

An Energy Cost Aware Cumulative

Helmut Simonis and Tarik Hadzic*

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{h.simonis,t.hadzic}@4c.ucc.ie

Abstract. We motivate and introduce an extension of the well-known cumulative constraint which deals with time and volume dependent cost of resources. Our research is primarily interested in scheduling problems under time and volume variable electricity costs, but the constraint equally applies to manpower scheduling when hourly rates differ over time and/or extra personnel incur higher hourly rates. We present a number of possible lower bounds on the cost.

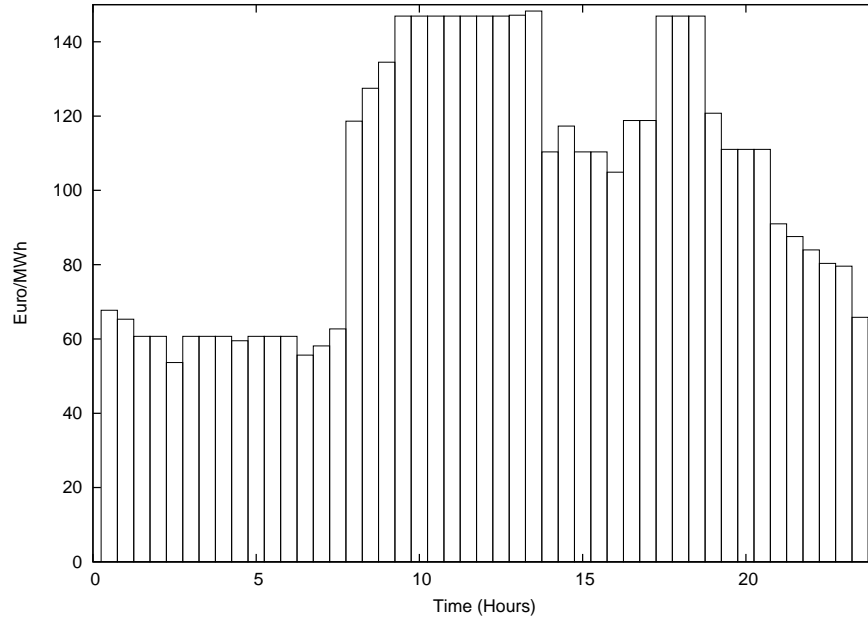
1 Introduction

The cumulative constraint [1, 2, 5] has long been a key global constraint allowing the effective modeling and resolution of complex scheduling problems with constraint programming. However, it is not adequate to handle problems where resource costs change with time and use, and thus must be considered as part of the scheduling. This problem increasingly arises with electricity cost, where time variable tariffs become more and more widespread. With the prices for electricity rising globally, the contribution of energy costs to total manufacturing cost is steadily increasing, thus making an effective solution of this problem more and more important. Figure 1 gives an example of the whole-sale price in the Irish electricity market for a sample day, in half hour intervals. Note that the range of prices varies by more than a factor of three, and the largest difference between two consecutive half-hour time periods is more than 50 units. Hence, large cost savings might be possible if the schedule of electricity-consuming activities takes that cost into account.

The problem of time and volume dependent resource cost equally arises in manpower scheduling, where hourly rates can depend on time, and extra personnel typically incurs higher costs. We suggest extending the standard CP scheduling framework to variable resource cost scenarios by developing a cost-aware extension of the cumulative constraint, `CumulativeCost`. The new constraint should support both the standard feasibility reasoning as well as reasoning about the cost of the schedule. As a first step in this direction we formally describe the semantics of `CumulativeCost` and discuss a number of algorithms for producing *lower-bounds* (used for bounding or pruning during optimization) for this constraint.

* This work was supported by Science Foundation Ireland (Grant Numbers 05/IN/I886 and 05/IN.1/I886 TIDA 09).

Fig. 1. Irish Electricity Price (Source: <http://allislandmarket.com/>)



2 The CumulativeCost Constraint

We start by formally defining our new constraint. It is an extension of the classical cumulative constraint [1]

$$\text{Cumulative}([s_1, s_2, \dots, s_n], [d_1, d_2, \dots, d_n], [r_1, r_2, \dots, r_n], l, p),$$

describing n tasks with start s_i , fixed duration d_i and resource use r_i , with an overall resource limit l and a scheduling period end p . Our new constraint is denoted as

$$\text{CumulativeCost}(\text{Areas}, \text{Tasks}, l, p, \text{cost}).$$

The *Areas* argument is a collection of m areas $\{A_1, \dots, A_m\}$, which do not overlap and partition the entire available resource area $[0, p] \times [0, l]$. Each area A_j has a fixed position x_j, y_j , fixed width w_j and height h_j , and fixed per-unit cost c_j . In our electricity example, in some environments, a limited amount of renewable energy may be available at low marginal cost, generated by wind-power or reclaimed process heat. Depending on the tariff, the electricity price may also be linked to the current consumption, enforcing higher values if an agreed limit is exceeded. We choose the numbering of the areas so that they are ordered by non-decreasing cost ($i \leq j \Rightarrow c_i \leq c_j$). There could be more than one area defined over the same time slot t (possibly spanning over other time-slots as well). If that is the case, we require that the area "above" has a higher cost.

