Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

ALEXANDER TESCH

Improving Energetic Propagations for Cumulative Scheduling

ZIB Report 18-29 (June 2018)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Improving Energetic Propagations for Cumulative Scheduling

Alexander Tesch

Zuse Institute Berlin (ZIB)
Takustraße 7, 14195 Berlin, Germany
tesch@zib.de

Abstract. We consider the Cumulative Scheduling Problem (CuSP) in which a set of n jobs must be scheduled according to release dates, due dates and cumulative resource constraints. In constraint programming, the CuSP is modeled as the cumulative constraint. Among the most common propagation algorithms for the CuSP there is energetic reasoning [1] with a complexity of $O(n^3)$ and edge-finding [21] with $O(kn \log n)$ where $k \leq n$ is the number of different resource demands. We consider the complete versions of the propagators that perform all deductions in one call of the algorithm. In this paper, we introduce the energetic edge-finding rule that is a generalization of both energetic reasoning and edge-finding. Our main result is a complete energetic edge-finding algorithm with a complexity of $O(n^2 \log n)$ which improves upon the complexity of energetic reasoning. Moreover, we show that a relaxation of energetic edge-finding with a complexity of $O(n^2)$ subsumes edge-finding while performing stronger propagations from energetic reasoning. A further result shows that energetic edge-finding reaches its fixpoint in strongly polynomial time. Our main insight is that energetic schedules can be interpreted as a single machine scheduling problem from which we deduce a monotonicity property that is exploited in the algorithms. Hence, our algorithms improve upon the strength and the complexity of energetic reasoning and edge-finding whose complexity status seemed widely untouchable for the last decades.

Note: We fixed a mistake in the definition of $\tilde{\omega}(t)$ in the original version.

1 Introduction

In the CuSP, we are given a set of non-preemptive jobs $i \in \{1, \dots, n\}$ with processing times $p_i > 0$, resource demands $c_i > 0$, release dates r_i, due dates d_i and a resource capacity C. We assume that $0 \leq r_i \leq d_i - p_i$ for all jobs $i = 1, \dots, n$ without loss of generality. In the CuSP, we compute start times S_i for every job $i = 1, \dots, n$ such that every job is scheduled within its time interval and the resource capacity is never exceeded. More formally, it can be stated as

$$\sum_{\substack{i=1,\dots,n: \\ S_i \leq t < S_i + p_i}} c_i \leq C \qquad \forall t \in [0, T] \tag{1}$$

$$r_i \leq S_i \leq d_i - p_i \qquad \forall i = 1, \dots, n. \tag{2}$$

where T is an upper bound on the time horizon. In general, the CuSP can be seen as a decision variant of the scheduling problem $P|r_j, size_j|L_{max}$ in which a set of n non-preemptive multiprocessor jobs is scheduled onto C machines where every job j allocates $c_j = size_j$ machines at the same time. The objective is to minimize the maximum lateness $L_{max} = \max_{j=1}^{n} \max\{0, S_j + p_j - d_j\}$ where $L_{max} = 0$ if and only if the CuSP is feasible. The CuSP is NP-complete since already the scheduling problem $1|r_j|L_{max}$ is strongly NP-hard [11].

In practice, the CuSP often occurs as a dedicated subproblem in more complex scheduling or optimization problems [9,17]. In order to reduce the size of the search tree, the idea is to derive stronger bounds on the release and due dates of the jobs without violating the feasibility of the problem in general. For the CuSP, many propagation algorithms have been suggested in the literature where one can observe a general trade-off between their propagation power and running time.

1.1 Previous Work

Many different propagators exist for the CuSP in the literature as well as relaxations and hybrids of them. In this paper, we focus on *complete* propagation algorithms that perform *all* reductions according to their respective rule. The *energetic reasoning* propagation rule has been first introduced in [6] where the methods were refined in [1] to give a complete $O(n^3)$ algorithm. Since energetic reasoning counts to the most powerful propagation rules for the CuSP many efforts have been made to improve the general $O(n^3)$ complexity, for example by reducing the number of considered intervals [2,5]. In [3] an algorithm is presented that detects at least one energetic reasoning propagation in $O(n^2 \log n)$ that was improved in [20] to an algorithm that detects at least one possible energetic reasoning propagation for every job in $O(n^2 \log n)$. Both algorithms, however, remain incomplete and do not compute all the maximum propagations in one run. However, they converge to the same fixpoint of energetic reasoning.

Another common propagation rule is *edge-finding* where a first complete algorithm was introduced in [14] with complexity $O(kn^2)$ and also a stronger variant called *extended edge-finding* with the same complexity. This complexity was improved in [21] to $O(kn \log n)$ and in [16] they use a modification of similar techniques to integrate propagations from extended edge-finding and timetabling also in $O(kn \log n)$. In general, energetic reasoning yields stronger propagations than (extended) edge-finding. However, its high complexity of $O(n^3)$ prevents it from being used in practice where edge-finding or hybrids such as *time-table edge-finding* [19,22] are preferred. Thus, a major open question is whether the $O(n^3)$ complexity of energetic reasoning can be improved to lower the discrepancy between the different propagation schemes.

Further propagation algorithms for the CuSP are *timetabling* [7,12] and *not-first/not-last* [18] where the latter is incomparable to energetic reasoning.

1.2 Results

In this paper, we introduce the novel *energetic edge-finding* propagation rule that generalizes and combines ideas from energetic reasoning and edge-finding. We first show that energetic edge-finding reaches its fixpoint in strongly polynomial time. But our main contribution is a complete energetic edge-finding algorithm that runs in $O(n^2 \log n)$ and therefore improves upon the previous $O(n^3)$ complexity of energetic reasoning. Moreover, by excluding some difficult time intervals, we obtain a relaxed version of energetic edge-finding that runs in $O(n^2)$ and subsumes the edge-finding rule while simultaneously performing stronger propagations from energetic reasoning. This also improves the general $O(kn \log n)$ of edge-finding [21]. A major component in our algorithms is the interpretation of energetic schedules as a single machine scheduling problem with release dates [8]. Energy profiles of single machine problems yield a monotonicity property that we highly exploit in our algorithms. We implemented and tested the algorithms on instances of the well-established PSPLIB [13]. We believe that the given approaches yields new insights for the development of more efficient propagation algorithms for the CuSP.

