SIAM J. OPTIM.
Vol. 20, No. 3, pp. 1185–1204

SEPARABLE CONVEX OPTIMIZATION PROBLEMS WITH LINEAR ASCENDING CONSTRAINTS*

ARUN PADAKANDLA† AND RAJESH SUNDARESAN‡

Abstract. Separable convex optimization problems with linear ascending inequality and equality constraints are addressed in this paper. An algorithm that explicitly characterizes the optimum point in a finite number of steps is described. The optimum value is shown to be monotone with respect to a partial order on the constraint parameters. Moreover, the optimum value is convex with respect to these parameters. This work generalizes the existing algorithms of Morton, von Randow, and Ringwald [*Math. Programming*, 32 (1985), pp. 238–241] and Viswanath and Anantharam [*IEEE Trans. Inform. Theory*, 48 (2002), pp. 1295–1318] to a wider class of separable convex objective functions. Computational experiments that compare the proposed algorithm with a standard convex optimization tool are also provided.

Key words. ascending constraints, convex optimization, linear constraints, separable problem

AMS subject classifications. 90C25, 52A41

DOI. 10.1137/07069729X

1. Problem description.

In this paper, we minimize the separable objective function $G : \mathbb{R}^L \to \mathbb{R}$ given by

$$G(y) := \sum_{m=1}^{L} g_m(y_m), \tag{1.1}$$

where $y = (y_1, \ldots, y_L)$, subject to the following linear inequality and equality constraints:

$$y_m \in [0, \beta_m], \quad m = 1, 2, \ldots, L, \tag{1.2}$$

$$\sum_{m=1}^{l} y_m \geq \sum_{m=1}^{l} \alpha_m, \quad l = 1, 2, \ldots, L-1, \tag{1.3}$$

$$\sum_{m=1}^{L} y_m = \sum_{m=1}^{K} \alpha_m. \tag{1.4}$$

Throughout the paper, we assume that the functions $g_m, m = 1, 2, \ldots, L$, satisfy the following conditions:

- $g_m : (a_m, b_m) \to \mathbb{R}$, where $a_m \in [-\infty, 0)$ and $b_m \in (0, +\infty]$ and therefore $a_m < 0 < b_m$;
- g_m is strictly convex in its domain (a_m, b_m);
- g_m is continuously differentiable in its domain (a_m, b_m);

*Received by the editors July 15, 2007; accepted for publication (in revised form) June 10, 2009; published electronically August 19, 2009. This work was supported by the Department of Science and Technology under grant DST0748 and by the University Grants Commission under grant Part (2B) UGC-CAS-(Ph.IV).

http://www.siam.org/journals/siopt/20-3/69729.html

†Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48103.

‡Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560012, India (rajeshs@ece.iisc.ernet.in).

- for each m, there is a point in the domain (a_m, b_m) where the derivative $h_m := g'_m$ equals $\underline{h}(0) := \min_{1 \le l \le L} h_l(0)$, the least slope at 0 among all functions g_m.

We now give a brief description of the constraints. We assume $\beta_m \in (0, b_m]$ for $m = 1, 2, \ldots, L$, $\alpha_m \ge 0$ for $m = 1, 2, \ldots, K$, where $K \ge L$, and naturally

$$\sum_{m=1}^{K} \alpha_m \le \sum_{m=1}^{L} \beta_m. \tag{1.5}$$

We also assume

$$\sum_{m=L}^{K} \alpha_m > 0. \tag{1.6}$$

The inequalities in (1.2) impose positivity and upper bound constraints. Note that if $\beta_m = b_m$, the upper bound constraint is irrelevant because the domain of g_m is (a_m, b_m). The inequalities in (1.3) impose a sequence of *ascending constraints* with increasing heights $\sum_{m=1}^{l} \alpha_m$ indexed by l. Assumption (1.5) is necessary for the constraint set to be nonempty. Without (1.6), it is easy to see that $y_L = 0$, and the problem reduces to a similar one with fewer variables. As a final remark on the constraints, observe that we may set $K = L$ without loss of generality. But allowing $K \ge L$ simplifies the exposition of our solution to this optimization problem.

Our main result is a finite-step algorithm that explicitly identifies the optimal vector that minimizes the separable convex objective function in (1.1).

Such a separable convex optimization problem with linear inequality and equality constraints arises in several settings. Morton, von Randow, and Ringwald [15] study the special case $g_m(y) = v_m y^p$, where $v_m > 0$ and $p > 1$ are constants. The constraints are as in (1.2)–(1.4). They cite two examples. The first is a problem of smoothing of Bellman and Dreyfus [1, p. 105]. The second arises as a special case of some network flow problems that have been transformed to implement the "string solution" of Dantzig [7] and its extension by Veinott [27]. Our interest in this problem stems from its application in the optimization of multiterminal communication systems. In such applications, either power utilized, measured in joules per second, is minimized subject to throughput constraints (see Padakandla and Sundaresan [17]), or throughput achieved, measured in bits per second, is maximized subject to power constraints (see Viswanath and Anantharam [28]). A brief description on how the aforementioned problem arises is given in Example 1 in section 3.

A special case obtained by setting $\alpha_1 = \cdots = \alpha_{L-1} = 0$ in (1.3) renders the ascending constraints irrelevant and results in the minimization of (1.1) subject to (1.2) and (1.4). This is a well-studied problem; see Patriksson's annotated bibliography [18] for several example applications, the history, and the classification of solution approaches. This special case also arises in several settings related to optimization of communication systems; see Tavildar and Viswanath [25], Farrokhi et al. [8], Munz, Pfletschinger, and Speidel [16], Telatar [26], and Chen-Nee et al. [6] for some recent applications. A related special case with upper bound constraints ($\beta_m < b_m$) arises in a multiserver job scheduling context in Bonomi and Kumar [4].

Approaches to solving the above special case fall broadly in two categories. The first is the dual method (see Patriksson [18, sect. 3.1]) that proceeds by identifying the dual variables, thereby implicitly identifying the primal variables (see Charnes and Cooper [5] and an extension to general convex functions by Luss and Gupta [13]).

Zipkin [30] unified several previous approaches and identified the dual variable via zeros of partial sums of inverses g'_m. Our approach falls in this category and extends Zipkin's ideas to the case of ascending constraints (see (2.2) and (2.3)). The second approach, denoted pegging algorithms [18, sect. 3.2], identifies the optimal point recursively via solutions to problems where the constraint (1.2) is relaxed. The dual variables are optimized only implicitly. See Sanathanan [19] for an early work using this approach, Luss and Gupta [13], Bitran and Hax [3],[1] and more recently Stefanov [20], [23]. See also Stefanov [22] and [24] for quadratic and logarithmic objective functions and his book [21] for an introduction to separable problems and iterative techniques. The pegging method sets at least one primal variable per iteration. Our proposed algorithm (Algorithm 1) for the case of ascending constraints (section 2) also sets at least one primal variable per iteration. However, this is done implicitly by identifying optimized dual variables and not via pegging as described above.

Morton, von Randow, and Ringwald [15] and Viswanath and Anantharam [28] minimize (1.1) for specific convex objective functions. Their algorithms are specific to the objective functions considered and do not work for other objective functions.[2] Our algorithm is a generalization that is applicable to any continuously differentiable, separable, and strictly convex objective function.

As in pegging algorithms (cf. Patriksson [18, sect. 3.2]), our algorithm sets at least one primal variable per iteration and therefore terminates after identifying the optimal point in at most L steps. Each step involves calculations of zeros of several (up to L) functions (cf. (2.2) and (2.3)). In cases where these zeros can be found explicitly, our algorithm is significantly faster than a standard convex optimization tool. In other cases, the zeros have to be found numerically using line searches. Since the number of such zero-calculations in each step is in the worst case linear in the number of variables, our algorithm may be inefficient for such large problem sets. Computational experiments where our algorithm outperforms and is outperformed by a standard optimization tool are given in section 4.

The paper is organized as follows. Section 2 contains a description of the algorithm. Under a further condition on the functions which will be stated in section 2, we argue that our algorithm provides the solution to the above optimization problem with the additional ordering constraint $y_1 \geq y_2 \geq \cdots \geq y_L$. Furthermore, when the equality constraint on the sum is relaxed to a lower bound constraint, our algorithm outputs the optimal point under the alternative condition that the slopes of g_m at 0 are all positive. Section 3 discusses two illustrative examples and section 4 some computational experiments. Section 5 contains the proof of optimality of the algorithm.

2. The main results. We begin with some remarks on notation.

- For integers i, j satisfying $i \leq j$, we let $[\![i, j]\!]$ denote the set $\{i, i+1, \ldots, j\}$.
- For a set $s \subseteq [\![1, L]\!]$, with $i = \min s$, $j = \max s$, and $l \in [\![i, j]\!]$, let $s\,[l]$ denote the subset $s \cap [\![i, l]\!]$.
- Recall that $h_m := g'_m$. Let $\mathbb{E}_m := h_m(\,(a_m, b_m)\,)$, the range of h_m. Also, recall the condition that for each m, there is a point in the domain (a_m, b_m) where the derivative $h_m = g'_m$ equals $\underline{h}\,(0) := \min_{1 \leq l \leq L} h_l\,(0)$, the least slope at 0

[1] Incidentally, the algorithm of Bitran and Hax [3] need not converge; see Kiwiel [12] for an example of nonconvergence and suggested improvements to restore convergence. Similar nonconvergence issues for dual methods were identified and resolved by Kiwiel [11].

