Simple k-crashing Plan with a Good Approximation Ratio

Extended Abstract

Ruixi Luo
Shenzhen Campus of Sun Yat-sen University, No. 66, Gongchang Road, Guangming District
Shenzhen, China
luorx@mail2.sysu.edu.cn

Kai Jin
Shenzhen Campus of Sun Yat-sen University, No. 66, Gongchang Road, Guangming District
Shenzhen, China
cscjjk@gmail.com

Zelin Ye
Shenzhen Campus of Sun Yat-sen University, No. 66, Gongchang Road, Guangming District
Shenzhen, China
zlrelay@outlook.com

ABSTRACT

A project is considered as an activity-on-edge network (AOE network, which is a directed acyclic graph) N, where each activity / job of the project is an edge. Some jobs must be finished before others can be started, as described by the topology structure of N.

It is known that job j_i in normal speed would take b_i days to be finished after it is started, and hence the (normal) duration of the project N, denoted by $d(N)$, is determined, which equals the length of the critical path (namely, the longest path) of N.

To speed up the project, the manager can crash a few jobs (namely, reduce the length of the corresponding edges) by investing extra resources into that job. However, the time for completing j_i has a lower bound due to technological limits - it requires at least a_i days to be completed. Following the convention, assume that the duration of a job has a linear relation with the extra resources put into this job; equivalently, there is a parameter c_i (slope), so that shortening j_i by d ($0 \leq d \leq b_i - a_i$) days costs $c_i \cdot d$ resources.

Given project N and an integer $k \geq 1$, the *k-crashing problem* asks the minimum cost to speed up the project by k days.

In this paper, we present a simple solution with the approximation ratio $\frac{1}{1} + \ldots + \frac{1}{k}$. For simplicity, we focus on the linear case throughout the paper, but our proofs are still correct for the convex case, where shortening an edge becomes more difficult after a previous shortening.

KEYWORDS

Project duration; Network optimization; Greedy algorithm; Maximum flow; Critical path

ACM Reference Format:

Ruixi Luo, Kai Jin, and Zelin Ye. 2024. Simple k-crashing Plan with a Good Approximation Ratio: Extended Abstract. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

This reasearch was supported by National Natural Science Foundation of China 62002394 and Shenzhen Science and Technology Program (Grant No. 202206193000001, 20220817175048002). Corresponding author: Kai Jin.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 RELATED WORK

The first solution to the k-crashing problem was given by Fulkerson [2] and by Kelley [5] respectively in 1961. The results in these two papers are independent, yet the approaches are essentially the same, as pointed out in [6]. In both of them, the problem is first formulated into a linear program problem, whose dual problem is a minimum-cost flow problem, which can then be solved efficiently.

Later in 1977, Phillips and Dessouky [6] reported another clever approach (denoted by Algorithm PD). Similar as the greedy algorithm mentioned above, Algorithm PD also consists of k steps, and each step it locates a minimal cut in a flow network derived from the original project network. This minimal cut is then utilized to identify the jobs which should be expedite or de-expedite in order to reduce the project reduction. It is however not clear whether this algorithm can always find an optimal solution. We have a tendency to believe the correctness, yet cannot find a proof in [6].

The greedy algorithm we considered is much simpler and easier to implement comparing to all the approaches above.

Other approaches for the problem are proposed by Siemens [7] and Goyal [4], but these are heuristic algorithms without any guarantee – approximation ratio are not proved in these papers.

Many variants of the k-crashing problem have been studied in the past decades; see [3], [1], and the references within.

2 ALGORITHM AND ANALYSIS

The greedy algorithm in the following (see Algorithm 1) finds a k-crashing plan efficiently. It finds the plan incrementally – each time it reduces the duration of the project by 1.

Algorithm 1 Greedy algorithm for finding a k-crashing plan.

Input: A project $N = (V, E)$.
Output: A k-crashing plan G.

$G \leftarrow \varnothing$;
for $i = 1$ to k **do**
    Find the optimum 1-crashing plan of $N(G)$, denoted by A_i;
    $G \leftarrow G \cup A_i$; (regard as multiset union)
end for

Observe that G is an i-crashing plan of network N after the i-th iteration $G \leftarrow G \cup A_i$, as the duration of $N(G)$ is reduced by 1 at each iteration. Therefore, G is a k-crashing plan at the end.