2 Energetic Edge-Finding

In this section we formally introduce the energetic edge-finding rule. It can be seen as a generalization and combination of both energetic reasoning and edge-finding. It takes advantage of the propagation strength of energetic reasoning and the stable fixpoint behavior of edge-finding by integrating additional propagations from subintervals. Our version of energetic edge-finding is not to be mixed-up with the one given in [10] who consider a similar but incomplete rule.

We begin with some notation. Let the minimum left-right-shift duration of a job i in a time interval $[t_1, t_2]$ be defined as

$$p_i(t_1, t_2) = \max\{0, \min\{p_i, t_2 - t_1, r_i + p_i - t_1, t_2 - d_i + p_i\}\} \tag{3}$$

where only the left-shift duration of job i in $[t_1, t_2]$ is given by

$$p_i^l(t_1, t_2) = \max\{0, \min\{t_2 - t_1, p_i, r_i + p_i - t_1, t_2 - r_i\}\}. \tag{4}$$

Moreover, let $e(t_1, t_2) = \sum_{i=1}^{n} c_i \cdot p_i(t_1, t_2)$ denote the energy in the interval $[t_1, t_2]$ where the *energy overload* in $[t_1, t_2]$ is given by

$$\omega(t_1, t_2) = e(t_1, t_2) - C \cdot (t_2 - t_1) \tag{5}$$

that equals the slack between the minimum consumed energy and available energy in $[t_1, t_2]$. If there exists an interval $[t_1, t_2]$ with $\omega(t_1, t_2) > 0$ then the CuSP is already infeasible. Thus we assume that $\omega(t_1, t_2) \leq 0$ for all intervals $[t_1, t_2]$. Analogously, for a job i and time interval $[t_1, t_2]$ let

$$\omega_i^l(t_1, t_2) = \omega(t_1, t_2) + c_i \cdot \left(p_i^l(t_1, t_2) - p_i(t_1, t_2)\right). \tag{6}$$

be the energy overload under the condition that job i is left-shifted. If there exists a time interval $[t_1, t_2]$ with $\omega_i^l(t_1, t_2) > 0$ then the release date of job i is invalid and can be increased to $r_i = t_2 - p_i(t_1, t_2) + \frac{\omega(t_1,t_2)}{c_i}$ in order to pull the exceeding energy out of the interval $[t_1, t_2]$. One could also consider right-shifts but they are symmetric to left-shifts at time $t = 0$, so we will stick to left-shifts if not mentioned otherwise. This propagation concept can be formalized by the following rules.

Energetic Reasoning [1]. A complete energetic reasoning algorithm computes for every job $i = 1, \ldots, n$ the release dates

$$r_i^* = \max_{\substack{(t_1,t_2)\in\mathcal{T}:\\ \omega_i^l(t_1,t_2)>0}} \left(t_2 - p_i(t_1, t_2) + \frac{\omega(t_1, t_2)}{c_i} \right) \tag{7}$$

where $\mathcal{T}$ is a set of relevant time intervals that is specified later.

In [1] it is shown that $|\mathcal{T}| \in O(n^2)$ and they present a complete algorithm with running time $O(n^3)$ to perform all propagations.

In this paper, we consider an even stronger variant of energetic reasoning that is highly inspired by the edge-finding rule [21] that takes into account additional propagations from subintervals.

Energetic Edge-Finding. A complete energetic edge-finding algorithm computes for every job $i = 1, \ldots, n$ the release dates

$$r_i^* = \max_{\substack{(t_1,t_2)\in\mathcal{T}:\\ \omega_i^l(t_1,t_2)>0}} \max_{\substack{(t'_1,t'_2)\in\mathcal{T}:\\ [t'_1,t'_2]\subseteq[t_1,t_2],\\ \omega(t'_1,t'_2)+c_i\cdot(t'_2-t'_1-p_i(t'_1,t'_2))>0}} \left(t'_2 - p_i(t'_1, t'_2) + \frac{\omega(t'_1, t'_2)}{c_i} \right) \tag{8}$$

where $\mathcal{T}$ is the same set of time intervals as for energetic reasoning. The energetic edge-finding rule can be explained as follows. Once a left-shift overload with $\omega_i^l(t_1, t_2) > 0$ for a job i and interval $(t_1, t_2) \in \mathcal{T}$ is detected, it includes propagations of a subinterval $[t'_1, t'_2]$ if the energy of all jobs *different* from i in $[t'_1, t'_2]$ prevent an earlier processing of i. In this case, its release date r_i can be updated accordingly for $[t'_1, t'_2]$. For the set of relevant time intervals $\mathcal{T}$ let

$$\begin{aligned} T_1 &= \{r_i, d_i - p_i : i = 1, \ldots, n\} \\ T_2 &= \{d_i, r_i + p_i : i = 1, \ldots, n\} \\ T_3(t) &= \{r_i + d_i - t : i = 1 \ldots, n\} \end{aligned}$$

and from these sets we define

$$\begin{aligned} T_{12} &= \{(t_1, t_2) : t_1 < t_2, t_1 \in T_1, t_2 \in T_2\} \\ T_{13} &= \{(t_1, t_2) : t_1 < t_2, t_1 \in T_1, t_2 \in T_3(t_1)\} \\ T_{23} &= \{(t_1, t_2) : t_1 < t_2, t_2 \in T_2, t_1 \in T_3(t_2)\} \end{aligned}$$

where $\mathcal{T} = T_{12} \cup T_{13} \cup T_{23}$. Note that there exist tighter characterizations for the set of relevant time intervals for energetic reasoning [5,20] but since they are more complex we will stick to this definition. Moreover, in the first sections we only consider propagations for intervals in T_{12} and T_{23}. The integration of intervals in T_{13} needs special treatment and is examined separately in Section 3.4.

Since propagations from energetic edge-finding are not idempotent in general, we apply the rule until a fixpoint is reached. Because their overload conditions are equivalent, energetic edge-finding converges to the same fixpoint as energetic reasoning. However, we show that the additional integration of subintervals in energetic edge-finding leads to a strongly polynomial fixpoint convergence. To the best of our knowledge, the fixpoint complexity for only energetic reasoning (as stated) is unknown. The paper [14] claims a non-strongly polynomial fixpoint complexity for energetic reasoning but they use a weaker update function that can lead to slower convergence.

