Errata: A polynomial time repeated cuts algorithm for the time cost tradeoff problem: The linear and convex crashing cost deadline problem

Dorit S. Hochbaum

June 6, 2023

Abstract

This is to address an error, in the use of the scaling version of PD-algorithm, in [Hochbaum (2016)]. The algorithm calls at each iteration for the PD-algorithm that uses only a minimum cut procedure. We present here another version of a scaling algorithm, based on the proximity-scaling algorithm of Hochbaum and Shanthikumar. This proximity-scaling algorithm solves the time-cost tradeoff problem using the PD algorithm at each iteration, resulting in a polynomial time scaling version of the PD-algorithm.

1 The error

In [Hochbaum (2016)] it was shown that the PD-algorithm does not run in polynomial time. However, it has a scaling implementation that does run in polynomial time. The specific *scaling repeated cuts* algorithm given in [Hochbaum (2016)] is incorrect. It uses at each scaling step the PD-algorithm and at the end of the scaling stage, it continues from the solution attained to the next scaling stage. This is incorrect as has been demonstrated by Megow and Simon [Megow and Simon (2019)] via a counter example depicted in Figure 1:

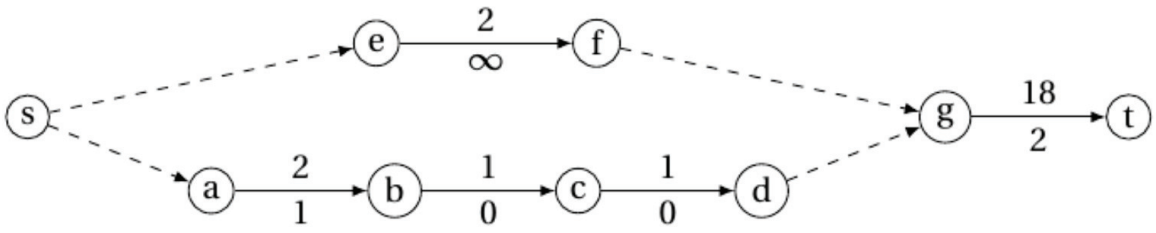

Figure 1: The label above each arc represents its normal duration, and the label below the arc is the cost per unit expediting. The initial normal project duration is 22, and the target duration is $T^* = 2$.

For the project in Figure 1, the scaling repeated cuts algorithm would first take $\Delta = \frac{20}{2*5} = 2$. The minimum cut in the respective crashing graph contains the arc (a, b), as the path containing

(s, g) is not 2-critical. So the algorithm reduces the duration (a, b) by 2 units at a cost of 2. In the following steps, it reduces the arc (g, t) at a cost of 36. In the subsequent iteration, the two paths from s to g are both critical and thus reducing the durations of (b, c) and (c, d) would not reduce the project duration since arc (e, f) is critical. This leads to a solution in which arc (a, b) is expedited by 2 units instead of expediting (b, c) and (c, d) each by 1 unit at a cheaper cost. One reason for this error is that the scaling repeated cuts algorithm *continues* with the solution from a prior scaling stage with the solution obtained, even if there should have been backing up on some of the expediting.

We show here a scaling algorithm based on the proven properties of the proximity-scaling algorithm of [Hochbaum and Shanthikumar (1990)]. Like the scaling repeated cuts algorithm it makes calls for each scaling value to PD-algorithm. However, the solution obtained is used to update the interval of durations' lower and upper bounds for each activity, which is guaranteed to contain the optimal solution. With the algorithm presented here the complexity of the scaling algorithm, called here SCALING PD-ALGORITHM, is $O(n^2 T(|V|, |A|)$ where $T(|V|, |A|)$ is the complexity of solving the minimum cut problem on graph $G = (V, A)$ for n the number of activity arcs. In the case of the project network, both $|V|$ and $|A|$ are $O(n)$.

2 Problem formulation

Let $G = (V, A)$ be the project network, t_i be the start time of event i, and x_{ij} the duration of activity (i, j). Let $c_{ij}(d_{ij} - x_{ij})$ be the convex cost function associated with expediting the activity by $d_{ij} - x_{ij}$.

