Overview and Possible Extensions of Shaving Techniques for Job-Shop Problems

Philippe Torres and Pierre Lopez

LAAS-CNRS, 7 avenue du colonel Roche,
F-31077 Toulouse Cedex 4, France
{ptorres,lopez}@laas.fr

Abstract. This paper proposes some new consistency enforcing techniques for job-shop scheduling problems with non relaxable time-windows. It proposes to extend propagation mechanisms known under the name of "shaving", based upon the refutation of a decision. We will present two possible extensions and discuss some experimental results obtained on hard job-shop problems. A possible use of these cutting techniques to help neighbourhood search procedures to escape from local optima is presented as a future work.

1 Introduction

This paper proposes some new consistency enforcing techniques for disjunctive scheduling problems with time-window and precedence constraints. A famous problem is the job-shop problem with non relaxable time-windows, which is NP-complete. To lower the average complexity of solving such scheduling constraint satisfaction problems (CSPs), specific consistency enforcing techniques based on pure constraint propagation such as edge-finding have been developed and successfully used in tree search procedures to prune the search space [1, 3, 5, 8, 18]. For general CSPs, a particular family of consistency enforcing techniques based upon the refutation of a decision by constraint propagation alone (SAC [10]) or tree search (MAC [17]) are worth their cost for hard problems; the scheduling problem at hand is one of them. Some refutation techniques based on the specific nature of the constraints of disjunctive scheduling have been developed [6, 14, 16]. They are powerful consistency enforcing techniques, mainly known under the name of "shaving".

Section 2 presents a rapid overview of the existing shaving techniques. In section 3 two extensions are derived and we discuss some experimental results obtained on job-shop instances from the literature. In section 4 some improvements of these new cutting techniques and their possible use to help neighbourhood search procedures to escape from local optima are finally presented as a future work.

2 Overview of Shaving Techniques

The general principle of consistency enforcing techniques based upon refutation in CSPs goes like this: a variable is assigned a value within its domain and constraint propagation or tree search is performed. If an inconsistency arises, the assigned value is suppressed from the domain of the variable hence providing a better characterization of the CSP. Some refutation techniques based on the specific nature of the constraints of disjunctive scheduling have been developed and known under the name of "shaving". In this case a local constraint is posted and the corresponding adjustments are propagated on the overall problem. An inconsistency imposes the negation of the posted constraint, which is added to the problem definition.

Shaving techniques have been first introduced by Carlier and Pinson in [6] under the name of "global operations" and further appeared as "shaving" in [14][1]. As we consider two major types of constraints, time constraints and resource constraints, two types of shaving techniques have been studied so far: the first technique tries to reduce the time-windows of the tasks and the second aims to solve disjunctions.

2.1 Reducing Time-Windows

The general principle goes like this: a task is constrained to start within a reduced part of its time-window and constraint propagation is performed. If an infeasibility arises, the task cannot be processed during this reduced time-window and an adjustment of its original time-window is achieved. A bisection search on the time-window of each task is performed to ensure the greatest possible reduction.

This time-window shaving has been efficiently implemented in [14] and also strengthened by way of a "double" shaving which iterates the Carlier-Pinson shaving over one level of recursion. Some problems seem only solvable by this powerful but very time consuming recursive shaving (the ABZ7 instance, a 20×15 job-shop problem open then, was solved after almost 2 days of computation).

Further developments are led in [16]. Conditions are proposed to reduce the bisection search on restricted intervals. When shaving the time-window of a task i, it is also proposed to keep the information propagated on the other tasks j. This information is used to derive time-windows reductions on tasks j which would remain unfound otherwise. Finally, a shaving technique based on a complex partitionning of time-windows is introduced and allowed the author to obtain remarkable results such as proving optimality and finding an optimal solution without backtracking (but at the expense of the CPU time) for the FT10 job-shop problem.

2.2 Solving Disjunctions

This technique has been introduced in [6] and consists in characterizing immediate selections. For each unscheduled pair of tasks (i, j), a sequencing is posted,

[1] We use indifferently both names in this paper

e.g. $i \ll j$ (resp. $i \gg j$), then constraint propagation rules are applied with the hope to raise an infeasibility, thus proving the necessary sequencing $i \gg j$ (resp. $i \ll j$). Péridy proved in [16] that this technique is dominated by the time-windows shaving.

3 Extending Shaving Techniques for Solving Disjunctions

The basic idea is to generalize and evaluate the shaving upon unscheduled pairs of tasks to disjunctive k-cliques (that is, clique of disjunctions of size k) with $k > 2$.

3.1 Shaving on Disjunctive Triplets

The first natural step towards this generalization is to consider triplets of conflicting tasks. The idea of deriving sequencing deductions from the local analysis of disjunctive triplets of tasks first appears in [12]. Six local conditions are tested on the triplet and if at least one is verified then one or more disjunctions on the triplet are solved. This technique is part of the local operations family [4] where the possible configurations of a subset of operations are derived feasible or not from the sole use of the durations and time-windows of the subset of tasks.

