Fair Rate Allocation over A Generalized Symmetric Polymatroid with Box Constraints

Peng-Jun Wan[*†], Zhu Wang[‡], Huaqiang Yuan[*], Jinling Zhang[§], and Xufei Mao[*]
[*] School of Computer Science, Dongguan University of Technology, P. R. China
[†] Department of Computer Science, Illinois Institute of Technology
[‡] Department of Mathematics, Computer Science, And Statistics, State University of New York at Oneonta
[§] School of Information, Renmin University of China, P. R. China

***Abstract*—Motivated by the fair rate allocation in a multi-access Gaussian channel, this paper studies the problem of fair rate allocation over a generalized symmetric polymatroid with box constraints. The best-known algorithm for this problem has time complexity $\mathcal{O}\left(n^5 \ln^{\mathcal{O}(1)} n\right)$. In this paper, we present a divide-and-conquer algorithm for this problem with quadratic running time. It is an implementation of a refined decomposing method for the more general separate concave maximization over a polymatroid with box constraints. A key ingredient of the algorithm is a linear-time algorithm for a generalized knapsack problem.**

I. Introduction

This paper is motivated by the following fair rate allocation problem in a multi-access Gaussian channel between a base station and a set E of n users [27]. For each user $j \in E$, let p_j be the signal to noise ratio of the signal from user j perceived by the base station. With the successive interference cancellation by the base station, the (rate) capacity region [27] of these users is the polytope

$$\Omega := \left\{ x \in \mathbb{R}_+^n : \sum_{i \in S} x_i \leq \log(1 + \sum_{i \in S} p_i), \forall S \subseteq E \right\}.$$

As such, the capacity region is a polymatroid [4], and indeed is a *generalized symmetric* polymatroid [27]. For practical applications, additional upper and/or lower bounds have to be imposed on the rate allocations. In many scenarios, serving a user at a rate below its minimum requirement is futile, while serving a user at a rate above its maximum requirement is wasteful. Furthermore, the maximum rate limits can be used to cap the rates of any set of users such as those that have subscribed to a lower tier of service. In general, the lower bound on rate allocation is specified by a vector $a \in \Omega$, and the upper bound on rate allocation is specified by a positive vector $b \in \mathbb{R}_+^n$ with $b \geq a$. The truncated capacity region is then the polytope

$$\Omega' := \Omega \cap \left\{ x \in \mathbb{R}_+^n : a \leq x \leq b \right\}.$$

Suppose that each user $j \in E$ has a weighted θ-fair utility function f_j [16] given by

$$f_j(x_j) = \begin{cases} \frac{w_j}{1-\theta} x_j^{1-\theta}, & \text{if } \theta > 0 \text{ and } \theta \neq 1; \\ w_j \ln x_j, & \text{if } \theta = 1 \end{cases} \tag{1}$$

where $w_j > 0$. The fair rate allocation problem seeks a rate allocation $x \in \Omega'$ maximizing $\sum_{j \in E} f_j(x_j)$.

The above fair rate allocation problem is a special case of the **Separable Concave Maximization** (**SCM**) problem over a polymatroid with box constraints. Let E be a finite set of n users, 2^E denote the collection of subsets of E, and $\mathbb{R}_+^E$ (respectively, $\mathbb{R}^E$) denote the set of non-negative (respectively, real) vectors indexed by E. This problem is specified by n univariate concave and non-decreasing utility functions f_j for $j \in E$, a (polymatroidal) rank function r on 2^E, a vector $a \in \mathbb{R}_+^E$ satisfying that $\sum_{j \in S} a_j \leq r(S)$ for any $S \subseteq E$, and a positive vector $b \in \mathbb{R}_+^E$ satisfying that $a \leq b$, and can be formulated as the following optimization problem:

$$\begin{array}{ll} \max & \sum_{j \in E} f_j(x_j) \\ s.t. & \sum_{j \in S} x_j \leq r(S), \forall S \subseteq E; \\ & a_j \leq x_j \leq b_j, \forall j \in E. \end{array} \tag{2}$$

Note that the set of vectors $x \in \mathbb{R}_+^E$ satisfying the first constraint is exactly the polymatroid of r. Thus, the feasibility region is a *bi-truncation* of the polymatroid by a from below and b from above. If r is a *generalized symmetric* function, i.e., there exist a strictly concave and increasing function ϕ on $\mathbb{R}_+$ with $\phi(0) = 0$ and a positive vector $p \in \mathbb{R}_+^E$ such that

$$r(S) = \phi(p(S)), \forall S \subseteq E, \tag{3}$$

then the set of vectors $x \in \mathbb{R}_+^E$ satisfying the first constraint is a generalized symmetric polymatroid, and the feasibility region is a bi-truncation of a generalized symmetric polymatroid. If the second constraint is replaced by the simple non-negative constraint $x_j \geq 0 \, \forall j \in E$, then the problem becomes the well-studied **SCM** *over a polymatroid* [9], [8]. If the first constraint is replaced by the single constraint $\sum_{j \in E} x_j \leq r(E)$, then the problem is referred to as the *generalized knapsack problem.*

Despite of exponential number inequalities defining polymatroid, the problem **SCM** over a polymatroid can be solved by a decomposing method [9], [8]. This method requires minimizing the difference between the rank function and a modulus function, which itself is submodular. A number of polynomial-time but very expensive algorithms [10], [20], [12], [6], [18], [13], [14] have been proposed for minimizing a general submodular function. Among them, the fastest one is that of [14] that requires $\mathcal{O}\left(n^4 \ln^{\mathcal{O}(1)} n\right)$ running time. Assume that the generalized knapsack problem can be solved in $t(n)$ time. Then, the overall running time is then $\mathcal{O}\left(n \max\left\{n^4 \ln^{\mathcal{O}(1)} n, t(n)\right\}\right)$. If the rank r is a generalized symmetric function [9], then the decomposing method may be implemented with running time $\mathcal{O}(n \max\{n, t(n)\})$, as there exists a linear-time algorithm for minimizing the difference between a generalized symmetric rank function and a modulus function [29].

The decomposing method [9], [8] is not directly applicable to the problem **SCM** over a bi-truncation of a polymatroid as the bi-truncation of a polymatroid is not a polymatroid in general. However, the maximum utility can only be achieved at maximal rates in the bi-truncation of the polymatroid, which also form the set of maximal rates of the polymatroid of a "bi-truncated" rank function [7], [8], [22] defined by

$$\overline{r}(S) = \min_{T \subseteq E} \{r(T) + b(S \setminus T) - a(T \setminus S)\}$$

for each $S \subseteq E$. Thus, the decomposing method [9], [8] can be applied to this polymatroid. But the minimization of the difference between the bi-truncated rank function and a modulus function has to resort to the very expensive algorithms for minimizing a general submodular function. Such challenge remains even if the rank r itself is a generalized symmetric function, as the bi-truncated rank $\overline{r}$ is no longer a generalized symmetric function. Thus, even with a generalized symmetric rank r, the implementation of the decomposing method [9], [8] for **SCM** over a bi-truncation of a polymatroid would still have $\mathcal{O}\left(n \max\left\{n^4 \ln^{\mathcal{O}(1)} n, t(n)\right\}\right)$ running time.

The main objective of this paper is to develop an efficient algorithm for the special case of the problem **SCM** over a truncated polymatroid in which each f_j is a weighted θ-fair utility function given in equation (1), and r is a generalized symmetric function given in equation (3). This special case is referred to as **Fair Rate Allocation (FRA)** over a bi-truncation of a generalized symmetric polymatroid, which models the fair rare allocations in a large class of communication systems [5], [22], [23], [24], [27]. For this purpose, we first provide a refined decomposing method for the more general **SCM** over a bi-truncation of a polymatroid. Compared to the classic decomposing method in [9], [8], the refined decomposing method has broader applicability: The feasibility region is a bi-truncation of a polymatroid, and each f_j may have arbitrary domain instead of the whole real line. The latter is pertinent as the domain of the θ-fair utility function given in equation (1) is either a non-negative real line or the positive real line. In addition, the refined decomposing method has more algorithmic advantages leading to fixing the rates of certain users iteratively and thus speeding up the algorithm. After that, we give a quadratic-time divide-and-conquer implementation of the refined decomposing method for **FRA** over a bi-truncation of a generalized symmetric polymatroid. A key ingredient of this implementation is a linear-time waterfilling algorithm for the generalized knapsack problem. By exploiting the special structure of the optimal solution of the generalized knapsack problem, we are able to achieve a linear-time implementation per divide-and-conquer step with a collection of data structures. The overall running time of the divide-and-conquer algorithm is *quadratic*, a significant improvement of the best-known $\mathcal{O}\left(n^5 \ln^{\mathcal{O}(1)} n\right)$ time complexity.

