Incremental Propagation of Time Windows on Disjunctive Resources

Roman Barták

Charles University in Prague, Faculty of Mathematics and Physics
Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic
roman.bartak@mff.cuni.cz

Abstract

Constraint-based techniques are frequently used in solving real-life scheduling problems thanks to natural modeling capabilities and strong constraint propagation techniques encoded within global constraints. In this paper we present new incremental propagation rules for shrinking time windows of activities allocated to a disjunctive resource. These rules use information about precedence constraints between the activities and support optional activities.

Introduction

Real-life scheduling problems usually include a variety of constraints so special scheduling algorithms (Brucker 2001) describing a single aspect of the problem can hardly be applied to solve the problem completely. Constraint-based scheduling (Baptiste, Le Pape, and Nuijten 2001) provides a natural framework for modelling and solving real-life problems because it allows integration of different constraints. The above mentioned special scheduling algorithms can be often transformed into propagators for the constraints so the big effort put in developing these algorithms is capitalised in constraint-based scheduling.

Many filtering algorithms for specialised scheduling constraints have been developed in recent years (Baptiste, Le Pape, and Nuijten 2001). In scheduling, these algorithms frequently use global information about time windows of all activities to shrink some of these time windows by removing infeasible time points. Precedence relations between the activities are usually kept there indirectly via the time windows and these relations are not directly accessible. As shown by Vilím (2002) and Laborie (2003), the precedence relations can by used to deduce further pruning of time windows and vice versa, time windows can be used to deduce new precedence relations. In this paper we will propose new incremental propagation rules for this type of reasoning. In particular, we will show how to realise the energy precedence constraint by Laborie (2003) on a disjunctive resource with optional activities.

The paper is organized as follows. We will first describe in detail the problem to be solved. Then we will discuss the existing works in the area and give an overview of our previous work on precedence graphs with optional activities. The main part of the paper will be devoted to the description of incremental propagation rules realising the energy precedence constraint by Laborie (2003). We will also prove the soundness of these rules and give their theoretical worst-case time complexity.

Motivation

In this paper we address the problem of modelling a *disjunctive resource* where activities must be allocated in such a way that they do not overlap in time. Let S_A be a start time of activity A and P_A be its processing time (duration). Then the above feature of the resource can be described by a set of constraints in the form of disjunction $(S_A + P_A \leq S_B \vee S_B + P_B \leq S_A)$ among all pairs of activities allocated to the resource. Therefore the resource is called disjunctive. Another frequent name for this type of resource is a *unary resource* indicating that at most one activity can be processed at any time.

We assume that there are time windows restricting the position of the activities. The time window $[R_A, D_A]$ for activity A specifies that the activity cannot start before R_A (release time) and cannot finish after D_A (deadline). Formally, $R_A \leq S_A \wedge S_A \leq D_A - P_A$. We assume the activity to be non-interruptible so the activity occupies the resource from its start till its completion, that is, for a time interval whose length is equal to the duration of the activity.

We also assume that there are precedence constraints between the activities. The precedence constraint A « B specifies that activity A must not finish later than activity B starts. This relation can be reformulated as $S_A + P_A \leq S_B$. The precedence constraints describe a partial order between the activities. The goal of scheduling is to decide a linear order that satisfies (extends) the partial order (this corresponds to the definition of a unary resource) in such a way that each activity is scheduled within its time window. The ultimate goal is to decide the start time of each activity in such a way that the above constraints are satisfied (discrete time is assumed). It is easy to show that if a feasible linear order of activities is found then a feasible assignment of start times exists.

Last but not least we allow some activities to be so called *optional*. It means that it is not known in advance whether such activities are allocated to the resource or not. If the optional activity is allocated to the resource, that is, it

is included in the final resource schedule then we call this activity *valid*. If the activity is known not to be allocated to the resource then we call the activity *invalid*. In other cases, that is, the activity is not decided to be or not to be allocated to the resource, we call the activity *undecided*. Optional activities are useful for modelling alternative resources for the activities (an optional activity is used for each alternative resource and exactly one optional activity becomes valid) or for modelling alternative processes to accomplish a job (each process may consist of a different set of activities).

