# A Distributed Algorithm for the Assignment Problem*

Dimitri P. Bertsekas

March 1979†

**Abstract**

This paper describes a new algorithm for solving the classical assignment problem. The algorithm is of a primal-dual nature and in some ways resembles the Hungarian and subgradient methods, but is substantially different in other respects. Its main feature is that it is well suited for distributed operation whereby each node participates in the computation on the basis of limited local information about the topology of the network and the data of the problem. The algorithmic process resembles an auction where economic agents compete for resources by making successively higher bids. The algorithm terminates in a finite number of iterations after resource prices reach levels where no further bidding is profitable.

*Laboratory for Information and Decision Sciences Working Paper, Massachusetts Institute of Technology, Cambridge, Mass. 02139

†This document is retyped for greater readability from a scanned copy of the original 1979 paper.

# 1 Introduction

The assignment (or weighted matching) problem was among the first linear programming problems to be studied extensively. It arises often in practice and it is usually solved by either the simplex method or Kuhn's Hungarian method (see e.g. [**?**], [**?**], [**?**]). Recently there has been considerable interest in developing distributed algorithms for optimization and other problems. There seems to be no precise definition of what is meant by a distributed algorithm, but the term usually refers to a situation where there are several computation centers (which could be, for example, microprocessors units) connected via communication links, and each center is responsible for executing part of the algorithm while coordination between centers is maintained by information exchanges via the communication links. One possible advantage of distributed operation is that it results in reduction of time needed to solve the problem when a significant amount of computation can be carried out in parallel by several processors. Another advantage of some distributed algorithms for network problems is that they may be more suitable for real-time operation under conditions where the network topology may be subject to change such as when existing nodes fail or new nodes become operational. Furthermore, when problem data is itself distributed among network nodes, the need for a central data collection mechanism may be eliminated.

Several distributed algorithms have been proposed for particular types of linear programming problems - for example the shortest path problem. The standard methods for the assignment problem, however, apparently cannot be easily modified so that they can be operated in a distributed manner. It is possible to use a subgradient method (see e.g. [**?**], [**?**]) for solving the dual of the assignment problem, and such an algorithm can be operated in a distributed manner. However, finite termination at an optimal dual solution cannot be guaranteed for a subgradient method and even if an optimal dual solution were made available one is still faced with the problem of finding an optimal primal solution in a distributed way.

The algorithm proposed in this paper is the first, to our knowledge, for the assignment problem that can be meaningfully viewed as being distributed and is guaranteed to terminate finitely at an optimal assignment. Since linear transportation problems with integer supplies and demands can be reduced to the assignment problem, the algorithm can be adapted to handle such problems as well. The algorithm may also be of interest as a.model of price formation in an auction, but this economic interpretation has not been pursued to any significant extent in this paper. The ideas underlying the algorithm bear some similarity with those of $\epsilon$-subgradient methods of the type discussed in [**?**] but the feature of finite termination is generically absent in the latter methods. Similarities with Kuhn's Hungarian method [**?**] will also be noticed by the reader, but from a mathematical programming point of view there is one significant difference. In the Hungarian method the dual objective function value is decreased at each iteration, but in the method of this paper this value may be increased in some iterations - a feature found in subgradient methods.

## 2 A Distributed Algorithm for the Assignment Problem

Consider a bipartite graph consisting of two finite sets of nodes $\mathcal{U}$ and $\mathcal{V}$, and a nonempty set of directed links $\mathcal{L}$ with elements denoted $(i, j)$ where $i \in \mathcal{U}$ and $j \in \mathcal{V}$. We refer to elements of $\mathcal{U}$ and $\mathcal{V}$ as *sources* and *sinks* respectively. Each link $(i, j)$ has a scalar weight $a_{ij}$ associated with it. By an *assignment* we mean a subset $\mathcal{A}$ of links such that for each source $i$ (sink $j$) there is at most one link in $\mathcal{A}$ with initial node $i$ (terminal node $j$). We wish to find an assignment that maximize $\sum_{(i,j)\in\mathcal{A}} a_{ij}$ over all assignments $\mathcal{A}$.

