NEW SCALING ALGORITHMS FOR THE ASSIGNMENT AND MINIMUM CYCLE MEAN PROBLEMS

James B. Orlin
and
Ravindra K. Ahuja*

Sloan W.P. No. 2019-88 May 1988

* On leave from Indian Institute of Technology, Kanpur, INDIA

New Scaling Algorithms for the Assignment and Minimum Cycle Mean Problems

James B. Orlin and Ravindra K. Ahuja*
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA. 02139, USA

Abstract

In this paper, we suggest new scaling algorithms for the assignment and minimum cycle mean problems. Our assignment algorithm is based on applying scaling to a hybrid version of the recent *auction* algorithm of Bertsekas and the sequential shortest path algorithm. The algorithm proceeds by relaxing the optimality conditions and the amount of relaxation is successively reduced to zero. On a network with $2n$ nodes, m arcs, and integer arc costs bounded by C, the algorithm runs in $O(\sqrt{n}\, m \log nC)$ time and uses simple data structures. This time bound is comparable to the time taken by Gabow and Tarjan's scaling algorithm and is better than all other time bounds under the *similarity assumption* , i.e., $C = O(n^k)$ for some k. We next consider the minimum cycle mean problem. The *mean cost* of a cycle is defined as the cost of the cycle divided by the number of arcs it contains. The minimum cycle mean problem is to identify a cycle whose mean cost is minimum. We show that by using ideas of the assignment algorithm in an approximate binary search procedure, the minimum mean cycle problem can also be solved in $O(\sqrt{n}\ m \log nC)$ time. Under the similarity assumption, this is the best available time bound to solve the minimum cycle mean problem.

---

* On leave from Indian Institute of Technology, Kanpur, INDIA

1. Introduction

In this paper, we propose new scaling algorithms for the linear assignment problem and the minimum cycle mean problem. The *linear assignment* problem consists of a set N_1 denoting persons, another set N_2 denoting jobs for which $|N_1| = |N_2|$, a collection of pairs $A \subseteq N_1 \times N_2$ representing possible person to job assignments, and an *integer* cost c_{ij} associated with each element (i, j) in A. The objective is to assign each person to exactly one job and minimize the cost of the assignment. This problem can be stated as the following linear program :

$$\text{Minimize} \quad \sum_{(i,j) \in A} c_{ij} x_{ij} , \tag{1a}$$

subject to

$$\sum_{\{j : (i,j) \in A\}} x_{ij} = 1 , \text{for all} \quad i \in N_1 , \tag{1b}$$

$$\sum_{\{i : (i,j) \in A\}} x_{ij} = 1 , \text{for all } j \in N_2 \tag{1c}$$

$$x_{ij} \geq 0, \text{ for all } (i, j) \in A. \tag{1d}$$

The assignment problem can be considered as a network optimization problem on the graph $G = (N, A)$ consisting of the node set $N = N_1 \cup N_2$ and the arc set A. In the network context, this problem is also known as the *weighted bipartite matching* problem. We shall assume without any loss of generality that $c_{ij} \geq 0$ for all $(i, j) \in A$, since a suitably large constant can be added to all arc costs without changing the optimum solution. Let $C = \max \{c_{ij}\} + 1$. We also assume that the assignment problem has a feasible solution. The infeasibility of the assignment problem can be detected in $O(\sqrt{n}\, m)$ time by the bipartite cardinality matching algorithm of Hopcroft and Karp [1973], or by using the maximum flow algorithms of Even and Tarjan [1975] and Orlin and Ahuja [1987].

Instances of the assignment problem arise not only in practical settings, but also are encountered as a subproblem in algorithms for hard combinatorial optimization problems such as the quadratic assignment problem (Lawler [1976]), the traveling salesman problem (Karp [1977]), crew scheduling and vehicle routing

problems (Bodin et al. [1983]). In view of these applications, the development of faster algorithms for the assignment problem deserves special attention.

The literature devoted to the assignment problem is very rich. Kuhn [1955] developed the first algorithm for this problem, which is widely known as the *Hungarian algorithm.* The subsequent algorithms for the assignment problem can be roughly classed into the following three basic approaches :

(i) *Primal-Dual Approach*: Kuhn [1955], Balinski and Gomory [1964], Bertsekas [1981], Engquist [1982], and Glover et al. [1986], Dinic and Kronrod [1969].

(ii) *Dual Simplex Approach*: Weintraub and Barahona [1979], Hung and Rom [1980], Balinski [1985], and Goldfarb [1985].

(iii) *Primal Simplex Approach*: Barr et al. [1977], Hung [1983], Akgul [1985], and Orlin [1985].

The algorithms based on the primal-dual and dual simplex approaches have, in general, better running times in the worst case. Most of these algorithms (including the Hungarian algorithm) consist of obtaining, either explicitly or implicitly, a sequence of n shortest paths and augmenting flows along these paths. Using very simple data structures these algorithms run in $O(n^3)$ time, but can be implemented to run in $O(nm + n^2 \log n)$ time using the *Fibonacci heap* data structure of Fredman and Tarjan [1984], or in $O(nm + n^2 \sqrt{\log C})$ time using the *Redistributive heap* data structure of Ahuja, Mehlhorn, Orlin and Tarjan [1988]. (Recall that C is a strict upper bound on the largest arc cost in the network.)

Recently, scaling based approaches have been pursued for the assignment problem by Gabow [1985] and Gabow and Tarjan [1987]. Gabow [1985] developed the first scaling algorithm, which runs in time $O(n^{3/4} m \log nC)$. This time bound was subsequently improved to $O(\sqrt{n}\, m \log nC)$ by Gabow and Tarjan [1987]. Under the *similarity assumption,* i.e., that $C = O(n^k)$ for some k, this scaling algorithm runs in $O(\sqrt{n}\, m \log n)$ time. In this case, they are substantially faster than non-scaling algorithms in the worst case and, further, they do not require any sophisticated data structures.

The improved scaling algorithms rely on the concept of ε -optimality and execute a number of ε-scaling phases. The input at the ε-scaling phase is an assignment solution whose value is within $2nk\varepsilon$ of the optimum solution value for some small constant k (usually 2 or 3). The output of the ε-scaling phase is a solution whose value is within $2n\varepsilon$ of the optimum solution value. At this point, the value of ε is replaced by ε/k and a new scaling phase begins. Initially, $\varepsilon = C$. When $\varepsilon < 1/(2n)$, the algorithm terminates with an optimum solution. Goldberg and Tarjan [1987] showed that ε-scaling has very nice theoretical properties, and they used the approach to give improved algorithms for the minimum cost flow problem. Gabow and Tarjan [1987] used this approach to give an $O(\sqrt{n}\ m \log nC)$ algorithm for the assignment problem based on alternatively solving shortest path and *blocking* flow problems.