The following standard notations and terms are used in this paper. For any $s, t \in \mathbb{R}$, $s \vee t$ stands for $\max\{s, t\}$ and $s \wedge t$ stands for $\min\{s, t\}$. For any $x \in \mathbb{R}^E$ and any $S \subseteq E$, let $x(S)$ denote $\sum_{j \in S} x_j$ and $x^S \in \mathbb{R}^S$ be the restriction of x on S. Suppose that A and B are disjoint subsets of E. For any $y \in \mathbb{R}^A$ and $z \in \mathbb{R}^B$, the *direct sum* of y and z, denoted by $y \oplus z$, is the vector $x \in \mathbb{R}^{A \cup B}$ with $x_j = y_j$ for each $j \in A$ and $x_j = z_j$ for each $j \in B$. For two vectors $x, y \in \mathbb{R}^E$, we write $x \leq y$ to denote that $x_j \leq y_j$ for each $j \in E$, and $x < y$ to denote that $x \leq y$ and $x \neq y$. For a subset Φ of $\mathbb{R}^E$, a vector $x \in \mathbb{R}^E$ is called *maximal* in Φ if $x \in \Phi$ and there is no $y \in \Phi$ such that $x < y$.

II. Fundamentals of Polymatroid

In this section, we introduce some fundamental properties of polymatroids that are relevant to this paper. We further refer the interested reader to [8] for background on submodular set functions, [21] for background on polymatroid and submodular function minimization.

Consider a finite ground set E. A real-valued function defined on 2^E is referred to as a *set function on* E. Let r be a set function on E. r is *submodular* if for any $A, B \subseteq E$,

$$r(A) + r(B) \geq r(A \cup B) + r(A \cap B).$$

If the above inequality holds strictly for any A and B neither of which is a subset of each other, then r is said to strictly submodular. r is *increasing* if $r(A) \leq r(B)$ for any $A \subset B \subseteq E$. r is a (*polymatroidal*) *rank* if r is increasing and submodular and $r(\emptyset) = 0$.

A vector $x \in \mathbb{R}^E$ is a *subbase* of r if $x \in \mathbb{R}^E_+$ and $x(A) \leq r(A)$ for each $A \subseteq E$. The set of all subbases of r is a polytope, which is called the *polymatroid* of r. A vector $x \in \mathbb{R}^E$ is a *base* of r if x is a maximal subbase of r. Equivalently, a subbase x of r is a base of r if and only if $x(E) = r(E)$ [8], [21]. The set of all bases of r is a also polytope, which is called the *base polytope* of r. The base polytope of r has the following symmetric exchange property [17]:

Lemma 2.1: Let x and y be two bases of r, and suppose $x_i < y_i$ for some $i \in E$. Then there exist $j \in E$ with $x_j > y_j$ and $\varepsilon > 0$ such that both the vector obtained from x by shifting ε from x_j to x_i and the vector obtained from y by shifting ε from y_i to y_j are bases of r.

Clearly, for any subbase a of r, there exists a base x of r such that $x \geq a$. A vector $x \in \mathbb{R}^n$ is a *superbase* of r if there exists a base y of r such that $y \leq x$.

Consider a vector $u \in \mathbb{R}^E_+$. By treating u as a modular function on 2^E defined by $u(S) = \sum_{e \in S} u(e)$ for each $S \subseteq E$, the difference $r - u$ is also a set function on 2^E, which is referred to as a *rank-modulus difference*. The collection of minimizers of $r - u$ is closed under taking intersections and unions [21]. In particular, the union of all minimizers of $r - u$

is also a minimizer of r, and hence it is the unique *maximal minimizer* of $r-u$. When r is strictly submodular, then the collection of minimizers of $r-u$ is a chain [25], [26], i.e., for any two minimizers T_1 and T_2 of $r-u$, either $T_1 \subseteq T_2$ or $T_2 \subseteq T_1$. Rank-modulus difference and its minimizers play essential roles in algorithmic applications. For example, for any $u \in \mathbb{R}_+^E$ the following statements hold:

- u is a subbase of r iff $\emptyset$ is a minimizer of $r-u$.
- u is a superbase of r iff E is a minimizer of $r-u$.
- u is a base of r iff both $\emptyset$ and E are minimizers of $r-u$.
- If u is not a subbase of r and $u(E) \leq r(E)$, then neither $\emptyset$ nor E is a minimizer of $r-u$.

In general, the minimizers of $r-u$ are fully characterized by the lemma [29] below.

Lemma 2.2: Let T be a subset of E. The following statements are equivalent:

1) T is a minimizer of $r-u$.
2) For each maximal subbase v of r satisfying $v \leq u$, we have $v(T) = r(T)$ and $v(e) = u(e)$ for any $e \in E \setminus T$.
3) For some maximal subbase v of r satisfying $v \leq u$, we have $v(T) = r(T)$ and $v(e) = u(e)$ for any $e \in E \setminus T$.

Suppose that r is a generalized symmetric function. It is easy to verify that r is strictly submodular, strictly increasing, and $r(\emptyset) = 0$. Thus, r is a rank function. In addition, it has the following essential algorithmic property:

Theorem 2.3: For any $u \in \mathbb{R}_+^E$, let $L^{u/p}$ be a list of E in the decreasing ordering of u_j/p_j. Then, the maximal minimizer of $r-u$ is a prefix of the list $L^{u/p}$.

A. Restriction And Contraction

Let A be a subset of E. The *restriction* of r on A, denoted by r^A, is the rank function on 2^A defined by $r^A(S) = r(S)$ for any $S \subseteq A$. The *contraction* of r on A, denoted by r_A, is the rank function on $2^{E\setminus A}$ defined by $r_A(S) = r(S \cup A) - r(A)$ for any $S \subseteq E \setminus A$. The following subbase/base "composition" property [8] holds.

Lemma 2.4: Suppose y is a subbase of r^A and z is a subbase of r_A. Then $x := y \oplus z$ is a subbase of r. Moreover, x is a base of r if and only if both y is a base of r^A and z is a base of r_A.

Conversely, the following "decomposition" property [8] holds.

Lemma 2.5: Let x be a subbase of r, and y and z be the restrictions of x to A and $E \setminus A$ respectively. Then, y is also a subbase of r^A. Furthermore, if y is a base of r^A then z is a also subbase of r_A.

In general, for any two disjoint subsets A and B of E, r_B^A denotes the set function on 2^A defined by $r_B^A(S) = r(S \cup B) - r(B)$ for any $S \subseteq A$. Then, r_B^A can be regarded as the restriction of r_B on A.

A pair of minimizers of $r-u$ have the following property.

Lemma 2.6: Suppose that $T_1 \subseteq T_2$ are two minimizers of $r-u$. Then, u^{T_1} is a superbase of r^{T_1}, $u^{T_2\setminus T_1}$ is a base of $r_{T_1}^{T_2\setminus T_1}$, and $u^{E\setminus T_2}$ is a subbase of $r_{T_2}^{E\setminus T_2}$.

Proof. Let v be a maximal subbase of r satisfying $v \leq u$. By Lemma 2.2, $v(T_1) = r(T_1)$, $v(T_2) = r(T_2)$, and $v_j = u_j$ for any $j \in E \setminus T_1$. Thus, v^{T_1} is a also a base of r^{T_1}, and $v^{T_1} \leq u^{T_1}$. So, u^{T_1} is a superbase of r^{T_1}.