The problem can be embedded into the linear program [**?**]

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in\mathcal{L}} a_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{\{j\,|\,(i,j)\in\mathcal{L}\}} x_{ij} \le 1, \quad \forall i \in \mathcal{U}, \\ & \sum_{\{i\,|\,(i,j)\in\mathcal{L}\}} x_{ij} \le 1, \quad \forall j \in \mathcal{V}, \\ & x_{ij} \ge 0, \quad \forall (i,j) \in \mathcal{L}. \end{aligned} \tag{1}$$

in the sense that an optimal solution of problem (1) which is an extreme point of its feasible set corresponds to an optimal assignment. The problem dual to (1) is the linear program in the vectors $m$ and $p$ given by

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in\mathcal{U}} m_i + \sum_{j\in\mathcal{V}} p_j \\ \text{subject to} \quad & m_i + p_j \ge a_{ij}, \quad \forall (i,j) \in \mathcal{L}, \\ & m_i \ge 0, \quad p_j \ge 0, \quad \forall i \in \mathcal{U},\, j \in \mathcal{V}. \end{aligned} \tag{2}$$

As an aid in understanding the following algorithm it is worth noting that if we view $a_{ij}$ and $p_j$ as the *value* and *price* respectively of including link $(i, j)$ in an assignment, then $(a_{ij} - p_j)$ may be view as the *profit margin* of source $i$ for getting assigned to sink $j$. From the complementary slackness conditions we have that if link $(i, j)$ is part of an optimal assignment then for any optimal solution $m^*$, $p^*$ of the dual problem we have

$$m_i^* = a_{ij} - p_j^* = \max\{a_{ik} - p_k^* \,|\, \text{all } k \text{ with } (i,k) \in \mathcal{L}\}, \tag{3}$$

i.e., at an optimum *source $i$ is assigned to the sink $j$ offering maximum profit margin* relative to the price vector $p^*$.

Given a vector $p$ of nonnegative prices we say that source $i$ is *operational* if

$$\max\{a_{ij} - p_j \,|\, \text{all } j \text{ with } (i,j) \in \mathcal{L}\} > 0,$$

i.e., source $i$ is operational if it has a positive maximum profit margin. For an operational source $i$ we say that $k$ is a *preferred sink for $i$* if it offers maximum

profit margin for $i$, i.e., if

$$a_{ik} - p_k = \max\{a_{ij} - p_j \,|\, \text{all } j \text{ with } (i,j) \in \mathcal{L}\}. \tag{4}$$

If $k$ is a preferred sink for an operational source $i$ we define the *price adjustment margin* of $i$ by

$$\pi_k = \begin{cases} a_{ik} - p_k, & \text{if there is no sink } j \neq k \text{ with } a_{ij} - p_j > 0, \\ a_{ik} - p_k - \max\{a_{ij} - p_j \,|\, \text{all } j \neq k \text{ with } (i,j) \in \mathcal{L}\}, & \text{otherwise.} \end{cases} \tag{5}$$

The scalar $\pi_k$ represents the amount that one can add to the price $p_k$ and still have $k$ be a preferred sink for source $i$. Finally, given an assignment $\mathcal{A}$ we say that source $i$ is *assigned* to sink $j$ if $(i,j) \in \mathcal{A}$, and we say that source $i$ or sink $j$ is *unassigned* if $(i,j) \notin \mathcal{A}$ for all $(i,j) \in \mathcal{L}$.

We now describe our algorithm. At the typical step a vector $p$ of nonnegative sink prices $p_j$, $j \in \mathcal{V}$ is available, together with an assignment $\mathcal{A}$. A scalar $\epsilon > 0$, fixed throughout the algorithm, is also given. If all sources are either assigned or not operational the algorithm terminates. Otherwise an arbitrary unassigned operational source $i$ is selected together with a preferred sink $k$ for $i$. To perform the step of the algorithm we do the following:

(a) Increase $p_k$ to a level $\overline{p}_k$ satisfying

$$p_k + \epsilon \leq \overline{p}_k \leq p_k + \pi_k + \epsilon \tag{6}$$

where $\pi_k$ is given by (5), and leave all other prices unchanged.