Recently, Bersekas [1987] developed a new approach for the assignment problem which assigns jobs to persons using "auction". The original version (Bertsekas [1979]) of this auction algorithms is not a polynomial time algorithm but Bertsekas [1987] pointed that ε - scaling can be used in conjunction with the auction algorithm to give an O(nm log nC) time bound for the assignment problem. In this paper, we improve Bertsekas's [1987] algorithm and show that a hybrid algorithm can achieve the same time bound as that of Gabow and Tarjan [1987] . Our algorithm also uses the ε-scaling approach, but at each scaling phase we run both the auction algorithm and the successive shortest path algorithm. The auction phase finds a partial assignment in which $O(\sqrt{n}\ )$ nodes remain unassigned. These remaining nodes are assigned via a successive shortest path algorithm. This hybrid algorithm runs in $O(\sqrt{n}\ m)$ time per scaling phase for a $O(\sqrt{n}\ m \log nC)$ time bound overall. Although this algorithm is comparable to that of Gabow and Tarjan in its worst case performance, it is different in many computational aspects. As such, the algorithms may run very differently in practice. Since the shortest path problem with arbitrary arc lengths can be reduced to the assignment problem (Lawler [1976]), our approach as well as that of Gabow and Tarjan yields $O(\sqrt{n}\ m \log nC)$ algorithm for the former problem.

We also use a variant of our assignment algorithm to solve the minimum cycle mean problem efficiently. The *mean cost* of a directed cycle W is defined as

$\sum_{(i,j) \in W} c_{ij} / |W|$. The *minimum cycle mean* of a network is a directed cycle whose

mean cost is minimum. Some applications of the minimum cycle mean problem can be found in periodic optimization (Orlin [1981]) and in the minimum cost flow algorithm of Goldberg and Tarjan [1988]. The minimum cycle mean problem can be solved in O(nm) time by the algorithm of Karp [1978] , or in O(nm log n) time by the algorithms of Karp and Orlin [1981] and Young [1987] . The minimum cycle mean problem is a special case of the minimum cost-to-time ratio cycle problem (see Lawler [1967]) and can also be solved as O(log nC) shortest path problems with arbitrary arc lengths. This gives an $O(\sqrt{n}\ m\ \log^2 nC)$ algorithms for the minimum cycle mean problem if our assignment algorithm is used to solve the shortest path problems. We show that the scaling algorithm for the assignment problem can be used more efficiently to obtain an $O(\sqrt{n}\ m \log nC)$ algorithm for the minimum mean cycle problem, i.e., the same running time as for the assignment problem. This improvement relies on "approximate binary search" as developed by Zemel [1982, 1987].

2. Exact and Approximate Optimality

A 0-1 solution x of (1) is called an *assignment*. If $x_{ij} = 1$ then i is *assigned* to j and j is *assigned* to i. A 0-1 solution x for which $\sum_{\{j:\ (i,j) \in A\}} x_{ij} \le 1$ for all $i \in N_1$ and $\sum_{\{i:\ (i,j) \in A\}} x_{ij} \le 1$ for all $j \in N_2$ is called a *partial assignment*. In other words, in a partial assignment, some nodes may be *unassigned*. Associated with any partial assignment x is an index set X defined as $X = \{(i, j) \in A : x_{ij} = 1\}$. We associate the dual variable $\pi(i)$ with the equation corresponding to node i in (1b) and the dual variable $-\pi(j)$ with the equation corresponding to node j in (1c). We subsequently refer to the 2n vector π as the vector of *node potentials*. We define the *reduced cost* of any arc $(i, j) \in A$ as $\bar{c}_{ij} = c_{ij} - \pi(i) + \pi(j)$. It follows from the linear programming duality theory that a partial assignment x is optimum for (1) if there exist node potentials π so that the following *(exact) optimality conditions* are satisfied :

C1. (*Primal Feasibility*) x is an assignment.

C2. (*Dual Feasibility*) (a) $\bar{c}_{ij} = 0$ for all $(i, j) \in X$; and
(b) $\bar{c}_{ij} \ge 0$ for all $(i, j) \in A$.

The concept of *approximate optimality* as developed by Bertsekas [1979] and Tardos [1985] is essential to our algorithms. This is a relaxation of the exact optimality conditions given above. A partial assignment x is said to be ε *-optimal* for some $\varepsilon \geq 0$ if there exist node potentials π so that the following ε - *optimality conditions* are satisfied:

C3. (*Primal Feasibility*) x is an assignment.

C4. (ε-*Dual Feasibility*) (a) $\bar{c}_{ij} \leq \varepsilon$ for all $(i, j) \in X$; and
(b) $\bar{c}_{ij} \geq -\varepsilon$ for all $(i, j) \in A$.

We associate with each partial assignment x a *residual network* G(x). The residual network G(x) has N as the node set, and the arc set consists of A plus an arc (j, i) of cost $-c_{ij}$ for every arc (i, j) for which $x_{ij} = 1$. Observe that $\bar{c}_{ij} = -\bar{c}_{ji}$, wherever both (i, j) and (j, i) are in the residual network.

The above conditions can be slightly simplified if presented in terms of the residual network. Observe that for all $(i, j) \in X$, (j, i) is also in the residual network G(x), and $\bar{c}_{ij} \leq \varepsilon$ is equivalent to $\bar{c}_{ji} = -\bar{c}_{ij} \geq -\varepsilon$. With respect to the residual network, the following conditions are equivalent to the ε- optimality conditions:

C5. (*Primal Feasibility*) x is an assignment.

C6. (ε- *Dual Feasibility*) $\bar{c}_{ij} \geq -\varepsilon$ for all (i, j) in G(x).

By setting $\varepsilon = 0$, the ε - optimality conditions reduce to the (exact) optimality conditions. In fact, when the costs are integer, the ε need not necessarily be equal to 0 for C6 to become equivalent to C2 as proved by Bertsekas [1979].

Lemma 1. (Bertsekas [1979]) If x^* is an optimum assignment and x is an ε- optimal assignment, then $c\,x - c\,x^* \leq 2n\varepsilon$

Proof. Let π be the node potentials corresponding to the ε- optimal assignment x. Since x is an ε - optimal assignment, it follows from condition C4 (a) that

$$\sum_{(i,j) \in X} \bar{c}_{ij} = c\,x - \sum_{i \in N_1} \pi(i) + \sum_{j \in N_2} \pi(j) \leq n\,\varepsilon\,. \qquad (2)$$

Further, since $\bar{c}_{ij} \geq -\varepsilon$ for every arc $(i, j) \in A$, it follows from condition C4(b) that

$$\sum_{(i,j) \in X^*} \bar{c}_{ij} = c\,x^* - \sum_{i \in N_1} \pi(i) + \sum_{j \in N_2} \pi(j) \geq -n\,\varepsilon. \qquad (3)$$

Combining (2) and (3), we get

$$c\,x - c\,x^* \leq 2n\,\varepsilon. \qquad (4) \quad \blacksquare$$

Corollary 1. *For any $\varepsilon < 1/(2n)$, an ε- optimal solution x of the assignment problem with integer costs is an optimum solution.*

Proof. For any $\varepsilon < 1/(2n)$, $c\,x - c\,x^* < 1$. Since x and all arc costs are integer, $c\,x = c\,x^*$. ■

3. The Scaling Algorithm

In this section, we present an O(nm log nC) algorithm for the assignment problem. The algorithm described here is a variant of Bertsekas's [1987] O(nm log nC) scaling algorithm. An improved algorithm running in time $O(\sqrt{n}\ m \log nC)$ will be described in the next section.