[2] Moreover, their algorithms are specific to the case when the slopes at the origin satisfy a certain ordering property. See the fifth remark following the description of Algorithm 1 in section 2.

among all functions g_m. This may be equivalently stated as $\underline{h}(0) \geq h_m(a_m+)$, given that h_m is continuous and strictly increasing, or equivalently

$$\underline{h}(0) \in \cap_{m=1}^{L} \mathbb{E}_m. \tag{2.1}$$

- Denote by $h_m^{-1} : \mathbb{E}_m \to (a_m, b_m)$ the inverse of the continuous and strictly increasing function h_m. The inverse is also continuous and strictly increasing in its domain.
- For convenience, define the functions $H_m : \mathbb{E}_m \to (a_m, \beta_m]$ to be

$$H_m := h_m^{-1} \wedge \beta_m. \tag{2.2}$$

H_m is clearly increasing.[3] Assignments to the variable y_m will be via evaluation of H_m so that the upper bound constraint in (1.2) is automatically satisfied.

- For $s \subseteq [\![1, L]\!]$, a nonempty subset with $i = \min s$, $j = \max s$, and $l \in [\![i, j]\!]$, let $\theta(s, l)$ denote the least $\theta \geq \underline{h}(0)$ that satisfies

$$\sum_{m \in s[l]} H_m(\theta) = \sum_{m=i}^{l} \alpha_m, \tag{2.3}$$

provided the set of such θ is nonempty. Otherwise we say $\theta(s, l)$ does not exist.

The left-hand side of (2.3) includes only those terms with indices m in $s[l]$, a subset of $[\![i, l]\!]$. The right-hand side includes all indices in $[\![i, l]\!]$. Such summations with gaps (on the left-hand side) arise when variables y_m for $m \in [\![i, l]\!] \setminus s[l]$ are already set and are not currently under consideration.

The domain of $\sum_{m \in s[l]} H_m$ is $\cap_{m \in s[l]} \mathbb{E}_m$. The function $\sum_{m \in s[l]} H_m$ is increasing, and, moreover, strictly increasing, until all functions in the sum saturate. So there is no solution to (2.3) when, for example, $\sum_{m=i}^{l} \alpha_m > \sum_{m \in s[l]} \beta_m$.

In general, if we can demonstrate the existence of $\underline{\theta}$ and $\overline{\theta}$, both in the set $\cap_{m \in s[l]} \mathbb{E}_m$, that satisfy

$$\sum_{m \in s[l]} H_m(\underline{\theta}) \leq \sum_{m=i}^{l} \alpha_m \leq \sum_{m \in s[l]} H_m\left(\overline{\theta}\right), \tag{2.4}$$

then the existence of $\theta(s, l) \in \cap_{m \in s[l]} \mathbb{E}_m$ is assured, thanks to the continuity of $\sum_{m \in s[l]} H_m$. Indeed, we may always take $\underline{\theta} = \underline{h}(0)$. This is because our assumption (2.1), $\underline{h}(0) \leq h_m(0)$, $m \in [\![1, L]\!]$, and the increasing property of $H_m, m \in [\![1, L]\!]$ imply

$$\begin{aligned} \sum_{m \in s[l]} H_m\left(\underline{h}(0)\right) &\leq \sum_{m \in s[l]} H_m\left(h_m(0)\right) \\ &= \sum_{m \in s[l]} \left(h_m^{-1}\left(h_m(0)\right) \wedge \beta_m\right) \\ &= 0 \leq \sum_{m=i}^{l} \alpha_m. \end{aligned}$$

[3] We say f is increasing if $a > b$ implies $f(a) \geq f(b)$. If there is strict inequality, we say f is strictly increasing. Similarly we say x is positive if $x \geq 0$ and strictly positive if $x > 0$.

The continuity and increasing property of $\sum_{m\in s[l]} H_m$ further imply that

$$(\underline{\theta} \wedge \underline{h}(0)) \leq \theta(s,l) \leq \overline{\theta}. \tag{2.5}$$

Thus, in order to show existence of $\theta(s,l)$, it is sufficient to identify a $\overline{\theta}$ that satisfies the right-hand side inequality of (2.4). We will have occasion to use this remark a few times in the proof of correctness of the forthcoming algorithm.

- Similarly, for $s \subseteq [\![1, L]\!]$ with $i = \min s$, $j = \max s$, we let $\Theta(s)$ denote the least $\theta \geq \underline{h}(0)$ that satisfies

$$\sum_{m\in s} H_m(\theta) = \sum_{m=i}^{K} \alpha_m, \tag{2.6}$$

provided the set of such θ is nonempty. Otherwise we say $\Theta(s)$ does not exist. Note the difference between (2.3), with $l = j$, and (2.6). The summation in the right-hand side of (2.6) is up to K. To highlight this difference with (2.3), we use the upper case Θ in $\Theta(s)$. The remarks made above on the existence of $\theta(s,l)$ are applicable to $\Theta(s)$.

- We now provide a description of the variables used in the algorithm for ease of reference.
  - n: Iteration number.
  - s_n: Index set of variables that are yet to be set.
  - $i_n := \min s_n$ and $j_n := \max s_n$.
  - N: The last iteration number in which a variable is set.
  - l, m: Temporary pointer locations that satisfy $l \in [\![i_n, j_n]\!]$, and $m \in s_n$, in iteration n.
  - t: Pointer to the variable that satisfies the corresponding ascending constraint with equality; $t \in [\![i_n, j_n]\!]$.
  - z: Pointer to the variable that is assigned the value 0; $z \in s_n$.
  - ξ_n: Choice of the best slope (marginal cost) in iteration n.
  - p_m: Iteration number when variable y_m is set. (This is needed only in the proof.)
  - c_m: A label that indicates the type of ξ_n that sets the variable y_m. The possible labels are $\{\mathcal{A}, \mathcal{A}^*\}$ for Step 3(a) of the algorithm, $\{\mathcal{B}\}$ for Step 3(b) of the algorithm, and $\{\mathcal{C}, \mathcal{C}^*\}$ for Step 3(c) of the algorithm. If c_m is assigned an asterisked label, then the ascending constraint is met with equality for y_m. (This is needed only in the proof.)

We now provide a generalization of the algorithms of Morton, von Randow, and Ringwald [15] and Viswanath and Anantharam [28].

ALGORITHM 1.

- **Inputs:** $K, L, (\alpha_1, \alpha_2, \ldots, \alpha_K), (\beta_1, \beta_2, \ldots, \beta_L)$.
- **Output:** $y^* = (y_1^*, y_2^*, \ldots, y_L^*)$.
- **Step 1: Initialization** `Set` $n \leftarrow 1, s_1 \leftarrow [\![1, L]\!]$ `and go to` **Step 2**.
- **Step 2:** `Set` $i_n \leftarrow \min s_n$, $j_n \leftarrow \max s_n$ `and go to` **Step 3**.
- **Step 3:** `Find` $\Theta(s_n)$, `the solution of (2.6) with` $s \leftarrow s_n$; `find` $\theta(s_n, l)$ `for` $l \in [\![i_n, j_n - 1]\!]$, `solutions of (2.3) for` $s \leftarrow s_n$ `and` l `as chosen.`[4] `Then set`

$$\xi_n = \max\{\Theta(s_n),\ h_l(0) : l \in s_n,\ \theta(s_n, l) : l \in [\![i_n, j_n - 1]\!]\}.$$

[4]Theorem 1 gives a sufficient condition when these quantities can be identified in every iteration.

- **Case 3(a):** If $\xi_n = \Theta(s_n)$, then set

$$\begin{aligned} y_m^* &\leftarrow H_m(\xi_n) \text{ for } m \in s_n \\ p_m &\leftarrow n \text{ for } m \in s_n \\ c_m &\leftarrow \mathcal{A} \text{ for } m \in s_n, m \neq j_n \\ c_{j_n} &\leftarrow \mathcal{A}^* \\ s_{n+1} &\leftarrow \emptyset \\ n &\leftarrow n+1. \end{aligned}$$

  Go to **Step 4**.
- **Case 3(b):** If $\xi_n = h_z(0)$ for some $z \in s_n$, pick the largest such z and set

$$\begin{aligned} y_z^* &\leftarrow 0 \\ p_z &\leftarrow n \\ c_z &\leftarrow \mathcal{B} \\ s_{n+1} &\leftarrow s_n \setminus \{z\} \\ n &\leftarrow n+1. \end{aligned}$$

  Go to **Step 4**.
- **Case 3(c):** If $\xi_n = \theta(s_n, t)$, for $t \in [\![i_n, j_n - 1]\!]$, pick the largest such t and set

$$\begin{aligned} y_m^* &\leftarrow H_m(\xi_n) \text{ for } m \in s_n[t] \\ p_m &\leftarrow n \text{ for } m \in s_n[t] \\ c_m &\leftarrow \mathcal{C} \text{ for } m \in s_n[t-1] \\ c_t &\leftarrow \mathcal{C}^* \\ s_{n+1} &\leftarrow s_n \setminus s_n[t] \\ n &\leftarrow n+1. \end{aligned}$$

  Go to **Step 4**.
- **Step 4: Termination** If $s_n = \emptyset$, then set $N \leftarrow n-1$, output the vector $y^* = (y_1^*, y_2^*, \ldots, y_L^*)$, and stop.
  Else go to **Step 2**. ∎

Remarks.