(b) Add link $(i,k)$ to the assignment $\mathcal{A}$, and, if $k$ was assigned to some $\ell$, remove link $(\ell,k)$ from $\mathcal{A}$.

The step of the algorithm is repeated with the new price vector and assignment until termination. The initial prices are taken to be all zero, and the initial assignment is taken to be the empty assignment.

A possible interpretation of a step of the algorithm is that an unassigned operational source $i$ bids for its preferred sink $k$ by adding an amount between $\epsilon$ and $\pi_k + \epsilon$ to the price of $k$, and gets assigned to $k$. Notice that when $p_k$ is changed to $\overline{p}_k$ the profit margin of $(i,k)$ is reduced by $(\overline{p}_k - p_k)$ and by using (5) and (6) we have

$$a_{ik} - \overline{p}_k \geq a_{ik} - p_k - \pi_k - \epsilon \geq a_{ij} - p_j - \epsilon, \quad \forall j \neq k \text{ with } (i,j) \in \mathcal{L}. \tag{7}$$

Thus the profit margin $a_{ik} - \overline{p}_k$ is within $\epsilon$ of the maximum profit margin after $p_k$ is increased to $\overline{p}_k$. The scalar $\epsilon$ can be viewed as the minimum bidding increment. Its role will become apparent in the convergence analysis that follows.

Note that steps of the algorithm involving different unassigned operational sources can actually be carried out independently and simultaneously. In distributed algorithmic operation a source need only know whether it is assigned or not together with the current prices of the neighboring sinks. If it is unassigned and operational it bids for a preferred sink and communicates the new

price. The sink in turn broadcasts the new price and assignment information to its neighboring sources. If two sources bid for the same sink simultaneously, the sink arbitrarily decides which one to accept and accordingly informs the neighboring sources.

We now turn to convergence analysis of the algorithm. We first observe that the *algorithm will terminate in a finite number of steps*. This follows from the fact that at each step one price is increased by at least $\epsilon > 0$ and if all prices increase to sufficiently high levels then there will be no operational sources left. Clearly the algorithm will terminate faster if $\overline{p}_k$ is set at the maximum possible value $p_k + \pi_k + \epsilon$ [cf. (6)] and this seems to be the best way to operate the algorithm. The number of steps needed for termination may also depend on the value $\epsilon$. Depending on the problem at hand the number of steps needed can remain constant or increase significantly as $\epsilon$ is reduced. On the other hand a value of $\epsilon$ below a certain threshold value is necessary in order to obtain an optimal assignment at termination as the following proposition shows.

**Proposition 1.** Let $v_1, v_2, \ldots, v_m$ be all the possible distinct values of assignments and assume $v_1 > v_2 > \cdots > v_m$. Let also

$$N = \min\{\text{number of sources, number of sinks}\}.$$

The value $v^*$ obtained upon termination of the algorithm satisfies

$$v_1 - N\epsilon \leq v^* \leq v_1$$

and hence, if $\epsilon$ satisfies

$$0 < \epsilon < \frac{v_1 - v_2}{N}, \tag{8}$$

the algorithm terminates at an optimal assignment.

*Proof.* Let $\mathcal{A}^*$ and $p^*$ be the final assignment and price vector obtained by the algorithm. Define the vectors $x^*$ and $m^*$ by

$$x^*_{ij} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{A}^* \\ 0, & \text{if } (i,j) \notin \mathcal{A}^* \end{cases} \tag{9}$$

$$m^*_i = \max\big\{\max\{0, a_{ij} - p^*_j\} \,|\, \text{all } j \text{ with } (i,j) \in \mathcal{L}\big\}, \quad \forall i \in \mathcal{U}. \tag{10}$$