Note that for the above defined problem of scheduling with time windows it is known that deciding about an existence of a feasible schedule is NP-hard in the strong sense (Garey and Johnson 1979) even when no precedence relations or optional activities are considered, so there is a little hope even for a pseudo-polynomial solving algorithm. Hence using propagation rules and constraint satisfaction techniques is justified here.

Related Works

Disjunctive resources are studied for a long time in constraint-based scheduling (Baptiste, Le Pape, and Nuijten 2001). There exist algorithms based for example on edge-finding (Baptiste and Le Pape 1996) or not-first/not-last (Torres and Lopez 1997) techniques that restrict the time windows of the activities. Edge-finding deduces that some activity must be processed before or after some subset of activities, while not-first/not-last rules deduce that some activity cannot be processed first or last in a subset of activities. Other algorithms are based on relative ordering of activities, for example filtering based on optimistic and pessimistic resource profiles (Cesta and Stella 1997). Recently, as scheduling and planning technologies are coming together, filtering algorithms combining filtering based on relative ordering and time windows appeared. Detectable precedences by Vilím (2002) are one of the first attempts for such a combination. Laborie (2003) presents a similar rule called energy precedence constraint for reservoir-like resources.

Filtering algorithms for scheduling constraints typically assume that all the constrained activities will be included in the final schedule. This is not always true, for example assume that there are alternative processes to accomplish a job or alternative resources per activity. These alternatives are typically modelled using optional activities that may or may not be included in the final schedule depending on which process or resource is selected. The optional activity may still participate in the constraints but it should not influence other activities until it is known to be in the schedule. This could be realised by allowing the duration of the optional activity to be zero for time-windows based filtering like edge-finding. However, this makes filtering weaker and as shown in (Vilím, Barták, and Čepek 2005) a stronger and faster filtering can be achieved if optional activities are assumed in the filtering algorithm directly. The paper (Focacci, Laborie, and Nuijten 2000) proposed a global precedence graph where alternative resources correspond to paths in the graph, but the graph is used merely for cost-based filtering (optimization of makespan or setup times).

In this paper we address the problem of integrated filtering based on precedence relations and time windows. From the beginning we assume the existence of optional activities. A filtering algorithm for so called detectable precedences with optional activities on a unary resource has been proposed in (Vilím, Barták, and Čepek 2005). This algorithm uses Θ-Λ-tree to achieve O(n.log n) time complexity and it is a monolithic algorithm (must be repeated completely if there is any change of domains). This algorithm handles detectable precedences only and it cannot accommodate precedences given from the external world. The algorithm proposed in this paper can handle any precedence relation.

The energy precedence constraint by Laborie (2003) can compute new bounds for time windows from arbitrary precedences and can discover new precedences from time windows in reservoir type of resource. However, the energy precedence constraint is not defined for optional activities and the paper gave just the principles without describing the filtering algorithm. Our propagation rules extend the work by Laborie to optional activities and we will present the rules in the implementation-friendly way (the rules can be immediately transformed into a filtering algorithm used by a constraint solver). To achieve this goal, we use "light" data structures, namely domains of variables, only. Moreover, the new rules are incremental so they directly react to changes of particular domains rather than running a monolithic algorithm from scratch. Such rules are much easier for implementation and for integration to existing constraint solvers. Our hope is that their incremental nature will also lead to a good practical efficiency.

Last but not least, there exists extensive literature on temporal networks and among them the disjunctive temporal networks (Stergiou and Koubarakis 1998) are the closest approach to our proposal. The main difference of our proposal from (most) works on temporal networks is that we are not solving the temporal problem completely. We propose a filtering algorithm that removes infeasible values. This algorithm is expected to serve as a look-ahead technique in the search frameworks solving the problem. Moreover, we are not covering arbitrary disjunctive temporal network but a specific one modelling a unary resource. By borrowing the ides from scheduling of unary resources we can achieve stronger domain pruning. Finally, we assume optional activities that may or may not appear in the network.

Precedence Graphs

Our propagation rules that will be presented in the next section require information about precedence relations between the activities. This information is usually kept in a *precedence graph* which is an acyclic directed graph where

nodes correspond to activities and there is an arc from A to B if A « B. Moreover, we require fast access to every predecessor and successor of each activity which can be guaranteed if a transitive closure of the graph is kept. Then, information about all predecessors and successors is available in time O(1). The transitive closure of a precedence graph with optional activities can be defined in the following way.