Theorem 1. *A complete energetic edge-finding algorithm reaches its fixpoint after at most $O(n^2)$ iterations.*

Proof. Let $T_2 = \{t_2^1, \ldots, t_2^N\}$ with $t_2^b < t_2^{b+1}$ for all $b = 1, \ldots, N-1$. We show that after at most $b \cdot n$ iterations no further propagation can be detected in any interval $[t_1, t_2^b]$ with $t_1 < t_2^b$.

Consider an arbitrary fixpoint iteration q of the complete energetic edge-finding algorithm and assume by induction hypothesis that no propagation can be found in any interval $[t_1, t_2^{b'}]$ for all $t_2^{b'} < t_2^b$. If job i is propagated in iteration q due to an interval $[t_1, t_2^b]$ with $\omega_i^l(t_1, t_2^b) > 0$ then since $\omega_i^l(t_1, t_2^b) = \omega(t_1, t_2^b) + c_i \cdot (p_i^l(t_1, t_2^b) - p_i(t_1, t_2^b))$ we have that $\omega(t_1, t_2^b)$ or $p_i^l(t_1, t_2^b) - p_i(t_1, t_2^b)$ increased from iteration $q-1$ to q.

If $p_i^l(t_1, t_2^b) - p_i(t_1, t_2^b)$ has increased then job i must have been propagated in iteration $q-1$. If $\omega(t_1, t_2^b)$ has not changed for all $t_1 < t_2^b$ from iteration $q-1$ to q then energetic edge-finding would have found the propagation already in iteration $q-1$ giving a contradiction[1].

Hence, $\omega(t_1, t_2^b)$ has changed from iteration $q-1$ to q. This happens if and only if a job $j \neq i$ with $d_j - p_j < t_2^b < d_j$ is propagated to end after time t_2^b. Then job j has a fixed part[2] in the interval $[d_j - p_j, t_2^b]$ to the beginning of iteration q. Moreover, the value $p_j(t_1, t_2^b)$ cannot be increased further by propagations from other intervals $[t_1', t_2^b]$, so job j cannot further increase $\omega(t_1, t_2^b)$. Because at most n jobs can increase $\omega(t_1, t_2^b)$ this way, we have that $q \leq b \cdot n$. Since $N \in O(n)$ the fixpoint is reached after $O(n^2)$ iterations. □

It is an open question whether this fixpoint complexity is tight and if there exist examples where energetic reasoning needs $\Omega(n)$ more iterations to reach the fixpoint.

[1] Unlike energetic reasoning that may need additional iterations to reach the maximum propagation for t_2^b.

[2] Unlike standard edge-finding that does not consider partial overlaps, which gives a short proof of its $\mathcal{O}(n)$ fixpoint complexity [14].

3 Algorithm

Our main algorithm iterates over all $t_2 \in T_2$ in an outer loop. We focus on the resulting subproblem in the time horizon $[0, t_2]$ for times $t \in T_1 \cup T_3(d)$. Time intervals $(t_1, t_2) \in T_{23}$ are integrated separately in Section 3.4. Throughout the rest of the paper we will consider a fixed t_2 value and set the due date $d = t_2$ as a *global constant*. For abbreviation, we therefore omit $d = t_2$ as function argument and rewrite $p_i(t) = p_i(t, d)$, $p_i^l(t) = p_i^l(t, d)$, $e(t) = e(t, d)$, $\omega(t) = \omega(t, d)$, $\omega_i^l(t) = \omega_i^l(t, d)$ and $T = \{t : t < d, t \in T_1 \cup T_3(d)\}$.

Our algorithm is divided into three phases: *decomposition phase*, *detection phase* and *update phase*.

3.1 Decomposition Phase

In the decomposition phase, we will decompose the available energy $e(0)$ in the interval $[0, d]$ into energy blocks $B_1, \ldots, B_m$ in order to get a stronger representation for the overload function $\omega(t)$.

Let the *energy envelope* at time $t < d$ be defined as

$$E(t) = t + \frac{e(t)}{C} \tag{9}$$

that is a lower bound on the maximum completion time of jobs i with $p_i(t) > 0$. Then we have $\omega(t) = C \cdot (E(t) - d)$.

From this definition, we create energy blocks as follows. Let $T = \{t_1, \ldots, t_H\}$ with $t_h < t_{h+1}$ for all $h = 1, \ldots, H-1$. Starting with $h = 1$, we compute the next greater $h' > h$ such that $E(t_h) < E(t_{h'})$. If $e(t_h) > e(t_{h'})$ we create a new *canonical block* B with release date $r(B) = t_h$, processing time $p(B) = \frac{e(t_h) - e(t_{h'})}{C}$ and resource demand C. Then we set $h = h'$ and repeat the procedure. If no h' can be found we finally set $p(B) = \frac{e(t_h)}{C}$ and stop.

The set of all canonical blocks $B_1, \ldots, B_m$ is also denoted as the *canonical decomposition* [8]. In the following, let $T_B = \{r(B_1), \ldots, r(B_m)\}$ be the set of release dates of the canonical blocks. Since every canonical block B_l has resource demand C, the canonical decomposition can be interpreted as a scaled single machine schedule, see Figure 1. Moreover, the energy values $e(t_h)$ can be computed and maintained by the algorithm of Baptiste et al. [1] in $O(n)$, so the canonical decomposition can be computed in $O(n)$ time.

