Journal of the Operations Research Society of Japan
Vol. 62, No. 3, July 2019, pp. 109–120

SEPARABLE CONVEX RESOURCE ALLOCATION PROBLEM WITH L1-DISTANCE CONSTRAINT

Norito Minamikawa Akiyoshi Shioura
Tokyo Institute of Technology

(Received August 6, 2018; Revised November 27, 2018)

Abstract Separable convex resource allocation problem aims at finding an allocation of a discrete resource to several activities that minimizes a separable convex function representing the total cost or the total loss. In this paper, we consider the separable convex resource allocation problem with an additional constraint that the L_1-distance between a given vector and a feasible solution is bounded by a given positive constant. We prove that the simplest separable convex resource allocation problem with the L_1-distance constraint can be reformulated as a submodular resource allocation problem. This result implies that the problem can be solved in polynomial time by existing algorithms for the submodular resource allocation problem. We present specialized implementations of the existing algorithms and analyze their running time.

Keywords: Discrete optimization, resource allocation problem, separable convex function, polymatroid constraint

1. Introduction

Resource allocation problem is a problem of finding an optimal allocation of some discrete resources to several activities, in the setting where cost (or loss) occurs according to resource allocation and total cost should be minimized. Investigation of resource allocation problem is initiated by Koopman [12] in 1953, and then various research has been done on the topic for more than fifty years. Resource allocation problem has many variations, which leads to wide range of applications such as investment planning, manpower planning, production planning and optimal armaments planning.

In this paper, we deal with the following resource allocation problem with a separable convex objective function and a simple constraint on the total resource to be allocated. This problem is often referred to as the *simple resource allocation problem* (see, e.g., [9, 11]).

$$\begin{array}{ll} \text{Minimize} & \displaystyle\sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & \displaystyle\sum_{i=1}^{n} x_i = N, \\ & x \in \mathbb{Z}_+^n. \end{array}$$

Here, n and N are positive integers, $f_i : \mathbb{R} \to \mathbb{R}$ is a convex function ($i = 1, \ldots, n$), and $\mathbb{Z}_+^n$ denotes the set of n-dimensional nonnegative integral vectors. We assume that each function f_i is given explicitly or given by a function evaluation oracle that, given x_i, returns the value $f_i(x_i)$ in constant time.

It is well known that the simple resource allocation problem can be solved by a greedy algorithm [7], which runs in $\mathrm{O}(N \log n)$ time. Note that the running time of the greedy algorithm is exponential in the input size $\mathrm{O}(n + \log N)$ of the problem, which means that the running time becomes huge for a problem with a large input size. Galil and Megiddo [6] and Katoh et al. [10] independently developed polynomial-time algorithms for the simple resource allocation problem, both of which run in $\mathrm{O}(n(\log N)^2)$ time. The fastest algorithm so far for the simple resource allocation problem is due to Frederickson and Johnson [3], which runs in $\mathrm{O}(n \log(N/n))$ time. It is known that this time complexity $\mathrm{O}(n \log(N/n))$ is the best possible under a certain standard computation model [8].

While the simple resource allocation problem is the most basic resource allocation problem, more general resource allocation problems with generalized upper bound constraint, nested constraint, tree constraint, and network constraint have been discussed in the literature [9, 11]. Polymatroid constraint is a common generalization of the constraints mentioned above, and the resource allocation problem with the polymatroid constraint is called the submodular resource allocation problem. A greedy algorithm is also applicable to the submodular resource allocation problem [2], and a polynomial-time scaling algorithm is proposed by Hochbaum [8] (see also Moriguchi and Shioura [13]).

In this paper, we consider the simple resource allocation problem with the L_1-distance constraint formulated as follows, where the L_1-distance constraint is a constraint that the L_1-distance from a given vector to a solution vector is bounded by a given constant.

$$\begin{array}{lll} \text{(L1SRA)} & \text{Minimize} & \displaystyle\sum_{i=1}^{n} f_i(x_i) \\ & \text{subject to} & \displaystyle\sum_{i=1}^{n} x_i = N, \\ & & \|x - y\|_1 \le K, \\ & & x \in \mathbb{Z}^n_+. \end{array}$$

Here, K is a nonnegative integer, and y is an n-dimensional nonnegative integral vector with $\sum_{i=1}^{n} y_i = N$.

The resource allocation problem with the L_1-distance constraint arises naturally when re-allocation of a given resource is required. Let us consider a situation where the original allocation of a resource is given by a vector y and we are required to re-allocate the resource. In such a situation it is often the case that we have a constraint that a new allocation x is close to the original allocation y, which can be represented by the L_1-distance constraint $\|x - y\|_1 \le K$. Indeed, such a constraint is used by Freund et al. [4] in the formulation of the bike allocation problem in a bike sharing system.

