

Dynamic task sequencing in temporal problems with uncertainty*

M.-J. Huguet
INSA and LAAS-CNRS
7, avenue du Colonel Roche
F-31077 Toulouse cdex
huguet@laas.fr

P. Lopez
LAAS-CNRS
7, avenue du Colonel Roche
F-31077 Toulouse cdex
lopez@laas.fr

T. Vidal
ENIT/LGP/PA
47, avenue Azereix – BP 1629
F-65016 Tarbes cdex
thierry@enit.fr

Abstract

In classical planning and scheduling approaches, a task schedule is first designed off line, then executed on line, simply releasing tasks at times compatible with temporal domains of their starting time-points. When durations of tasks are uncertain, one wishes to keep as much flexibility on line as possible so as to release each task according to effective durations taken by previous ones. One also wishes to ensure off line that the on-line schedule will be feasible whatever the uncertain durations will be, which has been called temporal controllability of the plan. Such proactive reasoning both leads to predictive schedules that are more robust when executed on line, and also to easier and more effective on-line rescheduling when needed. Going further, the notion of sequentiability has been defined with respect to resource constraints. It means here the ability to decide on line the sequencing of tasks that use the same discrete resource, according to effective durations taken by previous ones. In the non-uncertain framework, algorithms exist to prune the search and detect so-called forbidden precedences among tasks. In this paper we show how these techniques can be extended in temporal problems with uncertainty.

Introduction

Temporal Constraint Satisfaction Problems, and particularly Simple Temporal Problems (STPs) (Dechter, Meiri, & Pearl 1991; Schwalb & Dechter 1997) are frequently used in planning and scheduling applications that involve quantitative time constraints (e.g. (Laborie & Ghallab 1995; Morris, Muscettola, & Tsamardinos 1998)), as they allow fast checking of temporal consistency. A duration between two time-points or the temporal domain of a time-point (set of possible times of occurrence) are represented through intervals of possible values. However this formalism does not adequately address an important aspect of real execution domains: the time of occurrence of some events may not be under the complete control of the execution agent. For example, on a building site, a task might wait for a supply truck, which arrival time is dependent on the traffic, while the task duration itself might depend on the weather conditions. In such cases, the execution agent does not have freedom to select the precise delay between events. Instead, the value

is selected by Nature independently of the agent's choices. This can lead to constraint violations during execution even if the STP appeared consistent at plan generation time.

The problem of constraint satisfaction for STPs with Uncertainty was addressed formally in (Vidal & Fargier 1999). Uncertainty means here the effective duration of a task or the effective delay between two particular times (start or end of tasks) still lie within allowed bounds but cannot be decided, and will hence be observed on line during execution. In this setting, the question of temporal feasibility goes beyond mere consistency to encompass the main issue of *Dynamic Controllability*. Essentially, a network is dynamically controllable if there is a strategy for executing on line the time-points under the agent's control that satisfy all requirements. Such a property must consider that the agent will apply the strategy in a chronological way: to decide when to execute next task, she might take advantage of observations made on the occurrence of past uncontrolled events, but she must decide without knowing the effective durations of tasks still to come. Dynamic controllability was proven to be tractable (Morris, Muscettola, & Vidal 2001), through the application of a mere local consistency checking algorithm.

Actually the Dynamic Controllability (checked off line) means the ability to postpone effective timing of tasks until executing them on line, but resting assured that no constraint will ever be violated, whatever the observations are. This can be viewed as a least-commitment approach adding flexibility to the planning and execution loop, still ensuring the plan safe execution. In other words, our approach is proactive in the sense that most of the reasoning is made off line, to prove that a schedule will be feasible. But actual scheduling (i.e. assigning times to starting times of tasks), though being now straightforward, is made on line.

In scheduling, complex resource constraints must also be accounted for (Pinedo & Chao 1999). We are interested here in non-preemptive tasks (i.e. they cannot be interrupted) and disjunctive resources (i.e. discrete resources with capacity equal to one). In this framework two tasks competing for the same resource need be sequenced. For example, the same crane might be needed for two unloading tasks that should be both processed within a given time window. Usually, when executing an STP, such decisions have already been made (and the added precedence link proven consistent), since a constraint such as *task i before or after task j*

*AIPS 2002 Workshop on On-line Planning and Scheduling, pp.41–48, April 24, 2002, Toulouse, France

cannot be expressed through simple binary constraints between time-points. Nevertheless, in domains with temporal uncertainties, it might be the case that one sequencing choice is compulsory to make the network Dynamically Controllable in some situations, while the reverse choice is needed in other situations, none being valid in all situations. This calls for Dynamic Sequencing strategies, which means postponing such decisions until execution. We would then rather be able to check off line that this might be done on line without constraint violations. Such a new property has been called Dynamic Sequentiability (Vidal & Bidot 2001).

