Intern. J. Computer Math., Vol. 32, pp. 85–95
Reprints available directly from the publisher
Photocopying permitted by license only

A GRADIENT DRIVEN TRANSPORTATION ALGORITHM

TOM ALTMAN

Department of Computer Science, University of Kentucky, Lexington, KY 40506

(Received 28 November 1988)

In this work we present a new algorithm for the solution of the transportation problem. Although the algorithm is gradient-directed, it does not use the classical gradient approach. Instead, it first computes a reduced gradient and then determines an improved feasible direction in the dual space, thereby extending the initial gradient. To accomplish this, we make use of a special diagonal ordering of the basic vectors. After obtaining the intial basic feasible solution, the algorithm may transfer, without any computational overhead, to a streamlined dual simplex routine, or continue on with the gradient method.

KEY WORDS: Transportation problem, extended reduced gradient.
C.R. CATEGORY: 5.41.

1. INTRODUCTION

Since the early 1970's, a number of significant improvements have been made in the area of minimum cost network flow and transportation algorithms [5–10]. Most of them were obtained as a result of the following:

a) efficient representation of the basic solution (using basic-trees),

b) better methods of pivoting and of updating the basic solutions,

c) improved methods for selection of the entering basic variable,

d) improved methods for obtaining the initial basic solution.

Of the above, (a) has had the greatest impact on the performance of both primal and dual simplex-based (as well as other, e.g., see [9]) algorithms. R. Barr, F. Glover, D. Klingman, A. Napier, V. Srinivasan, G. Thompson and others have introduced a number of improvements in the representation of the basic-tree which allowed for faster solution of problems in exchange for greater space requirements [6, 9]. The algorithmic improvements proposed by the above researchers have been significant. In the majority of their implementations, the primal (or dual) simplex method was the basic tool used to drive the algorithms.

In this paper we plan to investigate the behavior and performance of the Extended Reduced Gradient (ERG) network flow algorithm [4]. There are several basic differences between the approach in this work and the majority of previous

and ongoing research. The underlying mathematical methods which are incorporated in our algorithm are different from those that drive the standard algorithms.

1.1 Problem Specification

In general, the transportation problem is concerned with distributing some commodity from a group of supply centers, called sources, to a group of receiving centers, called destinations, in such a way as to minimize the total distribution costs.

Given that we have m sources and n destinations, a matrix C denoting the cost of unit flow shipped from source i to destination j, and letting $a_i>0$ and $b_j>0$ denote the availability at source i and requirement at destination j, respectively, the primal problem can be stated mathematically as:

Select all x_{ij} (the flow from source i to destination j) so as to minimize the cost function

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

subject to

$$\sum_{j=1}^{n} x_{ij}=a_i, \quad i=1,2,\ldots,m; \sum_{i=1}^{m} x_{ij}=b_j, \quad j=1,2,\ldots,n, \tag{2}$$

where we assume

$$\sum_{i=1}^{m} a_i=\sum_{j=1}^{n} b_j, \quad c_{ij},x_{ij}\geqq 0, \quad i=1,2,\ldots,m; j=1,2,\ldots,n. \tag{3}$$

The dual of the transportation problem can be expressed as:

Choose u_i and v_j so as to maximize the linear form

$$\sum_{i=1}^{m} a_iu_i+\sum_{j=1}^{n} b_jv_j \tag{4}$$

subject to

$$u_i+v_j\leqq c_{ij}, \quad i=1,2,\ldots,m; \quad j=1,2,\ldots,n. \tag{5}$$

As described above, the transportation problem may be represented as a bipartite graph consisting of a set of origin nodes with supplies a_i and destination nodes with demands b_j. Directed arcs from origin nodes to destination nodes accommodate the flow and incur a cost c_{ij}. The objective is to determine a set of

arc flows that satisfy the supply and demand at a minimum total cost. Given an m by n transportation problem, a representation of a given basic solution corresponds to a spanning tree with $m+n-1$ arcs over the bipartite graph of m source and n destination nodes.

2. MATHEMATICAL INTERPRETATION OF ERG AND MOTIVATION

The ERG algorithm is based on a class of gradient methods presented in [1–4]. The algorithm first computes a reduced gradient which is then extended to an improved feasible direction in the dual space. Thereby, one obtains an *extended reduced gradient.*

Since the extended reduced gradient is a feasible direction which yields an increase of the dual cost function, the algorithm is actually based on a feasible direction method in the dual space. However, the space itself is not being reduced in the process, as is the case with the reduced gradient method for general linear programming problems. In the first phase of our algorithm, an initial basic feasible solution for the dual problem is obtained. In the second phase one has an option: either to find an appropriate feasible direction, or to continue with the dual simplex method (i.e., without any computational overhead, one can use the streamlined dual routine which exploits the unique structure of the transportation problem). In this paper we consider the second option.

