

Fully Polynomial Time Approximation Schemes for Time-Cost Tradeoff Problems in Series-Parallel Project Networks*

Nir Halman

Department of Civil and Environmental Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139, U.S.A.
and
School of Business Administration
Hebrew University of Jerusalem
Mount Scopus 91905, Israel

Chung-Lun Li[†]

Department of Logistics and Maritime Studies
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong

David Simchi-Levi

Department of Civil and Environmental Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139, U.S.A.

Abstract

We consider the deadline problem and budget problem of the nonlinear time-cost tradeoff project scheduling model in a series-parallel activity network. We develop fully polynomial time approximation schemes for both problems using K -approximation sets and functions, together with series and parallel reductions.

Keywords: Project management; time-cost tradeoff; approximation algorithms

*A preliminary version of this paper appeared in APPROX and RANDOM 2008, Lecture Notes in Computer Science 5171, pp. 91–103, 2008.

[†]Corresponding author: Chung-Lun Li, Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. Email address: lgtcli@polyu.edu.hk

1 Introduction

Project scheduling with time-cost tradeoff decisions plays a significant role in project management. In particular, discrete time-cost tradeoff models with deadline or budget constraints are important tools for project managers to perform time planning and budgeting for their projects. As a result, efficient and effective solution procedures for such models are highly attractive to those practitioners. Unfortunately, these models are computationally intractable, and constructing near-optimal polynomial-time heuristics for them is highly challenging. In this paper, we develop fully polynomial time approximation schemes (FPTASs) for an important class of time-cost tradeoff problems in which the underlying project network is series-parallel (see Section 4 for a discussion of how our results can be applied to problems with “near-series-parallel” networks).

Time-cost tradeoff problems in series-parallel networks have applications not only in project management. Rothfarb *et al.* [11] and Frank *et al.* [5] have applied the time-cost tradeoff model to natural-gas pipeline system design and centralized computer network design, respectively. In their applications, the underlying network is a tree network, which is a special kind of series-parallel network, and they proposed an (exponential time) enumeration method for their problems.

Consider the following time-cost tradeoff model for project scheduling: There is a (directed acyclic) project network of n activities in activity-on-arc representation. Associated with each activity i are two nonincreasing functions $f_i : T_i \rightarrow Z^+$ and $g_i : C_i \rightarrow Z^+$, where $f_i(t_i)$ is the cost incurred when the activity time is t_i , $g_i(c_i)$ is the activity time when an amount c_i of monetary resource is spent on the activity, $T_i = \{\underline{t}_i, \underline{t}_i+1, \dots, \bar{t}_i\} \subset Z^+$ is the set of all possible time duration of activity i , $C_i = \{\underline{c}_i, \underline{c}_i+1, \dots, \bar{c}_i\} \subset Z^+$ is the set of all possible cost consumption of activity i , and Z^+ is the set of all nonnegative integers. In other words, $g_i(c_i) = \min\{t \mid f_i(t) \leq c_i\}$ and $f_i(t_i) = \min\{c \mid g_i(c) \leq t_i\}$. Here, we assume that all activity times and costs are integer-valued.

Denote the activities as $1, 2, \dots, n$. Let $\phi(t_1, t_2, \dots, t_n)$ denote the total duration of the project (i.e., the length of the longest path in the network) when the time duration of activity i is t_i for $i = 1, 2, \dots, n$. We are interested in two different variants of the problem: (i) given a deadline d , determine t_1, t_2, \dots, t_n so that $\phi(t_1, t_2, \dots, t_n) \leq d$ and that $f_1(t_1) + f_2(t_2) + \dots + f_n(t_n)$ is minimized, and (ii) given a budget b , determine c_1, c_2, \dots, c_n so that $c_1 + c_2 + \dots + c_n \leq b$ and that

$\phi(g_1(c_1), g_2(c_2), \dots, g_n(c_n))$ is minimized. We refer to the first problem as the *deadline problem* and the second problem as the *budget problem*. In the deadline problem, we assume, for simplicity, that for each activity i , function f_i can be evaluated in constant time (i.e., for any given $t \in T_i$, $f_i(t)$ can be determined in constant time). In the budget problem, we assume, for simplicity, that for each activity i , function g_i can be evaluated in constant time. However, our FPTASs remain valid as long as f_i and g_i can be evaluated in an amount of time which is polynomial in the input size of the problems.

Note that in our model the time-cost tradeoff function of an activity can be any nonincreasing function (with nonnegative integer domain and range). In fact, our model is a generalization of the traditional *discrete* time-cost tradeoff model, which is defined in such a way that every activity i has $m(i)$ alternatives, of which alternative j requires $t(i, j) \in \mathbb{Z}^+$ time units and $c(i, j) \in \mathbb{Z}^+$ cost units ($j = 1, 2, \dots, m(i)$). In the discrete time-cost tradeoff model, for every activity i , all possible durations are explicitly given such that the encoding length of activity i is linear in the number of possible durations. In other words, in the discrete model functions f and g are specified pointwise, while in our model these functions can be *encoded compactly* via a fast oracle algorithm.

