Submodular Reassignment Problem for Reallocating Agents to Tasks with Synergy Effects

Naonori Kakimura
Keio University
kakimura@math.keio.ac.jp

Naoyuki Kamiyama
Kyushu University, and JST, PRESTO
kamiyama@imi.kyushu-u.ac.jp

Yusuke Kobayashi
Kyoto University
yusuke@kurims.kyoto-u.ac.jp

Yoshio Okamoto
University of Electro-Communications
okamotoy@uec.ac.jp

Abstract

We propose a new combinatorial optimization problem that we call the submodular reassignment problem. We are given k submodular functions over the same ground set, and we want to find a set that minimizes the sum of the distances to the sets of minimizers of all functions. The problem is motivated by a two-stage stochastic optimization problem with recourse summarized as follows. We are given two tasks to be processed and want to assign a set of workers to maximize the sum of profits. However, we do not know the value functions exactly, but only know a finite number of possible scenarios. Our goal is to determine the first-stage allocation of workers to minimize the expected number of reallocated workers after a scenario is realized at the second stage. This problem can be modeled by the submodular reassignment problem. We prove that the submodular reassignment problem can be solved in strongly polynomial time via submodular function minimization. We further provide a maximum-flow formulation of the problem that enables us to solve the problem without using a general submodular function minimization algorithm, and more efficiently both in theory and in practice. In our algorithm, we make use of Birkhoff's representation theorem for distributive lattices.

2012 ACM Subject Classification Theory of computation → Mathematical optimization, Computing methodologies

Keywords and phrases Submodular function, Stochastic optimization, Distributive lattice, Birkhoff's representation theorem, Minimum cut

Funding *Naonori Kakimura*: Partly supported by JSPS KAKENHI Grant Numbers JP17K00028, JP18H05291, JP20K21834 and JP20H05795, Japan.
Naoyuki Kamiyama: Partially supported by JST PRESTO Grant Number JPMJPR1753, Japan.
Yusuke Kobayashi: Partly supported by JSPS KAKENHI Grant Numbers JP18H05291, JP19H05485, JP20K11692, and JP20H05795, Japan.
Yoshio Okamoto: Partially supported by JSPS KAKENHI Grant Numbers JP15K00009, JP20K11670, JP20H05795, JST CREST Grant Number JPMJCR1402, and Kayamori Foundation of Informational Science Advancement.

Acknowledgements The authors are grateful to Takanori Maehara and Akiyoshi Shioura for discusssion related to this work, and also to an anonymous referee for the comments that improved the clarity and the readability of the paper. They also want to thank Takehiro Ito for his hospitality and kindness.

1 Introduction

In this paper, we introduce the following optimization problem. Let U be a finite set. We are given k submodular functions $f_1, \dots, f_k \colon 2^U \to \mathbb{R}$. We want to find $X \subseteq U$ and $Y_i \in \arg\min f_i$ for all $i \in \{1, \dots, k\}$ that

$$\text{minimize} \qquad \sum_{i=1}^{k} |X \mathbin{\triangle} Y_i|, \tag{1}$$

where $X \mathbin{\triangle} Y_i := (X \cup Y_i) \setminus (X \cap Y_i)$ is the *symmetric difference* of X and Y_i, and $\arg\min f_i$ refers to the set of minimizers of f_i. We will denote

$$d_i(X) := \min\{|X \mathbin{\triangle} Y_i| \mid Y_i \in \arg\min f_i\} \tag{2}$$

for $X \subseteq U$. Then, the problem (1) can be rewritten as

$$\underset{X \subseteq U}{\text{minimize}} \qquad \sum_{i=1}^{k} d_i(X). \tag{3}$$

We call this problem the *submodular reassignment problem*.

When $k = 1$, this is equivalent to the submodular function minimization problem, in which we are given a single submodular function $f \colon 2^U \to \mathbb{R}$ and want to find a set $X \subseteq U$ in $\arg\min f$.

1.1 Motivation

The problem is motivated by the following situation.

We are given two tasks T_1 and T_2 that need to be processed. We are employing some workers, and each worker is going to be assigned to exactly one task. A set of workers assigned to a task determines a certain value that represents the profit, the payoff, or the utility. We want to maximize the sum of the values by appropriately assigning the workers.

However, we do not know how much value we may obtain in advance. Rather, we know some possible scenarios that may happen in the future. In this situation, our goal can be summarized as follows. Before we know which scenario really happens, we first fix an assignment of workers to two tasks. Then, after a scenario is realized, we modify the assignment to maximize the obtained value. Our goal is to minimize the expected amount of modification by choosing the most appropriate assignment at the first stage. In this way, we will be able to minimize the cost of reallocating workers to the two tasks when some undesired events happen.

We describe the situation more formally. Let U be a finite set of workers, and we are facing k scenarios. For each scenario i, the associated value functions $g_i^1 \colon 2^U \to \mathbb{R}$ and $g_i^2 \colon 2^U \to \mathbb{R}$ are given, where $g_i^j(X)$ represents the value obtained by a set X of workers when they are assigned to the task $j \in \{1, 2\}$.

Suppose that we first fix an assignment in such a way that the set X of workers is assigned to the first task, and the set $U \setminus X$ of workers is assigned to the second task. Then, suppose that we learn that the i-th scenario occurs after fixing the assignment. We now change the assignment to maximize the obtained value. Let Y_i maximize $g_i^1(Y) + g_i^2(U \setminus Y)$ which is the value we obtain in the i-th scenario. Then, the magnitude of modification can be measured by the symmetric difference $|X \mathbin{\triangle} Y_i|$.

Let p_i be the probability with which the scenario i realizes. Then, our goal is to find $X \subseteq U$ such that the expected magnitude of modification

$$\sum_{i=1}^{k} p_i |X \triangle Y_i| \tag{4}$$

is minimized, where Y_i maximizes $g_i^1(Y) + g_i^2(U \setminus Y)$.

For the easier presentation, assume that each scenario occurs with the same probability: $p_i = 1/k$ for all $i \in \{1, \dots, k\}$. Further, we assume that the functions $g_i^j \colon 2^U \to \mathbb{R}$ are supermodular to model the synergy effect among workers (see an example below). Supermodularity is a natural assumption, which is also known as convexity in cooperative game theory [23] and known to have a lot of desirable properties. Then, the function $Y \mapsto -(g_i^1(Y)+g_i^2(U\setminus Y))$ is submodular for every $i \in \{1, \dots, k\}$. Therefore, the problem above can be rephrased in the following form: We are given k submodular functions $f_1, \dots, f_k \colon 2^U \to \mathbb{R}$, and our goal is to find $X \subseteq U$ and a minimizer Y_i of f_i for every $i \in \{1, \dots, k\}$ such that

$$\sum_{i=1}^{k} |X \triangle Y_i| \tag{5}$$

is minimized. This is exactly the submodular reassignment problem.

A typical setup arises when the synergy effect is binary in the following sense. Suppose we know that the worker $u \in U$ generates the value $v_j(u) \geq 0$ if she is assigned to the task T_j for $j \in \{1, 2\}$. Furthermore, suppose we know that there is a binary synergy effect such that if two workers $u, u' \in U$ are assigned to the same task, then they together generate the value $w(u, u') \geq 0$.

In this case, if a set $X \subseteq U$ of workers is assigned to T_1 and the complement $U \setminus X$ is assigned to T_2, then we obtain the value

$$\sum_{u \in X} v_1(u) + \sum_{u \in U \setminus X} v_2(u) + \sum_{u,u' \in X} w(u, u') + \sum_{u,u' \in U \setminus X} w(u, u'). \tag{6}$$

We want to maximize this value.