Clearly we have

$$m^*_i \geq 0,\ m^*_i + p^*_j \geq a_{ij}, \quad \forall i \in \mathcal{U},\ (i,j) \in \mathcal{L}, \tag{11}$$

$$p^*_j = 0,\ \text{if } 0 = \sum_{\{i \,|\, (i,j) \in \mathcal{L}\}} x^*_{ij} < 1, \quad p^*_j \geq 0,\ \forall j \in \mathcal{V}. \tag{12}$$

If $x^*_{ij} = 0$ for all $j$, then $i$ is not operational relative to $p^*$ for otherwise the algorithm would not terminate at $\mathcal{A}^*$. This implies $a_{ij} - p^*_j \leq 0$, for all $j$ with

$(i,j) \in \mathcal{L}$. Hence from (10)

$$m_i^* = 0, \quad \text{if } 0 = \sum_{\{j \mid (i,j)\in\mathcal{L}\}} x_{ij}^* < 1. \tag{13}$$

Also from (7), (8) and (10) we have

$$m_i^* + p_j^* \le a_{ij} + \epsilon \quad \text{if } (i,j) \in \mathcal{A}^*. \tag{14}$$

Define

$$a_{ij}^* = \begin{cases} m_i^* + p_j^*, & \text{if } (i,j) \in \mathcal{A}^*, \\ a_{ij}, & \text{if } (i,j) \notin \mathcal{A}^*. \end{cases} \tag{15}$$

From complementary slackness we have that (9), (11), (12), (13), (15) imply that $x^*$ solves the linear program

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in\mathcal{L}} a_{ij}^* x_{ij} \\ \text{subject to} \quad & \sum_{\{j \mid (i,j)\in\mathcal{L}\}} x_{ij} \le 1, \qquad \sum_{\{i \mid (i,j)\in\mathcal{L}\}} x_{ij} \le 1, \quad x_{ij} \ge 0 \end{aligned}$$

and $(m^*, p^*)$ solve its dual. Let $\tilde{x}$ be an optimal solution of problem (1). We have

$$\begin{aligned} \sum_{(i,j)\in\mathcal{L}} a_{ij}^* \tilde{x}_{ij} &\le \sum_{(i,j)\in\mathcal{L}} a_{ij}^* x_{ij}^* \\ \sum_{(i,j)\in\mathcal{L}} a_{ij} x_{ij}^* &\le \sum_{(i,j)\in\mathcal{L}} a_{ij} \tilde{x}_{ij} \end{aligned}$$

and by using (11), (14), (15) we obtain

$$\begin{aligned} v_1 = \sum_{(i,j)\in\mathcal{L}} a_{ij}\tilde{x}_{ij} &\le \sum_{(i,j)\in\mathcal{L}} a_{ij}^*\tilde{x}_{ij} \le \sum_{(i,j)\in\mathcal{L}} a_{ij}^* x_{ij}^* \\ &\le \sum_{(i,j)\in\mathcal{L}} a_{ij} x_{ij}^* + \epsilon \sum_{(i,j)\in\mathcal{L}} x_{ij}^* \le \sum_{(i,j)\in\mathcal{L}} a_{ij} x_{ij}^* + N\epsilon \\ &= v^* + N\epsilon \end{aligned} \tag{16}$$

which proves the proposition. QED

Notice that *if the weighting scalars $a_{ij}$ are all integers, then $v_1 - v_2 > 1$ so by choosing $\epsilon < \frac{1}{N}$ we are guaranteed termination at an optimal assignment.*

We have specified that the initial price vector is $p = 0$ and the initial assignment is empty. However, any price assignment pair $(p, \mathcal{A})$ that satisfies the following two rules:

a) Every unassigned sink $j$ has $p_j = 0$.

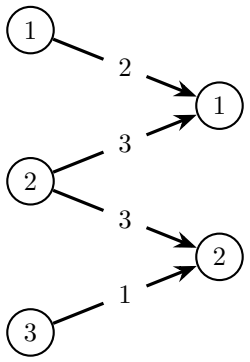


Figure 1

b) If $(i, j) \in \mathcal{A}$ then $j$ is a preferred sink for $i$ relative to $p$,

can serve as initial choice. The proof of Proposition 1 carries through for any initial choice of $(p, \mathcal{A})$ satisfying condition a) and b) above.