Similarly, v^{T_2} is also a base of r^{T_2}. By Lemma 2.5, $v^{T_2\setminus T_1}$ is a base of $r_{T_1}^{T_2\setminus T_1}$, and $v^{E\setminus T_2}$ is a subbase of $r_{T_2}^{E\setminus T_2}$. As $u^{T_2\setminus T_1} = v^{T_2\setminus T_1}$ and $u^{E\setminus T_2} = v^{E\setminus T_2}$, the lemma holds. ■

If r is a generalized symmetric function given in equation (3), then $r_B^A(S) = \phi(q_0 + p(S)) - \phi(q_0)$ for any $S \subseteq A$. So, r_B^A is also generalized symmetric on 2^A. Suppose that $u \in \mathbb{R}_+^E$ and L is a list of A in the decreasing ordering of u_j/p_j. Based on Theorem 2.3, a procedure **MinDiff-2**(L, q_0) was given in [29] to produce in *linear* time the minimum value δ of $r_B^A - u^A$ and the length k_1 (respectively, k_2) of the shortest (respectively, longest) prefix minimizer of $r_B^A - u^A$ in L.

B. Truncations

Consider a vector $b \in \mathbb{R}_+^E$. The *upper truncation* [8] of r by b, denoted by r^b, is a rank function on 2^E defined by for any $S \subseteq E$,

$$r^b(S) = b(S) + \min_{T \subseteq S} \{r(T) - b(T)\}.$$

For a vector $x \in \mathbb{R}_+^E$, the following equivalent properties holds:

- x is a subbase of r^b if and only if $x \leq b$ and x is a subbase of r;
- x is a base of r^b if and only if x is a maximal subbase of r satisfying that $x \leq b$;

Consider a subbase a of r. The *lower truncation* [8] of r by a, denoted by r_a, is a rank function on 2^E defined by for any $S \subseteq E$,

$$r_a(S) = a(S) + \min_{S \subseteq T \subseteq E} \{r(T) - a(T)\}.$$

For a vector $x \in \mathbb{R}_+^E$, x is a base of r_a if and only if $x \geq a$ and x is a base of r.

Consider a subbase a of r and a vector $b \in \mathbb{R}_+^E$ with $b \geq a$. The *bi-truncation* of r by a from below and b from above [7], [8], [22], denoted by r_a^b, is a rank function on 2^E defined by for any $S \subseteq E$,

$$r_a^b(S) = \min_{T \subseteq E} \{r(T) + b(S \setminus T) - a(T \setminus S)\}.$$

r_a^b is also the upper truncation of r_a by b, and the lower truncation of r^b by a (observing that a is also a subbase of r^b). In addition, we have the following characterization of the bases of r_a^b.

Lemma 2.7: For a vector $x \in \mathbb{R}_+^E$,

- x is a base of r_a^b if and only if x is a maximal subbase of r satisfying that $a \leq x \leq b$.
- when b is a superbase of r, x is a base of r_a^b if and only if x is a base of r and $a \leq x \leq b$.

The above lemma implies that the set of maximal rates of the intersection of the polymatroid of r with the box $\{x \in \mathbb{R}^E : a \leq x \leq b\}$ is exactly the base polytope of r_a^b. Thus the optimization problem described in equation (2) is equivalent to compute a base x of r_a^b maximizing $\sum_{j \in E} f_j(x_j)$.

The following lemma presents an important base composition property of r_a^b.

Lemma 2.8: Suppose that $a \le u \le b$ is not a subbase of r, and T_1 and T_2 are two minimizers of $r - u$ with $T_1 \subseteq T_2$. Let y be a base of $\left(r^{T_1}\right)_{a^{T_1}}^{u^{T_1}}$, and z be a base of $\left(r_{T_2}\right)_{u^{E\setminus T_2}}^{b^{E\setminus T_2}}$. Then,

1) y is base of r^{T_1};
2) $y \oplus u^{A_2\setminus A_1}$ is a base of r^{T_2};
3) $y \oplus u^{A_2\setminus A_1} \oplus z$ is a base of r_a^b.

Proof. (1). By Lemma 2.6, u^{T_1} is a superbase of r^{T_1}. Since y be a base of $\left(r^{T_1}\right)_{a^{T_1}}^{u^{T_1}}$, y is also a base of r^{T_1} by Lemma 2.7.

(2). By Lemma 2.6, $u^{T_2\setminus T_1}$ is a base of $r_{T_1}^{T_2\setminus T_1}$. Thus $y \oplus u^{T_2\setminus T_1}$ is also a base of r^{T_2} by Lemma 2.4.

(3). Denote $x := y \oplus u^{A_2\setminus A_1} \oplus z$. Clearly, $a \le x \le b$. Since z is also a subbase of r_{T_2}, x is also a subbase of r by Lemma 2.4. Assume to the contrary that x is not a base of r_a^b. Then, by by Lemma 2.7, there exists a subbase v of r such that $a \le x < v \le b$. By Lemma 2.5, v^{T_2} is a subbase of r^{T_2}. Since $y \oplus u^{A_2\setminus A_1} \le v^{T_2}$ and $y \oplus u^{A_2\setminus A_1}$ is a base of r^{T_2}, we must have that $v^{T_2} = y \oplus u^{T_2\setminus T_1}$ and hence is a base of r^{T_2}. Again by Lemma 2.5 $v^{E\setminus T_2}$ is also a subbase of r_{T_2}. Since $x < v$, we must have that $u^{E\setminus T_2} \le z < v^{E\setminus T_2} \le b^{E\setminus T_2}$. By Lemma 2.7, z is not a base of $\left(r_{T_2}\right)_{u^{E\setminus T_2}}^{b^{E\setminus T_2}}$, which is a contradiction, Thus, x must be a base of r_a^b. ■

III. Separable Concave Maximization

In this section, we consider the following general separable concave maximization problem

$$\max_{x\in Q} \sum_{j\in E} f_j\left(x_j\right) \tag{4}$$

where Q is a convex subset of $\mathbb{R}^E$, and f_j is a univariate concave and continuous function for each $j \in E$. We derive a necessary condition as well a sufficient condition for an optimal solution to this problem. Similar conditions have been proven rigorously in [8] under the stronger assumption that the domain of f_j is the entire real line for each $j \in E$, which is not applicable for our applications. In addition, the proofs in [8] are extraordinarily lengthy and complicated. In contrast, the proof given in this section is short and simple.

Given $x \in Q$, an ordered pair (i, j) of distinct elements of E is said to be an *exchangeable pair* of x w.r.t. Q if for some $\varepsilon > 0$, the vector obtained from x by shifting ε from x_j to x_i is still in Q. Since Q is convex, for any exchangeable pair (i, j) of x w.r.t. Q, then there exists $\varepsilon > 0$ such that for any $0 \le \varepsilon' \le \varepsilon$, the vector obtained from x by shifting ε' from x_j to x_i is in Q. In addition, the right derivative of f_i at x_i, denoted by $f_i'\left(x_i^+\right)$, exists; and the left derivative of f_j at x_j, denoted by $f_j'\left(x_j^-\right)$, exists (cf. [3], [19]).

For an arbitrary convex set Q, we have the following necessary conditions for x being an optimal solution.

Theorem 3.1: Suppose that x is an optimal solution to the problem (4). Then for any exchangeable pair (i, j) of x w.r.t. Q, $f_i'\left(x_i^+\right) \le f_j'\left(x_j^-\right)$.

Proof. Let $\varepsilon > 0$ be such that for any $0 \le \varepsilon' \le \varepsilon$, the vector obtained from x by shifting ε' from x_j to x_i is in Q. By the optimality of x, we have

$$f_i\left(x_i + \varepsilon'\right) + f_j\left(x_j - \varepsilon'\right) \le f_i\left(x_i\right) + f_j\left(x_j\right),$$

which implies that

$$f_i\left(x_i + \varepsilon'\right) - f_i\left(x_i\right) \le f_j\left(x_j\right) - f_j\left(x_j - \varepsilon'\right).$$

With ε'approaching to 0, we get $f_i'\left(x_i^+\right) \le f_j'\left(x_j^-\right)$. So, the theorem holds. ■

For a polymatroid Q, we have the following sufficient conditions for x being an optimal solution.

