A $O(n^2 \log(n))$ propagation for the Energy Reasoning

Nicolas Bonifas[1,2]

[1] LIX, Ecole Polytechnique, 91128 Palaiseau Cedex, France
nicolas.bonifas@polytechnique.edu
[2] IBM France Lab, 9 rue de Verdun 94250 Gentilly, France

Mots-clés : *constraint-based scheduling, constraint programming, scheduling, global constraints, energy reasoning, energetic reasoning, discrete cumulative resource, cumulative resource, cumulative constraint, propagation, line sweep algorithm, envelope of line segments.*

1 Introduction

Constraint programming has seen many successes in different areas of optimization, notably in scheduling [2]. In this paradigm, *variables* (e.g. the start times of tasks) are declared with an associated *domain* (possible values the variable can take), and relations between the variables are expressed in the form of *constraints*. Summarily, the resolution procedure consists in searching the domain space (for example in a branch-and-bound fashion) until either an incompatibility between domains is detected, in which case we backtrack, or all domains are reduced to a singleton, in which case we have a feasible solution. To improve the efficiency of the search, we apply after each domain reduction so-called *propagations*, which further reduce the domains by enforcing necessary conditions on the constraints for domains of variables to be compatible. Efficient propagations are the key to constraint programming performance.

A crucial constraint in constraint-based scheduling is the discrete cumulative resource, introduced in [1], an abstraction for the assignment of a fixed quantity of a resource to a task during its execution period. At any point in time t, the total resource consumption of the tasks being run must not exceed the total resource capacity. This abstraction is heavily used in constraint-based scheduling to model parallel machines, reservoirs, money, manpower, limited resource allocation. Formally, a *discrete cumulative resource* of capacity C on a set of n tasks with respective length p_i and resource consumption c_i is satisfied if and only if there exist dates s_i such that

$$\forall \text{ time } t, \quad \sum_{\substack{i \in [1,n] \\ s_i \leq t < s_i + p_i}} c_i \leq C$$

.

Propagations for the cumulative constraint include the *energy reasoning*, introduced in [5], the *cumulative edge-finding* technique, introduced in [6], then significantly improved with the introduction of timetable edge finding in [9], and then of the timetable extended edge finding in [7], and the not-first, not-last [6] algorithm, whose complexity was reduced to $O(n^2 \log n)$ in [8].

Energy reasoning has been known for 25 years and is the strongest of those propagations (with the exception of not-first, not-last, none of those propagations dominating each other) for the cumulative constraint (full propagation is NP-hard). Improving the propagation of energy reasoning has sparked some interest recently [4], but the best algorithm to date has a complexity of $O(n^3)$ for n tasks. This high complexity is the reason why this propagation is seldom used in practice and other, weaker but faster, propagations were introduced.

In this article, we introduce techniques to propagate energy reasoning in time $O(n^2 \log n)$. This is an important theoretical advance to a long-standing open question. Moreover, our experiments suggest that this algorithm should also be of practical interest for difficult problems.

Our approach is based on three novel ideas, which form the next sections of the article. First, we proved that the so-called *additional rule* supersedes the other energy reasoning rules in the literature, and we can study this case only. We then show that detecting a propagation with the

additional rule reduces to computing the maximum of a set of piecewise affine functions with special properties that we make use of. Finally, we give an algorithm to efficiently compute this maximum by using the point-line duality of projective geometry to reduce the problem to a convex hull computation. We conclude by discussing this new method and giving experimental results.

2 Notations and energy reasoning rule

Given a cumulative resource of capacity C, tasks of consumption c_i and of length p_i, we respectively denote est_i and lst_i their earliest starting time and latest starting time, in the context of constraint-based scheduling. We say that a task is *left-shifted* if we try to schedule it as early as possible, that is from est_i to $est_i + p_i$, and similarly we say that it is *right-shifted* if we try to schedule it as late as possible, that is from lst_i to $lst_i + p_i$.