As it turns out, the maximization of (6) can be done by the computation of a minimum s, t-cut in an undirected network. To this end, we construct the following network. Consider a graph with vertex set $U \cup \{s, t\}$, where s and t are two designated vertices such that $s, t \notin U$. The edge set of the graph is

$$\{\{u, u'\} \mid u, u' \in U, u \neq u'\} \cup \{\{s, u\} \mid u \in U\} \cup \{\{t, u\} \mid u \in U\}.$$

The capacity of the edge $\{s, u\}$ for each $u \in U$ is equal to $v_1(u)$, and the capacity of the edge $\{t, u\}$ for each $u \in U$ is equal to $v_2(u)$. The capacity of the edge $\{u, u'\}$ for each $u, u' \in U$ is equal to $w(u, u')$. Then, for a subset $X \subseteq U$ of workers, the capacity of the set $X \cup \{s\}$ is equal to

$$\sum_{u \in X} v_2(u) + \sum_{u \in U \setminus X} v_1(u) + \sum_{u \in X, u' \in U \setminus X} w(u, u'). \tag{7}$$

Since the values in (6) and (7) sum up to

$$\sum_{u \in U} (v_1(u) + v_2(u)) + \sum_{u,u' \in U} w(u, u'), \tag{8}$$

which does not depend on the choice of X, the maximization of (6) is equivalent to the minimization of (7). It is well-known that the cut function of a network is submodular.

So far, we assumed that we know the values $v_1(u), v_2(u), w(u, u')$ for workers $u, u' \in U$ exactly. However, today we only know a rough estimate for them, and we will learn their exact values tomorrow.

In this way, we may also model the following situations in staff scheduling by the network as follows.

Absence of a worker: It is possible that tomorrow a worker will be sick in bed, and cannot come. To model the absence of a worker $u \in U$, we consider a scenario in which the values $v_1(u)$, $v_2(u)$, and $w(u, u')$ for all $u' \in U \setminus \{u\}$ are zero.

Fixing a worker to a specific task: It is possible that tomorrow a worker will be injured, and can only join task T_1 because task T_2 needs a special treatment. To fix a worker $u \in U$ to task T_1, we consider a scenario in which the value $v_1(u)$ is $+\infty$, or a sufficiently large number.

In any case, the submodular reassignment problem formulates the situation where we want to minimize the number of reallocated workers.

We discussed the binary synergy effects above. We may also consider ternary or k-ary synergy effects. In that case, a non-negative value is given for a triple or a k-tuple of workers. Then, the computation of the value is reduced to the minimum s, t-cut computation of a hypergraph, which can be done in polynomial time (e.g. see [7]). Therefore, the problem can still be formulated as the submodular reassignment problem.

With help of hypergraphs, we can model a binary synergy effect that is dependent on the assigned task, too. Namely, suppose there is a binary synergy effect such that if two workers $u, u' \in U$ are assigned to the task i, then they together generate the value $w_i(u, u') \geq 0$. In this case, we use a hyperedge $\{s, u, u'\}$ with capacity $w_1(u, u')$, and a hyperedge $\{t, u, u'\}$ with capacity $w_2(u, u')$.

We are tempted to think about the same problem with three or more tasks. However, the problem is NP-hard. See Section 5.

1.2 Contributions

We prove that the submodular reassignment problem can be solved in strongly polynomial time. To this end, we provide two algorithms. The first algorithm reduces the problem to the submodular function minimization problem. The second algorithm reduces the problem to the maximum flow problem. The second maximum-flow based algorithm is more efficient while the first algorithm reveals a basic property of the problem.

Note that if each function f_i has a unique minimizer, then it is easy to see that the submodular reassignment problem can be solved in strongly polynomial time. This is because we can find a unique minimizer Y_i of each f_i in strongly polynomial time,[1] and an optimal solution X^* to the submodular reassignment problem can be obtained as

$$X^* = \{v \in U \mid |\{i \mid v \in Y_i\}| \geq |\{i \mid v \notin Y_i\}|\}.$$

Therefore, the difficulty of the problem arises from the fact that the number of minimizers of a submodular function can be exponentially large. A trick here is that we can represent the family of minimizers of a submodular function in a compact way via Birkhoff's representation theorem for distributive lattices (see Section 2 for details).

[1] The uniqueness can also be checked in strongly polynomial time by Theorem 2.3.

If the function is not necessarily submodular, the problem is hard even when $k = 1$. This is just a minimization of a set function, and we need to look at the value $f(X)$ for all $X \subseteq U$ to find the optimum.

1.3 Background and Literature

The submodular reassignment problem is a particular example of *two-stage stochastic optimization problems with recourse*. For a detailed treatment of stochastic optimization, refer to the books by Shapiro, Dentcheva and Ruszczyński [22] and Birge and Louveaux [2].

A two-stage stochastic optimization problem with recourse can be described as follows:

$$\begin{aligned} \text{minimize} \quad & \mathop{\mathbb{E}}_{\xi \sim P}[F(x,\xi)] \\ \text{subject to} \quad & x \in \mathcal{X}, \\ & F(x,\xi) = \min\{g(x,y,\xi) \mid y \in \mathcal{Y}(x,\xi)\}, \end{aligned} \tag{9}$$

where P is a probability distribution, $\mathcal{X}, \mathcal{Y}(x,\xi)$ are some sets, and g is some function. Intuitively, x represents the first-stage decision that should be made before the realization of the random variable ξ is known, and y represents the second-stage decision (or recourse) that will be made after ξ is realized. The objective is to minimize the expected cost $\mathbb{E}_\xi[F(x,\xi)]$.

For the computational purpose, we should be careful about how the probability distribution P is specified in the input. In the literature, we see two frequently used representations. The first one is the *finite scenario model* in which the probability distribution has a finite support and is given explicitly. The second one is the *black-box model* in which the probability distribution can be accessed only through a black box (or an oracle), and a sample can be chosen according to the distribution.

In the finite scenario model, suppose that the size of the support is k, and let ξ_i occur with probability p_i for each $i \in \{1,\dots,k\}$. Then, the two-stage stochastic optimization problem (9) can be rephrased as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{k} p_i F(x,\xi_i) \\ \text{subject to} \quad & x \in \mathcal{X}, \\ & F(x,\xi_i) = \min\{g(x,y,\xi_i) \mid y \in \mathcal{Y}(x,\xi_i)\} \quad \forall\, i \in \{1,\dots,k\}. \end{aligned}$$

This form is often called the *deterministic equivalent problem* of the problem (9).

The submodular reassignment problem can be formulated as a two-stage stochastic optimization problem with finite scenarios. The correspondence can be seen as $\mathcal{X} = 2^U$, $x = X$, $\xi_i = f_i$, $y = Y$, $\mathcal{Y}(x,\xi_i) = \arg\min f_i$, and $g(x,y,\xi_i) = |X \triangle Y|$.

One method to tackle the two-stage stochastic optimization problem in the black-box model is the *sample average approximation* (SAA). In this method, we choose a number of samples from the distribution, and approximate the probability distribution by the obtained empirical distribution. Some theoretical studies focus on the number of samples to guarantee a certain error. Thus, the sample average approximation reduces the problem to the finite scenario model by losing optimality. However, Dyer and Stougie [9, 10] proved that the two-stage stochastic linear programming can be solved in polynomial time in the finite scenario model, but the problem is #P-hard in the black-box model.