- Observe that in each iteration (i.e., a call to Step 3) at least one variable is set. So the algorithm terminates within L steps.
- The iterations are indexed by n, where $n \in [\![1, N]\!]$. In each iteration, say n, s_n contains indices of variables that are yet to be set. At the end of this iteration, either all the variables are set (Step 3(a)), or one of them is set to 0 (Step 3(b)), or the variables with indices within a subset of $s_n[t]$ are set (Step 3(c)). The corresponding sets of labels are $\{\mathcal{A}, \mathcal{A}^*\}$, $\{\mathcal{B}\}$, and $\{\mathcal{C}, \mathcal{C}^*\}$, respectively.
- Suppose y is the vector of production levels of L production units. Let g_m represent the cost of operation for production unit $m \in [\![1, L]\!]$, and let G be the overall cost. The production levels y_m set in a particular iteration are set to have the same marginal cost ξ_n, or they are set to operate at capacity. In symbols, $y_m^* = H_m(\xi_{p_m}) = h_m^{-1}(\xi_{p_m}) \wedge \beta_m$.

- Let $s = [\![1, L]\!]$. The quantity $\Theta(s)$ can be interpreted as the price per unit that will make production from all units meet the sum production constraint (1.4) when each production unit operates independently to maximize its profit. Similarly, the quantities $\theta(s, l)$ represent the price per unit that will ensure sufficient production that meets the lth ascending constraint, assuming the units operate to maximize profit at the price offered. The final price (for this iteration) should then be the maximum of all these quantities. This justifies the maximum operation in the evaluation of ξ_n. This price is offered to only a subset of the units, their production levels set, and the process continues in a similar iterative fashion for the remaining units.
- If the slopes of the functions at 0 satisfy $h_1(0) \le h_2(0) \le \cdots \le h_L(0)$, it is sufficient to just consider $h_{j_n}(0)$ in determining ξ_n. (The problem instances considered by Morton, von Randow, and Ringwald [15] and Viswanath and Anantharam [28] satisfy this slope-ordering condition and therefore admit the simplification in the algorithm.)
- Each iteration requires the evaluation of $\Theta(s_n)$ and $\theta(s_n, l)$, $l \in [\![i_n, j_n - 1]\!]$. These are zeros of continuous increasing functions. In Theorem 1, we provide sufficient conditions for these zeros to exist in each iteration step.
- The question of evaluation of these zeros naturally arises. In the specific examples in section 3, we give closed form expressions for $\Theta(s_n)$ and $\theta(s_n, l)$. We also show results of computational experiments for two such cases in section 4. Our specialized algorithm is more efficient than a standard tool in such cases (see section 4 for details). In general, such closed form expressions may not be available and one has to resort to numerical evaluation of the zeros. On one such example in section 4, our algorithm fares favorably for small-sized problems, but it fares poorly for larger problems because of the worst case $O(L)$ number of zeros that need to be calculated in each iteration step. However, the following observations can enable some efficiency in the calculation of the zeros using line search procedures. (1) The functions are continuous and increasing. (2) Closed form expressions for the derivatives can be obtained for implementation of the Newton–Raphson method. (3) In the proof of correctness of the algorithm, we identify $\underline{\theta}$ and $\overline{\theta}$ on either side of the zeros (see (2.4) and (2.5)) to narrow the search window.

We now state the main result of the paper.

THEOREM 1. *If $\theta([\![1, L]\!], l)$, $l \in [\![1, L-1]\!]$, and $\Theta([\![1, L]\!])$ exist, then the following hold:*

- *For every iteration n such that $s_n \neq \emptyset$, the quantities $\Theta(s_n)$ and $\theta(s_n, l)$, $l \in [\![i_n, j_n - 1]\!]$, exist.*
- *Algorithm 1 terminates in $N \le L$ iterations.*
- *The output of Algorithm 1 minimizes (1.1) under the stated constraints.*

We next state a simple corollary to this result which solves a related problem with additional constraints.

COROLLARY 2.1. *If the functions H_m satisfy $H_1 \ge H_2 \ge \cdots \ge H_L$, then under the conditions of Theorem 1, the optimum y^* satisfies $y_1^* \ge y_2^* \ge \cdots \ge y_L^*$.*

Remark. We may use Algorithm 1 to solve the minimization problem with the additional constraints $y_1 \ge y_2 \ge \cdots \ge y_L$ if H_m is pointwise monotone decreasing in the index m.

Before we state some properties of the optimum value function, we make some more definitions for convenience.

- Observe that if $K > L$, the optimum value defined below depends on the

K-tuple $\alpha \in \mathbb{R}_+^K$ only through the L-tuple $(\alpha_1, \ldots, \alpha_{L-1}, \sum_{m=L}^K \alpha_m) \in \mathbb{R}_+^L$. For studying the optimum value, we may therefore restrict our attention to $K = L$. Let $\alpha \in \mathbb{R}_+^L$ and define $\mathcal{G} : \mathbb{R}_+^L \to \mathbb{R} \cup \{+\infty\}$ as follows:

$$\alpha \overset{\mathcal{G}}{\mapsto} \mathcal{G}(\alpha) := \inf \left\{ G(y) : y \in \mathbb{R}^L \text{ satisfies constraints (1.2)–(1.4)} \right\}.$$

We do not place the restrictions (1.5) and (1.6) on α; if the optimization is over an empty set, the infimum is taken to be $+\infty$. Clearly $\mathcal{G} > -\infty$ because it is the infimum of a strictly convex function over a bounded convex set, the set being defined by the constraints (1.2)–(1.4).

- Define a partial order on $\mathbb{R}_+^K$ as follows. We say $\alpha \succeq \tilde{\alpha}$ if

$$\sum_{m=1}^{l} \alpha_m \geq \sum_{m=1}^{l} \tilde{\alpha}_m, \quad l \in [\![1, L]\!],$$

with equality when $l = L$.

This partial order is different from that of majorization (see, for example, Marshall and Olkin [14]). Loosely speaking, $\alpha \succeq \tilde{\alpha}$ indicates that the components for α are lopsided relative to those of $\tilde{\alpha}$. The proposition below says that lopsidedness increases cost.

PROPOSITION 2.2. *The function $\mathcal{G}$ satisfies the following properties:*

- *If $\alpha \succeq \tilde{\alpha}$, then $\mathcal{G}(\alpha) \geq \mathcal{G}(\tilde{\alpha})$.*
- *$\mathcal{G}$ is a convex function.*

The above results are generalizations of those of Viswanath and Anantharam [28]. Proposition 2.2 extends the one-parameter analysis of Zipkin [30, sect. 4] to the case of ascending constraints.

Finally, consider minimizing the separable objective function in (1.1) subject to (1.2), (1.3), and a lower bound constraint

$$\sum_{m=1}^{L} y_m \geq \sum_{m=1}^{K} \alpha_m \tag{2.7}$$

instead of (1.4). We then have the following result.

COROLLARY 2.3. *If the slopes of the functions g_m at 0 are positive, then under the conditions of Theorem* 1*, the output of Algorithm* 1 *minimizes* (1.1) *subject to the constraints* (1.2)*,* (1.3)*, and* (2.7)*.*

The proofs of the above statements are presented in section 5.

3. Examples. Our first example is from Viswanath and Anantharam [28]. It evaluates the sum throughput in a multiterminal communication setting. The following is a brief description.

EXAMPLE 1 (vector Gaussian multiple access channel). *K power-constrained transmitters communicate with a common receiver. Transmitter k can transmit at power at most α_k joules per sample. The set of transmitted signals is confined to a vector space of dimension L. An allocation of power y_m along the orthogonal direction m results in a net throughput of $\frac{1}{2}\log\left(1 + y_m/\sigma_m^2\right)$ in this dimension, where σ_m^2 is the corresponding noise variance. Maximize the throughput*

$$\sum_{m=1}^{L} \frac{1}{2} \log\left(1 + \frac{y_m}{\sigma_m^2}\right),$$

subject to constraints $y_m \geq 0$, $m \in [\![1, L]\!]$, (1.3), *and* (1.4).

The quantities y_m, $m \in [\![1, L]\!]$, are interpreted as the net signal energy per sample along dimension m. The ascending constraints arise because the transmitters are confined to signal along a single direction. We refer the interested reader to [28] for more details.