We note that given any integer $N > 0$ it is possible to construct an assignment problem where the weights $a_{ij}$ are integers and the algorithm with $\epsilon = \frac{1}{N}$ fails to find an optimal assignment. This shows that *the bound* $\epsilon < \frac{1}{N}$ *provided by Proposition 1 is sharp.* We give an example for $N = 2$ which is easily generalized for any $N > 2$. A similar example can be constructed for $N = 1$.

**Example 1.** Consider the assignment problem represented by the network of Figure 1 with the weights $a_{ij}$ shown along the corresponding links.

The optimal assignment is $(1, 1)$, $(2, 2)$ with value 5. The second best assignment is $(2, 1)$, $(3, 2)$ with value 4. Thus $v_1 - v_2 = 1$ in Proposition 1. Take $\epsilon = \frac{1}{N} = \frac{1}{2}$. A possible sequence of prices and assignments generated by the algorithm is

| Step | Link Entering the Assignment | New Price |
|---|---|---|
| 1 | $(3, 2)$ | $p_2 = 1 + \epsilon$ |
| 2 | $(2, 1)$ | $p_1 = 1 + 2\epsilon$ |

and the algorithm terminates at the second best assignment since when $p_1 = 1 + 2\epsilon$ and $\epsilon = \frac{1}{2}$ the unassigned source 1 is not operational. By contrast if we had $\epsilon < \frac{1}{2}$ the sequence generated would be

| Step | Link Entering the Assignment | New Price |
|---|---|---|
| 1 | $(3, 2)$ | $p_2 = 1 + \epsilon$ |
| 2 | $(2, 1)$ | $p_1 = 1 + 2\epsilon$ |
| 3 | $(1, 1)$ | $p_1 = 1 + 2\epsilon$ |
| 4 | $(2, 2)$ | $p_2 = 2 + 2\epsilon$ |

Thus the optimal assignment is obtained if $\epsilon < \frac{1}{2}$.

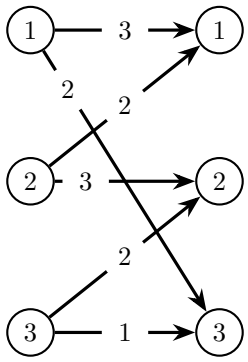


Figure 2

In the preceding example the number of sinks is smaller than the number of sources. If

$$\text{number of sources} \leq \text{number of sinks} \tag{17}$$

then it is possible to improve the bound on $\epsilon$ by the proposition and show that if $N \geq 2$ it is sufficient to have

$$0 < \epsilon < \frac{v_1 - v_2}{N - 1} \tag{18}$$

in order to guarantee that the algorithm terminates at an optimal assignment. If $N = 1$ then obviously the same is true for any $\epsilon > 0$. This can be shown by a slight modification of the proof of Proposition 1. (If $\mathcal{A}^*$ contains less than $N$ links we can modify (16) to show the result. Otherwise if $(\bar{i}, \bar{j})$ is the last link to enter the terminal assignment $\mathcal{A}^*$ we can modify $p^*_{\bar{j}}$ so that $a_{\bar{i}\bar{j}} = a^*_{\bar{i}\bar{j}}$ and use (16) again.)

We can show that when (17) holds and the weights $a_{ij}$ are integers the bound $\epsilon < \frac{1}{N-1}$ provided by (18) is sharp by means of the following example.