The aim of this paper is to reveal the combinatorial structure of the problem L1SRA and to show its polynomial-time solvability. For this, we show that L1SRA can be formulated as a submodular resource allocation problem. This result immediately implies the polynomial-time solvability of L1SRA since the submodular resource allocation problem can be solved in polynomial time. Furthermore, we apply to L1SRA the existing algorithms for the submodular resource allocation problem such as the greedy algorithm [2] and the scaling algorithm [8] and analyze the running time of the specialized implementations. In particular, we show that the greedy algorithm and the scaling algorithm for L1SRA run in $\mathrm{O}(\min(N, K) \log n)$ time and $\mathrm{O}(n \log n \min(\log(N/n), \log(K/n)))$ time, respectively.

2. Review of Submodular Resource Allocation Problem

The simple resource allocation problem considered in Introduction is the most fundamental resource allocation problem that has only one constraint on the total resource to be allocated.

Algorithm 1 procedure GREEDY

```
1: $x := (0, 0, \dots, 0)^\top, E' := E$;
2: while $\sum_{i=1}^{n} x_i < N$ do
3:     Find $j^* \in E'$ such that $d_{j^*}(x_{j^*}) = \min_{j \in E'} d_j(x_j)$.
4:     if $x + \chi_{j^*}$ satisfies the polymatroid constraint then
5:         $x_{j^*} := x_{j^*} + 1$;
6:     else
7:         $E' := E' \setminus \{j^*\}$;
8:     end if
9: end while
10: return $x$;
```

In this section, we explain a more general resource allocation problem called the submodular resource allocation problem and present its algorithms.

A set function $\rho : 2^E \to \mathbb{Z}$ is said to be *submodular* if it satisfies the submodular inequality:

$$\rho(S) + \rho(T) \geq \rho(S \cup T) + \rho(S \cap T) \quad (\forall S, T \in 2^E), \tag{2.1}$$

where $E = \{1, 2, \dots, n\}$. We also say that a set function ρ is *monotone nondecreasing* if the inequality $\rho(S) \leq \rho(T)$ holds for every $S, T \in 2^E$ with $S \subseteq T$. A *polymatroid constraint* is a constraint given as

$$x(S) \leq \rho(S) \quad (\forall S \in 2^E)$$

with a monotone nondecreasing submodular function $\rho : 2^E \to \mathbb{Z}$ with $\rho(\emptyset) = 0$, where $x(S) = \sum_{i \in S} x_i$. The simple resource allocation problem with a polymatroid constraint is called the *submodular resource allocation problem.* In the submodular resource allocation problem, we assume $\rho(E) = N$ without loss of generality.

In the following, we explain a greedy algorithm [2] and a scaling algorithm [8] as two fundamental algorithms for the submodular resource allocation problem. These algorithms are used to solve L1SRA in Section 3.

We first explain a greedy algorithm (see Algorithm 1). For each $i \in E$ and $x_i \in \mathbb{Z}_+$, we denote the increment of function f_i as

$$d_i(x_i) = f_i(x_i + 1) - f_i(x_i).$$

We also denote by $\chi_j \in \{0, 1\}^n$ the j-th unit vector. The greedy algorithm starts with an initial solution given by $x = (0, 0, \dots, 0)^\top$. In each iteration, some variable x_i with the minimum value of the increment $d_i(x_i)$ is increased by one. This step is repeated until the sum of all components of vector x is equal to N. The time complexity of this algorithm is $\mathrm{O}(N(\log n + F))$, where F denotes the time required to check whether a given vector satisfies a polymatroid constraint.

We then describe a scaling algorithm for the submodular resource allocation problem [8]. In the scaling algorithm, we increase variables by multiple units in each iteration, while in the greedy algorithm variables are increased by single unit.

This scaling algorithm consists of the main routine named **procedure** SCALING and the subroutine named **procedure** SM-INCREMENT(s, l) (see Algorithms 2 and 3). In **procedure** SM-INCREMENT(s, l), where $s \in \mathbb{Z}_+$ and $l \in \mathbb{Z}^n_+$, we start with the initial solution l and iteratively increment some variable by s units; if it is infeasible, then we find maximum feasible increment α and increase the variable by α units. **procedure** SCALING