In the above problem, $(a_m, b_m) = \left(-\sigma_m^2, +\infty\right)$ is the domain of g_m, where

$$g_m(y_m) = -\frac{1}{2}\log\left(1 + \frac{y_m}{\sigma_m^2}\right),$$

and we look at the corresponding minimization problem. Setting $\beta_m = +\infty$, it is easy to verify that g_m, $m \in [\![1, L]\!]$, satisfy all the conditions laid out in section 1:

$$h_m = \frac{-1}{x + \sigma_m^2} \quad \text{and} \quad H_m(\theta) = -\theta^{-1} - \sigma_m^2 \tag{3.1}$$

with domain $\mathbb{E}_m = (-\infty, 0)$. Consequently, θ_1^l, the solution to (2.3), is given by

$$\theta_1^l = \frac{-l}{\sum_{m=1}^{l} \sigma_m^2 + \sum_{m=1}^{l} \alpha_m}, \quad l \in [\![1, L-1]\!], \tag{3.2}$$

and

$$\Theta_1^L = \frac{-L}{\sum_{m=1}^{L} \sigma_m^2 + \sum_{m=1}^{K} \alpha_m}. \tag{3.3}$$

Theorem 1 therefore indicates that Algorithm 1 is applicable. It can be easily checked that Algorithm 1 is equivalent to [28, Algorithm $\mathcal{A}$] for this example. (For those readers interested in the connection between [28, Algorithm $\mathcal{A}$] and Algorithm 1, the following holds: if the output of [28, Algorithm $\mathcal{A}$] is $\mu = (\mu_1, \ldots, \mu_L)$, then the output of our algorithm is $y = (\mu_1 - \sigma_1^2, \ldots, \mu_L - \sigma_L^2)$.)

Remark. The following generalization is claimed in [28, Appendix A.5]. Suppose f_m, $m \in [\![1, L,]\!]$, are of the form $f_m(y) = f\left(1 + y/\sigma_m^2\right)$, where $f : \mathbb{R}_+ \to \mathbb{R}$ is any continuous, increasing, strictly concave function, and consider maximization of $\sum_{m=1}^{L} f_m(y_m)$ subject to (1.2), (1.3), and (1.4). Then $\bar{y} = \left(\mu_1 - \sigma_1^2, \ldots, \mu_L - \sigma_L^2\right)$, where $\mu = (\mu_1, \ldots, \mu_L)$ is the output of [28, Algorithm $\mathcal{A}$], maximizes $\sum_{m=1}^{L} f_m(y_m)$. This is incorrect as demonstrated by the following counterexample.

Let $K = L = 2$, $\alpha_1 = 6$, $\alpha_2 = 12$, $\sigma_1^2 = 2$, $\sigma_2^2 = 4$, and $f_m(y) = \left(1 + y/\sigma_m^2\right)^{1/2}$ for $m = 1, 2$. ($f(y) = y^{1/2}$ is continuous, increasing, strictly concave.) One can further verify that the output of [28, Algorithm $\mathcal{A}$] is $\mu = (12, 12)$, which yields $\bar{y} = (10, 8)$, a feasible vector under the constraints $y_1 \geq 0$, $y_2 \geq 0$, $y_1 \geq 6$, $y_1 + y_2 = 18$. However, the output of Algorithm 1 yields the optimum feasible vector $y^* = (14, 4)$. It is easy to verify that

$$3\sqrt{2} = f_1(y_1^*) + f_2(y_2^*) > f_1(\bar{y}_1) + f_2(\bar{y}_2) = \sqrt{3}(\sqrt{2} + 1),$$

and therefore $\bar{y}$ is not the optimum point.

For $f(y) = \log y$, however, [28, Algorithm $\mathcal{A}$] and Algorithm 1 are equivalent, and therefore their main result that [28, eq. (8)] is maximized by the output of [28, Algorithm $\mathcal{A}$] is indeed correct. □

Our next and final example illustrates the handling of the upper bound constraint. This is just a special case of Bertsekas [2, Ex. 5.1.2], but it illustrates the use of

the functions H_m. The example arises in a power optimization problem for sensor networks (see Zacharias and Sundaresan [29]).

EXAMPLE 2. *Let* $g_m(x) = \frac{1}{2}x^2$, $m \in [\![1, L]\!]$, $K = L$, $\alpha_m = 0$, $m \in [\![1, L-1]\!]$, $\alpha_L = \alpha > 0$, *and* $\beta_m \in (0, \infty)$ *for* $m \in [\![1, L]\!]$. *Further, order the indices so that* $\beta_1 \leq \beta_2 \leq \cdots \leq \beta_L$. *Minimize* (1.1) *under this setting.*

Once again, all conditions outlined in section 1 hold. It is easy to verify that $H_m(\theta) = \theta \wedge \beta_m$. The function $\sum_{m=1}^{l} H_m$ is a piecewise linear continuous function passing through the origin with slope l in $(-\infty, \beta_1)$, slope $(l-1)$ in (β_1, β_2), and so on, and zero-slope in $(\beta_l, +\infty)$. Clearly $\theta_1^l = 0$, $l \in [\![1, L-1]\!]$. We assume that Θ_1^L exists, which is equivalent to $\alpha \leq \sum_{m=1}^{L} \beta_m$. Yet again, Theorem 1 assures us that Algorithm 1 is applicable.

An application of Algorithm 1 results in the following. Identify the unique k such that $\alpha \in [a_k, a_{k+1})$, where $a_0 = 0$, and

$$a_k := (L-k)\beta_k + \sum_{m=1}^{k} \beta_m, \quad k = 1, \ldots, L.$$

The quantity a_k denotes the ordinate of the piecewise linear continuous function at abscissa β_k, a location where the slope changes. It is easy to see that $a_k \leq a_{k+1}$. Interpolating to find the abscissa when the ordinate is α, we get

$$\Theta_1^L = \frac{\alpha - \sum_{m=1}^{k} \beta_m}{L-k}.$$

Moreover, $y_m^* = H_m\left(\Theta_1^L\right) = \beta_m$ for $m \in [\![1, k]\!]$. For $m \in [\![k+1, L]\!]$, the values are suitably lowered from their upper bounds. Note that this assignment is completed in just one iteration of Algorithm 1. □

4. Computational experiments. In this section we discuss results of some computational experiments. We used `CVX` (see [9] and [10]) to solve three further examples and compared `CVX`'s performance against that of Algorithm 1.[5] `CVX` used the core solver `SDPT3` to solve the dual problem using a default precision of 1.5×10^{-8}. The number of variables L for each problem is indicated in Table 4.1. Each problem is parameterized by weights $v_m \in \mathbb{R}_+$ (see descriptions of objective functions below) and α_m, $m \in [\![1, L]\!]$, that determine the ascending constraints (1.3) and (1.4). All these parameters were chosen independently and with the uniform probability density on $[0, 1]$. The obtained v_m, $m \in [\![1, L]\!]$, were then sorted so that the resulting parameters satisfied $v_1 \leq v_2 \leq \cdots \leq v_m$ (see fifth remark immediately after description of Algorithm 1). For each objective function, 30 such random instances of the problem were chosen and provided to both the `SDPT3` solver and Algorithm 1. The mean time taken (averaged over the 30 instances), the mean number of iterations taken by the `SDPT3` solver, and the mean number of calls to the solver are indicated in Table 4.1 and compared with the mean time and iterations for Algorithm 1.

The first problem is

$$\min \sum_{m=1}^{L} \frac{v_m}{1 - y_m} \text{ subject to (1.2), (1.3), and (1.4),}$$

[5] We used `CVX` version 1.2 (build 706) and MATLAB version 7.5.0.338 (R2007b). The programs were run on an Intel Xeon CPU 3.20GHz machine running on SLES 9.0 OS.

TABLE 4.1
Performance comparisons.

Objective	L	SDPT3			Algorithm 1	
		Time	Iter.	Calls	Time	Iter.
$\sum_m v_m/(1-y_m)$	500	22.3400 s	27.9	1	0.0013 s	7.1
$-\sum_m \log(v_m+y_m)$	500	267.3403 s	-	5.3	0.0010 s	4.1
$\sum_m \left(\frac{1}{4}x_m^4 + v_m x_m\right)$	50	1.6767 s	24.7	1	0.3537 s	13.4
$\sum_m \left(\frac{1}{4}x_m^4 + v_m x_m\right)$	150	5.2433 s	29.1	1	7.7187 s	38.6

where in (1.2) we take $\beta_m = 1$ for all m. It is straightforward to see that (2.3) can be explicitly solved:

$$\theta = \left(\frac{\sum_{m\in s[l]} \sqrt{v_m}}{|s[l]| - \sum_{m=i}^{l} \alpha_m} \right)^2,$$

where l and i are as defined prior to (2.3). A similar solution holds for (2.6).

The second problem is

$$\min \ -\sum_{m=1}^{L} \log(v_m + y_m) \text{ subject to (1.2), (1.3), and (1.4),}$$

where in (1.2) we take $\beta_m = +\infty$ for all m. This problem is equivalent to that considered in Example 1. The zeros of (2.3) are explicitly solved (cf. (3.2)):

$$\theta = \frac{-|s[l]|}{\sum_{m\in s[l]} v_m + \sum_{m=i}^{l} \alpha_m},$$

where l and i are once again as defined prior to (2.3). A similar solution holds once again for (2.6). Since the log function has to be numerically evaluated, CVX uses a successive approximation method that requires multiple calls to the solver. This increases the time taken for CVX to solve the problem. The mean number of calls is indicated in Table 4.1. Since CVX does not yet report the number of iterations in each call, this column is left blank in Table 4.1.