**Example 2.** Consider the assignment problem represented by the network of Figure 2.

The optimal assignment is $(1,1)$, $(2,2)$, $(3,3)$ with value 7. The second best assignment is $(1,3)$, $(2,1)$, $(3,2)$ with value 6. Thus $v_1 - v_2 = 1$ in Proposition 1. Take $\epsilon = \frac{1}{N-1} = \frac{1}{2}$. A possible sequence of prices and assignments generated by the algorithm is

| Step | Link Entering the Assignment | New Price |
|---|---|---|
| 1 | $(3,2)$ | $p_2 = 1 + \epsilon$ |
| 2 | $(2,1)$ | $p_1 = 2\epsilon$ |
| 3 | $(1,3)$ | $p_3 = \epsilon$ |

and the algorithm terminates at the second best assignment. By contrast if we had $\epsilon < \frac{1}{2}$ the sequence generated would be

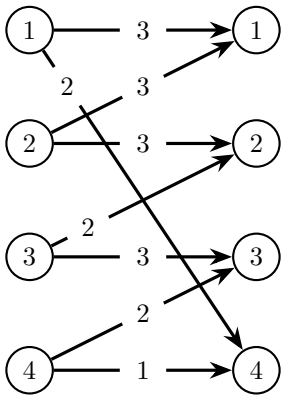


Figure 3

| Step | Link Entering the Assignment | New Price |
|---|---|---|
| 1 | $(3,2)$ | $p_2 = 1+\epsilon$ |
| 2 | $(2,1)$ | $p_1 = 2\epsilon$ |
| 3 | $(1,1)$ | $p_1 = 1+\epsilon$ |
| 4 | $(2,2)$ | $p_2 = 2+2\epsilon$ |
| 5 | $(3,3)$ | $p_3 = 1+\epsilon$ |

Thus the optimal assignment is obtained for $\epsilon < \frac{1}{2}$.

The reader can verify that the assignment problem of Figure 3 gives a related example for $N = 4$ and a similar example can be constructed for every $N > 4$, as well as for $N = 2$.

An important question relates to the number of steps $n$ necessary for the algorithm to terminate. Let us assume that the weights $a_{ij}$ are integers and let[1]

$$M = \max\{a_{ij} \,|\, (i,j) \in \mathcal{L}\}.$$

Assume further that $\epsilon = \frac{1}{k}$ for some integer $k$. Then it is clear that at most $kM$ steps will be needed before the price of any one sink reaches or exceeds the level $M$. Since the number of nonzero prices cannot exceed $N$, we obtain the estimates

$$n \leq kMN. \tag{19}$$

If $\epsilon = \frac{1}{N+1}$ we are guaranteed termination at an optimal assignment and we obtain

$$n \leq M(N^2 + N). \tag{20}$$

If $N =$ number of sources we can take $\epsilon = \frac{1}{N}$ in which case

$$n \leq MN^2. \tag{21}$$

[1]More generally, $M$ is an upper bound on reduced costs.

These upper bounds can be actually improved at the expense of a more refined analysis But this does not seem to be worth the effort. In any case these bounds are usually quite conservative. For instance in Example 2 the number of steps $n$ was 5 while the upper bound in (21) is 27. However, given any $\delta > 0$ it is possible to construct an example of a network where $v_1 - v_2 = 1$ and

$$1 - \delta \leq \frac{n}{M(N^2 + N)} \leq 1 \tag{22}$$

which implies that (20) is quite reliable as a worst case estimate. The reader can verify this by considering a network with $(N + 1)$ sources and $N$ sinks. Every source-sink pair is connected by a link of weight $M$ except for a single source-link pair that is connected by a link of weight $(M-1)$. Then $v_1 - v_2 = 1$ and if we take $\epsilon = \frac{1}{N+1}$ it is possible to show that for $M$ and $N$ sufficiently large (22) hold.

Now each step of the algorithm requires a number of additions and comparisons which is less or equal to three times the number of sinks. Thus in view of (20) the total number of arithmetic operations is bounded by

$$3M(N^2 + N)(\text{number of sinks}).$$

This is comparable to the number of arithmetic operations needed by the Hungarian method ([**?**], p. 205), but it should be mentioned that the computational burden in our method can be quite severe if $M$ is a large integer. For relative small value of $M$, however, our method, in addition to being distributed, seems to combine the advantage of simplicity with reasonably efficient computational operation.

