Edge Finding for Cumulative Scheduling

Luc Mercier and Pascal Van Hentenryck
Brown University, Box 1910, Providence, RI 02912
Email: {mercier,pvh}@cs.brown.edu

May 27, 2005

Abstract

Edge-finding algorithms for cumulative scheduling are at the core of commercial constraint-based schedulers. This paper shows that Nuijten's edge finder for cumulative scheduling, and its derivatives, are incomplete and use an invalid dominance rule. The paper then presents a new edge-finding algorithm for cumulative resources which runs in time $O(n^2k)$, where n is the number of tasks and k the number of different capacity requirements of the tasks. The new algorithm is organized in two phases and first uses dynamic programming to precompute the innermost maximization in the edge-finder specification. Finally, this paper also proposes the first extended edge-finding algorithm that runs in time $O(n^2k)$, improving the running time of available algorithms.

1 Introduction

Edge finding [CP94] is a fundamental pruning technique for disjunctive and cumulative scheduling[1] and is an integral part of commercial constraint-based schedulers. Informally speaking, an edge finder considers one resource at a time, identifies pairs (Ω, i) such that task i cannot precede (resp. follow) any task from Ω in all feasible schedules, and updates the earliest starting date (resp. latest finishing date) of task i accordingly. An edge-finding algorithm is a procedure that performs all such deductions.

Edge finding is well-understood for unary resources, i.e., resources with capacity one. Indeed, there exist efficient algorithms running in time $O(n \log n)$ or $O(n^2)$, where n is the number of tasks on the resource [CP94, Nui94, Vil04]. Edge finding is more challenging for cumulative resources whose capacity is a natural number $C \geq 1$ and whose tasks may require several capacity units. Nuijten [Nui94] (see also [NA96, BLPN01]) proposed an edge-finding algorithm running in time $O(n^2k)$, where $k \leq n$ is the number of distinct capacity requirements of the tasks. This algorithm was later refined to run in $O(n^2)$ [BLPN01].

This paper shows that Nuijten's algorithm, and its refinement, are incomplete and do not perform all the edge-finding updates. The mistake comes from the use of an incorrect dominance rule which holds for unary resources but does not carry over to the cumulative case. The paper also presents a new, two-phase, edge finder for cumulative resources that runs in $O(n^2k)$. The first phase is a

[1] This paper considers only non-preemptive problems, where tasks cannot be interrupted.

dynamic programming algorithm that precomputes the potential edge-finding updates. The second phase uses the precomputation to apply the actual updates. Moreover, similar ideas can be used to derive an $O(n^2k)$ for the extended edge-finding rule, improving the running time of the best available algorithms. The contributions of this paper can thus be summarized as follows:

1. This paper shows that Nuijten's algorithm and its derivatives are incomplete with respect to the edge-finding rule;
2. This paper presents a complete edge-finding algorithm that runs in time $O(n^2k)$;
3. This paper presents a complete extended edge-finding algorithm running in time $O(n^2k)$, improving the complexity of the best-known algorithm.

The rest of this paper is organized as follows. Section 2 specifies the problem and the notations used in the paper. Section 3 proves that Nuijten's algorithm is incomplete. Sections 4, 5, and 6 are the core of the paper: they present the dominance properties used for cumulative edge finding and the edge-finding algorithm itself. Section 7 presents the extended edge-finding algorithm, and Section 8 concludes the paper.

2 Problem Definition and Notations

Definition 1 (Cumulative Resource Problems) *A cumulative resource problem (CRP) is specified by a cumulative resource of capacity C and a set of tasks T. Each task $t \in T$ is specified by its release date r_t, its deadline d_t, its processing time p_t, and its capacity requirement c_t, all of which being natural numbers. A solution to a CRP $\mathcal{P}$ is a schedule that assigns a starting date s_t to each task t so that*

$$\forall t \in T : r_t \leq s_t \leq s_t + p_t \leq d_t$$

and

$$\forall i : \sum_{\substack{t \in T \\ s_t \leq i < s_t + p_t}} c_t \leq C.$$

The set of solutions to a CRP $\mathcal{P}$ is denoted by $sol(\mathcal{P})$. Finally, Sc denotes the set $\{c_t \mid t \in T\}$ of all capacity requirements, n denotes $|T|$, $N = \{1, \ldots, n\}$, k denotes $|Sc|$, and $e_t = c_t p_t$ denotes the energy of a task t.

In the following, we abuse notations and assume an underlying CRP with its resource and tasks specified as in Definition 1. We also lift the concepts of release dates, due dates, and energies to sets of tasks, i.e.,

$$r_\Omega = \min_{j \in \Omega} r_j$$

$$d_\Omega = \max_{j \in \Omega} d_j$$

$$e_\Omega = \sum_{j \in \Omega} e_j$$

where Ω is a set of tasks. By convention, when Ω is the empty set, $r_\Omega = \infty$, $d_\Omega = -\infty$ and $e_\Omega = 0$.
The CRP is NP-complete and constraint-based schedulers typically use a relaxation of feasibility to prune the search space.

Definition 2 (E-Feasibility) *A CRP is E-feasible if*

$$\forall \Omega \subseteq T : C(d_\Omega - r_\Omega) \geq e_\Omega.$$

Obviously, feasibility of a CRP implies E-feasibility. A critical aspect of constraint-based schedulers is to reduce the possible starting and finishing dates that appear in solutions. The edge-finding rule is one of the fundamental techniques to reduce these dates in disjunctive and cumulative scheduling. This paper restricts attention to starting dates only (the handling of finishing dates is similar), in which case the key idea underlying the edge-finding rule can be summarized as follows. Consider a set of tasks Ω and a task $i \in T \setminus \Omega$. If the condition

$$C(d_\Omega - r_{\Omega\cup\{i\}}) < e_{\Omega\cup\{i\}}$$

holds, then there exists no schedule in which task i precedes any operation in Ω. As a consequence, in any feasible schedule, the starting date s_i must satisfy

$$s_i \geq r_\Theta + \left\lceil \frac{1}{c_i} \operatorname{rest}(\Theta, c_i) \right\rceil$$

for all $\Theta \subseteq \Omega$ satisfying

$$\operatorname{rest}(\Theta, c_i) > 0$$

where

$$\operatorname{rest}(\Theta, c_i) = \begin{cases} e_\Theta - (C - c_i)(d_\Theta - r_\Theta) & \text{if } \Theta \neq \emptyset; \\ 0 & \text{otherwise.} \end{cases}$$

Informally speaking, $\operatorname{rest}(\Theta, c_i)$ is the energy of e_Ω that cannot be accommodated by a cumulative resource of capacity $C - c_i$ in the interval $[r_\Theta, d_\Theta)$. The proofs of these results can be found in [BLPN01]. We are now ready to specify the edge-finding algorithm.