The *Tasks* argument is a collection of n tasks. Each task T_i is described by its start s_i , and fixed duration d_i and resource use r_i . For a given task allocation, variable a_j states how many resource units of area A_j are used. Finally, we can define our constraint:

Definition 1. *Constraint `CumulativeCost` expresses the following relationships:*

$$\forall 0 \leq t < p : pr_t := \sum_{\{i | s_i \leq t < s_i + d_i\}} r_i \leq l \quad (1)$$

$$\forall 1 \leq i \leq n : 0 \leq s_i < s_i + d_i \leq p \quad (2)$$

$$ov(t, pr_t, A_j) := \begin{cases} \max(0, \min(y_j + h_j, pr_t) - y_j) & x_j \leq t < x_j + w_j \\ 0 & otherwise \end{cases} \quad (3)$$

$$\forall 1 \leq j \leq m : a_j = \sum_{0 \leq t < p} ov(t, pr_t, A_j) \quad (4)$$

$$cost = \sum_{j=1}^m a_j c_j \quad (5)$$

For each time point t we first define the resource profile pr_t (the amount of resource consumed at time t). That profile must be below the overall resource limit l , as in the standard cumulative. The term $ov(t, pr_t, A_j)$ denotes the intersection of the profile at time t with area A_j , and the sum of all such intersections is the total resource usage a_j . The cost is computed by weighting each intersection a_j with the per-unit cost c_j of the area.

Note that our constraint is a strict generalization of the standard cumulative. Since enforcing generalized arc consistency (GAC) for `Cumulative` is NP-hard [3], the complexity of enforcing GAC over `CumulativeCost` is NP-hard as well. In the remainder of the paper we study the ability of `CumulativeCost` to reason about the cost through computation of lower bounds.

3 Decomposition With Cumulative

We present a number of models where we decompose the `CumulativeCost` into a standard `Cumulative` and some other constraint(s) which allow the computation of a lower bound for the `cost` variable. This cost component exchanges information with the `Cumulative` constraint only through domain restrictions on the s_i variables.

3.1 Element Model

The first approach is to decompose the constraint into a `Cumulative` constraint and a set of `Element` constraints. For a variable $x \in \{1, \dots, n\}$ and a variable $y \in \{v_1, \dots, v_n\}$, the element constraint `element`($x, [v_1, v_2, \dots, v_n], y$) denotes that $y = v_x$. For each task i and start time t we precompute a cost of having a task T_i starting at time t in the cheapest area overlapping the time slot t (the bottom area). This value, denoted as v_{it} , is only a lower bound, and is incapable of expressing the exact cost if

the task starts at t but in a higher (more expensive) area. A lower bound can then be expressed as

$$\text{lb} = \min \sum_{i=1}^n u_i$$

$$\forall 1 \leq i \leq n : \text{element}(s_i, [v_{i1}, v_{i2}, \dots, v_{ip}], u_i)$$

This lower bound can significantly underestimate the optimal cost since each task is assumed to be placed at its “cheapest” start time, even if the overall available volume would be exceeded.

3.2 Flow Model

We can also describe our problem as a min-cost flow model, where the flow cost provides a lower bound to the cost variable of our constraint. We need to move the flow $\sum_i d_i r_i$ from tasks to areas. Figure 2 shows the flow graph used, where the tasks T_i are linked to the areas A_j through flow variables f_{ij} which indicate how much volume of task i is contained in area j . The sum of all flows a_j through all areas must be equal to the total task volume. Only the links from A_j to the terminal T have non-zero cost c_j . The lower bound estimate is a min-cost flow.

The model can also be expressed by a set of equations and inequalities.

$$\text{lb} = \min \sum_{j=1}^m a_j c_j \quad (6)$$

$$\forall 1 \leq j \leq m : a_j = \sum_{i=1}^n f_{ij} \quad (7)$$

$$\forall 1 \leq i \leq n, \forall 1 \leq j \leq m : 0 \leq \underline{f}_{ij} \leq f_{ij} \leq \overline{f}_{ij} \quad (8)$$

$$\forall 1 \leq j \leq m : 0 \leq \underline{a}_j \leq a_j \leq \overline{a}_j \leq w_j h_j \quad (9)$$

$$\forall 1 \leq i \leq n : \sum_{j=1}^m f_{ij} = d_i r_i \quad (10)$$

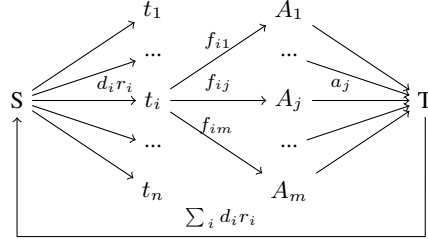
$$\forall 1 \leq i \leq n : \sum_{i=1}^n d_i r_i = \sum_{j=1}^m a_j \quad (11)$$

While the flow model avoids placing too much work in the cheapest areas, it does allow splitting tasks over several, even non-contiguous areas to reduce overall cost, i.e. it ignores resource use and non-preemptiveness of tasks.

