Polynomial-Time Approximation Schemes for a Class of Integrated Network Design and Scheduling Problems with Parallel Identical Machines

Yusuke Saito[1] Akiyoshi Shioura[1,a)]

Abstract: In the integrated network design and scheduling problem (INDS-P), we are asked to repair edges in a graph by using parallel machines so that the performance of the network is recovered by a certain level, and the objective is to minimize the makespan required to finish repairing edges. The main aim of this paper is to show that polynomial-time approximation schemes exist for some class of the problem (INDS-P), including the problems associated with minimum spanning tree, shortest path, maximum flow with unit capacity, and maximum-weight matching.

Keywords: network optimization, parallel machine scheduling, polynomial-time approximation scheme, approximation algorithm

1. Introduction and Results

Network optimization problems aim at constructing networks of good performance. Let us consider a situation where an existing network is damaged due to a disaster such as an earthquake, a flood, or a hurricane. In this situation, it is required to repair the network as quickly as possible so that the performance of the network is recovered by a certain level. A mathematical model of this problem is provided in [15] (see also [14]), which is referred to as the integrated network design and scheduling problem. In this paper, we deal with a variant of the integrated network design and scheduling problem, which we denote by (INDS-P), where parallel identical machines are used to repair edges and the objective of minimizing the makespan[*1].

1.1 Problem Definition

The definition of the problem (INDS-P) is explained in more detail. Let $G = (V, E)$ be a (directed or undirected) graph representing the network, where V is the vertex set and E is the edge set. We are given a set $E_0 \subseteq E$ of edges that are still alive after the disaster and the remaining edges in $E \setminus E_0$ are damaged. We select some damaged edges and repair them as quickly as possible so that the performance of the repaired graph reaches a certain desired level.

Selected damaged edges are repaired by using m parallel identical machines, where m is assumed to be a constant throughout this paper. Time required to repair an edge $e \in E \setminus E_0$ is given by a non-negative integer $p(e)$. Each edge can be processed by a single machine and is not allowed to be processed simultaneously by two or more machines. In addition, preemption is not allowed for each edge, i.e., once we start repairing an edge, then we need to continue it until the repair finishes. The objective of the problem (INDS-P) is to minimize the makespan, i.e., the time required to finish repairing edges. The makespan is represented as $\max\{p(X_i) \mid i = 1, 2, \ldots, m\}$, where X_i is a set of edges repaired by the i-th machine and $p(X_i) = \sum_{e \in X_i} p(e)$.

A set $X \subseteq E \setminus E_0$ of repaired edges is selected so that the performance of the repaired graph $(V, X \cup E_0)$ reaches a certain level. The performance level of the repaired graph is determined by the edge set $X \cup E_0$ of the repaired graph and represented by a function $\varphi : 2^E \to \mathbb{R} \cup \{\pm\infty\}$, which we call a *performance function*. The value $\varphi(X \cup E_0)$ is given by the optimal value of a certain network optimization problem on the graph $(V, X \cup E_0)$, such as minimum spanning tree, shortest path, and maximum flow (see [15]). With a performance function φ and a threshold value W, the constraint on the performance level is given as $\varphi(X \cup E_0) \leq W$ (resp., $\varphi(X \cup E_0) \geq W$) if φ is defined by a minimization problem (resp., a maximization problem).

In summary, the problem (INDS-P) is formulated as follows:

$$\begin{array}{ll} \text{Minimize} & \max\{p(X_i) \mid i = 1, 2, \ldots, m\} \\ \text{subject to} & \varphi(X \cup E_0) \leq W \quad (\text{or } \varphi(X \cup E_0) \geq W), \\ & X = \bigcup_{i=1}^{m} X_i, \\ & X_1, X_2, \ldots, X_m \subseteq E \setminus E_0 \text{ are mutually disjoint.} \end{array}$$

It is easy to see that this problem is NP-hard since it is a generalization of parallel identical machine scheduling minimizing the makespan (see also [15]). Hence, we focus on approximability of the problem (INDS-P), especially in the cases with the following specific families of performance functions:

- Peformance functions associated with the minimum spanning tree.