This scaling algorithm proceeds by obtaining ε- optimal assignments for successively smaller values of ε. In other words, the algorithm obtains solutions that are closer and closer to the optimum until $\varepsilon < 1/(2n)$, at which point the solution is optimum. The algorithm performs a number of *cost scaling phases*. In a cost scaling phase, the major subroutine of the algorithm is the *Improve-Approximation* procedure, which starts with a $k\varepsilon$- optimal assignment and produces an ε-optimal assignment. The parameter k is a constant and will typically have values 2, 3, or 4. A high-level description of the scaling algorithm is given below.

```
algorithm SCALING;
begin
    set π := 0 and ε := C;
    while ε ≥ 1/(2n) do
    begin
        ε := ε/k;
        IMPROVE-APPROXIMATION (c, k, ε, L, π, x);
    end;
    the solution x is an optimum assignment;
end;
```

We describe the *Improve-Approximation* in detail below. It iteratively performs three basic operations: (i) *assignment*, i.e., assigning a node i to a node j; (ii) *deassignment*, i.e., deassigning a node *l* from a node j; and (iii) *relabel*, i.e., increasing a node potential by ε units. An arc (i, j) is called *admissible* if $-\varepsilon \leq \bar{c}_{ij} < 0$. The procedure performs assignment only on *admissible* arcs. A node that is relabeled L + k times (for an appropriate choice of L) during an execution of the *Improve-Approximation* is not assigned anymore and is called *ineligible*. A node that is not ineligible is called an *eligible* node. For the algorithm in this section, we set L to a suitably large number so that all nodes are assigned at the termination of *Improve-Approximation*. A smaller value of L leads to the improvement in running time at the expense of leaving some nodes unassigned; we discuss improved choices of L in the next section. The *Improve-Approximation* works as follows.

```
procedure IMPROVE-APPROXIMATION (c, k, ε, L, π, x);
begin
    x := 0;
    π(j) := π(j) + kε  for all j ∈ N2 ;
    while there is an eligible unassigned node i in N1  do
        if there is an admissible arc (i, j)  then
        begin
            assign node i to node j;
            π(j) := π(j) + ε ;
            if node j was already assigned to some node l then
                deassign node l from node j ;
        end
        else π(i) := π(i) + ε ;
end;
```

We now make a few observations about the procedure.

Observation 1. The procedure stops examining a node in N_1 whenever its distance label has been increased (L + k) times. Consequently, at termination, it may not obtain a complete assignment if L is small. We will show that at most $2n(k + 1)/L$ nodes in N_1 are not assigned by the procedure. For $L = 2(k + 1)n + 1$, the procedure yields a complete assignment.

Observation 2. In each execution of the *Improve-Approximation*, the algorithm constructs an ε - optimal assignment starting with an empty assignment. The algorithm does not use the previous $k\varepsilon$ - optimal solution explicitly, it only implicitly uses the fact that there exists a $k\varepsilon$ - optimal solution. However, the node potentials are carried over to the next scaling phase and used explicitly.

Observation 3. The potential of an unassigned node $j \in N_2$ never changes until it is assigned (except at the beginning when it increases by $k\varepsilon$ units). Further, if a node $j \in N_2$ is assigned to a node in N_1 , then in all subsequent iterations of the *Improve-Approximation* procedure it remains assigned to some node in N_1.

In the above description, we left out the details on how to identify an admissible arc (i, j) incident on the node i. As per Goldberg and Tarjan [1986], we use

the following data structure to perform this operation. We maintain with each node $i \in N_1$, a list A(i) of arcs emanating from it. Arcs in this list can be arranged arbitrarily, but the order once decided remains unchanged throughout the algorithm. Each node i has a *current arc* (i, j) which is the next candidate for assignment. This list is examined sequentially and whenever the current arc is found to be inadmissible, the next arc in the arc list is made the current arc. When the arc list is completely examined, the node potential is increased and the current arc is again the first arc in its arc list. Note that if any arc (i, j) is found inadmissible during a scan of the arc list, it remains inadmissible until $\pi(i)$ is increased. This follows from the fact that all node potentials are nondecreasing.

In this section, we show that if $L = 2(k + 1)n + 1$, then *Improve-Approximation* runs in time O(nm). In the next section, we will allow L to be $2(k + 1)\lceil\sqrt{n}\rceil$. In this case *Improve-Approximation* subroutine runs in time $O(\sqrt{n}\, m)$ time and assigns all but at most $\lceil\sqrt{n}\rceil$ nodes. The remaining nodes can be assigned using a sequential shortest path algorithm that takes $O(\sqrt{n}\, m)$ time to assign to all unassigned nodes, leading to an $O(\sqrt{n}\, m)$ time bound for the *Improve-Approximation* subroutine, and an $O(\sqrt{n}\, m \log nC)$ time bound overall.

We now prove the correctness and the complexity results of the scaling algorithm. These results use ideas from bipartite matching theory given in Lawler [1976] , Bertsekas [1979] and the minimum cost flow algorithm of Goldberg and Tarjan [1987].

Lemma 2. *The partial assignment maintained by the Improve-Approximation procedure is always ε- optimal.*

Proof. This result is shown by performing induction on the number of iterations. At the beginning of the procedure, we have $c_{ij} - \pi(i) + \pi(j) \geq -k\varepsilon$ for all $(i, j) \in A$. Alternatively, $c_{ij} - \pi(i) + (\pi(j) + k\varepsilon) \geq 0$, for all $(i, j) \in A$. Hence when the potentials of all nodes in N_2 are increased by $k\varepsilon$, then condition C4(b) is satisfied for all arcs in A. The initial solution x is a null assignment and C4(a) is vacuously satisfied. The initial null assignment x, therefore, is $\varepsilon-$optimal (in fact, it is 0-optimal).

Each iteration of the **while** loop either assigns some unassigned node $i \in N_1$ to a job $j \in N_2$, or increases the node potential of node i. In the former case, $-\varepsilon \leq \bar{c}_{ij} < 0$ by the criteria of admissibility, and hence satisfies C4(a). Then $\pi(j)$ is

increased by ε and we get $0 \leq \bar{c}_{ij} < \varepsilon$. If node j was already assigned to some node l, then deassigning node l from node j does not affect the ε-optimality of the partial assignment. In the latter case, when the node potential of node i is increased, we have $\bar{c}_{ij} \geq 0$ for all $(i, j) \in A(i)$ before the increase, and hence $\bar{c}_{ij} \geq -\varepsilon$ for all $(i, j) \in A(i)$ after the increase. It follows that each iteration of the procedure preserved ε-optimality of the partial assignment. ■

Lemma 3. The Improve-Approximation procedure runs in O((k + L)m) time.

Proof. Each iteration of the procedure results in at least one of the following outcomes: (i) assignment; (ii) deassignment; and (iii) relabel of a node in N_1. Clearly, the number of relabels of nodes in N_1 is bounded by O((k + L)n). Initially, no node is assigned and, finally, all nodes may be assigned. Thus the number of assignments are bounded by n plus the number of deassignments. The deassignment of node l from node j causes the current arc of node l to advance to the next arc. After $|A(l)|$ such advancements for node l, a relabel operation takes place. Since any node $l \in N_1$ can be relabeled at most (k + L) times, we get a bound of

$O(\sum_{l \in N_1} |A(l)| (k + L)) = O((k + L)m)$ on the number of deassignments, and the lemma follows. ■

Lemma 4. At the termination of Improve-Approximation procedure, at most $n(k + 1)/L$ nodes in N_1 are unassigned in the partial assignment x.