Theorem 3.2: Suppose that Q is a polymatroid of r, and x is a base of r. Then x is an optimal solution to the problem (4) if for each exchangeable pair (i, j) of x w.r.t. Q, $f_i'\left(x_i^+\right) \le f_j'\left(x_j^-\right)$.

Proof. Let Q' be the set of bases of r which are optimal solutions to the problem (4). Then, Q' is a convex and compact subset of Q. Assume x is not optimal, then $x \notin Q'$. Let $y \in Q'$ be the one in Q' which has the least Manhattan distance from x. Then $x \ne y$. By Lemma 2.1, there is an pair (i, j) such that $x_i < y_i$ and $x_j > y_j$ and $0 < \varepsilon < \min\left\{x_i - y_i, y_j - x_j\right\}$ such that both the vector obtained from x by shifting ε from x_j to x_i and the vector obtained from y by shifting ε from y_i to y_j, denoted by z, are bases of r. The pair (i, j) is an exchange pair of x w.r.t. Q, and hence $f_i'\left(x_i^+\right) \le f_j'\left(x_j^-\right)$. Note that

$$\begin{aligned} y_i > z_i = y_i - \varepsilon > x_i, \\ y_j < z_j = y_j + \varepsilon < x_j. \end{aligned}$$

The concavity of f_i and f_j implies that

$$\begin{aligned} f_i\left(y_i\right) - f_i\left(z_i\right) &\le \varepsilon f_i'\left(z_i^+\right) \le \varepsilon f_i'\left(x_i^+\right), \\ f_j\left(y_j\right) - f_j\left(z_j\right) &\le -\varepsilon f_j'\left(z_j^-\right) \le -\varepsilon f_j'\left(x_j^-\right). \end{aligned}$$

Thus,

$$\begin{aligned} &\sum_{k\in[n]} f_k\left(y_k\right) - \sum_{k\in[n]} f_k\left(z_k\right) \\ &= \left[f_i\left(y_i\right) + f_j\left(y_j\right)\right] - \left[f_i\left(z_i\right) + f_j\left(z_j\right)\right] \\ &\le \varepsilon\left[f_i'\left(x_i^+\right) - f_j'\left(x_j^-\right)\right] \le 0. \end{aligned}$$

So, $z \in Q'$. On the other hand,

$$\begin{aligned} &\sum_{k\in[n]} \left|y_k - x_k\right| - \sum_{k\in[n]} \left|z_k - x_k\right| \\ &= \left(\left|y_i - x_i\right| + \left|y_j - x_j\right|\right) - \left(\left|z_i - x_i\right| + \left|z_j - x_j\right|\right) \\ &= \left(y_i - x_i + x_j - y_j\right) - \left(z_i - x_i + x_j - y_j\right) \\ &= \left(y_i - z_i\right) - \left(z_j - y_j\right) = \varepsilon + \varepsilon = 2\varepsilon > 0. \end{aligned}$$

Thus, z has strictly smaller Manhattan distance from x than y, which contradicts to the choice of y. So, x must be optimal. ■

IV. A Refined Decomposing Method

Suppose that r is a (polymatroidal) rank function on 2^E, and f_j is a univariate non-negative, non-decreasing, concave and continuous function f_j for each user $j \in E$. We define an extended problem $\mathbf{SCM}(A, A_0, a, b)$ where

- A is a non-empty subset of E, and is referred as the ground set,
- A_0 is a subset of E disjoint from A,
- a is a subbase of $r^A_{A_0}$ representing the lower bound, and
- b is a positive vector in $\mathbb{R}^A_+$ satisfying that $b \geq a$, representing upper bound.

A feasible solution to this extended problem is a subbases x of $r^A_{A_0}$ satisfying that $a \leq x \leq b$, and the objective of this extended problem is to compute a feasible solution x maximizing $\sum_{j\in A} f_j(x_j)$. In other words, it is the following optimization problem:

$$\begin{array}{ll} \max & \sum_{j\in A} f_j(x_j) \\ s.t. & \sum_{j\in S} x_j \leq r_{A_0}(S), \forall S \subseteq A; \\ & a_j \leq x_j \leq b_j, \forall j \in A. \end{array}$$

For the problem $\mathbf{SCM}(A, A_0, a, b)$, define

$$A^{=} := \{j \in A : a_j = b_j\},$$
$$A^{<} := \{j \in A : a_j < b_j\}.$$

Clearly, for each feasible solution x, we must have $x_j = b_j$ for any $j \in A^{=}$. Thus, users in $A^{=}$ are said to be *fixed*, and users in $A^{<}$ is said to be *free*. In the *trivial* instance that $A^{<} = \emptyset$, $b = a$ is the unique optimal solution. So, we assume that $A^{<} \neq \emptyset$. There are two *simple* cases of the instance in which the optimal solution is unique and directly computable:

- Case 1: b is a subbase of $r^A_{A_0}$. In this case, b is also the unique optimal solution.
- Case 2: $A^{<}$ is a singleton $\{j\}$. In this case, the unique optimal solution x is identical to b except the value of x_j, which is equal to $b_j + \min_{T\subseteq A}(r_{A_0}(T) - b(T))$.

Both cases can tested or solved by computing the minimum value of $r^A_{A_0} - b$. For the instance which is of neither of the above cases (i.e., $|A^{<}| \geq 2$ and b is not a subbase), we derive a recursive relation subsequently.

The following problem, denoted by $\mathbf{GKS}(A, A_0, a, b)$, is a relaxation of the problem $\mathbf{SCM}(A, A_0, a, b)$:

$$\begin{array}{ll} \max & \sum_{j\in A} f_j(x_j) \\ s.t. & \sum_{j\in A} x_j \leq r_{A_0}(A), \\ & a_j \leq x_j \leq b_j, \forall j \in A \end{array}$$

Let u be an optimal solution to the above relaxed problem. If u is a subbase of $r^A_{A_0}$, then u is an optimal solution to the problem $\mathbf{SCM}(A, A_0, a, b)$. So we assume that u is not a subbase of $r^A_{A_0}$. Suppose that A_1 and A_2 be two minimizers of $r^A_{A_0} - u$ with $A_1 \subseteq A_2$, and denote $\overline{A_2} = A \setminus A_2$. Then, neither A_1 nor $\overline{A_2}$ is empty. Since $a^{A_1} \leq u^{A_1}$ and a^{A_1} is a subbase of $r^{A_1}_{A_0}$, the problem $\mathbf{SCM}\left(A_1, A_0, a^{A_1}, u^{A_1}\right)$ is well-defined. Since $u^{\overline{A_2}} \leq b^{\overline{A_2}}$ and $u^{\overline{A_2}}$ is a subbase of $r^{\overline{A_2}}_{A_0\cup A_2}$, the problem $\mathbf{SCM}\left(\overline{A_2}, A_0 \cup A_2, u^{\overline{A_2}}, b^{\overline{A_2}}\right)$ is also well-defined. These two problems are referred as the "child" problems of the "parent" problem $\mathbf{SCM}(A, A_0, a, b)$. They have the following composition relations.

Theorem 4.1: Consider any optimal solution y^* to the problem $\mathbf{SCM}\left(A_1, A_0, a^{A_1}, u^{A_1}\right)$ and any optimal solution z^* to the problem $\mathbf{SCM}\left(\overline{A_2}, A_0 \cup A_2, u^{\overline{A_2}}, b^{\overline{A_2}}\right)$. Let $x^* = y^* \oplus u^{A_2\setminus A_1} \oplus z^*$. Then, x^* is an optimal solution to the problem $\mathbf{SCM}(A, A_0, a, b)$.

Proof. Suppose that y is an optimal base solution to the problem $\mathbf{SCM}\left(A_1, A_0, a^{A_1}, u^{A_1}\right)$, and z is an optimal base solution to the problem $\mathbf{SCM}\left(\overline{A_2}, A_0 \cup A_2, u^{\overline{A_2}}, b^{\overline{A_2}}\right)$. Let $x = y \oplus u^{A_2\setminus A_1} \oplus z$. By Lemma 2.8, x is a base solution to the problem $\mathbf{SCM}(A, A_0, a, b)$. We prove the optimality of x via Theorem 3.2. Suppose that (i, j) is an exchangeable pair of x w.r.t. the feasibility region of the problem $\mathbf{SCM}(A, A_0, a, b)$. Then, for some $\varepsilon > 0$, the vector $\widehat{x}$ obtained from x by shifting ε from x_j to x_i is also a feasible solution to the problem $\mathbf{SCM}(A, A_0, a, b)$. We show that $f'_i\left(x^+_i\right) \leq f'_j\left(x^-_j\right)$ in three cases.