  Assume that $G = (V, E)$ is an undirected graph with

[1] Department of Industrial Engineering and Economics, Tokyo Institute of Technology, Tokyo 152-8550, Japan
[a)] shioura.a.aa@m.titech.ac.jp

[*1] This problem is denoted as "$Pm|\beta|C_{\max}$-Threshold" in [15].

non-negative integer edge length $\ell(e)$ ($e \in E$). The value of $\varphi_{\mathrm{MST}}(Y)$ for $Y \subseteq E$ is defined as the length of a minimum spanning tree in the edge-induced subgraph (V, Y); $\varphi_{\mathrm{MST}}(Y) = +\infty$ if the graph (V, Y) has no spanning tree. We denote by Φ_{MST} the family of performance functions associated with the minimum spanning tree.

- Performance functions associated with the single-source single-destination shortest path.

  Assume that $G = (V, E)$ is a directed graph with two distinct vertices $s, t \in V$ and non-negative integer edge length $\ell(e)$ ($e \in E$). The value of $\varphi_{\mathrm{SP}}(Y)$ for $Y \subseteq E$ is defined as the length of a shortest path from s to t in the graph (V, Y); $\varphi_{\mathrm{SP}}(Y) = +\infty$ if the graph (V, Y) has no path from s to t. We denote by Φ_{SP} the family of performance functions associated with the single-source single-destination shortest path.
- Performance functions associated with the maximum-weight matching.

  Assume that $G = (V, E)$ is an undirected graph with non-negative integer edge weight $w(e)$ ($e \in E$). The value of $\varphi_{\mathrm{MM}}(Y)$ for $Y \subseteq E$ is defined as the weight of a maximum-weight matching in the graph (V, Y). We denote by Φ_{MM} the family of performance functions associated with the maximum-weight matching.
- Performances function associated with the maximum flow with unit capacity.

  Assume that $G = (V, E)$ is a directed graph with two distinct vertices $s, t \in V$. The value of $\varphi_{\mathrm{UMF}}(Y)$ for $Y \subseteq E$ is defined as the amount of a maximum flow from s to t in the graph (V, Y). We denote by Φ_{UMF} the family of performance functions associated with the maximum-weight matching.

1.2 Our Results

The main aim of this paper is to show that polynomial-time approximation schemes (PTASes, for short) exist for the problem (INDS-P) with some families of performance functions, including Φ_{MST}, Φ_{SP}, Φ_{UMF}, and Φ_{MM}. Recall that an approximation algorithm for (INDS-P) is a PTAS if for any feasible instance of (INDS-P) and any constant $\varepsilon > 0$, the algorithm finds in polynomial time a feasible solution $(X_1, X_2, \ldots, X_m)$ such that

$$\max\{p(X_i) \mid i = 1, 2, \ldots, m\} \leq (1 + \varepsilon)C^*,$$

where C^* is the optimal value of (INDS-P).

To state the main result, consider the special case of (INDS-P) with $m = 1$, i.e., only a single machine is available. We denote this special case as (INDS-S), which is formulated as

$$\begin{array}{ll} \text{Minimize} & p(X) \\ \text{subject to} & \varphi(X \cup E_0) \leq W \quad (\text{or } \varphi(X \cup E_0) \geq W), \\ & X \subseteq E \setminus E_0. \end{array}$$

We show that (INDS-P) admits a PTAS if its special case (INDS-S) admits a PTAS. That is, the existence of a PTAS for (INDS-P) depends on the performance function φ, and irrelevant to the number m of machines.

Theorem 1.1. *Suppose that the problem* (INDS-S) *with some performance function φ admits a PTAS (or a polynomial-time exact algorithm). Then, the problem* (INDS-P) *with the same performance function φ also admits a PTAS.*

As shown in Section 3, if $\varphi \in \Phi_{\mathrm{MST}} \cup \Phi_{\mathrm{SP}} \cup \Phi_{\mathrm{MM}}$ then a PTAS exists for the problem (INDS-S). We also show that if $\varphi \in \Phi_{\mathrm{UMF}}$, then (INDS-S) can be solved in polynomial time. Hence, the following result is obtained as an immediate corollary of Theorem 1.1.

Corollary 1.2. *If $\varphi \in \Phi_{\mathrm{MST}} \cup \Phi_{\mathrm{SP}} \cup \Phi_{\mathrm{UMF}} \cup \Phi_{\mathrm{MM}}$, then the problem* (INDS-P) *admits a polynomial-time approximation scheme.*

Theorem 1.1 follows from the next lemma, stating that (INDS-S) and (INDS-P) are almost equivalent in terms of the approximation ratio; see Section 2 for a proof of the lemma.