We also denote $W_i(a, b)$ with $b \geq a$ the *intersection energy* of task i in the interval $[a, b)$, that is $W_i(a, b) = c_i \cdot \min(b - a, p_i^+(a), p_i^-(b))$ with $p_i^+(a)$ being the length of time during which task i executes after a if it is left-shifted, and $p_i^-(b)$ being the length of time during which task i executes before b if it is right-shifted. The expression of $p_i^+(a)$ in terms of the est_i and lst_i variables is $p_i^+(a) = \max(0, \min(p_i, est_i + p_i - a))$ and $p_i^-(b) = \max(0, \min(p_i, b - lst_i))$.

$W_{\neq i}(a, b)$ is the sum of the intersection energies of all tasks different from i: $W_{\neq i}(a, b) = \sum_{j \neq i} W_j(a, b)$.

Finally, $W(a, b)$ is the sum of the intersection energies of all tasks: $W(a, b) = \sum_i W_i(a, b)$.

In the rest of this article, the propagations will be described in the case when we shift the task under consideration as much as possible to the left. There is obviously a symmetric version of all of them in the case when we shift the task to the right.

The energy reasoning rule is:

$$\text{If } W_{\neq i}(a, b) + c_i \cdot \min(b - a, p_i^+(a)) > C(b - a)$$
$$\text{then } est_i \geq b - \frac{C(b - a) - W_{\neq i}(a, b)}{c_i}$$

.

One can show that this rule supersedes the other rules in the literature.

3 Rule for detecting excess of intersection energy

We will now establish a simple condition to detect when the energy reasoning rule applies. Its application condition is:

$$\begin{aligned} & C \cdot (b - a) - c_i \cdot \min(p_i^+(a), b - a) < W_{\neq i}(a, b) \\ \Leftrightarrow\ & C \cdot (b - a) - c_i \cdot \min(p_i^+(a), b - a) < W(a, b) - W_i(a, b) \\ \Leftrightarrow\ & C \cdot (b - a) - W(a, b) < c_i \cdot \min(p_i^+(a), b - a) - W_i(a, b) \end{aligned}$$

The left-hand side of this inequality is constant on a time window, and the right-hand side depends on task i only. For each window, we thus compute $\max_i \left(c_i \cdot \min(p_i^+(a), b - a) - W_i(a, b)\right)$. If this value does not exceed $C \cdot (b - a) - W(a, b)$, there is no excess of intersection energy. If the maximum exceeds $C \cdot (b - a) - W(a, b)$, we know that we should propagate on the task $\text{argmax}_i \left(c_i \cdot \min(p_i^+(a), b - a) - W_i(a, b)\right)$ (and maybe on other tasks, but since our goal is just to detect if there is or not a propagation, we ignore these for now).

4 Efficient detection of intervals with excess intersection energy

As we saw in the previous section, we can find all intervals on which to propagate if we compute $\text{argmax}_i \left(c_i \cdot \min(p_i^+(a), b - a) - W_i(a, b)\right)$ for all windows $[a, b)$. We will now show a way of finding the maximum of $c_i \cdot \min(p_i^+(a), b - a) - W_i(a, b)$ for all dates b in time $O(n \log n)$.

We define the function $f_{i,a}(b) := c_i \cdot \min(p_i^+(a), b - a) - W_i(a, b)$ for $b \geq a$. This function is continuous and has a very simple shape in relation with the positions of the left and right shifts. More precisely, there are three cases to distinguish, depending on the relative positions of a and the two shifts:

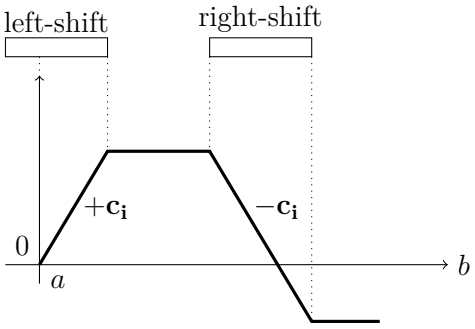

Figure 1: General shape of $f_{i,a}$.