A further important observation is that we can restrict to non-dominated energy envelopes. That is, if $E(t') > E(t)$ for $t' < t$ then we can consider $E(t')$ instead of $E(t)$ for time t. Hence, we can replace the function $E(t)$ by

$$\tilde{E}(t) = \max_{t' \leq t} E(t') \tag{10}$$

where $\tilde{\omega}(t) = C \cdot (\tilde{E}(t) - d)$ is a stronger lower bound on the overload in $[t, d]$. Consequently, we can replace $\omega(t)$ by $\tilde{\omega}(t)$ in our energetic edge-finding rule. In particular, we have $\omega(t) = \tilde{\omega}(t)$ for all $t \in T_B$ and since $\tilde{E}(t)$ is monotonically

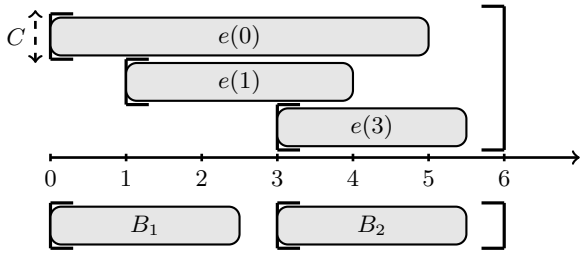

Fig. 1. Canonical decomposition: the three energy blocks (above) show the energy envelopes $E(t)$ for $t \in \{0, 1, 3\}$ with due date $d = 6$ for some CuSP instance. We obtain the canonical decomposition B_1, B_2 (below) that shows more precisely where the energy is induced. If there exists an energy overload for any interval $[t, d]$ then it is always generated by the last canonical block B_m.

non-decreasing in t we also have that $\tilde{\omega}(t)$ is monotonically non-decreasing in t. We will highly exploit this fact in our algorithms.

We can further use the canonical decomposition to efficiently compute $\tilde{\omega}(t)$. For given time t, let B_l be the canonical block with smallest index l such that $t \leq r(B_l) + p(B_l)$ then we have

$$\tilde{\omega}(t) = C \cdot \left(\min\{t, r(B_l)\} + \sum_{l' \geq l} p(B_{l'}) - d \right) \tag{11}$$

where this definition depends only on the canonical decomposition. We will use the canonical decomposition for the next phases that depend only on $T_B \subseteq T$.

3.2 Detection Phase

In the detection phase, we check for every job $i = 1, \ldots, n$ if there exists a time $t \in T$ such that $\omega_i^l(t) > 0$. In this case, job i must end strictly after time d and this relation is stored in an n-dimensional array *end* by setting $end[i] = d$, similar to [21]. For given due date d, we compute all such relationships as follows.

The condition $\omega_i^l(t) > 0$ is equivalent to

$$c_i \cdot \left(p_i^l(t) - p_i(t) \right) > -\omega(t) \tag{12}$$

where both left- and right-hand side are non-negative since $\omega(t) \leq 0$ for all $t \leq d$. We replace $\omega(t)$ by the stronger overload function $\tilde{\omega}(t)$ which yields

$$c_i \cdot \left(p_i^l(t) - p_i(t) \right) > -\tilde{\omega}(t) \tag{13}$$

and this is still a valid condition since $\omega(t) \leq \tilde{\omega}(t)$ and thus not less intervals are checked for energetic edge-finding (8).

Moreover, both functions on the left- and right-hand side of (13) are piecewise linear in t. Let $p_i^s = p_i^l(0) - p_i(0)$ then the function on the left-hand side

decomposes into the segments:

$$c_i \cdot (p_i^l(t) - p_i(t)) = \begin{cases} c_i \cdot p_i^s, & t \leq r_i \\ c_i \cdot (p_i^s - t + r_i), & r_i \leq t < r_i + p_i^s \\ 0, & else \end{cases} \tag{14}$$

for all jobs $i = 1, \ldots, n$ where we only consider jobs i with $p_i^s > 0$ or jobs with positive left-shift slack respectively.

We have to compute for every $i = 1, \ldots, n$ the time $t \in T$ for which inequality (13) is maximally satisfied. For this, we compute the upper envelope of the line segments (14) of all jobs $i = 1, \ldots, n$ by using a sweep line approach [4]. The sweep line data structure comprises a binary status tree W and an event heap H. The status tree W changes dynamically with t and stores the line segments as leaves while maintaining the line segment of maximum value $c_i \cdot (p_i^l(t) - p_i(t))$ for the current t in the root node using a bottom-up approach. The event heap H stores time events at which the order of two line segments in W is changing or the maximum line segment of W must be retrieved from the root node. New events are added dynamically to H. Since there are $O(n)$ line segments, one line segment can be added or deleted in $O(\log n)$ and sweeping over all $t \in T$ takes $O(n \log n)$ since for every node in W at most one event is added dynamically to H. In particular, we can restrict to $T_B \subseteq T$ since $\tilde{\omega}(t)$ is locally maximal there.

Thus, in our algorithm we sweep over all $t \in T_B$ in decreasing order and retrieve the maximum function value $v_i = c_i \cdot (p_i^l(t) - p_i(t))$ of the currently dominating job i from the root node of W. While the currently dominating job i satisfies $v_i > \tilde{\omega}(t)$ we set $end[i] = d$ and delete all line segments that belong to job i (the dominating job may change) and store the update value $r_i^d = d - p_i(t) + \frac{\tilde{\omega}(t)}{c_i}$. Hence, for the given due date d the detection phase takes $O(n \log n)$ time, compare Algorithm 1.

An improvement can be made by omitting the constant line segment for $t \leq r_i$ in (14) since $\tilde{\omega}(t)$ is monotonically non-decreasing and therefore inequality (13) is maximally satisfied at $t = r_i$ but this point is also covered by the second line segment. In this case, we need to add an event at $t = r_i$ to H to retrieve possible overloads from W.

A special case occurs when we have $c = c_i$ for all $i = 1, \ldots, n$ respectively. In this case, all line segments have the same slope so we can use a simple queue to process the line segments from right to left where the first line segment that is added remains the dominating one until it is deleted. This allows a processing in $O(n)$ and will yield a complete $O(n^2)$ energetic edge-finding algorithm.