Lemma 1.3. *Suppose that the problem* (INDS-S) *with some performance function φ admits a polynomial-time $(1 + \alpha)$-approximation algorithm with some $\alpha \geq 0$. Then, for any constant $\delta > 0$, the problem* (INDS-P) *with the same performance function φ admits a polynomial-time $(1 + \alpha + \delta)$-approximation algorithm*

The problem (INDS-P) can be regarded as a bicriteria optimization problem minimizing the makespan and also minimizing (or maximizing) the performance function. Therefore, we may consider the following problem, which is denoted as (INDS2-P), obtained by swapping the objective function and the constraint on the network performance level:

$$\begin{array}{ll} \text{Minimize (or Maximize)} & \varphi(X \cup E_0) \\ \text{subject to} & p(X_i) \leq C \ (i = 1, 2, \ldots, m), \\ & X = \bigcup_{i=1}^{m} X_i, \\ & X_1, X_2, \ldots, X_m \subseteq E \setminus E_0 \text{ are mutually disjoint.} \end{array}$$

We also consider the special case of (INDS2-P) with $m = 1$, which is denoted as (INDS2-S):

$$\begin{array}{ll} \text{Minimize (or Maximize)} & \varphi(X \cup E_0) \\ \text{subject to} & p(X) \leq C, \ X \subseteq E \setminus E_0. \end{array}$$

For $\alpha > 0$, we say that an algorithm for (INDS2-P) is a $(1, 1 + \alpha)$-approximation algorithm if it finds a feasible solution $X = \bigcup_{i=1}^{m} X_i$ satisfying

$$\begin{aligned} &\varphi(X \cup E_0) \leq W^* \text{ (for minimization problem)}, \\ &\varphi(X \cup E_0) \geq W^* \text{ (for maximization problem)}, \\ &p(X_i) \leq (1 + \alpha)C \quad (i = 1, 2, \ldots, m), \end{aligned}$$

where W^* is the optimal value of (INDS2-P). We also say that (INDS2-P) admits a bicriteria PTAS if it admits a polynomial-time $(1, 1 + \alpha)$-approximation algorithm for every constant $\alpha > 0$.

Theorem 1.4. *Suppose that the problem* (INDS2-S) *with some performance function φ admits a bicriteria PTAS (or a polynomial-time exact algorithm). Then, the problem* (INDS2-P) *with the same performance function φ also admits a bicriteria PTAS.*

As in the case of (INDS-P), we obtain the following corollary on the existence of bicriteria PTASes for some specific performance functions.

Corollary 1.5. *If $\varphi \in \Phi_{\mathrm{MST}} \cup \Phi_{\mathrm{SP}} \cup \Phi_{\mathrm{UMF}} \cup \Phi_{\mathrm{MM}}$, then the problem* (INDS2-P) *admits a bicriteria PTAS.*

Theorem 1.4 follows immediately from the next lemma, to be

proven in Section 2.

Lemma 1.6. *Suppose that the problem* (INDS2-S) *with some specific performance function φ admits a polynomial-time* $(1, 1+\alpha')$*-approximation algorithm with some* $\alpha' \geq 0$*. Then, for any constant* $\alpha > \alpha'$*, the problem* (INDS2-P) *with the same performance function* φ *admits a polynomial-time* $(1, 1+\alpha)$*-approximation algorithm.*

Remark 1.7. The problem (INDS2-P) includes as a special case the multiple knapsack problem with identical capacity knapsacks. The multiple knapsack problem is a natural generalization of the knapsack problem, where multiple knapsacks are given and items can be packed into one of the knapsacks. It is known that the multiple knapsack problem admits a PTAS, even in the case where capacity of knapsacks are different [6], [11]. It is not clear how to apply the techniques used in [6], [11] to obtain a PTAS for (INDS2-P). □

Remark 1.8. In the problems (INDS-P) and (INDS2-P), we may assume, without loss of generality, that $E_0 = \emptyset$. Indeed, Any problem instance with $E_0 \neq \emptyset$ can be reduced to the one with $E_0 = \emptyset$ by setting $p(e) = 0$ for all $e \in E_0$; it is easy to see that the instance after the reduction is essentially equivalent to the original one since all edges e with $p(e) = 0$ can be processed immediately when the machines start processing. □

1.3 Related Work

The integrated network design and scheduling problem is originated by Nurre et al. [14], where the network performance level is determined by the amount of a maximum flow, and the objective is to maximize the cumulative performance level over a finite time horizon. The same problem with different kinds of performance functions is considered in [15], where NP-hardness of the problems is proved and some heuristic algorithms based on dispatching rules are presented. The problem of this type is also referred to as the incremental network design problem and the incremental combinatorial optimization problem, and approximation algorithms with theoretical guarantee have been proposed [4], [7], [8], [10].