Case 1: $x_i < u_i$. Then, $i \in A_1$. In addition, $j \in A_1$ for otherwise by Lemma 2.8

$$\widehat{x}(A_1) = x(A_1) + \varepsilon = y(A_1) + \varepsilon = r_{A_0}(A_1) + \varepsilon,$$

which is impossible. We further claim that (i, j) is also an exchangeable pair of y w.r.t. the feasibility region of the problem $\mathbf{SCM}\left(A_1, A_0, a^{A_1}, u^{A_1}\right)$. Indeed, let $\varepsilon' = \min\{\varepsilon, u_i - x_i\}$. Then, $\varepsilon' > 0$. Let $\overline{x}$ be the vector obtained from x by shifting ε' from x_j to x_i. Then, $\overline{x}$ is also a feasible solution to the problem $\mathbf{SCM}(A, A_0, a, b)$. So, $\overline{x}$ is a subbase of $r^A_{A_0}$ and $a \leq \overline{x} \leq b$. By Lemma 2.5, $\overline{x}^{A_1}$ is also a subbase of $r^{A_1}_{A_0}$. Clearly, $\overline{x}^{A_1} \geq a^{A_1}$. In addition,

$$\overline{x}_i = x_i + \varepsilon' \leq x_i + u_i - x_i = u_i,$$

and for any $k \in A_1 \setminus \{i\}$, $\overline{x}_k \leq x_k = y_k \leq u_k$. Thus, $\overline{x}^{A_1} \leq u^{A_1}$. So, $\overline{x}^{A_1}$ is a feasible solution to the problem $\mathbf{SCM}\left(A_1, A_0, a^{A_1}, u^{A_1}\right)$. However, $\overline{x}^{A_1}$ is exactly the vector obtained from y by shifting ε' from y_j to y_i. Thus, our claim holds. As y is an optimal solution to the problem $\mathbf{SCM}\left(A_1, A_0, a^{A_1}, u^{A_1}\right)$, by Theorem 3.1 we have

$$f'_i\left(x^+_i\right) = f'_i\left(y^+_i\right) \leq f'_j\left(y^-_j\right) = f'_j\left(x^-_j\right).$$

Case 2: $x_j > u_j$. Then, $j \in \overline{A_2}$. In addition, $i \in \overline{A_2}$ for otherwise $i \in A_2$ and by by Lemma 2.8

$$\widehat{x}(A_2) = x(A_2) + \varepsilon = y(A_1) + u(A_2 \setminus A_1) + \varepsilon = r_{A_0}(A_2) + \varepsilon,$$

which is impossible. We further claim that (i, j) is also an exchangeable pair of z w.r.t. the feasibility region of the problem $\mathbf{SCM}\left(\overline{A_2}, A_0 \cup A_2, u^{\overline{A_2}}, b^{\overline{A_2}}\right)$. Indeed, let $\varepsilon' = \min\{\varepsilon, x_j - u_j\}$. Then, $\varepsilon' > 0$. Let $\overline{x}$ be the vector obtained from x by shifting ε' from x_j to x_i. Then, $\overline{x}$ is also a feasible solution to the problem $\mathbf{SCM}(A, A_0, a, b)$. Since $\overline{x}^{A_1}$ So, $\overline{x}$ is a subbase of $r^A_{A_0}$ and $a \leq \overline{x} \leq b$. As $\overline{x}^{A_2} = x^{A_2} = y \oplus u^{A_2\setminus A_1}$,

$\overline{x}^{A_2}$ is a base of $r_{A_0}^{A_2}$. By Lemma 2.5, $\overline{x}^{\overline{A_2}}$ is also a subbase of $r_{A_0\cup A_2}^{\overline{A_2}}$. Clearly, $\overline{x}^{\overline{A_2}} \leq b^{\overline{A_2}}$. In addition,

$$\overline{x}_j = x_j - \varepsilon' \geq x_j - (x_j - u_j) = u_j,$$

and for any $k \in \overline{A_2} \setminus \{j\}$, $\overline{x}_k \geq x_k = z_k \geq u_k$. Thus, $\overline{x}^{\overline{A_2}} \geq u^{\overline{A_2}}$. So, $\overline{x}^{\overline{A_2}}$ is a feasible solution to the problem $\mathbf{SCM}\big(A_1, A_0, a^{A_1}, u^{A_1}\big)$. However, $\overline{x}^{\overline{A_2}}$ is exactly the vector obtained from z by shifting ε' from z_j to z_i. Thus, our claim holds. As z is an optimal solution to the problem $\mathbf{SCM}\Big(\overline{A_2}, A_0 \cup A_2, u^{\overline{A_2}}, b^{\overline{A_2}}\Big)$, by Theorem 3.1 we have

$$f'_i\left(x_i^+\right) = f'_i\left(z_i^+\right) \leq f'_j\left(z_j^-\right) = f'_j\left(x_j^-\right).$$

Case 3: $x_i \geq u_i$ and $x_j \leq u_j$. It is easy to verify that (i,j) is also an exchangeable pair of u with respect to feasibility region of the problem $\mathbf{GKS}(A, A_0, a, b)$. As u is an optimal solution to the problem $\mathbf{GKS}(A, A_0, a, b)$, by Theorem 3.1 we have

$$f'_i\left(x_i^+\right) \leq f'_i\left(u_i^+\right) \leq f'_j\left(u_j^-\right) \leq f'_j\left(x_j^-\right).$$

Finally, we show that x^* is an optimal solution to the problem $\mathbf{SCM}(A, A_0, a, b)$. Clearly, $a \leq x^* \leq b$. As y^* is a subbase of $r_{A_0}^{A_1}$, $u^{A_2 \setminus A_1}$ is a base of $r_{A_0\cup A_1}^{A_2\setminus A_1}$, and z^* is a subbase of $r_{A_0\cup A_2}^{\overline{A_2}}$, x^* is a subbase of $r_{A_0}^{A_1}$ by Lemma 2.4. Thus, x^* is a feasible solution to the problem $\mathbf{SCM}(A, A_0, a, b)$. By the optimality of y^* and z^*, we have

$$\sum_{j\in A_1} f_j\left(y_j^*\right) \geq \sum_{j\in A_1} f_j\left(y_j\right),$$
$$\sum_{j\in \overline{A_2}} f_j\left(z_j^*\right) \geq \sum_{j\in \overline{A_2}} f_j\left(z_j\right).$$

Thus,

$$\sum_{j\in A} f_j\left(x_j^*\right) \geq \sum_{j\in A} f_j\left(x_j\right).$$

The optimality of x^* then follows from the optimality of x. ■

We remark that the above decomposing method is a refinement of the decomposing method presented in [9] in two aspects. First, a pair of minimizers A_1 and A_2 is utilized. The advantage of employing dual minimizers is obvious: more users may become fixed. In particular, the entire subset $A_2 \setminus A_1$ becomes fixed. The child problems may have smaller size. For certain rank r such as the generalized symmetric rank, the dual minimizers can be computed at no additional cost. Second, The child problems have closer upper bounds and lower bounds. Indeed, Some of them are identical, and thus the corresponding users also have fixed rates.

V. A Divide-And-Conquer Algorithm

In this section, we assume that the rank r is a generalized symmetric rank as defined in equation (3), and f_j is a fairness function as defined in (3). Then, the problem given in equation (2) is an instance of **FRA** over a bi-truncation of a generalized symmetric polymatoid. We present a divide-and-conquer implementation of the refined decomposing method given in the previous section with quadratic time complexity.

A. Strengthening Lower/Upper Bounds

In this subsection, we strengthen the lower bound a and the upper bound b without affecting the optimality while ensuring $[a_j, b_j]$ lies within the domain of f_j for each $j \in E$. This property is essential to the algorithm for the generalized knapsack problem to be developed in the next subsection. The strengthening of the lower/upper bounds is based on the following theorem.