In the context of theoretical computer science, most of the algorithmic studies for two-stage stochastic optimization problems look at approximation. Among others, Shmoys and Swamy [24] gave a fully polynomial-time approximation scheme for the two-stage

stochastic linear programming in the black-box model. Approximation for two-stage stochastic combinatorial optimization problems has also been studied by, for example, [24, 13, 6].

Robust optimization is another way of dealing with uncertainty in optimization. See [1] for an introduction. Robust optimization usually considers minimizing the cost of the worst-case scenario while stochastic optimization usually aims at minimizing the expected cost. We will comment on the robust counterpart of the submodular reassignment problem in Section 6.

Reoptimization is another concept related to the submodular reassignment problem. In general, the reoptimization problem can be phrased as follows. We are given an instance I of an optimization problem and one of its optimal solutions x. Then, for a slightly modified instance I' of the problem, we make a small change to x so that the resulting solution x' is a good approximation (or even an optimal solution) of I'. Reoptimization has been studied for several combinatorial optimization problems such as the minimum spanning tree problem [5], the traveling salesman problem [16], and the Steiner tree problem [4].

In the submodular reassignment problem, we want to minimize the expected amount of change from X to Y_i. This is similar to the situation in reoptimization.

In our algorithm, we make use of Birkhoff's representation theorem for distributive lattices. The theorem is well-known in combinatorics, but its algorithmic application is not abundant. We only give a few references here: one in the stable matching problem [14], and one in the rectangular layout problem [11].

2 Preliminaries

2.1 Distributive Lattices

In this paper, we will make use of properties of finite distributive lattices. To ease the notation, we minimize the use of terminology in poset theory, and we stick to the terminology in sets and graphs.

For our purpose, a *distributive lattice* is a set family $\mathcal{L} \subseteq 2^U$ that is closed under union and intersection: $X, Y \in \mathcal{L}$ implies $X \cup Y \in \mathcal{L}$ and $X \cap Y \in \mathcal{L}$. This is a partially ordered set with respect to set inclusion $\subseteq$, and has a unique minimal element (i.e., an element that has no proper subset in $\mathcal{L}$) and a unique maximal element (i.e., an element that has no proper superset in $\mathcal{L}$).

Birkhoff's representation theorem is a useful tool for studying distributive lattices, which states the following.

▶ **Theorem 2.1** (Birkhoff's representation theorem [3]). *Let $\mathcal{L} \subseteq 2^U$ be a distributive lattice. There exists a partition of U into $U_0, U_1, \ldots, U_b, U_\infty$, where U_0 and U_∞ can possibly be empty, such that the following hold.*

(1) Every set in $\mathcal{L}$ contains U_0.

(2) Every set in $\mathcal{L}$ has an empty intersection with U_∞.

(3) For every set $X \in \mathcal{L}$, there exists a set $J \subseteq \{1, \ldots, b\}$ of indices such that $X = U_0 \cup \bigcup_{j \in J} U_j$.

(4) There exists a directed acyclic graph $G(\mathcal{L})$ that has the following properties:

(4-1) The vertex set is $\{U_0, U_1, \ldots, U_b\}$;

(4-2) U_0 is a unique sink, i.e., a vertex of out-degree zero, of $G(\mathcal{L})$;

(4-3) For a set Z of vertices in $G(\mathcal{L})$, Z has no out-going edge if and only if $\bigcup_{j \in J} U_j \in \mathcal{L}$, where $J = \{j \mid U_j \in Z\}$.

Figure 1 shows an example.

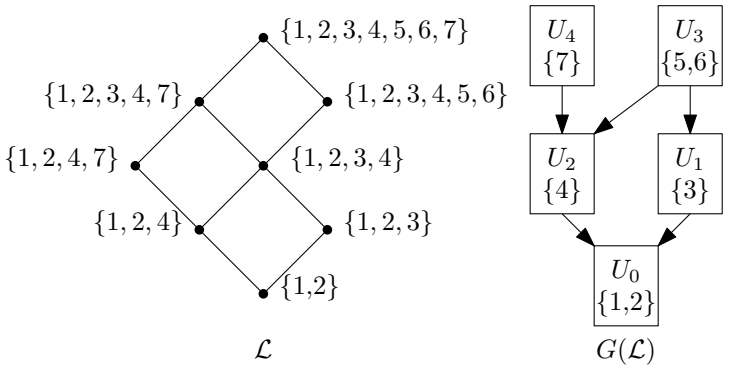


Figure 1 Example for Birkhoff's representation theorem. The left is a distributive lattice $\mathcal{L}$ on $U = \{1, \ldots, 8\}$ (shown by its Hasse diagram), and the right is the directed graph $G(\mathcal{L})$. In this case, $U_\infty = \{8\}$.

For a distributive lattice $\mathcal{L} \subseteq 2^U$, we call the directed graph $G(\mathcal{L})$ above a *compact representation* of $\mathcal{L}$. Note that the size of $G(\mathcal{L})$ is $O(|U|^2)$ while $|\mathcal{L}|$ can be as large as $2^{|U|}$. This justifies the use of word "compact." The graph $G(\mathcal{L})$ is unique if we delete transitive arcs.

2.2 Submodular Functions

Let U be a non-empty finite set. A function $f\colon 2^U \to \mathbb{R}$ is *submodular* if

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \tag{10}$$

for all $X, Y \subseteq U$. A function $f\colon 2^U \to \mathbb{R}$ is *supermodular* if $-f$ is submodular. A *minimizer* of the function f is $X \subseteq U$ such that $f(X) \leq f(Y)$ for all $Y \subseteq U$. The set of minimizers of f is denoted by $\arg\min f$.

For the computational purpose, we define a *value oracle* of a submodular function $f\colon 2^U \to \mathbb{R}$. A value oracle takes $X \subseteq U$ as an input, and returns the value $f(X)$. Assuming that we are given a value oracle, we can minimize a submodular function in polynomial time. The currently fastest algorithm for the submodular function minimization was given by Lee, Sidford, and Wong [15] and runs in $\tilde{O}(n^3\mathrm{EO} + n^4)$ time, where $n = |U|$ and EO is the query time of a value oracle.

The following is a well-known fact on submodular functions.

▶ **Lemma 2.2.** *Let $f\colon 2^U \to \mathbb{R}$ be a submodular function. Then,* $\arg\min f$ *forms a distributive lattice.*

Proof. Let $X, Y \in \arg\min f$, and denote $\alpha = \min\{f(Z) \mid Z \subseteq U\}$. Then, $f(X \cup Y) \geq \alpha$, $f(X \cap Y) \geq \alpha$, and

$$2\alpha = f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y) \geq \alpha + \alpha = 2\alpha.$$

Therefore, $f(X \cup Y) = f(X \cap Y) = \alpha$, which implies that $X \cup Y, X \cap Y \in \arg\min f$. ◀

It is known that a compact representation of the distributive lattice $\arg\min f$ can be constructed in strongly polynomial time. We phrase it as a theorem.

▶ **Theorem 2.3.** *Given a submodular function $f\colon 2^U \to \mathbb{R}$, a compact representation of the distributive lattice* $\arg\min f$ *can be computed in strongly polynomial time.*

Proof Sketch. As a proof sketch, we give a short description of an algorithm.