The third problem is

$$\min \sum_{m=1}^{L} \left(\frac{1}{4} y_m^4 + v_m y_m \right) \text{ subject to (1.2), (1.3), and (1.4),}$$

where in (1.2) we take $\beta_m = +\infty$ for all m. Then we seek a θ that solves (2.3) rewritten as

$$\sum_{m\in s[l]} (\theta - v_m)^{\frac{1}{3}} = \sum_{m=i}^{l} \alpha_m$$

with (2.6) rewritten in a similar fashion. We do not know an explicit expression for the root of this equation unlike for the previous two problems. However, the derivative

of the function on the left-hand side is easy to find and evaluate explicitly. This was used in our implementation of a modification of the Newton–Raphson method[6] to find the zeros to within precision 1.5×10^{-8}. Cases with 50 variables and 150 variables are reported in the third and fourth rows of the table.

Table 4.1 clearly shows several orders of magnitude improvement for Algorithm 1 over `SDPT3` for our specific convex optimization problem, when the zeros can be explicitly found. See the first two rows corresponding to the first and the second problems. The third problem is a small-sized problem where Algorithm 1 still outperforms `SDPT3`. However, in the fourth problem, the worst case number of zeros to be calculated is large and our implementation performs worse than `SDPT3`.[7]

5. Proofs.

5.1. Preliminaries.

We first prove some facts on the individual cases.

LEMMA 5.1. *Suppose that in iteration n, the quantities $\theta(s_n, l)$, $l \in [\![i_n, j_n - 1]\!]$, and $\Theta(s_n)$ exist. Suppose further that $\xi_n = h_z(0)$ for some $z \in s_n$, and Step* 3(b) *is executed. Then the following hold:*

- *The quantities $\theta(s_{n+1}, l)$, $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$, and $\Theta(s_{n+1})$ exist.*
- $\xi_n \geq \xi_{n+1}$.

Proof. Given that $\xi_n = h_z(0)$ for some $z \in s_n$, and Step 3(b) is executed, we see that $s_{n+1} = s_n \setminus \{z\}$. We may assume s_{n+1} is nonempty, or equivalently $i_{n+1} \leq j_{n+1}$; otherwise there is nothing to prove. We first prove the existence of $\theta(s_{n+1}, l)$, $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$.

If $z = j_n$, then $s_{n+1} = s_n \setminus \{j_n\}$. Since $i_{n+1} = i_n$, $j_{n+1} < j_n$, or equivalently $j_{n+1} - 1 < j_n - 1$, we have $s_{n+1}[l] = s_n[l]$ for $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$. Therefore, it is clear that $\theta(s_{n+1}, l) = \theta(s_n, l)$ for $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$ because $\theta(s_{n+1}, l)$ and $\theta(s_n, l)$ are zeros of identical functions for the indicated values of l.

We next consider $z = i_n$ and $z > i_n$.

If $z = i_n$, then $s_{n+1} = s_n \setminus \{i_n\}$. Since $i_{n+1} > i_n$, $j_{n+1} = j_n$, we have $i_{n+1} - 1 \in [\![i_n, j_n - 1]\!]$. Since Step 3(b) is executed with $z = i_n$, we must have $\xi_n = h_{i_n}(0) \geq \theta(s_n, i_{n+1} - 1)$, and therefore

$$0 = h_{i_n}^{-1}(h_{i_n}(0)) = h_{i_n}^{-1}(\xi_n) \geq H_{i_n}(\xi_n) \overset{\text{(a)}}{\geq} H_{i_n}(\theta(s_n, i_{n+1} - 1)) = \sum_{m=i_n}^{i_{n+1}-1} \alpha_m \geq 0,$$

where (a) follows because H_{i_n} is increasing. Consequently, we have

$$\sum_{m=i_n}^{i_{n+1}-1} \alpha_m = 0. \tag{5.1}$$

When $z > i_n$, $i_{n+1} = i_n$ and (5.1) holds trivially on account of being a sum over an empty index set. We may therefore write

$$\sum_{m=i_{n+1}}^{l} \alpha_m = \sum_{m=i_n}^{l} \alpha_m - \sum_{m=i_n}^{i_{n+1}-1} \alpha_m = \sum_{m=i_n}^{l} \alpha_m, \quad l \in [\![i_{n+1}, K]\!]. \tag{5.2}$$

[6]The modification is the following: if the iterations exhibited two consecutive sign changes for the function, the iterations would continue from the zero of the linear approximation through the last two points.

[7]The gains suggested in Table 4.1 are suggestive only of what one might expect since no special effort was made to optimize the performances of `SDPT3` or Algorithm 1.

If $i_{n+1} \le l < z$, we have $s_{n+1}[l] = s_n[l]$. On account of (5.2), $\theta(s_{n+1}, l)$ and $\theta(s_n, l)$ are zeros of identical functions and thus

$$\theta(s_{n+1}, l) = \theta(s_n, l), \quad i_{n+1} \le l < z, \tag{5.3}$$

thereby resolving the existence question for such $\theta(s_{n+1}, l)$.

For $l \in [\![z, j_{n+1} - 1]\!]$, we observe that $\xi_n = h_z(0) \ge \theta(s_n, l)$ and the increasing nature of H_z yield

$$H_z(\theta(s_n, l)) \le H_z(h_z(0)) \le h_z^{-1}(h_z(0)) = 0. \tag{5.4}$$

Using this and (5.2), we get

$$\begin{aligned}
\sum_{m=i_{n+1}}^{l} \alpha_m &= \sum_{m=i_n}^{l} \alpha_m \\
&= \sum_{m \in s_n[l]} H_m\left(\theta\left(s_n, l\right)\right) \quad \text{(from definition of } \theta\left(s_n, l\right)\text{)} \\
&= \sum_{m \in s_{n+1}[l]} H_m\left(\theta\left(s_n, l\right)\right) + H_z\left(\theta\left(s_n, l\right)\right) \\
&\le \sum_{m \in s_{n+1}[l]} H_m\left(\theta\left(s_n, l\right)\right) \quad \text{(from (5.4))}. \qquad (5.5)
\end{aligned}$$

So we may take $\overline{\theta} = \theta\left(s_n, l\right)$ in (2.4). On the lower side, we may simply use $\underline{\theta} = \underline{h}\left(0\right)$. $\theta\left(s_n, l\right)$ therefore exists and

$$\underline{h}\left(0\right) \le \theta\left(s_{n+1}, l\right) \le \theta\left(s_n, l\right). \tag{5.6}$$

We have thus verified the existence of $\theta(s_{n+1}, l)$ for all $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$.

Following the same argument leading to (5.5) with $\sum_{m=i_{n+1}}^{K} \alpha_m$ and s_n instead of $\sum_{m=i_{n+1}}^{l} \alpha_m$ and $s_n[l]$, respectively, establishes the existence of $\Theta(s_{n+1})$ and that

$$\underline{h}\left(0\right) \le \Theta\left(s_{n+1}\right) \le \Theta\left(s_n\right). \tag{5.7}$$

This establishes the existence part of the lemma.

To establish $\xi_n \ge \xi_{n+1}$, we simply observe that ξ_n is at least as large as all the candidates that determine ξ_{n+1}. Indeed, $\xi_n = h_z(0) \ge h_l(0)$ for $l \in s_n \setminus \{z\} = s_{n+1}$. Next $\xi_n \ge \theta\left(s_n, l\right) \ge \theta\left(s_{n+1}, l\right)$, $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$, follows from the right-hand side inequality of (5.6). Finally, using the right-hand side inequality of (5.7), $\xi_n > \Theta\left(s_n\right) \ge \Theta\left(s_{n+1}\right)$. This completes the proof of the lemma. ∎

LEMMA 5.2. *Suppose that in iteration n, the quantities $\theta\left(s_n, l\right)$, $l \in [\![i_n, j_n - 1]\!]$, and $\Theta\left(s_n\right)$ exist. Suppose further that $\xi_n = \theta\left(s_n, t\right)$ for some $t \in [\![i_n, j_n - 1]\!]$, and Step* 3(c) *is executed. Then the following hold:*

- $y_m^* \in [0, \beta_m]$, $m \in s_n\left[t\right]$.
- $\sum_{m=i_n}^{l} y_m^* \ge \sum_{m=i_n}^{l} \alpha_m$, $l \in [\![i_n, t]\!]$, *with equality when* $l = t$.
- *The quantities* $\theta\left(s_{n+1}, l\right)$, $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$, *and* $\Theta\left(s_{n+1}\right)$ *exist.*
- $\xi_n \ge \xi_{n+1}$.

Proof. Note that in this case t is chosen to be the largest one in $[\![i_n, j_n - 1]\!]$ that satisfies $\xi_n = \theta\left(s_n, t\right)$. Step 3(c) is executed; therefore $\theta\left(s_n, t\right) \ge h_m(0)$ for $m \in s_n\left[t\right]$. The assignment for y_m^* in the algorithm satisfies

$$y_m^* = H_m\left(\theta\left(s_n, t\right)\right) \ge H_m\left(h_m(0)\right) = h_m^{-1}\left(h_m(0)\right) \wedge \beta_m = 0 \wedge \beta_m = 0.$$

That $y_m^* \leq \beta_m$ is obvious from the definition of H_m. This proves the upper and lower bound constraints on y_m^*.