In general cases, this property cannot be checked in polynomial time since sequencing is already an NP-complete problem in many scheduling problems without uncertainty (Lenstra, Rinnooy Kan, & Brucker 1977). But there a number of propagation techniques exist to filter out values from temporal domains, that are not compatible with the sequencing constraints, allowing to speed up the search for a feasible schedule or to detect early an inconsistency (Baptiste, Le Pape, & Nuijten 2001). For instance the Forbidden Precedence rule checks whether one of the two possible sequences may be proven infeasible with respect to temporal domains of start and end times of two tasks i and j (Erschler, Roubellat, & Vernhes 1976; Torres & Lopez 2000). A more elaborated one, the Extended Forbidden Precedence rule, which is based on *energetic reasoning* (Lopez & Esquirol 1996), checks it taking also into account all other tasks that use the same resource and might occur between i and j .

In this paper, after recalling some background on the topic in section 2, we will focus in section 3 on the Forbidden Precedence rule; we will show how the rule should be written in the framework of STPs with uncertainty. Section 4 will follow the same lines addressing the Extended Forbidden Precedence rule. We will conclude with a couple of words about foreseen extensions of the work.

Background

A Simple Temporal Network (STN) (Dechter, Meiri, & Pearl 1991) is an STP represented as a graph $\langle V, G \rangle$ in which the vertices in V are the time-points that are the variables of the problem, while edges (or links) in G are binary numerical constraints g_{xy} , in the shape of simple intervals $[l(g_{xy}), u(g_{xy})]$ of possible durations between two time-points x and y . Please note that the inverse constraint implicitly exists and is $g_{yx} = [l(g_{yx}), u(g_{yx})] = [-u(g_{xy}), -l(g_{xy})]$, and that no specific constraint between x and y results in the initial interval $]-\infty, +\infty[$.

To check global consistency of an STN, one might use filtering techniques, namely arc-consistency (AC) and path-consistency (PC) techniques that both run in polynomial time in STNs. PC for instance is a local shortest path propagation algorithm: it computes any binary constraint g_{xy} between points x and y by intersecting it with all paths going through a third time-point z :

$$g_{xy} = g_{xy} \cap (g_{xz} \odot g_{zx})$$

which amounts to, considering that composition simply sums up the lower and upper bounds of the intervals, and

intersection takes the max of the lower bounds and the min of the upper bounds:

$$g_{xy} = [\max(l(g_{xy}), l(g_{xz}) + l(g_{zy})), \min(u(g_{xy}), u(g_{xz}) + u(g_{zy}))]$$

AC merely updates the temporal domain of each time-point y (i.e. the interval of possible times for y), by computing the sum of the temporal domain of another time-point x and the interval expressing the duration between x and y . This may be seen as a specific and more restricted case of PC since a temporal domain of x is the duration interval between the origin of time 0 and x . The advantage of PC is that the *complete minimal network* is computed: for any two time-points x and y , PC provides the duration interval containing these and only values that are consistent with the other constraints of the problem. Such strength of PC will be widely used in this paper.

Figure 1 illustrates AC and PC through small examples.

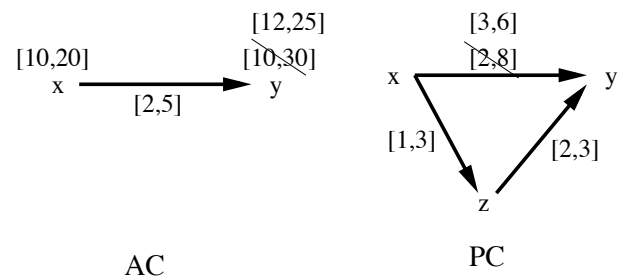


Figure 1: AC and PC algorithms

A Simple Temporal Network with Uncertainty (STNU) is similar to an STN except a subset of G called C represent specific links called *contingent*, which may be thought of as representing causal processes of uncertain duration; their finish timepoints, called *contingent* timepoints, are controlled by Nature, subject to the limits imposed by the bounds on the contingent links. All other timepoints, called *executable* timepoints, are controlled by the agent. Thus, an STNU is a 3-tuple $\Gamma = \langle V, G, C \rangle$. We require $0 < l(c) < u(c) < \infty$ for each contingent link $c \in C$.

An STNU may be regarded as a family of STNs: a *projection* (Vidal & Fargier 1999) of Γ is an STN derived from Γ , replacing each contingent link c by an interval with equal upper and lower bounds $[v, v]$ for some v such that $l(c) \leq v \leq u(c)$. The set of values v for all the contingent links represent one *situation* that the executing agent might face on line, when durations of contingent links are eventually observed. Then one can define a *schedule* as an assignment of fixed times to all time-points, and an *execution strategy* as a mapping from the set of projections (or situations) to the set of schedules. An execution strategy is *viable* if and only if for all situations the associated schedule is consistent.