First, we find a minimal minimizer X_0 of f in strongly polynomial time. This can be done by minimizing the submodular function $\tilde{f}\colon 2^U \to \mathbb{R}$ defined by $\tilde{f}(X) = f(X) + \varepsilon|X|$ for a sufficiently small $\varepsilon > 0$ [17, Notes 10.13]. The set X_0 is a unique sink in $G(\mathcal{L})$. Similarly, we find a unique maximal minimizer that we denote by X_b. Then, we take a maximal chain $X_0 \subsetneq X_1 \subsetneq \cdots \subsetneq X_b$ of minimizers of f. Such a chain can be obtained by minimizing submodular functions repeatedly. In fact, if we are given two minimizers Y and Y' with $Y \subsetneq Y'$, then we can decide whether there exists no minimizer Z such that $Y \subsetneq Z \subsetneq Y'$, by finding a minimal minimizer of a submodular function $f'\colon 2^{Y' \setminus (Y \cup \{v\})} \to \mathbb{R}$ defined by $f'(X) = f(X \cup Y \cup \{v\})$ for each $v \in Y' \setminus Y$. The maximal chain gives a partition of U, i.e., we define $U_0 = X_0$ and $U_i = X_i \setminus X_{i-1}$, $i = 1, \ldots, b$, and $U_\infty = U \setminus X_b$. By Theorem 2.1, it can be argued that this partition forms the vertex set of $G(\mathcal{L})$. Moreover, for $i, j \in \{1, \ldots, b\}$, $G(\mathcal{L})$ has an edge from U_i to U_j if and only if any minimizer of f containing U_i includes U_j, which can be checked by finding a minimal minimizer of a submodular function $f'\colon 2^{U \setminus U_i} \to \mathbb{R}$ defined by $f'(X) = f(X \cup U_i)$. Also, $G(\mathcal{L})$ has an edge from each U_i to U_0 and from U_∞ to each U_i. ◀

There exists a more efficient algorithm for creating a compact representation of the distributive lattice $\arg\min f$ for a submodular function f. For example, it is known that a compact representation of the distributive lattice $\arg\min f$ can be constructed in $\tilde{O}(n^5\mathrm{EO} + n^6)$ time via Orlin's submodular function minimization algorithm [18]. See [17, Notes 10.11 and 10.12].[2]

2.3 Minimum Cuts

Let $G = (V, A)$ be a directed graph, where V is its vertex set and A is its arc set. For an arc $a \in A$, $\partial^+ a$ denotes the tail, and $\partial^- a$ denotes the head.

An *s, t-network* consists of a directed graph $G = (V, A)$ with two designated vertices s, t ($s \neq t$), and a capacity function $c\colon A \to \mathbb{R}_+ \cup \{+\infty\}$, where $\mathbb{R}_+$ is the set of all non-negative real numbers. Such an s, t-network is denoted by $(G, c; s, t)$. Let $U = V \setminus \{s, t\}$. Then, the set function $\kappa\colon 2^U \to \mathbb{R}$ defined as

$$\kappa(X) = \sum_{a \in A\colon \partial^+ a \in X \cup \{s\}, \partial^- a \in (U \setminus X) \cup \{t\}} c(a) \tag{11}$$

is called the *cut function* of the s, t-network $(G, c; s, t)$. The value $\kappa(X)$ is called the *capacity* of an *s, t-cut* $X \cup \{s\}$.

It is well-known and easy to see that the cut function is submodular. For each minimizer X of the cut function, the set $X \cup \{s\}$ is called a *minimum s, t-cut*. A minimum s, t-cut can be deduced from a maximum flow of the network in linear time by constructing the so-called residual network and performing the depth-first search from s. Since the fastest known algorithm for the maximum flow computation takes $O(|V||A|)$ time [19], a minimum s, t-cut can also be computed in $O(|V||A|)$ time.

By Lemma 2.2, the family of minimum s, t-cuts forms a distributive lattice. A compact representation can be constructed from a maximum flow in the s, t-network in linear time. More specifically, from a maximum flow, construct the residual network, and find the strongly-connected-component decomposition of the residual network. The decomposition gives a

[2] It is not clear to us that the algorithm by Lee et al. [15] can be used to find a compact representation of $\arg\min f$.

directed graph which can be turned into a compact representation of the distributive lattice of minimum s,t-cuts [21]. In total, when we construct a compact representation from scratch, the running time is dominated by the maximum flow computation, and this can be done in $O(|V||A|)$ time [19].

We may also consider an undirected s,t-network $(G,c;s,t)$ in which G is an undirected graph instead of a directed graph. An undirected s,t-network can be turned into a directed s,t-network by changing each undirected edge $\{u,v\}$ of G to two directed arcs (u,v) and (v,u), and setting the capacity as $c((u,v)) = c((v,u)) := c(\{u,v\})$.

Similarly, we may consider a minimum s,t-cut of a hypergraph. By a *hypergraph*, we mean a pair $H=(V,E)$ of finite set V called the vertex set and a family $E \subseteq 2^V$ of hyperedges. For a hypergraph $H=(V,E)$ and two designated vertices $s,t \in V$, let $U = V \setminus \{s,t\}$. For a capacity function on the hyperedges $c\colon E \to \mathbb{R}_+$, the cut function $\kappa\colon 2^U \to \mathbb{R}$ is defined as

$$\kappa(X) = \sum_{e \in E\colon e \cap (X \cup \{s\}) \neq \emptyset, e \cap ((U \setminus X) \cup \{t\}) \neq \emptyset} c(e).$$

It is known that the computation of a minimum s,t-cut of a hypergraph $H=(V,E)$ can be reduced to the computation of a minimum s,t-cut of some directed s,t-network $G=(V',A')$, where $|V'| = O(|V|+|E|)$, and $|A'| = O(|E|+\sum_{e \in E}|e|)$. Furthermore, there exists a bijection between the family of minimum s,t-cuts of H and the family of minimum s,t-cuts of G (see [7]). Therefore, the family of all minimum s,t-cuts of a hypergraph forms a distributive lattice, and its compact representation can be obtained in $O(|V'||A'|)$ time.

3 Reduction to Submodular Minimization

3.1 Proof of Submodularity

In this section, we prove that the objective function $\sum_{i=1}^k d_i(X)$ in Problem (3) is submodular.

▶ **Theorem 3.1.** *The function $X \mapsto \sum_{i=1}^k d_i(X)$ is submodular.*

Proof. It is enough to prove that $d_i\colon 2^U \to \mathbb{R}$ is submodular for each $i \in \{1,\dots,k\}$, as the sum of submodular functions is submodular. Recall the definition in (2): $d_i(X) := \min\{|X \triangle Y_i| \mid Y_i \in \arg\min f_i\}$.

Let $X^1, X^2 \subseteq U$, and $Y^j \in \arg\min\{|X^j \triangle Y| \mid Y \in \arg\min f_i\}$ for $j \in \{1,2\}$. Namely, Y^j is a set in $\arg\min f_i$ that is the "closest" to X. Then, $d_i(X^j) = |X^j \triangle Y^j|$ holds.

Suppose that the following is true:

$$|X^1 \triangle Y^1| + |X^2 \triangle Y^2| \geq |(X^1 \cup X^2) \triangle (Y^1 \cup Y^2)| + |(X^1 \cap X^2) \triangle (Y^1 \cap Y^2)|. \tag{12}$$

Then, we obtain

$$\begin{aligned} d_i(X^1) + d_i(X^2) &= |X^1 \triangle Y^1| + |X^2 \triangle Y^2| \\ &\geq |(X^1 \cup X^2) \triangle (Y^1 \cup Y^2)| + |(X^1 \cap X^2) \triangle (Y^1 \cap Y^2)| \\ &\geq d_i(X^1 \cup X^2) + d_i(X^1 \cap X^2), \end{aligned}$$

where the second inequality follows from Lemma 2.2: $Y^1, Y^2 \in \arg\min f_i$ implies $Y^1 \cup Y^2, Y^1 \cap Y^2 \in \arg\min f_i$.