De *et al.* [3] have shown that both the deadline problem and the budget problem are NP-hard in the strong sense for the discrete time-cost tradeoff model when the underlying project network is a general directed acyclic network. Thus, it is unlikely that there exists an FPTAS for either the deadline problem or the budget problem of our model. In fact, developing polynomial-time approximation algorithms for the discrete time-cost tradeoff model is a challenging task. Skutella [13] has developed a polynomial-time algorithm for the budget problem with performance guarantee $O(\log l)$, where l is the ratio of the maximum duration and minimum nonzero duration of any activity. However, as pointed out by Deineko and Woeginger [4], unless $P=NP$, the budget problem does not have a polynomial-time approximation algorithm with performance guarantee strictly less than $\frac{3}{2}$. Skutella [13] has also developed a polynomial-time algorithm for the deadline problem with performance guarantee $O(l)$. However, as pointed out by Grigoriev and Woeginger [6], unless $P=NP$, the deadline problem does not have a polynomial-time approximation algorithm with performance guarantee strictly less than $\frac{7}{6}$. (Note: The deadline problem and the budget problem

are clearly equivalent to each other in terms of polynomial-time solvability. However, these problems may behave differently with respect to their approximability.)

When the underlying network is series-parallel, the time-cost tradeoff problems are more “computationally tractable.” Grigoriev and Woeginger [6] have developed an $O(nd^3)$ algorithm for the deadline problem of the discrete time-cost trade-off model when the underlying network is series-parallel. This also implies that the budget problem of the discrete time-cost trade-off model is polynomial-time solvable when the underlying network is series-parallel. On the other hand, they have given an elegant proof that both the deadline problem and the budget problem are NP-hard in the ordinary sense for the compactly encoded time-cost tradeoff model, even when the project network consists of only two activities that are connected in series.

Although the deadline and budget problems for the compactly encoded time-cost tradeoff model are NP-hard, they can be solved in pseudo-polynomial time by dynamic programming whenever the underlying network is series-parallel [3, 10, 6]. However, to the best of our knowledge, no known polynomial-time approximation scheme has been developed for these problems. Note that Woeginger [15] and Halman *et al.* [7] have developed different frameworks for deriving FPTASs for dynamic programs. Our problems do not fit into either of these frameworks. They do not fit into Woeginger’s framework because his framework requires the cardinality of the action space of the dynamic program to be bounded by a polynomial of the binary input size (see Condition C.4(ii) in [15]). They do not fit into Halman *et al.*’s framework because their framework is presented as a finite-horizon dynamic program, and our problems, if formulated as dynamic programs, do not appear to match the form required by the framework. In addition, Halman *et al.*’s framework does not support a min-max operation that is needed when dealing with the parallel activities in the budget problem. We summarize in Table 1 the past results, as well as our new results, for the deadline problem under different models and underlying networks.

A series-parallel network can be reduced to a single-arc network efficiently via a sequence of simple series and parallel reduction operations [14]. In what follows, we will make use of series and parallel reductions, together with the K -approximation sets and functions introduced by Halman *et al.* [8], to develop FPTASs for the deadline and budget problems in series-parallel networks. To

Table 1: Past and new results of the deadline problem

Underlying network \ Model	Discrete	Compactly encoded
General	Strongly NP-hard [3]; not $< \frac{7}{6}$ -approximable unless P=NP [6]; admits an $O(l)$ -approximation [13]*	Strongly NP-hard [3]
Series-parallel	Solvable in $O(nd^3)$ time [6]	Ordinarily NP-hard [6]; admits an FPTAS (Section 3.1)

*[13] achieved an $O(\log l)$ -approximation for the budget problem

simplify the discussion, we only consider the case where the problem is feasible. Note that it is easy to detect feasibility of the problem. The budget problem is feasible if and only if $\sum_i f_i(\bar{t}_i) \leq b$. The feasibility of the deadline problem can be detected by setting all activity times to their lower limits, solving the problem by the standard critical path method, and comparing the resulting project completion time with the deadline d .

To simplify our analysis, we expand the domains of functions f_i and g_i to $\{0, 1, \dots, U\}$ for each activity i , where $U = \max_i \{\max\{\bar{t}_i, \bar{c}_i\}\}$. We can do so by defining $f_i(t) = M$ for $t = 0, 1, \dots, \underline{t}_i - 1$, defining $f_i(t) = f_i(\bar{t}_i)$ for $t = \bar{t}_i + 1, \bar{t}_i + 2, \dots, U$, defining $g_i(c) = M$ for $c = 0, 1, \dots, \underline{c}_i - 1$, and defining $g_i(c) = g_i(\bar{c}_i)$ for $c = \bar{c}_i + 1, \bar{c}_i + 2, \dots, U$, where M is a large integer. (Note: It suffices to set $M = \max\{\sum_i f_i(\underline{t}_i), \sum_i g_i(\underline{c}_i)\} + 1$.)

Throughout the paper, all logarithms are base 2 unless otherwise stated.

2 K -approximation Sets and Functions

Halman *et al.* [8] have introduced K -approximation sets and functions, and used them to develop an FPTAS for a stochastic inventory control problem. Halman *et al.* [7] have applied these tools to develop a general framework for constructing FPTASs for stochastic dynamic programs. In this section we provide an overview of K -approximation sets and functions. In the next section we will use them to construct FPTASs for our time-cost tradeoff problems. To simplify the discussion, we modify Halman *et al.*'s definition of the K -approximation function by restricting it to integer-valued

functions.