# 3 The Algorithm Applied to the Maximal Complete Assignment Problem

If for an assignment $\mathcal{A}$ every source and sink is assigned, we say that $\mathcal{A}$ is *complete*. A variation of the assignment problem is to find an assignment that maximize $\sum_{(i,j)\in\mathcal{A}} a_{ij}$ over all complete assignments $\mathcal{A}$. The corresponding linear program is

$$\begin{aligned} \text{maximize} \quad & \sum_{(i,j)\in\mathcal{L}} a_{ij}x_{ij} \\ \text{subject to} \quad & \sum_{\{j\,|\,(i,j)\in\mathcal{L}\}} x_{ij} = 1, \quad \forall i \in \mathcal{U}, \\ & \sum_{\{i\,|\,(i,j)\in\mathcal{L}\}} x_{ij} = 1, \quad \forall j \in \mathcal{V}, \\ & x_{ij} \geq 0, \quad \forall (i,j) \in \mathcal{L}. \end{aligned} \tag{23}$$

Its dual is given by

$$\begin{aligned} \text{minimize} \quad & \sum_{i\in\mathcal{U}} m_i + \sum_{j\in\mathcal{V}} p_j \\ \text{subject to} \quad & m_i + p_j \geq a_{ij}, \quad \forall (i,j) \in \mathcal{L}. \end{aligned} \tag{24}$$

We *assume in what follows that there exists at least one complete assignment* and, among other things, this implies that

$$\text{Number of Sources} = \text{Number of Sinks} = N.$$

*We also assume that for each source $i$ there are at least two sinks $j$ such that $(i,j) \in \mathcal{L}$.* This is no real loss of generality but simplifies somewhat the following algorithm.

Held, Wolfe and Crowder [**?**] have considered solution of the dual problem (24) by means of subgradient method. They report favorable results and observe that if there exists a unique optimal assignment then a subgradient method will typically yield an optimal dual solution in a finite number of steps. Our algorithm bears some similarity with their method with the main differences lying in the stepsize procedure, and in the fact that we change only one price at each step rather than the entire price vector.

We now show how to modify the algorithm of the preceding section to make it applicable to problems (23) and (24).

Given a price vector $p$ with elements $p_j$, $j \in \mathcal{V}$, we define for any source $i$ the *preferred sink for $i$* to be the sink $k$ for which

$$a_{ik} - p_k = \max\{a_{ij} - p_j \,|\, \text{all } j \text{ with } (i,j) \in \mathcal{L}\}.$$

We define the *price adjustment margin of $i$* by

$$\pi_k = a_{ik} - p_k - \max\{a_{ij} - p_j \,|\, \text{all } j \neq k \text{ with } (i,j) \in \mathcal{L}\}.$$

The typical step of the algorithm is as follows. Given a vector $p$ of sink prices $p_j$, $j \in \mathcal{V}$, we select an arbitrary unassigned source $i$ together with a preferred sink $k$ for $i$. We increase $p_k$ to a level $\overline{p}_k$ satisfying

$$p_k + \epsilon \leq \overline{p}_k \leq p_k + \pi_k + \epsilon,$$

we add link $(i,k)$ to $\mathcal{A}$ and, if $k$ was assigned to some $\ell$, we remove link $(\ell,k)$ from $\mathcal{A}$. The step of the algorithm is repeated with the new price and assignment. If every source is assigned the algorithm terminates. The initial price vector $p$ is arbitrary. The initial assignment is either empty, or any assignment $\mathcal{A}$ such that if $(i,j) \in \mathcal{A}$ then $j$ is a preferred sink for $i$ with respect to the initial price vector $p$.

We have the following result.

**Proposition 2.** a) The algorithm terminates in a finite number of steps.

b) Let $v_1, v_2, \ldots, v_m$ be all the possible distinct values of complete assignments and assume $v_1 > v_2 > \cdots > v_m$. The value $v^*$ of the assignment obtained upon termination satisfies

$$v_1 - (N-1)\epsilon \leq v^* \leq v_1$$

and hence, if $\epsilon$ satisfies

$$0 < \epsilon < \frac{v_1 - v_2}{N-1},$$

the algorithm terminates at an optimal complete assignment.