Thus, it is sufficient to prove (12). To this end, we consider the difference of the left-hand side and the right-hand side of (12). Keeping the identities $|A \triangle B| = |A| + |B| - 2|A \cap B|$

and $|A| + |B| = |A \cup B| + |A \cap B|$ for all finite sets A, B in mind, we obtain

$$\begin{aligned}
&\frac{1}{2}\left(|X^1 \triangle Y^1| + |X^2 \triangle Y^2| - |(X^1 \cap X^2) \triangle (Y^1 \cap Y^2)| - |(X^1 \cup X^2) \triangle (Y^1 \cup Y^2)|\right) \\
&= \frac{1}{2}|X^1| + \frac{1}{2}|Y^1| - |X^1 \cap Y^1| + \frac{1}{2}|X^2| + \frac{1}{2}|Y^2| - |X^2 \cap Y^2| \\
&\quad - \frac{1}{2}|X^1 \cap X^2| - \frac{1}{2}|Y^1 \cap Y^2| + |(X^1 \cap X^2) \cap (Y^1 \cap Y^2)| \\
&\quad - \frac{1}{2}|X^1 \cup X^2| - \frac{1}{2}|Y^1 \cup Y^2| + |(X^1 \cup X^2) \cap (Y^1 \cup Y^2)| \\
&= -\left(|X^1 \cap Y^1| + |X^2 \cap Y^2|\right) + |(X^1 \cap X^2) \cap (Y^1 \cap Y^2)| + |(X^1 \cup X^2) \cap (Y^1 \cup Y^2)| \\
&= -\left(|(X^1 \cap Y^1) \cap (X^2 \cap Y^2)| + |(X^1 \cap Y^1) \cup (X^2 \cap Y^2)|\right) \\
&\quad + |(X^1 \cap X^2) \cap (Y^1 \cap Y^2)| + |(X^1 \cup X^2) \cap (Y^1 \cup Y^2)| \\
&= -|(X^1 \cap Y^1) \cup (X^2 \cap Y^2)| + |(X^1 \cup X^2) \cap (Y^1 \cup Y^2)| \\
&= -|(X^1 \cap Y^1) \cup (X^2 \cap Y^2)| + |(X^1 \cap Y^1) \cup (X^1 \cap Y^2) \cup (X^2 \cap Y^1) \cup (X^2 \cap Y^2)| \\
&\geq 0,
\end{aligned}$$

where the last inequality holds because $|A| \leq |A \cup B|$ for any finite sets A and B. This finishes the proof of (12), and thus the submodularity of $\sum_{i=1}^{k} d_i(X)$ holds. ◀

Note that we can also prove the submodularity of d_i by using known results on submodular functions as follows. Define a submodular function $g\colon 2^U \to \mathbb{R} \cup \{+\infty\}$ by

$$g(X) = \begin{cases} 0 & \text{if } X \in \arg\min f_i \\ +\infty & \text{otherwise.} \end{cases}$$

Then, $d_i(X)$ is represented by

$$d_i(X) = \min_{Y \subseteq U} \left(g(Y) + \sum_{v \in U} |\chi_X(v) - \chi_Y(v)| \right),$$

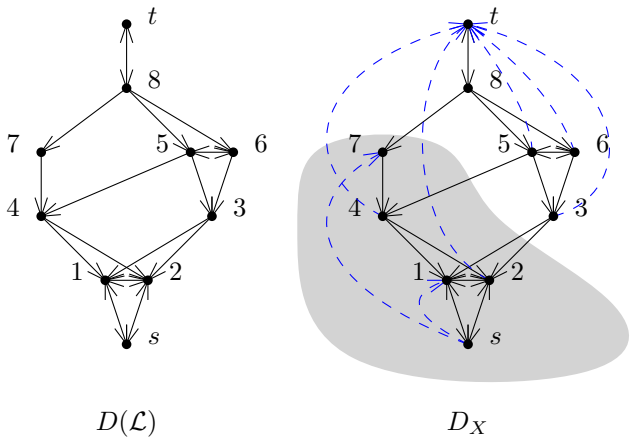
where χ_X, χ_Y are the characteristic vectors of X, Y. The right-hand side is the convolution of a submodular function and a separable convex function, which is known to be submodular (see e.g., [17, Theorem 7.10]). Thus, d_i is submodular.

We here emphasize that for a submodular function $f\colon 2^U \to \mathbb{R}$ and a set $S \subseteq U$, the function $f_{\triangle S}\colon 2^U \to \mathbb{R}$ defined as $f_{\triangle S}(X) = f(X \triangle S)$ is not necessarily submodular. For example, consider the following case: $U = \{1, 2\}, S = \{1\}$ and the submodular function $f\colon 2^U \to \mathbb{R}$ defined as $f(\emptyset) = 0, f(\{1\}) = 2, f(\{2\}) = 2, f(\{1, 2\}) = 3$. Then, $f_{\triangle S}(\emptyset) = f(\{1\}) = 2$, $f_{\triangle S}(\{1\}) = f(\emptyset) = 0$, $f_{\triangle S}(\{2\}) = f(\{1, 2\}) = 3$, and $f_{\triangle S}(\{1, 2\}) = f(\{2\}) = 2$. Therefore, $f_{\triangle S}(\{1\}) + f_{\triangle S}(\{2\}) < f_{\triangle S}(\emptyset) + f_{\triangle S}(\{1, 2\})$, and thus $f_{\triangle S}$ is not submodular.

3.2 Implementation of a Submodular Value Oracle

The goal of the section is to give a polynomial-time algorithm for the submodular reassignment problem. It follows from Theorem 3.1 that the objective function $\sum_{i=1}^{k} d_i(X)$ is submodular. As discussed in Section 2, it suffices to design a value oracle for $\sum_{i=1}^{k} d_i(X)$, that is, to design a polynomial-time algorithm to compute $d_i(X)$ for a given subset X and an integer i.

The algorithm is similar to Picard's method to reduce the maximum closure problem to the maximum flow problem [20]. For the sake of later convenience, we explain the reduction in detail. Refer to Figure 2.



$D(\mathcal{L})$ D_X

Figure 2 Implementation of a submodular value oracle. The example is taken from Figure 1, and $X = \{1, 7\}$. The solid black arcs have infinite capacities, and blue broken arcs have the unit capacities. A minimum s, t-cut of D_X is $\{s, 1, 2, 4, 7\}$, which is shown in gray, and its capacity is two. On the other hand, $Y^* = \{1, 2, 4, 7\} \in \mathcal{L}$ minimizes the value $|X \triangle Y|$ over all $Y \in \mathcal{L}$, and $|X \triangle Y^*| = 2$.

Fix a subset X of U and an integer i in $\{1, 2, \ldots, k\}$. Define $\mathcal{L} := \arg\min f_i$. Recall that computing $d_i(X)$ is to find a subset $Y \in \mathcal{L}$ that minimizes $|X \triangle Y|$.