Let $K \geq 1$, and let $\psi : \{0, 1, \dots, U\} \rightarrow Z^+$ be an arbitrary function. We say that $\hat{\psi} : \{0, 1, \dots, U\} \rightarrow Z^+$ is a K -approximation function of ψ if $\psi(x) \leq \hat{\psi}(x) \leq K\psi(x)$ for all $x = 0, 1, \dots, U$. The following property of K -approximation functions is extracted from Proposition 4.1 of [7], which provides a set of general computational rules of K -approximation functions. Its validity follows directly from the definition of the K -approximation function.

Property 1 For $i = 1, 2$, let $K_i \geq 1$, let $\psi_i : \{0, 1, \dots, U\} \rightarrow Z^+$ be an arbitrary function, let $\tilde{\psi}_i : \{0, 1, \dots, U\} \rightarrow Z^+$ be a K_i -approximation function of ψ_i , and let $\alpha, \beta \in Z^+$. The following properties hold:

Summation of approximation: $\alpha\tilde{\psi}_1 + \beta\tilde{\psi}_2$ is a $\max\{K_1, K_2\}$ -approximation function of $\alpha\psi_1 + \beta\psi_2$.

Approximation of approximation: If $\psi_2 = \tilde{\psi}_1$ then $\tilde{\psi}_2$ is a K_1K_2 -approximation function of ψ_1 .

Let $K > 1$. Let $\varphi : \{0, 1, \dots, U\} \rightarrow Z^+$ be a nonincreasing function and $S = (k_1, k_2, \dots, k_r)$ be an ordered subset of $\{0, 1, \dots, U\}$, where $0 = k_1 < k_2 < \dots < k_r = U$. We say that S is a K -approximation set of φ if $\varphi(k_j) \leq K\varphi(k_{j+1})$ for each $j = 1, 2, \dots, r-1$ that satisfies $k_{j+1} - k_j > 1$. (The term used in [7] is *weak K -approximation set of φ* .) Given φ , there exists a K -approximation set of φ with cardinality $O(\log_K \bar{U})$, where \bar{U} is any constant upper bound of $\varphi(0)$. Furthermore, this set can be constructed in $O((1 + \tau(\varphi)) \log_K \bar{U} \log U)$ time, where $\tau(\varphi)$ is the amount of time required to evaluate φ (see Lemma 3.1 of [7]).

Given φ and a K -approximation set $S = (k_1, k_2, \dots, k_r)$ of φ , a K -approximation function of φ can be obtained easily as follows (Definition 3.4 of [7]): Define $\hat{\varphi} : \{0, 1, \dots, U\} \rightarrow Z^+$ such that

$$\hat{\varphi}(x) = \varphi(k_j) \text{ for } k_j \leq x < k_{j+1} \text{ and } j = 1, 2, \dots, r-1,$$

and that

$$\hat{\varphi}(k_r) = \varphi(k_r).$$

Note that $\varphi(x) \leq \hat{\varphi}(x) \leq K\varphi(x)$ for $x = 0, 1, \dots, U$. Therefore, $\hat{\varphi}$ is a nonincreasing K -approximation function of φ . We say that $\hat{\varphi}$ is the K -approximation function of φ corresponding to S .

3 Series and Parallel Reductions

Two-terminal edge series-parallel networks (or simply “series-parallel networks”) are defined recursively as follows [14]: (i) A directed network consisting of two vertices (i.e., a “source” and a “sink”) joined by a single arc is series-parallel. (ii) If two directed networks G_1 and G_2 are series-parallel, then so are the networks constructed by each of the following operations: (a) Two-terminal series composition: Identify the sink of G_1 with the source of G_2 . (b) Two-terminal parallel composition: Identify the source of G_1 with the source of G_2 and the sink of G_1 with the sink of G_2 .

As mentioned in Section 1, a series-parallel network can be reduced to a single-arc network via a sequence of series and parallel reduction operations. A *series reduction* is an operation that replaces two series arcs by a single arc, while a *parallel reduction* is an operation that replaces two parallel arcs by a single arc (see Figure 1). In a project network, a reduction of two series activities with time duration t' and t'' will result in a single activity with time duration $t' + t''$, while a reduction of two parallel activities with time duration t' and t'' will result in a single activity with time duration $\max\{t', t''\}$. For a given series-parallel project network of n activities, it takes $n - 1$ series/parallel reduction operations to reduce it to a single-activity network. However, when there are time-cost tradeoff decisions for the activities, the integration of the two time-cost tradeoff functions during a series/parallel reduction operation becomes a challenge if we want to perform the computation efficiently. In the following subsections, we explain how to apply series and parallel reductions, together with K -approximation sets and functions, to develop FPTASs for the deadline and budget problems.

Note that series-parallel graphs have tree-width 2 (see [12], where “tree-width” was first introduced). It is known that many optimization problems on low tree-width graphs admit dynamic programs, which often lead to efficient exact/approximation algorithms that are unlikely to exist if the graphs were general [1]. Our paper goes along this line of research.

3.1 The Deadline Problem

For a given error tolerance $\epsilon \in (0, 1]$, our approximation algorithm for the deadline problem can be described as follows:

Step 1: Let $K = 1 + \frac{\epsilon}{2n}$.