Definition 1: We say that a precedence graph G with optional activities is *transitively closed* if for any two arcs A to B and B to C such that B is a valid activity and A and C are either valid or undecided activities there is also an arc A to C in G.

It is easy to prove that if there is a path from A to B such that A and B are either valid or undecided and all inner nodes in the path are valid then there is also an arc from A to B in a transitively closed graph (by induction on the path length). Hence, if no optional activity is used (all activities are valid) then Definition 1 corresponds to a standard definition of the transitive closure.

In (Barták and Čepek 2005), we proposed a *double precedence graph* that models both standard precedence relations and direct precedence relations. Briefly speaking, the direct precedence between A and B means that A can be directly before B, that is, there is no activity that must be scheduled between A and B. We also proposed there a constraint model for such double precedence graphs. This model includes propagation rules for keeping the transitive closure of the precedence graph as well as for keeping information about direct precedence relations. The model uses the following data structures to model precedence relations (we will not use information about direct precedence relations in this paper, hence we omit their description). Each activity A has a variable $Valid_A$ attached to it. If activity A is valid then $Valid_A = 1$. If activity A is invalid then $Valid_A = 0$. If activity A is undecided then the value of $Valid_A$ is not defined, in particular it is different from 0 and 1 (in terms of constraint satisfaction, the domain of $Valid_A$ for undecided activity A is {0,1}). Information about precedence relations is accessible via the following sets (« is a transitive closure of precedence relations):

CanBeBefore(A) – activities that can be before A
(B ∈ CanBeBefore(A) ⇔ ¬ A « B)

CanBeAfter(A) – activities that can be after A
(B ∈ CanBeAfter(A) ⇔ ¬ B « A)

MustBeBefore(A) – activities that must be before A
(B ∈ MustBeBefore(A) ≡ B « A)

MustBeAfter(A) – activities that must be after A
(B ∈ MustBeAfter(A) ≡ A « B)

Unknown(A) – activities whose position relatively to activity A is unknown
(B ∈ Unknown(A) ⇔ ¬ A « B ∧ ¬ B « A)

Sets MustBeBefore, MustBeAfter, and Unknown are used just to simplify notation – these sets can be derived from the other two sets in the following way (see also Figure 1):

MustBeBefore(A) = CanBeBefore(A) \ CanBeAfter(A)
MustBeAfter(A) = CanBeAfter(A) \ CanBeBefore(A)
Unknown(A) = CanBeBefore(A) ∩ CanBeAfter(A).

Note finally, that if an activity becomes invalid then it is removed from the above sets of all valid and undecided activities, which corresponds to removing the activity from the precedence graph.

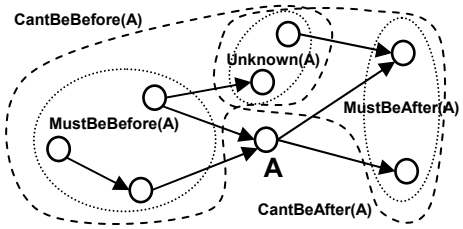

Figure 1. Representation of the precedence graph.

Propagating Time Windows

The standard constraint model for time allocation of the activity assumes two variables – S_A and E_A – describing when the activity A starts and finishes. If the processing time P_A of activity A is constant then the variable E_A is not necessary. However, we use this variable to simplify notation used in the propagation rules. The position of activity A in time is restricted by *release time* R_A and *deadline* D_A that define a *time window* for processing the activity. For a valid activity A, we define the initial domain for the variable S_A to be $[R_A, D_A - P_A]$ and, similarly, the initial domain for the variable E_A to be $[R_A + P_A, D_A]$. For an undecided activity A, we define the initial domain of S_A to be $[R_A, sup]$ and the initial domain for E_A to be $[inf, D_A]$. The reason is that an empty domain means failure in constraint satisfaction, however, if the time window becomes empty for an undecided activity then this activity should become invalid instead of generating a failure. Hence, we use unrestricted domains for undecided activities and we will check emptiness of the time window explicitly in the propagation rules. As soon as the activity becomes valid, domains of time variables are restricted on both sides.