Algorithm 2 procedure SM-INCREMENT(s, l)

```
1: $x := l, E' := E$;
2: while $E' \neq \emptyset$ do
3:     Find $j^* \in E'$ such that $d_{j^*}(x_{j^*}) = \min_{j \in E'} d_j(x_j)$.
4:     if $x + s \cdot \chi_{j^*}$ satisfies the polymatroid constraint then
5:         $x := x + s \cdot \chi_{j^*}$;
6:     else
7:         $\alpha := \max\{\alpha' \mid x + \alpha' \cdot \chi_{j^*}$ satisfies the polymatroid constraint$\}$;
8:         $x := x + \alpha \cdot \chi_{j^*}$, $E' := E' \setminus \{j^*\}$;
9:     end if
10: end while
11: return $x^{(s)} := x$;
```

Algorithm 3 procedure SCALING

```
1: $s := \lceil N/2n \rceil,\ l := (0, 0, \ldots, 0)^\top$;
2: while $s \geq 2$ do
3:     Call SM-INCREMENT$(s, l)$, and let $x^{(s)}$ be its output;
4:     Let $l$ be defined by $l_i := \max(x_i^{(s)} - s, 0)$ for each $i \in E$;
5:     $s := \lceil s/2 \rceil$;
6: end while
7: Call SM-INCREMENT$(1, l)$, and let $x^*$ be its output;
8: return $x^*$;
```

repeatedly executes **procedure** SM-INCREMENT(s, l), where s and l are initially set to $s = \lceil N/2n \rceil$ and $l = (0, 0, \ldots, 0)^\top$, and in each iteration the step size s is gradually decreased and the vector l is updated. The time complexity of **procedure** SCALING is $\mathrm{O}(n(\log n + \tilde{F}) \log(N/n))$, where $\tilde{F}$ denotes the time required to compute α in Step 7. Note that $\tilde{F} = \mathrm{O}(F \log n)$ since α can be computed by binary search.

We can use any lower bound of an optimal solution as an initial solution of these algorithms instead of the zero vector (see, e.g., [5]). As a candidate of such an initial vector, we can use any vector b that satisfies $x \geq b$ for every feasible solution x of the problem. For example, this condition is satisfied by the vector b given by

$$b_i = \rho(E) - \rho(E \setminus \{i\}) \qquad (i \in E).$$

The time complexity of the modified greedy algorithm and the modified scaling algorithm depend on $\tilde{N} = \rho(E) - b(E)$ instead of the parameter N and given as $\mathrm{O}(\tilde{N}(\log n + F))$ and $\mathrm{O}(n(\log n + \tilde{F}) \log(\tilde{N}/n))$, respectively.

3. Structure and Algorithms of L1SRA

We show that L1SRA can be formulated as a submodular resource allocation problem and can be solved efficiently.

3.1. Connection with submodular resource allocation problem

To show that L1SRA can be formulated as a submodular resource allocation problem, we use a set function $\rho : 2^E \to \mathbb{Z}$ defined by

$$\rho(S) = \begin{cases} 0 & (\text{if } S = \emptyset), \\ N & (\text{if } S = E), \\ \min(N, y(S) + k) & (\text{otherwise}), \end{cases} \tag{3.1}$$

where $k = \lfloor (K/2) \rfloor$.

Lemma 3.1. (i) *The function ρ is monotone nondecreasing and submodular.*
(ii) *The feasible region $R \subseteq \mathbb{Z}^n$ of L1SRA is equal to the set $R_\rho \subseteq \mathbb{Z}^n$ given by*

$$R_\rho = \{x \in \mathbb{Z}^n_+ \mid x(S) \le \rho(S)\ (\forall S \in 2^E),\ x(E) = N\}.$$

Proof. [Proof of (i)] The function ρ is monotone nondecreasing by definition. We prove that ρ satisfies the submodular inequality (2.1) for every $S, T \in 2^E$. Recall that y is a nonnegative vector with $y(E) = N$. The submodular inequality holds obviously if one of S and T is in $\{\emptyset, E\}$. In the following, we discuss the case where S and T are nonempty proper subsets of E.

Suppose that $\rho(S) = N$. Then we obtain $\rho(S \cup T) = N$, and therefore it holds that

$$\rho(S) + \rho(T) = N + \min(N, y(T) + k) \ge N + \min(N, y(S \cap T) + k) \ge \rho(S \cup T) + \rho(S \cap T).$$

The proof for the case $\rho(T) = N$ is similar. We then assume $\rho(S) = y(S) + k$ and $\rho(T) = y(T) + k$. Then it holds that

$$\begin{aligned} \rho(S) + \rho(T) &= (y(S) + k) + (y(T) + k) \\ &= (y(S \cup T) + k) + (y(S \cap T) + k) \\ &\ge \min(N, y(S \cup T) + k) + \min(N, y(S \cap T) + k) \\ &\ge \rho(S \cup T) + \rho(S \cap T). \end{aligned}$$