Step 2: For each activity i , obtain a K -approximation set S_i of f_i , and obtain the K -approximation function \hat{f}_i of f_i corresponding to S_i .

Step 3: Select any pair of series or parallel activities i_1 and i_2 .

Case (a): If i_1 and i_2 are series activities, then perform a series reduction to replace these two activities by an activity i . Obtain a K -approximation set \bar{S}_i of \bar{f}_i , where

$$\bar{f}_i(t) = \min_{t' \in \{0,1,\dots,t\} \cap (S_{i_1} \cup \{t-x \mid x \in S_{i_2}\})} \{\hat{f}_{i_1}(t') + \hat{f}_{i_2}(t-t')\}. \quad (1)$$

Obtain the K -approximation function \hat{f}_i of \bar{f}_i corresponding to \bar{S}_i (i.e., obtain and store the values of $\{\hat{f}_i(t) \mid t \in \bar{S}_i\}$ in an array arranged in ascending order of t).

Case (b): If i_1 and i_2 are parallel activities, then perform a parallel reduction to replace these two activities by an activity i . Obtain a K -approximation set \bar{S}_i of \bar{f}_i , where

$$\bar{f}_i(t) = \hat{f}_{i_1}(t) + \hat{f}_{i_2}(t). \quad (2)$$

Obtain the K -approximation function \hat{f}_i of \bar{f}_i corresponding to \bar{S}_i .

Step 4: If the project network contains only one activity i_0 , then the approximated solution value is given by $\hat{f}_{i_0}(d)$. Otherwise, return to Step 3.

We first discuss Case (a) of Step 3. Suppose that we allocate t time units to a pair of series activities i_1 (along arc $u \rightarrow v$) and i_2 (along arc $v \rightarrow w$); that is, we allow these two activities to spend no more than a total of t time units. Then, the merged activity i (along with merged arc $u \rightarrow w$, as shown in Figure 1(a)), which has a duration of t , will incur a cost of

$$f_i(t) = \min_{t'=0,1,\dots,t} \{f_{i_1}(t') + f_{i_2}(t-t')\}, \quad (3)$$

where $f_{i_1}(t')$ and $f_{i_2}(t-t')$ are the costs of the original activities i_1 and i_2 if they are allocated t' and $t-t'$ time units, respectively. Suppose we do not know the exact time-cost tradeoff functions f_{i_1} and f_{i_2} of these two activities, but instead we have: (i) a nonincreasing K^{k-1} -approximation function \bar{f}_{i_1} of f_{i_1} and a nonincreasing $K^{\ell-1}$ -approximation function \bar{f}_{i_2} of f_{i_2} , where k and ℓ are positive integers, and (ii) a K -approximation set S_{i_j} of \bar{f}_{i_j} and the K -approximation function \hat{f}_{i_j} of \bar{f}_{i_j} corresponding to S_{i_j} for $j = 1, 2$. Then, we obtain \bar{f}_i using equation (1). We first show that \bar{f}_i is a nonincreasing function.

Property 2 \bar{f}_i defined in (1) is a nonincreasing function.

Proof: Consider any $t \in \{0, 1, \dots, U-1\}$. Then $\bar{f}_i(t) = \hat{f}_{i_1}(t^*) + \hat{f}_{i_2}(t-t^*)$ for some $t^* \in \{0, 1, \dots, t\} \cap (S_{i_1} \cup \{t-x \mid x \in S_{i_2}\})$. We have $t^* \in S_{i_1}$ or $t-t^* \in S_{i_2}$ (or both). If $t^* \in S_{i_1}$, then $t^* \in \{0, 1, \dots, t, t+1\} \cap (S_{i_1} \cup \{t+1-x \mid x \in S_{i_2}\})$, which implies that

$$\bar{f}_i(t+1) \leq \hat{f}_{i_1}(t^*) + \hat{f}_{i_2}(t+1-t^*) \leq \hat{f}_{i_1}(t^*) + \hat{f}_{i_2}(t-t^*) = \bar{f}_i(t).$$

If $t-t^* \in S_{i_2}$, then $t^*+1 \in \{t+1-x \mid x \in S_{i_2}\} \subseteq \{0, 1, \dots, t, t+1\} \cap (S_{i_1} \cup \{t+1-x \mid x \in S_{i_2}\})$, which implies that

$$\bar{f}_i(t+1) \leq \hat{f}_{i_1}(t^*+1) + \hat{f}_{i_2}(t-t^*) \leq \hat{f}_{i_1}(t^*) + \hat{f}_{i_2}(t-t^*) = \bar{f}_i(t).$$

Therefore, \bar{f}_i is nonincreasing. ■

The following property is modified from Theorem 4.1 of [7].

