September 1986 (revised January1987) **Report LIDS - P - 1606**
To appear in:
Proceedings of IFAC '87, Munich, Germany, July 1987, Pergamon Press, Oxford, England

Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems*

by
Dimitri P. Bertsekas and Jonathan Eckstein
Laboratory for Information and Decision Systems
and Operations Research Center
Massachusetts Institute of Technology
Cambridge, Mass. 02139

Abstract

We propose a new, massively parallelizable algorithm for the minimum cost network flow problem. We formulate a dual problem which is unconstrained, piecewise linear, and involves a dual variable for each node. Our algorithm solves the dual by an approach resembling a Gauss-Seidel relaxation method. At each iteration a single node is chosen, and its dual variable and its incident arc flows are changed in an attempt to improve the dual cost. In a distributed implementation, each node is a processor adjusting its own dual variable on the basis of local information communicated by adjacent nodes. We show finite convergence of a totally asynchronous (chaotic), distributed version of the algorithm whereby some processors compute faster than others, some processors communicate faster than others, and there can be arbitrarily large communication delays. A scaled version of the algorithm is shown to have $O(N^3\log(NC))$ sequential complexity, where N is the number of nodes and C the maximum absolute value of the arc costs. This bound is competitive with the best bounds for existing sequential methods. These methods, however, are not well suited for distributed implementation.

Key Words: Network flows; relaxation; distributed algorithms; complexity; asynchronous algorithms.

*Supported by Grant NSF-ECS-8217668. Thanks are due to Paul Tseng for his helpful comments.

1. Introduction

Consider a directed graph with set of nodes N and set of arcs A. Each arc (i, j) has associated with it an integer a_{ij} referred to as the cost coeficient of (i, j). Let f_{ij} be the flow of the arc (i, j), and consider the classical min cost flow problem ([17], Ch.7)

$$\text{minimize } \Sigma_{(i,j)\in A}\ a_{ij}f_{ij} \qquad \text{(MCF)}$$

subject to

$$\Sigma_{(j,i)\in A}\ f_{ji} - \Sigma_{(i,j)\in A}\ f_{ij} = s_i \quad \forall\, i \in N \qquad (1)$$

$$b_{ij} \le f_{ij} \le c_{ij} \quad \forall\, (i, j) \in A\,, \qquad (2)$$

where a_{ij}, b_{ij}, c_{ij}, and s_i, are given *integers*. We assume throughout that there exists at most one arc in each direction between any pair of nodes, but this assumption is made for notational convenience and can be easily dispensed with. The numbers of nodes and arcs are denoted N and A respectively.

We first formulate a dual problem associated with (MCF) by associating a Lagrange multiplier p_i with the ith conservation of flow constraint (1). By denoting by f and p the vectors with elements f_{ij}, $(i, j) \in A$, and p_i, $i \in N$ respectively, we can write the corresponding Lagrangian function as

$$L(f, p) = \Sigma_{(i,j)\in A}\,(a_{ij} + p_j - p_i)\,f_{ij} - \Sigma_{i\in N}\,s_ip_i \qquad (3)$$

The dual function value q(p) at a vector p is obtained by minimizing L(f, p) over all f satisfying the capacity constraints (2). This leads to the dual problem

$$\text{maximize } q(p) \qquad (4)$$

subject to no constraint on p,

with the dual functional q given by

$$q(p) = \min_f\{L(f, p) \mid b_{ij} \le f_{ij} \le c_{ij},\ (i, j) \in A\}$$

$$= \Sigma_{(i,j)\in A}\ q_{ij}(p_i - p_j)\ - \Sigma_{i\in N}\ s_ip_i \qquad (5a)$$

where

$$q_{ij}(p_i-p_j) = \min_{f_{ij}}\{(a_{ij} + p_j - p_i)f_{ij} \mid b_{ij} \le f_{ij} \le c_{ij}\} \qquad (5b)$$

The function q_{ij} is shown in Fig. 1. This formulation of the dual problem is consistent with

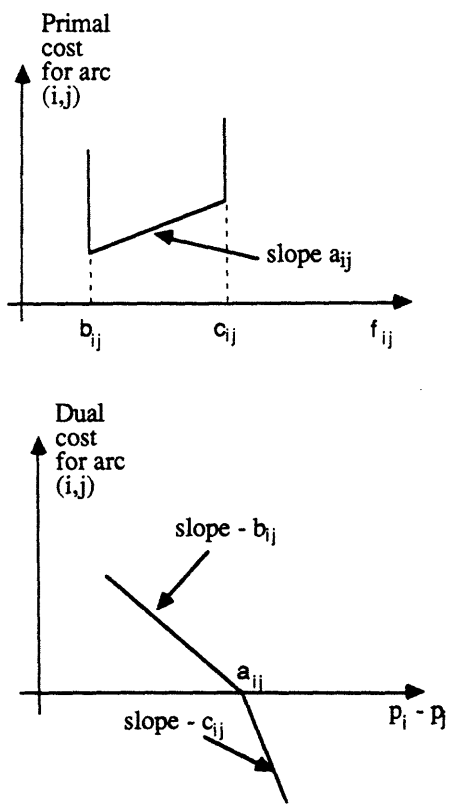

Figure 1: Primal and dual costs for arc (i, j).

classical duality frameworks [18], but can also be obtained via linear programming duality theory [16], [17]. We henceforth refer to (MCF) as the *primal problem*, and note that, based on standard duality results, the optimal primal cost equals the optimal dual cost. The dual variable p_i will be referred to as the *price of node i*.

The form of the dual cost (5) motivates solution by Gauss-Seidel relaxation (or coordinate ascent methods). A number of methods of this type have been developed recently, and have led to remarkably successful computer codes [5], [7], [8], [19]. The idea is to choose a single node i and change its price p_i in a direction of improvement of the dual cost, while keeping the other prices unchanged. Unfortunately there is a fundamental difficulty; the dual cost q is piecewise linear, and the relaxation idea may encounter difficulty at some "corner points" as illustrated in Fig. 2. The problem is that there are nonoptimal price vectors at which the dual cost cannot be improved by changing any single node price.

The main idea of this paper is to allow single node price changes even if these worsen the dual cost. The rationale is that if the cost deterioration is small, then the algorithm can eventually approach the optimal solution (see Fig. 3). Indeed we will show that not only this is so, but in fact an *exact* solution of the problem can be obtained in a *finite* (indeed, *a priori* bounded) number of iterations owing to the integer nature of the problem data. As will be discussed in Section 3, a key idea is that each price change improves the dual function of a perturbed problem where some of the arc cost coefficients are modified by a "small" amount ε. Implementation of this idea is based on the notion of ε-complementary slackness first used in [4], and more formally introduced in [7] and [8]. The corresponding algorithm was first proposed in Sept. 1986 and published in [3]. Special cases of this algorithm for the assignment problem date to 1979; see [4], [6].

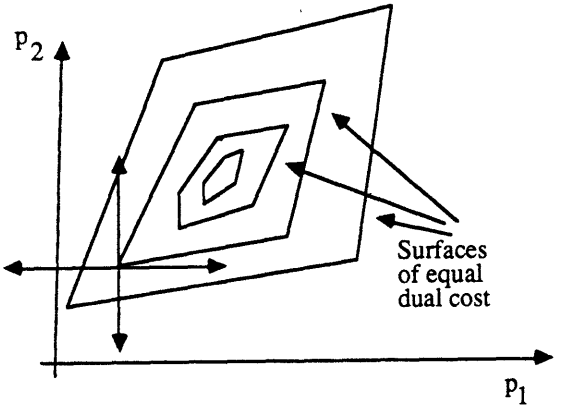

Figure 2: At the indicated point, it is impossible to improve the cost by changing any *single* price.

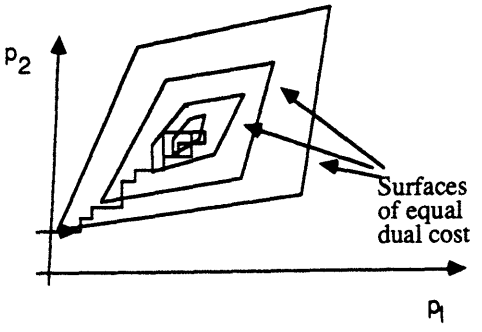

Figure 3: By making small changes in the coordinate directions, it is possible to approach the optimal solution even if each step does not result in a dual cost improvement. The method eventually stays in a small neighborhood of the optimal solution.

