Submission to Mathematics of Operations Research

On Hochbaum's Proximity-Scaling Algorithm for the General Resource Allocation Problem

Satoko MORIGUCHI and Akiyoshi SHIOURA

January, 2002; revised November, 2002, June 2003

Keywords: resource allocation, discrete optimization, combinatorial optimization, concave function.

Abstract

It is pointed out that the polynomial-time scaling algorithm by Hochbaum (1994) does not work correctly for the general resource allocation problem. Hochbaum's algorithm increases a variable by one unit if the variable cannot be feasibly increased by the scaling unit. We modify the algorithm to increase such a variable by the largest possible amount and show that with this modification the algorithm works correctly. The effect is to modify the factor F in the running time of Hochbaum's algorithm for finding whether a certain solution is feasible by the factor $\widetilde{F}$ of finding the maximum feasible increment (also called the saturation capacity). Therefore, the corrected algorithm runs in $\mathrm{O}(n(\log n + \widetilde{F}) \log(B/n))$ time.

Satoko MORIGUCHI: Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Tokyo, 152-8552, Japan, `Satoko.Moriguchi@is.titech.ac.jp`.

Akiyoshi SHIOURA: Graduate School of Information Sciences, Tohoku University, Sendai, 980-8579, Japan, `shioura@dais.is.tohoku.ac.jp`.

1 Introduction

The general resource allocation problem, also called the resource allocation problem under submodular constraints (see Ibaraki and Katoh (1988) and Katoh and Ibaraki (1998)), is formulated as follows:

$$
\text{(GAP)} \quad \left| \begin{array}{ll} \text{Maximize} & f(\boldsymbol{x}) = \sum_{j=1}^{n} f_j(x_j) \\ \text{subject to} & \sum_{j=1}^{n} x_j = B, \\ & \sum_{j \in A} x_j \le r(A) \quad (A \subset N), \\ & \boldsymbol{x} \ge \boldsymbol{l}, \quad \boldsymbol{x} \in \mathbb{Z}^n, \end{array} \right.
$$

where $N = \{1, 2, \ldots, n\}$, $f_j : \mathbb{Z} \to \mathbb{R}$ is a concave function for $j = 1, 2, \ldots, n$, $B \in \mathbb{Z}_+$, $r : 2^N \to \mathbb{Z}$ is a submodular set function, and $\boldsymbol{l} \in \mathbb{Z}^n$. In Hochbaum (1994), a "proximity theorem" is presented for the general resource allocation problem by using a procedure called *greedy*(s). Based on this result, Hochbaum proposed a polynomial-time scaling algorithm called *GAP*, which was supposed to be the fastest so far for the general resource allocation problem. We, however, noticed incorrectness of the procedure *greedy*(s) and the algorithm *GAP*; we found some examples for which the algorithm *GAP* does not find an optimal solution.

The main aim of this paper is to modify Hochbaum's scaling algorithm so that it works correctly. Based on this observation, we modify the procedure *greedy*(s) so that the statement of the proximity theorem holds true. We also show that with a slight modification the algorithm *GAP* works correctly by using the corrected version of *greedy*(s).

2 Hochbaum's Algorithm and Counterexamples

In each scaling phase, Hochbaum's algorithm for (GAP) uses the following procedure. A vector $\boldsymbol{x} \in \mathbb{Z}^n$ is said to be *feasible* if it satisfies $\sum_{j=1}^{n} x_j \le B$, $\sum_{j \in A} x_j \le r(A)$ $(A \subset N)$, and $\boldsymbol{x} \ge \boldsymbol{l}$. We denote by $\boldsymbol{e}^j \in \{0, 1\}^n$ the j-th unit vector, and by $\boldsymbol{e}$ the vector $(1, 1, \ldots, 1)$. For each $j \in N$ and $x_j \in \mathbb{Z}$, we define $\Delta_j(x_j) = f_j(x_j + 1) - f_j(x_j)$.

Procedure greedy(s)
Step 0: $\boldsymbol{x} = \boldsymbol{l}$, $B = B - \boldsymbol{l} \cdot \boldsymbol{e}$, $E = \{1, 2, \ldots, n\}$.
Step 1: Find i such that $\Delta_i(x_i) = \max_{j \in E}\{\Delta_j(x_j)\}$.
Step 2: (Feasibility check), is $\boldsymbol{x} + \boldsymbol{e}^i$ infeasible?
  If yes, $E \leftarrow E - \{i\}$, and $\delta_i = s$.
  Else, is $\boldsymbol{x} + s \cdot \boldsymbol{e}^i$ infeasible?
    $(*)$ If yes, $E \leftarrow E - \{i\}$, $x_i \leftarrow x_i + 1$, $B \leftarrow B - 1$, and $\delta_i = 1$.
    Else, $x_i \leftarrow x_i + s$, and $B \leftarrow B - s$.
Step 3: If $B = 0$, or $E = \emptyset$, stop, output $\boldsymbol{x}$. Otherwise go to Step 1. □