The motivation behind attempting the gradient approach is the following:

The reduced gradient method of finding an initial basic solution for the dual problem is well known for general linear programming problems, (see, e.g., [11]*). The structure of the problem is such that the dual variables are pairs (u_i, v_j) associated with the cost c_{ij}. Such pairs can be depicted as arcs in a graph. The reduced gradient method systematically eliminates the variables, and if, e.g., u_i is eliminated from the pair (u_i, v_j), then the symmetry is destroyed and the pair disappears. Fortunately, with the extended gradient we can still benefit from the reduced gradient and, at the same time, restore the symmetry which again produces the pairs.

A solution to (1) or (4) will have at most $m+n-1$ basic variables. Let B denote a set that will contain the index pairs (i,j) of these variables. We say that a vector y, denoted by $y=(u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n)^T$, is a dual feasible solution for (4) if

$$u_i+v_j=c_{ij} \quad \text{for} \quad (i,j)\in B^k, \tag{6}$$

and

$$u_i+v_j<c_{ij} \quad \text{for} \quad (i,j)\notin B^k,$$

---

*Translation from Russian, (p. 507): "... *as a rule, the obtained initial basic solution (by the gradient method) turns out to be a good approximate solution to the problem*" of linear programming. "*This is clear: one moves in the direction of steepest decrease of the linear form*". On page 531 an example is given which confirms the above statement and the authors conclude: "*As mentioned before, this is not accidental: the gradient method usually leads to an initial basic solution which is close to the optimal one*".

where B^k is a set that contains k index pairs of the current dual feasible solution ($k \leqq m+n-1$). If $k=m+n-1$, then y is a basic dual feasible solution, and we set $B=B^k$, where $(i,j) \in B$ are the indices of the basic variables.

In order to construct the extended reduced gradient directions, we organize B^k by a diagonal ordering of the basic vectors. Suppose

$$B^k=\{(i_1,j_1),(i_2,j_2),\ldots,(i_k,j_k)\}. \tag{7}$$

The diagonal ordering of the basic variables can be interpreted as a vector-pair

$$B^k=\begin{bmatrix} i_1,i_2,\ldots,i_h,i_{h+1},i_{h+2},\ldots,i_k \\ j_1,j_2,\ldots,j_h,j_{h+1},j_{h+2},\ldots,j_k \end{bmatrix}^T \tag{8}$$

such that

$$i_1<i_2<\cdots<i_h \leqq m \quad \text{and} \quad j_{h+1}<j_{h+2}<\cdots<j_k \leqq n.$$

2.1 The Reduced Gradient

The steepest descent method of minimizing a (nonlinear) function without constraints is based on the gradient direction approach. In the case of the linear form (4), the gradient vector g is:

$$g=(a_1,a_2,\ldots,a_m,b_1,b_2,\ldots,b_n)^T. \tag{9}$$

The reduced gradient is obtained by eliminating from (4) the basic variables (u_i,v_j) with $(i,j) \in B^k$, i.e., using the equations

$$u_i+v_j=0, \quad \text{with} \quad (i,j) \in B^k. \tag{10}$$

The reduced gradient is then extended to an $(m+n)$-vector s,

$$s=(\alpha_1,\alpha_2,\ldots,\alpha_m,\beta_1,\beta_2,\ldots,\beta_n)^T, \tag{11}$$

where the α_i and β_j values are determined by Eqs. (16) and (17), and s is a feasible direction in the dual space, hence, increasing the value of the objective function (4).