The first interesting aspect of the algorithm is that it admits massive parallelization; indeed it is the first such algorithm for the min cost flow problem. We envision here the possibility of a separate processor assigned to each node. The processor changes the node's price on the basis of price and flow information received from adjacent nodes. By allowing processors corresponding to nonadjacent nodes to iterate simultaneously, one may obtain a synchronous distributed algorithm that is mathematically equivalent to a sequential algorithm. The algorithm can also be made to work correctly in a chaotically asynchronous environment. By this we mean that processors can update their own prices as fast as they please with whatever information they might have available at the time, regardless of the speed of computation at other processors, the frequency of communication of information from other processors, and the magnitude of the communication delays (see Section 6). Adjacent nodes, in particular, may compute simultaneously with different and arbitrarily out-of-date local information. Furthermore the communication channels are not assumed to preserve the order of transmission, so a node may at any time receive information that is older than what it received earlier. This formulation involves a far more flexible type of asynchronism than can be obtained with the use of synchronizers [1]. Algorithmic convergence is often difficult to establish for chaotic models, but powerful results are now available to aid in this process [2], [9]-[12]. The algorithm given here is more complex than a related algorithm for strictly convex arc costs [9], and requires a novel method of convergence proof.

The second interesting aspect of our relaxation algorithm is that even without the benefit of parallelism it is competitive with the alternatives for some classes of problems. An example is the max-flow problem for which an $O(N^3)$ worst case complexity is proved (see Section 5). When applied to this problem in a special way the algorithm resembles a max-flow algorithm developed (without reference to duality or ε-complementary slackness) in [14]. The latter algorithm is somewhat more complicated than ours in that it requires two separate phases to obtain an optimal primal solution; ours requires only one, and furthermore admits a simpler proof of the complexity bound. Our algorithm also allows arbitrary initial node prices (this is useful in sensitivity studies involving reoptimization); in [14] the initial prices are in effect required to be zero. For general min cost flow problems the relaxation algorithm is pseudopolynomial, being sensitive to the arc cost coefficients.

An important contribution of the present paper is a new version of the main min cost flow algorithm that uses cost scaling. The worst case complexity of the scaled version is $O(N^3\log(NC))$, where C is the largest absolute value of the arc costs. This bound is better than any bound that has been obtained up to now for min-cost flow problems using simply implementable algorithms that do not employ complex data structures. It is likely that the bound can be somewhat improved for sparse networks through the use of sophisticated data structures such as dynamic trees, but we have not pursued the matter further. A result of this type has been claimed in [15] for a scheme that embeds our main algorithm in a different scaling procedure. The practical performance of the algorithms of this paper is currently being evaluated.

2. Optimality Conditions and ε-Complementary Slackness

The necessary and sufficient conditions for a pair (f, p) to be an optimal primal and dual solution pair are primal feasibility and complementary slackness. To state these conditions, we first introduce some terminology. For any price vector p we say that an arc (i, j) is

Inactive if $p_i < a_{ij} + p_j$ (6a)

Balanced if $p_i = a_{ij} + p_j$ (6b)

Active if $p_i > a_{ij} + p_j$ (6c)

For any flow vector f and node i the scalar

$$g_i = \Sigma_{(j,i)\in A} f_{ji} - \Sigma_{(i,j)\in A} f_{ij} - s_i \qquad (7)$$

is called the *surplus* of node i. It represents the difference of total flow imported and total flow exported by the node.

The primal feasibility and complementary slackness conditions for a vector pair (f, p) are:

$$g_i = 0, \quad \text{for all } i \in N \tag{8a}$$

$$f_{ij} = b_{ij}, \quad \text{for all inactive arcs } (i, j) \tag{8b}$$

$$b_{ij} \le f_{ij} \le c_{ij}, \quad \text{for all balanced arcs } (i, j) \tag{8c}$$

$$f_{ij} = c_{ij}, \quad \text{for all active arcs } (i, j). \tag{8d}$$

For any price vector p, and $\varepsilon \ge 0$ we say that an arc (i, j) is

$$\varepsilon\text{-Inactive} \quad \text{if } p_i < a_{ij} + p_j - \varepsilon \tag{9a}$$

$$\varepsilon^-\text{-Balanced} \quad \text{if } p_i = a_{ij} + p_j - \varepsilon \tag{9b}$$

$$\varepsilon\text{-Balanced} \quad \text{if } a_{ij} + p_j - \varepsilon \le p_i \le a_{ij} + p_j + \varepsilon \tag{9c}$$

$$\varepsilon^+\text{-Balanced} \quad \text{if } p_i = a_{ij} + p_j + \varepsilon \tag{9d}$$

$$\varepsilon\text{-Active} \quad \text{if } p_i > a_{ij} + p_j + \varepsilon \tag{9e}$$

An ε^-- balanced arc for which $b_{ij} < f_{ij}$ is called ε^-- *unblocked*; an ε^+- balanced arc for which $f_{ij} < c_{ij}$ is called ε^+- *unblocked*. Given $\varepsilon \ge 0$ we say that a vector pair (f, p) satisfies *ε-complementary slackness* (ε-CS for short) if for each arc (i, j)

$$f_{ij} = b_{ij} \quad \text{if} \quad (i, j) \text{ is } \varepsilon\text{-inactive} \tag{10a}$$

$$b_{ij} \le f_{ij} \le c_{ij} \quad \text{if} \quad (i, j) \text{ is } \varepsilon\text{-balanced} \tag{10b}$$

$$f_{ij} = c_{ij} \quad \text{if} \quad (i, j) \text{ is } \varepsilon\text{-active} \tag{10c}$$

An equivalent statement of the ε-CS conditions is that the flows f_{ij} satisfy the capacity constraints (2), and that

$$f_{ij} < c_{ij} \quad \Rightarrow \quad p_i - p_j \le a_{ij} + \varepsilon \tag{10d}$$

$$b_{ij} < f_{ij} \quad \Rightarrow \quad p_i - p_j \ge a_{ij} - \varepsilon \tag{10e}$$

The algorithm described in the next section maintains at all times a price vector p and an integer flow vector f satisfying ε-CS. It terminates when the flow vector f satisfies the primal feasibility condition $g_i = 0$ for all $i \in N$. A key fact is that if ε is sufficiently small then the final flow vector f is optimal.

Proposition 1: If $\varepsilon < 1/N$ and the flow vector f together with the price vector p satisfy ε-CS and primal feasibility ($g_i = 0$ for all $i \in N$), then f is optimal for (MCF).

Proof: If f is not optimal then there must exist a simple directed cycle along which flow can be increased while the primal cost is improved. Let Y^+ and Y^- be the sets of arcs of the cycle that are positively and negatively oriented, respectively. Then we must have

$$\Sigma_{(i,j)\in Y^+} a_{ij} - \Sigma_{(i,j)\in Y^-} a_{ij} < 0 \tag{11}$$

$$f_{ij} < c_{ij} \quad \text{for } (i, j) \in Y^+$$

$$b_{ij} < f_{ij} \quad \text{for } (i, j) \in Y^-.$$

By using (10d-e), we have

$$p_i \le p_j + a_{ij} + \varepsilon \quad \text{for } (i, j) \in Y^+$$

$$p_j \le p_i - a_{ij} + \varepsilon \quad \text{for } (i, j) \in Y^-,$$

which by adding and using the hypothesis $\varepsilon < 1/N$ yields

$$\Sigma_{(i,j)\in Y^+} a_{ij} - \Sigma_{(i,j)\in Y^-} a_{ij} \ge -N\varepsilon > -1.$$

Since the a_{ij} are integer, we obtain a contradiction of (11). **Q. E. D.**

3. Sequential Versions of the Algorithm

The pure form of the algorithm, first proposed in [3], uses a fixed value of $\varepsilon > 0$, and starts with any pair (f, p) satisfying ε-CS. A possible starting procedure is to arbitrarily choose the vector p, and to set $f_{ij} = b_{ij}$ if (i, j) is inactive or balanced, and $f_{ij} = c_{ij}$ otherwise. The flow vector f is taken integer initially, and the algorithm preserves the integrality of f throughout. At the start of each iteration a node i with positive surplus g_i is chosen. (If all nodes have zero surplus the algorithm terminates; then f is primal feasible and, together with p, satisfies ε-CS.) At the end of the iteration, the surplus g_i is driven to zero, while another pair (f, p) satisfying ε-CS is obtained. During an iteration, all node prices stay unchanged except possibly for the price of the chosen node i. Similarly all arc flows stay unchanged except for the flows of some of the arcs incident to node i. As a result of these flow changes, the surplus of some of the nodes adjacent to i is increased.