We now prove that the ascending constraints (with the sum starting from i_n) hold for $l \in [\![i_n, t]\!]$. For $m \in [\![i_n, l]\!] \setminus s_n[l]$, we have $c_m = \mathcal{B}$, and from the assignment in Step 3(b), $y_m^* = 0$. Thus, it is sufficient to prove

$$\sum_{m \in s_n[l]} y_m^* \geq \sum_{m=i_n}^{l} \alpha_m$$

for $l \in [\![i_n, t]\!]$. Indeed, $\theta(s_n, t) \geq \theta(s_n, l)$ and the increasing property of H_m imply

$$\sum_{m \in s_n[l]} y_m^* = \sum_{m \in s_n[l]} H_m(\theta(s_n, t)) \geq \sum_{m \in s_n[l]} H_m(\theta(s_n, l)) = \sum_{m=i_n}^{l} \alpha_m,$$

with equality when $l = t$, and the second statement is proved.

We next consider the existence of $\theta(s_{n+1}, l)$. We claim that $i_{n+1} = t + 1$. Assuming the contrary, i.e., $i_{n+1} > t + 1$, implies $s_n[t+1] = s_n[t]$. From the positivity of the components of α, we have

$$\begin{aligned}\sum_{m \in s_n[t]} H_m(\theta(s_n, t)) = \sum_{m=i_n}^{t} \alpha_m \leq \sum_{m=i_n}^{t+1} \alpha_m &= \sum_{m \in s_n[t+1]} H_m(\theta(s_n, t+1)) \\ &= \sum_{m \in s_n[t]} H_m(\theta(s_n, t+1)).\end{aligned}$$

From the increasing nature of H_m, we have $\theta(s_n, t+1) \geq \theta(s_n, t)$, thus contradicting the maximality of $\theta(s_n, t)$ and the choice of t. This proves the claim.

We now identify $\underline{\theta}$ and $\overline{\theta}$ to prove the existence of $\theta(s_{n+1}, l)$. Fix $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$. We simply set $\underline{\theta} = \underline{h}(0)$ in (2.4). Moreover,

$$\begin{aligned}\sum_{m=i_{n+1}}^{l} \alpha_m &= \sum_{m=i_n}^{l} \alpha_m - \sum_{m=i_n}^{t} \alpha_m \qquad (t = i_{n+1} - 1) \\ &= \sum_{m \in s_n[l]} H_m(\theta(s_n, l)) - \sum_{m \in s_n[t]} H_m(\theta(s_n, t)) \ \text{(by definition)} \\ &\leq \sum_{m \in s_n[l]} H_m(\theta(s_n, l)) - \sum_{m \in s_n[t]} H_m(\theta(s_n, l)) \ \text{(because } H_m \text{ is increasing)} \\ &= \sum_{m \in s_{n+1}[l]} H_m(\theta(s_n, l)),\end{aligned}$$

and therefore we may set $\overline{\theta} = \theta(s_n, l)$ in (2.4). $\theta(s_{n+1}, l)$ therefore exists and

$$\underline{h}(0) \leq \theta(s_{n+1}, l) \leq \theta(s_n, l). \tag{5.8}$$

The same argument (mutatis mutandis to account for the sum of α_m up to K) establishes the existence of $\Theta(s_{n+1})$ and that

$$\underline{h}(0) \leq \Theta(s_{n+1}) \leq \Theta(s_n). \tag{5.9}$$

Finally, to show $\xi_n \geq \xi_{n+1}$, observe that $\xi_n \geq h_l(0)$, $l \in s_{n+1}$, $\xi_n \geq \Theta(s_n) \geq \Theta(s_{n+1})$, and $\xi_n \geq \theta(s_n, l) \geq \theta(s_{n+1}, l)$, $l \in [\![i_{n+1}, j_{n+1} - 1]\!]$. The last two facts follow from (5.9) and (5.8). So ξ_n is at least as large as all the candidates that determine ξ_{n+1}, i.e., $\xi_n \geq \xi_{n+1}$, and the proof is complete. ∎

LEMMA 5.3. *Suppose that in iteration n, the quantities $\theta(s_n, l)$, $l \in [\![i_n, j_n - 1]\!]$, and $\Theta(s_n)$ exist. Suppose further that $\xi_n = \Theta(s_n)$. Then the following hold:*

- $y^*_m \in [0, \beta_m]$, $m \in s_n$.
- $\sum_{m=i_n}^{l} y^*_m \geq \sum_{m=i_n}^{l} \alpha_m$, $l \in [\![i_n, j_n]\!]$, *with equality when* $l = j_n$.

Proof. Under the hypotheses, Step 3(a) is executed. The proofs of the statements are identical to the proofs of the first two parts of Lemma 5.2 and are omitted. ∎

PROPOSITION 5.4. *If $\theta([\![1, L]\!], l)$, $l \in [\![1, L-1]\!]$, and $\Theta([\![1, L]\!])$ exist, then the following statements hold:*

- *For every iteration step n, with s_n nonempty, the quantities $\Theta(s_n)$ and $\theta(s_n, l)$, $l \in [\![i_n, j_n - 1]\!]$, exist.*
- *Algorithm 1 terminates in $N \leq L$ steps.*
- *The output y^* of Algorithm 1 is feasible.*
- $\xi_1 \geq \xi_2 \geq \cdots \geq \xi_N$.
- *In iteration N, Step* 3(a) *is executed.*

Proof. The key issue is the existence of $\theta(s_n, l)$ and $\Theta(s_n)$ in Step 3 of each iteration. The hypothesis of this proposition resolves the issue for $n = 1$. Lemmas 5.1, 5.2, and 5.3 resolve the issue for subsequent iterations via induction. The first statement follows.

At least one variable is set in every iteration. The algorithm thus runs to completion in $N \leq L$ iterations, and the second statement holds.

The third and fourth statements also follow from Lemmas 5.1, 5.2, and 5.3 and induction.

We now argue that Step 3(a) is executed in the last iteration. If this is not the case, the last iteration must be Step 3(b). This implies $s_N = \{j_N\}$ $(= \{i_N\})$ and $h_{j_N}(0) > \Theta(s_N)$. The latter inequality and the definition of $\Theta(s_N)$ yield

$$\sum_{m=j_N}^{K} \alpha_m = H_{j_N}(\Theta(s_N)) \leq h_{j_N}^{-1}(\Theta(s_N)) \leq h_{j_N}^{-1}(h_{j_N}(0)) = 0,$$

contradicting our assumption (1.6) that $\sum_{m=L}^{K} \alpha_m > 0$. ∎

5.2. Proof of Theorem 1. Proposition 5.4 implies the first two statements of Theorem 1. We now proceed to show the optimality of y^* to complete the proof of Theorem 1.

We use the Karush–Kuhn–Tucker (KKT) conditions (see, for example, [2, sect. 3.3]) to show that the vector put out by the algorithm is a stationary point of a Lagrangian function with appropriately chosen Lagrange multipliers. The Lagrangian function for the problem is

$$\begin{aligned} &\sum_{m=1}^{L} g_m(y_m) + \sum_{m=1}^{L} \lambda_m^{(1)}(-y_m) + \sum_{m=1}^{L} \lambda_m^{(2)}(y_m - \beta_m) \\ &\quad + \sum_{l=1}^{L-1} \lambda_l^{(3)} \left(-\sum_{m=1}^{l} y_m + \sum_{m=1}^{l} \alpha_m \right) + \mu \left(-\sum_{m=1}^{L} y_m + \sum_{m=1}^{K} \alpha_m \right), \end{aligned} \tag{5.10}$$

where $\lambda_m^{(1)}$ is the Lagrange multiplier that relaxes the positivity constraint $-y_m \leq 0$, $\lambda_m^{(2)}$ relaxes the upper bound constraint $y_m - \beta_m \leq 0$, $\lambda_m^{(3)}$ relaxes the ascending constraint (1.3), and μ relaxes the equality constraint (1.4). The KKT necessary and sufficient conditions for optimality of this convex optimization problem are given by

$$\lambda_m^{(1)} y_m = 0, \quad m \in [\![1, L]\!], \tag{5.11}$$

$$\lambda_m^{(2)} (y_m - \beta_m) = 0, \quad m \in [\![1, L]\!], \tag{5.12}$$

$$\lambda_l^{(3)} \left(\sum_{m=1}^{l} y_m - \sum_{m=1}^{l} \alpha_m \right) = 0, \quad l \in [\![1, L-1]\!], \tag{5.13}$$

$$\lambda_m^{(1)} \geq 0, \quad \lambda_m^{(2)} \geq 0, \quad m \in [\![1, L]\!], \quad \text{and} \quad \lambda_l^{(3)} \geq 0, \quad l \in [\![1, L-1]\!], \tag{5.14}$$

$$h_m(y_m) - \lambda_m^{(1)} + \lambda_m^{(2)} - \sum_{l=m}^{L-1} \lambda_l^{(3)} - \mu = 0, \quad m \in [\![1, L]\!]. \tag{5.15}$$

Conditions (5.11), (5.12), and (5.13) are the complementary slackness conditions, (5.14) are the positivity conditions, and (5.15) identifies a stationary point for the Lagrangian function. We now choose appropriate values for the Lagrange multipliers and verify the KKT conditions.