Specification 1 (Edge Finding) *The edge-finding algorithm receives as input an E-feasible CRP. It produces as output a vector*

$$\left\langle \overline{LB_2}(1), \ldots, \overline{LB_2}(n) \right\rangle$$

where

$$\overline{LB_2}(i) = \max(r_i, LB_2(i))$$

and

$$LB_2(i) = \max_{\substack{\Omega \subseteq T \\ i \notin \Omega \\ \alpha(\Omega, i)}} \; \max_{\substack{\Theta \subseteq \Omega \\ \operatorname{rest}(\Theta, c_i) > 0}} \; r_\Theta + \left\lceil \frac{1}{c_i} \operatorname{rest}(\Theta, c_i) \right\rceil$$

with

$$\alpha(\Omega, i) \iff \left(C(d_\Omega - r_{\Omega\cup\{i\}}) < e_{\Omega\cup\{i\}} \right).$$

3 Incompleteness of Nuijten's Algorithm

We now consider algorithm CalcLB (Figure 4.9 in [Nui94]; see also [BLPN01]), which is is reproduced in Algorithm 1 for simplicity.[2] Nuijten claims that CalcLB computes $\overline{LB_2}(i)$ for all $i \in T$, which is incorrect. Consider the following instance on a resource of capacity 4:

task	r	d	p	c
a	0	69	4	1
b	1	2	1	4
c	0	3	1	2
d	0	3	1	2
e	2	3	1	1

Consider the pair (Ω, Θ) where $\Omega = T \setminus \{a\}$ and $\Theta = \{b\}$. The condition $\alpha(\Omega, a)$ holds because $e_{\Omega \cup \{a\}} = 13$ and $C(d_\Omega - r_{\Omega \cup \{a\}}) = 4 \times 3 = 12$. Moreover, we have $\Theta \subseteq \Omega$ and $\text{rest}(\Theta, c_a) = 1$ which implies

$$LB_2(a) \geq r_\Theta + \frac{1}{c_a} \text{rest}(\Theta, c_a) = 2.$$

Algorithm CalcLB does not perform this deduction because it never considers the pair (Ω, Θ). Instead, CalcLB considers the pair (Ω, Ω). But since $\text{rest}(\Omega, c_a) = 0$, no update takes place. The problem with CalcLB is apparent in line 7 which maintains l as the maximum due date of Ω. This maximal value is then used to compute (incorrectly) the rest in line 9, performing no update on the relevant g_j and thus no update on the release date of task a (in lines 19 and 22).

It is easy to understand why Nuijten made this mistake. The algorithm CalcLB for cumulative scheduling is derived from a similar algorithm for disjunctive scheduling (resources have capacity 1). In disjunctive scheduling, $C - c_i$ is always zero and $\text{rest}(\Theta, c_i)$ does not depend on d_Θ. It is thus always beneficial for a given r_Θ to add more tasks when computing the inner maximization. This is not the case in cumulative scheduling, where this dominance relation does not hold as the instance above indicates. We now prove formally that CalcLB does not compute $\overline{LB_2}(a)$ by tracing the algorithm.

Theorem 1 *Algorithm* CalcLB *does not compute* $\overline{LB_2}(i)$ $(i \in T)$.

Proof Consider the following instance on a resource of capacity 4:

task	r	d	p	c
a	0	69	4	1
b	1	2	1	4
c	0	3	1	2
d	0	3	1	2
e	2	3	1	1

We showed earlier that $LB_2(a) \geq 2$ by considering the pair (Ω, Θ) where $\Omega = T \setminus \{a\}$ and $\Theta = \{b\}$. Algorithm CalcLB considers only three due dates $\{2, 3, 69\}$ and performs the following processing.

[2] In CalcLB, $lct(t)$ corresponds to d_t, $est(t)$ to r_t, $a(t)$ to e_t, $sz(t)$ to $c(t)$, and $LB_{est}(t)$ to $LB_2(t)$. Our notations are consistent with [BLPN01].

Algorithm 1 CalcLB

Require: X is an array of tasks sorted by non-decreasing release dates;
Require: Y is an array of tasks sorted by non-decreasing due dates;
1: **for** $y \leftarrow 1$ **to** n **do**
2:   **if** $y = n \vee d_{Y[y]} \neq d_{Y[y+1]}$ **then**
3:     $E \leftarrow 0$; $l \leftarrow -\infty$; **for all** $c \in Sc$ **do** $g_c \leftarrow -\infty$; **endfor**
4:     **for** $i \leftarrow n$ **downto** 1 **do**
5:       **if** $d_{X[i]} \leq d_{Y[y]}$ **then**
6:         $E \leftarrow E + e_{X[i]}$;
7:         **if** $d_{X[i]} > l$ **then** $l \leftarrow d_{X[i]}$; **endif**
8:         **for all** $c \in Sc$ **do**
9:           $rest \leftarrow E - (l - r_{X[i]})(C - c)$;
10:           **if** $rest/c > 0$ **then** $g_c \leftarrow \max(g_c, r_{X[i]} + \lceil rest/c \rceil)$;
11:         **end for**
12:       **end if**
13:       **for all** $c \in Sc$ **do** $G[i][c] \leftarrow g_c$; **endfor**
14:     **end for**
15:     $H \leftarrow -\infty$;
16:     **for** $x \leftarrow 1$ **to** n **do**
17:       **if** $d_{X[x]} > d_{Y[y]}$ **then**
18:         **if** $E + e_{X[x]} > (d_{Y[y]} - r_{X[x]}) \times C$ **then**
19:           $LB[x] \leftarrow \max(LB[x], G[x][c_{X[x]}])$;
20:         **end if**
21:         **if** $H + (e_{X[x]}/C) > d_{Y[y]}$ **then**
22:           $LB[x] \leftarrow \max(LB[x], G[1][c_{X[x]}])$;
23:         **end if**
24:       **else**
25:         $H \leftarrow \max(H, r_{X[x]} + E/C)$;
26:         $E \leftarrow E - e_{X[x]}$;
27:       **end if**
28:     **end for**
29:   **end if**
30: **end for**

$\mathbf{d}_\Omega = 69$. All tasks have due dates not greater than 69, and the test $d_{X[x]} > d_{Y[y]}$ always fails in line 17. No bound is improved.

$\mathbf{d}_\Omega = 3$. The only task satisfying $d_{X[x]} > d_{Y[y]}$ (i.e., $d_i > d_\Omega$) is a, so only $\overline{LB_2}(a)$ can be updated. Since the tasks are considered by decreasing release dates starting with e, l is updated to 3 immediately and never decreases. As a consequence, $rest$ is never positive and all values $G[t][c]$ are equal to $-\infty$ at the end of the first inner loop. No update can take place in the second inner loop.

$\mathbf{d}_\Omega = 2$. The only task with a due date not greater than 2 is b. Since $\alpha(\{b\}, i)$ does not hold for any task $i \neq b$, no bound is improved.

This shows that algorithm CalcLB does not improve any bound on this instance, contradicting the claim that CalcLB is an edge-finding algorithm. ■

Note that the proof shows an even stronger result: s_a will not be updated even by iterating CalcLB, since a fixpoint is reached after the first iteration.

The result directly propagates to the $O(n^2)$ algorithm NBLP (algorithm 8, section 3.3.3 in [BLPN01]). Indeed, NBLP refines the first inner loop of CalcLB and suffers from the same defect. (The same instance exhibits the mistake).[3]

It is also unlikely that the structure of CalcLB can be salvaged. Indeed, this would require the correct computation of all the G values in time $O(nk)$, which seems to be intrinsically two-dimensional. The algorithm proposed in this paper remedies this problem by removing the first inner loop and using dynamic programming to precompute the inner maximizations in the $LB_2(i)$ definitions. The dynamic programming algorithm exploits some new dominance rules, which are also used to simplify the second inner loop.

4 Dominance Properties

Before presenting the algorithm, it is important to review the dominance properties used by the algorithms.

4.1 Dominance Property for E-Feasibility

Testing E-feasibility only relies on a single dominance property based on the concept of task intervals [CL94].

Definition 3 (Task Intervals) *Let $L, U \in T$. The task interval Ω_L^U is the set of tasks*

$$\Omega_L^U = \{k \in T \mid r_k \geq r_L \ \wedge \ d_k \leq d_U\}.$$

Note that it is not always the case that $d_{\Omega_L^U} = d_U$ and $r_{\Omega_L^U} = r_L$. Indeed, the tasks L and U are not necessarily included in Ω_L^U. Algorithms for testing E-feasibility only need to consider task intervals.