Proposition 1. *Let $t^* \in T_B$ be the time where we set $end[i] = d$ for a job i. Then we have $t^* < r_i + p_i^s$ and t^* also maximizes*

$$\max_{\substack{t \in T: \\ \tilde{\omega}(t) + c_i \cdot (p_i^l(t) - p_i(t)) > 0}} \left(d - p_i(t) + \frac{\tilde{\omega}(t)}{c_i} \right). \tag{15}$$

Algorithm 1: detection phase

Input: due date d, canonical decomposition $B_1, \ldots, B_m$
Output: $end[i]$ and r_i^d for all $i = 1, \ldots, n$,

initialize sweep line tree W for all jobs i with $r_i < d < d_i$
$E \leftarrow 0$
forall $l = m, \ldots, 1$ **do**
  $E \leftarrow E + C \cdot p(B_l)$
  $t \leftarrow r(B_l)$
  $sweep\_to(W, t)$
  while $i \leftarrow W.root.job$ *and* $E + c_i \cdot (p_i^l(t) - p_i(t)) > C \cdot (d - t)$ **do**
    $end[i] \leftarrow \max\{end[i], d\}$
    $r_i^d \leftarrow d - p_i(t) + \frac{1}{c_i} \cdot (E - C \cdot (d - t))$
    delete line segments of i from W
return $end[i]$ and r_i^d for all $i = 1, \ldots, n$

Proof. As mentioned, we can replace T by T_B in the given formula. Whenever the algorithm sets $end[i] = d$ for $t^* \in T_B$, we have $\tilde{\omega}(t^*) + c_i \cdot (p_i^l(t^*) - p_i(t^*)) > 0$ and therefore $p_i^l(t^*) - p_i(t^*) > 0$ which holds only if $t^* < r_i + p_i^s$.

For all $t < r_i + p_i^s$ we have $p_i(t) = p_i(0)$ is constant. Thus, the overload function turns into $\tilde{\omega}(t) + c_i \cdot (p_i^l(t) - p_i(0)) > 0$ and the update function into $d - p_i(0) + \frac{\tilde{\omega}(t)}{c_i}$ that is equivalent to maximizing $\tilde{\omega}(t)$. Since both overload and update function are monotonically non-decreasing in t, the maximum t that satisfies $\tilde{\omega}(t) + c_i \cdot (p_i^l(t) - p_i(0)) > 0$ is optimal for (15). Since the algorithm iterates over all $t \in T_B$ in decreasing order, we obtain the optimal time $t = t^*$. □

Adding the $O(n)$ iterations of every due date d from the outer loop and adding the intervals in T_{23} treated later yields the following result.

Corollary 1. *Computing the detection phase for every due date $d \in T_2$ yields a complete energetic reasoning algorithm with running time $O(n^2 \log n)$.*

In Practice. The sweep line algorithm improves the theoretical performance of energetic reasoning. The necessity of computing the upper envelope is due to the fact that there might exist instances in which, say $O(n)$ many, line segments are active at the same time t. In this case, we always need to keep track of the maximum line segment that may change for small deviations of t such that the upper envelope needs to be explored completely. In practice however, the line segments of the jobs are mostly disjoint and generally only few are active at the same time. Moreover, upper envelope computations involve complex data structures that need to be build in every iteration that generates overhead for instances where n is small.

Thus, we decided to implement the following more output-sensitive but still complete version: we sweep from right to left but whenever a line segment becomes active we add it into a list L. If it becomes inactive we delete it from L,

both can be done in $O(1)$ by storing job pointers. Then for every $t \in T_B$ we iterate all elements in L to detect the line segment of maximum value. As mentioned, the size of L is practically small (mostly zero). This leads to a complete complexity of $O(n + m \cdot h)$ for the detection phase where h is the maximum number of simultaneously active line segments. Multiplying the $O(n)$ iterations of the outer loop, this is $O(n^3)$ in general but much faster on practical instances.

In order to guarantee a stable fixpoint behavior we additionally include possible propagations from all subintervals. The next section describes how to compute them efficiently in an additional phase.

3.3 Update Phase

For a given due date d, the update phase computes for every job $i = 1, \ldots, n$ with $p_i^s > 0$ and $d \leq end[i]$ stronger release dates by including propagations from subintervals. Again, let $p_i^s = p_i^l(0) - p_i(0)$. By Proposition 1, the maximum propagation for all $t \in T_B$ with $t < r_i + p_i^s$ is already found in the detection phase. Thus, assume that $t \geq r_i + p_i^s$ which leads to the problem of computing

$$r_i^d = \max_{\substack{t \in T: \\ t \geq r_i + p_i^s \\ \omega(t) + c_i \cdot (d - t - p_i(t)) > 0}} \left(d - p_i(t) + \frac{\omega(t)}{c_i} \right). \tag{16}$$

Again, we replace $\omega(t)$ by its stronger version $\tilde{\omega}(t)$ and set

$$r_i^d = \max_{\substack{t \in T_B: \\ t \geq r_i + p_i^s \\ \tilde{\omega}(t) + c_i \cdot (d - t - p_i(t)) > 0}} \left(d - p_i(t) + \frac{\tilde{\omega}(t)}{c_i} \right) \tag{17}$$

where we can restrict to $t \in T_B$ by the same argument as for the detection phase. We want to simplify formula (17). Consider the parametrization

$$r^d(c) = \max_{\substack{t \in T_B: \\ \tilde{\omega}(t) + c \cdot (d - t) > 0}} \left(d + \frac{\tilde{\omega}(t)}{c} \right) \tag{18}$$

for variable resource demands c. We use the following lemma.

Lemma 1. *Given a job $i \in \{1, \ldots, n\}$ then $t \in T_B$ with $t \geq r_i + p_i^s$ is optimal for (17) if and only if t is optimal for (18) with $r_i^d = r^d(c_i) - p_i(t) > t$.*

Proof. Assume first that $t \in T_B$ with $t \geq r_i + p_i^s$ is optimal for (17). Then $r_i^d + p_i(t) = d + \frac{\tilde{\omega}(t)}{c_i}$ and since $r_i^d > t$ we also have that $r_i^d + p_i(t) > t$ that yields $\tilde{\omega}(t) + c_i \cdot (d - t) > 0$, hence the point $t \in T_B$ is valid for (18) that implies $r_i^d + p_i(t) \leq r^d(c_i)$.