Global consistency needed to be redefined in terms of *controllabilities*. We will not get into the details of these properties in this paper (see (Morris, Muscettola, & Vidal 2001) for details), but instead just focus on the most relevant one, the *Dynamic Controllability*: in short, an STNU is

dynamically controllable if and only if there exists a viable execution strategy that can be carried out on line *as far as* the contingent durations are observed. Thus, a Dynamic execution strategy might be safely run on line, since it assigns a time to each executable timepoint that may depend on the outcomes of contingent links in the past, but not on those in the future (or present). This corresponds to requiring that only information available from observation may be used in determining the schedule on line.

To check dynamic controllability, one might first run a PC algorithm. If a contingent link is squeezed, then it means some of its uncontrollable values are not consistent, therefore the problem will be infeasible for such values. But this is not enough, since the contrary is not true in general. A further PC-like filtering algorithm that still runs in polynomial time might be designed to get the minimal STNU. This algorithm called 3DC+ (Morris, Muscettola, & Vidal 2001) either proves inconsistency or provides the executing agent with duration intervals restricted to dynamically controllable values only, making it possible to safely and easily execute the schedule on line according to observations made. We just quickly recall here the basics of this algorithm (see (Morris, Muscettola, & Vidal 2001) for details): one needs only considering triangles xyz in which one constraint g_{xy} is contingent (if two are contingent the triangle will be considered twice, and if all three links are contingent the triangle is trivially not dynamically controllable). We consider $g_{xy} = [u, v]$, $g_{xz} = [a, b]$ and $g_{zy} = [c, d]$ (figure 2; a contingent constraint is depicted as a dotted arrow).

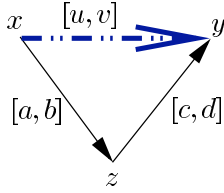


Figure 2: Triangular network for 3DC+

1. case 1: if $d < 0$, then necessarily y is before z and no further restriction is necessary.
2. case 2: if $c \geq 0$, then necessarily z will be executed before y , hence without having observed it yet, therefore g_{xz} must be restricted so that things will work fine whatever values are taken by g_{xy} , which raises $g_{xz} = [a, b] \cap [v - d, u - c]$. For instance, the initial network of figure 3(a) is stable after PC, but is further restricted by 3DC+ to the network of figure 3(b).
3. case 3: $c < 0$ and $d \geq 0$, i.e. z might occur before or after y . Here the general rule to apply is to update $g_{xz} = [a, b] \cap [\min(v - d, u), b]$ and in the case where $v - d > u$, one needs to add a *wait* $\langle y, v - d \rangle$ on the link g_{xz} , which means that after activating x , one should wait either for the occurrence of y or at least $v - d$ time units before activating z . Such *wait* constraints are added to the STNU model and *regressed*, which is a backward propagation process, still made locally through triangles, and

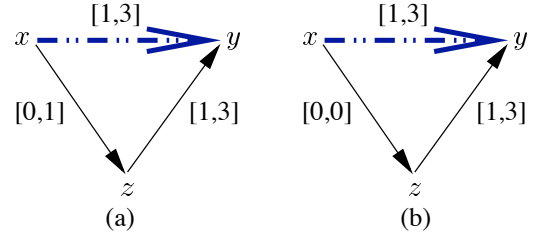


Figure 3: Illustration of 3DC+ case 2

thus still tractable. Figure 4 presents an example where $v - d > u$. As previously, the figure shows the initial network (a) and the network propagated by 3DC+ (b).

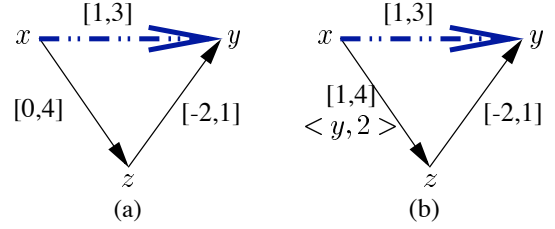


Figure 4: Illustration of 3DC+ case 3

Existing work only focused on temporal constraints, disregarding resource constraints that are of high significance in planning and scheduling applications. As mentioned in the introduction, a disjunctive resource compels all tasks needing this resource to be sequenced. Sequencing decisions, just as task activation time decisions, should be left to the on-line executing agent, since in some situations one sequencing will be needed while the inverse choice must be taken in another situation. Anyhow, to ease feasibility checking of the problem and detect some sequencing that are forbidden anyway, filtering algorithms over sequencing decisions are of high added value. Efficient though non-complete ones exist in classical scheduling; this paper aims at adapting them in the uncertain framework.