A project network is a DAG (Directed Acyclic Graph) $G = (V, A)$ with start and finish nodes s and f. In the AOA (Activity On Arc) representation of project network the arcs of the network are associated with activities or precedence relations. Each activity $(i, j) \in A$ has normal duration d_{ij}, minimum duration $\underline{d}_{ij}$ and expediting cost $c_{ij}()$ which is a convex function of the duration x_{ij} (or the amount of reduction from the normal duration $d_{ij} - x_{ij}$). A precedence arc has 0 duration and expediting cost ∞. The "normal" finish time, or makespan, of the project, T^0, is the length of the *longest* path from s to f, which is easily computed using a dynamic programming procedure known as CPM (Critical Path Method). We address the minimum cost expediting of activities so the project finish time is reduced to $T^* < T^0$. This problem is also known as the *Time Cost Tradeoff* problem, TCT. Using the notation x_{ij} for the expedited duration of activity (i, j), and t_i the start time of event i, associated with node i, the TCT problem with target makespan T^* is formulated as the following linear program:

$$
\begin{array}{lll}
 & \min & \sum_{(i,j)\in A} c_{ij}(x_{ij}) \\
 & \text{subject to} & t_j - t_i \geq x_{ij} \quad \text{for all} \quad (i,j) \in A \\
(\mathsf{TCT}) & & t_s = 0 \\
 & & t_f \leq T^* \\
 & & \underline{d}_{ij} \leq x_{ij} \leq d_{ij} \text{ for all} \quad (i,j) \in A.
\end{array}
$$

Without loss of generality, all durations' upper bounds and lower bounds are integers, and the optimal solution has integer durations. Since the constraint coefficients matrix is totally unimodular, then for linear objective function this problem can be solved as linear programming where an optimal basic solution is integer. [Hochbaum and Shanthikumar (1990)] showed that any convex

separable objective minimization over totally unimodular constraint matrix, and more generally when the constraint matrix has maximum subdeterminant bounded by a polynomial in the size of the data, is solved in polynomial time by the *proximity-scaling* algorithm. The algorithm requires to solved at each iteration a scaled version of the problem. It follows that the TCT problem is solved in polynomial time.

Let the largest range of a duration variable be $U = \max_{(i,j)\in A}\{d_{ij} - \underline{d}_{ij}\}$. The proximity-scaling algorithm of [Hochbaum and Shanthikumar (1990)] works by solving in each iteration a scaled version of the problem where each duration range is at most $O(n)$ scaled units long, and deduces from that solution an updated maximum range whose length is reduced by a factor of 2 as compared to the previous iteration. When the largest range is at most $O(n)$ then the problem is solved, in this case, with the PD-algorithm, to attain the optimal solution to TCT.

We show here how to apply the proximity-scaling algorithm of [Hochbaum and Shanthikumar (1990)] where the PD-algorithm solves the problem at each scaling iteration, the Δ-stage, resulting in a polynomial time algorithm that utilizes only a minimum cut procedure.

2.1 The Δ-scaling problem

For a scaling unit Δ, let the function $c_{ij}^{\Delta}(d_{ij} - x_{ij})$ be the piecewise linear convex function that coincides with the function $c_{ij}()$ on integer multiples of Δ.

In the Δ-stage the durations of the activities are in the range $[\underline{d}_{ij}(\Delta), d_{ij}(\Delta)]$, where $d_{ij}(\Delta) = \lceil \frac{d_{ij}}{\Delta} \rceil \cdot \Delta$ and $\underline{d}_{ij}(\Delta) = \lfloor \frac{d_{ij}}{\Delta} \rfloor \cdot \Delta$. Since this range contains the original range $[\underline{d}_{ij}, d_{ij}]$ the convex function $c_{ij}()$ is extended to the function $c_{ij}^{\Delta}(x_{ij})$ defined for values of x_{ij} in this range that are integer multiples of Δ:

$$c_{ij}^{\Delta}(x_{ij}) = \begin{cases} 0 & \text{if } \ d_{ij} \le x_{ij} \le d_{ij}(\Delta) \\ c_{ij}(x_{ij}) & \text{if } \ \underline{d}_{ij} \le x_{ij} \le d_{ij} \\ M \cdot (\underline{d}_{ij} - x_{ij}) & \text{if } \ \underline{d}_{ij}(\Delta) \le x_{ij} < \underline{d}_{ij} \end{cases}$$