The goal of time window filtering is to remove those time points from the time window in which the activity cannot be processed. Usually, only the lower and upper bounds of the time window change, that is, the earliest start time increases and the latest completion time decreases. Notice that this information is kept for both valid and undecided activities. We will use the following notation to describe bounds of the above domains:

$est_A = min(S_A)$	earliest start time
$lst_A = max(S_A)$	latest start time
$ect_A = min(E_A)$	earliest completion time
$lct_A = max(E_A)$	latest completion time

This notation can be extended in a natural way to sets of activities. Let Ω be a set of activities, then:

$est_\Omega = \min\{est_A \mid A \in \Omega\}$
$lst_\Omega = \max\{lst_A \mid A \in \Omega\}$
$ect_\Omega = \min\{ect_A \mid A \in \Omega\}$
$lct_\Omega = \max\{lct_A \mid A \in \Omega\}$
$p_\Omega = \sum\{p_A \mid A \in \Omega\}$

During propagation, we will be increasing est and decreasing lct of activities which corresponds to shrinking their time windows.

The earliest start time of activity A is influenced by the activities that are processed before A. To be more specific activity A must start after the earliest completion time of any set of valid activities that must be processed before A. This time can be formally expressed using the formula

$$\max\{est_\Omega + p_\Omega \mid \Omega \subseteq \{X \mid X \ll A \wedge Valid_X = 1\}\}.$$

Similarly, the latest completion time of A is influenced by the valid activities that must be processed after A. Hence, we can use the following filtering rules to shrink the time window of activity A:

$est_A \leftarrow \max\{est_\Omega + p_\Omega \mid \Omega \subseteq \{X \mid X \ll A \wedge Valid_X = 1\}\}$
$lct_A \leftarrow \min\{lct_\Omega - p_\Omega \mid \Omega \subseteq \{X \mid A \ll X \wedge Valid_X = 1\}\}$

Notice that only valid activities influence time windows of other (non invalid) activities. This corresponds to our requirement that undecided activities should not influence other activities but they can be influenced. Hence, it may happen that a time window for some undecided activity becomes empty (or too small for the activity) which will lead to invalidation of the activity.

The above two rules are special cases of the *energy precedence constraint* (Laborie 2003) for unary resources. We will now show how to compute these bounds in time $O(n.\log n)$, where n is the number of activities. Let Ω' be the subset of the set $\{X \mid X \ll A \wedge Valid_X = 1\}$ with the maximal $est_{\Omega'} + p_{\Omega'}$ (the set Ω' defines the new bound for est_A). Then $\Omega' \supseteq \{X \mid X \ll A \wedge Valid_X = 1 \wedge est_{\Omega'} \le est_X\}$, because otherwise adding such X to Ω' will increase $p_{\Omega'}$ and hence also $est_{\Omega'} + p_{\Omega'}$. Consequently, it is enough to explore sets $\Omega_{\langle X,A\rangle} = \{Y \mid Y \ll A \wedge Valid_Y = 1 \wedge est_X \le est_Y\}$ for each valid X that must be before A. In particular, the new bound est_A is computed using the formula:

$$\max\{ est_{\Omega_{\langle X,A\rangle}} + p_{\Omega_{\langle X,A\rangle}} \mid X \ll A \wedge Valid_X = 1 \}.$$

This computation can be realised in time $O(n.\log n)$ using the following algorithm. The new bound for est_A is computed in the variable *end*.

dur ← 0
end ← inf
for each $Y \in \{X \mid X \ll A \wedge Valid_X = 1\}$ in the non-increasing order of est_Y **do**
    dur ← *dur* + p_Y
    end ← max(*end*, est_Y + *dur*)

We need time $O(n.\log n)$, where n is the number of activities, to sort the activities and time $O(n)$ for the loop. The bound for lct_A can be computed in a symmetrical way in time $O(n.\log n)$.

If there is any change in the precedence graph like adding a new arc or making some activity valid then the bounds of time windows may be influenced because the sets $\Omega_{\langle X,A\rangle}$ are changed. Also, if bounds of the time window for some activity are changed then the above filtering rules should also be recomputed. We will now present the propagation rules realising the above-described incremental updates. To prove soundness of these rules it is enough to verify that all sets $\Omega_{\langle X,A\rangle}$ that might be influenced by the change in the precedence graph or in the time windows are explored.