In this paper, we mainly prove that

Theorem 2.1. *Let $G = A_1 \cup \ldots \cup A_k$ be the k-crashing plan found by Algorithm 1. Let* OPT *denote the optimal k-crashing plan. Then,*

$$\mathsf{cost}(G) \leq \sum_{i=1}^{k} \frac{1}{i} \mathsf{cost}(\mathsf{OPT}).$$

By applying the following Lemma 2.2 below in every step of the greedy algorithm, we can directly have the theorem.

Lemma 2.2. *For any project N, its k-crashing plan (where $k \leq k_{max}$) costs at least k times the cost of the optimum 1-crashing plan.*

2.1 Proof of Lemma 2.2

The *critical graph* of network H, denoted by H^*, is formed by all the critical edges of H; all the edges not critical are removed in H^*.

We first have

Proposition 2.3. *A k-crashing plan X of N contains a cut of N^*.*

In the following, suppose X is a k-crashing plan of N. We introduce a decomposition of X which is crucial to our proof.

First, define

$$\begin{cases} N_1 &= N, \\ X_1 &= X, \\ C_1 &= \mathsf{mincut}(N_1^*, X_1). \end{cases} \tag{1}$$

Next, for $1 < i \leq k$, define

$$\begin{cases} N_i &= N_{i-1}^*(C_{i-1}), \\ X_i &= X_{i-1} \setminus C_{i-1}. \\ C_i &= \mathsf{mincut}(N_i^*, X_i). \end{cases} \tag{2}$$

Note that $C_i = \mathsf{mincut}(N_i^*, X_i)$ means C_i is this minimum cut of N_i^* from X_i.

The following lemma easily implies Lemma 2.2.

Lemma 2.4. *$\mathsf{cost}(C_i) \leq \mathsf{cost}(C_{i+1})$ for any i $(1 \leq i < k)$.*

We show how to prove Lemma 2.2 in the following. The proof of Lemma 2.4 will be shown in the next subsection.

Proof of Lemma 2.2. Suppose X is k-crashing to N.

By Lemma 2.4, we know $\mathsf{cost}(C_1) \leq \mathsf{cost}(C_i)$ $(1 \leq i \leq k)$.

Further since $\bigcup_{i=1}^{k} C_i \subseteq X$,

$$k \cdot \mathsf{cost}(C_1) \leq \mathsf{cost}(\bigcup_{i=1}^{k} C_i) \leq \mathsf{cost}(X).$$

Because C_1 is the minimum cut of N^* that is contained in X, whereas A_1 is the minimum cut of N^* among all, $\mathsf{cost}(A_1) \leq \mathsf{cost}(C_1)$. To sum up, we have $k \cdot \mathsf{cost}(A_1) \leq \mathsf{cost}(X)$.

□

2.2 Proof of Lemma 2.4

Assume i $(1 \leq i < k)$ is fixed. In the following we prove that $\mathsf{cost}(C_i) \leq \mathsf{cost}(C_{i+1})$, as stated in Lemma 2.4, which is a core result.

Assume the cut C_i of N_i^* divides the vertices of N_i^* into two parts, U_i, W_i, where $s \in U_i$ and $t \in W_i$. The edges of N_i^* are divided into four parts: 1. S_i – the edges within U_i; 2. T_i – the edges within W_i; 3. C_i – the edges from U_i to W_i; 4. RC_i – the edges from W_i to U_i.

We can prove that

Proposition 2.5. *(1) $C_{i+1} \cap RC_i = \emptyset$ and (2) $C_i \subseteq N_{i+1}^*$.*

Because C_{i+1} is a subset of the edges of N_{i+1}^*, and the edges of N_{i+1}^* are also in N_i^*, we see $C_{i+1} \subseteq T_i \cup S_i \cup C_i \cup RC_i$. Further since $C_{i+1} \cap RC_i = \emptyset$ (proposition 2.5), set C_{i+1} consists of three disjoint parts:

$$\begin{cases} C_{i+1}^+ = C_{i+1} \cap T_i; \\ C_{i+1}^0 = C_{i+1} \cap C_i; \\ C_{i+1}^- = C_{i+1} \cap S_i. \end{cases} \tag{3}$$