Positive Surplus Node Iteration (or Up Iteration):

Let (f, p) satisfy ε-CS, and let i be a node with $g_i > 0$.

Step 1: (Scan incident arc) Select a node j such that (i, j) is an ε^+-unblocked arc and go to Step 2, or select a node j such that (j, i) is an ε^--unblocked arc and go to Step 3. If no such node can be found go to Step 4.

Step 2: (Decrease surplus by increasing f_{ij}) Set

$$f_{ij} := f_{ij} + \delta$$

$$g_i := g_i - \delta, \qquad g_j := g_j + \delta$$

where $\delta = \min\{g_i, c_{ij} - f_{ij}\}$. If $g_i = 0$ stop; else go to Step 1.

Step 3: (Decrease surplus by reducing f_{ji}) Set

$$f_{ji} := f_{ji} - \delta$$

$$g_i := g_i - \delta, \qquad g_j := g_j + \delta$$

where $\delta = \min\{g_i, f_{ji} - b_{ji}\}$. If $g_i = 0$ stop; else go to Step 1.

Step 4: (Increase price of node i) Set

$$p_i := \min\{\{p_j + a_{ij} + \varepsilon \mid (i, j) \in A \text{ and } f_{ij} < c_{ij}\},$$
$$\{p_j - a_{ji} + \varepsilon \mid (j, i) \in A \text{ and } b_{ji} < f_{ji}\}\} \tag{12}$$

Go to Step 1. (*Note:* If the set over which the minimum in (12) is taken is empty, the problem is infeasible and the algorithm terminates - see the comments below.)

To see that (12) leads to a price increase note that when Step 4 is entered we have

$$f_{ij} = c_{ij} \text{ for all } (i, j) \text{ such that } p_i \ge p_j + a_{ij} + \varepsilon$$

$$b_{ji} = f_{ji} \text{ for all } (j, i) \text{ such that } p_i \ge p_j - a_{ji} + \varepsilon.$$

Therefore, when Step 4 is entered we have

$$p_i < \min\{p_j + a_{ij} + \varepsilon \mid (i, j) \in A \text{ and } f_{ij} < c_{ij}\}$$

$$p_i < \min\{p_j - a_{ji} + \varepsilon \mid (j, i) \in A \text{ and } b_{ji} < f_{ji}\}$$

It follows that p_i must be increased via (12). Another issue is that, for Step 4 to be well defined, we must exclude the case where $f_{ij} = c_{ij}$ for all (i, j) outgoing from i, and $b_{ji} = f_{ji}$ for all (j, i) incoming to i. In this case, maximal flow is going out of i while minimal flow is coming in, and since we have $g_i > 0$ when Step 4 is executed, it follows that the problem must be infeasible. Therefore, when the minimum in Step 4 is taken over an empty set, we can terminate the algorithm with an indication that the problem is infeasible.

It is possible to show that, at the end of the iteration, the price of the node i equals ε plus the smallest value that maximizes the dual cost with respect to p_i with all other prices kept fixed. This can be done by calculating the directional derivatives of the dual cost along the coordinate directions

using (5) (see Figure 4). We thus obtain the interpretation of the algorithm as a relaxation method, although "approximate relaxation" may be a better term.

Figure 4 illustrates an up iteration. Each time Step 2 or 3 is executed, flow is "pushed" away from i along an ε^+-unblocked or an ε^--unblocked arc respectively. If no more flow can be pushed, the price of i is increased in Step 4.

Note that a symmetric iteration may be used for nodes with negative surplus (called a *down iteration*). One can construct an example (essentially the same example as the one of [19], Appendix C) showing that the algorithm may not terminate if up and down iterations are mixed arbitrarily. It is therefore necessary to impose some assumptions either on the problem structure or on the method by which up and down iterations are interleaved. We henceforth assume that the algorithm consists of up iterations only.

Proposition 2: If problem (MCF) is feasible, the algorithm terminates with (f, p) satisfying ε-CS, and with f being integer and primal feasible.

Proof: The following facts can be verified based on the construction of the up iteration:

1) The integrality of f and the ε-CS property of (f, p) are preserved throughout the algorithm.

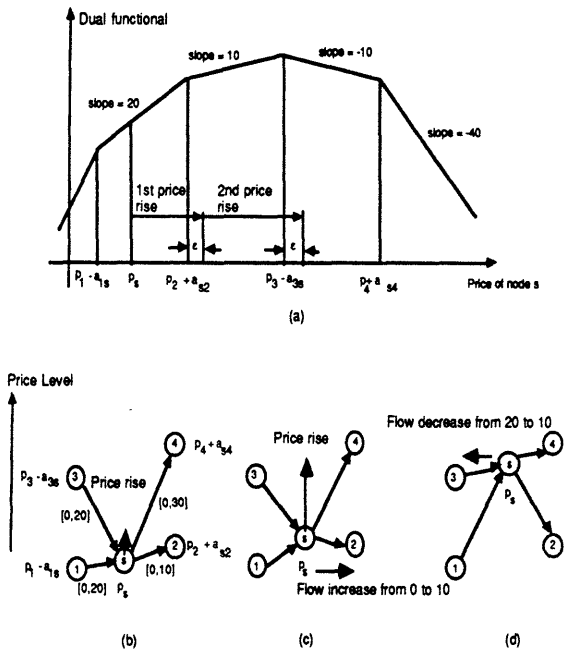

Figure 4: Illustration of an up iteration involving a single node s with four incident arcs (1,s), (3,s), (s,2), and (s,4), with feasible arc flow ranges [1,20], [0,20], [0,10], and [0,30], respectively.
(a) Form of the dual functional along p_s for given values of p_1, p_2, p_3, and p_4. The breakpoints correspond to the levels of p_s for which the corresponding arcs become balanced. For values of p_s between two successive breakpoints there are no balanced arcs incident to node s. The corresponding slope of the dual cost is equal to the surplus g_s resulting when all active arc flows are set to their upper bounds and all inactive arc flows are set to their lower bounds; compare with (5).
(b) Illustration of a price rise of p_s from a value between the first two breakpoints to a value ε above the breakpoint at which (s,2) becomes balanced (Step 4).
(c) Price rise of p_s to a value ε above the breakpoint at which arc (3,s) becomes balanced. When this is done, arc (s,2) has changed from ε^+-balanced to ε-active, and its flow has increased from 0 to 10, maintaining ε-CS.
(d) Step 3 of the algorithm reduces the flow of arc (3,s) from 20 to 10, driving the surplus of node s to zero.

2) The prices of all nodes are monotonically nondecreasing during the algorithm.

3) Once a node gets nonnegative surplus, its surplus stays nonnegative thereafter. (This follows from the fact that an up iteration at some node i cannot drive the surplus of i below zero, and can only increase the surplus of adjacent nodes.)

4) If at some time a node has negative surplus it must never have been iterated upon up to that time, and therefore its price must be equal to its initial price. (This a consequence of 3) above and the fact that only nodes with positive surplus are iterated upon by up iterations.)

Based on 2) above there are two possibilities; either a) the prices of a nonempty subset N^∞ of N diverge to $+\infty$, or else b) the prices of all nodes in N stay bounded from above.

Suppose that case a) holds. Then, since N^∞ is nonempty, it follows that the algorithm never terminates, implying that at all times there must exist a node with negative surplus which, by 4) above, must have a constant price. It follows that N^∞ is a strict subset of N. To preserve ε-CS, we must have after a sufficient number of iterations

$$f_{ij} = c_{ij} \quad \text{for all } (i, j) \in A \text{ with } i \in N^\infty, j \notin N^\infty$$

$$f_{ji} = b_{ji} \quad \text{for all } (j, i) \in A \text{ with } i \in N^\infty, j \notin N^\infty$$

while the sum of surpluses of the nodes in N^∞ is positive. This means that even with as much flow as arc capacities allow coming out of N^∞ to nodes $j \notin N^\infty$, and as little flow as arc capacities allow coming into N^∞ from nodes $j \notin N^\infty$, the total surplus $\Sigma\{g_i \mid i \in N^\infty\}$ of nodes in N^∞ is positive. It follows that there is no feasible flow vector, contradicting the hypothesis. Therefore case b) holds (all prices of nodes in N stay bounded).