Let $G(\mathcal{L})$ be a compact representation of $\mathcal{L}$ with a partition $U_0, U_1, \ldots, U_b, U_\infty$ of U. We first construct a directed graph $D(\mathcal{L})$ from $G(\mathcal{L})$ by expanding each vertex in $G(\mathcal{L})$ to a complete graph. More specifically, $D(\mathcal{L})$ has a vertex set $U \cup \{s, t\}$, and its arc set A is defined as follows.

- $(u, v) \in A$ if $u, v \in U_j$ for some j.
- $(u_j, u_{j'}) \in A$ for any $u_j \in U_j$ and $u_{j'} \in U_{j'}$ if $G(\mathcal{L})$ has an arc from U_j to $U_{j'}$.
- $(u, s), (s, u) \in A$ if $u \in U_0$.
- $(u, t), (t, u) \in A$ if $u \in U_\infty$.

The capacity $c(e)$ is set to be $+\infty$ for any $e \in A$. We denote the cut function of the s, t-network $(D(\mathcal{L}), c; s, t)$ by κ. Note that for any subset $Z \subseteq U$, $\kappa(Z)$ is either zero or $+\infty$.

Then, the following lemma holds by Birkhoff's representation theorem (Theorem 2.1).

▶ **Lemma 3.2.** *Let Z be a subset of U. Then, $Z \in \mathcal{L}$ if and only if $\kappa(Z) = 0$.*

Proof. Suppose that $Z \in \mathcal{L}$. Then, Z is partitioned into $\{U_0\} \cup \{U_j \mid j \in J\}$ for some $J \subseteq \{1, \ldots, k\}$. Moreover, there is no outgoing arc from $\{U_0\} \cup \{U_j \mid i \in J\}$ in $G(\mathcal{L})$. By construction, there is no outgoing arc from $Z \cup \{s\}$ in $D(\mathcal{L})$. Hence, $\kappa(Z)$ is equal to zero.

On the other hand, suppose that $\kappa(Z) = 0$. Then, there is no outgoing arc from $Z \cup \{s\}$ in $D(\mathcal{L})$, as each arc has infinite capacity. Hence, it follows that $U_0 \subseteq Z$, $Z \cap U_\infty = \emptyset$, and, for each $j \in \{1, 2, \ldots, b\}$, either $U_j \subseteq Z$ or $Z \cap U_j = \emptyset$. Moreover, letting $J = \{j \in \{1, 2, \ldots, b\} \mid U_j \subseteq Z\}$, we see that there is no outgoing arc from $\{U_0\} \cup \{U_j \mid j \in J\}$ in $G(\mathcal{L})$. This implies that Z is in $\mathcal{L}$. ◀

Using X, we add arcs to $D(\mathcal{L})$; For each element u in X, we add an arc (s, u) with $c((s, u)) = 1$, and, for each element u in $U \setminus X$, we add an arc (u, t) with $c((u, t)) = 1$. We denote the resulting graph by $D_X = (U \cup \{s, t\}, A_X)$. Define κ_X as the cut function of $(D_X, c; s, t)$.

▶ **Lemma 3.3.** *Let Z be a subset of U. If $\kappa_X(Z) < +\infty$, then $Z \in \mathcal{L}$ and $\kappa_X(Z) = |X \triangle Z|$.*

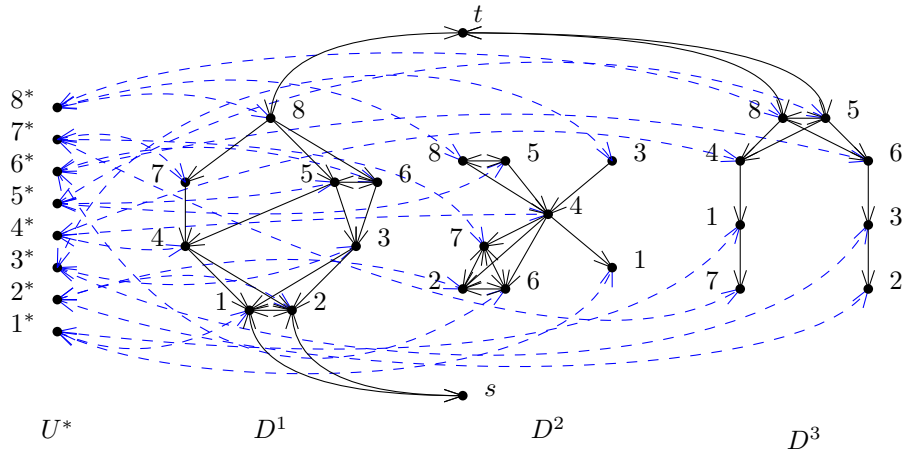


Figure 3 Reduction of the submodular reassignment problem to the maximum flow problem. The solid black arcs have infinite capacities, and the blue broken arcs have unit capacities. In this example, $k = 3$.

Proof. Since $\kappa_X(Z) < +\infty$, we have $\kappa(Z) = 0$ for $D(\mathcal{L})$, and thus $Z \in \mathcal{L}$ by Lemma 3.2. Moreover, we have

$$\kappa_X(Z) = \kappa(Z) + \sum_{u \in Z \setminus X} c((u,t)) + \sum_{v \in X \setminus Z} c((s,v)) = |Z \setminus X| + |X \setminus Z|,$$

which proves the lemma. ◀

These lemmas show that $d_i(X)$ is equal to the capacity of a minimum s,t-cut in D_X, which can be computed in polynomial time. Summarizing the discussion above with Theorem 3.1, we obtain the following theorem.

▶ **Theorem 3.4.** *The submodular reassignment problem can be solved in strongly polynomial time.* ◀

4 Reduction to Maximum Flow

In Section 3, we proved the submodular reassignment problem can be solved in strongly polynomial time by using a submodular function minimization algorithm. However, the submodular function minimization is a big hammer that we like to avoid. Thus, in this section, we will give a reduction of the submodular reassignment problem to the minimum s,t-cut problem (or equivalently via duality theorem, the maximum flow problem) that can be solved more efficiently both in theory and in practice. For an illustration, see Figure 3.

We here note that in our algorithm we need to have a compact representation of $\arg\min f_i$ for each submodular function f_i. In this sense, a submodular function minimization algorithm is actually required. On the other hand, if the submodular functions f_i are represented by cut functions of graphs or hypergraphs, then we can avoid using a submodular function minimization algorithm to obtain compact representations.

For each integer i in $\{1, 2, \ldots, k\}$, we define $\mathcal{L}_i := \arg\min f_i$. Let $G(\mathcal{L}_i)$ be a compact representation of $\mathcal{L}_i$, where the vertex set is $\{U_0^i, U_1^i, \ldots, U_b^i\}$ with $U_\infty^i = U \setminus \bigcup_{j=0}^b U_j^i$. We construct a directed graph $D^i := D(\mathcal{L}_i)$ and a capacity c similarly to Section 3.2. We denote the vertex set of D^i except for s,t by U^i, and the arc set by A^i. Since the vertices of D^i

other than s, t are formed by a copy of U, we denote a vertex of D^i that corresponds to $u \in U$ by u_i. Namely, $U^i = \{u_i \mid u \in U\}$. The cut function of the s, t-network $(D^i, c; s, t)$ is denoted by κ_{D^i}. Then, it follows from Lemma 3.2 that, for each subset X in U^i, $\kappa_{D^i}(X)$ is zero if and only if X is in $\mathcal{L}_i$.