The quantity M must be large enough, e.g. it suffices to choose $M = \sum_{(i,j)\in A} c'_{ij}(\underline{d}_{ij})$, where $c'_{ij}()$ is the absolute value of the derivative, or left subgradient, of the function $c_{ij}()$. An illustration of the function $c_{ij}^{\Delta}(x_{ij})$ is given in Figure 2.

Since there is a feasible solution to TCT with project finish time equal to T^*, there is a feasible solution to TCT with activity durations in the interval $[\underline{d}_{ij}(\Delta), d_{ij}(\Delta)]$, that achieves project finish time $T^*(\Delta) = \lfloor \frac{T^*}{\Delta} \rfloor \cdot \Delta$. The Δ-scaling problem, TCT(Δ), is then:

$$(\mathsf{TCT}(\Delta)) \qquad \begin{array}{ll} \min & \sum_{(i,j)\in A} c_{ij}^{\Delta}(d_{ij} - x_{ij}) \\ \text{subject to} & t_j - t_i \ge x_{ij} \ \text{ for all } \ (i,j) \in A \\ & t_s = 0 \\ & t_f \le T^*(\Delta) \\ & \underline{d}_{ij}(\Delta) \le x_{ij} \le d_{ij}(\Delta) \text{ for all } \ (i,j) \in A. \end{array}$$

If TCT(Δ) is feasible then there is an optimal solution which is an integer multiple of Δ. This is because the constraint matrix is totally unimodular and the right hand sides are integer mutliples of Δ. This is so even for the convex objective function since it is piecewise linear with breakpoints on the Δ-grid.

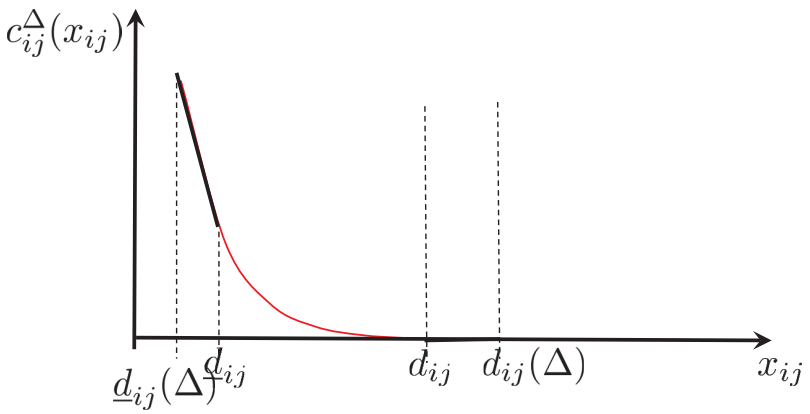

Figure 2: The function $c_{ij}^{\Delta}(x_{ij})$ is an extension of function $c_{ij}(x_{ij})$.

Denote the optimal solution to TCT by $\mathbf{x}^*$ and an optimal solution to TCT(Δ) by $\mathbf{x}^{\Delta}$. The **proximity theorem** of [Hochbaum and Shanthikumar (1990)] states that,

$$||\mathbf{x}^* - \mathbf{x}^{\Delta}||_{\infty} \leq n\Delta.$$

The scaling procedure is initialized with $\Delta = \lceil \frac{U}{4n} \rceil$, maintaining the variant that at each iteration the maximum range of the duration variables is at most $4n$ units of Δ. From the proximity theorem, once the Δ-scaling problem is solved, the optimal duration of each variable, x_{ij}^*, falls in the range $[\max\{\underline{d}_{ij}(\Delta), x_{ij}^{\Delta} - n\Delta\}, \min\{d_{ij}(\Delta), x_{ij}^{\Delta} + n\Delta\}]$, resulting in a maximum range which is at most $2n$ units of Δ, or $4n$ units of $\frac{\Delta}{2}$. Therefore the length of the maximum range goes down by a factor of 2 at least. After $\log \frac{U}{n}$ iterations Δ becomes $O(1)$ and the range of each variable is at most $O(n)$ and then the problem can be solved with PD-algorithm reducing the project duration one unit at a time to achieve the optimal solution $\mathbf{x}^*$.