Therefore, the function ρ satisfies the submodular inequality.
[Proof of (ii)] We first show that $x \in R_\rho$ holds for all $x \in R$. For $x \in R$, the vector x is nonnegative and satisfies $x(E) = N$ since R is the feasible region of L1SRA. Therefore, it suffices to show that x satisfies the polymatroid constraint $x(S) \le \rho(S)$ $(S \in 2^E)$. The polymatroid constraint holds obviously if $S = \emptyset$ and $S = E$. Therefore, in the following, we discuss the case with $\emptyset \subset S \subset E$ and show the inequality $x(S) \le \rho(S) = \min(N, y(S) + k)$.

This inequality is equivalent to the two inequalities $x(S) \le N$ and $x(S) \le y(S) + k$. The former inequality follows from $x(E) = N$ and $x \ge 0$. We use the L_1-distance constraint $\|x - y\|_1 \le K$ in order to show the latter inequality $x(S) \le y(S) + k$. The L_1-distance $\|x - y\|_1$ is an even number since $x(E) = y(E)$. Therefore, it holds that $\|x - y\|_1 \le 2\lfloor (K/2) \rfloor = 2k$. Since the sum of all components of the vector $x - y$ is zero, it holds that

$$\sum_{i \in E} \max(0, x(i) - y(i)) = \sum_{i \in E} \max(0, -x(i) + y(i)).$$

Consequently, we obtain

$$\begin{aligned} 2k &\ge \|x - y\|_1 \\ &= \sum_{i \in E} \max(0, x(i) - y(i)) + \sum_{i \in E} \max(0, -x(i) + y(i)) \\ &= 2 \sum_{i \in E} \max(0, x(i) - y(i)). \end{aligned}$$

By the inequality above, it holds that

$$x(S) - y(S) \le \sum_{i \in S} \max(0, x(i) - y(i)) \le \sum_{i \in E} \max(0, x(i) - y(i)) \le k.$$

This inequality implies $x \in R_\rho$.

We then show that $x \in R$ holds for all $x \in R_\rho$. For $x \in R_\rho$, it is easy to see that x satisfies the constraints of L1SRA, except for the L_1-distance constraint. In the following, we show that the vector x satisfies the L_1-distance constraint $\|x - y\|_1 \le K$.

We define subsets $S_+, S_- \subseteq E$ by

$$S_+ = \{i \in E \mid x_i \ge y_i\}, \quad S_- = \{i \in E \mid x_i < y_i\}.$$

It is noted that $S_+ \cap S_- = \emptyset$ and $S_+ \cup S_- = E$. Also, the set S_+ is nonempty since $x(E) = y(E)$. If $S_- = \emptyset$, then we have $x = y$ since $x(E) = y(E) = N$, implying that $\|x-y\|_1 = 0 \le K$. In the following, we assume $S_- \ne \emptyset$ and prove the inequality $\|x-y\|_1 \le K$.

It holds that

$$\begin{aligned} \|x - y\|_1 &= \sum_{i \in S_+} (x_i - y_i) - \sum_{j \in S_-} (x_j - y_j) \\ &= (x(S_+) - y(S_+)) - (x(S_-) - y(S_-)) \\ &= x(S_+) - x(S_-) - y(S_+) + y(S_-). \end{aligned}$$

Since $x(S_+) + x(S_-) = y(S_+) + y(S_-) = N$, it follows that

$$\|x - y\|_1 = 2x(S_+) - 2y(S_+). \tag{3.2}$$

Since the vector x satisfies the submodular constraint $x(S) \le \rho(S)$ for $S \subseteq E$, we obtain

$$x(S_+) \le \rho(S_+) = \min(N, y(S_+) + k) \le y(S_+) + k,$$

which, combined with (3.2), implies

$$\|x - y\|_1 = 2(x(S_+) - y(S_+)) \le 2k \le K.$$

This concludes the proof of $x \in R_\rho$. □

From Lemma 3.1 the next theorem follows immediately.

Theorem 3.1. *L1SRA can be reformulated as a submodular resource allocation problem with a polymatroid constraint associated with the submodular function* $\rho : 2^E \to \mathbb{Z}$ *in* (3.1).

Remark 3.1. Theorem 3.1 shows that if we add the L_1-distance constraint to the simple resource allocation problem, then the resulting problem can be reformulated as a submodular resource allocation problem. On the other hand, the example below shows that if the L_1-distance constraint is added to a slightly more general resource allocation problem, then the resulting problem cannot be reformulated as a submodular resource allocation problem.