Theorem 5.1: For any base v of r_a^b and any $j \in E$,

$$a_j \vee \{b_j \wedge [r(E) - r(E\setminus\{j\})]\} \leq v_j \leq b_j \wedge r(\{j\}).$$

Proof. By Lemma 2.7, $a_j \leq v_j \leq b_j \wedge r(\{j\})$. We only need to prove that

$$v_j \geq b_j \wedge [r(E) - r(E\setminus\{j\})].$$

Since r_a^b is the lower truncation of r^b by a, we have

$$r^b(E) = r_a^b(E) = v_j + \sum_{i\in E\setminus\{j\}} v_i$$
$$\leq v_j + r_a^b(E\setminus\{j\}) \leq v_j + r^b(E\setminus\{j\}),$$

which implies that $v_j \geq r^b(E) - r^b(E\setminus\{j\})$. Thus, it is sufficient to show that

$$r^b(E) - r^b(E\setminus\{j\}) \geq b_j \wedge [r(E) - r(E\setminus\{j\})].$$

We consider two cases.

Case 1: Some $T \subseteq E\setminus\{j\}$ is a minimizer of $r-b$ over all subsets of E. Then

$$r^b(E) = b(E) + [r(T) - b(T)],$$
$$r^b(E\setminus\{j\}) = b(E\setminus\{j\}) + [r(T) - b(T)].$$

So, $r^b(E) - r^b(E\setminus\{j\}) = b_j$.

Case 2: For some $T \subseteq E\setminus\{j\}$, $T\cup\{j\}$ is a minimizer of $r-b$ over all subsets of E. Then

$$r^b(E) = b(E) + [r(T\cup\{j\}) - b(T\cup\{j\})]$$
$$= b(E\setminus\{j\}) + [r(T\cup\{j\}) - b(T)],$$

and

$$r^b(E\setminus\{j\}) \leq b(E\setminus\{j\}) + [r(T) - b(T)].$$

So,

$$r^b(E) - r^b(E\setminus\{j\}) \geq r(T\cup\{j\}) - r(T)$$
$$\geq r(E) - r(E\setminus\{j\}).$$

Thus, the theorem holds. ■

Since an optimal solution is achieved some base of r_a^b, the above theorem implies that for each $j \in E$, we can replace b_j by $b_j \wedge r(\{j\})$ and then replace a_j by

$$a_j \vee \{b_j \wedge [r(E) - r(E\setminus\{j\})]\}$$

without affecting the optimality. Note that for the generalized symmetric rank r as defined in equation (3), $r(E) = \phi(p(E))$, and $r(E\setminus\{j\}) = \phi(p(E) - p_j)$ for each $j \in E$. By first calculating $p(E)$ in linear time, the above replacements can be done in linear time. After the replacements, for each $j \in E$ we have $b_j \geq a_j > 0$ and consequently, $[a_j, b_j]$ lies within the domain of f_j.

B. Waterfilling for Generalized Knapsack Problem

Let S be a non-empty subset of E, and μ be a positive parameter s.t. $a(A) < \mu < b(A)$. By strengthening of the upper/lower bounds, we assume that $[a_j, b_j]$ lies in the domain of f_j for each $j \in S$. The following optimization problem is a general form of the generalized knapsack problem arising from the decomposing method:

$$\begin{array}{ll} \max & \sum_{j\in S} f_j(x_j) \\ s.t. & \sum_{j\in S} x_j \le \mu, \\ & a_j \le x_j \le b_j, \forall j \in S \end{array} \tag{5}$$

In this subsection, we present a *linear-time* algorithm which not only computes an optimal solution to the above problem, but also produces two useful orderings of S by taking advantage of the special structure of the optimal solution. Thus, our algorithm is stronger and more general than the classic ones in [2], [15], [28].

Since the objective function is strictly concave and the feasibility region is a convex polytope, there is a unique optimal solution u. The Karush-Kuhn-Tucker conditions (e.g., [1]) for the problem (5) stipulate that there exists a unique $\lambda > 0$, which is the Lagrangian multiplier of the budget constraint, such that for each $i \in S$,

$$u_i = a_i \vee \left[b_i \wedge (f_i')^{-1}(\lambda)\right].$$

The optimal solution u and the Lagrangian multiplier λ have the following water-filling interpretation. Each user $i \in S$ is treated as a water container, and those water containers form a water tank with a stair-case floor. The Lagrangian multiplier $\lambda > 0$ is reparameterized by the water level $\ell = \lambda^{-1/\theta} > 0$, and

$$(f_i')^{-1}(\lambda) = (f_i')^{-1}\left(\ell^{-1/\theta}\right) = w_i^{1/\theta}\ell = s_i\ell.$$

For each $i \in S$, the container i is specified by three parameters:

- the floor area $s_i = w_i^{1/\theta}$;
- the *floor height* $f_i'(a_i)^{-1/\theta} = a_i w_i^{-1/\theta} = a_i/s_i$;
- the *ceiling height* $f_i'(b_i)^{-1/\theta} = b_i w_i^{-1/\theta} = b_i/s_i$.

At a water level ℓ, since the container i holds water of volume $u_i = a_i \vee [b_i \wedge (s_i\ell)]$. Thus, the maximum water level ℓ is the unique water level at which the total water volume held by all containers in S is exactly the budget μ. Accordingly, u and ℓ can be computed in three steps:

1) Find the maximum value ℓ among the values in
$$\{a_i/s_i : i \in S\} \cup \{b_i/s_i : i \in S\}$$
such that $\sum_{i\in S} a_i \vee [b_i \wedge (s_i\ell)] \le \mu$.
2) Partition S into three sets:
$$S_0 = \{i \in S : a_i/s_i > \ell\},$$
$$S_1 = \{i \in S : a_i/s_i \le \ell < b_i/s_i\},$$
$$S_2 = \{i \in S : b_i/s_i \le \ell\}.$$
Then, $u_i = a_i$ for each $i \in S_0$; $u_i = b_i$ for each $i \in S_2$; and $a_i \le u_i < b_i$ for each $i \in S_1$.
3) Compute $\ell = \frac{\mu - a(S_0) - b(S_1)}{s(S_1)}$. Then, $u_i = s_i\ell$ for each $i \in S_1$.

In order to be able to compute u and ℓ in linear time, the following two lists of S are made available as part of the input: a list $L^{a/s}$ (respectively, $L^{b/s}$) of S in the non-decreasing order of floor (respectively, ceiling) heights. These two lists also enable to compute in linear time a list $L^{u/s}$ of S in the non-decreasing order of u_i/s_i: it is the concatenation of the sublist of $L^{b/s}$ consisting of the users in S_2, and the sublist of $L^{a/s}$ consisting of the users in $S \setminus S_2 = S_0 \cup S_1$. Suppose we further want to produce in linear time a list $L^{u/p}$ of S in the decreasing order of u_i/p_i, where p_i is the parameter appearing equation (3) defining a generalized symmetric rank. Then, the following three lists of S are also made available as part of the input: a list $L^{a/p}$ (respectively, $L^{b/p}$, $L^{s/p}$) of S in the decreasing order of a_j/p_j (respectively, b_j/p_j, s_j/p_j). Due to the special structure of u, $L^{u/p}$ can be merged from the sublist of $L^{b/p}$ consisting of the users in S_2, the sublist of $L^{a/p}$ consisting of the users in S_0, and the sublist of $L^{s/p}$ consisting of the users in S_1.

Now, we describe an linear-time procedure **WF** implementing the above waterfilling method. The vectors a, b, s and p are stored as globally accessible arrays indexed by E and thus are not included in the input of the procedure **WF**. The vector u is also maintained as a global array indexed by E and thus are not included in the output of the procedure **WF**. Thus, the input of the procedure **WF** is a tuple $\left(L^{a/p}, L^{b/p}, L^{s/p}, L^{a/s}, L^{b/s}, \mu\right)$, and the output of the procedure **WF** is a pair $\left(L^{u/p}, L^{u/s}\right)$. The following data structures and variables are used in the procedure **WF**. The procedure **WF** iteratively scans the two lists $L^{a/s}$ and $L^{b/s}$ to find the next lowest floor/ceiling height, which is chosen as the next water level ℓ, and bookkeeps the previous water level by a variable ℓ'. At any moment, each user (or water container) i is in one of the states:

- state 0, if the container i has not yet been visited
- state 1, if the container i has been visited and but not yet filled fully.
- state 2, if the container i has been visited and also filled fully.