We decided to test the efficiency of this technique when used as a shaving (or global operations) technique. Given a disjunctive resource where n tasks must be performed, there are $\binom{n}{3}$ triplets to consider one by one and sequence in $3! = 6$ possible ways. Each permutation is successively added as a constraint and propagated by inference rules to the whole problem. If (i, j) is a pair of the triplet, i must be sequenced before j if all the permutations of the triplet where j is sequenced before i are proved infeasible by constraint propagation.

3.2 Shaving on Greater k-cliques ($k > 3$)

It is possible to generalize the former technique to greater k-cliques. In [2], the " r-set conditions " are local operations based on the generalization of the immediate selections on disjunctive pairs of Carlier and Pinson (case $r = 2$) to subsets of size> 2. We intent to try to evaluate the feasibility and efficiency of this technique when used as a global operations technique.

The principle is exactly the same as the shaving on triplets. Given a disjunctive resource where n tasks must be performed, a k-clique is chosen. The $k!$ possible sequencings are posted successively and constraint propagation is performed on the whole problem. The goal is to derive invariant pair sequencings.

Two obvious shortcomings appear: on the one hand, the cost implied by the $k!$ constraint propagations needed for each k-tuple and on the other hand the $\binom{n}{k}$ k-tuples to consider. The combinatorial explosion of the number of possible permutations for a k-tuple implies the limitation of k. Depending of the size of the problem at hand and of the amount of constraint propagation performed to analyse each permutation, values of k up to 7 or even 8 seem reasonable. For

instance, $k = 7$ means 5040 constraint propagations are necessary to analyse a 7-tuple; for information only, the complete constraint propagation performed by a set of most common rules is obtained in less than a hundredth of second on 10×10 job shop instances on a Pentium II 350 MHz.

To save CPU time, we also have guards to stop analysing a k-tuple without going to the end of the $k!$ propagations as soon as it is shown that no disjunctions can be solved on this k-tuple. The set of disjunctions still solvable on the k-tuple at hand is updated every time one of its permutations is not proved inconsistent, and an empty set triggers the forsaking of the k-tuple. To test a minimal number of permutations on hopeless k-tuples, we generate the permutations in a precise manner. For instance, each permutation is tried in succession with its reverse since failing to prove infeasible both orderings induces the fact that no ordering on a pair of the k-tuple can be decided; therefore the k-tuple can be dropped.

Another factor of cost is the $\binom{n}{k}$ k-tuples by resource one must consider. To keep this shaving technique tractable, we chose to apply it only on the most critical k-tuple of each resource. Caseau and Laburthe clearly show the pertinent insight provided by "task intervals" [7] (sets of tasks viewed as intervals) on the bottlenecks. Their evaluation of a task interval criticality can be used to determine on which sets of tasks apply the above k-tuple shaving technique.

3.3 Experiments: Computation of Lower Bounds

We experimented this k-tuple shaving technique with $k = 7$ to compute lower bounds on 55 job-shop instances from the literature with a total number of operations ranging from 100 to 400. The general results can be summed up by saying that, on the one hand, the shaving based upon triplets only slightly dominates the simple shaving based on pairs while, on the other hand, the more local k-tuple shaving improves all but 2 of the non-optimal lower bounds obtained by applying only the time-windows shaving described in section 2. It seems that the focus put on one critical set per resource brings out much more information than the analysis of the $\binom{n}{3}$ triplets of each resource for a CPU time increase ranging from 2 for the large problems to 6 for the small ones. In fact, the increasing number of triplets or pairs to consider on large problems tend to level the cost difference in favour of the single k-tuple technique.

4 Future Works

4.1 Perspectives in Shaving Techniques

Our k-clique shaving, if promising, is at the moment being quite perfectible. For instance, when enforcing the ordering of a permutation of a k-clique by constraint propagation on the whole problem, the only information we keep concerns the feasibility or not of this ordering. It could be very interesting to incrementally maintain the minimal adjustments performed on the time-windows of tasks j not belonging to the k-clique. After the analysis of all the possible orderings of

this k-clique, these adjustments are valid lower and upper bounds of the earliest starting times and deadlines of these tasks j and can be propagated.

Another way of research could be the improvement of an existing shaving technique. For instance, the double shaving, introduced by Martin and Shmoys in [14], seems to bring a significantly superior amount of characteristics on the problem than the simple shaving. But the recursivity, even if restricted to a single level, induces a time factor increase varying from 100 for a 10×10 problem to several thousands when the problem becomes large, thus making this method untractable on large problems. It could be interesting to study dominance properties and perform experiments to obtain most of the double shaving deductions at a fraction of its cost. Reducing the size of the sets of tasks on which perform recursion when a given task time-window is being shaved seems an interesting prospect.