Proof. Let x' denote some $k\varepsilon$-optimal assignment. Let π (resp., π') and $\bar{c}_{ij}$ (resp., $\bar{c}'_{ij}$) denote the node potentials and reduced costs corresponding to x (resp., x'). Let $i_0 \in N_1$ be an unassigned node in x. We intend to show that there exists a sequenceof nodes $i_0 - i_1 - i_2 - \ldots - i_l$ such that $i_l \in N_2$ and i_l is unassigned in x. Further, this sequence satisfies the property that (i_0, i_1), $(i_2, i_3), \ldots, (i_{l-1}, i_l)$ are in the assignment x' and the arcs $(i_2, i_1), (i_4, i_3), \ldots, (i_{l-1}, i_{l-2})$ are in the assignment x. Finally, we will show that $l \geq L/k+1$.

Suppose that node $i_0 \in N_1$ is assigned to node i_1 in x'. If node $i_1 \in N_2$ is unassigned in x, then we have discovered such a path; otherwise let node $i_2 \in N_1$ be assigned to the node i_1 in x. Observe that $i_2 \neq i_0$. Now let node i_2 be assigned to node $i_3 \in N_2$ in x'. Observe that $i_3 \neq i_1$. If node i_3 is unassigned in x, then we are

done ; else repeating the same logic yields that eventually a sequence satisfying the above properties would be discovered. Now observe that the path P consisting of the node sequence $i_0 - i_1 - i_2 - \ldots - i_l$ is in the residual network G(x) and its reversal $\overline{P}$ consisting of node sequence $i_l - i_{l-1} - \ldots i_2 - i_1 - i_0$ is in the residual network G(x'). The ε-optimality of the solution x yields that

$$\sum_{(i,j) \in \overline{P}} \overline{c}_{ij} = \pi(i_l) - \pi(i_0) + \sum_{(i,j) \in P} c_{ij} \geq -l\varepsilon. \qquad (5)$$

Further, from the kε - optimality of the solution x it follows that

$$\sum_{(i,j) \in \overline{P}} \overline{c}'_{ij} = \pi'(i_0) - \pi'(i_l) + \sum_{(i,j) \in \overline{P}} c_{ij} \geq -kl\varepsilon. \qquad (6)$$

Note that $\pi(i_l) = \pi'(i_l) + k\varepsilon$ since the potential of node i_l increases by kε at the beginning of the scaling phase and then remains unchanged. Also note that $\sum_{(i,j) \in \overline{P}} c_{ij} = - \sum_{(i,j) \in P} c_{ij}$. Using these facts in (5) and (6), we get

$$\pi(i_0) \leq \pi'(i_0) + \{k + (k+1)l\}\varepsilon \qquad (7)$$

Finally, we use the fact that $\pi(i_0) = \pi'(i_0) + (L + k)\varepsilon$, i.e., the potential of each unassigned node in N_1 has been increased (L + k) times. This gives $l \geq L/(k+1)$. To summarize, we have shown that for every unassigned node i_0 in N_1 there exists an unassigned node i_l in N_2 and a path P between these two nodes consisting of at least L/(k+1) nodes such that P is in the G(x) and its reversal $\overline{P}$ is in G(x'). The facts that x' is an assignment and x is a partial assignment imply that these paths corresponding to two different unassigned nodes in N_1 are node disjoint. Consequently, there are at most 2n(k + 1)/L unassigned nodes in N_1 . ■

The Lemmas 3 and 4 immediately yield the following result:

Theorem 1. By setting L = 2(k + 1)n + 1, the Improve-Approximation procedure yields an ε- optimal assignment in O(nm) time. Consequently, the scaling algorithm solves the assignment problem in O(nm log nC) time. ■

We derive another result from the proof of Lemma 4 that will be used in the next section.

Corollary 2. $\sum_{(i,j) \in P} \bar{c}_{ij}/\varepsilon \leq k + kl$.

Proof. Using the facts that $\pi(i_0) \geq \pi'(i_0)$ and $\pi(i_l) = \pi'(i_l) + k\varepsilon$ in (6) we get

$$\pi(i_l) - \pi(i_0) + \sum_{(i,j) \in P} \bar{c}_{ij} \leq k\varepsilon + kl\varepsilon \qquad (8)$$

The corollary is now evident. ■

3. The Improved Scaling Algorithm

In this section, we improve the complexity of the scaling algorithm to $O(\sqrt{n}\, m \log nC)$ This improvement is based on the observation that the procedure *Improve-Approximation* takes a relatively small amount of time to assign a large number of nodes in N_1 and then a potentially large amount of time to assign a small number of nodes. To explain this in rigorous terms, consider the procedure with $L = 2(k + 1)\sqrt{n}$. It follows from Lemmas 3 and 4 that the procedure runs in $O(\sqrt{n}\, m)$ time and yields an assignment in which at least $n - \lceil\sqrt{n}\rceil$ nodes in N_1 are assigned. If the procedure is allowed to run further, then it would take another $O((n - \sqrt{n})m)$ time to assign the remaining $\lceil\sqrt{n}\rceil$ nodes in N_1 . Our improvement results from setting $L = 2(k + 1)\lceil\sqrt{n}\rceil$ and then assigning the remaining at most $\lceil\sqrt{n}\rceil$ nodes more efficiently by using the successive shortest path method for the minimum cost flow problem. The successive shortest path method takes O(m) time to assign an unassigned node and hence takes a total of $O(\sqrt{n}\, m)$ time. Consequently, the improved scaling algorithm runs in $O(\sqrt{n}\, m \log nC)$ time.

Our algorithm is related to the assignment algorithm of Gabow and Tarjan in the following manner. During an ε-scaling phase, the algorithm of Gabow and Tarjan performs $O(\sqrt{n})$ iterations. Each iteration consists of solving a shortest path problem to update node potentials, defining a layered network and then establishing a maximal flow in the layered network; these operations take O(m) time per iteration. On the other hand, our algorithm applies auction algorithm until at most

$\sqrt{n}$ nodes are unassigned, and then uses the successive shortest path algorithm to assign the unassigned nodes.

Let x°, π°, and $\bar{c}^\circ_{ij}$ respectively denote the partial assignment, the node potentials and reduced costs at the end of the *Improve-Approximation* procedure. We use the following method to assign the remaining unassigned nodes.

procedure SUCCESSIVE SHORTEST PATH;
begin
$x^* := x^\circ$;
$\pi^1(i) := 0$ for all $i \in N$;
$d_{ij} := \begin{cases} 0, & \text{if } -\varepsilon \le \bar{c}^\circ_{ij} \le \varepsilon \text{, for all (i, j) in } G(x^\circ); \\ \lceil \bar{c}^\circ_{ij} / \varepsilon \rceil, & \text{otherwise;} \end{cases}$
while there are unassigned nodes in N_1 **do**
begin
select an unassigned node i_0 in N_1;
define reduced costs as $\bar{d}_{ij} = d_{ij} - \pi^1(i) + \pi^1(j)$;
apply Dijkstra's algorithm with $\bar{d}_{ij}$ as arc lengths
starting at node i_0 until some node i_l in N_2 is
permanently labeled;
let w(i) denote the distance labels of nodes;
set $\pi^1(i) := \pi^1(i) - w(i)$ for all permanently labeled nodes i and
set $\pi^1(i) := \pi^1(i) - w(i_l)$ for all temporarily labeled node i;
augment one unit of flow along the shortest path
from node i_0 to i_l and update x^*;
end;
end;

At the beginning of the above method we define the new arc lengths as d_{ij}. The manner in which the new arc lengths are defined it follows that

$$\left(\bar{c}^\circ_{ij}/\varepsilon\right) - 1 \le d_{ij} \le \left(\bar{c}^\circ_{ij}/\varepsilon\right) + 1 \qquad (9)$$

Now observe from Corollary 3 that for every unassigned node $i_0 \in N_1$ in x° there exists an unassigned node $i_l \in N_2$ and a path P of length l between these two

nodes in the residual network such that $\sum_{(i,j) \in P} \bar{c}^{\circ}_{ij} / \varepsilon \leq k + k\,l$. Using (9) we obtain that $\sum_{(i,j) \in P} d_{ij} \leq k + l + k\,l = O(n)$. The fact that this path has a *small* length is used to improve the complexity of Dijkstra's shortest path algorithm.