We now show by contradiction that the algorithm terminates. If that were not so, then there must exist a node $i \in N$ at which an infinite number of iterations are executed. There must also exist an adjacent ε^--balanced arc (j, i), or ε^+-balanced arc (i, j) whose flow is decreased or increased (respectively) by an integer amount during an infinite number of iterations. For this to happen, the flow of (j, i) or (i, j) must be increased or decreased (respectively) an infinite number of times due to iterations at the adjacent node j. This implies that the arc (j, i) or (i, j) must become ε^+-balanced or ε^--balanced from ε^--balanced or ε^+-balanced (respectively) an infinite number of times. For this to happen, the price of the adjacent node j must be increased by at least 2ε an infinite number of times. It follows that $p_j \to \infty$ which contradicts the boundedness of all node prices shown earlier. Therefore the algorithm must terminate. **Q.E.D.**

Note that Proposition 2 holds for all $\varepsilon > 0$. If $\varepsilon < 1/N$, however, we see, by combining Propositions 1 and 2, that the algorithm terminates with an *optimal* flow vector. Note also that the integrality of a_{ij} was not needed for the proof of Proposition 2, while the integrality of b_{ij}, c_{ij}, s_i, and the starting flow vector were only needed to establish that the flow change increments are bounded from below during the course of the algorithm. The integrality assumptions are essential, however, for the complexity analysis of the next section.

Proposition 2 applies without modification to the variation of the algorithm where up iterations are not necessarily carried to completion. In this variation, it is permissible to execute only *partial* up iterations, in which the algorithm may select a new node for iteration immediately following the completion of any Step 2, 3, or 4, even if the current node surplus is not yet zero.

In other types of relaxation methods for network flow problems [5], [7], [8], a price change never degrades the dual cost. This is not true for the price changes effected in Step 4 of the up iteration. However, there is still an interesting interpretation of Step 4 as a dual cost improvement. It can be shown by calculating the directional derivative of the dual cost in the direction of any single price change (or by using results of [5], [7]), that price changes in Step 4 yield a dual cost improvement of a *perturbed* problem obtained after the cost

coefficients of some ε-balanced arcs are changed by ε to make them balanced. In effect, the algorithm improves, with each price change, a slightly perturbed dual cost and ends up with a slightly suboptimal dual solution. The corresponding primal solution, however, is optimal thanks to the rounding introduced by the integer nature of the problem data.

A final issue has to do with detection of infeasibility. By using the argument of the proof of Proposition 2, it follows that for an infeasible problem the prices of some nodes diverge to ∞. In Section 5 we derive a precomputable upper bound for the prices when the problem is feasible [cf. (26)]. Once this bound is exceeded, we know that the problem is infeasible.

The Scaled Version of the Algorithm

The running time of the method just described is sensitive to the arc cost coefficients, as illustrated in Figure 5. It is therefore natural to consider cost scaling procedures [20] - [23] in which one solves a sequence of approximations to the original problem ("subproblems"), gradually increasing the accuracy of the input data.

Consider the problem (SMCF) obtained from (MCF) by multiplying all arc costs by N+1, that is, the problem with arc cost coefficients

$$a_{ij}' = (N+1)a_{ij} \qquad \text{for all } (i, j).$$

If the pair (f ', p') satisfies 1-CS (namely ε-CS with ε = 1) with respect to (SMCF), then clearly the pair

$$(f, p) = (f', p'/(N+1))$$

satisfies $(N+1)^{-1}$-CS with respect to (MCF), and hence f ' is optimal for (MCF) by Proposition 1. In the scaled algorithm, we seek a 1-CS solution to (SMCF).

Let

$$C = \max_{(i,j)\in A} |a_{ij}| \qquad (13)$$

$$M = \lfloor \log_2 (N+1)C \rfloor + 1 = O(\log(NC)) . \qquad (14)$$

In the scaled algorithm we solve M subroblems. The mth subproblem is a minimum cost flow problem where the cost coefficient of each arc (i, j) is

$$a_{ij}(m) = \text{Trunc}(a_{ij}' / (2^{M-m})), \qquad (15)$$

where Trunc(•) denotes integer rounding in the direction of zero, that is, down for positive and up for negative numbers. Note that $a_{ij}(m)$ is the integer consisting of the m most significant bits in the M-bit binary representation of a_{ij}'. In particular each $a_{ij}(1)$ is 0, +1, or -1, while $a_{ij}(m+1)$ is obtained by doubling $a_{ij}(m)$ and adding (subtracting) one if the (m+1)st bit of the M-bit representation of a_{ij}' is a one and a_{ij}' is positive (negative). Note also that

$$a_{ij}(M) = a_{ij}',$$

so the last problem of the sequence is (SMCF).

For each subproblem, we apply the unscaled version of the algorithm with ε = 1, yielding upon termination a pair ($f^t(m)$, $p^t(m)$) satisfying 1-CS with respect to the cost coefficients $a_{ij}(m)$.

The starting price vector for the (m+1)st problem (m = 1, 2, . . . , M-1) is

$$p^0(m+1) = 2p^t(m). \qquad (16)$$

Doubling $p^t(m)$ as above roughly maintains complementary slackness since $a_{ij}(m)$ is roughly doubled when passing to the (m+1)st problem. Indeed it can be seen that every arc that was 1 - balanced (1 - active, 1 - inactive) upon termination of the algorithm for the mth problem will be 3 - balanced (1 - active, 1 - inactive, respectively) at the start of the (m+1)st problem.

The starting flow vector $f^0(m+1)$ for the (m+1)st problem

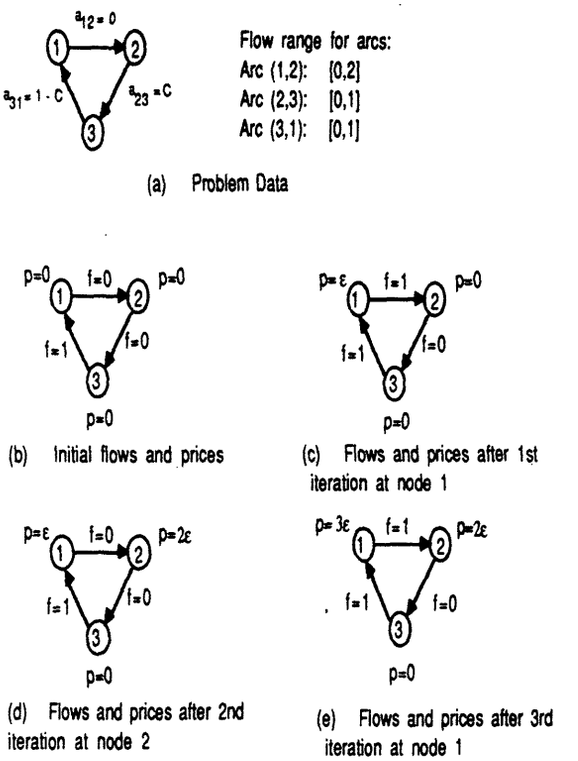

Figure 5: Example showing that the computation required by the pure form of the algorithm can be proportional to the cost-dependent factor C. Here, up iterations at node 1 alternate with up iterations at node 2 until the time when p_1 rises to the level C-1+ε and arc (3,1) becomes ε^--balanced, so that a unit of flow can be pushed back along that arc. At this time, the optimal solution is obtained. Since prices rise by increments of no more than 2ε, the number of up iterations is $\Omega(C/\varepsilon)$.

is obtained from $f^t(m)$ by setting

$$f^0_{ij}(m+1) = f^t_{ij}(m) \qquad \text{for all balanced arcs } (i, j),$$

$$f^0_{ij}(m+1) = c_{ij} \qquad \text{for all active arcs } (i, j),$$

$$f^0_{ij}(m+1) = b_{ij} \qquad \text{for all inactive arcs } (i, j).$$

Note that this implies that there will be no 1^+ - unblocked and 1^- - unblocked arcs initially for the (m+1)st problem, and this turns out to be significant for the complexity analysis of the next section.

4. Complexity Analysis

We derive worst-case complexity bounds for two algorithms. The first is the unscaled algorithm where the cost coefficients are fixed at their given values. We refer to this as the *pure form of the algorithm* to distinguish it from the *scaled form of the algorithm*, which is the pure form embedded in the scaling procedure just described. Our analysis assumes that:

(a) There exists at least one feasible solution of (MCF).