In this procedure δ_j denotes the last increment in x_j for $j = 1, 2, \ldots, n$. Using the output $\boldsymbol{x}^{(s)}$ of Procedure *greedy*(s), the following "proximity theorem" was stated:

Statement A (Theorem 4.1 of Hochbaum (1994)). *If there is a feasible solution to* (GAP) *then there exists an optimal solution* $\boldsymbol{x}^*$ *such that* $\boldsymbol{x}^* > \boldsymbol{x}^{(s)} - \boldsymbol{\delta} \geq \boldsymbol{x}^{(s)} - s \cdot \boldsymbol{e}$ (*where the inequalities are component-wise*).

Based on this, Hochbaum (1994) proposed the following scaling algorithm for (GAP), and claimed that it runs in $\mathrm{O}(n(\log n + F)\log(B/n))$ time (more precisely, $\mathrm{O}(n(\log n + F)\max\{\log(B/n), 1\})$ time), where F is the time complexity for the feasibility check of a vector.

Algorithm GAP
Step 0: Put $s = \lceil B/2n \rceil$.
Step 1: If $s = 1$, call Procedure *greedy*(1). The output is $\boldsymbol{x}^*$. Stop. $\boldsymbol{x}^*$ is an optimal solution.
Step 2: Call *greedy*(s). Let $\boldsymbol{x}^{(s)}$ be the output. Set $\boldsymbol{l} = \boldsymbol{x}^{(s)} - s\boldsymbol{e}$. Set $s \leftarrow \lceil s/2 \rceil$. Go to Step 1. □

Statement B (Theorem 5.1 (a) of Hochbaum (1994)). *Algorithm GAP finds an optimal solution of* (GAP).

Example 2.1. Consider a simple resource allocation problem with upper bounds:

$$\left|\begin{array}{ll} \text{Maximize} & \sum_{i=1}^{n-1}(n-i)x_i \\ \text{subject to} & \sum_{i=1}^{n} x_i = 8n, \\ & 0 \leq x_i \leq 7 \ (i = 1, 2, \ldots, n-1), \ \ x_n \geq 0, \ \ \boldsymbol{x} \in \mathbb{Z}^n, \end{array}\right. \tag{2.1}$$

where n is a multiple of 4 with $n \geq 8$. Put $s = 4$ and $\boldsymbol{l} = \boldsymbol{0}$, and apply *greedy*($s$) to the problem (2.1). Then, the vector $\boldsymbol{x}$ and the set E change as follows.

Initially, we have $\boldsymbol{x} = (0, 0, \ldots, 0, 0)$ and $E = \{1, 2, \ldots, n\}$. The first iteration increments x_1 by 4 since $\Delta_1(x_1) = n - 1 \geq n - j = \Delta_j(x_j)$ for $j \in E$. Then, the second iteration increments x_1 by one and removes x_1 from the set E since $\boldsymbol{x} + \boldsymbol{e}^1$ is feasible but $\boldsymbol{x} + 4 \cdot \boldsymbol{e}^1$ is infeasible. Hence, we have $\boldsymbol{x} = (5, 0, \ldots, 0, 0)$ and $E = \{2, 3, \ldots, n\}$.

Similarly in the $(2k-1)$-st and $2k$-th iterations with $2 \leq k \leq n-1$, the variable x_k is incremented from 0 to 5, and removed from E. Hence, we have $x_1 = \cdots = x_k = 5$, $x_{k+1} = \cdots = x_n = 0$, and $E = \{k+1, k+2, \ldots, n\}$ at the end of $2k$-th iteration.

At the beginning of $(2n-1)$-st iteration, we have $\boldsymbol{x} = (5, 5, \ldots, 5, 0)$, $E = \{n\}$ and $B = 3n + 5$. Therefore, the variable x_n is incremented by 4 repeatedly until the $(11n/4 - 1)$-st iteration. Finally in the $(11n/4)$-th iteration, x_n is incremented by one since $\boldsymbol{x} + \boldsymbol{e}^n$ is feasible but $\boldsymbol{x} + 4 \cdot \boldsymbol{e}^n$ is infeasible. Then, the variable x_n is removed from the set E, B is decreased to zero, and the procedure terminates with the output $\boldsymbol{x}^{(4)} = (5, 5, \ldots, 5, 3n + 5)$ and $\boldsymbol{\delta} = (1, 1, \ldots, 1, 1)$.