Property 3 *Let f_i and \bar{f}_i be the functions defined in (3) and (1), respectively. Then, \bar{f}_i is a $K^{\max\{k,\ell\}}$ -approximation function of f_i .*

Proof: Consider any fixed $t \in \{0, 1, \dots, U\}$. Let $t^* = \arg \min_{t'=0,1,\dots,t} \{f_{i_1}(t') + f_{i_2}(t-t')\}$ (with ties broken arbitrarily). Let $t^{**} = \arg \min_{t' \in \{0,1,\dots,t\} \cap (S_{i_1} \cup \{t-x \mid x \in S_{i_2}\})} \{\hat{f}_{i_1}(t') + \hat{f}_{i_2}(t-t')\}$ (with ties broken arbitrarily). We have

$$\bar{f}_i(t) = \hat{f}_{i_1}(t^{**}) + \hat{f}_{i_2}(t-t^{**}) \geq f_{i_1}(t^{**}) + f_{i_2}(t-t^{**}) \geq f_{i_1}(t^*) + f_{i_2}(t-t^*) = f_i(t). \quad (4)$$

Because \hat{f}_{i_1} is the K -approximation function of \bar{f}_{i_1} corresponding to S_{i_1} , there exists $t_0 \in S_{i_1}$ such that $t_0 \leq t^*$ and $\hat{f}_{i_1}(t_0) = \hat{f}_{i_1}(t^*)$. This implies that $\hat{f}_{i_1}(t_0) \leq K\bar{f}_{i_1}(t^*) \leq K^k f_{i_1}(t^*)$. Note that $\hat{f}_{i_2}(t-t_0) \leq \hat{f}_{i_2}(t-t^*) \leq K\bar{f}_{i_2}(t-t^*) \leq K^\ell f_{i_2}(t-t^*)$. Thus,

$$\begin{aligned} \bar{f}_i(t) &= \hat{f}_{i_1}(t^{**}) + \hat{f}_{i_2}(t-t^{**}) \leq \hat{f}_{i_1}(t_0) + \hat{f}_{i_2}(t-t_0) \\ &\leq K^k f_{i_1}(t^*) + K^\ell f_{i_2}(t-t^*) \leq K^{\max\{k,\ell\}} f_i(t). \end{aligned} \quad (5)$$

Combining (4) and (5) yields the desired result. ■

In Case (a) of Step 3, \bar{S}_i is a K -approximation set of \bar{f}_i . Due to Property 2, \bar{S}_i is well defined. Function \hat{f}_i is the (nonincreasing) K -approximation function of \bar{f}_i corresponding to \bar{S}_i . By approximation of approximation (Property 1), \hat{f}_i is a nonincreasing $K^{\max\{k,\ell\}+1}$ -approximation function

of f_i . The amount of time required to evaluate $\bar{f}_i(t)$ for each t is

$$\tau(\bar{f}_i) = O((|S_{i_1}| + |S_{i_2}|)(\tau(\hat{f}_{i_1}) + \tau(\hat{f}_{i_2}))).$$

Let \bar{U} be any constant upper bound of $\hat{f}_{i_0}(d)$. Then, $|S_{i_1}| = O(\log_K \bar{U})$, $|S_{i_2}| = O(\log_K \bar{U})$, and $\tau(\hat{f}_{i_j}) = O(\log |S_{i_j}|)$ for $j = 1, 2$ (because the values of $\{\hat{f}_{i_j}(t) \mid t \in S_{i_j}\}$ are stored in an array arranged in ascending order of t , for any $t = 0, 1, \dots, U$, it takes only $O(\log |S_{i_j}|)$ time to search for the value of $\hat{f}_{i_j}(t)$). Thus, $\tau(\bar{f}_i) \leq O(\log_K \bar{U} \log \log_K \bar{U})$, and therefore the time required for constructing \bar{S}_i is $O((1 + \tau(\bar{f}_i)) \log_K \bar{U} \log U) \leq O(\log_K^2 \bar{U} \log U \log \log_K \bar{U})$.

Next, we discuss Case (b) of Step 3. Suppose that we allocate t time units to a pair of parallel activities i_1 and i_2 ; that is, we allow each of these two activities to spend no more than t time units. Then, the merged activity, which has a maximum duration of t , will incur a cost of

$$f_i(t) = f_{i_1}(t) + f_{i_2}(t), \tag{6}$$

where $f_{i_1}(t)$ and $f_{i_2}(t)$ are the costs of the original activities i_1 and i_2 , respectively. Suppose we do not know the exact time-cost tradeoff functions f_{i_1} and f_{i_2} , but instead we have: (i) a nonincreasing K^{k-1} -approximation function \bar{f}_{i_1} of f_{i_1} and a nonincreasing $K^{\ell-1}$ -approximation function \bar{f}_{i_2} of f_{i_2} , where k and ℓ are positive integers, and (ii) a K -approximation set S_{i_j} of \bar{f}_{i_j} and the K -approximation function \hat{f}_{i_j} of \bar{f}_{i_j} corresponding to S_{i_j} for $j = 1, 2$. Then, \hat{f}_{i_1} is a K^k -approximation function of f_{i_1} , and \hat{f}_{i_2} is a K^ℓ -approximation function of f_{i_2} .

By summation of approximation (Property 1), \bar{f}_i defined in (2) is a $K^{\max\{k, \ell\}}$ -approximation function of f_i . Clearly, \bar{f}_i is nonincreasing. Let \bar{S}_i be a K -approximation set of \bar{f}_i , and \hat{f}_i be the (nonincreasing) K -approximation function of \bar{f}_i corresponding to \bar{S}_i . By approximation (Property 1), \hat{f}_i is a $K^{\max\{k, \ell\}+1}$ -approximation function of f_i . The amount of time required to evaluate \bar{f}_i is

$$\tau(\bar{f}_i) = O(\tau(\hat{f}_{i_1}) + \tau(\hat{f}_{i_2})) = O(\log |S_{i_1}| + \log |S_{i_2}|) \leq O(\log \log_K \bar{U}).$$

The amount of time required to construct \bar{S}_i is $O((1 + \tau(\bar{f}_i)) \log_K \bar{U} \log U)$, which is dominated by the running time for constructing \bar{S}_i in the series reduction case.