Propagating Changes of Time Windows

Let us first describe propagation rules that are evoked when the time window of some activity shrinks. If bounds of a time window change then we can deduce several conclusions. First, if the time window becomes too small for the activity, formally $est_A + p_A > lct_A$, then either failure is detected, if activity A is valid, or undecided activity A is made invalid. Second, time windows can be used to deduce a new precedence between activities. In particular, if $est_A + p_A + p_B > lct_B$ then activity A cannot be processed before activity B and hence we can deduce $B \ll A$. This is called a *detectable precedence* in (Vilím 2002). Third, if the time window of valid activity A changes then it influences time windows of activities whose relative position to a given activity is known (they are before or after the given activity).

Let us now focus on changes of time windows enforced by an increase of the earliest start time of valid activity A (a symmetrical deduction can be done for a decrease of the latest completion time). Clearly, this increase may influence earliest start time of all valid or undecided activities B that must be after A. Recall that $est_B \leftarrow \max\{ est_{\Omega_{\langle X,B\rangle}} + p_{\Omega_{\langle X,B\rangle}} \mid X \ll B \wedge Valid_X = 1 \}$ and $\Omega_{\langle X,B\rangle} = \{Y \mid Y \ll B \wedge Valid_Y = 1 \wedge est_X \le est_Y\}$. If est_A increases then $est_{\Omega_{\langle A,B\rangle}}$ increases as well and hence it may define a new bound for est_B. Moreover, if est_A increases then A may become a new member of some set $\Omega_{\langle C,B\rangle}$, such that $est_C \le est_A$ and $\underline{est}_A < est_C$, where $\underline{est}_A$ is the original value of est_A. Hence $p_{\Omega_{\langle C,B\rangle}}$ will increase by p_A and so it may define a new bound for est_B. Together, we need to explore all sets $\Omega_{\langle X,B\rangle}$ such that $est_X \le est_A$ and $\underline{est}_A < est_X$. Note that it also includes the set $\Omega_{\langle A,B\rangle}$. Also if $est_C = est_A$ and $\underline{est}_A < est_C$ then $\Omega_{\langle C,B\rangle} = \Omega_{\langle A,B\rangle}$ so it is not necessary to explore $\Omega_{\langle C,B\rangle}$ separately. Finally, there is no other set $\Omega_{\langle X,B\rangle}$ that is influenced by the increase of the earliest start time of activity A.

The following propagation rule /1/ reacts to increase of the earliest start time. "est_A is increased" is its trigger meaning that the rule is invoked any time when est_A is increased. The part after → is a propagator describing pruning of domains. "exit" means that the constraint represented by the propagation rule is entailed so the propagator is not further invoked (its invocation does not cause further domain pruning). "fail" means a failure so no solution satisfying the constraints exists. We will use the same notation in all propagation rules.

```
est_A is increased →                                          /1/
  if Valid_A = 0 then exit
  else if est_A + p_A > lct_A then
      if Valid_A = 1 then fail
      else Valid_A ← 0; exit
  else
      for each B∈Unknown(A) do
          if est_A + p_A + p_B > lct_B then
              add B«A          /* detectable precedence */
      if Valid_A = 1 then
          ect_A ← est_A + p_A
          for each B∈MustBeAfter(A) do
              dur ← ∑{p_X | X « B ∧ est_A ≤ est_X ∧ Valid_X = 1}
              end ← est_A + dur
              for each Y∈{X | X « B ∧ Valid_X = 1 ∧
                                  est_A < est_X ∧ est_X < est_A }
                      in the non-increasing order of est_Y do
                  dur ← dur + p_Y
                  end ← max(end, est_Y + dur)
              est_B ← max(end, est_B)
```