(b) All arc cost coefficients are integer multiples of ε.

(c) All starting prices are integer multiples of ε, all starting flows are integer, and together they satisfy ε-CS. Furthermore, initially there are no ε^+-unblocked or ε^--unblocked arcs. (Note that these conditions hold for all subproblems of the scaled form of the algorithm, provided that they hold for the original pair ($f^0(1)$, $p^0(1)$).)

(d) For the scaled algorithm only, the starting price differentials for the first subproblem $(p^0_i(1) - p^0_j(1))$ are O(1) for all arcs (i, j).

(e) A special procedure -- to be described shortly -- is used to select the nodes on which up iterations are performed.

To simplify the subsequent presentation we introduce some notation and terminology. For any path H we denote by s(H) and t(H) the start and end nodes of H, respectively, and by H^+ and H^- the sets of arcs that are positively and negatively oriented, respectively, as the path is traversed in the direction from s(H) to t(H). We call a path *simple* if it is not a circuit and has no repeated nodes. For any price vector p and simple path H we define

$$d_H(p) = \max\{0, \Sigma_{(i,j)\in H^+}(p_i - p_j - a_{ij}) - \Sigma_{(i,j)\in H^-}(p_i - p_j - a_{ij})\}$$

$$= \max\{0, p_{s(H)} - p_{t(H)} - \Sigma_{(i,j)\in H^+} a_{ij} + \Sigma_{(i,j)\in H^-} a_{ij}\} \quad (17)$$

Note that the second term in the maximum above may be viewed as a "reduced cost length of H" being the sum of the reduced costs $(p_i - p_j - a_{ij})$ over all arcs $(i, j)\in H^+$ minus the sum of $(p_i - p_j - a_{ij})$ over all arcs $(i, j)\in H^-$. For any flow vector f satisfying the capacity constraints (2) we say that a simple path H is *unblocked with respect to f* if we have $f_{ij} < c_{ij}$ for all arcs $(i, j) \in H^+$, and we have $f_{ij} > b_{ij}$ for all arcs $(i, j) \in H^-$. In words, H is unblocked with respect to f if there is margin for sending positive flow along H (in addition to f) from s(H) to t(H) without violating the capacity constraints.

For any price vector p, and flow vector f satisfying both the conservation of flow and the capacity constraints (1) and (2) denote

$$D(p, f) = \max\{d_H(p) \mid H \text{ is a simple unblocked path with respect to f}\}. \quad (18)$$

In the exceptional case where there is no simple unblocked path with respect to f we define D(p, f) = 0. In this case we must have $b_{ij} = c_{ij}$ for all (i, j) since any arc (i, j) with $b_{ij} < c_{ij}$ gives rise to a one-arc unblocked path with respect to f. Let

$$\beta(p) = \min\{D(p, f) \mid f \text{ satisfies constraints (1) and (2)}\} \quad (19)$$

Since, for a given p, there are only a finite number of values that D(p, f) can take it follows that the minimum in (19) is actually attained by some f. The following lemma shows that β(p) provides a measure of suboptimality of the price vector p. The computational complexity estimate to be obtained shortly is proportional to $\beta(p^0)$, where p^0 is the initial price vector.

Lemma 1: (a) If there exists a flow vector f satisfying the constraints (1), (2), and satisfying γ-CS together with p for some $\gamma \geq 0$ then

$$0 \leq \beta(p) \leq (N-1)\gamma. \quad (20)$$

(b) p is dual optimal if and only if β(p) = 0.

Proof: (a) For each simple path H which is unblocked with respect to f and has |H| arcs we have, by adding the γ-CS conditions given by (10d-e) along H and using (17),

$$d_H(p) \leq |H|\gamma \leq (N-1)\gamma,$$

and the result follows from (18) and (19).

(b) If p is optimal then it satisfies complementary slackness together with some primal optimal vector f, so from (20) (with γ = 0) we obtain β(p) = 0. Conversely if β(p) = 0, then from (19) we see that there must exist a primal feasible f such that D(p, f) = 0. Hence $d_H(p) = 0$ for all unblocked simple paths H with respect to f. Applying this fact to single-arc paths H and using the definition (17) we obtain that f together with p satisfy complementary slackness. It follows that p and f satisfy all the optimality conditions (8), and p is optimal. **Q. E. D.**

In order to get the sharpest possible complexity bounds we need a restriction in the way the algorithms are operated. We introduce an order for choosing nodes in iterations. A *cycle* is a set of iterations whereby all nodes are chosen once in a given order, and an up iteration is executed at each node having positive surplus at the time its turn comes. The order in which nodes are taken up may change from one cycle to the next. This node order is maintained in a linked list that is traversed from its first to last element in each cycle. Each time a node i changes price as a result of its up iteration within a cycle, node i is removed from its present list position, and is placed in the first list position. (This does not change the order in which the remaining nodes are taken up in the current cycle; only the order for the subsequent cycle is affected.) The initial list is arbitrary.

In the next section we will prove the following proposition:

Proposition 3: Under Assumptions (a) - (e) the complexity of the pure form of the algorithm operated in cycles as described above is bounded by $O(N^2(\beta(p^0)/\varepsilon + N))$, where p^0 is the initial price vector.

An upper bound for $\beta(p^0)$ is given by $(N-1)C + p^+ - p^-$ where

$$p^+ = \max\{p_i^0 \mid i \in N\} \quad (21)$$

$$p^- = \min\{p_i^0 \mid i \in N\}. \quad (22)$$

Assuming that $p^+ - p^- = O(1)$, we obtain the complexity bound $O(N^3C)$. The algorithm is indeed sensitive to C as shown in the example of Figure 5. The example of Figure 6 demonstrates that assumption (c) concerning the initial conditions is necessary, for otherwise a factor relating to the arc flow bounds must be included in the complexity estimate.

For classes of problems with special structure a better estimate of $\beta(p^0)$ may be possible. As an example, consider the max-flow problem formulation shown in Figure 7. The artificial arc (t, s) connecting the sink t with the source s has cost coefficient -1, and flow bounds $b_{ts} = 0$ and $c_{ts} = \Sigma_i c_{si}$. We assume that $a_{ij} = 0$ and $b_{ij} = 0 < c_{ij}$ for all other arcs (i, j), and that $s_i = 0$ for all i. We apply the relaxation algorithm with initial prices and arc flows satisfying ε-CS, where $\varepsilon = 1/(N+1)$ and $p^+ - p^- = O(1)$. Then we obtain $d_H(p^0) = O(1)$ for all paths H, $\beta(p^0) = O(1)$, and an $O(N^3)$ complexity bound.

Based on Proposition 3 and the fact $\varepsilon = 1$, the complexity of the scaled form of the algorithm is $O(N^2B + N^3M)$ where

$$B = \Sigma_{m=1,\ldots,M} \beta_m(p^0(m)), \quad (23)$$

and $\beta_m(\cdot)$ is defined by (17) - (19) but with the modified cost coefficients $a_{ij}(m)$ replacing a_{ij} in (17). We will show that

$$\beta_m(p^0(m)) = O(N) \quad \text{for all } m$$

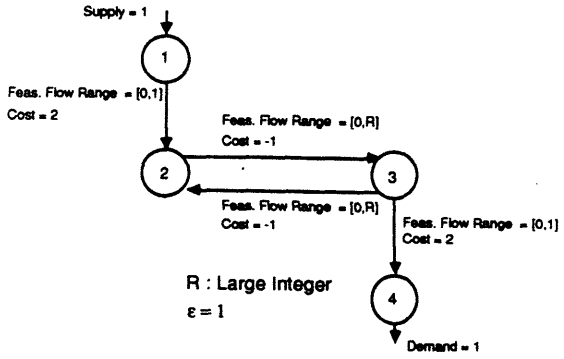

Figure 6: Example showing that assumption (c) on the initial conditions is essential for polynomial complexity. Initially, we choose f=0, p=0, which do satisfy 1-CS, but not (c). The algorithm will push one unit of flow R times around the cycle 2-3-2, implying Ω(R) complexity.