Let $f^*(d)$ denote the optimal total cost of the project for a given deadline d . We now analyze how close $\hat{f}_{i_0}(d)$ is to $f^*(d)$. Note that after performing r series/parallel reduction operations ($0 \leq r \leq n-1$), the project network has $n-r$ activities, namely i_1, i_2, \dots, i_{n-r} . Associated with each activity i_j is a function \hat{f}_{i_j} , which is a K^{β_j} -approximation function of f_{i_j} for some positive integer β_j . We define $\sum_{j=1}^{n-r} \beta_j$ as the *approximation level* of this project.

Before performing any series/parallel reduction, the project has an approximation level n . Since $\max\{k, \ell\} + 1 \leq k + \ell$, neither a series reduction operation nor a parallel reduction operation will increase the approximation level of the project. Hence, at the end of the solution procedure, the approximation level of the project is at most n , which implies that \hat{f}_{i_0} is a K^n -approximation of f^* . Recall that $K = 1 + \frac{\epsilon}{2n}$. Because $(1 + \frac{\epsilon}{2n})^n \leq 1 + \epsilon$, we conclude that $\hat{f}_{i_0}(d)$ is a $(1 + \epsilon)$ -approximation solution to the deadline problem.

Finally, we analyze the running time of the approximation algorithm. Step 2 obtains a K -approximation set and function for each activity. The running time of this step is dominated by that of the series/parallel reduction operations in Step 3. The construction of \bar{S}_i in each series/parallel reduction takes $O(\log_K^2 \bar{U} \log U \log \log_K \bar{U})$ time. Thus, the running time of the entire solution procedure is $O(n \log_K^2 \bar{U} \log U \log \log_K \bar{U})$. Since $\log_K \bar{U} \leq \frac{1}{K-1} \log_2 \bar{U}$ (because $\log_2 K \geq K-1$ for $1 < K < 2$), the running time is $O(\frac{n^3}{\epsilon^2} \log^2 \bar{U} \log U \log(\frac{n}{\epsilon} \log \bar{U}))$. By setting $\bar{U} = K^n \sum_{i=1}^n f_i(\underline{t}_i)$, we get that our solution scheme is an FPTAS.

3.2 The Budget Problem

We now consider the budget problem. Let $g^*(b)$ denote the optimal duration of the project for a given budget b . Suppose we allocate c units of monetary resources to a pair of series activities i_1 (along arc $u \rightarrow v$) and i_2 (along arc $v \rightarrow w$). Then, the merged activity i (along the merged arc $u \rightarrow w$), which has a budget of c , will have a duration of

$$g_i(c) = \min_{c'=0,1,\dots,c} \{g_{i_1}(c') + g_{i_2}(c - c')\}, \quad (7)$$

where $g_{i_1}(c')$ and $g_{i_2}(c - c')$ are the activity times of the original activities i_1 and i_2 if they are allocated monetary resources of c' and $c - c'$, respectively. Let \bar{g}_{i_1} be a nonincreasing K^{k-1} -approx-

imation function of g_{i_1} , and \bar{g}_{i_2} be a nonincreasing $K^{\ell-1}$ -approximation function of g_{i_2} . Let S_{i_j} be a K -approximation set of \bar{g}_{i_j} , and \hat{g}_{i_j} be the K -approximation function of \bar{g}_{i_j} corresponding to S_{i_j} ($j = 1, 2$). Then, \hat{g}_{i_1} is a K^k -approximation function of g_{i_1} , and \hat{g}_{i_2} is a K^ℓ -approximation function of g_{i_2} . Following the same argument as in Section 3.1, we define function \bar{g}_i such that for $t = 0, 1, \dots, U$,

$$\bar{g}_i(c) = \min_{c' \in \{0, 1, \dots, c\} \cap (S_{i_1} \cup \{c-x \mid x \in S_{i_2}\})} \{\hat{g}_{i_1}(c') + \hat{g}_{i_2}(c - c')\}.$$

By Properties 2 and 3, \bar{g}_i is a nonincreasing $K^{\max\{k, \ell\}}$ -approximation function of g_i . Let \bar{S}_i be a K -approximation set of \bar{g}_i , and \hat{g}_i be the (nonincreasing) K -approximation function of \bar{g}_i corresponding to \bar{S}_i . Then, \hat{g}_i is a nonincreasing $K^{\max\{k, \ell\}+1}$ -approximation function of g_i , and \bar{S}_i can be constructed in $O(\log_K^2 \bar{U} \log U \log \log_K \bar{U})$ time, where \bar{U} is any constant upper bound of $\hat{g}_{i_0}(b)$.