As a more general resource allocation problem, we consider the resource allocation problem with the *generalized upper bound constraint*; the generalized upper bound constraint is a constraint of the form $x(S_j) \le b_j$ $(j = 1, 2, \ldots, m)$, where $\{S_1, S_2, \ldots, S_m\}$ is a partition of the set E and $b_1, b_2, \ldots, b_m$ are nonnegative integers. If the L_1-distance constraint is added to the problem, then the feasible region is given as the set of nonnegative integral vectors $x \in \mathbb{Z}^n_+$ satisfying

$$\begin{aligned} &x(E) = N, \\ &\|x - y\|_1 \le K, \\ &x(S_j) \le b_j \quad (j = 1, 2, \ldots, m). \end{aligned}$$

As a concrete example, we consider the following special case with $E = \{1, 2, 3, 4\}$:

$$\begin{aligned} &x_1 + x_2 + x_3 + x_4 = 4, \\ &|x_1 - 1| + |x_2 - 1| + |x_3 - 1| + |x_4 - 1| \le 2, \\ &x_1 + x_2 \le 2, \;\; x_3 + x_4 \le 4. \end{aligned}$$

Assume, to the contrary, that this feasible region can be represented by a polymatroid constraint. Then, the set function $\rho : 2^E \to \mathbb{Z}$ defined by

$$\rho(S) = \max\{x(S) \mid x \in \mathbb{Z}^4_+ \text{ is a feasible solution}\} \quad (S \in 2^E)$$

must be a submodular function (see, e.g., [5]). By the constraint $x_1 + x_2 \le 2$, it holds that

$$\rho(\{1, 2\}) \le 2.$$

Since the vectors $(0, 2, 1, 1)$ and $(1, 1, 2, 0)$ are feasible solutions, we obtain

$$\rho(\{2\}) = 2, \;\; \rho(\{2, 3\}) = 3, \;\; \rho(\{1, 2, 3\}) = 4.$$

Therefore, for $S = \{1, 2\}, T = \{2, 3\}$, it holds that

$$\rho(S) + \rho(T) \le 5 < 6 = \rho(S \cup T) + \rho(S \cap T).$$

Consequently, the set function ρ does not satisfy the submodular inequality (2.1), a contradiction. This implies that the feasible region given above cannot be represented by a polymatroid constraint. □

Remark 3.2. It can be shown that the set of vectors satisfying the L_1-distance constraint $\|x - y\|_1 \le K$ has a nice structure called a bisubmodular polyhedron. This fact, however, does not imply the statement of Theorem 3.1. That is, the intersection of a bisubmodular polyhedron and a hyperplane of the form $\sum_{i \in E} x_i = N$ cannot be represented by a polymatroid constraint, as shown below.

We denote $3^E = \{(X, Y) \mid X, Y \subseteq E, \; X \cap Y = \emptyset\}$. A function $\mu : 3^E \to \mathbb{R}$ is called a *bisubmodular function* if it satisfies the following inequality:

$$\begin{aligned} \mu(X_1, Y_1) + \mu(X_2, Y_2) \ge \mu((X_1 \cup X_2) \setminus (Y_1 \cup Y_2), (Y_1 \cup Y_2) \setminus (X_1 \cup X_2)) \\ + \mu(X_1 \cap X_2, Y_1 \cap Y_2) \qquad (\forall (X_1, Y_1), (X_2, Y_2) \in 3^E). \end{aligned}$$

A *bisubmodular polyhedron* is a polyhedron given as follows by using a bisumobular function μ:

$$P_*(\mu) = \{x \in \mathbb{R}^n \mid x(X) - x(Y) \le \mu(X, Y) \; (\forall (X, Y) \in 3^E)\}.$$

It is known that $P_*(\mu)$ is an integral polyhedron for an integer-valued bisubmodular function μ [5]. Optimization of a linear function over a bisubmodular polyhedron can be solved by a certain greedy algorithm, and minimization of a separable convex function over an integral bisubmodular polyhedron can be solved in polynomial time (see, e.g., [5]).

Note that the set of vectors satisfying the L_1-distance constraint $\|x - y\|_1 \leq K$ is represented by a bisubmodular polyhedron $P_*(\mu)$ associated with the bisubmodular function $\mu : 3^E \to \mathbb{R}$ given by $\mu(X, Y) = K + y(X) - y(Y)$ $(\forall (X, Y) \in 3^E)$; bisubmodularity of this μ is easy to see from the definition.

In the following, we show by a concrete example that the intersection of an integral bisubmodular polyhedron and a hyperplane of the form $\sum_{i \in E} x_i = N$ cannot be represented by a polymatroid constraint in general.