In the remaining of the paper, we will use the following notations for each task k :

- $k^- \in [k^-, \bar{k}^-]$: start time-point
- $k^+ \in [k^+, \bar{k}^+]$: end time-point
- $p_k = k^+ - k^- \in [p_k, \bar{p}_k]$: duration
- $w_k^\Delta \in [w_k^\Delta, \bar{w}_k^\Delta]$: consumption over an interval Δ

Forbidden Precedence

The reader should keep in mind that the meaning of a forbidden precedence is that adding this precedence to the problem (before execution) would lead to an inconsistency. When temporal uncertainties are accounted for, it is enough to find one situation in which adding this precedence would make the projection inconsistent: such a sequencing may be decided on line when facing a situation that supports it, but it

should not be added during off-line scheduling when all situations are still likely to occur. We will first recall the classical way of expressing the rule in scheduling without temporal uncertainty. This has not been extended to the STN; we need to do this first before extending to the STNU.

The STN context

Let us consider two (non contingent) tasks i and j which must be sequenced (for instance because they compete for the same resource).

Classical rule formulation in scheduling Proposition 1, illustrated by figure 5, allows us to conclude that “ i before j ” is forbidden (denoted by $i \not\prec j$) (Erschler, Roubellat, & Vernhes 1976).

Proposition 1 If $\bar{j}^+ - \underline{i}^- < \underline{p}_i + \underline{p}_j$ then $i \not\prec j$.

Proof. \underline{i}^- is the earliest possible start time for i , therefore $\underline{i}^- + \underline{p}_i$ is the earliest possible end time for i . Similarly, $\bar{j}^+ - \underline{p}_j$ is the latest possible start time for j . Therefore $i \prec j$ implies that $\underline{i}^- + \underline{p}_i < \bar{j}^+ - \underline{p}_j$. Reverting this proposition gets to Proposition 1. \square

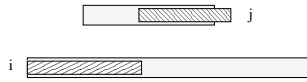


Figure 5: Forbidden precedence $i \not\prec j$

New formulation based on a minimal STN One can see the previous rule uses information on the possible times of i^- and j^+ , which are in an STN the temporal domains, i.e. the duration between 0 and the time-point.

That may be generalized in a complete minimal network obtained through a PC propagation (figure 6). In this minimal network, a forbidden precedence between i and j , exists if and only if a lower bound of the link j^-i^+ is positive, that means j^- is before i^+ , which forbids $i \prec j$. It yields the following proposition.

Proposition 2 If $a' > 0$ then $i \not\prec j$.

Symmetrically if $c' > 0$ then $j \not\prec i$.

Proof. In a complete minimal network obtained through a PC propagation, all paths from j^- to i^+ have been searched for. Therefore in the edge j^-i^+ valued by $[a', b']$, a' is the highest lower bound for j^-i^+ . Then $a' > 0$ means $i \not\prec j$. \square

Proposition 3 Proposition 2 subsumes Proposition 1.

Proof. The proof is straightforward since after PC propagation a' stands for the greatest minimal path between j^- and i^+ . In particular $a' \geq \underline{p}_j - \bar{j}^+ + \underline{i}^- + \underline{p}_i$. \square

Adaptation to STNUs

We will now assume that some of the tasks may be contingent, which means we need considering an STNU prop-

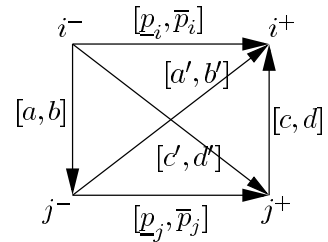


Figure 6: STN after propagation through PC

agated through 3DC+ instead of an STN. In figure 6, that means the links i^-i^+ and j^-j^+ may be contingent¹.

For instance let i be contingent. After propagation a' corresponds to a shortest path from j^- to i^+ , but it might result from a path going through i^- , being the sum $-b + \underline{p}_i$. Since i is contingent we must consider the worst situation in which this path would take longer, this is when i takes its upper bound \bar{p}_i . A precedence is indeed forbidden if there exists at least one situation in which it is forbidden. Otherwise the off-line scheduling process could be allowed to enforce such a precedence in the network, which would lead to a possible failure at execution time if i takes its upper bound.

So let us look at the case where $p_i = \bar{p}_i$. One can see there will be a problem if $\bar{p}_i > b$ since j^- would necessarily be released by the execution process before i^+ has occurred. It is easy to check that this new condition $\bar{p}_i - b > 0$ subsumes the previous one $a' > 0$, that is $\bar{p}_i - b \geq a'$. Indeed, if only PC is applied to this network, the link i^-j^- valued by $[a, b]$ is intersected with the path $i^-i^+j^-$, i.e. $[\underline{p}_i, \bar{p}_i] \odot [-b', -a']$, then one has $b \leq \bar{p}_i - a'$. Since 3DC+ will only restrict further the value of the link i^-j^- , the property $\bar{p}_i - b \geq a'$ is still enforced by 3DC+. In other words, there might be cases in which $\bar{p}_i - b > 0$ while $a' < 0$ as the next example will show. Considering all other possible cases, one gets the following updated rule.