In addition to the maximization of the cumulative performance level, Nurre and Sharkey [15] deal with the problem of minimizing the makespan under the performance level constraint, which is nothing but the problem (INDS-P) discussed in this paper. To the best of our knowledge, no approximation algorithm with nontrivial bound is known for (INDS-P) and its special cases with specific performance functions, while a PTAS exists for (INDS-S) if the performance function is given by minimum-weight spanning tree, shortest path, or maximum-weight matching (see Section 3 for details).

A network optimization problem similar to (INDS-P) is discussed by Averbakh, Pereira, et al. (see [1], [2], [18], etc.) in the name "the network construction/reconstruction problem," and various theoretical results as well as computational study have been presented. A major difference between the network construction/reconstruction problem and (INDS-P) is that in the former problem each edge to be repaired should be connected with a fixed depot vertex by existing (alive and already repaired) edges, while there is no such restriction in our problem.

The problem (INDS-P) in this paper is also related to *scheduling problems with rejection* (see, e.g., [3], [17]). In scheduling problem with rejection, we do not need to process all jobs and can reject some jobs, which yields penalty cost. This scheduling problem is a bicriteria optimization problem, where the objective is to optimize some standard objective function such as makespan, total flow time, etc., and also to minimize the total rejection cost. The problem (INDS-P) can be regarded as a scheduling problem with nonlinear rejection cost; the edge set $E \setminus (X \cup X_0)$ can be regarded as the set of rejected edges, and the difference of the values $\varphi(E)$ and $\varphi(X \cup E_0)$ can be regarded as the rejection cost, which is nonlinear in general.

In the literature of scheduling with rejection, the total rejection cost is often assumed to be linear, i.e., it is given as the sum of rejection cost for rejected jobs. Recently, a nonlinear (submodular) rejection cost function is considered in some papers [12], [13], [19], where constant-factor approximation algorithms are proposed, while we focus on PTASes in this paper.

2. Proofs of Lemmas 1.3 and 1.6

To the end of this paper, we assume, without loss of generality, that $E_0 = \emptyset$ in the problems (INDS-P) and (INDS2-P) (see Remark 1.8).

2.1 Proof of Lemma 1.3

We give a proof of Lemma 1.3 in the case where the performance level constraint is $\varphi(X) \leq W$; the case of $\varphi(X) \geq W$ can be proven similarly and omitted.

Let $C^* \in \mathbb{Z}$ be the optimal value of the problem (INDS-P), which is not known in advance. Also, let $\delta' > 0$ be a real number satisfying

$$(1+\alpha+\delta')(1+\delta') \leq 1+\alpha+\delta.$$

In the following, we present a polynomial-time algorithm such that for a given real number C, if $C \geq C^*$ then the algorithm finds a feasible solution $(X_1, X_2, \ldots, X_m)$ of (INDS-P) satisfying

$$p(X_i) \leq (1+\alpha+\delta')C \ (i = 1, 2, \ldots, m), \ \ \varphi(X) \leq W \text{ with } X = \bigcup_{i=1}^{m} X_i; \tag{2.1}$$

note that if $C < C^*$ then the algorithm may find a feasible solution satisfying (2.1), or stop without finding any solution. Using this algorithm and binary search with respect to C, we can find C with $C^* \leq C \leq (1+\delta')C^*$ and a feasible solution $X_1, X_2, \ldots, X_m$ of (INDS-P) satisfying (2.1), for which the approximation ratio is $(1+\alpha+\delta')(1+\delta') \leq 1+\alpha+\delta$. Hence, $(X_1, X_2, \ldots, X_m)$ is an $(1+\alpha+\delta)$-approximate solution of (INDS-P).