An array $state$ indexed by E is used to store and update the states of the containers (users). The total floor area of the containers in the state 1 is maintained by the variable σ. The procedure **WF** is split into two steps. Step 1 computes the water level ℓ, the optimal solution u, the array $state$, and a list $L^{u/s}$; while Step 2 computes a list $L^{u/p}$. They are elaborated below.

Step 1 of the procedure **WF** begins with initialization. Along the list $L^{a/s}$, each entry of the array $state$ is initialized to 0, and μ is reduced by $a(S)$. The previous water level ℓ' is initialized to the floor height of the first user in $L^{a/s}$. The variable σ is initialized to 0. The list $L^{u/s}$ is initialized to an empty list.

After the initialization, the following iteration is repeated until $\mu = 0$. First along the two lists $L^{a/s}$ and $L^{b/s}$ the next lowest floor or ceiling height ℓ (ties is broken by choosing the floor height) and the corresponding user (water container) k are selected. The binary variable χ distinguishes whether the height is a floor height ($\chi = 0$) or a ceiling height ($\chi = 1$). Then ℓ can be achieved if and only if the volume of the additional water to be filled, which is $\sigma(\ell - \ell')$, does not exceed the available water volume μ. Accordingly, we consider two cases:

- Case 1: ℓ can be achieved. Then μ is reduced by $\sigma(\ell - \ell')$, and ℓ' is replaced by ℓ. If ℓ is the floor height of the user k (i.e., $\chi = 1$), then the user k enters the state 1; hence $state_k$ is updated to 1 and σ is increased by s_k. If ℓ is the ceiling height of the user k (i.e., $\chi = 1$), then the user k enters the state 2; hence $state_k$ is updated to 2, σ is decreased by s_k, u_k is fixed to b_k, and k is appended to the list $L^{u/s}$.
- Case 2: ℓ cannot be achieved. Then we find the maximum water level $\ell' + \mu/\sigma$ using up all water; hence ℓ is updated to this level and μ is reduced to 0.

After the exit of the above iterations, along the list $L^{a/s}$, for each user i in the state 0 (respectively, 1), u_i is fixed to a_i (respectively, $s_i\ell$) and appended to $L^{u/s}$. Clearly, this steps takes linear time.

Step 2 of the procedure **WF** produces the list $L^{u/p}$ of S. First, the sublist $\widetilde{L}^{a/p}$ (respectively, $\widetilde{L}^{s/p}$, $\widetilde{L}^{b/p}$) of $L^{a/p}$ (respectively, $L^{s/p}$, $L^{b/p}$) consisting of the users in the state 0 (respectively, 1, 2) is extracted from $L^{a/p}$ (respectively, $L^{s/p}$, $L^{b/p}$). Then, the three lists $\widetilde{L}^{a/p}$, $\widetilde{L}^{s/p}$, and $\widetilde{L}^{b/p}$ are merged into a single list $L^{u/p}$ of S. Clearly, this step also takes linear time.

In summary, we have the following result.

Theorem 5.2: $\mathbf{WF}\left(L^{a/p}, L^{b/p}, L^{s/p}, L^{a/s}, L^{b/s}, \mu\right)$ computes an optimal solution u to the problem (5) as well as two lists $L^{u/p}$ and $L^{u/s}$ in linear time.

C. Putting Together

We begin with the introduction of the data structures to be used. First, there are *seven* arrays indexed by E which are globally accessible to all the procedure calls. Among them, a static array p is used for storing the vector p in the definition of r; and static array s defined by $s_j = w_j^{1/\theta}$ for each $j \in E$, where w_j and θ are the two parameters in the definition of f_j. Two arrays a and b are used for storing and updating the *strengthened* lower bound a and upper bound b respectively. At the end of the algorithm, b is returned as the final solution. The array u is to store the optimal solution to the generalized knapsack problem. The array $class$ is to indicate the membership of A_1, $A_2 \setminus A_1$, and $\overline{A_2}$ in a problem with ground set A by $class_j = 0$ (respectively, 1, 2) if j is in $A_2 \setminus A_1$ (respectively, A_1, $\overline{A_2}$). The array $state$ is used by the procedure **WF**. Second, five lists are used as the input to the procedure **WF**: a list $L^{a/p}$ (respectively, $L^{b/p}$, $L^{s/p}$) of A in the decreasing order of a_j/p_j (respectively, b_j/p_j, s_j/p_j), and a list $L^{a/s}$ (respectively, $L^{b/s}$) of A in the increasing order of a_j/s_j (respectively, b_j/s_j). Third, two lists are used as the output by the procedure **WF**: a list $L^{u/p}$ of A in the decreasing order of u_j/p_j, and a list $L^{u/s}$ of A in the increasing order of u_j/s_j.

With the above data structures, we describe a proper representation of a problem $\mathbf{SCM}(A, A_0, a, b)$. First, a and b can be skipped as they are globally accessible and updated throughout the execution of algorithm. Second, A_0 can be replaced by a single parameter $q_0 = p(A_0)$ as

$$r_{A_0}^A(S) = \phi(q_0 + p(S)) - \phi(q_0)$$

for each $S \subseteq A$. Third, for the achieving linear-time computation, A is represented by the five lists $L^{a/p}$, $L^{b/p}$, $L^{s/p}$, $L^{a/s}$, and $L^{b/s}$ for A. Thus, the problem $\mathbf{SCM}(A, A_0, a, b)$ is represented by the tuple $\left(L^{a/p}, L^{b/p}, L^{s/p}, L^{a/s}, L^{b/s}, q_0\right)$. Accordingly, we name the recursive procedure to solve the problem $\mathbf{SCM}(A, A_0, a, b)$ by problem $\mathbf{DC}\left(L^{a/p}, L^{b/p}, L^{s/p}, L^{a/s}, L^{b/s}, q_0\right)$.

The main algorithm for the problem $\mathbf{SCM}(E, \emptyset, a, b)$, named **DC-Main**, simply prepares the seven arrays $p, w, a, b, u, class, state$ and the five lists $L^{a/p}, L^{b/p}, L^{s/p}, L^{a/s}, L^{b/s}$ of E, calls the procedure $\mathbf{DC}\left(L^{a/p}, L^{b/p}, L^{s/p}, L^{a/s}, L^{b/s}, 0\right)$, and finally returns b. The preparation takes linearithmatic time. In the remaining of this subsection, we elaborate on the divide-and-conquer design of the procedure $\mathbf{DC}\left(L^{a/p}, L^{b/p}, L^{s/p}, L^{a/s}, L^{b/s}, q_0\right)$ for the problem $\mathbf{SCM}(A, A_0, a, b)$.

Direct computation: First, $|A^<|$ is computed along the list L^{bp}. If $|A^<| = 0$, then b^A is optimal and the procedure stops and returns. Otherwise, the minimum δ of $r_{A_0}^A - b^A$ is computed along the list $L^{b/p}$ with parameter q_0 by applying the procedure **MinDiff-2**. If $\delta = 0$, then b^A is a subbase of $r_{A_0}^A$ and hence is optimal; so the procedure stops and returns. If $|A^<| = 1$ then for the unique $j \in A^<$, b_j is reset to $b_j + \delta$ and the procedure stops and returns. Clearly, this part takes linear time.