Proposition 1 *E-feasibility testing only needs to consider task intervals.*

Proof Consider a set Ω such that $C(d_\Omega - r_\Omega) < e_\Omega$ and a set Ω_L^U such that $r_L = r_\Omega \wedge d_U = d_\Omega$. Since $\Omega \subseteq \Omega_L^U$, $C(d_{\Omega_L^U} - r_{\Omega_L^U}) < e_{\Omega_L^U}$. ■.

4.2 Dominance Properties for Edge Finding

Edge-finding algorithms heavily rely on dominance properties in order to reduce the pairs (Ω, Θ) to consider when updating a task i. This section reviews the dominance properties used in our algorithm. Some of them are well-known, others are new. The first three properties reduce the sets Ω that must be considered in the pairs (Ω, Θ) for a task i. The last two reduce the sets Θ to consider. In the following, we restrict attention to E-feasible CRPs only.

Definition 4 (Valid Pair) *A pair (Ω, Θ) is* valid *wrt task i if*

$$i \notin \Omega \ \wedge \ \alpha(\Omega, i) \ \wedge \ \Theta \subseteq \Omega \ \wedge \ \mathrm{rest}(\Theta, c_i) > 0.$$

[3] We will discuss NBLP again, once we have presented a correct edge-finding algorithm for cumulative resources.

Definition 5 (Maximal Pair) *A pair* (Ω, Θ) *is* maximal *wrt task* i *if it is valid and satisfies*

$$LB_2(i) = r_\Theta + \left\lceil \frac{1}{c_i} \operatorname{rest}(\Theta, c_i) \right\rceil .$$

Proposition 2 *The computation of* $LB_2(i)$ *for an E-feasible CRP only needs to consider pairs of the form* (Ω_L^U, Θ) $(L, U \in T)$.

Proof Consider a maximal pair (Ω, Θ) and a set Ω_L^U such that $r_L = r_\Omega \wedge d_U = d_\Omega$. Since $\Omega \subseteq \Omega_L^U$ and the inner maximization in $LB_2(i)$ only involves Θ, it suffices to prove that (Ω_L^U, Θ) is valid. Since $\alpha(\Omega, i)$ holds and the CRP is E-feasible, $i \notin \Omega_L^U$. Moreover, by definition of Ω_L^U and since $\Omega \subseteq \Omega_L^U$, $\alpha(\Omega_L^U, i)$ holds and the pair (Ω_L^U, Θ) is valid and maximal. ■

The following dominance property relates the pairs with task i.

Proposition 3 *The computation of* $LB_2(i)$ *for an E-feasible CRP may restrict attention to pairs* (Ω_L^U, Θ) *where* $d_U = d_{\Omega_L^U} < d_i$.

Proof Consider a maximal pair (Ω_L^U, Θ). There exists a task $U' \in \Omega_L^U$ such that $d_{U'} = d_{\Omega_L^U}$. Since $\Omega_L^U = \Omega_L^{U'}$, the pair $(\Omega_L^{U'}, \Theta)$ is also maximal. Assume now that $d_{U'} \geq d_i$ and let $\Omega' = \Omega_L^{U'} \cup \{i\}$. Since $d_{\Omega'} = d_{U'}$ and $\alpha(\Omega_L^{U'}, i)$ holds, it follows that $C\,(d_{\Omega'} - r_{\Omega'}) < e_{\Omega'}$, which contradicts E-feasibility. ■

Proposition 3 allows us to remove the constraint $i \notin \Omega$ from $LB_2(i)$, since it is implied by $d_U < d_i$. The following dominance property is new and imposes a restriction on the tasks L used to define the sets Ω_L^U for $LB_2(i)$.

Proposition 4 *The computation of* $LB_2(i)$ *for an E-feasible CRP only needs to consider pairs* (Ω_L^U, Θ) *where* $d_{\Omega_L^U} = d_U < d_i$ *and* $r_L = r_{\Omega_L^U \cup \{i\}}$.

Proof Consider a maximal pair (Ω_L^U, Θ) such that $d_U < d_i$ and let $L' \in T$ be a task such that $r_{L'} = \min(r_i, r_{\Omega_L^U})$. Since $\Omega_L^U \subseteq \Omega_{L'}^U$ and the inner maximization only depends on Θ, it suffices to show that $\Omega_{L'}^U$ is valid. Since $d_U < d_i$, $i \notin \Omega_{L'}^U$. Moreover, since $r_{\Omega_L^U \cup \{i\}} = r_{\Omega_{L'}^U \cup \{i\}}$ and $\Omega_L^U \subseteq \Omega_{L'}^U$, $\alpha(\Omega_{L'}^U, i)$ holds and the result follows. ■

The following proposition summarizes the first three dominance properties.

Proposition 5 *For a E-feasible CRP,* $LB_2(i)$ *may be computed as*

$$LB_2(i) = \max_{\substack{L, U \in T \\ \alpha(\Omega_L^U, i) \\ d_U = d_{\Omega_L^U} < d_i \\ r_L = r_{\Omega_L^U \cup \{i\}}}} \; \max_{\substack{\Theta \subseteq \Omega_L^U \\ \operatorname{rest}(\Theta, c_i) > 0}} r_\Theta + \left\lceil \frac{1}{c_i} \operatorname{rest}(\Theta, c_i) \right\rceil$$

The next two dominance properties concern the choice of Θ. The first one is the counterpart of Proposition 2 for Θ.

Proposition 6 *The computation of* $LB_2(i)$ *for an E-feasible instance only needs to consider pairs* (Ω_L^U, Ω_l^u) $(r_L \leq r_l \leq d_u \leq d_U)$ *satisfying* $r_l = r_{\Omega_l^u}$ *and* $d_u = d_{\Omega_l^u}$.

Proof Consider a maximal pair (Ω_L^U, Θ) and a set $\Omega_l^u \subseteq \Omega_L^U$ such that $r_l = r_\Theta \wedge d_u = d_\Theta$. It follows that $\Theta \subseteq \Omega_l^u$, $r_\Theta = r_{\Omega_l^u}$, and $\text{rest}(\Theta, c_i) \leq \text{rest}(\Omega_l^u, c_i)$. Hence, (Ω_L^U, Ω_l^u) is maximal for task i. ■

The above dominance properties restrict the set of pairs to consider in computing $LB_2(i)$. The next property is of a fundamentally different nature: it increases the set of pairs (Ω, Θ) to consider by relaxing the constraint $r_l \geq r_L$ (and thus $\Theta \subseteq \Omega$). This dominance relation, which generalizes Theorem 4.13 in [Nui94], enables us to amortize the precomputation of inner maximizations of $LB_2(i)$ $(i \in T)$ effectively and to simplify the second inner loop of CALCLB.

Proposition 7 *Consider the function LB_2' defined as*

$$LB_2'(i) = \max_{\substack{L,U \in T \\ \alpha(\Omega_L^U, i) \\ d_U = d_{\Omega_L^U} < d_i \\ r_L = r_{\Omega_L^U \cup \{i\}}}} \quad \max_{\substack{l,u \in T \\ r_{\Omega_l^u} = r_l \\ d_{\Omega_l^u} = d_u \leq d_U \\ \text{rest}(\Omega_l^u, c_i) > 0}} \quad r_l + \left\lceil \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) \right\rceil$$

For any E-feasible CRP, $\overline{LB_2}(i) = \overline{LB_2'}(i)$, where $\overline{LB_2'}(i) = max(r_i, LB_2'(i))$.

