]. The technique we propose is to compute

[1]Because q_i is the symmetric notion of r_i but from the end of the schedule.

[2]$r_i + p_i + q_i \geq UB$

in advance Forbidden Intervals and then to use them during the search as a propagation technique to reduce the domains of the heads and tails of the tasks. Knowing an upper bound UB, Forbidden Intervals are obtained by solving subset-sum problems.

We can associate to each machine M_i a subset-sum problem composed of the processing times of the tasks executed on M_i and with capacity the load of M_i. The resolution of that subset-sum problem by a dynamic programming algorithm ([Martello and Toth, 1990] for instance) provides, sorted in increasing order, all the sums of processing times achievable by a subset of tasks of machine M_i. Each pair a, b of consecutive values defines a possible interval for the heads and tails of tasks of M_i. If we consider that a task k executed on M_i has its head r_k in such an interval $[a, b-1]$, then the maximum amount of work that can be executed before r_k is a (greatest sum of processing times achieved by a set of tasks, but lower than r_k). For each possible value c of r_k in $[a, b-1]$, we can then compute the following lower bound for the end of the schedule: $c + p_k + L_i - a$ where L_i is the load of machine i. If this lower bound is greater than an upper bound UB then c, and also all the values of $[c, b-1]$, are forbidden for r_k. Similar deductions can be made by reasoning on heads.

Forbidden Intervals can also be computed for each job.

The total complexity to compute Forbidden Intervals for each machine and for each job is $O(m.n.LB)$, where LB is the classical lower bound equal to the maximum of machine loads and job durations.

Those Forbidden Intervals can be incorporated in a Branch and Bound method to adjust heads and tails, thus improving the search. During the search, each time a better solution is found, those intervals are updated.

4 Computational results

We introduced our new approaches in the Branch and Bound method by Brucker *et al.* and tested them on benchmarks proposed in [Taillard, 1993]. Those benchmarks are composed of 40 square instances of size 4, 5, 7 and 10 (10 for each size). Among the 10 problems of size 10, 4 are still unsolved.

The integration of only one of those two techniques in the branch and bound method of Brucker *et al.* provides interesting results. But we observe more significant improvements by combining both of them: indeed, those techniques reduce by about 3% the number of backtracks for problems of size 4, close to 22% for 5x5 problems and more than 11% for 7x7 problems. Whereas no problems of size 10 is solved in less than 250000 backtracks with the initial version, our techniques find the optimal solution for 3 of them (among which an open problem solved in 4843 backtracks only), and the value obtained after 250000 backtracks are better for four other problems of that size.

As regards execution times, for large problems ($\geq 7 \times 7$ problems) our techniques are a little slower in each node than the version without them, but this additional time is widely compensated by the decrease of the number of explored nodes.

In [Alcaide *et al.*, 1997] and [Liaw, 1999] several tabu search procedures are tested on Taillard's problems. Alcaide *et al.* find the optimal solution for 2 problems out of 10 of size 7 and 1 problem out of 10 of size 10. In [Liaw, 1999], the tabu search of the author finds the optimal solution for 5 problems out of 10 of size 7 and 6 problems out of 10 of size 10.

Those results are similar to ours but tabu search is a meta heuristic and cannot guarantee optimality (except if the lower bound is reached which is the case in the cited examples) whereas our method is systematic and complete.

5 Conclusion and further work

In this article, we presented both a new propagation rule based on OR techniques and an AI intelligent backtracking technique to improve a branch and bound method for the Open-Shop problem. Each of those techniques is efficient, but by combining them we obtain better results.

Forbidden Intervals and our intelligent backtracking technique can easily be adapted to Flow-Shop and Job-Shop problems.

Currently, we are working on a new way of improving the search: Dynamic Backtracking. In this technique intelligent backtracking is completely replaced with a *dynamic backtracking* [Ginsberg, 1993] which, instead of backtracking, consists in *jumping* from solutions to other solutions in the search tree.

References

[Alcaide *et al.*, 1997] Alcaide, Sicilia, and Vigo. A tabu search algorithm for the open shop problem. *TOP (Trabajos de investigación operativa)*, (5):283–296, 1997.

[Brucker *et al.*, 1997] P. Brucker, J. Hurink, B. Jurish, and B. Wöstmann. A branch and bound algorithm for the open-shop problem. *Discrete Applied Mathematics*, 76:43–59, 1997.

[Bruynooghe and Pereira, 1984] M. Bruynooghe and L. M. Pereira. *Implementations of Prolog*, chapter Deduction revision by intelligent backtracking, pages 194–215. Ellis Horwood, 1984.

[Carlier and Pinson, 1989] J. Carlier and É. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35:164–176, 1989.

[Ginsberg, 1993] Matthew L. Ginsberg. Dynamic backtracking. *Journal of Artificial Intelligence Research*, 1:25–46, 1993.

[Gonzalez and Sahni, 1976] T. Gonzalez and S. Sahni. Open-shop scheduling to minimize finish time. *Journal of the Association for Computing Machinery*, 23(4):665–679, 1976.

[Guéret and Prins, 1998] C. Guéret and C. Prins. Forbidden intervals for open-shop problems. Technical report, École des Mines de Nantes, 1998. 98/10/AUTO.

[Jussien and Lhomme, 1998] Narendra Jussien and Olivier Lhomme. Dynamic domain splitting for numeric csp. In *European Conference on Artificial Intelligence*, pages 224–228, Brighton, United Kingdom, August 1998.

[Liaw, 1999] Liaw. A tabu search algorithm for the open shop scheduling problem. *Computers and Operations Research*, (26):109–126, 1999.

[Martello and Toth, 1990] S. Martello and P. Toth. *Knapsack problems*. Wiley, 1990.

[Taillard, 1993] É. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operations Research*, 64:278–285, 1993.