The derivation of the initial gradient basic solution takes $m+n-1$ steps. At the kth step, the set B^k is created as follows. We define a sequence $\{y^{(k)}\}$ of feasible solutions to the dual problem as:

$$y^{(k)}=(u_1^{(k)},u_2^{(k)},\ldots,u_m^{(k)},v_1^{(k)},v_2^{(k)},\ldots,v_n^{(k)})^T \tag{12}$$

for $k=1,\ldots,m+n-1$. The sequence can be constructed iteratively by setting:

$$y^{(0)}=0, s^{(0)}=g=(a_1,a_2,\ldots,a_m,b_1,b_2,\ldots,b_n)^T$$

$$y^{(k+1)} = y^{(k)} + \theta^{(k)} s^{(k)}, \text{ where}$$

$$s^{(k)} = (\alpha_1^{(k)}, \alpha_2^{(k)}, \ldots, \alpha_m^{(k)}, \beta_1^{(k)}, \beta_2^{(k)}, \ldots, \beta_n^{(k)})^T$$

from which it follows that

$$y^{(k+1)} = (u_1^{(k+1)}, u_2^{(k+1)}, \ldots, u_m^{(k+1)}, v_1^{(k+1)}, v_2^{(k+1)}, \ldots, v_n^{(k+1)})^T,$$

where

$$u_i^{(k+1)} = u_i^{(k)} + \theta^{(k)} \alpha_i^{(k)}, \quad i = 1, 2, \ldots, m,$$

$$v_j^{(k+1)} = v_j^{(k)} + \theta^{(k)} \beta_j^{(k)}, \quad j = 1, 2, \ldots, n,$$

and $\theta^{(k)}$ is defined by the equation:

$$\theta^{(k)} = \min_{\alpha_i^{(k)} + \beta_j^{(k)} > 0} \frac{c_{ij} - u_i^{(k)} - v_j^{(k)}}{\alpha_i^{(k)} + \beta_j^{(k)}} = \frac{c_{pq} - u_p^{(k)} - v_q^{(k)}}{\alpha_p^{(k)} + \beta_q^{(k)}}. \tag{13}$$

From (13) we obtain a new pair of indices (p, q) of variables to be included in the basic set.

The set B^{k+1} is defined as:

$$B^{k+1} = B^k \cup \{(p, q)\}. \tag{14}$$

Continuing this process for $k = 1, \ldots, m+n-1$, we arrive at a dual basic feasible solution. Note that a basic solution consists of a spanning tree [5, 9].

2.2 Extending the Reduced Gradient

Below we present a procedure for obtaining the vector s.

The set B^k is organized as a forest of trees of connected pairs, where two pairs (i_1, j_1) and (i_2, j_2) are in the same tree if $i_1 = i_2$ or $j_1 = j_2$, or if there exists some sequence of connecting pairs between (i_1, j_1) and (i_2, j_2) in the tree. Hence, the set B^k is made up of at most k trees, i.e.,

$$B^k = B_1 \cup B_2 \cup \cdots \cup B_f, \text{ where } f \leqq k. \tag{15}$$

Let B_1 be one of the trees in B^k. Let us denote the variables of (4) which are in B_1 by

$$u_{i_1}, u_{i_2}, \ldots, u_{i_d}, v_{j_1}, v_{j_2}, \ldots, v_{j_e},^*$$

and let

*Note that d and e may be different.

$$\Delta_1=(b_{j_1}+b_{j_2}+\cdots+b_{j_e})-(a_{i_1}+a_{i_2}+\cdots+a_{i_d}). \tag{16}$$

Also let

$$\alpha_{i_1}=\alpha_{i_2}=\cdots=\alpha_{i_d}=-\Delta_1, \tag{17}$$

$$\beta_{j_1}=\beta_{j_2}=\cdots=\beta_{j_e}=\Delta_1.$$

This is a portion of the vector s in (11) with coordinates corresponding to the B_1-tree of the set B^k. In the same way we obtain the α and β values corresponding to the remaining trees $B_2, B_3, \ldots, B_f$. It is clear that if a tree consists of only one pair (i,j), then $\alpha_i=-(b_j-a_i)$ and $\beta_j=(b_j-a_i)$. The remaining coefficients of the linear form (4) are then included in s without any changes. Notice that (16) is a result of the consecutive elimination process of the basic variables associated with B^k in the diagonal ordering given by (8), whereas (17), or $\alpha_i+\beta_j=0$, makes sure that the feasible direction (11) acts as a reduced gradient in the reduced dual space. This completes the procedure for obtaining the vector s.

After obtaining the initial gradient basic solution, there are two options for the ERG algorithm. The first one, which turned out to be impractical, was to obtain a second basic solution using the above gradient approach (with the vector y as a new starting point). Although the method does converge, the computational costs at each gradient evaluation become too costly for practical consideration.

A second alternative is to transfer to another dual routine for the pivoting phase. Experimental results have shown the streamlined dual simplex method to be a good choice.