Proof By Proposition 5 and Proposition 6, $LB_2(i)$ can be rewritten as

$$LB_2(i) = \max_{\substack{L,U \in T \\ \alpha(\Omega_L^U, i) \\ d_U = d_{\Omega_L^U} < d_i \\ r_L = r_{\Omega_L^U \cup \{i\}}}} \quad \max_{\substack{l,u \in T \\ r_{\Omega_l^u} = r_l \\ d_{\Omega_l^u} = d_u \leq d_U \\ r_l \geq r_L \\ \text{rest}(\Omega_l^u, c_i) > 0}} \quad r_l + \left\lceil \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) \right\rceil$$

It follows that $LB_2(i) \leq LB_2'(i)$. Moreover, by definition of $\overline{LB_2}(i)$ and $\overline{LB_2'}(i)$, it is sufficient to consider the case where $LB_2'(i) > r_i$ and to show that $LB_2(i) \geq LB_2'(i)$. Consider $L, U, l, u \in T$ satisfying

$$\begin{cases} r_L = r_{\Omega_L^U \cup \{i\}} \\ \alpha(\Omega_L^U, i) \\ r_{\Omega_l^u} = r_l \\ d_{\Omega_l^u} = d_u \leq d_U = d_{\Omega_L^U} < d_i \\ \text{rest}(\Omega_l^u, c_i) > 0 \\ LB_2'(i) = r_l + \left\lceil \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) \right\rceil \end{cases}$$

If $r_l \geq r_L$, then $LB_2(i) \geq LB_2'(i)$. Otherwise, partition Ω_l^u in $\Theta \cup \Omega_L^u$ where $\Theta = \Omega_l^u \setminus \Omega_L^u$. The rest of the proof proceeds by a case analysis. Informally speaking, in the first case, the set Θ has enough energy to cover $C(r_L - r_l)$ and the computation of $LB_2(i)$ for (Ω_l^U, Ω_l^u) is at least as good as the computation of $LB_2'(i)$ on (Ω_L^U, Ω_l^u). In the second case, Θ does not cover $C(r_L - r_l)$ and $LB_2(i)$ on (Ω_L^U, Ω_L^u) is at least as good as $LB_2'(i)$.

Assumption 1: Consider the case

$$r_l + \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) > r_L + \frac{1}{c_i} \text{rest}(\Omega_L^u, c_i). \tag{1}$$

We first rewrite the left-hand side of (1). By definition of rest, we have

$$c_i r_l + \text{rest}(\Omega_l^u, c_i) = c_i r_l + e_{\Omega_l^u} - (C - c_i)(d_u - r_l) \tag{2}$$

since $d_{\Omega_l^u} = d_u$, and $r_{\Omega_l^u} = r_l$. We now handle the right-hand side of (1) and show that

$$\text{rest}(\Omega_L^u, c_i) \geq e_{\Omega_L^u} - (C - c_i)(d_u - r_L). \tag{3}$$

If $\Omega_L^u \neq \emptyset$, $\text{rest}(\Omega_L^u, c_i) = e_{\Omega_L^u} - (C - c_i)(d_{\Omega_L^u} - r_{\Omega_L^u})$ and the result follows since $d_{\Omega_L^u} \leq d_u$ and $r_{\Omega_L^u} \geq r_L$. If $\Omega_L^u = \emptyset$, $\text{rest}(\Omega_L^u, c_i) = 0$ by definition and $e_{\Omega_L^u} = 0$. To show (3), we must prove that $d_u > r_L$. The inequality (1) then becomes

$$c_i r_l + e_{\Omega_l^u} - (C - c_i)(d_u - r_l) > c_i r_L.$$

By E-feasibility of Ω_l^u, $C(d_u - r_l) \geq e_{\Omega_l^u}$. These two last inequalities show that

$$c_i r_l + c_i(d_u - r_l) > c_i r_L.$$

and thus $d_u > r_L$, establishing (3). We now show that

$$e_\Theta > C(r_L - r_l).$$

Rewriting (1) using (2) and (3) gives

$$\begin{aligned} c_i r_l + e_{\Omega_l^u} - (C - c_i)(d_u - r_l) &> c_i r_L + e_{\Omega_L^u} - (C - c_i)(d_u - r_L) \\ e_{\Omega_l^u} - e_{\Omega_L^u} - C d_u + C d_u &> (C - c_i)(r_L - r_l) + c_i(r_L - r_l) \\ e_\Theta &> C(r_L - r_l). \end{aligned}$$

Finally, it remains to show that $\alpha(\Omega_l^U, i)$ holds. Since $\Theta \cap \Omega_L^u = \emptyset$, $\Theta \subseteq \Omega_l^u$, and $d_u \leq d_U$, we have that $\Theta \cap \Omega_L^U = \emptyset$ and $\Theta \cup \Omega_L^U \subseteq \Omega_l^U$. Hence,

$$\begin{array}{rcl} e_{\Omega_l^U} & = & e_\Theta + e_{\Omega_L^U} \\ e_{\Omega_l^U} & > & C(r_L - r_l) + e_{\Omega_L^U} \end{array}$$

and thus

$$C r_l + e_{\Omega_l^U} > C r_L + e_{\Omega_L^U}.$$

Since $\alpha(\Omega_l^U, i)$ holds, we have

$$\begin{array}{ll} C(d_{\Omega_L^U} - r_{\Omega_L^U \cup \{i\}}) < e_{\Omega_L^U \cup \{i\}} & \\ C(d_U - r_L) < e_{\Omega_L^U \cup \{i\}} & \text{since } U = d_{\Omega_L^U} \text{ \& } r_L = r_{\Omega_L^U \cup \{i\}} \\ C(d_U - r_L) < e_{\Omega_L^U} + e_i & \text{since } d_U < d_i \\ C(d_U - r_l) < e_{\Omega_l^U} + e_i & \text{since } C r_l + e_{\Omega_l^U} > C r_L + e_{\Omega_L^U}. \end{array}$$

Since $d_U \geq d_{\Omega_l^U}$ and $r_l \leq r_{\Omega_l^U \cup \{i\}}$, it follows that

$$C(d_{\Omega_l^U} - r_{\Omega_l^U \cup \{i\}}) < e_{\Omega_l^U \cup \{i\}}$$

and $\alpha(\Omega_l^U, i)$ holds. As a consequence,

$$LB_2(i) \geq r_l + \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i)$$

and the result $LB_2(i) \geq LB_2'(u)$ follows from the properties of ceil.

Assumption 2: Consider the case

$$r_l + \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) \leq r_L + \frac{1}{c_i} \text{rest}(\Omega_L^u, c_i).$$

If $\text{rest}(\Omega_L^u, c_i) \leq 0$, it follows directly that $r_l + \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) \leq r_L$ and thus that $LB_2'(i) \leq r_L$. But this contradicts our hypothesis $LB_2'(i) > r_i \geq r_L$. Hence $\text{rest}(\Omega_L^u, c_i) > 0$ and, since $\Omega_L^u \subseteq \Omega_L^U$,

$$LB_2(i) \geq r_L + \left\lceil \frac{1}{c_i} \text{rest}(\Omega_L^u, c_i) \right\rceil \geq LB_2'(i). \blacksquare$$

Algorithm 2 E-FEASIBILITY

Require: X is an array of tasks sorted by non-decreasing release dates;
Require: Y is an array of tasks sorted by non-decreasing due dates;
Ensure: returns true iff the instance is E-feasible;

for $y \leftarrow 1$ **to** n **do**
    $D \leftarrow d_{Y[y]}$
    $e \leftarrow 0$
    for $x \leftarrow n$ **downto** 1 **do**
        if $d_{X[x]} \leq D$ **then**
            $e \leftarrow e + e_{X[x]}$
            if $C \cdot (D - r_{X[x]}) < e$ **then**
                return false;
            end if
        end if
    end for
end for
return true;

5 Testing E-Feasibility

This section presents the standard algorithm for testing E-feasibility [Nui94]. The algorithm only considers task intervals and uses two arrays of tasks: an array X where the tasks are sorted by non-decreasing release dates and an array Y where the tasks are sorted by non-decreasing due dates. Because several tasks may have the same release dates or the same due dates, the algorithm works in fact with pseudo task intervals expressed in terms of the indices of the tasks in the arrays. More precisely, the pseudo task intervals are defined as

$$\widetilde{\Omega}_x^y = \left\{ X[j] \;\mid\; x \leq j \leq n \;\&\; d_{X[j]} \leq d_{Y[y]} \right\}$$

Note that $\widetilde{\Omega}_x^y \subseteq \Omega_{X[x]}^{Y[y]}$ and $\widetilde{\Omega}_x^y = \Omega_{X[x]}^{Y[y]}$when $x = 1$ or $r_{X[x]} > r_{X[x-1]}$. The key insight underlying the algorithm is to amortize the energy computation by using an inner-loop on the release dates, iterating down from the largest release date to the smallest release date. The algorithm is depicted in Algorithm 2 and its correctness follows from Proposition 1.