Proposition 4 If i is not contingent and $a' > 0$ then $i \not\prec j$.

If i is contingent and $\bar{p}_i - b > 0$ then $i \not\prec j$.

If j is contingent and $\bar{p}_j + c > 0$ then $i \not\prec j$.

Symmetrically,

If j is not contingent and $c' > 0$ then $j \not\prec i$.

If i is contingent and $\bar{p}_i - d > 0$ then $j \not\prec i$.

If j is contingent and $\bar{p}_j + a > 0$ then $j \not\prec i$.

In the example of figure 7, i is a contingent task while j is not. Propagation through PC provides us with $[-1, 9]$ on the link j^-i^+ . Condition $i \not\prec j$ might be deduced by Proposition 4 (since $\bar{p}_i - b = 5 - 4 > 0$), but cannot be deduced considering the length of the shortest path from i^+ to j^- , that is a' , since $a' = -1 < 0$.

Extended forbidden precedence

We will now introduce a more effective rule in terms of deduction by taking into account other tasks that compete for

¹We still consider that other links are not contingent, which is usually the case in real-life planning, but our scheme might easily be extended to cases with contingent links between tasks.

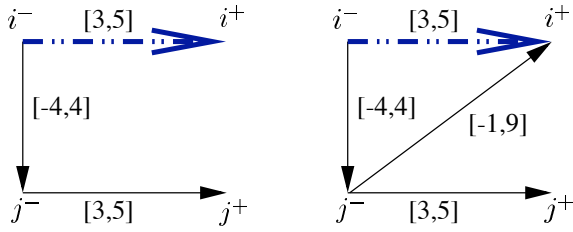


Figure 7: Illustrative example of Proposition 4

the same resource and that are to occur within the same reference interval. For that purpose we need to define the *minimal consumption* of a task k over a reference interval Δ (Esquirol, Lopez, & Huguet 2001). As in the previous section, we first give mathematical expressions as classically stated in scheduling, improve them for the STN framework, then adapt them in the STNU.

Classical rule formulation in scheduling

Consumption We denote by w_k^Δ the consumption of task k (i.e. how long k uses the resource) over a reference interval $\Delta = t_1 t_2$. Two cases must be distinguished:

1. $k^- k^+ \cap \Delta = \emptyset \implies w_k^\Delta = 0$;
2. $k^- k^+ \cap \Delta \neq \emptyset \implies w_k^\Delta = \min(k^+, t_2) - \max(k^-, t_1)$.

This is illustrated by figure 8 where striped areas represent the consumption of each task between t_1 and t_2 .

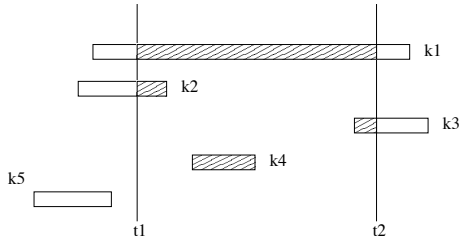


Figure 8: Consumption of five tasks

One then gets:

$$w_k^\Delta = \max[0, \min(k^+, t_2) - \max(k^-, t_1)]$$

which amounts to, knowing that $k^+ - k^- = p_k$

$$w_k^\Delta = \max[0, \min(p_k, t_2 - t_1, k^+ - t_1, t_2 - k^-)] \quad (1)$$

One is usually especially interested in computing the lower and upper bounds of the consumption: for the consumption of task k over interval Δ , we might derive from equation (1) the *minimal* (or *necessary*) consumption noted \underline{w}_k^Δ , and the maximal consumption noted \overline{w}_k^Δ .

The former is obtained by considering the minimal duration of the task, and by shifting it to its left and right utmost positions, retaining the minimum value of all intersections between such positions and the reference interval Δ . That is illustrated in figure 9 and raises:

$$\underline{w}_k^\Delta = \max[0, \min(\underline{p}_k, t_2 - t_1, \underline{k}^+ - t_1, t_2 - \overline{k}^-)] \quad (2)$$

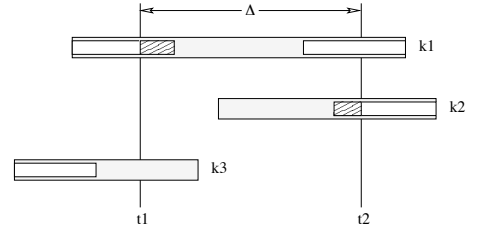


Figure 9: Minimal consumption of three tasks

The maximal consumption is on the contrary obtained by considering the maximal duration and positions which intersection with interval Δ is maximal:

$$\overline{w}_k^\Delta = \max[0, \min(\overline{p}_k, t_2 - t_1, \overline{k}^+ - t_1, t_2 - \underline{k}^-)] \quad (3)$$

The relevant notion for our purpose is obviously the minimal consumption: when trying to check whether i before j is feasible, we intend to take into account that another task k will *necessarily* consume the resource, between i^- and j^+ , for *at least* some time T . Therefore we will not consider anymore the maximal consumption in the remainder of the paper.