We now show that if $C \geq C^*$ then a feasible solution satisfying (2.1) can be computed in polynomial time, even when the value C^* is not known in advance. Let

$$E_S = \{e \in E \mid p(e) < \delta' C\}, \qquad E_L = E \setminus E_S;$$

i.e., E_S (resp., E_L) is the set of edges with small (resp., large) processing time. Since C^* is the optimal value of (INDS-P), there exists a feasible (and optimal) solution $X_1^*, X_2^*, \ldots, X_m^*$ of (INDS-P) such that

$$p(X_i^*) \le C^* \quad (i = 1, 2, \ldots, m), \quad \varphi(X^*) \le W \text{ with } X^* = \bigcup_{i=1}^{m} X_i^*.$$

We first guess the assignment of "large" edges $X_1^* \cap E_L, X_2^* \cap E_L, \ldots, X_m^* \cap E_L$ to m machines in the optimal solution[*2]. Since $p(e) \ge \delta' C$ for each $e \in E_L$, it holds that

$$|X_i^* \cap E_L| \le C^*/(\delta' C) \le C/(\delta' C) = 1/\delta' \qquad (i = 1, 2, \ldots, m),$$

where the second inequality is by the assumption $C \ge C^*$. This implies that there exist at most $n^{m/\delta'}$ possible choices of the assignment $X_1^* \cap E_L, \ldots, X_m^* \cap E_L$. Note that m and δ' are assumed to be constant numbers in this paper, and therefore $n^{m/\delta'}$ is polynomial in the input size of the problem.

Suppose that the sets $X_1^* \cap E_L, X_2^* \cap E_L, \ldots, X_m^* \cap E_L$ are guessed correctly, and let $X_L = \bigcup_{i=1}^{m} (X_i^* \cap E_L)$. We then consider the following problem:

$$\text{Minimize} \quad \widehat{p}(X) \qquad \text{subject to} \quad \varphi(X) \le W,\ X \subseteq E, \quad (2.2)$$

where

$$\widehat{p}(e) = \begin{cases} p(e) & (e \in E \setminus E_L), \\ 0 & (e \in X_L), \\ (1+\alpha)mC + 1 & (e \in E_L \setminus X_L), \end{cases}$$

This is an instance of (INDS-S), and its optimal value is bounded by $\widehat{p}(X^*)$. Since the optimal value of (INDS-P) is at most C and the set X_L is guessed correctly, an upper bound of the value $\widehat{p}(X^*)$ can be obtained as follows:

$$\begin{aligned} \widehat{p}(X^*) = \widehat{p}(X^* \cap E_S) + \widehat{p}(X_L) &= \widehat{p}(X^* \cap E_S) \\ &= p(X^* \cap E_S) \\ &= p(X^*) - p(X_L) \le mC - p(X_L). \end{aligned} \quad (2.3)$$

Let $X^{**} \subseteq \widehat{E}$ be a $(1+\alpha)$-approximate solution of the problem (INDS-S) given above and $X_S = X^{**} \cap E_S$. Since the optimal value of (INDS-S) is at most $mC - p(X_L)$ by (2.3), it holds that

$$\widehat{p}(X^{**}) \le (1+\alpha)(mC - p(X_L)). \quad (2.4)$$

This inequality, together with the definition of $\widehat{p}$, implies that

$$X^{**} \cap (E_L \setminus X_L) = \emptyset.$$

The inequality (2.4) also implies that

$$p(X_S) + p(X_L) = \widehat{p}(X_S) + p(X_L) = \widehat{p}(X^{**}) + p(X_L) \le (1+\alpha)mC. \quad (2.5)$$

We finally construct a feasible solution $(X_1, X_2, \ldots, X_m)$ of (INDS-P) satisfying (2.1) by assigning edges in $X_S \cup X_L$ to m machines appropriately. Edges in X_L are assigned to m machines according to the guessed assignment. Then, edges in X_S are assigned to m machines in a greedy way as follows. We first assign edges in X_S to the 1st machine until its makespan exceeds $(1+\alpha)C$; then assign remaining edges in X_S to the 2nd machine until its makespan exceeds $(1+\alpha)C$, and so on. We stop this iteration if we assign all edges in X_S. Due to the inequality (2.8), all edges in X_S can be assinged to some of the m machines.