However, there is no optimal solution $\boldsymbol{x}^* \in \mathbb{Z}^n$ of the problem (2.1) satisfying the inequality in Statement A:

$$\boldsymbol{x}^* \geq \boldsymbol{x}^{(4)} - 4 \cdot \boldsymbol{e} = (1, 1, \ldots, 1, 3n + 1).$$

The unique optimal solution of (2.1) is $\widehat{\boldsymbol{x}} = (7, 7, \ldots, 7, n + 7)$. Thus, Statement A fails. □

Example 2.2. We show by an example that the output of Algorithm *GAP* may be an infeasible solution. Let us apply the algorithm to the following simple resource allocation problem:

$$\left|\begin{array}{ll} \text{Maximize} & x_1 \\ \text{subject to} & \sum_{i=1}^{n} x_i = 2n + 1, \ \ \boldsymbol{x} \geq \boldsymbol{0}, \ \ \boldsymbol{x} \in \mathbb{Z}^n, \end{array}\right.$$

where n is an integer with $n \geq 2$. Initially we have $s = \lceil (2n+1)/2n \rceil = 2$ and $\boldsymbol{l} = \boldsymbol{0}$. In the first iteration, we call *greedy*(2) and obtain a vector $\boldsymbol{x}^{(2)} = (2n+1, 0, 0, \ldots, 0)$. Then, set $\boldsymbol{l} = \boldsymbol{x}^{(2)} - 2 \cdot \boldsymbol{e} = (2n-1, -2, -2, \ldots, -2)$ and $s = \lceil 2/2 \rceil = 1$. In the second iteration, we call *greedy*(1) and obtain a vector $\boldsymbol{x}^* = (4n-1, -2, -2, \ldots, -2)$, which is infeasible. $\square$

3 Correction to Hochbaum's Algorithm

In this section, we first discuss the reason for the failure of the above algorithm. Based on this observation, we modify Procedure *greedy*(s) so that Statements A and B hold true.

The following fact is useful for finding an optimal solution of (GAP).

Theorem 3.1 (Corollary 1 of Girlich et al. (1996)). *Let $\boldsymbol{x} \in \mathbb{Z}^n$ be any feasible vector, and put $\widetilde{E}(\boldsymbol{x}) = \{j \mid \boldsymbol{x} + \boldsymbol{e}^j : \text{ feasible}\}$. If $i \in \{1, 2, \ldots, n\}$ satisfies*

$$i \in \widetilde{E}(\boldsymbol{x}), \qquad \Delta_i(x_i) = \max\{\Delta_j(x_j) \mid j \in \widetilde{E}(\boldsymbol{x})\}, \tag{3.1}$$

then there exists an optimal solution $\boldsymbol{x}^*$ of (GAP) *satisfying $x_i^* > x_i$.*

Theorem 3.1 shows that it is necessary to choose a variable x_i satisfying the condition (3.1) and to increment it in each iteration to guarantee the existence of an optimal solution $\boldsymbol{x}^*$ satisfying the inequality $\boldsymbol{x}^* > \boldsymbol{x} - \boldsymbol{\delta}$. On the other hand, the variable x_i chosen in Step 2 of Procedure *greedy*(s) does not necessarily satisfy the condition (3.1) since x_i is chosen from the set E, which is a subset of $\widetilde{E}(\boldsymbol{x})$. Each variable x_j is removed from the set E even if $\boldsymbol{x} + \boldsymbol{e}^j$ is still feasible, once the vector $\boldsymbol{x} + s \cdot \boldsymbol{e}^j$ becomes infeasible. This is the main reason why Hochbaum's algorithm fails for certain examples.

Example 2.1 (continued). In each iteration, we have $\widetilde{E}(\boldsymbol{x}) = \{1, 2, \ldots, n\}$ and $\Delta_1(x_1) = n - 1 \geq n - j = \Delta_j(x_j)$ for any j. The variable x_1, however, is never incremented after the third iteration, and the procedure increments other variables which do not satisfy the condition (3.1). In particular, the variable x_n is incremented by one in the last iteration, but we cannot assure the existence of an optimal solution $\boldsymbol{x}^*$ with $x_n^* > 3n + 4$ since x_n does not satisfy (3.1). $\square$

Based on this observation, we replace the line ($*$) in Step 2 of *greedy*(s) with the following:

If yes, $E \leftarrow E - \{i\}$, $\alpha' = \widehat{c}(\boldsymbol{x}, i)$, $x_i \leftarrow x_i + \alpha'$, $B \leftarrow B - \alpha'$, and $\delta_i = \alpha'$,

where the *saturation capacity* $\widehat{c}(\boldsymbol{x}, i)$ is defined by $\widehat{c}(\boldsymbol{x}, i) = \max\{\beta \in \mathbb{Z}_+ \mid \boldsymbol{x} + \beta \boldsymbol{e}^i : \text{feasible}\}$, which is also given as follows (see Fujishige (1991)):

$$\widehat{c}(\boldsymbol{x}, i) = \min\left[B - \sum_{j=1}^{n} x_j,\ \min\left\{r(A) - \sum_{j \in A} x_j \;\middle|\; i \in A \subset \{1, 2, \ldots, n\}\right\}\right]. \tag{3.2}$$

We denote by *greedy*$'$(s) the modified version of *greedy*(s). From Theorem 3.1 follows that in each iteration of *greedy*$'$(s) there exists an optimal solution $\boldsymbol{x}^*$ satisfying $\boldsymbol{x}^* > \boldsymbol{x} - \boldsymbol{\delta}$. This shows that Statement A holds true for *greedy*$'$(s).

Theorem 3.2. *Let $\boldsymbol{x}^{(s)}$ be the output of greedy$'(s)$. If there is a feasible solution to* (GAP) *then there exists an optimal solution $\boldsymbol{x}^*$ such that $\boldsymbol{x}^* > \boldsymbol{x}^{(s)} - \boldsymbol{\delta} \geq \boldsymbol{x}^{(s)} - s \cdot \boldsymbol{e}$.*

Even if *greedy*(s) is replaced by *greedy*$'(s)$, Algorithm GAP may output an infeasible solution; indeed, the behavior of Algorithm GAP for Example 2.2 does not change, and GAP outputs an infeasible solution. This is because the lower bound l_j may decrease when it is updated in Step 2 of GAP and therefore the vector $\boldsymbol{x}$ in *greedy*$'(s)$ (and in *greedy*(s)) can be an infeasible solution, as shown in Example 2.2. To obtain an optimal solution of (GAP) correctly, we modify Algorithm GAP by replacing the update of $\boldsymbol{l}$ in Step 2 with the following:

Put $l_j := \max\{x_j^{(s)} - s + 1, l_j\}$ $(j = 1, 2, \ldots, n)$.

Algorithm GAP with this replacement is denoted by GAP'. The correctness of GAP' follows immediately from Theorem 3.2. The running time of GAP' can be analyzed in the same way as in Hochbaum (1994). We denote by $\widetilde{F}$ the time complexity required for computing the saturation capacity $\widehat{c}(\boldsymbol{x}, i)$.

Theorem 3.3. *Algorithm GAP' finds an optimal solution of* (GAP) *in* $\mathrm{O}(n(\log n + \widetilde{F}) \log(B/n))$ *(more precisely,* $\mathrm{O}(n(\log n + \widetilde{F}) \max\{\log(B/n), 1\})$*).*

Finally, we compare the time complexity of GAP' with that of Hochbaum's algorithm GAP; in particular, we compare $\widetilde{F}$ with F. Saturation capacity computation corresponds to finding the most violated constraint in the sense of (3.2), whereas feasibility check needs to find a violated constraint. Hence, we have $F \leq \widetilde{F}$ for the general resource allocation problem, and it is still an open question whether $F = \widetilde{F}$ or not. For special cases such as the nested and the tree constrained problems dealt with in Hochbaum (1994), on the other hand, we have $F = \widetilde{F}$, i.e., the time complexity of our corrected algorithm is the same as the one by Hochbaum (1994) for such special cases. It is left for future research to discuss whether our corrected algorithm runs as fast as Hochbaum's.

Acknowledgement

The authors thank Kazuo Murota, Akihisa Tamura, and anonymous referees for their valuable comments. This work is supported by Grant-in-Aid of the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

Fujishige, S. 1991. *Submodular Functions and Optimization*, North-Holland, Amsterdam.

Girlich, E., M. Kovalev, A. Zaporozhets. 1996. A polynomial algorithm for resource allocation problems with polymatroid constraints. *Optimization* **37** 73–86.

Hochbaum, D. S. 1994. Lower and upper bounds for the allocation problem and other nonlinear optimization problems. *Math. Oper. Res.* **19** 390–409.

Ibaraki, T., N. Katoh. 1988. *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, Boston, MA.

Katoh, N., T. Ibaraki. 1998. Resource allocation problems. D. -Z. Du and P. M. Pardalos, eds., *Handbook of Combinatorial Optimization II*, Kluwer Academic Publishers, Boston, MA, 159–260.