Rule formulation based on temporal domains In the STN context, extending the forbidden precedence rule means taking as a reference interval $\Delta = \underline{i}^- \overline{j}^+$. Proposition 5, illustrated by figure 10, allows the deduction of a forbidden precedence between tasks i and j by considering the minimal consumptions of other tasks competing for the same resource over Δ (Lopez & Esquirol 1996; Esquirol, Lopez, & Huguet 2001).

Proposition 5 If $\overline{j}^+ - \underline{i}^- < \underline{p}_i + \underline{p}_j + \sum_{k \neq i, j} \underline{w}_k^{\underline{i}^- \overline{j}^+}$ then $i \not\prec j$.

The proof is straightforward since the minimal consumption expresses the necessity that the other tasks will consume the resource over a subset of the interval.

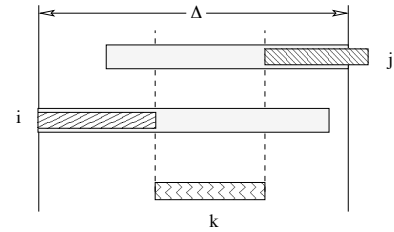


Figure 10: Extended forbidden precedence $i \not\prec j$ accounting for k

New formulation in the STN context

General formulation of the consumption Let us consider the minimal consumption of a task k between i^- and j^+ according to the outcome of a PC algorithm in an STN. Figure 11 illustrates the situation will all relevant links.

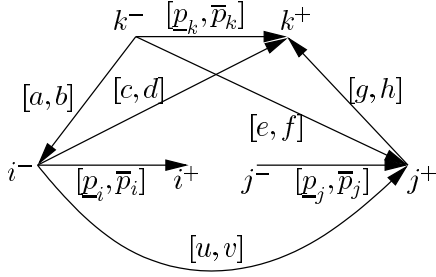


Figure 11: Minimal consumption of a task k in an STN

Here $\Delta = i^-j^+ = [u, v]$ is the reference interval. The general formulation of the minimal consumption of k over this interval is as follows:

$$\underline{w}_k^\Delta = \max\{0, \min[p_k, v, \max(c, p_k - b), \max(e, p_k - h)]\} \quad (4)$$

Let us justify the formulation. The two former terms p_k and v correspond to the obvious cases, when the task occurs fully outside Δ , and when it covers the interval Δ , respectively.

The third term is when the task is shifted to the left. In such a situation i^-k^+ will take its lower bound c , and actually the interval i^-k^+ looks like the amount of k that lies after i^- , therefore c should be the quantity to consider. But one may also consider k is left-shifted when k^-i^- takes its upper bound. Then b represents the part of k that lies outside the reference interval. Therefore the quantity $p_k - b$ could be considered as the minimal consumption of k . The dominating value will be the maximal one, since a propagation algorithm retains the max of all lower bounds. Similar reasoning leads to e and $p_k - h$ as possible quantities to represent the minimal consumption of k when it is right-shifted.

Now all that follows consists in determining which quantity subsumes the other, distinguishing between contingent and non-contingent cases for k .

Extended forbidden precedence rule in a minimal STN

Formula (4) can be simplified considering how PC updates the links. Composition of intervals $[-b, -a]$ and $[p_k, \bar{p}_k]$ raises $[p_k - b, \bar{p}_k - a]$ which is intersected with $[c, d]$. Since the network is stable $p_k - b$ should not update c , therefore $c \geq p_k - b$. Similarly $e \geq p_k - h$. We should then use c for a task shifted to the left and e for a task shifted to the right.

In the network of figure 12, after a propagation through PC one gets: $p_k - b = 1 - 1 = 0$ and $c = 1$; $p_k - h = 1 - (-1) = 2$ and $e = 4$.

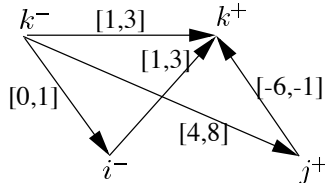


Figure 12: Example for calculating the consumption

The minimal consumption is then expressed by:

$$\underline{w}_k^\Delta = \max[0, \min(p_k, v, c, e)] \quad (5)$$

To get the Extended Forbidden Precedence Rule, one needs to consider the reference interval as being $i^-j^+ = [u, v]$. Proposition 5 ends up being:

Proposition 6 If $v < p_i + p_j + \sum_{k \neq i, j} \underline{w}_k^{i^-j^+}$ then $i \not\prec j$.