We consider the convex hull $\overline{S} \subseteq \mathbb{R}^4$ of the set

$$S = \{(0,0,0,0), (1,1,0,0), (0,0,1,1), (1,1,1,1)\}.$$

Each vector in the set S corresponds to a *matchable* set of an undirected graph with vertex set $\{1,2,3,4\}$ and edge set $\{(1,2),(3,4)\}$; a vertex set of a graph is said to be matchable if there exists a matching that exactly covers the vertex set. It is known that given an undirected graph, the convex hull of the characteristic vectors of all matchable sets is an integral bisubmodular polyhedron [1]. Intersection of the convex hull $\overline{S}$ and the hyperplane $x_1 + x_2 + x_3 + x_4 = 2$ is given as the convex hull of the sets $\{(1,1,0,0), (0,0,1,1)\}$, and it is not difficult to see that this set cannot be represented by a polymatroid constraint. □

3.2. Algorithms for L1SRA

We present algorithms for solving L1SRA and analyze the running time. Theorem 3.1 implies that we can apply to L1SRA the greedy algorithm and the scaling algorithm in Section 2. We analyze the running time of the algorithms.

Theorem 3.2. *L1SRA can be solved by the greedy algorithm and the scaling algorithm in* $\mathrm{O}(N \log n)$ *time and in* $\mathrm{O}(n \log n \log(N/n))$ *time, respectively.*

Proof. To obtain the bound $\mathrm{O}(N \log n)$ for the greedy algorithm, it suffices to show that $F = \mathrm{O}(1)$, i.e., we can check in constant time whether a given vector x in each iteration of the algorithm satisfies the polymatroid constraint associated with the submodular function ρ in (3.1).

By keeping the value $x(E)$ during the run of the algorithm, we can check the constraint $x(E) \leq \rho(E)$ in constant time. In the case where the set S is nonempty proper subsets of E, we can check the polymatroid constraint $x(S) \leq \rho(S) = \min(N, y(S) + k)$ $(\emptyset \subset \forall S \subset E)$, by checking the two inequalities $x(S) \leq N$ and $x(S) \leq y(S) + k$ for each S. Since the vector x is nonnegative, $x(S) \leq N$ holds automatically if the constraint $x(E) \leq \rho(E) = N$ holds. On the other hand, we can rewrite the inequality $x(S) \leq y(S) + k$ as

$$\sum_{i \in S} (x_i - y_i) \leq k.$$

Since the maximum value of the left-hand side is $\sum_{i: x_i > y_i} (x_i - y_i)$, it suffices to check the inequality

$$\sum_{i: x_i > y_i} (x_i - y_i) \leq k.$$

We can check the inequality by keeping the value $\sum_{i: x_i > y_i} (x_i - y_i)$ in each iteration. It is noted that the set $S^* = \{i \in E \mid x_i > y_i\}$ is not equal to the set E in each iteration since $x(E) \leq N = y(E)$. It is easy to see that we can update the value $\sum_{i: x_i > y_i} (x_i - y_i)$ in constant time whenever the vector x is updated. Therefore, we can decide in $F = \mathrm{O}(1)$ time whether a vector x satisfies the polymatroid constraint.

To obtain the bound $\mathrm{O}(n \log n \log(N/n))$ for the scaling algorithm, we also need to show that $\tilde{F} = \mathrm{O}(1)$, i.e., we can compute in constant time the maximum feasible increment α of the variable x_j. Since the maximum feasible increment α is computed by

$$\alpha = \min(N - x(E), k - \sum_{i:x_i>y_i} (x_i - y_i) - \min(0, x_j - y_j)),$$

the discussion above shows that the value α can be computed in constant time, i.e., $\tilde{F} = \mathrm{O}(1)$. □

In the following, we show that the running time of the greedy algorithm and the scaling algorithm can be made faster in the case where K is smaller than N. Our idea is to replace the initial solution of the algorithms, which is originally set to the zero vector (see Section 2).

Consider a typical situation where K is much smaller than N. Since an optimal solution x^* of L1SRA satisfies $\|x^* - y\|_1 \leq K$ by the L_1-distance constraint, the vector x^* is much closer to y than to the zero vector. We see from this observation that it makes sense that the initial solution is set to some vector close to y instead of the zero vector. The initial solution of the algorithms, on the other hand, must be a lower bound of some optimal solution of L1SRA (see the discussion in Section 2). Since the vector y is not a lower bound of any optimal solution for L1SRA in general, we cannot use y as an initial solution. Hence, as a good initial solution we need a vector that is close to y and is a lower bound of some optimal solution for L1SRA.