*Proof.* a) Partition the set $\mathcal{V}$ into three disjoint sets $\mathcal{V}_o$, $\mathcal{V}_s$, $\mathcal{V}_\infty$ defined as follows:

$\mathcal{V}_o = \{j \,|\, j \text{ is unassigned in all steps of the algorithm}\}$,
$\mathcal{V}_s = \{j \,|\, j \text{ is assigned to the same source after a finite number of steps}\}$,
$\mathcal{V}_\infty = \{j \,|\, j \notin \mathcal{V}_o,\ j \notin \mathcal{V}_s\}$.

If the algorithm does not terminate in a finite number of steps the sets $\mathcal{V}_o$ and $\mathcal{V}_s$ are nonempty. The sequence of prices of any sink in $\mathcal{V}_\infty$ tends to $\infty$, while the prices of any sink in $\mathcal{V}_o$ or $\mathcal{V}_s$ are constant after some step. Let $\mathcal{U}_s$ be the subset of sources that remain assigned to sinks in $\mathcal{V}_s$ after some step, and let $\mathcal{U}_\infty$ be the set of sources not in $\mathcal{U}_s$. The set $\mathcal{U}_\infty$ has a larger cardinality than $\mathcal{V}_\infty$ since $\mathcal{U}_s$ and $\mathcal{V}_s$ have the same cardinality and $\mathcal{V}_o$ is nonempty. Furthermore, there can be no link $(i,j) \in \mathcal{L}$ such that $i \in \mathcal{V}_\infty$ and $j \in \mathcal{V}_s \cup \mathcal{V}_o$, for if such a link existed then, after some step, $j$ would offer higher profit margin for $i$ than any of the sinks in $\mathcal{V}_\infty$. This would compel $i$ to be assigned infinitely often to some sink in $\mathcal{V}_o \cup \mathcal{V}_s$ and would violate the assumption that the price $p_j$, $j \in \mathcal{V}_o \cup \mathcal{V}_s$ stabilize after a finite number of steps. Thus we have that

$$\{j \,|\, (i,j) \in \mathcal{L},\ i \in \mathcal{U}_\infty\} = \mathcal{V}_\infty.$$

Since $\mathcal{U}_\infty$ has larger cardinality than $\mathcal{V}_\infty$ we arrive at the conclusion that there cannot exist a complete assignment - a contradiction.

b) The proof of this part is very similar to the proof of Proposition 1. Compare also with the discussion following (18).

QED

Example 2 shows that the bound $\epsilon < \frac{1}{N-1}$ provided by Proposition 2 for $a_{ij}$: integer cannot be improved. The fact that an arbitrary initial price vector

may be used in the algorithm of this section suggests the possibility of obtaining a good initial choice of $p$ by first running the algorithm with a large value of $\epsilon$. Regarding computational complexity, it is easy to show similarly as in the previous section that, if $a_{ij}$: integer and $\epsilon = \frac{1}{N}$, the total number of arithmetic operations required for termination is bounded by $bN^3$ where $b$ is a constant depending on the data of the problem. An easily computable upper bound for $b$ is $3(\max_{(i,j)\in\mathcal{L}} a_{ij} - \min_{(i,j)\in\mathcal{L}} a_{ij})$.[2]

# References


[BW75] Michel Louis Balinski and Philip Wolfe. *Nondifferentiable optimization, Mathematical Programming Study 3*. North-Holland, 1975.

[HWC74] Michael Held, Philip Wolfe, and Harlan P Crowder. Validation of subgradient optimization. *Mathematical programming*, 6(1):62–88, 1974.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[Law01] Eugene L Lawler. *Combinatorial optimization: networks and matroids*. Courier Corporation, 2001.

[Nur78] EA Nurminski. Nondifferentiable optimization with epsilon subgradient methods. WP-78-055, 1978.

[Pol69] Boris Teodorovich Polyak. Minimization of unsmooth functionals. *USSR Computational Mathematics and Mathematical Physics*, 9(3):14–29, 1969.

[Sim66] Michel Simonnard. *Linear programming*. Prentice-Hall, 1966.


[2] More generally, the range of reduced costs.