Symmetrically, rule /2/ updates lct_B for activities B such that B « A, if necessary. Recall, that $\underline{lct}_A$ is the value of lct_A before the decrease. Together, the propagation rules /1/ and /2/ incrementally maintain the energy precedence constraint after changes of time windows.

```
lct_A is decreased →                                          /2/
  if Valid_A = 0 then exit
  else if est_A + p_A > lct_A then
      if Valid_A = 1 then fail
      else Valid_A ← 0; exit
  else
      for each B∈Unknown(A) do
          if est_B + p_B + p_A > lct_A then
              add A«B          /* detectable precedence */
      if Valid_A = 1 then
          lst_A ← lct_A – p_A
          for each B∈MustBeBefore(A) do
              dur ← ∑{p_X | B « X ∧ lct_X ≤ lct_A ∧ Valid_X = 1}
              start ← lct_A – dur
              for each Y∈{X | B « X ∧ Valid_X = 1 ∧
                                  lct_X < lct_A ∧ lct_A < lct_X }
                      in the non-decreasing order of lct_Y do
                  dur ← dur + p_Y
                  start ← min(start, lct_Y – dur)
              lct_B ← min(start, lct_B)
```

Time complexity of rules /1/ and /2/ is $O(n^2)$, where n is a number of activities. We need time $O(n.\log n)$ to sort the activities and time $O(n^2)$ to process the nested loops. Note finally, that if the propagation rules change est or lct of some activity then the propagation rules for these changes are also invoked and so on (the above time complexity is for a single run of the rule). This ensures that any change is propagated through the precedence graph. The open question is whether it is necessary to update all successors and predecessors of a given activity or whether it is enough to update just the direct predecessors and direct successors and propagate the change through them. This will not decrease the worst-case time complexity but, if possible, it may improve practical time efficiency.

Propagating Changes in the Precedence Graph

The second group of propagation rules realises the energy precedence constraint in an incremental way by reacting to changes in the precedence graph. The rules are invoked by making the activity valid or by adding a new precedence relation.

Assume that some activity A becomes valid. Then, this activity may influence other activities in the precedence graph if their relative position to activity A is known. In particular, if there is an activity B such that A « B then A will become a member of sets $\Omega_{\langle X,B\rangle}$ such that X « B, $Valid_X = 1$ and $est_X \le est_A$ (this also includes a new set $\Omega_{\langle A,B\rangle}$). These sets are used to compute the earliest start time of B and hence they should be re-explored. The other sets $\Omega_{\langle X,B\rangle}$ used to update est_B have already been explored or will be explored when calling the rules for some valid activity in $\Omega_{\langle X,B\rangle}$. A symmetrical analysis can be done for activities B before A. The propagation rule /3/ realises the above deduction.

```
Valid_A is instantiated →                                     /3/
  if Valid_A = 1 then
      ect_A ← est_A + p_A
      lst_A ← lct_A – p_A
      for each B∈MustBeAfter(A) do
          dur ← ∑{p_X | X « B ∧ est_A ≤ est_X ∧ Valid_X = 1}
          end ← est_A + dur
          for each Y∈{X | X « B ∧ est_X < est_A ∧ Valid_X = 1}
                  in the non-increasing order of est_Y do
              dur ← dur + p_Y
              end ← max(end, est_Y + dur)
          est_B ← max(end, est_B)
      for each B∈MustBeBefore(A) do
          dur ← ∑{p_X | B « X ∧ lct_X ≤ lct_A ∧ Valid_X = 1}
          start ← lct_A – dur
          for each Y∈{X | B « X ∧ lct_A < lct_X ∧ Valid_X = 1}
                  in the non-decreasing order of lct_Y do
              dur ← dur + p_Y
              start ← min(start, lct_Y – dur)
          lct_B ← min(start, lct_B)
  exit
```

Deduction similar to making an activity valid can be used if a new arc, say A « B, is added to the precedence graph. Note, than in such a case activities A and B are either valid or undecided (neither A nor B is invalid). If A is valid when A « B is added then A will become a member of sets $\Omega_{\langle X,B\rangle}$ such that X « B, $Valid_X = 1$ and $est_X \le est_A$ (including a new set $\Omega_{\langle A,B\rangle}$) and these sets must be re-explored to find out whether they lead to change of est_B or not. Note, that we are exploring only the influence of A to B rather than exploring influence to all successors of B.

This is because any successor C of B that is not yet a successor of A will be explored when the arc A « C is added to the graph by incremental maintenance of the transitive closure as described in (Barták and Čepek 2005). Similarly, if B is valid when arc A « B is added then we must explore its possible influence on lct_A. The above deductions are realised by the propagation rule /4/.