2.2 Solving the Δ-scaling problem TCT(Δ)

We describe here how to solve TCT(Δ) with PD algorithm which decreases the project finish time by Δ units at each iteration. We denote this adaptation of PD by PD(Δ).

The input to PD(Δ) is the project graph $G = (V, A)$ and the durations' lower bounds and upper bounds, $\underline{d}_{ij}(\Delta)$ and $d_{ij}(\Delta)$ for each arc $(i, j) \in A$. Let the project earliest finish time corresponding to the durations upper bounds $\mathbf{d}(\Delta)$, be $T(\mathbf{d}(\Delta))$, and let the respective set of critical arcs be $A(\mathbf{d}(\Delta))$.

For a vector of durations $\mathbf{x}$ that are integer multiples of Δ, we define the Δ-crashing graph on the set of the critical arcs $A(\mathbf{x})$, $G^{\Delta}(\mathbf{x}) = (V, A(\mathbf{x}))$. The capacity lower and upper bounds of the arcs of the Δ-crashing graph with respect to durations vector $\mathbf{x}$ use the quantities $c_{ij}^{+}(x_{ij})$ and $c_{ij}^{-}(x_{ij})$ defined as:

For $x_{ij} < d_{ij}(\Delta)$, $c_{ij}^{+}(x_{ij}) = c_{ij}(x_{ij}) - c_{ij}(x_{ij} + \Delta)$;
For $x_{ij} > \underline{d}_{ij}(\Delta)$, $c_{ij}^{-}(x_{ij}) = c_{ij}(x_{ij} - \Delta) - c_{ij}(x_{ij})$.

With this definition the capacity lower and upper bounds of the arcs of the Δ-crashing graph with respect to durations vector $\mathbf{x}$ are:

$$(\ell_{ij}(x_{ij}), u_{ij}(x_{ij})) = \begin{cases} (0, c_{ij}^{-}(x_{ij})) & \text{if } \underline{d}_{ij}(\Delta) < x_{ij} = d_{ij}(\Delta) \\ (c_{ij}^{+}(x_{ij}), c_{ij}^{-}(x_{ij})) & \text{if } \underline{d}_{ij}(\Delta) < x_{ij} < d_{ij}(\Delta) \\ (c_{ij}^{+}(x_{ij}), \infty) & \text{if } \underline{d}_{ij}(\Delta) = x_{ij} < d_{ij}(\Delta) \\ (0, \infty) & \text{if } \underline{d}_{ij}(\Delta) = x_{ij} = d_{ij}(\Delta). \end{cases}$$

The problem TCT(Δ) is then solved with the PD(Δ)-algorithm:

PROCEDURE PD(Δ) ($G = (V, A)$, $\mathbf{d}(\Delta)$, $\underline{\mathbf{d}}(\Delta)$, $T^*(\Delta)$).

Step 0 Let $\mathbf{x} := \mathbf{d}(\Delta)$; $\bar{T} = T(\mathbf{d}(\Delta))$

Step 1 Find the critical activities $A(\mathbf{x})$ in $G_{\mathbf{x}}$;

Step 2 Construct the Δ-crashing graph $G^{\Delta}(\mathbf{x})$ with capacity lower and upper bounds $(\ell_{ij}(\mathbf{x}), u_{ij}(\mathbf{x}))$;

Step 3 Find a minimum s,t-cut in $G^{\Delta}(\mathbf{x})$, (S, T). Update $\mathbf{x}$ by setting:

$$x_{ij} := \begin{cases} x_{ij} - \Delta & \text{if } i \in S, j \in T \\ x_{ij} + \Delta & \text{if } i \in T, j \in S \text{ and } \ell_{ij}(x_{ij}) = c_{ij}^{+}(x_{ij}) > 0; \end{cases}$$

Set $\bar{T} := \bar{T} - \Delta$.