6 The Edge-Finding Algorithm

A simple use of the dominance relations leads to an $O(n^5)$ edge finder by exploring all tuples (i, L, U, l, u). However, the inner maximization of

$$r_\Theta + \left\lceil \frac{1}{c_i} \operatorname{rest}(\Theta, c_i) \right\rceil .$$

does not depend on Ω, except for the fact that $\Theta \subseteq \Omega$ or, more precisely, its relaxation $d_u \leq d_U$ due to Proposition 7. As a consequence, the loops on l and u may be outside the loops on L and U, reducing the runtime complexity. The new edge-finding algorithm is thus organized in two phases. The first phase uses dynamic programming to precompute the inner maximizations, while the second phase computes the updates using the precomputed results. We start by presenting the precomputation.

6.1 The Precomputation

The precomputation performs the inner maximization in LB_2', i.e.,

$$\max_{\substack{l,u \in T \\ d_u \leq d_U \\ \text{rest}(\Omega_l^u, c) > 0}} r_{\Omega_l^u} + \left\lceil \frac{1}{c} \text{rest}(\Omega_l^u, c) \right\rceil$$

for all $c \in Sc$ and $U \in T$. Once again, in practice, the algorithm works with pseudo task intervals and computes the values $R[c, y]$ defined as

$$R[c, y] = \max_{\substack{l,u \in T \\ d_u \leq d_{Y[y]} \\ \text{rest}(\Omega_l^u, c) > 0}} r_{\Omega_l^u} + \left\lceil \frac{1}{c} \text{rest}(\Omega_l^u, c) \right\rceil .$$

To obtain $R[c, y]$, the algorithm computes the values

$$RT[c, x, y] = \max_{\substack{x \leq x' \ \& \ y' \leq y \\ \text{rest}(\widetilde{\Omega}_{x'}^{y'}, c) > 0}} r_{x'} + \left\lceil \frac{1}{c} \text{rest}(\widetilde{\Omega}_{x'}^{y'}, c) \right\rceil .$$

and we have that $R[c, y] = RT[c, x, y]$. The RT values can be computed by the following recurrence relation.

Proposition 8 *Let $RT[c, x, 0] = -\infty$ $(x \in N)$ and $RT[c, n + 1, y] = -\infty$ $(y \in N)$. For $2 \leq x \leq n + 1$ and $0 \leq y \leq n - 1$, we have*

$$RT[c, x - 1, y + 1] = \max \begin{cases} RT[c, x, y + 1] \\ RT[c, x - 1, y] \\ r_{X[x-1]} + \left\lceil \frac{1}{c} f\left(\text{rest}(\widetilde{\Omega}_{x-1}^{y+1}, c)\right) \right\rceil \end{cases}$$

where f is defined by $f(x) = x$ if $x > 0$ and $-\infty$ otherwise.

Proof The base cases correspond to empty sets and are valid. For the inductive case, consider x^* and y^* $(x \leq x^*$ & $y^* \leq y)$ such that

$$RT[c, x - 1, y + 1] = r_{x^*} + \left\lceil \frac{1}{c} \text{rest}(\widetilde{\Omega}_{x^*}^{y^*}, c) \right\rceil .$$

Either $x^* > x - 1$ or $y^* < y$ or $x^* = x - 1 \ \wedge \ y^* = y + 1$. In the first two cases, $RT[c, x - 1, y + 1]$ is correct by induction. The third case is correct by definition of RT. ■

Algorithm 3 depicts a dynamic programming algorithm to compute the R values using the recurrence relation above. The algorithm, for a given c, computes the columns $RT[c, n, y], \ldots, RT[c, 1, y]$ in $O(n^2)$ time and $O(n)$ space. It dynamically computes the energy of task intervals instead of using an $O(n^2)$ array, which is the purpose of lines 8-9.

Algorithm 3 CalcR: Precomputation of the Bounds Updates in $O(n^2k)$ time

Require: X array of task sorted by non-decreasing release date
Require: Y array of task sorted by non-decreasing due date
Ensure: $R[c, y]$ is computed according to its specification

for all $c \in Sc$ **do**
  for all $y \in T$ **do**
    $E[y] \leftarrow 0;$
    $R[c, y] \leftarrow -\infty;$
  end for
  for $x \leftarrow n$ **downto** 1 **do**
    for $y \leftarrow 1$ **to** n **do**
      if $d_{X[x]} \leq d_{Y[y]}$ **then**
        $E[y] \leftarrow E[y] + e_{X[x]};$
      end if
      $a \leftarrow R[c, y];$
      $b \leftarrow R[c, y-1];$
      $rest \leftarrow E[y] - (C-c)(d_{Y_{[y]}} - r_{X[x]});$
      $c \leftarrow$ **if** $rest > 0$ **then** $r_{X[x]} + \frac{1}{c}\lceil rest \rceil$ **else** $-\infty;$
      $R[c, y] \leftarrow \max(a, b, c);$
    end for
  end for
end for

Theorem 2 *Algorithm 3 is correct for E-feasible CRPs.*

Proof Direct consequence of Proposition 8. ■

6.2 The Edge Finding Algorithm

Once the precomputation is available, an $O(n^3)$ algorithm can be easily derived (see Algorithm 4). The key idea is to iterate over all Ls and Us in the definition of LB_2', using the values $R[c, U]$ to update the bounds. The algorithm is a direct implementation of $\overline{LB_2'}$, with lines 7-12 computing the energy $E[x]$ of $\widetilde{\Omega}_{X[x]}^{Y[y]}$.

Theorem 3 *Algorithm 4 is correct for E-feasible CRPs.*

Proof Direct consequence of Theorem 2 and Proposition 7. ■

Algorithm CalcEFI can be improved by using an idea already present in CalcLB. Observe that line 17 in CalcEFI does not depend on x: only the condition in line 15 does. Hence the update in line 17 can be applied if there exists an x satisfying the condition in line 15 (provided that the condition in line 16 also holds) and we do not need to know x explicitly. As a consequence, the loop on x can be removed and replaced by an incremental computation of the condition in line 15 as the loop on i proceeds. More precisely, the idea of algorithm CalcEF, depicted in Algorithm 5, is to maintain the part of the condition which does not depend on i, i.e.,

$$ECF = \max_{x \leq i} \left(E[x] - C(d_{Y[y]} - r_{X[x]})\right)$$

at each iteration of the loop.