From D^i, $i = 1, \ldots, k$, we construct an s, t-network $(D = (V, A), c; s, t)$ as follows. We identify all s (and t, respectively) in D^i into one vertex s (and t, respectively). We further prepare another copy of U, i.e., $U^* = \{u^* \mid u \in U\}$. Thus, the vertex set V of D is defined by

$$V := \bigcup_{i=1}^{k} U^i \cup U^* \cup \{s, t\}.$$

The arc set A of D consists of A^i and arcs connecting u^* and a copy of u in D^i. That is,

$$A := \bigcup_{i=1}^{k} \left(A^i \cup \{(u^*, u_i), (u_i, u^*) \mid u \in U\} \right).$$

The capacity c on A is defined by

$$c(e) := \begin{cases} +\infty & \text{if } e \in \bigcup_{i=1}^{k} A^i, \\ 1 & \text{otherwise.} \end{cases} \tag{13}$$

Define κ_D as the cut function of $(D, c; s, t)$.

▶ **Lemma 4.1.** *Let Z be a subset of $V \setminus \{s, t\}$ such that $\kappa_D(Z)$ is finite. Define $Y_i = \{u \mid u_i \in Z \cap U^i\}$ for $i = 1, \ldots, k$. Then, Y_i is in $\mathcal{L}_i$.*

Proof. Since $\kappa_D(Z)$ is finite, there is no outgoing arc in $\bigcup_{i=1}^{k} A^i$ from $Z \cup \{s\}$, as each arc has infinite capacity. Hence, we have $\kappa_{D^i}(Z \cap U^i) = 0$ for each $i = 1, \ldots, k$, and thus the corresponding set Y_i is in $\mathcal{L}_i$ by Lemma 3.2. ◀

▶ **Lemma 4.2.** *Let X be a subset of U, and Y_i be a minimizer in $\mathcal{L}_i$ for each integer i in $\{1, 2, \ldots, k\}$. Define $X^* := \{u^* \mid u \in X\}$ and $U^i(Y_i) := \{u_i \mid u \in Y_i\}$. Then, the set $Z := X^* \cup \bigcup_{i=1}^{k} U^i(Y_i)$ satisfies that*

$$\kappa_D(Z) = \sum_{i=1}^{k} |X \triangle Y_i|.$$

Proof. Since Y_i is in $\mathcal{L}_i$, it holds that $\kappa_{D^i}(U^i(Y_i)) = 0$ for each i by Lemma 3.2. Therefore, we have

$$\begin{aligned} \kappa_D(Z) &= \sum_{i=1}^{k} \left(\sum_{u \in X^*, v \notin V(Y_i)} c((u, v)) + \sum_{u \notin X^*, v \in V(Y_i)} c((v, u)) \right) \\ &= \sum_{i=1}^{k} (|X \setminus Y_i| + |Y_i \setminus X|) = \sum_{i=1}^{k} |X \triangle Y_i|. \end{aligned}$$

◀

These lemmas show that finding $X \subseteq U$ and $Y_i \in \arg\min f_i$ for all $i \in \{1, \ldots, k\}$ that minimize $\sum_{i=1}^{k} |X \triangle Y_i|$ is equivalent to finding a minimum s, t-cut in D, which can be done in $O(|V||A|)$ time [19]. Since D contains $O(kn)$ vertices and $O(kn^2)$ arcs, where $n = |U|$, we have the following theorem.

▶ **Theorem 4.3.** *The submodular reassignment problem can be reduced to the minimum s, t-cut problem. Furthermore, it can be solved in $O(k^2 n^3)$ time if we are given a compact representation of $\mathcal{L}_i$ for each i, where $n = |U|$.*

5 Hardness of Variants

5.1 More Than Two Tasks

The submodular reassignment problem deals with assigning workers to two tasks. Therefore, we may also think about more general problem in which we assign workers to m tasks, $m \geq 3$. This generalized setup can be cast into the following optimization problem. Let U be a finite set and we are given mk submodular functions $f_i^j \colon 2^U \to \mathbb{R}$, $i \in \{1, \dots, k\}$, $j \in \{1, \dots, m\}$. Then, our goal is to find $X^j, Y_i^j \subseteq U$ for $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, m\}$ such that

- $X^1, \dots, X^m$ partition U,
- $Y_i^1, \dots, Y_i^m$ partition U for every $i \in \{1, \dots, k\}$, and
- $Y_i^1, \dots, Y_i^m$ minimize $\sum_{j=1}^m f_i^t(Y_i^j)$.

Under these constraints, we want to minimize

$$\sum_{i=1}^{k} \sum_{j=1}^{m} |X^j \mathbin{\triangle} Y_i^j|.$$

Unfortunately, this problem is NP-hard already when $m = 3$, $k = 1$ and the involved submodular functions are cut functions of a graph with unit capacity. This can be proved by a reduction of the minimum 3-terminal cut problem [8].

In the minimum 3-terminal cut problem, we are given an undirected graph $G = (V, E)$ and three designated vertices $s_1, s_2, s_3 \in V$. Then, we want to find a partition of V into three parts V_1, V_2, V_3 such that $s_i \in V_i$ for each $i \in \{1, 2, 3\}$, and the following sum $|E(V_1, V_2)| + |E(V_1, V_3)| + |E(V_2, V_3)|$ is minimized, where $E(V_i, V_j)$ is the set of edges in G that have one endpoint in V_i and the other endpoint in V_j. The minimum 3-terminal cut problem is NP-hard [8].

Given an instance $G = (V, E)$ of the minimum 3-terminal cut problem, we construct an instance of our problem under investigation. Let $U = V \setminus \{s_1, s_2, s_3\}$, $k = 1$ and $m = 3$. For each $j \in \{1, 2, 3\}$, the function $f_1^j \colon 2^U \to \mathbb{R}$ is defined as follows:

$$f_1^j(X) = |E(X \cup \{s_j\}, V \setminus (X \cup \{s_j\}))|.$$

Note that $f_1^{(j)}$ is a cut function, and thus submodular.

Let $(X^1, X^2, X^3, Y_1^1, Y_1^2, Y_1^3)$ be an optimal solution to our problem. Then, Y_1^1, Y_1^2, Y_1^3 partition U and minimize $\sum_{j=1}^3 f_1^j(Y_1^j)$. Therefore, $Y_1^1 \cup \{s_1\}, Y_1^2 \cup \{s_2\}, Y_1^3 \cup \{s_3\}$ yield an optimal solution to the minimum 3-terminal cut problem. Thus, our problem variant with three tasks is NP-hard.

5.2 A Robust Variant

The submodular reassignment problem can be thought of as the "min-sum" problem, or the "median" problem for $\arg\min f_1$, $\arg\min f_2$, $\dots$, $\arg\min f_k$. This motivates us to study the "min-max" version, or the "center" version of the problem. Namely, for given submodular functions $f_1, \dots, f_k \colon 2^U \to \mathbb{R}$, we want to solve the following problem:

$$\underset{X \subseteq U}{\text{minimize}} \quad \max_{i=1,\dots,k} d_i(X), \tag{14}$$

where d_i is defined by (2). This is the robust counterpart of the submodular reassignment problem.

However, the above min-max version is NP-hard as the closest string problem is NP-hard [12]. Here, we give a reduction even when the submodular functions are cut functions of s,t-networks.

We reduce the *closest string* problem to the problem above. In the closest string problem, we are given k strings $\sigma_1,\dots,\sigma_k$ of length n over an alphabet Σ, and want to find a string τ of length n over Σ that minimizes

$$\max_{i=1,\dots,k} d(\sigma_i,\tau), \tag{15}$$

where $d(\sigma_i,\tau)$ denotes the Hamming distance between σ_i and τ:

$$d(\sigma_i,\tau)=|\{j\in\{1,\dots,n\}\mid \sigma_i[j]\neq\tau[j]\}|. \tag{16}$$

It is known that the closest string problem is NP-hard even when $\Sigma=\{0,1\}$ [12].

