Computing Explanations for Global Scheduling Constraints

Petr Vilím

Charles University
Faculty of Mathematics and Physics
Malostranské náměstí 2/25, Praha 1, Czech Republic
vilim@kti.mff.cuni.cz

Abstract. Integration of explanations into a CSP solver is a technique addressing difficult question *"why my problem has no solution"*. Besides providing some sort of answer to the user, explanations can be used for debugging, solving dynamic problems, and advanced search algorithms. Explanations work pretty well with simple constraints. However, in order to use explanations together with a global constraint, its filtering algorithm (*i.e.* propagation) has to be enhanced to be explanation-aware. This paper proposes such a technique, suitable for some sort of propagation algorithms (*e.g.* some versions of edge-finding and not-first/not-last). For edge-finding algorithm, this technique is then further improved.

1 Introduction

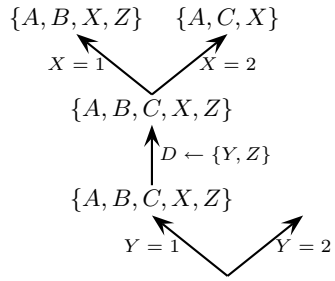
Our goal to to find such a subset of variables that it is not possible to satisfy all the constraints between them. Of course, such subset should be as small as possible, ideally set-wise minimal, but this property is not guaranteed. Let us call such set a *conflict set*.

In order to find such a subset, whenever a filtering algorithm reduces a domain of a variable, an explanation for this reduction is put on the stack. In this paper, the explanation is the set of variables[1] involved in the constraint. Not all of the variables used by the constraint have to be included into explanation, however the domain reduction must be justified only by the current state of the domains of included variables, the domains of the rest of variables do not matter.

When the search comes to a dead end, generally there is a variable with an empty domain. For current state of domains, this variable forms the minimum conflict set we are looking for. Now we can go back through the stack of explanations and reconstruct the conflict set while returning to the last choice-point: each time an explanation for some variable from the conflict set is found, all variables from this explanation are added into the conflict set. Thus, until a domain of same variable from conflict set is changed, the problem has no solution.

[1] Paper [3] defines an (elimination) explanation as a subset of the constraints, not variables. However, if we consider an explanation as a subset of variables, we can omit some unimportant variables used in a global constraint and make the explanation more accurate.

For example, let the domain of the variable A become empty, and let the last explanations on the stack are $A \leftarrow \{B\}$, $C \leftarrow \{A, D, X\}$ and $B \leftarrow \{X, Z\}$. Then the conflict set immediately after the choice-point is $\{A, B, X, Y\}$. Suppose now that in this choice-point we are branching on two possible values 1, 2 of variable X, and the second branch returned conflict set $\{A, C, X\}$. Before the choice-point, the problem has no solution until at least one domain of the variables from these two explanations is changed. Therefore, the the conflict set immediately before the choice-point is $\{A, B, C, X, Z\}$.



We continue to the next choice-point, which is branching on Y; explanation $D \leftarrow \{Y, Z\}$ does not change the conflict set. Notice, that because Y is not in the conflict set, the branch from $Y = 2$ does not have to be explored – it has no solution for the same reason as the branch $Y = 1$.

2 Adapting a Global Constraint for Explanations

There are two main problems with adapting a filtering algorithm for explanations: to find the actual explanation and to not slow the algorithm down too much.

However, if the filtering algorithm has several common properties listed below (*e.g.* edge-finding in [4] and not-first/not-last in [1] have them[2]), there is a general way how to modify the algorithm to record explanations.

Explanations are already known: Some filtering algorithms can be easily modified (without changing their time complexity) so that they know the explanation for each reduction they make. However, writing down each explanation can take $\mathcal{O}(n)$ time, because the explanation can have even n variables. The filtering algorithm usually assumes that domain reduction is $\mathcal{O}(1)$ operation, therefore simple explanation writing can make the algorithm n times slower.