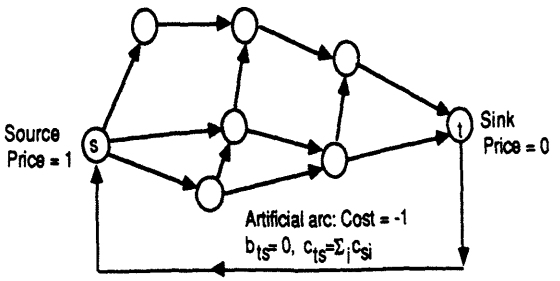

Figure 7: Formulation of the max-flow problem.

thereby obtaining an $O(N^3\log(NC))$ complexity bound. We note that the same bound has been claimed for a scheme that embeds the pure form of the algorithm in a different scaling procedure [15]. However this bound has not been proved thus far, and its correctness is unclear.

Proposition 4: Under Assumptions (a) - (e), the complexity of the scaled form of the algorithm, operated in cycles as described above, is bounded by $O(N^3\log(NC))$.

Proof: Since initially we have

$$p_i - p_j = O(1), \quad a_{ij}(1) = O(1) \quad \text{for all arcs } (i, j),$$

we obtain $d_H(p^0(1)) = O(N)$ for all H, and $\beta_1(p^0(1)) = O(N)$. We also have that the final flow vector $f^t(m)$ obtained from the m-th problem satisfies constraints (1) and (2), and together with $p^0(m+1)$ it may be easily seen to satisfy 3-CS. It follows from Lemma 1(a) that

$$\beta_m(p^0(m+1)) \le 3(N-1) = O(N)$$

and the result is obtained from (23) as discussed above. **Q. E. D.**

5. Proof of Proposition 3

To economize on notation, we write β in place of $\beta(p^0)$. We first show the following:

Lemma 2: Under the conditions of Proposition 3, the number of price increases at each node is $O(\beta/\varepsilon + N)$.

Proof: Let f^0 be a flow vector attaining the minimum in the definition (19) of $\beta(p^0)$. To explain the main argument better we assume that f^0 is the zero vector. This can be done without loss of generality because we can transform the problem by replacing c_{ij}, b_{ij}, f_{ij} and s_i by $c_{ij} - f_{ij}^0$, $b_{ij} - f_{ij}^0$, $f_{ij} - f_{ij}^0$ and 0 respectively. The transformation does not change the surplus of any node, and does not change the sequence of prices or flow increments generated by the algorithm. Let (f, p) be a vector pair generated by the algorithm. If $g_t > 0$ for some node t, there must exist a node s with $g_s < 0$ and a simple path H with s(H) = s, t(H) = t, and such that $f_{ij} > 0$ for all $(i, j) \in H^+$ and $f_{ij} < 0$ for all $(i, j) \in H^-$. (This follows from the Conformal Realization Theorem, [18], p. 104. It can also be shown quickly from first principles: Take $T_0 = \{t\}$, and given T_k, define

$$T_{k+1} = T_k \cup \{j \notin T_k \mid \text{there is a node } i \in T_k,$$
$$\text{and either an arc } (i, j) \text{ such that } f_{ij} < 0,$$
$$\text{or an arc } (j, i) \text{ such that } f_{ji} > 0\}.$$

If none of the negative surplus nodes belongs to any of the sets T_k, then the total surplus of the nodes in $\cup T_k$ is positive, while the forward arcs of the cut separating $\cup T_k$ and its complement have nonnegative flow and the backward arcs have nonpositive flow. This is a contradiction, showing that a node s with $g_s < 0$ and the aforementioned properties can be found.)

The conclusion is that the path H is unblocked with respect to f^0. Hence, from (18) we must have $d_H(p^0) \le D(p^0, f^0) = \beta$, and by using (17),

$$p_s^0 - p_t^0 - \Sigma_{(i,j)\in H^+} a_{ij} + \Sigma_{(i,j)\in H^-} a_{ij} \le \beta. \qquad (24)$$

Also using ε-CS we have $p_i + a_{ij} \le p_j + \varepsilon$ for all $(i, j) \in H^+$ and $p_j \le p_i + a_{ij} + \varepsilon$ for all $(i, j) \in H^-$. By adding these conditions along H we obtain

$$-p_s + p_t + \Sigma_{(i,j)\in H^+} a_{ij} - \Sigma_{(i,j)\in H^-} a_{ij} \le |H|\,\varepsilon \le (N-1)\varepsilon \quad (25)$$

where |H| is the number of arcs of H. We have $p_s^0 = p_s$ since the condition $g_s < 0$ implies that the price of s has not yet changed. Therefore, by adding (24) and (25) we obtain

$$p_t - p_t^0 \le \beta + (N-1)\varepsilon \qquad (26)$$

throughout the algorithm for all nodes t with $g_t > 0$. Since all the starting prices and arc cost coefficients are integer multiples of ε, it follows that the size of a price increase is a positive integer multiple of ε, and we see from (26) that the number of price increases of each node is $O(\beta/\varepsilon + N)$. **Q. E. D.**

Note that the price bound (26) ensures that an infeasible problem instance can be detected by checking whether the total price rise of any node exceeds a known upper bound to $\beta + (N-1)\varepsilon$, such as $(N-1)(C+\varepsilon) + p^+ - p^-$.

We now proceed with the proof of Proposition 3. The dominant computational requirements are:

1) The computation required for price increases in Step 4.

2) The computation required for Steps 2 or 3 for which the flow of the corresponding arc is set to its upper or its lower bound.

3) The computation required for Steps 2 or 3 for which the flow of the corresponding arc is set to a value strictly between its upper and its lower bound.

Since there are $O(\beta/\varepsilon + N)$ price increases for each node, the requirements in 1) above are $O(A(\beta/\varepsilon + N))$ operations. Whenever an arc flow is set to either the upper or the lower bound due to an iteration at one of the end nodes, it takes a price increase of at least 2ε by the opposite end node before the arc flow can change again. Therefore there are $O(\beta/\varepsilon + N)$ steps 2 or 3 per arc for which the flow of the arc is set to its upper or lower bound, and the total requirements for 2) above are $O(A(\beta/\varepsilon + N))$ operations.

There remains to estimate the computational requirements for 3) above. At this point we will use the fact that the algorithm is operated in cycles with the node order in each cycle determined by a linked list that is restructured in the course of the algorithm as described earlier. We will demonstrate an $O(N(\beta/\varepsilon + N))$ estimate for the number of cycles up to termination. Given this, the proof of Proposition 3 can be completed as follows: For each cycle there can be only one arc flow per node set to a value strictly between the upper and lower arc flow bound in Step 2 or 3. Therefore the total number of operations required for these steps [cf. 3) above] is $O(N^2(\beta/\varepsilon + N))$. Adding the computational requirements for 1) and 2) calculated earlier we obtain an $O(N^2(\beta/\varepsilon + N)) + O(A(\beta/\varepsilon + N))$ or $O(N^2(\beta/\varepsilon + N))$ worst case complexity bound.

To show that the number of cycles up to termination is $O(N(\beta/\varepsilon + N))$ we argue as follows: At any given stage in the algorithm consider the subgraph $\mathbf{G^*} = (\mathbf{N}, \mathbf{A^*})$ where an arc (i, j) belongs to $\mathbf{A^*}$ if and only if it is possible to "push" flow from i to j according to the rules of the algorithm in Step 2 or 3. In other words $\mathbf{A^*}$ contains (i, j) whenever $(i, j) \in \mathbf{A}$ and (i, j) is ε^+-unblocked, or $(j, i) \in \mathbf{A}$ and (j, i) is ε^--unblocked. A node i is called a *predecessor* of node j if a directed path from i to j exists in $\mathbf{G^*}$.

First, we claim that immediately following a price rise at node j, the in-degree of j in $\mathbf{G^*}$ is 0, and hence j has no predecessors. To see this, note that if $(i, j) \in \mathbf{A}$ is ε^+-balanced after the price change, it must have been ε-active beforehand, and hence $f_{ij} = c_{ij}$, implying that (i, j) is not in $\mathbf{A^*}$. The ε^--balanced case is similar, establishing the claim. We next claim that $\mathbf{G^*}$ is always acyclic. This is true initially because assumption (c) of the previous section implies that $\mathbf{A^*}$ is

empty. Flow change operations (steps 2 and 3) can only remove arcs from A^*, so G^* can acquire a cycle only immediately after a price rise at some node j, and the cycle must include that node. But since j must then have in-degree 0, no such cycle is possible. This establishes the second claim. Finally, we claim that the node list maintained by the algorithm will always be compatible with the partial order induced by G^*, in the sense that every node will always appear in the list after all its predecessors. Again, this is initially true because A^* starts out empty. Furthermore a flow change operation does not create new predecessor relationships, while after a price rise at some node i, i can have no predecessors and is moved to the head of the list, before any possible descendants. This establishes the last claim.