The next lemma shows that such a vector can be obtained by solving a certain optimization problem. Note that any vector $x \in \mathbb{Z}^n_+$ with $\sum_{i=1}^n x_i = N - k$ and $x \leq y$ satisfies the L_1-distance constraint $\|x - y\|_1 \leq K$.

Lemma 3.2. *Let x' be an optimal solution of the following optimization problem:*

$$\begin{array}{lll} (\mathrm{SRA}^-) & \text{Minimize} & \displaystyle\sum_{i=1}^n f_i(x_i) \\ & \text{subject to} & \displaystyle\sum_{i=1}^n x_i = N - k, \\ & & x \leq y, \\ & & x \in \mathbb{Z}^n_+. \end{array}$$

Then, there exists an optimal solution x^* of L1SRA *with $x^* \geq x'$.*

Proof. Let x^* be an optimal solution of L1SRA, and assume that the value $\sum_{i=1}^n \max(0, x'_i - x^*_i)$ is the minimum among all optimal solutions. Since $\sum_{i=1}^n \max(0, x'_i - x^*_i) \leq 0$ implies $x^* \geq x'$, we assume $\sum_{i=1}^n \max(0, x'_i - x^*_i) > 0$ and derive a contradiction.

By the assumption, there exists $h \in E$ with $x'_h > x^*_h$. We define

$$S'_+ = \{i \in E \mid x'_i \geq x^*_i\}, \quad S'_- = \{i \in E \mid x'_i < x^*_i\}.$$

Then, $h \in S'_+$ holds. Since

$$\sum_{i=1}^n x'_i = N - k < N = \sum_{i=1}^n x^*_i,$$

we have $S'_- \neq \emptyset$. In the following, we show the existence of $j \in S'_-$ with $x'_j < y_j$.

We have

$$\sum_{i \in S'_+} (y_i - x'_i) < \sum_{i \in S'_+} (y_i - x^*_i) \leq \sum_{i \in E} \max(0, y_i - x^*_i),$$

where the first strict inequality is by the definition of S'_+ and $h \in S'_+$. Since the vector x^* satisfies the L_1-distance constraint $\|x^* - y\|_1 \leq K$, we can obtain the inequality $\sum_{i=1}^n \max(0, y_i - x^*_i) \leq k$, as in the proof of Theorem 3.1, from which follows that

$$\sum_{i \in S'_+} (y_i - x'_i) < k. \tag{3.3}$$

On the other hand, it holds that

$$\begin{aligned}\sum_{i \in S'_+} (y_i - x'_i) + \sum_{i \in S'_-} (y_i - x'_i) &= \sum_{i=1}^n (y_i - x'_i) \\ &= \sum_{i=1}^n y_i - \sum_{i=1}^n x'_i \\ &= N - (N - k) = k,\end{aligned}$$

which, together with (3.3), implies that $\sum_{i \in S'_-} (y_i - x'_i) > 0$. This inequality shows that $x'_j < y_j$ holds for some $j \in S'_-$.

By the convexity of functions f_h and f_j and the inequalities $x'_h > x^*_h$ and $x'_j < x^*_j$, it holds that

$$f_h(x^*_h + 1) - f_h(x^*_h) \leq f_h(x'_h) - f_h(x'_h - 1), \tag{3.4}$$

$$f_j(x'_j + 1) - f_j(x'_j) \leq f_j(x^*_j) - f_j(x^*_j - 1). \tag{3.5}$$

Denoting $f(x) = \sum_{i=1}^n f_i(x_i)$ $(x \in \mathbb{Z}^n)$, we have the following inequality by (3.4) and (3.5):

$$f(x^*) + f(x') \geq f(x^* + \chi_h - \chi_j) + f(x' - \chi_h + \chi_j). \tag{3.6}$$

The vector $x' - \chi_h + \chi_j$ is a feasible solution of SRA$^-$ since $x'_j < y_j$, and therefore it holds that $f(x') \leq f(x' - \chi_h + \chi_j)$. This inequality and (3.6) imply $f(x^*) \geq f(x^* + \chi_h - \chi_j)$. On the other hand, the vector $x^* + \chi_h - \chi_j$ satisfies the L_1-distance constraint since the inequality $y_h \geq x'_h > x^*_h$ holds, and therefore $x^* + \chi_h - \chi_j$ is a feasible solution of L1SRA, from which follows the inequality $f(x^*) \leq f(x^* + \chi_h - \chi_j)$. Hence, it holds that $f(x^* + \chi_h - \chi_j) = f(x^*)$, implying that the vector $x^* + \chi_h - \chi_j$ is also an optimal solution of L1SRA, a contradiction to the choice of x^* since $x'_h > x^*_h$. Therefore, there exists an optimal solution x^* of L1SRA with $x^* \geq x'$. □

By Lemma 3.2, L1SRA can be solved by the greedy algorithm and the scaling algorithm by using an optimal solution of SRA$^-$ as an initial solution. We estimate the time complexity of the resulting algorithms.