It can be easily verified that the initial partial assignment x° satisfies the exact optimality conditions C1 and C2 with respect to the arc lengths d_{ij} and the node potentials $\pi^1 = 0$. The optimality of the partial assignment is then maintained and the partial assignment is gradually converted into an assignment by applying the well known successive shortest path method for the minimum cost flow problem (see Lawler [1976], Glover et. al [1986]). We use reduced costs as arc lengths so that they are nonnegative, and hence Dijkstra's [1956] algorithm can be applied. Observe that we stop Dijkstra's algorithm whenever an unassigned node in N_2 is permanently labeled. The manner in which node potentials are updated assures that the dual feasible conditions C2(a) and (b) are satisfied and there is a path of zero reduced arcs from node i_0 to node i_l in the residual network. We use Dial's implementation (Dial et al. [1979]) of Dijkstra's algorithm to solve the shortest path problems. This implementation consists of maintaining the linked lists LIST(r) to store nodes whose temporary distance labels equal r, and sequentially scanning these lists to identify a node with smallest distance label. It can be easily shown that this implementation will solve the above shortest path problems in O(m) time. As there are at most $\lceil \sqrt{n} \rceil$ such iterations, the above method would take $O(\sqrt{n}\, m)$ time to terminate. Let x^* be the final assignment and π^1 be the final node potentials.

We now show that the assignment x^* is an ε - optimal assignment with respect to the node potentials $\pi^* = \pi^{\circ} + \varepsilon\,\pi^1$. It follows from the optimality of the solution x^* with respect to the arc costs d_{ij} and node potential π^1 that

$$d_{ij} - \pi^1(i) + \pi^1(j) = 0, \text{ for all } (i,j) \in X^*, \tag{10}$$

$$d_{ij} - \pi^1(i) + \pi^1(j) \geq 0, \text{ for all } (i,j) \in A. \tag{11}$$

The inequalities in (9) can be alternatively stated as

$$\varepsilon\, d_{ij} \geq c_{ij} - \pi^{\circ}(i) + \pi^{\circ}(j) - \varepsilon, \tag{12}$$

$$\varepsilon d_{ij} \leq c_{ij} - \pi^{\circ}(i) + \pi^{\circ}(j) + \varepsilon. \qquad (13)$$

Multiplying (10) and (11) by ε and substituting for εd_{ij} from (12) and (13) respectively yields

$$c_{ij} - (\pi^{\circ}(i) + \varepsilon\pi'(i)) + (\pi(j) + \varepsilon\pi(j)) \leq \varepsilon, \text{ for all } (i, j) \in X, \qquad (14)$$

$$c_{ij} - (\pi^{\circ}(i) + \varepsilon\pi'(i)) + (\pi^{\circ}(j) + \varepsilon\pi'(j)) \geq -\varepsilon, \text{ for all } (i, j) \in A. \qquad (15)$$

Hence the assignment x^* satisfies C3 and C4 with respect to the node potentials $\pi^{\circ} + \varepsilon\pi^1$ and is ε - optimal. We have thus obtained the following result :

Theorem 2. The modified Improve-Approximation procedure runs in $O(\sqrt{n}\, m)$ time. The modified scaling algorithm determines an optimum assignment in $O(\sqrt{n}\, m \log nC)$ time. ■

5. The Minimum Cycle Mean Problem

In this section, we develop scaling algorithms for the minimum cycle mean problem. Recall that the mean cost of a directed cycle W is defined as

$\sum_{(i, j) \in W} c_{ij} / |W|$ and the minimum cycle mean problem is to identify a cycle of minimum mean cost. The minimum cycle mean problem is a special case of the minimum cost-to-time ratio cycle problem studied by Dantzig, Blattner and Rao [1967], Lawler [1967], and Fox [1969], and we use its special structure to develop more efficient algorithms.

We consider the network G = (N, A) where (possibly negative) integer c_{ij} represents the cost of each arc (i, j) $\in$ A. Let n = |N| and m = |A|. Let C = 1 + max ($|c_{ij}|$: (i, j) $\in$ A). We assume that the network contains at least one cycle, otherwise there is no feasible solution. Acyclicity of a network can be determined in O(m) time (see, for example, Aho, Hopcroft and Ullman [1974]). We first show how the minimum mean cycle problem can be solved by performing binary search on the interval containing the optimum value and solving an assignment problem to reduce the interval length by a constant factor. We then improve this algorithm by using Zemel's [1982, 1987] "approximate binary search." Each approximate binary search iteration is solved using a single application of

Improve-Approximation procedure. This refined algorithm is shown to run in $O(\sqrt{n}\, m \log nC)$ time, which is same as solving an assignment problem.

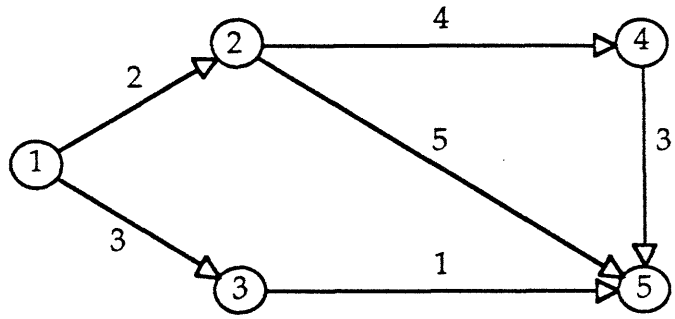
Our algorithms are based on a well known transformation using *node splitting*. We split each node $i \in N$ into two nodes i and i', and replace each arc (i, j) in the original network by the arc (i, j'). The cost of the arc (i, j') is same as that of arc (i, j) . We also add arcs (i, i') of cost δ for each $i \in N$ in the transformed network. This gives us an assignment problem with $N_1 = \{1, 2, \ldots, n\}$ and $N_2 = \{1', 2', \ldots, n'\}$. An example of this transformation is given in Figure 1.

We treat δ as a parameter in the transformation. We represent the costs of the assignment problem by $C(\delta)$ and refer to the problem as *ASSIGN(δ)*. An optimum solution of ASSIGN(δ) either consists solely of the arcs (i, i') for all i, or does not contain (i, i') for some i. In the former case, the assignment is called the *uniform* assignment, and in the latter case the assignment is called a *nonuniform* assignment. We represent the minimum mean length in the network by μ^*. Our first algorithm is based on the following result :

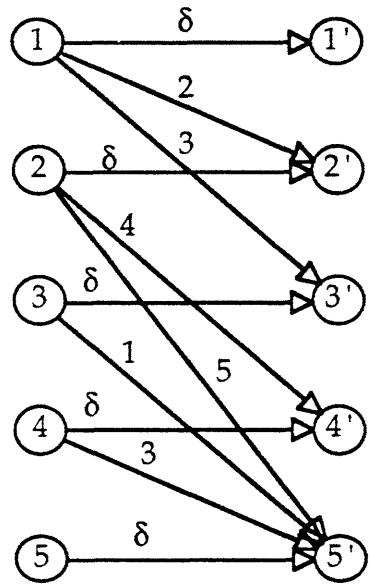
Lemma 5. (a) If an optimum assignment of ASSIGN(δ) is uniform, then $\delta \le \mu^*$. (b) If an optimum assignment of ASSIGN(δ) is nonuniform, then $\mu^* \le \delta$.