4.2 Use of Shaving in Neighbourhood Search

Shaving techniques, applied at the root of a search tree, are known to drastically reduce the size of the tree when searching for optimal solutions. Despite that, large job-shops are still out of reach of tree search procedures and one must rely on local search to obtain good solutions in reasonable time.

Neighbourhood search procedures (NSP) for combinatorial scheduling problems as job-shops are today very popular and efficient local search procedures on small to medium instances. Nevertheless, the quality of the solutions tends to drop quickly when size increases. The most efficient NSPs [15] often use very small neighbourhoods without the property of connectivity [13]. Searching for very good solutions on large problems lead to consider either different small non-connective neighbourhoods [9] or a connective but large one [11].

A promising way of research could be the evaluation of the efficiency of cuts provided by strong consistency enforcing techniques such as shaving above the optimum. Usual neighbourhoods in the disjunctive sheduling problems are often based on the swapping of one or more tasks located on a critical path. The eventual use of the disjunctions solved by shaving techniques in an NSP based on swapping of tasks could be proved as worthy to escape from local optima, given a sufficiently important number of permitted iterations. Indeed, if the non-improving swapping of tasks normally chosen by the retraction heuristic has been shown to be inconsistent by a previous consistency enforcing, then no improvement of the best solution can be expected from the path starting with this move.

It remains to be seen whether the rather high cost of shaving techniques would be worthy of their use. We conjecture that this hybrid exploitation of constraints and neighbourhood search could be interesting on large job-shops (e.g. 15×10 and beyond). [2]

[2] The authors want to thank the reviewers for some very valuable and informed comments.

References

1. P. Baptiste and C. Le Pape. A theoretical and experimental comparison of constraint propagation techniques for disjunctive scheduling. In *Proc. of the 14^{th} IJCAI*, Montréal, Canada, 1995.
2. P. Brucker, B. Jurisch, and A. Kraemer. The job shop scheduling problem and immediate selections. *Annals of Operations Research*, 50:73–114, 1994.
3. P. Brucker, B. Jurisch, and B. Sievers. A branch and bound algorithm for the job shop scheduling problem. *Discrete Applied Mathematics*, 49(1):107–127, 1994.
4. J. Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Management Science*, 35(2):164–176, 1989.
5. J. Carlier and E. Pinson. A practical use of Jackson's preemptive schedule for solving the job-shop problem. *Annals of Operations Research*, 26:269–287, 1990.
6. J. Carlier and E. Pinson. Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, 78:146–161, 1994.
7. Y. Caseau and F. Laburthe. Improved CLP scheduling with task intervals. In P. Van Hentenryck, editor, *Proc. of the 11^{th} International Conference on Logic Programming*, pages 369–383, Santa Margherita, Ligure, Italy, 1994. MIT Press.
8. Y. Caseau and F. Laburthe. Improving branch and bound for jobshop scheduling with constraint propagation. In M. Deza, R. Euler, and Y. Manoussakis, editors, *Proc. of the 8^{th} Franco-Japanese–4^{th} Franco-Chinese Conference on Combinatorics and Computer Science*, Brest, France, July 1995.
9. Y. Caseau and F. Laburthe. Effective forget-and-extend heuristics for scheduling problems. In *Proc. of the Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimization Problems*, Ferrara, Italy, 1999.
10. R. Debruyne and C. Bessière. Some practical filtering techniques for the constraint satisfaction. In *Proc. of the 15^{th} IJCAI*, pages 412–417, Nagoya, Japan, 1997.
11. M. Dell'Amico and M.Trubian. Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research*, 41:231–252, 1993.
12. G. Dewess. An existence theorem for packing problems with implications for the computation of optimal schedules. *Optimization*, 25:261–269, 1992.
13. P.J.M. Van Laarhoven, E.H.L Aarts, and J.K. Lenstra. Job shop scheduling by simulated annealing. *Operations Research*, 40(1):113–125, 1992.
14. P. Martin and D.B. Schmoys. A new approach to computing optimal schedules for the job-shop scheduling problem. In *Proc. of the 5^{th} Conference on Integer Programming and Combinatorial Optimization*, 1996.
15. E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job-shop problem. *Management Science*, 42(6):797–813, 1996.
16. L. Peridy. *Le problème de job-shop : arbitrages et ajustements.* Thèse de Doctorat, Université Technologique de Compiègne, 1996.
17. D. Sabin and E. Freuder. Contradicting conventional wisdom in constraint satisfaction. In A. Borning, editor, *Proc. of PPCP-94*, Seattle, USA, May 1994.
18. P. Torres and P. Lopez. A generic algorithm for solving the not-first/not-last problem in disjunctive scheduling. In *Proc. of the 6^{th} International Workshop on Project Management and Scheduling*, pages 305–308, Istanbul, Turkey, 1998.