ARTICLE IN PRESS

Discrete Applied Mathematics ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Redundant cumulative constraints to compute preemptive bounds

Philippe Baptiste [a], Nicolas Bonifas [a,b,*]

[a] *LIX, École Polytechnique, 91128 Palaiseau, France*
[b] *IBM, 9 rue de Verdun, 94250 Gentilly, France*

A R T I C L E I N F O

Article history:
Received 1 June 2015
Received in revised form 27 April 2017
Accepted 8 May 2017
Available online xxxx

Keywords:
Constraint-based scheduling
Cumulative constraint
Reformulation
Preemptive bound

A B S T R A C T

We introduce a reformulation of the cumulative resource in scheduling. This reformulation yields a lower bound on the makespan which is at least as good as that of a preemptive schedule on the original resource. Its computation relies on a linear program whose size depends on the resource capacity but not on the number of tasks, which enables to precompute the reformulations. It provides a significant improvement for all algorithms which rely on energy-based reasonings for propagating cumulative constraints, such as edge-finding and energy reasoning techniques. We improve the lower bounds of some RCPSP instances from the PSPLIB.

1. Introduction

We focus on the discrete cumulative resource, an abstraction for the assignment of a fixed quantity of a resource to a task during its execution period. At any point in time, the total resource usage by the tasks being run must not exceed the total resource capacity. This abstraction is heavily used in constraint-based scheduling to model the allocation of a limited resource, such as manpower, budget, parallel machines, and reservoirs. In this context, we talk of the cumulative *constraint*.

Definition 1. Given a discrete cumulative resource of capacity C on a set of n tasks with respective length p_i and demand c_i on the resource, we say that the resource can be satisfied if and only if for each task there exists a start time t_i such that

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

In this case, $t_1, \ldots, t_n$ is called a *valid schedule*.

An example of a common application of this constraint is in the famous Resource Constrained Project Scheduling Problem: this problem can be seen as minimizing the makespan of a set of non-preemptive tasks of given lengths, under a cumulative constraint such as we just introduced, in addition to precedence constraints between tasks.

The cumulative constraint was introduced as a constraint in the Constraint Programming framework in [1].

Note that this problem is related but clearly different from the two-dimensional packing problems [11], such as strip-packing [9]. One can obtain excellent relaxations of two-dimensional packing problems with cumulative resources though, and all the techniques in this article apply to two-dimensional packing as well.

* Corresponding author at: LIX, École Polytechnique, 91128 Palaiseau, France.
E-mail addresses: philippe.baptiste@cnrs-dir.fr (P. Baptiste), bonifas@lix.polytechnique.fr (N. Bonifas).

http://dx.doi.org/10.1016/j.dam.2017.05.001
0166-218X/© 2017 Elsevier B.V. All rights reserved.

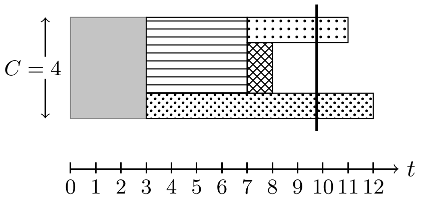

Fig. 1. Example instance. Energy bound: 9.75.

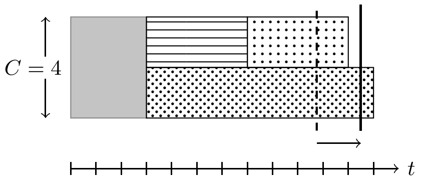

Fig. 2. Reformulation of the previous instance. Energy bound: 11.5.

Many successful approaches have been suggested recently to deal with such constraints, including a reformulation as a SAT problem [17,18], Lagrangian relaxation [15], a variant of Jackson's preemptive schedule [7], the generation of redundant cumulative constraints thanks to so-called *dual feasible functions* [6], configuration LPs to get strong lower-bounds [3,5], energy reasoning [8] and improved edge-finding algorithms [4,14,16,19].

The last two approaches rely on the notion of *energy* of a set of tasks over a cumulative resource:

Definition 2. Given a discrete cumulative resource of capacity C and a set of n tasks with respective lengths p_i and demands c_i, the *energy bound* of the tasks on the resource is $E = \sum_{i=1}^{n} \frac{p_i c_i}{C}$.

Notice that the energy gives a lower bound for the total time needed to run all the tasks.

For example (see Fig. 1), on a resource of capacity $C = 4$, if we have $n = 5$ tasks of lengths $p = (3, 4, 9, 1, 4)$ and respective demands $c = (4, 3, 1, 2, 1)$, then the energy lower bound on the makespan is $\frac{3\times4+4\times3+9\times1+1\times2+4\times1}{4} = 9.75$ (the actual minimum makespan for this example is 12).

We use the cumulative resource in the context of *constraint programming*. In this paradigm, *variables* 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 a branch and bound search of 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 enforce necessary conditions on the constraints after each domain reduction. This further reduces the domains of the variables. The algorithms which enforce these necessary conditions are called *propagations*. Efficient propagations are the key to constraint programming performance, which is critical since we aim for the exact resolution of problems difficult both in theory and in practice.

In the context of constraint-based scheduling [4], we maintain for each task an earliest starting time and a latest ending time, and adjust those using filtering algorithms. These filtering algorithms check for necessary conditions for the constraints to be satisfied and strengthen the earliest starting time and a latest ending time, thus cutting branches of a branch and bound tree, in a similar fashion to dual bounds in integer programming. Examples of such filtering algorithms for the cumulative constraint are the *energy-based reasoning*, introduced in [8], and the *cumulative edge-finding* technique, first introduced in [16]. If forcing a task to start at its earliest starting time would result in an excess of energy over an interval, we can increase the earliest starting time for the task until no such constraint is violated. There is a symmetric rule to adjust the latest ending time.

The objective of this article is not to give a new filtering algorithm, but to improve all the reasonings based on the notion of energy, which will automatically improve the existing filtering algorithms which rely on this notion, notably timetabling and edge-finding. See for example Fig. 2, which gives a possible reformulation for the instance of Fig. 1: the energy bound is increased from 9.75 to 11.5, and this will help the filtering algorithms adjust the time bounds for this instance.

The reformulation of cumulative constraints that we introduce in this paper is such that all valid schedules remain valid for the reformulated constraint, but we now have a guarantee that the energy lower bound matches the makespan of the preemptive relaxation. The *preemptive relaxation* of a cumulative resource problem is a solution which satisfies the cumulative constraints but enables tasks to be interrupted and restarted later (compared to a non-preemptive schedule where a task of length p must run uninterrupted from its starting time t until $t+p$). These reformulations are relatively cheap to compute (after a precomputation which does not depend on the instance) and they provide a significant improvement for all algorithms which rely on energy reasonings. In our experiments, we used them within an edge-finding algorithm.