Proof. (Part(a)) Suppose that the optimum assignment of ASSIGN(δ) be uniform. Suppose further that there is a cycle W in G consisting of node sequence $i_1 - i_2 - \cdots - i_r - i_1$ whose mean cost is less than δ. Then $\sum_{(i,j) \in W} c_{ij} < r\delta$. Now, note that replacing the arcs (i_1, i_1'), (i_2, i_2'), ..., (i_r, i_r') in the uniform assignment by arcs (i_1, i_2'), (i_2, i_3'), ..., (i_{r-1}, i_r'), (i_r, i_1') yields an assignment of lower cost, which contradicts the optimality of the uniform assignment.

(Part (b)) Since the optimum assignment is nonuniform there exists a node $j_1 \in N_1$ that is assigned to a node $j_2' \in N_2$ where $j_1 \neq j_2$. Suppose that node j_2 is assigned to some node j_3'. Note that $j_2 \neq j_3$. Extending this logic further indicates that eventually we would discover a node $j_r \in N_1$ which is assigned to the node $j_1' \in N_2$. Since the partial assignment (j_1, j_2'), (j_2, j_3'), ..., (j_r, j_1') is not more expensive than the partial assignment (j_1, j_1'), (j_2, j_2'), (j_3, j_3'), ..., (j_r, j_r'), it follows



(a) The original network.



(b) The transformed network.

FIGURE 1. AN EXAMPLE OF TRANSFORMATION USING NODE SPLITTING.

that the cost of the cycle W in G consisting of the node sequence $j_1 - j_2 - j_3 - \ldots - j_r - j_1$ is no more than $r\delta$. Thus W is a cycle with mean cost no more than δ. ■

Notice that $-C < \mu^* < C$ since the absolute value of all arc costs are strictly less than C. Furthermore, if the mean cycle costs of two cycles are unequal, then they must differ by at least $1/n^2$. This can be seen as follows. Let W_1 and W_2 be two distinct cycles and $L_k = \sum_{(i,j) \in W_k} c_{ij}$ and $r_k = |W_k|$ for each $k = 1, 2$. Then

$$\left| \frac{L_1}{r_1} - \frac{L_2}{r_2} \right| = \left| \frac{r_2 L_1 - r_1 L_2}{r_1 r_2} \right| \neq 0.$$

The numerator in the above expression is at least 1 and the denominator is at most n^2, thereby proving our claim. These observations and the above lemma yield the following minimum mean cycle algorithm using binary search.

procedure MIN-CYCLE-MEAN I;
begin
  set LB : = – C and UB : = C;
  while (UB – LB) ≥ $1/n^2$ **do**
  begin
    δ : = (UB + LB)/2;
    solve the problem ASSIGN(δ) ;
    if the optimum assignment x is uniform **then** LB : = δ
    else UB : = δ and x* : = x;
  end;
  use the nonuniform assignment x* to construct
    the minimum mean cycle of cost μ* in the interval [LB, UB] ;
end;

The above algorithm starts with an interval [LB, UB] of length 2C containing the minimum mean cost and halves its length in every iteration. After $2 + \lceil \log(n^2 C) \rceil = O(\log nC)$ iterations, the interval length goes below $1/n^2$ and the algorithm terminates. Since the minimum mean cycle value is strictly less than the initial upper bound (which is C), and the network has at least one cycle, the final interval [LB, UB] has UB < C and there is a nonuniform assignment corresponding to this value of UB. If we use the improved scaling algorithm described in

Section 4 to solve the assignment problems, then the above algorithm runs in time $O(\sqrt{n}\, m \log^2 nC)$.

We now improve the above algorithm to obtain a bound of $O(\sqrt{n}\, m \log nC)$. This improvement is based on the following generalization of Lemma 5.

Lemma 6 *(a) If an ε-optimal assignment x of ASSIGN(δ) is uniform, then $\delta - 2\varepsilon \le \mu^*$.*

(b) If an ε-optimal assignment x of ASSIGN(δ) is nonuniform, then $\mu^* \le \delta + 2\varepsilon$

Proof. (Part(a)) Let π be the node potentials corresponding to the ε - optimal assignment x. Suppose the cycle W^* in G consisting of the node sequence $i_1 - i_2 - i_3 - \ldots - i_r - i_1$ is a minimum mean cycle of cost μ^*. Let $I^* = \{(i_1, i_2'), (i_2, i_3'), \ldots, (i_{r-1}, i_r'), (i_r, i_1')\}$. It follows from condition C4(b) that

$$\sum_{(i,j) \in I^*} \bar{c}_{ij} = r\mu^* - \sum_{i=i_1}^{i_r} \pi(i) + \sum_{i=i_1}^{i_r'} \pi(i) \ge -r\varepsilon. \qquad (16)$$

Let $I = \{(i_1, i_1'), (i_2, i_2'), \ldots, (i_r, i_r')\}$. As I is part of the ε-optimal assignment X, it follows from condition C3 that

$$\sum_{(i,j) \in X} \bar{c}_{ij} = r\delta - \sum_{i=i_1}^{i_r} \pi(i) + \sum_{i=i_1}^{i_r'} \pi(i) \le -r\varepsilon. \qquad (17)$$

Combining (16) and (17) we get

$$\delta - 2\varepsilon \le u^* . \qquad (18)$$

(Part(b)) Let W be a cycle consisting of the node sequence $j_1 - j_2 - \ldots - j_r - j_1$ constructed from the nonuniform assignment X in the same manner as described in the proof of Lemma 5(b). Let μ be the mean cost of W. Then $\mu^* \le \mu$. Let $I = \{(j_1, j_2'), (j_2, j_3'), \ldots, (j_{r-1}, j_r'), (j_r, j_1')\}$ and $J = \{(j_1, j_1'), (j_2, j_2'), \ldots, (j_r, j_r')\}$. It follows from C4(b) and C3(a), respectively, that

$$\sum_{(i,j)\in J} \bar{c}_{ij} = r\delta - \sum_{j=j_1}^{j_r} \pi(j) + \sum_{j=j_1}^{j_r'} \pi(j) \geq -r\varepsilon. \qquad (19)$$

$$\sum_{(i,j)\in I} \bar{c}_{ij} = r\mu - \sum_{j=j_1}^{j_r} \pi(j)) + \sum_{j=j_1}^{j_r'} \pi(j) \leq r\varepsilon. \qquad (20)$$

Combining (19) and (20) we get

$$\mu^* \leq \mu \leq \delta + 2\varepsilon. \qquad (21) \quad \blacksquare$$

We use this result to obtain an improved algorithm for the minimum mean cycle problem. The algorithm maintains an interval [LB, UB] containing the minimum mean cycle value μ^* and in one application of the *Improve-Approximation* procedure with carefully selected values of δ and ε reduces the interval length by a constant factor. A formal description of this algorithm is given below followed by its justification.

algorithm MIN-CYCLE-MEAN II;
begin
set LB : = – C and UB : = C;
set $\pi(i)$: = – C/2 for all $i \in N_1$ and $\pi(i')$: = 0 for all $i' \in N_2$;
let x be the uniform assignment;
while (UB – LB) $\geq 1/n^2$ **do**
begin
δ : = (UB + LB)/2;
ε : = (UB – LB)/8;
k : = 3;
IMPROVE-APPROXIMATION (c(δ), k, ε, L, π, x);
if x is an uniform assignment **then** LB : = δ – 2ε
else UB : = δ + 2ε and x^* : = x;
end;
end;

At termination, we consider the nonuniform assignment x^* and identify a cycle as described in the proof of Lemma 5(b). This cycle has the mean cost in the range (LB, UB) and since $UB - LB < 1/n^2$, the cycle must be a minimum mean cycle.