Step 4 If $\bar{T} = T^*(\Delta)$ terminate, output $\mathbf{x}^{\Delta} = \mathbf{x}$. Else, go to Step 1.

Since the duration of each activity $\mathbf{x}_{ij}^{\Delta}$ may take any of the $4n$ integer mutliples of Δ in the range $[\underline{\mathbf{d}}_{ij}(\Delta), \mathbf{d}_{ij}(\Delta)]$, it is possible that PD(Δ)-algorithm would reduce, at each iteration, one unit of Δ from only one activity, resulting in at most $O(n^2)$ iterations, or calls to a minimum cut procedure. Therefore the complexity of PD(Δ)-algorithm is $O(n^2 T(|V|, |A|))$, where $T(|V|, |A|)$ is the complexity of minimum cut on a graph with $|V|$ nodes and $|A|$ arcs. This proves that,

Lemma 2.1 *The complexity of the Δ-stage is $O(n^2 T(|V|, |A|))$.*

2.3 The proximity-scaling algorithm: Scaling PD-algorithm

Let $U = \max_{(i,j)\in A}\{d_{ij} - \underline{d}_{ij}\}$, and $p = \lceil \log \frac{U}{4n} \rceil$.

SCALING PD-ALGORITHM ($G = (V, A)$, $\mathbf{d}$, $\underline{\mathbf{d}}$, T^*, p).

Step 0 $\Delta = 2^p$. Set $d_{ij}(\Delta) = \lceil \frac{d_{ij}}{\Delta} \rceil \cdot \Delta$; $\underline{d}_{ij}(\Delta) = \lfloor \frac{d_{ij}}{\Delta} \rfloor \cdot \Delta$; $T^*(\Delta)$.

Step 1 Call PROCEDURE PD(Δ) ($G = (V, A)$, $\mathbf{d}(\Delta)$, $\underline{\mathbf{d}}(\Delta)$, $T^*(\Delta)$). Return output $\mathbf{x}^{\Delta}$.

Step 2 If $\Delta \leq 1$, stop. Output $\mathbf{x}^* = \mathbf{x}^{\Delta}$. Else, continue.

Step 3 {UPDATE } For all arcs $(i, j) \in A$:
$d_{ij}(\frac{\Delta}{2}) := \min\{d_{ij}(\Delta), x_{ij}^{\Delta} + n\Delta\}$
$\underline{d}_{ij}(\frac{\Delta}{2}) := \max\{\underline{d}_{ij}(\Delta), x_{ij}^{\Delta} - n\Delta\}$
Set $T^*(\frac{\Delta}{2}) = \lfloor \frac{T^*}{\Delta} \rfloor \cdot \Delta$.

Step 4 Set $\Delta := \frac{1}{2}\Delta$, $\mathbf{x} = \mathbf{d}(\Delta)$ and go to Step 1.

The Scaling PD-algorithm makes at most $\log \frac{U}{4n}$ calls to the PD(Δ). From Lemma 2.1, each call takes $O(n^2T(|V|, |A|))$ steps. Hence, the total complexity of the procedure is $O(n^2T(|V|, |A|) \log \frac{U}{4n})$. We conclude that,

Theorem 2.1 *For a project network on n activities and maximum duration range U, the Scaling PD-algorithm runs in time $O(n^2T(|V|, |A|) \log \frac{U}{4n})$.*

References

[Hochbaum (2016)] D. S. Hochbaum. A polynomial time repeated cuts algorithm for the time cost tradeoff problem: The linear and convex crashing cost deadline problem. *Computers & Industrial Engineering*, Vol. 95 pp. 64–71, May (2016).

[Hochbaum and Shanthikumar (1990)] D. S. Hochbaum and J. G. Shanthikumar. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM* 37:843–862, (1990).

[Megow and Simon (2019)] N. Megow and B. Simon. Private communication, March (2019).