Let N^+ be the set of nodes with positive surplus that have no predecessor with positive surplus, and let N^0 be the set of nodes with nonpositive surplus that have no predecessor with positive surplus. Then, as long as no price increase takes place, all nodes in N^0 remain in N^0, and execution of a *complete* up iteration at a node $i \in N^+$ moves i from N^+ to N^0. If no node changed price during a cycle, then all nodes of N^+ will be added to N^0 by the end of the cycle, which implies that the algorithm terminates. Therefore there will be a node price change during every cycle except possibly for the last cycle. Since the number of price increases per node is $O(\beta/\epsilon + N)$, this leads to an estimate of a total of $O(N(\beta/\epsilon + N))$ cycles, and an $O(N^2(\beta/\epsilon + N))$ overall worst case complexity based on the argument given earlier. The proof is complete.

6. The Distributed Asynchronous (Chaotic) Version of the Algorithm

In this section, we assume that each node i is a processor that updates its own price and incident arc flows, and exchanges information with its "forward" adjacent nodes

$$F_i = \{j \mid (i, j) \in A\},$$

and its "backward" adjacent nodes

$$B_i = \{j \mid (j, i) \in A\}.$$

The following distributed asynchronous implementation applies to both the pure algorithm and to the subproblems of the scaled method. The information available at node i for any time t is as follows:

$p_i(t)$: The price of node i

$p_j(i, t)$: The price of node $j \in F_i \cup B_i$ communicated by j at some earlier time

$f_{ij}(i, t)$: The estimate of the flow of arc (i, j), $j \in F_i$, available at node i at time t

$f_{ji}(i, t)$: The estimate of the flow of arc (j, i), $j \in B_i$, available at node i at time t

$g_i(t)$: The estimate of the surplus of node i at time t given by

$$g_i(t) = \Sigma_{(j,i)\in A}\ f_{ji}(i, t) - \Sigma_{(i,j)\in A}\ f_{ij}(i, t) - s_i \qquad (27)$$

A more precise description of the algorithmic model is possible, but for brevity we will keep our description somewhat informal. We assume that, for every node i, the quantities above do not change except possibly at an increasing sequence of times $t_0, t_1, \ldots$, with $t_m \to \infty$. At each of these times, generically denoted t, and at each node i, one of three events happens:

* * * * * * * * * * * * * * * * * *

Event 1. Node i does nothing.

Event 2. Node i checks $g_i(t)$. If $g_i(t) \le 0$, node i does nothing further. Otherwise node i executes either a complete or partial up iteration based on the available price and flow information

$$p_i(t),\ \ p_j(i, t), j \in F_i \cup B_i,\ \ f_{ij}(i, t), j \in F_i,\ \ f_{ji}(i, t), j \in B_i,$$

and accordingly changes

$$p_i(t),\ \ f_{ij}(i, t), j \in F_i,\ \ f_{ji}(i, t), j \in B_i.$$

Event 3. Node i receives, from one or more adjacent nodes $j \in F_i \cup B_i$ a message containing the corresponding price and arc flow $(p_j(t'), f_{ij}(j, t'))$ (in the case $j \in F_i$), or $(p_j(t'), f_{ji}(j, t'))$ (in the case $j \in B_i$) stored at j at some earlier time $t' < t$. If

$$p_j(t') < p_j(i, t),$$

node i discards the message and does nothing further. Otherwise, node i stores the received value $p_j(t')$ in place of $p_j(i, t)$. In addition, if $j \in F_i$, node i stores $f_{ij}(j, t')$ in place of $f_{ij}(i, t)$ if

$$p_i(t) < p_j(t') + a_{ij}, \quad \text{and} \quad f_{ij}(j, t') < f_{ij}(i, t)$$

and otherwise leaves $f_{ij}(i, t)$ unchanged; in the case $j \in B_i$, node i stores $f_{ji}(j, t')$ in place of $f_{ji}(i, t)$ if

$$p_j(t') \ge p_i(t) + a_{ji}, \quad \text{and} \quad f_{ji}(j, t') > f_{ji}(i, t)$$

and otherwise leaves $f_{ij}(i, t)$ unchanged. (Thus, in case of a balanced arc, the "tie" is broken in favor of the flow of the start node of the arc.)

* * * * * * * * * * * * * * * * * *

Let T^i be the set of times for which an update by node i as in # 2 above is attempted, and let $T^i(j)$ be the set of times when a message is received at i from j as in # 3 above. We assume the following:

Assumption 1. Nodes never stop attempting to execute an up iteration, and receiving messages from all their adjacent nodes, *i.e.*, T^i and $T^i(j)$ have an infinite number of elements for all i and $j \in F_i \cup B_i$.

Assumption 2. Old information is eventually purged from the system, *i.e.*, given any time t_k, there exists a time $t_m \ge t_k$ such that the time of generation of the price and flow information received at any node after t_m (*i.e.*, the time t' in #3 above), exceeds t_k.

Assumption 3. For each i, the initial arc flows $f_{ij}(i, t_0)$, $j \in F_i$, and $f_{ji}(i, t_0)$, $j \in B_i$ are integer, and satisfy ϵ-CS together with $p_i(t_0)$ and $p_j(i, t_0)$, $j \in F_i \cup B_i$. Furthermore there holds

$$p_i(t_0) \ge p_i(j, t_0), \qquad \text{for all } j \in F_i \cup B_i$$

$$f_{ij}(i, t_0) \ge f_{ij}(j, t_0), \qquad \text{for all } j \in F_i$$

One possible set of initial conditions satisfying Assumption 3 but requiring very little cooperation between processors is $p_j(i, t_0) = -\infty$ for all i and $j \in F_i \cup B_i$, $f_{ij}(i, t_0) = c_{ij}$ and $f_{ji}(j, t_0) = b_{ij}$ for all i and $j \in F_i$.

The choice of initial conditions in Assumption 3 guarantees that for all $t \ge t_0$

$$p_i(t) \ge p_i(j, t''), \qquad \text{for all } j \in F_i \cup B_i,\ t'' \le t \qquad (28)$$

To see this, note that $p_i(t)$ is monotonically nondecreasing in t, and $p_i(j, t'')$ equals $p_i(t')$ for some $t' < t''$.

An important fact is that for all nodes i, and times t, $f_{ij}(i, t)$ and $f_{ji}(i, t)$ are integer, and satisfy ϵ-CS together with $p_i(t)$ and $p_j(i, t)$, $j \in F_i \cup B_i$. This is seen from (28), the logic of the up iteration, and the rules for accepting information from adjacent nodes.

Another important fact is that for all i, and $t \ge t_0$

$$f_{ij}(i, t) \ge f_{ij}(j, t), \qquad \text{for all } j \in F_i \qquad (29)$$

i.e., the start node of an arc has at least as high an estimate of arc flow as the end node. For a given $(i, j) \in A$, condition (29)

holds initially by Assumption 3, and it is preserved by up iterations since an up iteration at i cannot decrease $f_{ij}(i, t)$, while an up iteration at j cannot increase $f_{ij}(j, t)$. Therefore (29) can first be violated only at the time of a message reception. To show that this cannot happen throughout the algorithm we argue by contradiction: Suppose (29) first fails to hold at some time t, implying

$$f_{ij}(i, t) < f_{ij}(j, t). \qquad (30)$$

Let r be the last time up to or including t that i accepted a message from j that reduced i's flow estimate for (i, j), and let r' be the time that this message originated at j. Similarly, let s be the last time up to or including t that j increased its flow estimate for (i, j), and s' be the time the message responsible was sent from i. Using Assumption 3, it may be easily shown that both these times must exist, for otherwise the violation (30) cannot have occurred (note also that $t = \max\{r, s\}$). The acceptance of these messages at times r and s implies

$$p_i(r) < p_j(r') + a_{ij} \qquad (31)$$

$$p_i(s') \geq p_j(s) + a_{ij}\,. \qquad (32)$$

It also follows that $r' \leq s$ and $s' \leq r$, or, again, the violation at time t could not have occurred. By the monotonicity of prices, it then follows that

$$p_i(r') \leq p_i(s), \qquad p_i(s') \leq p_i(r).$$

Substituting these into (31) and (32), respectively, one obtains

$$\begin{aligned} &p_i(r) < p_j(s) + a_{ij} \\ &p_i(r) \geq p_j(s) + a_{ij}\,, \end{aligned} \qquad (33)$$

a contradiction. Thus, (29) must always hold for all $(i, j) \in A$.