Algorithm 4 CalcEFI: An Edge-Finder in $O(n^3)$ Time and $O(nk)$ Space

Require: X array of task sorted by non-increasing release date
Require: Y array of task sorted by non-decreasing due date
Ensure: $LB[i] = \overline{LB}_2(X[i])$ $(1 \leq i \leq n)$

1: $R \leftarrow CalcR()$;
2: **for** $x \leftarrow 1$ **to** n **do**
3:     $LB[x] \leftarrow r_{X[x]}$
4: **end for**
5: **for** $y \leftarrow 1$ **to** $n-1$ **do**
6:     $E \leftarrow 0$;
7:     **for** $x \leftarrow n$ **downto** 1 **do**
8:         **if** $d_{X[x]} \leq d_{Y[y]}$ **then**
9:             $E \leftarrow E + e_{X[x]}$;
10:         **end if**
11:         $E[x] \leftarrow E$;
12:     **end for**
13:     **for** $x \leftarrow 1$ **to** n **do**
14:         **for** $i \leftarrow x$ **to** n **do**
15:             **if** $E[x] + e_{X[i]} > C(d_{Y[y]} - r_{X[x]})$ **then**
16:                 **if** $d_{X[i]} > d_{Y[y]}$ **then**
17:                     $LB[i] \leftarrow \max(LB[i], R[c_{X[i]}, y])$
18:                 **end if**
19:             **end if**
20:         **end for**
21:     **end for**
22: **end for**

Theorem 4 *Algorithm 5 is correct for E-feasible CRPs.*

Proof Consequence of Theorem 3 and the fact that CalcEF maintains the invariant

$$ECF = \max_{x \leq i} \left(E[x] - C(d_{Y[y]} - r_{X[x]})\right)$$

after line 15. ■

6.3 Discussion

It is interesting to mention a couple of properties of CalcEF. The bottleneck of the algorithm is the computation of the R values which takes $O(n^2k)$ time. However, in practice, there is no need to precompute the entire array, since many values $R[c, y]$ may not be needed by the algorithm. A lazy implementation, which computes $R[c, y]$ on demand, runs in time $O(n^2 + \Delta n^2)$, where Δ is the number of distinct capacities required by the set of tasks whose bounds are updated. Worst-case improvements to the algorithm however require a way to compute the R values more efficiently.

The reader may also wonder if the "refinement" of NBLP over CalcLB would transpose to CalcEF. It appears however that NBLP uses another incorrect dominance rule in the computation of the first inner loop of algorithm CalcLB. Indeed, NBLP only considers those Θ that maximize $Cr_\Theta + e_\Theta$, which

Algorithm 5 CALCEF: An Edge-Finder in $O(n^2k)$ Time and $O(nk)$ Space

Require: X array of task sorted by non-increasing release date
Require: Y array of task sorted by non-decreasing due date
Ensure: $LB[i] = \overline{LB_2}(X[i])$ $(1 \leq i \leq n)$

1: $R \leftarrow CalcR()$;
2: **for** $x \leftarrow 1$ **to** n **do**
3:     $LB[x] \leftarrow r_{X[x]}$
4: **end for**
5: **for** $y \leftarrow 1$ **to** n **do**
6:     $E \leftarrow 0$;
7:     **for** $x \leftarrow n$ **downto** 1 **do**
8:         **if** $d_{X[x]} \leq d_{Y[y]}$ **then**
9:             $E \leftarrow E + e_{X[x]}$;
10:         **end if**
11:         $E[x] \leftarrow E$;
12:     **end for**
13:     $CEF \leftarrow -\infty$;
14:     **for** $i \leftarrow 1$ **to** n **do**
15:         $CEF \leftarrow \max(CEF, E[i] - C(d_{Y[y]} - r_{X[i]}))$;
16:         **if** $CEF + e_{X[i]} > 0$ **then**
17:             **if** $d_{X[i]} > d_{Y[y]}$ **then**
18:                 $LB[i] \leftarrow \max(LB[i], R[c_{X[i]}, y])$
19:             **end if**
20:         **end if**
21:     **end for**
22: **end for**

is not valid. As a consequence, there exist instances for which CALCLB returns the correct lower bounds, but not NBLP. Consider the following instance with a resource of capacity 2 and tasks with capacity requirements equal to one.

task	r	d	p
a	0	69	51
b	1	5	4
c	4	6	2

NBLP does not make any update, although $LB_2(a) = 2$. Indeed, when $d_{Y[c]} = 6$ is considered, the release date d_a should be improved with respect to the set $\Omega = \Theta = \{b, c\}$. Instead of that, only $\Omega = \{b, c\}, \Theta = \{c\}$ is considered, due to the test of line 9 as $Cr_{\{b,c\}} + e_{\{b,c\}} = 8$ is smaller than $Cr_{\{c\}} + e_{\{c\}} = 10$.

7 Extended Edge Finding

This section considers the extended edge-finding rule from [Nui94]. Nuijten gives an $O(n^3k)$ algorithm for the extended edge finger and reference [BLPN01] claims the existence of an $O(n^3)$ algorithm but does not give the algorithm. This section proposes an extended edge-finding algorithm that runs $O(n^2k)$ time and $O(nk)$ space.

7.1 The Extended Edge-Finding Rule

Consider a set $\Omega \subseteq T$ and a task $i \in T \setminus \Omega$ such that $r_i \leq r_\Omega \leq r_i + p_i$. This new condition is interesting, since no tasks in Ω can be scheduled in $[r_i, r_\Omega)$. Under these conditions, Nuijten [Nui94] shows that if

$$C(d_\Omega - r_\Omega) < e_\Omega + (r_i + p_i - r_\Omega)c_i$$

then any feasible schedule satisfies

$$s_i \geq r_\Theta + \left\lceil \frac{1}{c_i} \operatorname{rest}(\Theta, c_i) \right\rceil$$

for all $\Theta \subseteq \Omega$ satisfying

$$\operatorname{rest}(\Theta, c_i) > 0.$$

The preconditions can be specified by the property $\beta(\Omega, i)$ defined as

$$\beta(\Omega, i) \iff \begin{cases} r_i \leq r_\Omega \leq r_i + p_i \\ C(d_\Omega - r_\Omega) < e_\Omega + (r_i + p_i - r_\Omega)c_i \end{cases}$$

The following proposition justifies why this rule is called the extended edge-finder.

Proposition 9 $r_i \leq r_\Omega \leq r_i + p_i \ \wedge\ \alpha(\Omega, i) \implies \beta(\Omega, i)$.

Proof Since $r_i \leq r_\Omega$, we have

$$C(d_\Omega - r_{\Omega \cup \{i\}}) = C(d_\Omega - r_\Omega) + C(r_\Omega - r_i).$$

Since $i \notin \Omega$, $e_{\Omega \cup \{i\}} = e_\Omega + p_i c_i$ and, since $\alpha(\Omega, i)$ holds,

$$C(d_\Omega - r_\Omega) + C(r_\Omega - r_i) < e_\Omega + p_i c_i.$$

Since $C \geq c_i$,

$$C(d_\Omega - r_\Omega) + c_i(r_\Omega - r_i) < e_\Omega + p_i c_i$$

and the result follows. ∎

We now specify the extended edge-finder algorithm.

Specification 2 (Extended Edge-Finder) *An extended edge-finder is an algorithm which, given an E-feasible CRP, computes a vector*

$$\left\langle \overline{LB_4}(1), \ldots, \overline{LB_4}(n) \right\rangle$$

where

$$\overline{LB_4}(i) = \max(r_i, LB_2(i), LB_3(i))$$

and

$$LB_3(i) = \max_{\substack{\Omega \subseteq T \\ i \notin \Omega \\ \beta(\Omega, i)}} \ \max_{\substack{\Theta \subseteq \Omega \\ \operatorname{rest}(\Theta, c_i) > 0}} r_\Theta + \left\lceil \frac{1}{c_i} \operatorname{rest}(\Theta, c_i) \right\rceil$$

7.2 Dominance Properties

In general, the dominance properties of the extended edge finder are similar in nature to those of the standard edge finder. In the following, we focus on the differences and define *valid* pairs as before, except that the condition $\alpha(\Omega, i)$ is replaced by $\beta(\Omega, i)$. The first proposition simplifies the definition of $\beta(\Omega, i)$.