Theorem 3. The above algorithm correctly determines a minimum mean cycle in $O(\sqrt{n}\, m \log nC)$ time.

Proof. The algorithm always maintains an interval [LB, UB] containing the minimum mean cost. If the execution of *Improve-Approximation* procedure yields a uniform assignment x, then the new lower bound is $\delta - 2\varepsilon = (UB + 3LB)/4$; otherwise the new upper bound is $\delta + 2\varepsilon = (3UB + LB)/4$. In either case, the interval length (UB – LB) decreases by 25%. Since, initially UB – LB = 2C, after at most $1 + \lceil \log_{4/3} n^2C \rceil = O(\log nC)$ iterations, $UB-LB < 1/n^2$ and the algorithm terminates. Each execution of the *Improve-Approximation* procedure takes $O(\sqrt{n}\, m)$ time and the algorithm runs in $O(\sqrt{n}\, m \log nC)$ time.

To show the correctness of the algorithm, we need to show that the solution x is 3ε-optimal with respect to the costs $C(\delta)$ before calling *Improve-Approximation* procedure. Initially, x is the uniform assignment, $\delta = 0$ and $\varepsilon = C/4$. Hence for each arc (i, i') in the uniform assignment, $\bar{c}_{ii'} = C/2 \le 3C/4$, thereby satisfying C4(a). Further, for each arc (i, j') in ASSIGN(δ), we have $\bar{c}_{ij'} = c_{ij'} + C/2 \ge -C + C/2 = -C/2 \ge -3(C/4)$, which is consistent with C4(a).

Now consider any general step. Let the assignment x be an ε - optimal solution for ASSIGN(δ) and let $\bar{c}_{ij}$ denote the reduced costs of the arcs at this stage. Further, let UB', LB', δ', ε' and $\bar{c}'_{ij}$ be the corresponding values in the next iteration. We need to show that the solution x is $3\varepsilon'$-optimal for ASSIGN(δ'). We consider the case when $\delta' > \delta$. The case when $\delta' < \delta$ can be proved similarly. It follows from the above discussion that $\varepsilon' = 3\varepsilon/4$ and $\delta' = (UB' + LB')/2 = (UB + (UB + 3LB)/4)/2 = \delta + \varepsilon$. Since arc costs do not decrease in ASSIGN(δ') , all arcs keep satisfying C4(b). For an arc $(i, j') \in X$, we have $\bar{c}'_{ij'} \le \bar{c}_{ij'} + \varepsilon$ because the costs increase by at most ε units. We then use the fact that $\bar{c}_{ij'} \le \varepsilon$ to obtain $\bar{c}'_{ij'} \le 2\varepsilon \le 3\varepsilon'$, which is consistent with C4(a). The theorem now follows. ■

The minimum cycle mean algorithm can possibly be sped up in practice by using a better estimate of the upper bound UB. Whenever an application of the *Improve-Approximation* procedure yields a non-uniform assignment x, then a cycle

is located in the original network using the assignment x and the upper bound UB is set of the mean cost of this cycle. Further, if it is desirable to perform all computations in integers, then we can multiply all arc costs by n^2, initially set UB = $k^{\lceil \log_k n^2 C \rceil}$, and terminate the algorithm when $\varepsilon < 1$. The accuracy of this modified version can be easily established.

6. Conclusions

In this paper, we have developed an ε-scaling approach for both the assignment problem and the minimum cycle mean problem for which each scaling phase is a hybrid between the auction algorithm and the successive shortest path algorithm. In doing so, we have developed an algorithm that is comparable to the best for the assignment problem and an algorithm that is currently the best for the minimum cycle mean problem, in terms of worst-case complexity.

The algorithms presented here illustrate the power of scaling approaches. Under the similarity assumption, scaling based approaches are currently the best available in worst-case complexity for all of the following problems : the shortest path problem with arbitrary arc lengths, the assignment problem, the minimum cycle mean problem, the maximum flow problem, the minimum cost flow problem, the weighted non-bipartite matching problem, and the linear matroid intersection problem.

The algorithm presented here also illustrates the power of hybrid approaches. The auction algorithm used with scaling would lead to an O(nm log nC) time bound. The successive shortest path algorithm used with scaling would lead to an O(nm log nC) time bound. (This latter time bound is not even as good as the time bound achievable for the successive shortest path algorithm without scaling.) Yet combining the auction algorithm and the successive shortest path algorithm leads to an improvement in the running time by $\sqrt{n}$. As an example, if n = 10, 000, then in each scaling phase the auction algorithm without scaling would assign the first 99% of the nodes in 1% in the overall running time, and would assign the remaining 1% of the nodes in the remaining 99% of the running time. By using a successive shortest path algorithm, we can improve the running time for assigning the last 1% of the nodes by a factor of nearly 100 in the worst case.

In addition, our algorithm for the minimum cycle mean problem illustrates that one can get computational improvements by using Zemels' approach of solving binary search approximately. Suppose that μ^* is the optimal solution value for the minimum cycle mean problem. In classical binary search, at each stage we would have a lower bound LB and an upper bound UB that bound μ^*. Then we would determine whether $\mu^* \leq (LB + UB)/2$ or $\mu^* \geq (LB + UB)/2$. In either case, the size of the bounding interval is decreased by 50%. In our version, we only determine the interval for μ^* approximately. In particular, we determine either

i. $\mu^* \leq LB/4 + 3UB/4$, or

ii. $\mu^* \geq 3LB/4 + UB/4$.

In either case, the size of the bounding interval is decreased by 25%. The number of iterations is approximately twice that of the traditional binary search; but each iteration can be performed much more efficiently

We close the paper with an open question. The question is whether the problem of detecting a negative cost cycle in a network can be solve faster than the problem of finding the minimum cycle maan of a network. This latter problem is more general in that the minimum cycle mean is negative if and only if there is a negative cost cycle. In addition, the minimum cycle mean problem seems as though it should be more difficult to solve. Nevertheless, the best time bounds for these two problems are the same. The best time bound under the similarity assumption is $O(\sqrt{n}\, m \log nC)$. Under the assumption that C is exponentially large, the best time bound for each of these problems is O(nm). Determining a better time bound for the problem of detecting a negative cost cycle is both of theoretical and practical importance.

Acknowledgements

We thank Michel Goemans and Hershel Safer for a careful reading of the paper and many useful suggestions. This research was supported in part by the Presidential Young Investigator Grant 8451517-ECS of the National Science Foundation, by Grant AFOSR-88-0088 from the Air Force Office of Scientific Research, and by Grants from Analog Devices, Apple Computer, Inc., and Prime Computer.