Now assume that $t' \in T_B$ is optimal for $r^d(c_i)$ with $r^d(c_i) > r_i^d + p_i(t)$ where $t' > t$ by monotonicity of $\tilde{\omega}(t)$. Since $t \geq r_i + p_i^s$ it holds $p_i(t) - p_i(t') =$

Algorithm 2: update phase

Input: due date d, canonical decomposition $B_1, \dots, B_m$, resource demands $\{c_1, \dots, c_k\}$
Output: $t^d(c)$ and $r^d(c)$ for all $c \in \{c_1, \dots, c_k\}$

```
h ← k, l ← m, E ← 0
while h ≥ 1 and l ≥ 1 do
    t ← r(B_l)
    if E − (C − c_h) · (d − t) > 0 then
        r^d(c_h) ← d + (1/c_h) · (E − C · (d − t))
        t^d(c_h) ← t
        h ← h − 1
    else
        l ← l − 1
        E ← E + p(B_l) · C
return t^d(c) and r^d(c) for all c ∈ {c_1, ..., c_k}
```

$\max\{0, t' - t\}$. Thus, if $p_i(t') = p_i(t) - t + t'$ we get $\tilde{\omega}(t') + c_i \cdot (d - t' - p_i(t')) \geq \omega(t) + c_i \cdot (d - t - p_i(t)) > 0$ by feasibility of t for (17). Otherwise, if $p_i(t') = p_i(t)$ we also have $\tilde{\omega}(t') + c_i \cdot (d - t' - p_i(t')) \geq \tilde{\omega}(t) + c_i \cdot (d - t - p_i(t)) > 0$. Hence, the solution at $t' \in T_B$ is also feasible for (17) and its objective value is equal to $r^d(c_i) - p_i(t') > r_i^d$ which contradicts the assumption that r_i^d is optimal for (17). It follows that $r^d(c_i) \leq r_i^d + p_i(t)$ and therefore $r^d(c_i) = r_i^d + p_i(t)$ which shows the statement. □

Lemma 1 allows us to work with the parametrized version (18): if we have computed $r^d(c_i)$ with optimal time $t \in T_B$ we only have to verify that $r^d(c_i) - p_i(t) > t$ and set $r_i^d = r^d(c_i) - p_i(t)$, otherwise there exists no better update.

In the following we show how to compute $r^d(c)$ for all $c \in \{c_1, \dots, c_n\}$ with $k = |\{c_1, \dots, c_n\}|$ efficiently. Substituting $g(c,t) = d + \frac{\tilde{\omega}(t)}{c}$ yields

$$r^d(c) = \max_{\substack{t \in T_B: \\ g(c,t) > t}} g(c,t) \tag{19}$$

that now has a quite simple form. For fixed c, the function $g(c,t)$ is piecewise linear and non-decreasing in t. Thus, for given resource demand $c \in \{c_1, \dots, c_n\}$ the largest value $t \in T_B$ with $g(c,t) > t$ is optimal for $r^d(c)$. However, $g(c,t)$ is also non-decreasing in c since $\tilde{\omega}(t) \leq 0$. Hence, we iterate over all $t \in T_B$ in decreasing order as long as $g(c,t) \leq t$. Starting with the last considered resource demand c, we set $r^d(c) = g(c,t)$ as long as $g(c,t) > t$ for all c in decreasing order. We repeat the same search starting with the next smaller t until either $r^d(c)$ is determined for the smallest c value or the minimum of T_B is reached. This procedure takes $O(k+m)$ to compute all $r^d(c)$, see also Algorithm 2.

Since the full algorithm iterates over all due dates $d \in T_2$ in an outer loop and by including the detection phase every inner iteration takes $O(n \log n + k + m)$ we

Algorithm 3: energetic edge-finding

Input: CuSP instance
Output: propagated release dates r_i^* from energetic edge-finding

$end[i] \leftarrow -\infty$ for all $i = 1, \ldots, n$
forall $d \in T_2$ *in decreasing order* **do**
  $B_1, \ldots, B_m \leftarrow decomposition\_phase(d)$
  $(end, r^d) \leftarrow detection\_phase(d, B)$
  $(t^d(c), r^d(c)) \leftarrow update\_phase(d, B)$
forall $i = 1, \ldots, n$ **do**
  $r_i = \max\{r_i^d, r^d(c_i) - p_i(t^d(c_i)) : d \leq end[i], d \in T_2\}$
return r_i for all $i = 1, \ldots, n$

get a final running time of $O(n^2 \log n)$ for energetic edge-finding that is subsumed in Algorithm 3. After termination of the outer loop, we finally update the release date of every job $i = 1, \ldots, n$ with $p_i^s > 0$ by

$$r_i^* = \max\{r_i, \max\{r_i^d : d \leq end[i], d \in T_2\}\} \tag{20}$$

that takes an additional $O(k \cdot n)$ time.

3.4 Integration of Symmetric Intervals

In this section, we show how the remaining set of time intervals T_{13} can be integrated into the given concepts. The main problem here is that d depends on t and not vice versa. A first important observation is that left-shift propagations on T_{13} are symmetric to *right-shift* propagations on T_{23}. Thus, our approach is to include the line segments of the corresponding right-shift slack function $c_i \cdot (p_i^r(t) - p_i(t)) > -\tilde{\omega}(t)$ with $p_i^r(t) = \max\{0, \min\{p_i, d-t, d_i - t, d - d_i + p_i\}\}$ with the propagation $d_i^* = t + p_i(t) - \frac{\tilde{\omega}(t)}{c_i}$ and set $start[i] = t$. In order to perform the update phase we have to build the canonical decomposition in $[t, \infty)$ but now according to due dates and from right to left and proceed symmetrically. The last step is not implemented in our algorithm. However, it is a complete energetic reasoning algorithm. In respect to the scope of this paper, we will not explicitly formulate this case during the next sections.

4 Relation to Standard and Extended Edge-Finding

Energetic edge-finding is strongly motivated by standard edge-finding [21]. In particular, energetic edge-finding can be slightly modified to give a complete $O(n^2)$ edge-finding but that additionally performs stronger propagations from energetic reasoning.

For every interval $[t, d]$ standard edge-finding considers only propagations of jobs i with $t \leq r_i < d$. We can replace our detection phase to include all such

intervals for fixed d in linear time as follows: we iterate over all r_i with $r_i < d$ in non-increasing order and check immediately if $\tilde{\omega}(t) + c_i \cdot (p_i^l(t) - p_i(t)) > 0$ holds. Under the condition $t \leq r_i$, we have by monotonicity of $\tilde{\omega}(t)$ that $t = r_i$ maximally satisfies this inequality and yields the maximum propagation update. If the inequality is satisfied we set $end[i] = d$, otherwise no propagation can be found for job i. The update phase is executed as previously introduced. Note that we have to iterate over $t \in T_B$ simultaneously to get the $\tilde{\omega}(t)$ values.