Proposition 10 *For any E-feasible CRP,*

$$\beta(\Omega, i) \iff \begin{cases} r_i \leq r_\Omega \\ C(d_\Omega - r_\Omega) < e_\Omega + (r_i + p_i - r_\Omega)c_i \end{cases}$$

Proof We only need to show that the right-hand side implies the left-hand side. If $r_\Omega > r_i + p_i$, then $e_\Omega + (r_i + p_i - r_\Omega)c_i \leq e_\Omega$. Thus $C(d_\Omega - r_\Omega) < e_\Omega$, which contradicts E-feasibility. ■

The following proposition restricts the sets of pairs (Ω, Θ) to consider. These are the same as in the standard case, except that $r_L = r_{\Omega_L^U}$ because of the nature of the extended rule.

Proposition 11 *The computation of $LB_3(i)$ for an E-feasible CRP only needs to consider pairs of the form (Ω_L^U, Ω_l^u) such that $r_L = r_{\Omega_L^U}$, $d_U = d_{\Omega_L^U}$, $d_u = d_{\Omega_l^u} \leq d_U < d_i$ and $r_l = r_{\Omega_l^u} \geq r_L$.*

Proof Similar to the proofs of Propositions 2, 3, and 4. ■

The following proposition is the counterpart of Proposition 7. It refers both to the standard and extended edge finders.

Proposition 12 *Let LB_3' be defined by*

$$LB_3'(i) = \max_{\substack{L, U \in T \\ \beta(\Omega_L^U, i) \\ d_U < d_i \\ r_L = r_{\Omega_L^U} \\ d_U = d_{\Omega_L^U}}} \; \max_{\substack{l, u \in T \\ r_l = r_{\Omega_l^u} \\ d_u = d_{\Omega_l^u} \leq d_U \\ \mathrm{rest}(\Omega_l^u, c_i) > 0}} r_l + \left\lceil \frac{1}{c_i} \mathrm{rest}(\Omega_l^u, c_i) \right\rceil$$

Then, for any E-feasible CRP, $LB_3'(i) \leq \max(r_i, LB_3(i), LB_2(i))$.

Proof The previous propositions claim that

$$LB_2(i) = \max_{\substack{L, U \in T \\ d_U < d_i \\ r_L = r_{\Omega_L^U} \\ d_U = d_{\Omega_L^U} \\ \beta(\Omega_L^U, i)}} \; \max_{\substack{l, u \in T \\ d_u = d_{\Omega_l^u} \leq d_U \\ r_l = r_{\Omega_l^u} \geq r_L \\ \mathrm{rest}(\Omega_l^u, c_i) > 0}} r_l + \left\lceil \frac{1}{c_i} \mathrm{rest}(\Omega_l^u, c_i) \right\rceil$$

It follows that $LB_3(i) \leq LB_3'(i)$. Moreover, it is sufficient to consider the case where $LB_3'(i) > r_i$ and to show that $\max(LB_2(i), LB_3) \geq LB_3'(i)$. Suppose that $LB_3'(i) > r_i$.

Let $L, U, l, u \in T$ satisfying:

$$\begin{cases} r_L = r_{\Omega_L^U} \\ d_{\Omega_L^U} = d_U < d_i \\ \beta(\Omega_L^U, i) \\ r_{\Omega_l^u} = r_l \\ d_u = d_{\Omega_l^u} \leq d_U \\ \text{rest}(\Omega_l^u, c_i) > 0 \\ LB_3'(i) = r_l + \left\lceil \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) \right\rceil \end{cases}$$

If $r_l \geq r_L$, (Ω_L^U, Ω_l^u) is a maximal valid pair and $LB_3(i) \geq LB_3'(i)$. Now suppose that $r_l < r_L$. As in Proposition 7, partition Ω_l^u in $\Theta \cup \Omega_L^u$, with $\Theta = \Omega_l^u \setminus \Omega_L^u$.

Assumption 1: Assume first that

$$r_l + \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) > r_L + \frac{1}{c_i} \text{rest}(\Omega_L^u, c_i)$$

which implies $e_\Theta > C(r_L - r_l)$. Now we have two cases.

case $\mathbf{r_l \geq r_i}$. We show that $LB_3(i) \geq LB_3'(i)$. Since $r_l = r_{\Omega_l^u}$, $d_u \leq d_U$, and $r_l < r_L$,

$$e_{\Omega_l^U} + c_i(r_i + p_i - r_{\Omega_l^U}) \geq e_{\Omega_l^U} + c_i(r_i + p_i - r_L).$$

Since $\Theta \cap \Omega_L^U = \emptyset$, $e_{\Omega_l^U} = e_\Theta + e_{\Omega_L^U}$ and

$$e_{\Omega_l^U} + c_i(r_i + p_i - r_{\Omega_l^U}) \geq e_\Theta + e_{\Omega_L^U} + c_i(r_i + p_i - r_L).$$

Since $\beta(\Omega_L^U, i)$ holds and $r_L = r_{\Omega_L^U}$, we have

$$e_{\Omega_l^U} + c_i(r_i + p_i - r_{\Omega_l^U}) > C(d_U - r_L) + e_\Theta$$

which implies by $e_\Theta > C(r_L - r_l)$ that

$$e_{\Omega_l^U} + c_i(r_i + p_i - r_{\Omega_l^U}) > C(d_U - r_L) + C(r_L - r_l).$$

Since $r_l = r_{\Omega_l^u}$ and $d_u \leq d_U$, we have $r_l = r_{\Omega_l^U}$ and thus

$$e_{\Omega_l^U} + c_i(r_i + p_i - r_{\Omega_l^U}) > C(d_U - r_{\Omega_l^U})$$

which implies $\beta(\Omega_l^U, i)$.

case $\mathbf{r_l < r_i}$. We show that $LB_2(i) \geq LB_3'(i)$. Since $e_{\Omega_l^U} = e_\Theta + e_{\Omega_L^U}$,

$$e_{\Omega_l^U} + e_i = e_\Theta + e_{\Omega_L^U} + e_i$$

and, since $e_\Theta > C(r_L - r_l)$, $\beta(\Omega_L^U, i)$ holds, and $e_i = p_i c_i$, we have

$$\begin{aligned} &e_{\Omega_l^U} + e_i > C(r_L - r_l) + C(d_U - r_L) - c_i(r_i + p_i - r_L) + p_i c_i \\ &e_{\Omega_l^U} + e_i > C(d_U - r_l) + c_i(r_L - r_i) \\ &e_{\Omega_l^U} + e_i > C(d_U - r_l) \end{aligned}$$

which implies $\alpha(\Omega_l^U, i)$.

Assumption 2: It remains to consider the case

$$r_l + \frac{1}{c_i} \text{rest}(\Omega_l^u, c_i) \leq r_L + \frac{1}{c_i} \text{rest}(\Omega_L^u, c_i),$$

which is similar to the same case in Proposition 7. ■

Corollary 1 *For any E-feasible CRP, we have*

$$\overline{LB_4}(i) = \max(r_i, LB_2'(i), LB_3'(i))$$

Algorithm 6 CALCEEFI: An Extended Edge-Finder in $O(n^3)$ Time.