The basic solution $y=y^{(m+n-1)}$ gives us the first set B of basic pairs (i,j). Two cases arise when the primal solution corresponding to this basis is computed:

Case (1): $x_{ij}\geqq 0$ for all $(i,j)\in B$,

Case (2): $x_{i'j'}<0$ for some $(i',j')\in B$.

In the first case, the solution obtained is optimal since it is both dual and primal feasible. In the second case, we find an integer solution to the system

$$\alpha_i+\beta_j=0 \text{ for } (i,j)\in B \text{ and } (i,j)\neq(i',j')$$

$$\alpha_{i'}+\beta_{j'}=-1. \tag{18}$$

In solving (18) we obtain a dual feasible direction

$$s=(\alpha_1,\alpha_2,\ldots,\alpha_m,\beta_1,\beta_2,\ldots,\beta_n)^T \tag{19}$$

which increases (4) and removes the pair (i',j') from B. At the same time a new basic pair (k,l) is generated and added to the set B. The pair (k,l) is defined by the relation

$$\theta=\min_{\alpha_i+\beta_j=1}(c_{ij}-u_i-v_j)=c_{kl}-u_k-v_l, \tag{20}$$

where

$$y = y^{(m+n-1)} = (u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n)^T$$

Solving (2) with the new set B, we again arrive at Cases (1) or (2), described above. Assuming no degeneracy, this process will produce an optimal solution. Even with degeneracy, a slight modification of the above algorithm (e.g., see [5] for details) will produce the desired result.

2.3 Description of the Algorithm

In this section we give a general description of the ERG algorithm as it has been implemented using the specifications from the previous sections. The algorithm is divided into two distinct phases, which are outlined below.

Phase 1

Step 0. $k := 1$;
Step 1. Determine θ^k and the new pair (p, q) to be added to the basic set B^k, as in (13);
Step 2. Insert the new pair (p, q), preserving the diagonal order of (8);
Step 3. Update the y vector as in (12);
Step 4. Determine the new feasible direction vector s, as in (17);
Step 5. $k := k+1$; If $(k < m+n-1)$ Go to Step 1;

Phase 2:

Step 6. Solve (2) with $(i,j) \in B$ to obtain the primal variables x_{ij};
Step 7. If all x_{ij} are non-negative, STOP;
Step 8. Find the smallest negative $x_{i'j'}$ and remove the pair (i',j') from the basis as in (18);
Step 9. Determine the new feasible direction vector s, as in (19);
Step 10. Find the new θ and the pair (k, l), as in (20);
Step 11. Insert the new pair (k, l), preserving the diagonal order of (8);
Step 12. Update the y vector as in (12);
Step 13. Go to Step 6;

In the first phase of the algorithm, the initial basic solution is determined. That phase consists of $m+n-1$ steps, where at each step a new extension to the gradient vector is computed. Upon termination of the first phase, the primal feasibility (and hence, optimality) condition is checked. If it is violated, then the second phase is initiated. There the arc corresponding to the "*most*" negative flow $x_{i'j'}$ is eliminated from the basic set. A new arc is then introduced into the basis in the same fashion as in the last step of the first phase. The process continues until primal feasibility, i.e., optimality, is reached.

3. EMPIRICAL RESULTS

In this section we present results of the computational study that was performed

to evaluate the ERG algorithm. The ERG algorithm was compared with several implementations of the standard algorithms for the transportation problem. First, we describe the methodology used for creating the random transportation problems used for our evaluation. We then present an analysis of ERG's performance in comparison to the primal simplex transportation method (often referred to as the Row-Column Sum, or MODI method).

3.1 Problem Generation

In this section we describe the method used to create the random transportation problem and the parameters that were varied to obtain a broad population sample for our study.

We have used the following five parameters to determine an instance of a newly created random transportation problem:

1] m–the number of sources,

2] n—the number of destinations,

3] *density*—the relative density of the cost matrix,

4] *range*—the upper bound on the value of the entries in the cost matrix,

5] *tflow*—the total number of units to be shipped.

Each parameter was varied independently so that *small*, *medium*, and *large* groups of problems (with respect to the given parameter) were created. In particular, the values for small m were between 30 and 40, for medium m: 60 to 70, and for large m: 90 to 100. The same method was used for grouping the values of n. The *density* was grouped into 70 to 79 percent, 80 to 89 percent, and 100 percent populations. The groupings of *range* were 1 to 10, 1 to 100 and 1 to 1000 respectively. Finally, the *tflow* was defined small for values of 10(mn), medium for 100(mn), and large for 1000(mn). Using the methodology described above, we have created over 1600 random transportation problems that were then solved by the ERG and MODI algorithms.