Consequently, for fixed due date d the detection phase takes $O(n+m)$, see also Algorithm 2. In total, this yields a complete edge-finding algorithm in $O(n^2)$ time but that additionally performs stronger propagations because the basic energy in every interval $[t, d]$ is taken from energetic reasoning. Since the currently best complete edge-finding algorithm has complexity $O(kn \log n)$ [21], our algorithm constitutes a further improvement in the landscape of energetic propagators.

Corollary 2. *There exists a complete $O(n^2)$ algorithm for edge-finding that additionally takes energetic reasoning as energy lower bound in each time interval.*

A further open question asks for the relation to *extended edge-finding* [14] that includes propagations for partially overlapping jobs. In particular, the case of partially overlapping jobs is exactly the bottleneck of our algorithm since by the 'usual' approach of fixing one interval limit, say $t_2 = d$, and computing the partially overlapping job i in an interval $[t, d]$ with $r_i < t < r_i + p_i^s$ that *maximally* contributes to the energy in $[t, d]$ will always lead to an upper envelope problem. Hence, it is unlikely that there exists a complete $O(n^2)$ algorithm for extended edge-finding by using known techniques. In turn, if there exists an efficient method to compute such propagations then we believe it can also be used for energetic reasoning, thus indicating that extended edge-finding and energetic reasoning are of the same complexity.

5 Improving Propagations by Detectable Precedences

In this section we present a relaxation of energetic edge-finding and a possible improvement for energetic edge-finding.

In every propagation that is performed for job i due to a left-shift overload in the time interval $[t, d]$ we can improve the propagation by considering the minimum earliest completion time of the jobs that are contributing to the energy in $[t, d]$. In other words, if $\tilde{\omega}(t) + c_i \cdot (p_i^l(t) - p_i(t)) > 0$ then we can propagate

$$r_i^d = \min_{\substack{j \neq i: \\ p_j(t) > 0}} (r_j + p_j). \tag{21}$$

To the best of our knowledge, we believe that this rule, in its generality to energetic reasoning, has not been investigated so far. Since our algorithms iterates over all t in non-increasing order we can include such propagations by storing an update value that is set to $r_j + p_j$ whenever $t = r_j + p_j$ for jobs j with $p_j(0) > 0$. As for energetic edge-finding, we can extend this rule by propagations from subintervals of $[t, d]$ or subsets of jobs if there is still an overload.

In the following we show that detectable precedences naturally extends energetic edge-finding since both propagations are incomparable.

Example 1. Given a CuSP instance of four jobs with $p_1 = p_2 = p_3 = p$ and $p_4 = 2p$ for some large p. Moreover, let $c_i = 1$ for all $i = 1, \ldots, n$ and $C = 2$. The scheduling intervals $[r_i, d_i]$ for $i = 1, \ldots, 4$ are given by $[0, 3p-1]$, $[0, 2p]$, $[0, 2p]$ and $[0, \infty)$. Energetic reasoning propagates job 4 according to the interval $[0, 2p]$ since for $d = 2p$ we have $\tilde{\omega}(0) + c_i \cdot (p_i^l(0) - p_i(0)) = (2p+1-4p) + (2p-0) = 1 > 0$ and thus we propagate $r_4^{2p} = d - p_4(0) + \tilde{\omega}(0) = 2p - 0 + (2p+1-4p) = 1$. After that, no further propagation is detected, so the fixpoint is reached for energetic edge-finding and all dominated rules. In contrast, detectable precedences (21) finds that $\min\{r_j + p_j : p_j(0) > 0, j \neq i\} = p$ and thus we propagate $r_4^{2p} = p$. This is the strongest possible propagation. Note that the not-first/not-last rule does not consider the partial overlap of job 1, so nothing is propagated.

Example 2. Given a CuSP instance of three jobs with $p_1 = p_2 = p$ and $p_3 = 1$ for some $p > 0$. Moreover, let $c_1 = c_2 = c_3 = 1$ and $C = 1$. The scheduling intervals $[r_i, d_i]$ for $i = 1, 2, 3$ are given by $[0, 2p]$, $[0, 2p]$, $[0, \infty)$. The interval $[0, 2p]$ contains full energy of $2p$. Thus, detectable precedences updates $r_3 = p$ while energetic reasoning (even edge-finding) updates the strongest possible propagation $r_3 = 2p$.

Corollary 3. *Detectable precedences and energetic reasoning are incomparable.*

The given examples reveal the weakness of current energetic approaches: the lack of knowledge about the inner structure of an interval and its possible realizations. Future propagators should make stronger predictions by analyzing how possible realizations of a certain interval may look like.

6 Computational Results

In the following we give a rough analysis of the computational performance of the presented methods. We use the test sets J30, J60 and J120 that contain 480, 480 and 600 instances respectively from the well-established PSPLIB [13] for the Resource-Constrained Project Scheduling Problem (RCPSP). The name of the test set indicates the number of considered jobs and every instance has four resources with additional precedence constraints.

Our algorithms are programmed in C language with GCC compiler version 7.3.0 and executed on a Intel Xeon CPU E5-2660 with 2.60 GHz using a single thread. We compute lower bounds to the RCPSP by destructive improvement, that is we start with a lower bound on the makespan and increase it as long as we detect infeasibility or the time limit of 600 seconds is reached.

We use a *static branching rule* [17] that schedules the jobs in order of earliest release dates. Ties are broken by minimum domain value $d_i - p_i - r_i$. Additionally, we apply a dominance rule that skips the right branch of the currently branched job i if it does not contribute to the earliest resource conflict when

	J30			J60			J120		
	opt	Δ LB	nodes/s	opt	Δ LB	nodes/s	opt	Δ LB	nodes/s
ER	384	0	48.70	347	0	15.96	141	0	8.32
EnEF	393	126	581.35	349	86	249.68	143	77	92.93
EnEF (w/o up)	394	174	662.76	349	86	246.44	143	77	94.12
EnEF (relaxed)	405	221	680.25	348	85	245.26	143	79	96.63

Table 1. Computational results for the J30, J60 and J120 test sets.

all jobs are scheduled at their release dates. We consider a static branching rule rather than a dynamic one to compare the real performance between the different propagators. For the computation of best possible solutions to the RCPSP, we more sophisticated methods [17] that exceed the scope of this paper.