First, let

$$\lambda_m^{(1)} = \begin{cases} \xi_{p_m} - \xi_N & \text{if } c_m = \mathcal{B} \text{ and } m > j_N, \\ \xi_{p_m} - \xi_{t_m} & \text{if } c_m = \mathcal{B} \text{ and } m < j_N, \\ 0 & \text{otherwise,} \end{cases} \tag{5.16}$$

where

$$t_m := \min \{ p_l : l \in [\![m, L]\!], p_l > p_m, c_l \in \{\mathcal{C}^*, \mathcal{A}^*\} \}. \tag{5.17}$$

The last iteration is always via Step 3(a) (Proposition 5.4). Thus, when $c_m = \mathcal{B}$ and $m < j_N$, there is a later iteration that executes Step 3(a) which implies that the set in (5.17) is nonempty and that the assignment (5.16) is well-defined. Recall that p_m is the iteration number in which variable y_m was set, and that $c_m = \mathcal{B}$ whenever Step 3(b) is executed, i.e., $y_m^* = 0$. From the assignment in (5.16), $\lambda_m^{(1)} \neq 0$ implies that $c_m = \mathcal{B}$ and therefore $y_m^* = 0$. Thus the complementary slackness condition (5.11) is satisfied for $m \in [\![1, L]\!]$.

Second, let

$$\lambda_m^{(2)} = \begin{cases} 0 & \text{if } c_m = \mathcal{B}, \\ \xi_{p_m} - h_m \left(H_m \left(\xi_{p_m} \right) \right) & \text{otherwise.} \end{cases} \tag{5.18}$$

If $\lambda_m^{(2)} \neq 0$, then from (5.18) we have $\xi_{p_m} \neq h_m \left(H_m \left(\xi_{p_m} \right) \right)$. From the strictly increasing property of h_m and the definition of H_m, we have

$$h_m \left(H_m \left(\xi_{p_m} \right) \right) = h_m \left(h_m^{-1} \left(\xi_{p_m} \right) \wedge \beta_m \right) = h_m \left(h_m^{-1} \left(\xi_{p_m} \right) \right) \wedge h_m \left(\beta_m \right) = \xi_{p_m} \wedge h_m \left(\beta_m \right), \tag{5.19}$$

so that $h_m \left(H_m \left(\xi_{p_m} \right) \right) \neq \xi_{p_m}$ implies that $H_m \left(\xi_{p_m} \right)$ must have saturated to β_m, i.e., $y_m^* = H_m \left(\xi_{p_m} \right) = \beta_m$. The complementary slackness condition (5.12) is therefore fulfilled.

Third, for $m \in [\![1, L-1]\!]$ let

$$\lambda_m^{(3)} = \begin{cases} \xi_{p_m} - \xi_{t_m} & \text{if } c_m = \mathcal{C}^*, \\ 0 & \text{otherwise,} \end{cases} \tag{5.20}$$

where t_m is defined via (5.17). $\lambda_m^{(3)}$ is well-defined because $c_m = \mathcal{C}^*$ implies that there is a later iteration that executes Step 3(a) and that the set in (5.17) is nonempty. Suppose $\lambda_m^{(3)} \neq 0$. Then $c_m = \mathcal{C}^*$, an asterisked assignment. The second statement of Lemma 5.2 therefore ensures that the ascending constraint is satisfied with equality for this m. The complementary slackness condition (5.13) is thus fulfilled for $m \in [\![1, L-1]\!]$.

The assignment of $\lambda_m^{(3)}$ in (5.20) can be equivalently expressed as

$$\lambda_m^{(3)} = \begin{cases} \sum_{n=p_m}^{t_m-1} \xi_n - \xi_{n+1} & \text{if } c_m = \mathcal{C}^*, \\ 0 & \text{otherwise,} \end{cases} \tag{5.21}$$

where t_m is given by (5.17). This will be useful in verifying (5.15).

Finally, we set $\mu = \xi_N$.

The Lagrange multiplier assignments in (5.16), (5.18), and (5.20) are positive. Indeed, the positivity in (5.16) and (5.20) follows from the monotonicity property $\xi_n \geq \xi_{n+1}$, $n \in [\![1, N-1]\!]$ (Proposition 5.4). The positivity of $\lambda_m^{(2)}$ follows from

$$h_m\left(H_m\left(\xi_{p_m}\right)\right) \leq h_m\left(h_m^{-1}\left(\xi_{p_m}\right)\right) = \xi_{p_m}.$$

All that remains is to verify (5.15). To do this, first consider $m > j_N$. Then the assignments $y_m^* = 0$ and $y_l^* = 0$, $l \in [\![m+1,\ L-1]\!]$, are via Step 3(b); therefore $\xi_{p_l} = h_l(0)$ and $c_l = \mathcal{B}$. The latter implies $\lambda_m^{(1)} = \xi_{p_m} - \xi_N$, $\lambda_m^{(2)} = 0$, and $\lambda_l^{(3)} = 0$ for $l \in [\![m, L-1]\!]$. Substitution of these assignments in (5.15) yields

$$h_m(0) - \lambda_m^{(1)} + \lambda_m^{(2)} - \sum_{l=m}^{L-1} \lambda_l^{(3)} - \mu = \xi_{p_m} - (\xi_{p_m} - \xi_N) + 0 - 0 - \xi_N = 0.$$

Now consider $m \in [\![1, j_N]\!]$ and $c_m = \mathcal{B}$. Substitution of (5.16), (5.18), and (5.20) in (5.15) yields

$$\begin{aligned} & h_m\left(y_m^*\right) - \lambda_m^{(1)} + \lambda_m^{(2)} - \sum_{l=m}^{L-1} \lambda_l^{(3)} - \mu \\ &= \xi_{p_m} - (\xi_{p_m} - \xi_{t_m}) + 0 - \sum_{l=m}^{L-1} \lambda_l^{(3)} - \xi_N \\ &= \xi_{t_m} - \xi_N - \sum_{l=m}^{L-1} \lambda_l^{(3)} \qquad (5.22) \\ &= \xi_{t_m} - \xi_N - \sum_{n=t_m}^{N-1} (\xi_n - \xi_{n+1}) \qquad (5.23) \\ &= \xi_{t_m} - \xi_N - (\xi_{t_m} - \xi_N) \\ &= 0. \end{aligned}$$

In the above sequence of inequalities, (5.23) holds because of the following. In (5.22), the summation over l has only one nonzero entry per iteration, i.e., whenever $c_l = \mathcal{C}^*$. We may therefore sum over the iteration index n instead of the variable index l. Iterations t_m to $N-1$ involve the execution of either Step 3(b) or Step 3(c). Substitution of (5.21) in (5.22) then results in (5.23).

Suppose $m \in [\![1, j_N]\!]$ and $c_m \in \{\mathcal{C}, \mathcal{C}^*\}$. Substitution of (5.16), (5.18), and (5.20) in (5.15) yields

$$\begin{aligned}
&h_m\left(y_m^*\right) - \lambda_m^{(1)} + \lambda_m^{(2)} - \sum_{l=m}^{L-1} \lambda_l^{(3)} - \mu \\
&= h_m\left(H_m\left(\xi_{p_m}\right)\right) - 0 + \xi_{p_m} - h_m\left(H_m\left(\xi_{p_m}\right)\right) - \sum_{l=m}^{L-1} \lambda_l^{(3)} - \xi_N \\
&= \xi_{p_m} - \xi_N - \sum_{l=m}^{L-1} \lambda_l^{(3)} \\
&= \xi_{p_m} - \xi_N - \sum_{n=p_m}^{N-1} \left(\xi_n - \xi_{n+1}\right) \qquad \text{(follows from substitution of (5.21))} \\
&= 0.
\end{aligned}$$

Last, consider $m \in [\![1, j_N]\!]$ and $p_m = N$; i.e., y_m is assigned in the last iteration. From Proposition 5.4, Step 3(a) is executed in this iteration, and therefore $c_m \in \{\mathcal{A}, \mathcal{A}^*\}$. Then

$$\begin{aligned}
&h_m\left(y_m\right) - \lambda_m^{(1)} + \lambda_m^{(2)} - \sum_{l=m}^{L-1} \lambda_l^{(3)} - \mu \\
&\quad = h_m\left(H_m\left(\xi_{p_m}\right)\right) - 0 + \left(\xi_N - h_m\left(H_m\left(\xi_{p_m}\right)\right)\right) - 0 - \xi_N = 0.
\end{aligned}$$

The output y^* of Algorithm 1 and the Lagrange multiplier assignments satisfy the KKT conditions; y^* therefore minimizes (1.1), and the proof of Theorem 1 is complete. ∎

5.3. Proof of Corollary 2.1. We begin by arguing that if $H_1 \geq H_2 \geq \cdots \geq H_L$, then slopes at the origin are ordered as

$$h_1(0) \leq h_2(0) \leq \cdots \leq h_L(0). \tag{5.24}$$

To see this, Let $\underline{h}\,(0) = h_l(0)$ for some l. By the ordering of H_m we have

$$\beta_1 > 0 = H_1(h_1(0)) \geq H_1(h_l(0)) \geq H_l(h_l(0)) = 0,$$

implying that $H_1(h_l(0)) = H_1(\underline{h}\,(0)) = 0$. This and the strictly increasing nature of H_1 prior to saturation imply that $h_1(0) = \underline{h}\,(0)$.