Now, suppose that we allocate c units of monetary resources to a pair of parallel activities i_1 and i_2 . Then, the merged activity will have an activity time of

$$g_i(c) = \min_{c'=0,1,\dots,c} \left\{ \max \{g_{i_1}(c'), g_{i_2}(c - c')\} \right\}. \quad (8)$$

We define function \bar{g}_i such that for $t = 0, 1, \dots, U$,

$$\bar{g}_i(c) = \min_{c' \in \{0, 1, \dots, c\} \cap (S_{i_1} \cup \{c-x \mid x \in S_{i_2}\})} \left\{ \max \{\hat{g}_{i_1}(c'), \hat{g}_{i_2}(c - c')\} \right\},$$

with S_{i_1} , S_{i_2} , \hat{g}_{i_1} , and \hat{g}_{i_2} having the same definitions as before. Using the same argument as in the proofs of Properties 2 and 3, we can show that \bar{g}_i is a nonincreasing $K^{\max\{k, \ell\}}$ -approximation function of g_i . Let \bar{S}_i be a K -approximation set of \bar{g}_i , and \hat{g}_i be the K -approximation function of \bar{g}_i corresponding to \bar{S}_i . Then, \hat{g}_i is a $K^{\max\{k, \ell\}+1}$ -approximation function of g_i , and \bar{S}_i can be constructed in $O(\log_K^2 \bar{U} \log U \log \log_K \bar{U})$ time.

Similar to the deadline problem, we determine an approximation solution to the budget problem by first obtaining a K -approximation set S_i and the K -approximation function of f_i corresponding to S_i for each activity i , and then applying series and parallel reductions recursively until the project is reduced to a single activity i_0 . The solution value is given by $\hat{g}_{i_0}(b)$, which is a K^n -approximation of $\hat{g}^*(b)$. Let $K = 1 + \frac{\epsilon}{2n}$, where $0 < \epsilon \leq 1$. Then, $\hat{g}_{i_0}(b)$ is a $(1 + \epsilon)$ -approximation solution to the budget problem, and the running time of the solution procedure is $O(\frac{n^3}{\epsilon^2} \log^2 \bar{U} \log U \log(\frac{n}{\epsilon} \log \bar{U}))$. Therefore, our solution scheme is an FPTAS.

4 Concluding Remarks

We have developed FPTASs for both the deadline and budget problems. Note that although these FPTASs generate solutions with relative errors bounded by ϵ , the actual relative error of a solution is affected by the sequence of series and parallel reduction operations. For example, consider the deadline problem with only four activities i_1, i_2, i_3, i_4 arranged in series, where i_j is the immediate predecessor of i_{j+1} ($j = 1, 2, 3$). At the beginning of the solution procedure, we obtain a K -approximation set and a K -approximation function for each of these activities. Suppose we perform series reductions in the following sequence: (i) merge i_1 and i_2 to form a new activity i_{12} ; (ii) merge i_{12} and i_3 to form a new activity i_{123} ; and (iii) merge i_{123} and i_4 to form a network with a single activity i_0 . Then, step (i) generates a K^2 -approximation function $\hat{f}_{i_{12}}$ of $f_{i_{12}}$. Step (ii) generates a K^3 -approximation function $\hat{f}_{i_{123}}$ of $f_{i_{123}}$. Step (iii) generates a K^4 -approximation function \hat{f}_{i_0} of f_{i_0} .

Now, suppose we perform the series reductions in another sequence: (i) merge i_1 and i_2 to form a new activity i_{12} ; (ii) merge i_3 and i_4 to form a new activity i_{34} ; and (iii) merge i_{12} and i_{34} to form a network with a single activity i_0 . Then, step (i) generates a K^2 -approximation function $\hat{f}_{i_{12}}$ of $f_{i_{12}}$. Step (ii) generates a K^2 -approximation function $\hat{f}_{i_{34}}$ of $f_{i_{34}}$. Step (iii) generates a K^3 -approximation function \hat{f}_{i_0} of f_{i_0} . Hence, this sequence of series reduction operations yields a better approximation than the previous one.

Our FPTAS for the deadline problem uses only the “primal” dynamic program in (3) and (6). It not only approximates the value of the optimal solution $f^*(d)$ for the deadline problem, but also stores an approximation of the function f^* over the entire domain $\{0, 1, \dots, d\}$ in a sorted array of size $O(\frac{n}{\epsilon} \log \bar{U})$. Therefore, for any integer $x \in \{0, 1, \dots, d\}$, only $O(\log(\frac{n}{\epsilon} \log \bar{U}))$ additional time is needed to determine the approximated value of $f^*(x)$.

We note that it is also possible to approximate the deadline and budget problems using the traditional “scaling and rounding the data” approach. On one hand, for doing so one needs to use the “dual” dynamic program (e.g., recursions (7) and (8) for the deadline problem). On the other hand, by applying the elegant technique of Hassin [9], it is possible to reduce the $\log \bar{U}$ term in the running time to $\log \log \bar{U}$. This is done by performing binary search in the log domain and

rounding/scaling $g_i(c)$ in (7) and (8) for every value c where these functions are computed. Unlike our approach, approximating $f^*(x)$ for any additional x will require the same running time.