References

Aho, A.V., J.E. Hopcroft, and J.B. Ullman. 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA.

Ahuja, R. K., K. Mehlhorn, J.B. Orlin, and R.E. Tarjan. 1988. Faster Algorithms for the Shortest Path Problem. Technical Report, Operations Research Center, M.I.T., Cambridge, MA.

Akgul, M. 1985. A Genuinely Polynomial Primal Simplex Algorithm for the Assignment Problem. Research Report, Department of Computer Science and Operations Research, N.C.S.U., Raleigh, N.C.

Balinski, M.L. 1985. Signature Methods for the Assignment Problem. *Oper. Res.* 33, 527-536.

Balinski, M.L., and R.E. Gomory. 1984. A Primal Method for the Assignment and Transportation Problems. *Man. Sci.* 10, 578-593.

Barr, R. , F. Glover, and D. Klingman. 1977. The Alternating Path Basis Algorithm for the Assignment Problem. *Math. Prog.* 12, 1–13.

Bertsekas, D. P. 1979. A Distributed Algorithm for the Assignment Problem. LIDS Working Paper, M.I.T., Cambridge, MA.

Bertsekas, D. P. 1981. A New Algorithm for the Assignment Problem. *Math. Prog.* 21, 152–171.

Bertsekas, D.P. 1987. The Auction Algorithm : A Distributed Relaxation Method for the Assignment Problem. Technical Report LIDS-P-1653, M.I.T. Cambridge, MA. (To Appear in *Annals of Operations Research,* 1988.)

Bodin, L.D., B.L. Golden, A.A. Assad, and M.O. Ball. 1983. Routing and Scheduling of Vehicles and Crews. *Comp. and Oper. Res.*10, 65–211.

Dantzig, G.B., W.O. Blattner, and M.R. Rao. 1967. Finding a Cycle in a Graph with Minimum Cost-to-Time Ratio with Application to a Ship Routing Problem. *Theory*

of Graphs: International Symposium, Dunod, Paris, and Gordon and Breach, New York, 77–83.

Dial, R., F. Glover, D. Karney, and D. Klingman. 1979. A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees. *Networks* 9, 215–248.

Dijkstra, E. 1959. A Note on Two Problems in Connexion with Graphs. *Numeriche Mathematics* 1, 269–271.

Dinic, E. A. 1970. Algorithm for Solution of a Problem of Maximum Flow in Networks with Power Estimation. *Soviet Maths. Doklady* 11, 1277-1280.

Dinic, E.A., and M.A. Kronrod. 1969. An Algorithm for Solution of the Assignment Problem. *Soviet Maths. Doklady* 10, 1324–1326.

Engquist, M. 1982. A Successive Shortest Path Algorithm for the Assignment Problem. *INFOR* 20, 370–384.

Even, S., and R.E. Tarjan. 1975. Network Flow and Testing Graph Connectivity. *SIAM J. of Computing* 4, 507-518.

Fox, B. 1969. Finding Minimal Cost-Time Ratio Circuits. *Oper. Res.* 17, 546–551.

Fredman, M.L., and R.E. Tarjan. 1984. Fibonacci Heaps and Their Uses in Network Optimization Algorithms. *Proc. 25th Annual IEEE Symp. on Found. of Comp. Sc.i*, 338-346.

Gabow, H.N. 1985. Scaling Algorithms for Network Problems. *J. of Comp. and Sys. Sci.* 31, 148–168.

Gabow, H.N., and R.E. Tarjan. 1987. Faster Scaling Algorithms for Graph Matching. Technical Report, Dept. of Computer Science, Princeton University, Princeton, NJ.

Glover, F., R. Glover, and D. Klingman. 1986. Threshold Assignment Algorithm. *Math. Prog. Study* 26, 12-37.

Goldberg, A.V., and R.E. Tarjan. 1986. A New Approach to the Maximum Flow Problem. *Proc. 18th ACM Symp. on Theory of Computing*, 136–146.

Goldberg, A.V., and R.E. Tarjan. 1987. Finding Minimum-Cost Circulations by Successive Approximation. *Proc. 19th ACM Symp. on Theory of Computing* , 7-18.

Goldberg, A.V., and R.E. Tarjan. 1988. Finding Minimum-Cost Circulations by Canceling Negative Cycles. To appear in *Proc. 20th ACM Symp. on Theory of Computing*.

Goldfarb, D. 1985. Efficient Dual Simplex Algorithms for the Assignment Problem. *Math. Prog.* 33, 187–203.

Hopcroft, J.E., and R.M. Karp. 1973. An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs. *SIAM J. of Comp.* 2, 225-231.

Hung, M.S. 1983. A Polynomial Simplex Method for the Assignment Problem. *Oper. Res.* 31, 595–600.

Hung, M.S., and W.O. Rom. 1980. Solving the Assignment Problem by Relaxation. *Oper. Res.* 28, 969–892.

Karp, R.M. 1977. Probabilistic Analysis of Partitioning Algorithms for the Traveling Salesman Problem in the Plane. *Maths. Oper. Res.* 2, 209–224.

Karp, R.M. 1978. A Characterization of the Minimum Cycle Mean in a Diagraph. *Discrete Mathematics* 23, 309-311.

Karp, R.M., and J. B. Orlin. 1981. Parametric Shortest Path Algorithms with an Application to Cyclic Staffing. *Disc. Appl. Maths.* 3, 37–45.

Kuhn, H.W. 1955. The Hungarian Method for the Assignment Problem. *Naval Res. Log. Quart.* 2, 83–97.

Lawler, E.L. 1967. Optimal Cycles in Doubly Weighted Directed Linear Graphs. *Theory of Graphs : International Symposium*, Dunod, Paris, and Gordon and Breach, New York, 209–213.

Lawler, E.L. 1976. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, New York.

Megiddo, M. 1979. Combinatorial Optimization with Rational Objective Functions. *Maths. of Oper. Res.* 4, 414–424.

Orlin, J.B. 1981. The Complexity of Dynamic Languages and Dynamic Optimization Problems. *Proc. of the 13th Annual ACM Symp. on the Theory of Computing,* 218-227.

Orlin, J.B. 1985. On the Simplex Algorithm for Networks and Generalized Networks. *Math. Prog. Studies* 24, 166–178.

Orlin, J.B., and R.K. Ahuja, 1987. New Distance–Directed Algorithms for Maximum Flow and Parametric Maximum Flow Problems. Sloan W.P. No. 1908–87, Sloan School of Management, M.I.T., Cambridge, MA.

Tardos, E. 1985. A Strongly Polynomial Minimum Cost Circulation Algorithm. *Combinatorica* 5, 247–255.

Weintraub, A., and F. Barahona. 1979. A Dual Algorithm for the Assignment Problem. Departmente de Industrias Report No. 2, Universidad de Chile–Sede Occidente.

Young, N. 1987. Finding a Minimum Cycle Mean: An O(nm log n) Time Primal-Dual Algorithm. Unpublished Manuscript, Department of Computer Science, Princeton University, Princeton, NJ.

Zemel, E. 1982. A Parallel Randomized Algorithm for Selection with Applications. WP # 761–82. Israel Institute for Business Research, Tel Aviv University, Israel.

Zemel, E. 1987. A Linear Time Randomizing Algorithm for Searching Ranked Functions. *Algorithmica* 2, 81–90.