A«B is added → /4/
  if $Valid_A = 1$ **then**
    $dur \leftarrow \sum\{p_X \mid X \text{ « } B \wedge est_A \leq est_X \wedge Valid_X = 1\}$
    $end \leftarrow est_A + dur$
    for each $Y \in \{X \mid X \text{ « } B \wedge est_X < est_A \wedge Valid_X = 1\}$
        in the non-increasing order of est_Y **do**
      $dur \leftarrow dur + p_Y$
      $end \leftarrow \max(end, est_Y + dur)$
    $est_B \leftarrow \max(end, est_B)$
  if $Valid_B = 1$ **then**
    $dur \leftarrow \sum\{p_X \mid A \text{ « } X \wedge lct_X \leq lct_B \wedge Valid_X = 1\}$
    $start \leftarrow lct_B - dur$
    for each $Y \in \{X \mid A \text{ « } X \wedge lct_B < lct_X \wedge Valid_X = 1\}$
        in the non-decreasing order of lct_Y **do**
      $dur \leftarrow dur + p_Y$
      $start \leftarrow \min(start, lct_Y - dur)$
    $lct_A \leftarrow \min(start, lct_A)$
  exit

Let us now analyze the worst-case time complexity of the above propagation rules. Time complexity of the propagation rule /3/ is $O(n^2)$, where n is a number of activities. We need time $O(n.\log n)$ to sort the activities and time $O(n^2)$ to process the nested loops. Again, this time complexity does not include invocation of rules /1/ and /2/ after changes of est_B or lst_B. Time complexity of the propagation rule /4/ is $O(n.\log n)$, where n is a number of activities. We need time $O(n.\log n)$ to sort the activities and time $O(n)$ to process the loops.

Conclusions

The main contribution of the paper is description of filtering algorithms for incremental maintenance of energy precedence constraint from (Laborie 2003) applied to a disjunctive resource with optional activities. Rather than proposing a monolithic algorithm, we focused on incremental propagation of changes and on description of rules in the form trigger-propagator that is used in existing constraint solvers. We extended Laborie's work by using optional activities, we proved soundness of the proposed propagation rules and we described their theoretical worst-case time complexity. We assumed fix duration of activities but the propagation rules can be easily extended to variable duration by assuming a minimal duration in the presented rules and by adding a new rule reacting to the increase of the minimal duration. There are still some open questions concerning incrementality of the rules, namely, whether it is possible to propagate some information only to direct predecessors and successors of the activity while still keeping the same filtering power.

Acknowledgements

The research is supported by the Czech Science Foundation under the contract no. 201/04/1102. My thanks go to Ondřej Čepek for proof-reading of the paper draft and to anonymous reviewers for pointing my attention to temporal networks.

References

Baptiste, P., and Le Pape, C. 1996. Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling. In *Proceedings of the Fifteenth Workshop of the U.K. Planning Special Interest Group (PlanSIG)*.

Baptiste, P.; Le Pape, C.; and Nuijten, W. 2001. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers.

Barták, R., and Čepek, O. 2005. Double Precedence Graphs. In *Proceedings of the 24th Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG)*, 28–33. London: City University.

Brucker P. 2001. *Scheduling Algorithms*. Springer Verlag.

Cesta, A., and Stella, C. 1997. A Time and Resource Problem for Planning Architectures. In *Recent Advances in AI Planning (ECP'97)*, 117–129. Springer Verlag.

Focacci, F.; Laborie, P.; and Nuijten, W. 2000. Solving Scheduling Problems with Setup Times and Alternative Resources. In *Proceedings of AIPS 2000*.

Garey, M. R., and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, San Francisco.

Laborie, P. 2003. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence* 143: 151–188.

Stergiou, K., and Koubarakis, M. 1998. Backtracking algorithms for disjunctions of temporal constraints. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 248–253. AAAI Press.

Torres, P., and Lopez, P. 1999. On Not-First/Not-Last conditions in disjunctive scheduling. *European Journal of Operational Research* 127: 332–343.

Vilím, P. 2002. Batch Processing with Sequence Dependent Setup Times: New Results. In *Proceedings of the 4th Workshop of Constraint Programming for Decision and Control, CPDC'02*. Gliwice, Poland.

Vilím, P.; Barták, R.; and Čepek, O. 2005. Extension of O(n.log n) filtering algorithms for the unary resource constraint to optional activities. *Constraints* 10(4): 403–425.