Since each edge in X_S has processing time at most $\delta' C$, the makespan of each machine is at most

$$(1+\alpha)C + \delta' C = (1 + \alpha + \delta')C.$$

Hence, $(X_1, X_2, \ldots, X_m)$ is a feasible solution of (INDS-P) satisfying (2.1). This concludes the proof of Lemma 1.3.

Remark 2.1. We discuss the behavior of the algorithm explained above in the case with $C < C^*$. In the algorithm we solve the problem (2.2) for all possible choices of the assignment $X_1^* \cap E_L, \ldots, X_m^* \cap E_L$. As explained above, if $C \ge C^*$ then one of the possible assignments is the correct guess, and therefore the problem (2.2) corresponding to the assignment is feasible. This means that if the problem (2.2) is infeasible for all possible assignments, then we have $C < C^*$. Otherwise, the problem (2.2) is feasible for some assignment; this is possible when $(1+\alpha)C \ge C^*$. In this case, the analysis of the algorithm is still valid, and we obtain a feasible solution of (INDS-P) satisfying (2.1). □

2.2 Proof of Lemma 1.6

We give a proof of Lemma 1.6 in the case where the objective is the minimization of performance function $\varphi(X)$; the case of maximization can be proven similarly and omitted.

Let W^* be the optimal value of (INDS2-P). Then, there exists a feasible (and optimal) solution $X_1^*, X_2^*, \ldots, X_m^*$ of (INDS2-P) such that

$$\varphi(X^*) = W^* \text{ with } X^* = \bigcup_{i=1}^{m} X_i^*, \qquad p(X_i^*) \le C \ (i = 1, 2, \ldots, m).$$

Also, let $\delta = \alpha - \alpha'$ and

$$E_S = \{e \in E \mid p(e) < \delta C\}, \qquad E_L = E \setminus E_S;$$

i.e., E_S (resp., E_L) is the set of edges with small (resp., large) processing time. We first guess the assignment of "large" edges $X_1^* \cap E_L, X_2^* \cap E_L, \ldots, X_m^* \cap E_L$ to m machines in the optimal solution. Since $p(e) \ge \delta C$ for each $e \in E_L$, it holds that

$$|X_i^* \cap E_L| \le C/(\delta C) = 1/\delta \qquad (i = 1, 2, \ldots, m).$$

This implies that there exist at most $n^{m/\delta}$ possible choices of the assignment $X_1^* \cap E_L, \ldots, X_m^* \cap E_L$. Since m and δ are constant numbers, $n^{m/\delta}$ is polynomial in the input size of the problem.

Suppose that the sets $X_1^* \cap E_L, X_2^* \cap E_L, \ldots, X_m^* \cap E_L$ are guessed correctly, and let $X_L = \bigcup_{i=1}^{m} (X_i^* \cap E_L)$. We then consider the following problem:

$$\text{Minimize} \quad \varphi(X) \qquad \text{subject to} \quad \widehat{p}(X) \le mC - p(X_L),\ X \subseteq E, \quad (2.6)$$

where

$$\widehat{p}(e) = \begin{cases} p(e) & (e \in E \setminus E_L), \\ 0 & (e \in X_L), \\ (1+\alpha)mC + 1 & (e \in E_L \setminus X_L), \end{cases}$$

This is an instance of (INDS2-S), and its optimal value is at most W^* since X^* is a feasible solution; indeed, X^* satisfies the inequality constraint:

[*2] By guessing we mean trying all possible assignments by enumeration.

$$\begin{aligned}\widehat{p}(X^*) = \widehat{p}(X^* \cap E_S) + \widehat{p}(X_L) &= p(X^* \cap E_S) \\ &= p(X^*) - p(X_L) \\ &= \sum_{i=1}^{m} p(X_i^*) - p(X_L) \leq mC - p(X_L).\end{aligned}$$

Let $X^{**} \subseteq \widehat{E}$ be a $(1, 1+\alpha')$-approximate solution of the problem (INDS2-S) given above and $X_S = X^{**} \cap E_S$. Since the optimal value of (INDS2-S) is at most W^*, it holds that

$$\begin{aligned}&\varphi(X^*) \leq W^*, \\ &\widehat{p}(X^*) \leq (1+\alpha')(mC - p(X_L)).\end{aligned} \tag{2.7}$$

This inequality, together with the definition of $\widehat{p}$, implies that

$$X^{**} \cap (E_L \setminus X_L) = \emptyset.$$

The inequality (2.7) also implies that

$$p(X_S) + p(X_L) = \widehat{p}(X_S) + p(X_L) = \widehat{p}(X^{**}) + p(X_L) \leq (1+\alpha')mC. \tag{2.8}$$