Due to $C_i \subseteq N_{i+1}^*$ (proposition 2.5), set C_i consists of four disjoint parts:

$$\begin{cases} C_i^+ = C_i \cap T_{i+1} \\ C_i^0 = C_i \cap C_{i+1} \\ C_i^- = C_i \cap S_{i+1} \\ C_i^R = C_i \cap RC_{i+1} \end{cases} \tag{4}$$

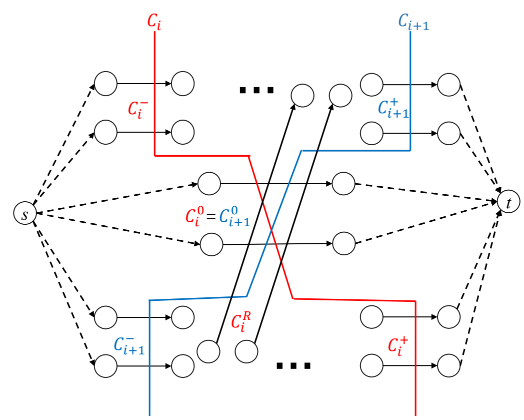

Figure 1: Key notation used in the proof of Lemma 2.4.

Note that $C_i^0 = C_{i+1}^0$, we can prove that

Proposition 2.6.
1. $C_{i+1}^+ \cup C_i^0 \cup C_i^+$ contains a cut of N_i^*.
2. $C_{i+1}^- \cup C_i^0 \cup C_i^-$ contains a cut of N_i^*.

We are ready to prove Lemma 2.4. By proposition 2.6 and $C_i = \mathsf{mincut}(N_i^*, X_i)$, we derive that

$$\mathsf{cost}(C_i) = \mathsf{cost}(C_i^+ \cup C_i^0 \cup C_i^- \cup C_i^R) \leq \mathsf{cost}(C_{i+1}^+ \cup C_i^0 \cup C_i^+)$$
$$\mathsf{cost}(C_i) = \mathsf{cost}(C_i^+ \cup C_i^0 \cup C_i^- \cup C_i^R) \leq \mathsf{cost}(C_{i+1}^- \cup C_i^0 \cup C_i^-)$$

By adding the inequalities above, we obtain Lemma 2.4 $\mathsf{cost}(C_i) \leq \mathsf{cost}(C_{i+1})$, completing the proof.

3 SUMMARY & FUTURE WORK

We have shown that simple greedy algorithms achieve pretty small approximation ratio in k-crashing problems. And the analysis is non-trivial.

Hopefully, the techniques developed in this paper can be used for analyzing greedy algorithms of other related problems.

We would like to end up this paper with one challenging problem: Can we prove a constant approximation ratio for Algorithm 1?

REFERENCES

[1] Pablo Ballesteros-Pérez, Kamel Mohamed Elamrousy, and Mª Carmen González-Cruz. 2019. Non-linear time-cost trade-off models of activity crashing: Application to construction scheduling and project compression with fast-tracking. *Automation in Construction* 97 (2019), 229–240. https://doi.org/10.1016/j.autcon.2018.11.001

[2] Delbert Ray Fulkerson. 1961. A Network Flow Computation for Project Cost Curves. *Management Science* 7, 2 (1961), 167–178. http://www.jstor.org/stable/2627099

[3] JosÉ Eduardo Vinhaes Gerk and Raad Yahya Qassim. 2008. Project Acceleration via Activity Crashing, Overlapping, and Substitution. *IEEE Transactions on Engineering Management* 55, 4 (2008), 590–601. https://doi.org/10.1109/TEM.2008.927786

[4] Suresh K. Goyal. 1996. A simple time-cost tradeoff algorithm. *Production Planning & Control* 7, 1 (1996), 104–106. https://doi.org/10.1080/09537289608930331

[5] James E. Kelley. 1961. Critical-Path Planning and Scheduling: Mathematical Basis. *Operations Research* 9, 3 (1961), 296–320. http://www.jstor.org/stable/167563

[6] Steve Phillips and Mohamed I. Dessouky. 1977. Solving the Project Time/Cost Tradeoff Problem Using the Minimal Cut Concept. *Management Science* 24, 4 (1977), 393–400. http://www.jstor.org/stable/2630261

[7] Nicolai Siemens. 1971. A Simple CPM Time-Cost Tradeoff Algorithm. *Management Science* 17, 6 (1971), B354–B363. http://www.jstor.org/stable/2629138