Independence: A frequent property of the filtering algorithms is that all the reductions made by one run of the algorithm are based on the state of the domains *before* the filtering starts and these reductions are *independent* on each other. For example, let the conflict set is $\{A\}$ and a filtering algorithm generated

[2] Their modification is straightforward, because of space limitation I do not include them here.

these two explanations: $A \leftarrow \{B\}$ and $B \leftarrow \{C\}$. Then new conflict set is only $\{A, B\}$, because the reduction of domain D_A is based on the state of the domain D_B before filtering.

2.1 Modification of global constraint

Because the reductions are independent on each other, it is not necessary to record them all. An explanation is needed only the best reduction for each variable. Let us run the algorithm twice. First time we just remember the best found reduction for each variable, second time we find all of the reductions again, but generate explanations only for the best of them. This way the number of generated explanation is $\mathcal{O}(n)$ and the total time of modified algorithm is $2t(n) + \mathcal{O}(n^2)$, where $t(n)$ is the running time of the original algorithm.

In the rest of the paper I show that in the case of edge-finding we can make it even better.

3 Explanations for Disjunctive Resources

The methods presented in this section are designed for disjunctive resource, but they can be easily extended to another resources, *e.g.* batch processing [6], [5]. Another approach to explanations for disjunctive scheduling can be found in [2].

In disjunctive scheduling there is a set of activities T which have to be processed on one resource. The resource can process only one activity at a time, preemption is not allowed. Time needed for processing the activity $i \in T$ is p_i, processing cannot start before time r_i and it has to complete at latest at time d_i. Filtering algorithms usually increase the values r_i or decrease d_i (*i.e.* they reduce the domain $D_i = \{r_i, r_i + 1, \ldots, d_i - p_i\}$).

3.1 Edge-Finding

Edge-finding is based on the following idea: Consider a subset of activities Ω. Let r_Ω denotes the minimum starting time of activities from Ω, *i.e.* $r_\Omega = \min\{r_j,\ j \in \Omega\}$ and similarly $d_\Omega = \max\{d_j,\ j \in \Omega\}$. Minimum time needed for processing the set Ω is $p_\Omega = \sum_{j \in \Omega} p_j$. Now consider an activity $i \notin \Omega$. If the activity i is not processed after all the activities Ω, then processing of Ω ends at first at time $t = \min\{r_i, r_\Omega\} + p_i + p_\Omega$. However, if $t > d_\Omega$ then such processing is impossible. Thus the activity i has to be processed after the set Ω and the release time r_i of the activity i can be increased to $r_i := \max\{r_i, \max\{r_U + p_U,\ U \subseteq \Omega\}\}$.

This way we get the filtering rule and its symmetrical version:

$$\min\{r_i, r_\Omega\} + p_i + p_\Omega > d_i \Rightarrow r_i \geq \max\{r_U + p_U,\ U \subseteq \Omega\}$$
$$\max\{d_i, d_\Omega\} - p_i - p_\Omega < r_i \Rightarrow d_i \leq \min\{d_U - p_U,\ U \subseteq \Omega\}$$

Explanation for each such reduction is exactly the set $\Omega \cup \{i\}$.

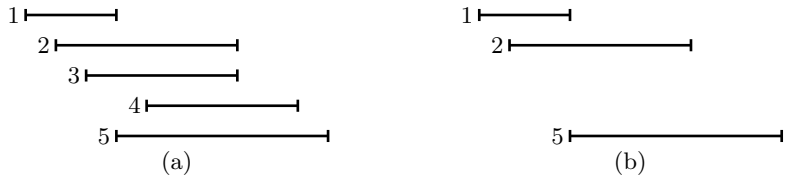
Fortunately, filtering algorithm does not have to consider all the 2^n sets Ω, but only sets in the form of a task interval. Such a set is determined by two activities a, b (possibly the same): $\Omega = \{j,\ j \in T\ \&\ r_j \geq r_a\ \&\ d_j \leq d_b\}$. Therefore to record the set Ω into explanation, it is sufficient to write down the values r_a and d_b, the set Ω can be reconstructed[3] in time $\mathcal{O}(n)$.