The quality of the bound lb can be improved by computing tight upper bounds \overline{f}_{ij} . Given a task i with domain $d(s_i)$, we can compute the maximal overlap between the task and the area j as:

$$\overline{f}_{ij} = \max_{t \in d(s_i)} \max(0, (\min(x_j + w_j, t + d_i) - \max(x_j, t))) * \min(h_j, r_i) \quad (12)$$

Fig. 2. Flow Graph



3.3 LP Models

The quality of the lower bound can be further improved by adding to the flow model some linear constraints, which limit how much of some task can be placed into low-cost areas. These models are no longer flows, but general LP models. We consider two variations denoted as *LP1* and *LP2*. We obtain *LP1* by adding equations (13) to the LP formulation of the flow model (Constraints 6-11).

$$\forall 1 \leq j \leq m : \sum_{i=1}^n \sum_{k=1}^j f_{ik} = \sum_{k=1}^j a_k \leq \overline{B}_j = \sum_{i=1}^n \overline{b}_{ij} \quad (13)$$

The \overline{b}_{ij} values tell us how much of task i can be placed into the combination of cheaper areas $\{A_1, A_2, \dots, A_j\}$. Note that this is a stronger estimate, since $\overline{b}_{ij} \leq \sum_{k=1}^j \overline{f}_{ij}$. Therefore, the LP models dominate the flow model, but require a more expensive LP optimization at each step. \overline{B}_j denotes the total amount of resources (from all tasks) that is possible to place into the first j areas. We can compute the \overline{b}_{ij} values with a sweep along the time axis.

The *LP2 model* produces potentially even stronger bounds by replacing constraints (13) with more detailed constraints on individual \overline{b}_{ij} :

$$\forall 1 \leq i \leq n, \forall 1 \leq j \leq m : \sum_{k=1}^j f_{ik} \leq \overline{b}_{ij} \quad (14)$$

4 Coarse Models and Greedy Algorithms

If we want to avoid running an LP solver inside our constraint propagator, we can derive other lower bounds based on greedy algorithms. A first approximation of the cost can

be obtained with Algorithm A, which can be described by the recursive equations:

$$\text{lb} = \sum_{j=1}^m u_j c_j \quad (15)$$

$$\forall 1 \leq j \leq m : u_j = \min\left(\sum_{i=1}^n d_i r_i - \sum_{k=1}^{j-1} u_k, \sum_{i=1}^n \overline{f_{ij}}, w_j h_j\right) \quad (16)$$

The algorithm tries to fill the cheapest areas first as far as possible. For each area we compute the minimum of the remaining unallocated task area, and the maximum amount that can be placed into the area based on the $\overline{f_{ij}}$ bounds and the overall area size.

We can extend Algorithm A by also considering the $\overline{B_j}$ bounds (constraints (13)), and define Algorithm B as:

$$\text{lb} = \sum_{j=1}^m u_j c_j \quad (17)$$

$$\forall 1 \leq j \leq m : u_j = \min\left(\sum_{i=1}^n d_i r_i - \sum_{k=1}^{j-1} u_k, \sum_{i=1}^n \overline{f_{ij}}, w_j h_j, \sum_{i=1}^n \overline{b_{ij}} - \sum_{k=1}^{j-1} u_k\right) \quad (18)$$

Both algorithms A and B only compute a bound on the cost, and do not produce a complete assignment of tasks to areas. On the other hand, they only require a single iteration over the areas, provided the $\overline{f_{ij}}$ and $\overline{b_{ij}}$ bounds are known.

5 Conclusions

We have presented a number of lower bounds for an energy cost aware extension of the cumulative constraint. A formal comparison and an experimental evaluation of the proposed and some additional lower bound methods is given in [4]. This forms the basis of our current research in appropriate domain filtering methods using the best of the lower bounds.

References

1. A. Aggoun and N. Beldiceanu. Extending CHIP in order to solve complex scheduling problems. *Journal of Mathematical and Computer Modelling*, 17(7):57–73, 1993.
2. P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, Dordrecht, 2001.
3. John Hooker. *Integrated Methods for Optimization*. Springer, New York, 2007.
4. H. Simonis and T. Hadzic. A resource cost aware cumulative, 2010. Submitted for publication <http://4c.ucc.ie/~hsimonis/tidal.pdf>.
5. Petr Vilím. Max energy filtering algorithm for discrete cumulative resources. In Willem Jan van Hoeve and John N. Hooker, editors, *CPAIOR*, volume 5547 of *Lecture Notes in Computer Science*, pages 294–308. Springer, 2009.