We have used two criteria by which the two algorithms were evaluated. The first was the cpu time required to solve a given random transportation problem. The second was the total number of pivots needed by each method. Moreover, we intended to determine which of the five parameters has had the most influence on the cpu time and the number of pivots for each method.

3.2 The Results

In this section we present the results of our computational study performed on the randomly generated transportation problems. As expected, the ERG algorithm was considerably faster than the MODI method. The average number of pivots required by ERG was less than half the number of pivots required by MODI.

Moreover, the average time per pivot for ERG was considerably less than for MODI.

In analyzing the behavior of ERG and MODI (see Plots 1 and 2), it is clear that, as the problems increased in size, the rate of increase in the number of pivots is a lot smaller in ERG than in MODI. As a result, the same behavior held true for the execution time. Hence, we can expect the performance of ERG to become increasingly better than MODI's for larger size transportation problems.

3.2.1 The behavior of ERG

As expected, the parameter which had the most statistically significant effect on the cpu time and the number of pivots for ERG was the total number of arcs, which is a function of m, n, and *density*. Of the three parameters, m—the number of sources, had slightly more influence than n—the number of destinations. However, the difference between the effects of m and n was not statistically significant.

The *range* parameter had no significant effect on the cpu time, but it did have a slight effect on the number of pivots. This could be attributed to the following conjecture: the arithmetic operations involving small cost entries decreased the overall arithmetic computation time, but these savings were offset by the fact that the initial gradient did not point close to the optimum (since the transportation polytope is practically flat when the range of the c_{ij}'s is small); hence, the number of pivots in the second phase increased slightly.*

The parameter *tflow* had a negligible effect on the cpu time and no significant effect on the number of pivots. This can be explained by the fact that in the computation of the minimal θ-ratio, the arithmetic operations take slightly longer for larger integers. Since the direction of the initial gradient is not dependent on the total flow (e.g., we can multiply all a_i's and b_j's by 10 and still obtain the same initial gradient basic solution), the number of pivots is not affected by this parameter.

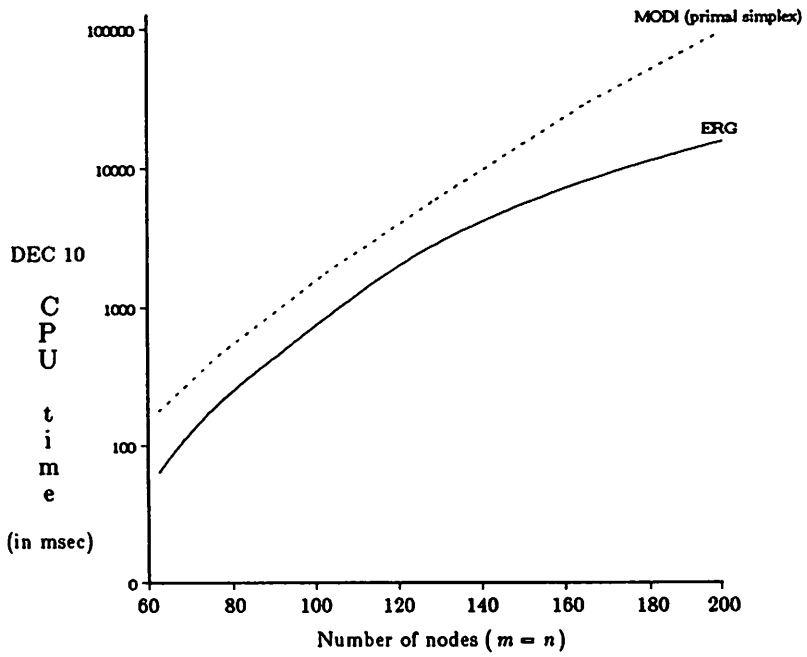
3.2.2 The behavior of MODI

Because only about one percent of the cpu time was spent in the initial phase of MODI, it should be clear that the parameters affecting the number of pivots will be the major factor in the determination of the overall cpu time. The size of the problem was again the overwhelming factor in determining the number of pivots and, hence, the cpu time. An interesting observation was that the *density* parameter was a more significant factor for MODI than ERG. To a lesser degree, the same held for the *range* and *flow* parameters.