Require: X array of task sorted by non-increasing release date
Require: Y array of task sorted by non-decreasing due date
Ensure: $LB[i] = \overline{LB_4}(X[i])$ $(1 \le i \le n)$

1: CALCEF();
2: **for** $y \leftarrow 1$ **to** $n-1$ **do**
3:   $E \leftarrow 0$;
4:   **for** $x \leftarrow n$ **downto** 1 **do**
5:     **if** $d_{X[x]} \le d_{Y[y]}$ **then**
6:       $E \leftarrow E + e_{X[x]}$;
7:     **end if**
8:     $E[x] \leftarrow E$;
9:   **end for**
10:   **for** $x \leftarrow 1$ **to** n **do**
11:     **for** $i \leftarrow 1$ **to** x **do**
12:       **if** $E[x] + c_{X[i]}(r_{X[i]} + p_{X[i]} - r_{X[x]}) > C(d_{Y[y]} - r_{X[x]})$ **then**
13:         **if** $d_{X[i]} > d_{Y[y]}$ **then**
14:           $LB[i] \leftarrow \max(LB[i], R[c_{X[i]}, y])$
15:         **end if**
16:       **end if**
17:     **end for**
18:   **end for**
19: **end for**

7.3 The Extended Edge-Finding Algorithm

The extended edge-finding algorithm uses the same precomputation as the standard procedure, since the only change is the condition $\beta(\Omega, i)$ which replaces $\alpha(\Omega, i)$. Moreover, it is possible to derive an $O(n^3)$ algorithm CALCEEFI, which is essentially similar to CALCEFI. The only changes are the initialization of the LB values in line 1 by CALCEF, the loop on i that now goes from 1 to x and, of course, the condition $\beta(\Omega, i)$. CALCEEFI is shown in Algorithm 6.

Theorem 5 *Algorithm 6 is correct for E-feasible CRPs.*

Proof Direct consequence of Theorem 2 and Proposition 12. ■

The optimization to move from $O(n^3)$ to $O(n^2k)$ is slightly more complex for the extended edge finder. Once again, observe that line 14 in CALCEEFI does not depend on x: only the condition in line 12 does. Moreover, the condition can be rewritten as

$$(C - c_{X[i]})r_{X[x]} + E[x] - Cd_{Y[y]} > -(c_{X[i]}(r_{X[i]} + p_{X[i]})).$$

It does not matter which x satisfies this test, only that there exists such a value. As a consequence, the algorithm precomputes the expression

$$CEEF[c, i] = \max_{x \ge i} ((C - c)r_{X[x]} + E[x] - Cd_{Y[y]}).$$

Observe that these expressions are precomputed for all capacities, since we do not know in advance the capacities of the tasks the test will be applied to.

Algorithm 7 CalcEEF: An Extended Edge-Finder in $O(n^2k)$ Time.

Require: X array of task sorted by non-increasing release date
Require: Y array of task sorted by non-decreasing due date
Ensure: $LB[i] = \overline{LB_4}(X[i])$ $(1 \leq i \leq n)$

1: CalcEF();
2: **for** $y \leftarrow 1$ **to** $n-1$ **do**
3: $\quad E \leftarrow 0$;
4: $\quad$ **for** $x \leftarrow n$ **downto** 1 **do**
5: $\quad\quad$ **if** $d_{X[x]} \leq d_{Y[y]}$ **then**
6: $\quad\quad\quad E \leftarrow E + e_{X[x]}$;
7: $\quad\quad$ **end if**
8: $\quad\quad E[x] \leftarrow E$;
9: $\quad$ **end for**
10: $\quad$ **for** $x \leftarrow 1$ **to** $n-1$ **do**
11: $\quad\quad$ **if** $r_{X[x]} = r_{X[x+1]}$ **then**
12: $\quad\quad\quad E[x+1] \leftarrow E[x]$;
13: $\quad\quad$ **end if**
14: $\quad$ **end for**
15: $\quad$ **for all** $c \in Sc$ **do**
16: $\quad\quad CEEF[c, n+1] \leftarrow \infty$;
17: $\quad$ **end for**
18: $\quad$ **for** $x \leftarrow n$ **downto** 1 **do**
19: $\quad\quad$ **for all** $c \in Sc$ **do**
20: $\quad\quad\quad CEEF[c, x] \leftarrow \max(CEEF[c, x+1], (C-c)r_{X[x]} + E[x] - Cd_{Y[y]})$;
21: $\quad\quad$ **end for**
22: $\quad$ **end for**
23: $\quad$ **for** $i \leftarrow 1$ **to** n **do**
24: $\quad\quad$ **if** $CEEF[c_{X[i]}, i] + c_{X[i]}(r_{X[i]} + p_{X[i]}) > 0$ **then**
25: $\quad\quad\quad$ **if** $d_{X[i]} > d_{Y[y]}$ **then**
26: $\quad\quad\quad\quad LB[i] \leftarrow \max(LB[i], R[c_{X[i]}, y])$
27: $\quad\quad\quad$ **end if**
28: $\quad\quad$ **end if**
29: $\quad$ **end for**
30: **end for**

Hence it necessary to compute them prior to the loop instead of incrementally as in CalcEF. The resulting edge finder CalcEEF is shown in Algorithm 7. Observe lines 10-13 which establish the correspondence between $\widetilde{\Omega}_x^y$ and $\Omega_{X[x]}^{Y[y]}$ by ensuring that

$$E[x] = \max\{\ E[j] \ \mid\ r_{X[j]} = r_{X[x]}\}.$$

These lines are not necessary in CalcEF since its loops scan array X from 1 to n contrary to the loop in lines 18-20.

Theorem 6 *Algorithm 5 is correct for E-feasible CRPs.*

Proof Consequence of Theorem 5 and the correctness of the $CEEF[c, x]$ values which satisfy the specification

$$CEEF[c, i] = \max_{x \geq i} ((C-c)r_{X[x]} + E[x] - Cd_{Y[y]}). \blacksquare$$

8 Conclusion

This paper reconsidered edge-finding algorithms for cumulative scheduling. These algorithms are at the core of constraint-based schedulers and update the earliest starting dates and latest finishing dates of tasks that must be scheduled after or before a set of other tasks. The paper made three contributions. First, it indicated that Nuijten's algorithm, and its derivatives, are incomplete because they use an invalid dominance rule inherited from disjunctive scheduling. Second, the paper presented a novel edge-finding algorithm for cumulative resources which runs in time $O(n^2k)$, where n is the number of tasks and k the number of different capacity requirements of the tasks. The key design decision is to organize the algorithm in two phases: The first phase uses dynamic programming to precompute the innermost maximization in the edge-finder specification, while the second phase performs the updates based on the precomputation. Finally, the paper proposed the first extended edge-finding algorithms that run in time $O(n^2k)$, improving on the running time of existing algorithms.

References

[BLPN01] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling.* Kluwer Academic Publishers, 2001.

[CL94] Y. Caseau and F. Laburthe. Improving CLP Scheduling with Task Intervals. In *Proceedings of the 11th International Conference on Logic Programming (ICLP'94)*, pages 369–383, Santa Margherita Ligure, Italy, 1994.

[CP94] J. Carlier and E. Pinson. Adjustment of Heads and Tails for the Job-shop Problem. *European Journal of Operational Research*, 78:146–161, 1994.

[NA96] W. Nuijten and E. Aarts. A Computational Study of Constraint Satisfaction for Multiple Capacitated Job Shop Scheduling. *European Journal of Operational Research*, 90(2):269–284, 1996.

[Nui94] W. Nuijten. *Time and Resource Constrained Scheduling: A Constraint Satisfaction Approach.* PhD thesis, Eindhoven University of Technology, 1994.

[Vil04] Petr Vilim. O(n log n) Filtering Algorithms for Unary Resource Constraint. In *Proceedings of the First International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'04)*, pages 319–334, Nice, 2004.