Our solution method can be extended to non-series-parallel project networks. However, the running time of the approximation algorithm will no longer be polynomial. To tackle non-series-parallel project networks, besides series and parallel reductions, we also make use of node reduction. Any two-terminal directed acyclic network can be reduced to a single arc via series, parallel, and node reductions (see [2]). A node reduction operation can be applied when the node concerned has either in-degree 1 or out-degree 1. Suppose node v has in-degree 1. Let $u \rightarrow v$ be the arc into v , and $v \rightarrow w_1, v \rightarrow w_2, \dots, v \rightarrow w_k$ be the arcs out of v . Then a node reduction at v is to replace these $k + 1$ arcs by arcs $u \rightarrow w_1, u \rightarrow w_2, \dots, u \rightarrow w_k$. The case where v has out-degree 1 is defined symmetrically. In our deadline and budget problems, such a node reduction implies a decomposition of the problem into $m(i)$ separate problems, where $m(i)$ is the number of time-cost alternatives of the activity i corresponding to arc $u \rightarrow v$. In each decomposed problem, we obtain the time-cost tradeoff functions for arcs $u \rightarrow w_1, u \rightarrow w_2, \dots, u \rightarrow w_k$ by adding the time duration and activity cost of $u \rightarrow v$ to the time-cost tradeoff functions of $v \rightarrow w_1, v \rightarrow w_2, \dots, v \rightarrow w_k$, respectively. Bein *et al.* [2] have developed an efficient method for determining the minimum number of node reductions in order to reduce the given project network to a single activity. They refer to this minimum number of node reductions as reduction complexity. Therefore, a discrete time-cost tradeoff problem in a non-series-parallel project network can be decomposed into \bar{m}^h time-cost tradeoff problems with series-parallel networks, where $\bar{m} = \max_i\{m(i)\}$ and h is the reduction complexity. If h is bounded by a constant (i.e., the network is near-series-parallel) and \bar{m} is bounded by a polynomial of the problem input size, then making such a decomposition and applying the algorithms presented in Section 3 will give us an FPTAS for the problem.

Note that the computational complexity of this decomposition method increases exponentially as the reduction complexity increases. Hence, this method is practical only if h is small. As mentioned in Section 1, for general non-series-parallel project networks, it is very difficult to obtain an ϵ -approximation algorithm for the time-cost tradeoff problem (for example, the budget problem does not even have a polynomial-time approximation algorithm with performance guarantee better

than $\frac{3}{2}$ unless P=NP).

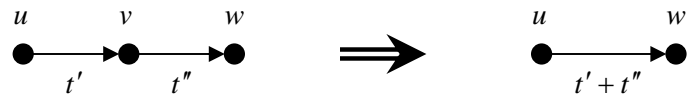
Acknowledgments

This research was supported in part by the Research Grants Council of Hong Kong under grant PolyU5228/08E. Research is also supported by NSF Contract CMMI-0758069.

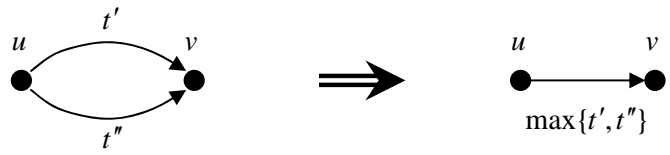
References

- [1] B.S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the Association for Computing Machinery* 41 (1994) 153–180.
- [2] W.W. Bein, J. Kamburowski, M.F.M. Stallmann. Optimal reduction of two-terminal directed acyclic graphs. *SIAM Journal on Computing* 21 (1992) 1112–1129.
- [3] P. De, E.J. Dunne, J.B. Ghosh, C.E. Wells. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research* 45 (1997) 302–306.
- [4] V.G. Deineko, G.J. Woeginger. Hardness of approximation of the discrete time-cost tradeoff problem. *Operations Research Letters* 29 (2001) 207–210.
- [5] H. Frank, I.T. Frisch, R. Van Slyke, W.S. Chou. Optimal design of centralized computer networks. *Networks* 1 (1971) 43–58.
- [6] A. Grigoriev, G.J. Woeginger. Project scheduling with irregular costs: complexity, approximability, and algorithms. *Acta Informatica* 41 (2004) 83–97.
- [7] N. Halman, D. Klabjan, C.-L. Li, J. Orlin, D. Simchi-Levi. Fully polynomial time approximation schemes for stochastic dynamic programs. *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2008) 700–709.
- [8] N. Halman, D. Klabjan, M. Mostagir, J. Orlin, D. Simchi-Levi. A fully polynomial time approximation scheme for single-item stochastic inventory control with discrete demand. Working paper submitted for publication, Massachusetts Institute of Technology, Cambridge, MA, 2006.

- [9] R. Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research* 17 (1992) 36–42.
- [10] T.J. Hindelang, J.F. Muth. A dynamic programming algorithm for decision CPM networks. *Operations Research* 27 (1979) 225–241.
- [11] B. Rothfarb, H. Frank, D.M. Rosebbaum, K. Steiglitz, D.J. Kleitman. Optimal design of offshore natural-gas pipeline systems. *Operations Research* 18 (1970) 992–1020.
- [12] N. Robertson, P.D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms* 7 (1986) 309–322.
- [13] M. Skutella. Approximation algorithms for the discrete time-cost tradeoff problem. *Mathematics of Operations Research* 23 (1998) 909–929.
- [14] J. Valdes, R.E. Tarjan, E.L. Lawler. The recognition of series-parallel digraphs. *SIAM Journal on Computing* 11 (1982) 298–313.
- [15] G.J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing* 12 (2000) 57–74.



(a) series reduction



(b) parallel reduction

Figure 1. Series and parallel reductions