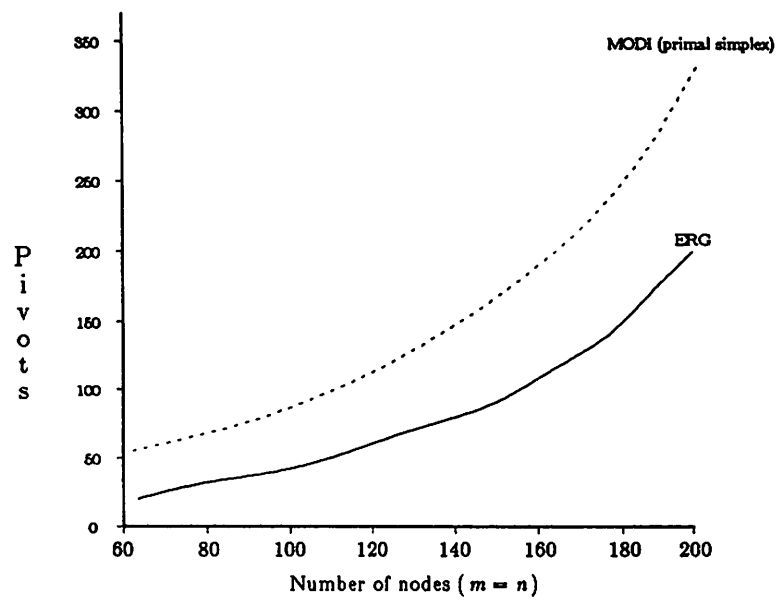
In the plots that follow, we show the results of our computational comparisons between ERG and MODI. Plots 1 and 2 display the execution times and number of pivots performed in solving our sample of randomly generated problems.

---

*Note that scaling a small-range cost matrix (i.e., multiplying all entries by a large number) will not change the *shape* of the polytope to make it more suitable for ERG.



Plot 1 Execution time vs. Problem size.



Plot 2 Number of pivots vs. Problem size.

4. CONCLUSIONS

In this work we have developed a new gradient-directed algorithm for the solution of the transportation problem. Some of the benefits of ERG are:

1] Because it uses a gradient to arrive at the initial basic solution, the number of pivots required is substantially less than that required for the simplex-based algorithms; this was substantiated by our empirical study.

2] The underlying mathematics of ERG, combined with the diagonal ordering, makes it possible to use less elaborate data structures than those employed by the standard algorithms.

3] Since ERG is a dual algorithm, it can be used to solve problems whose *supplies* and *demands* may not be permanently fixed, but rather are subject to change. In such situations, the ability to begin from an optimal basis to a given problem and proceed via the dual method to an optimal solution for the altered problem is extremely useful.

Acknowledgement

The author wishes to thank Professor Carl Lee for his helpful discussions and comments.

References

[1] M. Altman, Two gradient method algorithms for linear programming with application to the decomposition problem, *Bulletin de l'Académie Polonaise des sciences Série des sciences math., astr. et phys.* **13** (1965), 465–472.

[2] M. Altman, The sign-gradient method algorithms in linear programming, *Analele Stiintifice, Universit. Al.I. Cusa (Jasi)* **II** (1965), 323–330.

[3] M. Altman, A method of solving the transportation problem, *Bulletin de l'Académie Polonaise des sciences Série des sciences math., astr. et phys.* **15** (1967), 801–811.

[4] T. Altman, A gradient directed primal-dual algorithm for the transportation problem, *XI International Symposium on Mathematical Programming*, Bonn, W. Germany, 1982.

[5] R. D. Armstrong, D. Klingman and D. Whitman, Implementation and analysis of a variant of the dual method for the capacitated transshipment problem, *European Journal of Operational Research* **4** (1980), 403–420.

[6] F. Glover and D. Klingman, Recent developments in computer implementation technology for network flow algorithms, *Research Report CCS* 377, University of Texas, Austin, 1980.

[7] Y. Ikura and G. L. Nemhauser, A polynomial-time dual simplex algorithm for the transportation problem, Tech. Report 602, *School of Operations Research and Industrial Engineering*, Cornell University, Ithaca, N.Y., 1983.

[8] J. B. Orlin, On the simplex method for networks and generalized networks, *Mathematical Programming*, to appear.

[9] V. Srinivasan and G. L. Thompson, Cost operator algorithms for the transportation problem, *Mathematical Programming* **12** (1977), 372–391.

[10] E. Tardos, A strongly polynomial minimum cost circulation algorithm, *Combinatorica* **5** (1985), 247–255.

[11] D. B. Yudin and E. G. Gol'stein, *Linear Programming*, pp. 507–531, Φ.M. (in Russian), 1963.