The following example (figure 13) illustrates that Proposition 6 subsumes the classical formulation of Proposition 5.

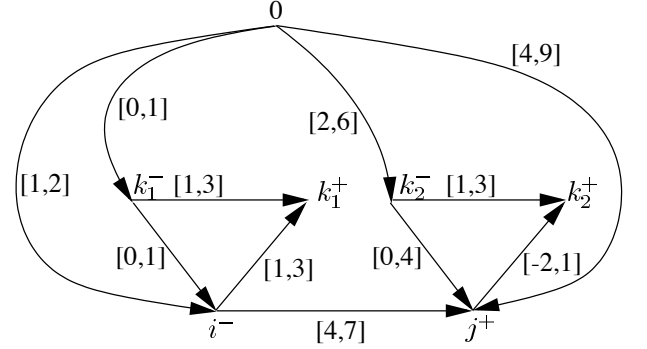


Figure 13: Illustrative example of Proposition 6

For the minimal consumption over the interval $\Delta = i^-j^+$, with the original formulation:

$$\underline{w}_k^\Delta = \max[0, \min(p_k, t_2 - t_1, k^+ - t_1, t_2 - \bar{k}^-)]$$

one gets $\underline{w}_{k_1}^\Delta = \max[0, \min(1, 9 - 1, 1 - 1, 9 - 1)] = 0$ and

$\underline{w}_{k_2}^\Delta = \max[0, \min(1, 9 - 1, 3 - 1, 9 - 6)] = 1$. Over the

interval $\Delta = i^-j^+$, with the new formulation

$$\underline{w}_k^\Delta = \max[0, \min(p_k, v, c, e)]$$

one gets $\underline{w}_{k_1}^\Delta = \max[0, \min(1, 7, 1, 4)] = 1$ and $\underline{w}_{k_2}^\Delta =$

$\max[0, \min(1, 7, 2, 0)] = 0$. Then if $p_i = 4$ and $p_j = 3$, with Proposition 5 the test $8 < 4 + 3 + 0 + 1$ does not permit any deduction, while rule 6 provides us with the test $7 < 4 + 3 + 1 + 0$ which implies $i \not\prec j$.

Adaptation to STNUs

We now suppose that k is contingent.

Minimal consumption The formulation of the minimal consumption is based upon the one in STNs with k not being contingent (formula (4)). For each k contingent, one must consider the upper duration \bar{p}_k instead of p_k since, as said before, we should manage to infer a forbidden precedence if at least one situation forbids it, and since the consumption will always be higher with k taking its upper duration it is enough to consider the situation with $p_k = \bar{p}_k$. We hence get the following general formulation:

$$\underline{w}_k^\Delta = \max\{0, \min[\bar{p}_k, v, \max(c, \bar{p}_k - b), \max(e, \bar{p}_k - h)]\} \quad (6)$$

As in STNs, we will now try to simplify the two latter terms of this formula, i.e. for a task k shifted to the left and shifted to the right. Figure 14 shows which are the links of interests in these two cases.

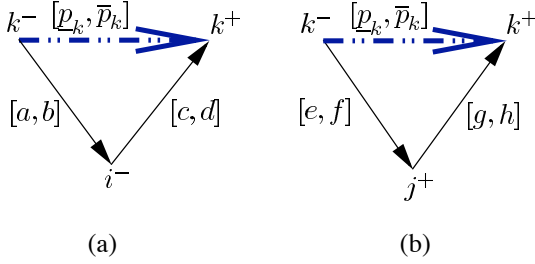


Figure 14: A left-shifted (a) and right-shifted (b) task

First, if only a classical PC algorithm was run on the network, then one would get in the former case $\bar{p}_k - c \geq b$ (otherwise as seen before b would have been updated). That amounts to $\bar{p}_k - b \geq c$, which allows us to simplify the term for a task shifted to the left, only retaining the stronger term $\bar{p}_k - b$. Similarly, for a task shifted to the right, $\bar{p}_k - e \geq h$ (otherwise h would have been updated by the PC algorithm), which amounts to $\bar{p}_k - h \geq e$.

This is actually enough to get the simplification, since 3DC+ will only restrict the constraint domains further and therefore can only make the inequalities stronger. Anyhow we will consider all cases of 3DC+ to show how the result holds. This algorithm indeed restricts the constraints in different ways according to the relative placement of i^- , j^+ , and the ending points of k .

• **Task shifted to the left**

1. $d < 0$ (i^- after k^+).