We finally construct a feasible solution $(X_1, X_2, \ldots, X_m)$ of (INDS2-P) satisfying

$$\varphi(X) \leq W^* \text{ with } X = \bigcup_{i=1}^{m} X_i, \quad p(X_i) \leq (1+\alpha)C \ (i = 1, 2, \ldots, m)$$

by assigning edges in $X_S \cup X_L$ to m machines appropriately. Edges in X_L are assigned to m machines according to the guessed assignment. Then, edges in X_S are assigned to m machines in a greedy way as follows. We first assign edges in X_S to the 1st machine until its makespan exceeds $(1+\alpha')C$; then assign remaining edges in X_S to the 2nd machine until its makespan exceeds $(1+\alpha')C$, and so on. We stop this iteration if we assign all edges in X_S. Due to the inequality (2.8), all edges in X_S can be assinged to some of the m machines.

Since each edge in X_S has processing time at most δC, the makespan of each machine is at most

$$(1+\alpha')C + \delta' C = (1+\alpha)C.$$

Hence, $(X_1, X_2, \ldots, X_m)$ is a feasible solution of (INDS2-P) satisfying the desired conditions. This concludes the proof of Lemma 1.6.

3. Proofs of Corollaries 1.2 and 1.5

Throughout this section we assume, without loss of generality, that $E_0 = \emptyset$.

3.1 Case of $\Phi \in \{\Phi_{\mathrm{MST}}, \Phi_{\mathrm{SP}}\}$.

The problem (INDS-S) with $\varphi \in \Phi_{\mathrm{MST}} \cup \Phi_{\mathrm{SP}}$ is reformulated as follows:

$$\begin{aligned}&\text{Minimize} && p(X) \\ &\text{subject to} && \ell(X) \leq W, \\ &&& X \subseteq E \text{ is a spanning tree (or } s\text{-}t \text{ path)} \\ &&&\qquad\qquad \text{in the graph } (V, E).\end{aligned}$$

This is the minimum spanning tree problem (or the shortest s-t path problem) with a knapsack constraint. It is easy to see that (INDS2-S) has the same problem structure as (INDS-S). The minimum spanning tree problem with a knapsack constraint has a PTAS due to Ravi and Goemans [16], and the shortest s-t path problem with a knapsack constraint has a fully PTAS due to Hassin [9]. Hence, Corollary 1.2 holds in the case $\Phi \in \{\Phi_{\mathrm{MST}}, \Phi_{\mathrm{SP}}\}$. Note that a PTAS can be converted into a $(1, 1+\varepsilon)$-approximation algorithm for every $\varepsilon > 0$ (see, e.g., [16], Section 1). Hence, Corollary 1.5 also holds in this case.

3.1.1 Case of $\Phi = \Phi_{\mathrm{MM}}$.

The problem (INDS2-S) with $\varphi \in \Phi_{\mathrm{MM}}$ is reformulated as follows:

$$\begin{aligned}&\text{Maximize} && w(X) \\ &\text{subject to} && p(X) \leq C, \quad X \subseteq E \text{ is a matching in the graph } (V, E).\end{aligned}$$

This is the maximum-weight matching problem with a knapsack constraint, for which a PTAS exists (see Berger et al. [5]). By using this PTAS for (INDS2-S) and binary search, we can easily obtain a $(1, 1+\varepsilon)$-approximation algorithm of (INDS2-S) for every $\varepsilon > 0$, and also a PTAS for (INDS-S) (see, e.g., [16], Section 1). Hence, Corollaries 1.2 and 1.5 hold if $\Phi = \Phi_{\mathrm{MM}}$.

3.1.2 Case of $\Phi = \Phi_{\mathrm{UMF}}$.

In the problem (INDS-S) with $\varphi \in \Phi_{\mathrm{UMF}}$, we need to find an edge set $X \subseteq E$ that minimizes the value $p(X)$ under the constraint that the amount of a maximum flow in (V, X) is at least W. Since for each edge $e \in E$ its capacity is one and $p(e) \geq 0$, we can assume that for each $e \in X$ contained in the optimal solution of the problem, we have positive flow on the edge e. Hence, we can regard this problem as a minimum-cost flow problem, where we minimize the total flow cost $\sum_{e \in E} p(e) f(e)$ under the constraint that the amount of the s-t flow represented by the vector $f(e)$ $(e \in E)$ is at least W. This observation shows that (INDS-S) with $\varphi \in \Phi_{\mathrm{UMF}}$ can be solved in polynomial time.