Next we show $h_{m+1}(0) \geq h_m(0)$. Assuming the contrary, we get $h_{m+1}(0) \in [h_1(0), h_m(0))$ and therefore $h_{m+1}(0) \in \mathbb{E}_m$. Proceeding as before,

$$\beta_m > 0 = H_m(h_m(0)) \geq H_m(h_{m+1}(0)) \geq H_{m+1}(h_{m+1}(0)) = 0,$$

implying that $H_m(h_m(0)) = H_m(h_{m+1}(0)) = 0$. Again, the strictly increasing nature of H_m prior to saturation implies that $h_{m+1}(0) = h_m(0)$, a contradiction. We therefore conclude that $h_m(0) \leq h_{m+1}(0)$, $m \in [\![1, L-1]\!]$, and (5.24) holds.

The assignments in Algorithm 1 are

$$y_m^* = \begin{cases} H_m\left(\xi_{p_m}\right), & m \in [\![1, j_N]\!], \\ 0, & m \in [\![j_N + 1, L]\!]. \end{cases}$$

Moreover, when the slopes are ordered as in (5.24), $c_m \neq \mathcal{B}$ for $i \in [\![1, j_N]\!]$; i.e., only Steps 3(a) and 3(c) can set variables $y_1, \cdots, y_{j_N}$. Since these steps set the lower valued indices first, we have $p_m \leq p_{m+1}$, $m \in [\![1, j_N - 1]\!]$, and therefore $\xi_{p_m} \geq \xi_{p_{m+1}} \geq h_1(0)$. Thus

$$y^*_m = H_m(\xi_{p_m}) \geq H_m(\xi_{p_{m+1}}) \geq H_{m+1}(\xi_{p_{m+1}}) = y^*_{m+1}, \quad m \in [\![1, j_N - 1]\!].$$

For $m \geq j_N + 1$, the optimum values are 0. This completes the proof. □

5.4. Proof of Proposition 2.2. Recall that here $K = L$. Define

$$\mathcal{L}(\alpha) := \left\{ y \in \mathbb{R}^L : y \text{ satisfies (1.2)–(1.4)} \right\}.$$

$\mathcal{L}(\alpha)$ is convex, but it may not be closed because the domains (a_m, b_m) may not be closed. From the ascending constraints (1.3) and (1.4), it is clear that if $\alpha \succeq \tilde{\alpha}$, then $\mathcal{L}(\alpha) \subseteq \mathcal{L}(\tilde{\alpha})$, and therefore $\mathcal{G}(\alpha) \geq \mathcal{G}(\tilde{\alpha})$. The first statement is therefore proved.

To show convexity, consider $\alpha, \tilde{\alpha} \in \mathbb{R}^L_+$. Fix $\lambda \in (0, 1)$. If either of $\mathcal{L}(\alpha)$ or $\mathcal{L}(\tilde{\alpha})$ is empty, there is nothing to prove. We may therefore assume both are nonempty and therefore $\mathcal{G}(\alpha)$ and $\mathcal{G}(\tilde{\alpha})$ are finite. For every $\varepsilon > 0$, there exist $y \in \mathcal{L}(\alpha)$ and $\tilde{y} \in \mathcal{L}(\tilde{\alpha})$ satisfying $G(y) < \mathcal{G}(\alpha) + \varepsilon$ and $G(\tilde{y}) < \mathcal{G}(\tilde{\alpha}) + \varepsilon$. The linearity of the constraints implies $\lambda y + (1 - \lambda)\tilde{y} \in \mathcal{L}(\lambda\alpha + (1 - \lambda)\tilde{\alpha})$. The convexity of G implies

$$\begin{aligned} \mathcal{G}(\lambda\alpha + (1 - \lambda)\tilde{\alpha}) &\leq G(\lambda y + (1 - \lambda)\tilde{y}) \\ &\leq \lambda G(y) + (1 - \lambda) G(\tilde{y}) \\ &\leq \lambda \mathcal{G}(\alpha) + (1 - \lambda)\mathcal{G}(\tilde{\alpha}) + \varepsilon. \end{aligned}$$

Since ε is arbitrary, the convexity of $\mathcal{G}$ is established. □

5.5. Proof of Corollary 2.3. Since the output y^* of Algorithm 1 satisfies (1.2), (1.3), and (2.7) with equality, it is feasible. To prove the optimality of y^*, apart from the conditions (5.11)–(5.15), which hold for the assignments under Theorem 1, we need only verify that $\mu = \xi_N \geq 0$. But this is easily verified because in iteration N, $\xi_N \geq h_{j_N}(0) \geq \underline{h}(0) \geq 0$, by hypothesis. □

REFERENCES

[1] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, NJ, 1962.

[2] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed., Athena Scientific, Belmont, MA, 2003.

[3] G. R. Bitran and A. C. Hax, *Disaggregation and resource allocation using convex knapsack problems with bounded variables*, Management Sci., 27 (1981), pp. 431–441.

[4] F. Bonomi and A. Kumar, *Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler*, IEEE Trans. Comput., 39 (1990), pp. 1232–1250.

[5] A. Charnes and W. W. Cooper, *The theory of search: Optimum distribution of search effort*, Management Sci., 5 (1958), pp. 44–50.

[6] C. Chen-Nee, D. N. C. Tse, J. M. Kahn, and R. A. Valenzuela, *Capacity scaling in MIMO wireless systems under correlated fading*, IEEE Trans. Inform. Theory, 48 (2002), pp. 637–650.

[7] G. B. Dantzig, *A control problem of Bellman*, Management Sci., 17 (1971), pp. 542–546.

[8] F. R. Farrokhi, G. J. Foschini, A. Lozano, and R. A. Valenzuela, *Link-optimal space-time processing with multiple transmit and receive antennas*, IEEE Commun. Lett., 5 (2001), pp. 85–87.

[9] M. Grant and S. Boyd, `CVX`*: Matlab Software for Disciplined Convex Programming (Web Page and Software)*, http://stanford.edu/∼boyd/cvx (2008).

[10] M. Grant and S. Boyd, *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control (a tribute to M. Vidyasagar), Lecture Notes in Control and Inform. Sci. 371, V. Blondel, S. Boyd, and H. Kimura, eds., Springer, London, 2008, pp. 95–110.
[11] K. C. Kiwiel, *Breakpoint searching algorithms for the continuous quadratic knapsack problem*, Math. Program., 112 (2008), pp. 473–491.
[12] K. C. Kiwiel, *Variable fixing algorithms for the continuous quadratic knapsack problem*, J. Optim. Theory Appl., 136 (2008), pp. 445–458.
[13] H. Luss and S. K. Gupta, *Allocation of effort resources among competing activities*, Operations Res., 23 (1975), pp. 360–366.
[14] A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*, Academic Press, New York, London, 1979.
[15] G. Morton, R. von Randow, and K. Ringwald, *A greedy algorithm for solving a class of convex programming problems and its connection with polymatroid theory*, Math. Program., 32 (1985), pp. 238–241.
[16] G. Munz, S. Pfletschinger, and J. Speidel, *An efficient waterfilling algorithm for multiple access OFDM*, in Proceedings of the 2002 IEEE Global Telecommunications Conference, Vol. 1, 2002, pp. 681–685.
[17] A. Padakandla and R. Sundaresan, *Power minimization for CDMA under colored noise*, IEEE Trans. Comm., to appear.
[18] M. Patriksson, *A survey on the continuous nonlinear resource allocation problem*, European J. Oper. Res., 185 (2008), pp. 1–46.
[19] L. Sanathanan, *On an allocation problem with multistage constraints*, Operations Res., 19 (1971), pp. 1647–1663.
[20] S. M. Stefanov, *Convex separable minimization subject to bounded variables*, Comput. Optim. Appl., 18 (2001), pp. 27–48.
[21] S. M. Stefanov, *Separable Programming: Theory and Methods*, 1st ed., Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.
[22] S. M. Stefanov, *Convex quadratic minimization subject to a linear constraint and box constraints*, AMRX Appl. Math. Res. Express, no. 1 (2004), pp. 17–42.
[23] S. M. Stefanov, *Convex separable minimization problems with a linear constraint and bounded variables*, Int. J. Math. Math. Sci., no. 9 (2005), pp. 1339–1363.
[24] S. M. Stefanov, *An efficient method for minimizing a convex separable logarithmic function subject to a convex inequality constraint or linear quality constraint*, J. Appl. Math. Decis. Sci., Art. ID 89307 (2006), pp. 1–19.
[25] S. Tavildar and P. Viswanath, *Approximately universal codes over slow-fading channels*, IEEE Trans. Inform. Theory, 52 (2006), pp. 3233–3258.
[26] İ. E. Telatar, *Capacity of multi-antenna Gaussian channels*, European Transactions on Telecommunications, 10 (1999), pp. 585–595.
[27] A. F. Veinott, Jr., *Least d-majorized network flows with inventory and statistical applications*, Management Sci., 17 (1971), pp. 547–567.
[28] P. Viswanath and V. Anantharam, *Optimal sequences for CDMA with colored noise: A Schur-saddle function property*, IEEE Trans. Inform. Theory, 48 (2002), pp. 1295–1318.
[29] L. Zacharias and R. Sundaresan, *Decentralized sequential change detection using physical layer fusion*, IEEE Trans. Wireless Commun., 7 (2008), pp. 4999–5008.
[30] P. H. Zipkin, *Simple ranking methods for allocation of one resource*, Management Sci., 26 (1980), pp. 34–43.