In every search node we first apply timetabling [7,12] in combination with one of the following: energetic reasoning as given in [1] with complexity $O(n^3)$, energetic edge-finding, energetic edge-finding without the update phase and the incomplete $O(n^2)$ energetic edge-finding algorithm of Section 4. All energetic edge-finding propagators include propagations from detectable precedences.

Table 1 displays the number of optimal solutions found (opt), the deviation in the total sum of lower bounds (Δ LB) normed to the weakest propagator and the average number of backtracking nodes for instances that took longer than five seconds to solve. As expected the $O(n^3)$ energetic reasoning propagator (ER) is slower than energetic edge-finding (EnEF) such that energetic edge-finding is able to compute better lower bounds in total. The exclusion of the update phase and the relaxed variant yield speedups only on small instances. We conclude that integrating all subintervals has mainly a theoretical rather than practical relevance. Surprisingly, the relaxed version of energetic edge-finding cannot profit from its $O(n^2)$ complexity on large instances. The reason is that our output-sensitive implementation of the detection phase runs in almost the same time because the left-shift slack intervals of the jobs are mostly disjoint.

7 Conclusion

We believe that the monotonicity in the single machine interpretation of energetic schedules can lead to new ideas for faster propagation algorithms for the CuSP and related problems. Since energetic arguments have natural limitations, one future direction can be to combine energetic propagations with arguments about the inner structure of time intervals to derive stronger propagations. A first approach is made with detectable precedences. Another open question is to resolve the fixpoint complexity of energetic reasoning.

Acknowledgements. The author would like to thank anonymous reviewers for their helpful comments on the paper, especially for the advice to take a simpler representation of $\tilde{\omega}(t)$ as given in the current version of the paper.

References

1. Baptiste, P., Le Pape, C., Nuijten, W.: Satisfiability tests and time bound adjustments for cumulative scheduling problems. Annals of Operations research, 92, 305-333. (1999)
2. Berthold, T., Heinz, S., Schulz, J.: An approximative criterion for the potential of energetic reasoning. In Theory and Practice of Algorithms in (Computer) systems (pp. 229-239). Springer Berlin Heidelberg. (2011)
3. Bonifas, N.: A $O(n^2log(n))$ propagation for the Energy Reasoning, Conference Paper, Roadef 2016. (2016)
4. De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O. C.: Computational Geometry. In Computational Geometry (pp. 1-17). Springer, Berlin, Heidelberg (2000)
5. Derrien, A., Petit, T.: A new characterization of relevant intervals for energetic reasoning. In Principles and Practice of Constraint Programming (pp. 289-297). Springer International Publishing. (2014)
6. Erschler, J., Lopez, P.: Energy-based approach for task scheduling under time and resources constraints. In 2nd international workshop on project management and scheduling, pp. 115-121. (1990)
7. Gay, S., Hartert, R., Schaus, P.: Simple and scalable time-table filtering for the cumulative constraint. In International conference on principles and practice of constraint programming (pp. 149-157). Springer, Cham. (2015)
8. Goemans, M. X.: A supermodular relaxation for scheduling with release dates. In International Conference on Integer Programming and Combinatorial Optimization (pp. 288-300). Springer, Berlin, Heidelberg. (1996)
9. Hooker, J. N.: A hybrid method for planning and scheduling. In International Conference on Principles and Practice of Constraint Programming (pp. 305-316). Springer, Berlin, Heidelberg. (2004)
10. Kameugne, R., Fetgo, S. B., Fotso, L. P. . Energetic extended edge finding filtering algorithm for cumulative resource constraints. American Journal of Operations Research, 3(06), 589. (2013)
11. Lenstra, J. K., Kan, A. R., Brucker, P.: Complexity of machine scheduling problems. In Annals of Discrete Mathematics (Vol. 1, pp. 343-362). Elsevier. (1977)
12. Letort, A., Beldiceanu, N., Carlsson, M. . A scalable sweep algorithm for the cumulative constraint. In Principles and Practice of Constraint Programming (pp. 439-454). Springer, Berlin, Heidelberg. (2012)
13. Kolisch, R., Sprecher, A.: PSPLIB-a project scheduling problem library: OR software-ORSEP operations research software exchange program. (1997) European journal of operational research, 96(1), 205-216.
14. Mercier, L., Van Hentenryck, P.: Edge finding for cumulative scheduling. INFORMS Journal on Computing, 20(1), 143-153. (2008)
15. Mercier, L., & Van Hentenryck, P.: Strong polynomiality of resource constraint propagation. Discrete Optimization, 4(3-4), 288-314. (2007)
16. Ouellet, P., Quimper, C. G.: Time-table extended-edge-finding for the cumulative constraint. In Principles and Practice of Constraint Programming (pp. 562-577). Springer Berlin Heidelberg. (2013)
17. Schutt, A., Feydy, T., Stuckey, P. J., Wallace, M. G.: Explaining the cumulative propagator. Constraints, 16(3), 250-282. (2011)
18. Schutt, A., Wolf, A.: A New $\mathcal{O}(n^2 \log n)$ Not-First/Not-Last Pruning Algorithm for Cumulative Resource Constraints. In Principles and Practice of Constraint Programming - CP 2010 (pp. 445-459). Springer Berlin Heidelberg. (2010)

19. Schutt, A., Feydy, T., Stuckey, P. J.: Explaining time-table-edge-finding propagation for the cumulative resource constraint. In Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (pp. 234-250). Springer Berlin Heidelberg. (2013)
20. Tesch, A. A Nearly Exact Propagation Algorithm for Energetic Reasoning in $\mathcal{O}(n^2 \log n)$. In International Conference on Principles and Practice of Constraint Programming (CP 2016) (pp. 493-519). Springer International Publishing. (2016)
21. Vilim, P.: Edge Finding Filtering Algorithm for Discrete Cumulative Resources in $\mathcal{O}(kn \log n)$. In Principles and Practice of Constraint Programming - CP 2009 (pp. 802-816). Springer Berlin Heidelberg. (2009)
22. Vilim, P.: Timetable edge finding filtering algorithm for discrete cumulative resources. In Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (pp. 230-245). Springer Berlin Heidelberg. (2011)