Similarly, the problem (INDS2-S) with $\varphi \in \Phi_{\mathrm{UMF}}$ can be regarded as the problem of maximizing the amount of an s-t flow under the constraint that the total cost of the flow is at most C. Hence, we can obtain an optimal solution of (INDS2-S) by solving the problem (INDS-S) for all possible value of W. Since each edge has unit capacity, the amount of flow is at most $|E|$, the number of edges in G. That is, it suffices to solve (INDS-S) for $W = 0, 1, \ldots, |E|$, and then find a maximum s-t flow under the cost constraint, which can be done in polynomial time.

References

[1] I. Averbakh and J. Pereira, "The flowtime network construction problem," IIE Transactions 44 (2012), 681–694.
[2] I. Averbakh and J. Pereira, "Network construction problems with due dates," European Journal of Operational Research 244 (2015), 715–729.
[3] Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, L. Stougie, "Multiprocessor scheduling with rejection," SIAM Journal on Discrete Mathematics 13 (2000), 64–78.
[4] M. Baxter, T. Elgindy, A. T. Ernst, T. Kalinowski, and M.W.P. Savelsbergh, "Incremental network design with shortest paths," European Journal of Operational Research 238 (2014), 675–684.
[5] A. Berger, V. Bonifaci, F. Grandoni, and G. Schäfer, "Budgeted matching and budgeted matroid intersection via the gasoline puzzle," Mathematical Programming 128 (2011), 355–372.
[6] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," SIAM Journal on Computing 35 (2005), 713–728.
[7] K. Engel, T. Kalinowski, and M.W.P. Savelsbergh, "Incremental network design problem with minimum spanning trees," Journal of Graph

Algorithms and Applications 21 (2017), 417–432.
[8] M.X. Goemans and F. Unda, “Approximating incremental combinatorial optimization problems,” Proceedings of APPROX/RANDOM 2017, LIPIcs 81, 6:1–6:14.
[9] R. Hassin, “Approximation schemes for the restricted shortest path problem,” Mathematics of Operations Research 17 (1992), 36–42.
[10] T. Kalinowski, D. Matsypura, and M.W.P. Savelsbergh, “Incremental network design with maximum flows,” European Journal of Operational Research 242 (2015), 51–62.
[11] H. Kellerer, “A polynomial time approximation scheme for the multiple knapsack problem,” Proceedings of APPROX/RANDOM 1999, LNCS 1671, 51–62.
[12] X. Liu and W. Li, “Approximation algorithm for the single machine scheduling problem with release dates and submodular rejection penalty,” Mathematics 8 (2020), 133.
[13] X. Liu and W. Li, “Approximation algorithms for the multiprocessor scheduling with submodular penalties,” Optimization Letters 15 (2021),2165–2180.
[14] S. G. Nurre, B. Cavdaroglu, J. E. Mitchell, T. C. Sharkey, and W. A. Wallace, “Restoring infrastructure systems: an integrated network design and scheduling (INDS) problem,” European Journal of Operational Research 223 (2012), 794–806.
[15] S. G. Nurre and T. C. Sharkey, “Integrated network design and scheduling problems with parallel identical machines: complexity and dispatching rules,” Networks 63 (2014), 303–326.
[16] R. Ravi and M. X. Goemans, “The constrained minimum spanning tree problem,” Proceedings of 5th Scandinavian Workshop on Algorithm Theory (1996), LNCS 1097, 66–75.
[17] D. Shabtay, N. Gaspar, and M. Kaspi, “A survey on offline scheduling with rejection,” Journal of Scheduling 16 (2013), 3–28.
[18] T. Wang and I. Averbakh, “Network construction/restoration problems: cycles and complexity,” Journal of Combinatorial Optimization (2021), published online.
[19] H. Zheng, S. Gao, W. Liu, W. Wu, D.-Z. Du, and B. Hou, “Approximation algorithm for the parallel-machine scheduling problem with release dates and submodular rejection penalties,” Journal of Combinatorial Optimization (2022), published online.