So for the edge-finding, it is not necessary to run the filtering algorithm twice as suggested in the previous section. The best reduction and the values r_a, d_b for each variable can be recorded and explanations can be subsequently generated in time $\mathcal{O}(n)$.

Reading the explanations During the step back, the explanations are read from the stack and the conflict set is reconstructed. In the case of a global constraint, we are facing a block of $\mathcal{O}(n)$ explanations; processing each relevant of them takes $\mathcal{O}(n)$, that makes total $\mathcal{O}(n^2)$ time needed. Can we make it more quickly?

A little change in generation of the explanations can make the sequence of d_Ω nondecreasing. Just sort the activities by time d_i (in fact edge-finding already sorted them), then for each activity generate the explanations which have $d_\Omega = d_i$.

Consider a sequence of explanations on the stack. First, ignore all the explanations which are not for the activities from conflict set. Now we have to add into conflict set all the activities from the rest of sets, as shown on the figure (a):



However, not all of these sets have to be considered. From the sets with the same d_Ω, only the one with minimal r_Ω is needed, therefore we can eliminate the set number 3. Also, we can eliminate a set Ω if it is a subset of its consequent set. This way we can eliminate the set number 4. Resulting explanations are on the figure (b), notice that both sequences r_i and d_i are increasing.

Now we can take the activities in nondecreasing order of d_i and use "merge sort idea" to process the explanations:

i:=first activity in ordering;
Ω:=first explanation;
while not out of activities or explanations **do**
begin
  if $d_i > d_\Omega$ **then** Ω:=next explanations;
  else begin

[3] Under the assumption that all changes of the values r_i, d_i are restored during step back together with explanations.

```
      if  r_i ≥ r_Ω  then  add j into nogood;
      j:=next activity in ordering;
    end;
end;
```

Time complexity of this algorithm is $\mathcal{O}(n \log n)$ (because of sorting). This algorithm does not pay off when the number of relevant explanations is small. It is possible to choose the right algorithm at runtime with respect to the number of explanations.

4 Conclusions and Further Work

This paper proposes a method of modifying filtering algorithms to become explanations-aware. This method is suitable for a number of algorithms. Modification of edge-finding algorithm was further improved. Similar technique can be also used for the not-first/not-last algorithm [1].

In the near future I plan to finish the implementation of these algorithms and compare running time and number of backtracks with non-explanation version on some benchmark openshop and jobshop problems.

References

[1] Philippe Baptiste and Claude Le Pape. Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling. In *Proceedings of the Fifteenth Workshop of the U.K. Planning Special Interest Group*, 1996.

[2] Christelle Guéret, Narendra Jussien, and Christian Prins. Using intelligent backtracking to improve branch and bound methods: an application to open-shop problems. *European Journal of Operational Research*, 127(2):344–354, 2000. ISSN 0377-2217. URL http://www.emn.fr/jussien/publications/gueret-EJOR00.pdf.

[3] Narendra Jussien. e-constraints: explanation-based constraint programming. In *CP01 Workshop on User-Interaction in Constraint Satisfaction*, Paphos, Cyprus, 1 December 2001. URL http://www.emn.fr/jussien/publications/jussien-WCP01.pdf.

[4] Paul Martin and David B. Shmoys. A New Approach to Computing Optimal Schedules for the Job-Shop Scheduling Problem. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Proceedings of the 5th International Conference on Integer Programming and Combinatorial Optimization, IPCO'96*, pages 389–403, Vancouver, British Columbia, Canada, 1996.

[5] Petr Vilím. Batch processing with sequence dependent setup times: New results. In *Proceedings of the 4th Workshop of Constraint Programming for Decision and Control, CPDC'02*, 2002.

[6] Petr Vilím and Roman Barták. Filtering algorithms for batch processing with sequence dependent setup times. In *Proceedings of the 6th International Conference on AI Planning and Scheduling, AIPS'02*. The AAAI Press, 2002.