An optimal solution of SRA$^-$ can be found in $\mathrm{O}(\min(K \log n, n \log(K/n)))$ time since SRA$^-$ is essentially equivalent to the simple resource allocation problem. If we use an optimal solution of SRA$^-$ as an initial solution b of the greedy algorithm and the scaling algorithm, the value of the parameter $\tilde{N}$ (see Section 2) satisfies

$$\tilde{N} = \rho(E) - b(E) = N - (N - k) = k = \mathrm{O}(K).$$

Therefore, the time complexity of the greedy algorithm is

$$\mathrm{O}(K\log n)+\mathrm{O}(K\log n)=\mathrm{O}(K\log n),$$

and the time complexity of the scaling algorithm is

$$\mathrm{O}(n\log(K/n))+\mathrm{O}(n\log n\log(K/n))=\mathrm{O}(n\log n\log(K/n)).$$

From the discussion above and Theorem 3.2, we obtain the following result:

Theorem 3.3. *L1SRA can be solved by the greedy algorithm and the scaling algorithm in* $\mathrm{O}(\min(N,K)\log n)$ *time and* $\mathrm{O}(n\log n\min(\log(N/n),\log(K/n)))$ *time, respectively, by using the zero vector or an optimal solution of SRA*$^-$ *as an initial solution of the algorithm.*

Acknowledgements

The authors thank the anonymous referees for careful reading of the manuscript and valuable comments. This work is partially supported by JSPS/MEXT KAKENHI Grand Numbers 15K00030, 18K11177.

References

[1] A. Bouchet and W.H. Cunningham: Delta-matroids, jump systems, and bisubmodular polyhedra. *SIAM Journal on Discrete Mathematics*, **8** (1995), 17–32.

[2] A. Federgruen and H. Groenevelt: The greedy procedure for resource allocation problems: Necessary and sufficient conditions for optimality. *Operations Research*, **34** (1986), 909–918.

[3] G.N. Frederickson and D.B. Johnson: The complexity of selection and ranking in $X+Y$ and matrices with sorted columns. *Journal of Computer and System Sciences*, **24** (1982), 197–208.

[4] D. Freund, S.G. Henderson, and D.B. Shmoys: Minimizing multimodular functions and allocating capacity in bike-sharing systems. In *Proceedings of the 19th Conference on Integer Programming and Combinatorial Optimization (IPCO'17)*, (2017), 186–198.

[5] S. Fujishige: *Submodular Functions and Optimization, 2nd Edition* (Elsevier, Amsterdam, 2005).

[6] Z. Galil and N. Megiddo: A fast selection algorithm and the problem of optimum distribution of effort. *Journal of ACM*, **26** (1979), 58–64.

[7] O. Gross: A class of discrete type minimization problems. Research Memorandum RM-1644, Rand Corporation (1956).

[8] D.S. Hochbaum: Lower and upper bounds for the allocation problem and other nonlinear optimization problems. *Mathematics of Operations Research*, **19** (1994), 390–409.

[9] T. Ibaraki and N. Katoh: *Resource Allocation Problems: Algorithmic Approaches* (MIT Press, Cambridge, 1988).

[10] N. Katoh, T. Ibaraki, and H. Mine: A polynomial time algorithm for the resource allocation problem with a convex objective function. *Journal of Operational Research Society*, **30** (1979), 449–455.

[11] N. Katoh, A. Shioura, and T. Ibaraki: Resource allocation problems. In P.M. Pardalos, D.Z. Du, and R.L. Graham (eds.): *Handbook of Combinatorial Optimization* (Springer, New York, 2013), 2897–2988.

[12] O. Koopman: The optimum distribution of effort. *Journal of Operations Research Society of America*, **1** (1953), 52–63.

[13] S. Moriguchi and A. Shioura: On Hochbaum's proximity-scaling algorithm for the general resource allocation problem. *Mathematics of Operations Research*, **29** (2004), 394–397.

Norito Minamikawa
Department of Industrial Engineering and Economics
Tokyo Institute of Technology
Oh-okayama 2-12-1, Meguro-ku
Tokyo 152-8550, Japan
E-mail: `minamikawa.n.aa@m.titech.ac.jp`