For our reduction, we create an undirected s,t-network $(G_i,c;s,t)$ based on each σ_i. The vertex set of G_i is $\{1,\dots,n\}\cup\{s,t\}$. The edge set of G_i is

$$\{\{s,j\}\mid\sigma_i[j]=0\}\cup\{\{j,t\}\mid\sigma_i[j]=1\},$$

and each edge has a unit capacity.

Let $X\subseteq\{1,\dots,n\}$. Then, the cut function κ_i of the network is determined as

$$\kappa_i(X)=|\{j\in X\mid\sigma_i[j]=1\}|+|\{j\notin X\mid\sigma_i[j]=0\}|$$

for every $X\subseteq\{1,\dots,n\}$. Therefore, this capacity is equal to the Hamming distance $d(\sigma_i,\tau_X)$, where the string τ_X is constructed from X as $\tau_X[j]=0$ if and only if $j\in X$. Thus, minimizing $\max_{i=1,\dots,k} d(\sigma_i,\tau_X)$ is equivalent to minimizing $\max_{i=1,\dots,k}\kappa_i(X)$. This finishes the proof of NP-hardness.

6 Concluding Remarks

You may think Theorem 3.4 is useless for the algorithmic purpose since it is beaten by Theorem 4.3. However, Theorem 3.4 can be used to solve a more general problem. For example, for another submodular function $f_0\colon 2^U\to\mathbb{R}$, we may consider minimizing $f_0(X)+\sum_{i=1}^k d_i(X)$ where $f_0(X)$ represents the cost associated with the first-stage decision X. Then, Theorem 3.4 (and its proof) can be used to prove that the problem above can be solved in strongly polynomial time by a submodular function minimization algorithm.

As we have seen in the introduction, we can consider the situation in which each scenario happens with non-uniform probability p_i. That is, the problem is to find $X\subseteq U$ and $Y_i\in\arg\min f_i$ for all $i\in\{1,\dots,k\}$ that minimize $\sum_{i=1}^k p_i|X\triangle Y_i|$. Since this problem is equivalent to finding $X\subseteq U$ that minimizes a submodular function $\sum_{i=1}^k p_i d_i(X)$, it can be solved in strongly polynomial time via a submodular function minimization algorithm in the same way as in Section 3. We can also reduce the problem to the minimum s,t-cut problem in the same way as in Section 4 by modifying the definition of the capacity c on A. More precisely, we replace (13) with

$$c(e):=\begin{cases}+\infty & \text{if } e\in\bigcup_{i=1}^k A^i,\\ p_i & \text{if } e=(u^*,u_i) \text{ or } e=(u_i,u^*)\end{cases}$$

in the definition of $(D,c;s,t)$. Then, finding $X\subseteq U$ and $Y_i\in\arg\min f_i$ for all $i\in\{1,\dots,k\}$ that minimize $\sum_{i=1}^k p_i|X\triangle Y_i|$ is equivalent to finding a minimum s,t-cut in D, which can be done in strongly polynomial time.

References

1 Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

2 John R. Birge and François Louveaux. *Introduction to Stochastic Programming (Second Edition)*. Springer Series in Operations Research and Financial Engineering. Springer, 2011.

3 Garrett Birkhoff. Rings of sets. *Duke Math. J.*, 3(3):443–454, 09 1937. doi:10.1215/S0012-7094-37-00334-X.

4 Hans-Joachim Böckenhauer, Karin Freiermuth, Juraj Hromkovic, Tobias Mömke, Andreas Sprock, and Björn Steffen. Steiner tree reoptimization in graphs with sharpened triangle inequality. *J. Discrete Algorithms*, 11:73–86, 2012. doi:10.1016/j.jda.2011.03.014.

5 Nicolas Boria and Vangelis Th. Paschos. Fast reoptimization for the minimum spanning tree problem. *J. Discrete Algorithms*, 8(3):296–310, 2010. doi:10.1016/j.jda.2009.07.002.

6 Jaroslaw Byrka and Aravind Srinivasan. Approximation algorithms for stochastic and risk-averse optimization. *SIAM Journal on Discrete Mathematics*, 32(1):44–63, 2018. doi:10.1137/15M1043790.

7 Chandra Chekuri and Chao Xu. Minimum cuts and sparsification in hypergraphs. *SIAM J. Comput.*, 47(6):2118–2156, 2018. doi:10.1137/18M1163865.

8 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994. doi:10.1137/S0097539792225297.

9 Martin E. Dyer and Leen Stougie. Computational complexity of stochastic programming problems. *Math. Program.*, 106(3):423–432, 2006. doi:10.1007/s10107-005-0597-0.

10 Martin E. Dyer and Leen Stougie. Erratum to: Computational complexity of stochastic programming problems. *Math. Program.*, 153(2):723–725, 2015. doi:10.1007/s10107-015-0935-9.

11 David Eppstein, Elena Mumford, Bettina Speckmann, and Kevin Verbeek. Area-universal and constrained rectangular layouts. *SIAM J. Comput.*, 41(3):537–564, 2012. doi:10.1137/110834032.

12 Moti Frances and Ami Litman. On covering problems of codes. *Theory Comput. Syst.*, 30(2):113–119, 1997. doi:10.1007/s002240000044.

13 Daniel Golovin, Vineet Goyal, Valentin Polishchuk, R. Ravi, and Mikko Sysikaski. Improved approximations for two-stage min-cut and shortest path problems under uncertainty. *Math. Program.*, 149(1-2):167–194, 2015. doi:10.1007/s10107-013-0742-0.

14 Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. Approximation algorithms for the sex-equal stable marriage problem. *ACM Trans. Algorithms*, 7(1):2:1–2:17, 2010. doi:10.1145/1868237.1868239.

15 Yin Tat Lee, Aaron Sidford, and Sam Chiu-wai Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17–20 October, 2015*, pages 1049–1065. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.68.

16 Jérôme Monnot. A note on the traveling salesman reoptimization problem under vertex insertion. *Inf. Process. Lett.*, 115(3):435–438, 2015. doi:10.1016/j.ipl.2014.11.003.

17 Kazuo Murota. *Discrete Convex Analysis*. SIAM, 2003. doi:10.1137/1.9780898718508.

18 James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009. doi:10.1007/s10107-007-0189-2.

19 James B. Orlin. Max flows in $O(nm)$ time, or better. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1–4, 2013*, pages 765–774. ACM, 2013. doi:10.1145/2488608.2488705.

20 Jean-Claude Picard. Maximal closure of a graph and applications to combinatorial problems. *Management Science*, 22(11):1268–1272, 1976. doi:10.1287/mnsc.22.11.1268.

21 Jean-Claude Picard and Maurice Queyranne. On the structure of all minimum cuts in a network and applications. *Mathematical Programming Studies*, 13:8–16, 1980. doi:10.1007/BFb0120902.

22 Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. SIAM, 2009. doi:10.1137/1.9780898718751.

23 Lloyd S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1(1):11–26, 1971. doi:10.1007/BF01753431.

24 David B. Shmoys and Chaitanya Swamy. An approximation scheme for stochastic linear programming and its application to stochastic integer programs. *J. ACM*, 53(6):978–1012, 2006. doi:10.1145/1217856.1217860.