Division: This part is logically split into three steps. Step 1 essentially computes u and the two minimizers A_1 and A_2. First, $a(A)$, $b(A)$, $p(A)$ and $\mu = \phi(q_0 + p(A)) - \phi(q_0)$ are computed along the list $L^{b/p}$. If $a(A) = \mu$ then a^A is a base of $r_{A_0}^A$ and is optimal; hence b^A is reset to a^A and the procedure stops and returns. If $b(A) \leq \mu$ then u^A is set to b^A, and $L^{u/p}$ and $L^{u/s}$ are set to $L^{b/p}$ and $L^{b/s}$ respectively; otherwise, the procedure **WF** is applied to the tuple $\left(L^{a/s}, L^{b/s}, L^{a/p}, L^{b/p}, L^{s/p}, \mu\right)$ to compute an optimal solution u^A together with the two lists $L^{v/s}$ and $L^{u/p}$ of A. Finally, the minimum value δ of $r_{A_0}^A - u^A$ and the length k_1 (respectively, k_2) of the shortest (respectively, longest) prefix minimizer A_1 (respectively, A_2) in $L^{u/p}$ are computed by applying the procedure $\mathbf{MinDiff\text{-}2}\left(L^{u/p}, q_0\right)$. If $\delta = 0$, then b^A is reset to u^A, and and the procedure stops and returns. Clearly, this step takes linear time.

Step 2 of the **Division** part computes $q_2 = p(A_2)$, updates the array arrays $class$, resets a and b, and create the list $L_1^{b/p}$ of A_1 and the list $L_2^{a/p}$ of $\overline{A_2}$. Note that $L_1^{b/p}$ consists of the first k_1 elements in $L^{u/p}$, and $L_2^{a/p}$ consists of the last $|A| - k_2$ elements in $L^{u/p}$. Thus, by a single scanning of L^{up}, the two lists L_1^{bp} and L_2^{ap}, the three arrays $class$, a and b, and the scalar variable q_2 can be computed and updated.

Step 3 of the **Division** part creates the four lists $L_1^{a/p}, L_1^{s/p}, L_1^{a/s}, L_1^{b/s}$ of A_1 and the four lists $L_2^{b/p}, L_2^{s/p}, L_2^{a/s}, L_2^{b/s}$ of $\overline{A_2}$. At the assistance of the array $class$,

- the list $L_1^{a/p}$ (respectively, $L_2^{b/p}$) is extracted from $L^{a/p}$ (respectively, $L^{b/p}$),
- the two lists $L_1^{s/p}$ and $L_2^{s/p}$ are extracted from $L^{s/p}$,
- the two lists $L_1^{b/s}$ and $L_2^{a/s}$ are extracted from $L^{u/s}$,
- the list $L_1^{a/s}$ (respectively, $L_2^{b/s}$) is extracted from $L^{a/s}$ (respectively, $L_2^{b/s}$).

Clearly, this step also takes linear time.

Conquer: This part is trivial. It simply makes a call to $\mathbf{DC}\left(L_1^{a/p}, L_1^{b/p}, L_1^{s/p}, L_1^{a/s}, L_1^{b/s}, q_0\right)$ and a call to $\mathbf{DC}\left(L_2^{a/p}, L_2^{b/p}, L_2^{s/p}, L_2^{a/s}, L_2^{b/s}, q_0 + q_2\right)$, and then returns.

All the problems generated in the recursive computation can be organized as a rooted binary tree according to the parent-child relations, known as the recurrence tree. The depth of the recurrence tree is known as the recurrence depth. Since the number of users in a child problem is strictly less than that of its parent problem, the recurrence depth is $O(n)$. As the **Direct computation** part and **Division** part takes linear time, all the problems at the same level (or depth) have $O(n)$ time complexity since they have disjoint ground sets. So, the overall running time of the algorithm **DC-Main** is also quadratic. In summary, we have the following result.

Theorem 5.3: The algorithm **DC-Main** produces an optimal solution in quadratic time.

VI. Conclusion

For a large class of communication systems [5], [22], [23], [24], [27], their capacity regions can be represented by a generalized symmetric polymatroid with box constraints. The best-known algorithm for fair rate allocation over a generalized symmetric polymatroid with box constraints has time complexity $\mathcal{O}\left(n^5 \ln^{\mathcal{O}(1)} n\right)$. In this paper, we develop a divide-and-conquer algorithm for this problem with quadratic running time. The underlying refined decomposing method applies for the more general separate concave maximization problem over a polymatroid with box constraints. A key ingredient of the algorithm is a linear-time algorithm for a generalized knapsack problem, which is of independent interest.

ACKNOWLEDGEMENTS: This work was supported in part by the National Natural Science Foundation of P. R. China under grants 61502505, 61529202, 61572131, and 61872081, by the National Science Foundation of USA under grant CNS-1526638, and by SUNY Oneonta Faculty Development Funds.

References

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programing: Theory and Algorithms*. Third Edition, Wiley-Interscience, John Wiley & Sons Inc., 2006.

[2] L. Bodin, Optimization procedures for the analysis of coherent structures, *IEEE Transactions on Reliability* R-18(3): 118-126, 1969.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[4] J. Edmonds, Submodular functions, matroids and certain polyhedra. In *Combinatorial structures and their applications*, eds. R. Guy, H. Hanani, N. Sauer and J. Schonheim, Pages 69-87, 1970.

[5] A. Federgruen and H. Groenevelt, Characterization and optimization of achievable performance in general queueing systems, *Operations Research* 36(5): 733 - 741, 1988.

[6] L. Fleischer and S. Iwata, A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics* 131(2): 311–322, 2003.

[7] A. Frank and E. Tardos, Generalized polymatroids and submodular flows, *Mathematical Programming* 42: 489–563, 1988.

[8] S. Fujishige, *Submodular Functions and Optimization*, 2nd ed. Annals of Discrete Mathematics vol. 58, Elsevier, Amsterdam, 2005.

[9] H. Groenevelt, Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *European Journal of Operational Research* 54(2): 227–236, 1991.

[10] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1: 169–197, 1981.

[11] S. Iwata, Submodular function minimization. *Mathematical Programming* 112(1): 45–64, 2008.

[12] S. Iwata and L. Fleischer and S. Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Proc. of 32nd Symposium on Theory of Computing* (STOC'00), pp. 97-106, 2000.

[13] S. Iwata and J. B. Orlin, A simple combinatorial algorithm for submodular function minimization. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'09), pages 1230–1237, 2009.

[14] Y. T. Lee, A. Sidford, and S. C. Wong, A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science* (FOCS'15), pages 1049–1065, 2015.

[15] H. Luss and S. Gupta, Allocation of effort resources among competing activities. *Operations Research* 23(2): 360-366, 1975.

[16] J. Mo and J. Walrand, Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Networking* 8(5): 556–567, 2000.

[17] K. Murota and A. Shioura, Extension of M-Convexity and L-Convexity to Polyhedral Convex Functions. *Advances in Applied Mathematics* 25(4): 352-427, 2000.

[18] J. B. Orlin, A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming* 118(2): 237–251, 2009.

[19] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1996.

[20] A. Schrijver, A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory*, Series B, 80(2): 346–355, 2000.

[21] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, Springer, 2003.

[22] N. V. Shakhlevich, A. Shioura, and V. A. Strusevich, Single machine scheduling with controllable processing times by submodular optimization. *Int. J. Found. Comput. Sci.* 20: 247–269, 2009.

[23] J. G. Shanthikumar and D. D. Yao, Multiclass queueing systems: polymatroidal structure and optimal scheduling control, *Operations Research* 40(S2): 293-299, 1992.

[24] A. Shioura, N. V. Shakhlevich, and V. A. Strusevic, Decomposition algorithms for submodular optimization with applications to parallel machine scheduling with controllable processing times. *Math. Program., Ser. A* 153: 495–534, 2015.

[25] D. M. Topkis. Minimizing a submodular function on a lattice. *Operations Research* 26(2): 305– 321, 1978.

[26] D. M. Topkis. *Supermodularity and complementarity*, Princeton University Press, 2011.

[27] D. Tse and S. V. Hanly. Multiaccess fading channels - part I: polymatroid structure, optimal resource allocation and throughput capacities. *IEEE Transactions on Information Theory* 44(7): 2796–2994, 1998.

[28] P. H. Zipkin, Simple Ranking Methods for Allocation of One Resource. *Management Science* 26(1): 34-43, 1980.

[29] P.-J. Wan, et al., MWSR over an Uplink Gaussian Channel with Box Constraints: A Polymatroidal Approach, under review.