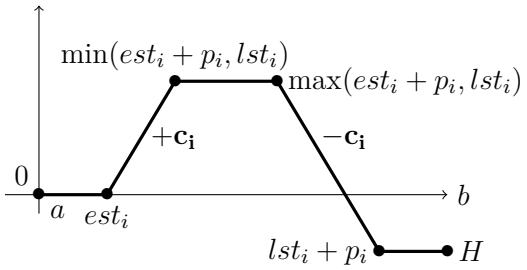

Figure 2: $f_{i,a}$ when $est_i > a$.

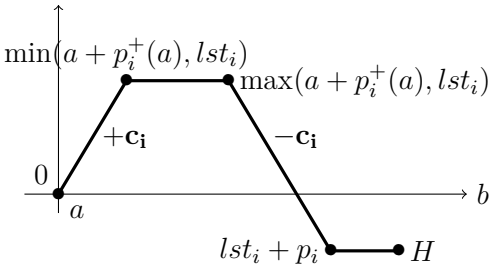

Figure 3: $f_{i,a}$ when $est_i \leq a < lst_i$.

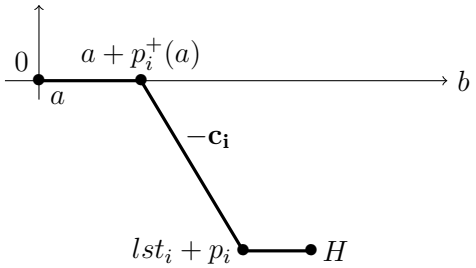

Figure 4: $f_{i,a}$ when $a \geq lst_i$.

In all of these three cases, the function is continuous, piecewise-linear, with a slope of $+c_i$, 0 or $-c_i$ depending on the piece, and the coordinates of the slope changes are indicated on the figures above. H is the planning horizon (upper bound on the makespan).

Remember that we want to compute the supremum over i of the $f_{i,a}$ functions. Since these functions have a very special structure, we will resort to geometry instead of analysis to compute their supremum. Specifically, we decompose these functions into line segments, and use the algorithm described in section 15.3.2 of [3] to compute the upper envelope of these segments. This will exactly yield the function $\max_i f_{i,a}$.

In our context, this algorithm works as follows: The end dates of the $O(n)$ line segments are projected on the x-axis and define $O(n)$ non-overlapping intervals. A balanced binary tree is built, whose leaves are these non-overlapping intervals in ascending order. To a node of the tree corresponds the union of the intervals of the leaves of the subtree. Each line segment is assigned to the node deepest in the tree which contains the projected endpoints of the segment. The upper envelope of all the segments assigned to one node can be computed in time linear with the number of segments at the node. Now, since the upper envelopes of two nodes at the same level in the tree do not overlap (by construction of the tree), we can merge them in linear time, and we can merge all the envelope of one level of the tree in time $O(n \log n)$. Finally, we can merge the envelopes of the $O(\log n)$ levels of the tree in time $O(n\alpha(n) \log\log n)$ (cf. section 15.3.2 of [3]), which is $O(n \log n)$.

We now have an algorithm to compute the supremum of $O(n)$ line segments in time $O(n \log n)$.

5 Complete algorithm and complexity analysis

An important property of energy reasoning, which we make use of, is that when we have n tasks, only $O(n)$ starting dates a have to be considered as candidates for the intervals $[a, b)$ [5, 2, 4]. More precisely, when the tasks are left-shifted, the a dates to be considered are the $O(n)$ dates in the set $O_1 = \{est_i, 1 \leq i \leq n\} \cup \{lst_i, 1 \leq i \leq n\} \cup \{est_i + p_i, 1 \leq i \leq n\}$ (cf. Proposition 19 in chapter 3 of [2]). With our algorithm, we will study at once all dates b located after a.

Therefore there are only $O(n)$ interesting starting dates to consider, on which we might propagate. We start by precomputing all $O(n^2)$ values $W(a, b)$ in time $O(n^2)$ with the algorithm described in section 3.3.6.2 of [2].