Note that once a node i gets nonnegative surplus $g_i(t) \geq 0$, it maintains a nonnegative surplus for all subsequent times. The reason is that an up iteration at i can at most decrease $g_i(t)$ to zero, while in view of the rules for accepting a communicated arc flow, a message exchange with an adjacent node j can only increase $g_i(t)$. Note also that from (29) we obtain

$$\Sigma_i g_i(t) \leq 0, \qquad \text{for all } t \geq t_0\,. \qquad (34)$$

This implies that, at any time t, there is at least one node i with negative surplus $g_i(t)$ if there is a node with positive surplus. This node i must not have executed any up iteration up to time t, and therefore its price $p_i(t)$ must still be equal to the initial price $p_i(t_0)$.

We say that the algorithm terminates if there is a time t_k such that for all $t \geq t_k$ we have

$$g_i(t) = 0 \qquad \text{for all } i \in N \qquad (35)$$

$$f_{ij}(i, t) = f_{ij}(j, t) \qquad \text{for all } (i, j) \in A \qquad (36)$$

$$p_j(t) = p_j(i, t) \qquad \text{for all } j \in F_i \cup B_i. \qquad (37)$$

Termination can be detected by using an adaptation of the protocol for diffusing computations of Dijkstra and Sholten [13]. Our main result is:

Proposition 5: If problem (MCF) is feasible and Assumptions 1, 2, and 3 hold, the distributed, totally asynchronous version of the algorithm terminates.

Proof: Suppose no up iterations are executed at any node after some time t*. Then (35) must hold for large enough t. Because no up iterations occur after t*, all the $p_i(t)$ must thenceforth remain constant, and Assumption 1, (28), and the message acceptance rules imply (37). After t*, furthermore, no flow estimates may change except by message reception. By (37), the nodes will eventually agree on whether each arc is active, inactive, or balanced. The message reception rules, (29), Assumption 1, and Assumption 2 then imply the eventual agreement on arc flows (36). (The start node of each inactive arc will eventually accept the flow of the end node, and the end node of a balanced or active arc will eventually accept the flow of the start node.)

We assume therefore the contrary, *i.e.*, that up iterations are executed indefinitely, and hence for every t there is a time t' > t and a node i such that $g_i(t') > 0$. There are two possibilities: The first is that $p_i(t)$ converges to a finite value p_i for every i. In this case we assume without loss of generality that there is at least one node i at which an infinite number of up iterations are executed, and an adjacent arc (i, j) whose flow $f_{ij}(i, t)$ is changed by an integer amount an infinite number of times with (i, j) being ε^+-balanced. For this to happen there must be a reduction of $f_{ij}(i, t)$ through communication from j an infinite number of times. This means that $f_{ij}(j, t)$ is reduced an infinite number of times which can happen only if an infinite number of up iterations are executed at j with (i, j) being ε^--balanced. But this is impossible since, when p_i and p_j converge, arc (i, j) cannot become both ε^+-balanced and ε^--balanced infinitely often.

The second possibility is that there is nonempty subset of nodes N^∞ whose prices increase to ∞. From (34) it is seen that there is at least one node that has negative surplus for all t, and therefore also a constant price. It follows that N^∞ is a strict subset of N. Since the algorithm maintains ε-CS, we have for all sufficiently large t that

$$f_{ij}(i, t) = f_{ij}(j, t) = c_{ij} \qquad \text{for all } (i, j) \in A \text{ with } i \in N^\infty,\ j \notin N^\infty$$

$$f_{ji}(i, t) = f_{ji}(j, t) = b_{ji} \qquad \text{for all } (j, i) \in A \text{ with } i \in N^\infty,\ j \notin N^\infty\,.$$

Note now that all nodes in N^∞ have nonnegative surplus, and each must have positive surplus infinitely often. Adding (27) for all i in N^∞, and using both (29) and the above relations, we find that the sum of c_{ij} over all $(i, j) \in A$ with $i \in N^\infty$, $j \notin N^\infty$, plus the sum of s_i over $i \in N^\infty$ is less than the sum of b_{ji} over all $(j, i) \in A$ with $i \in N^\infty$, $j \notin N^\infty$. Therefore, there can be no feasible solution, violating the hypothesis. It follows that the algorithm must terminate. **Q.E.D.**

References

[1] Awerbuch, B., "Complexity of Network Synchronization", *Journal of the ACM*, Vol. 32, 1985, pp. 804-823

[2] Baudet, G. M., "Asynchronous Iterative Methods for Multiprocessors", *Journal of the ACM*, Vol. 15, 1978, pp. 226-244

[3] Bertsekas, D. P., "Distributed Relaxation Methods for Linear Network Flow Problems", *Proceedings of 25th IEEE Conference on Decision and Control*, Athens, Greece, 1986, pp. 2101-2106

[4] Bertsekas, D. P., "A Distributed Algorithm for the Assignment Problem", Unpublished LIDS Working Paper, M. I. T., March 1979

[5] Bertsekas, D. P., "A Unified Framework for Primal-Dual Methods in Minimum Cost Network Flow Problems", *Math. Programming*, Vol. 32, 1985, pp. 125-145

[6] Bertsekas, D. P., "A Distributed Asynchronous Relaxation Algorithm for the Assignment Problem", *Proc. 24th IEEE Conference on Decision and Control*, Ft Lauderdale, Fla., Dec. 1985, pp. 1703-1704

[7] Bertsekas, D. P., and Tseng, P., "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems", LIDS Report P-1462, M. I. T., May 1985, to appear in *Operations Research Journal*

[8] Bertsekas, D. P., Hossein, P., and Tseng, P., "Relaxation Methods for Network Flow Problems with Convex Arc Costs", LIDS Report P-1523, Dec. 1985, to appear in *SIAM J. on Control and Optimization*

[9] Bertsekas, D. P., and El Baz, D., "Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems", LIDS Report P-1417, M. I. T., Oct. 1984, to appear in *SIAM J. on Control and Optimization*

[10] Bertsekas, D. P., "Distributed Asynchronous Computation of Fixed Points", *Math. Programming*, Vol. 27, 1983, pp. 107-120

[11] Bertsekas, D. P., Tsitsiklis, J. N., and Athans, M., "Convergence Theories of Distributed Asynchronous Computation: A Survey", LIDS Report P-1412, M. I. T., Oct.

1984; also in *Stochastic Programming*, by F. Archetti, G. Di Pillo, and M. Lucertini (eds.), Springer-Verlag, N.Y., 1986, pp. 107-120
[12] Chazan, D., and Miranker, W., "Chaotic Relaxation", *Linear Algebra and its Applications*, Vol. 2, 1969, pp. 199-222
[13] Dijkstra, E. W., and Sholten, C.S., "Termination Detection for Diffusing Computations", *Information Processing Letters*, Vol. 11, 1980, pp. 1-4.
[14] Goldberg, A. V., "A New Max-Flow Algorithm", Tech. Mem. MIT/LCS/TM-291, Laboratory for Computer Science, M. I. T., 1985
[15] Goldberg, A. V., "Solving Minimum-Cost Flow Problems by Successive Approximations", extended abstract, submitted to *STOC 87*, Nov 1986
[16] Luenberger, D. G., *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1984
[17] Papadimitriou, C. H., and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J. 1982
[18] Rockafellar, R. T., *Network Flows and Monotropic Programming*, J. Wiley, N. Y., 1984
[19] Tseng, P., "Relaxation Methods for Monotropic Programming Problems", PhD Thesis, Dept. of Electrical Engineering and Computer Science, M. I. T., May 1986.
[20] Bland, R. G., and Jensen, D. L., "On the Computational Behavior of a Polynomial-Time Network Flow Algorithm", Tech. Report 661, School of Operations Research and Industrial Engineering, Cornell University, June 1985
[21] Edmonds, J. and Karp, R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems", *Journal of the ACM*, Vol. 19, 1972, pp. 248-264
[22] Orlin, J. B., "Genuinely Polynomial Simplex and Non-Simplex Algorithms for the Minimum Cost Flow Problem", Working Paper No. 1615-84, Sloan School of Management, M. I. T., Dec. 1984
[23] Rock, H., "Scaling Techniques for Minimal Cost Network Flows", in *Discrete Structures and Algorithms*, by V. Page (ed.), Carl Hansen, Munich, 1980