3DC+ does not make any specific restriction, therefore the former result simply holds. Moreover, in this case task k will necessarily be before i^- , thus outside the reference interval: since $c < d < 0$ the term $\bar{p}_k - c$ is greater and hence dominated by \bar{p}_k in formula (6).

2. $c \geq 0$ (i^- comes before k^+).

With 3DC+, one gets $b \leq \underline{p}_k - c$, thus $\bar{p}_k - b \geq \underline{p}_k - b \geq c$.

In the example of figure 3 (where k^- stands for x , k^+ for y , and i^- for z), the initial network (a) cannot be updated by PC; one then gets $\bar{p}_k - b = 3 - 1 = 2$ and $c = 1$. In the final network (b), 3DC+ has restricted further value b . This allows us to deduce a higher necessary consumption since now $\bar{p}_k - b = 3 - 0 = 3$.

3. $c < 0$ and $d \geq 0$.

With 3DC+, only a is further restricted. Adding a *wait* on k^-i^- will also only affect the lower bound of k^-i^- , but the value of the *wait* still must be lower than b . Therefore here as in case 1 of 3DC+ the former result $\bar{p}_k - b \geq c$ holds from PC updates.

• **Task shifted to the right**

We refer here to figure 14(b).

1. $h < 0$ (j^+ after k^+).

Here again, 3DC+ provides no further restriction. Moreover, in this special case k will be fully contained within the interval Δ : since $h < 0$, the term $\bar{p}_k - h$ will be greater and hence dominated by \bar{p}_k in formula (6).

2. $g \geq 0$ (j^+ before k^+).

With 3DC+, one gets $e \geq \bar{p}_k - h$. With PC we had $e \leq \bar{p}_k - h$, which means necessarily $\bar{p}_k - h = e$. This is a property of 3DC+, that makes both terms equivalent here. We will choose $\bar{p}_k - h$ to remain consistent with other cases.

Using the propagated network of figure 3 (where k^- stands for x , k^+ for y , and j^+ for z), one gets $e = 0$ and $\bar{p}_k - h = 3 - 3 = 0$.

3. $g < 0$ et $h \geq 0$.

With 3DC+, one gets $e \leftarrow \max(e, \min(\bar{p}_k - h, \underline{p}_k))$, which amounts to two distinct cases:

- If $\bar{p}_k - h \leq \underline{p}_k$ then $e \geq \bar{p}_k - h$. Since we had $e \leq \bar{p}_k - h$ from PC, we get $e = \bar{p}_k - h$: as for case 2, this is a special case where both terms could be used;
- If $\bar{p}_k - h > \underline{p}_k$ then e might simply be updated with \underline{p}_k , which means $e \geq \underline{p}_k$, which as said before does not affect the result of PC: $\bar{p}_k - h \geq e$. The only interesting part is that one might get a *wait* of $< k^+, \bar{p}_k - h >$ on k^-j^+ . That means in the worst case the lower bound e will be increased up to $\bar{p}_k - h$ if k^+ does not occur before. Which still entails $\bar{p}_k - h \geq e$.

In figure 4, on the network obtained by 3DC+ (b) (where k^- stands for x , k^+ for y , and j^+ for z), one gets $e = 1$ and $\bar{p}_k - h = 3 - 1 = 2$.

As a matter of conclusion, when k is contingent, the minimal consumption of k over a reference interval Δ is:

$$\underline{w}_k^\Delta = \max\{0, \min[\bar{p}_k, v, \bar{p}_k - b, \bar{p}_k - h]\} \quad (7)$$

Extended forbidden precedence rule in an STNU The reference interval being $\Delta = i^-j^+ = [u, v]$, the rule is similar to Proposition 6, but the minimal consumption $\underline{w}_k^{i^-j^+}$ will be computed from equation (5) if k is not contingent and from equation (7) if k is contingent. Moreover, the terms \underline{p}_i and \underline{p}_j should be replaced by \bar{p}_i if i is contingent and \bar{p}_j if j is contingent.

Proposition 7 *i and j not contingent: If $v < \underline{p}_i + \underline{p}_j + \sum_{k \neq i, j} \underline{w}_k^{i^-j^+}$ then $i \not\prec j$.*
i and j contingent: If $v < \bar{p}_i + \bar{p}_j + \sum_{k \neq i, j} \underline{w}_k^{i^-j^+}$ then $i \not\prec j$.
i contingent and j not contingent: If $v < \bar{p}_i + \underline{p}_j + \sum_{k \neq i, j} \underline{w}_k^{i^-j^+}$ then $i \not\prec j$.
i not contingent and j contingent: If $v < \underline{p}_i + \bar{p}_j + \sum_{k \neq i, j} \underline{w}_k^{i^-j^+}$ then $i \not\prec j$.

We consider the same example as in figure 13, where now k_1 and k_2 are contingent. Figure 15 shows the network obtained after 3DC+ propagation.