Then, for every interesting date a, we use the algorithm of the previous section to detect interesting intervals. Since there are $O(n)$ dates to study and that the study of each of them

has complexity $O(n \log n)$, the complexity of this step is $O(n^2 \log n)$. If we find a propagation, we adjust the earliest starting time of the task according to the additional energy reasoning rule, which is done in constant time. Finally, the total complexity with this algorithm of finding one date adjustment, or finding that there is nothing to propagate, is $O(n^2 \log n)$.

Moreover, we run the energetic checker (cf. [2]) before our algorithm. The energetic checker is a $O(n^2)$ test that either guarantees that energy reasoning will not find a propagation, in which case we do not need to run the energy reasoning, or that it will find one without telling which one, in which case we do run the energy reasoning.

6 Discussion

In a similar fashion to what is done with classical algorithms to propagate the energy reasoning, we can then re-apply this algorithm on the adjusted dates until we reach a fixpoint and there is nothing else to propagate.

Since this algorithm will detect at least one excess of intersection energy if there is one, and we showed already that energy reasoning propagates exactly when there is an excess of intersection energy, this algorithm will make at least one energy reasoning adjustment if the classical $O(n^3)$ algorithm would have done so. The propagation we get is equivalent to the $O(n^3)$ algorithm.

One difference though is in the rare case where the intersection energy is exceeded for several tasks on the same interval $[a, b)$. In this case this algorithm will only tighten the bound for the task with the highest excess of interval energy, while the original algorithm would have made the adjustments for all the tasks. We believe that in the context of constraint programming, when propagations are rare and we need to run the algorithm iteratively until we reach the fix point anyway, the relevant question is "Given that the $O(n^2)$ checker reports that there is something to propagate, how to find one such propagation as quickly as possible?" In this case our algorithm needs time $O(n^2 \log n)$ versus $O(n^3)$ for the original algorithm. More generally, if there are $k \leq n$ tasks for which to propagate, our algorithm needs time $O(kn^2 \log n)$ versus $O(n^3)$ for the original algorithm and thus constitutes an improvement.

We implemented this algorithm on top of IBM Ilog CP Optimizer 12.6 and noticed that this improved algorithm gives a significant performance improvement in practice, on hard instances which require the use of energy reasoning.

References

[1] Abderrahmane Aggoun and Nicolas Beldiceanu. Extending chip in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling*, 17(7):57–73, 1993.

[2] Philippe Baptiste, Claude Le Pape, and Wim Nuijten. *Constraint-based scheduling: applying constraint programming to scheduling problems*, volume 39. Springer, 2001.

[3] Jean-Daniel Boissonnat, Mariette Yvinec, and Hervé Brönnimann. *Algorithmic geometry*, volume 5. Cambridge university press Cambridge, 1998.

[4] Alban Derrien and Thierry Petit. A new characterization of relevant intervals for energetic reasoning. In *Principles and Practice of Constraint Programming*, pages 289–297. Springer, 2014.

[5] Jacques Erschler and Pierre Lopez. Energy-based approach for task scheduling under time and resources constraints. In *2nd international workshop on project management and scheduling*, pages 115–121, 1990.

[6] Wim Nuijten. *Time and resource constrained scheduling: a constraint satisfaction approach*. PhD thesis, Technische Universiteit Eindhoven, 1994.

[7] Pierre Ouellet and Claude-Guy Quimper. Time-table extended-edge-finding for the cumulative constraint. In *Principles and Practice of Constraint Programming*, pages 562–577. Springer, 2013.

[8] Andreas Schutt and Armin Wolf. A new $o(n^2 \log n)$ not-first/not-last pruning algorithm for cumulative resource constraints. In *Principles and Practice of Constraint Programming–CP 2010*, pages 445–459. Springer, 2010.

[9] Petr Vilím. Timetable edge finding filtering algorithm for discrete cumulative resources. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 230–245. Springer, 2011.