

July 1990

RELAXT-III: A NEW AND IMPROVED VERSION  
OF THE RELAX CODE<sup>1</sup>

D.P. Bertsekas<sup>2</sup>  
P. Tseng<sup>3</sup>

ABSTRACT

RELAX is a popular public domain Fortran code for solving the linear minimum cost flow problem. In this paper we briefly describe a new release of RELAX, called RELAXT-III, and we present computational results demonstrating that RELAXT-III outperforms two state-of-the-art simplex codes, RNET and NETFLO, by a broad margin on randomly generated problems.

---

<sup>1</sup>This research was supported in part by the Army Research Office under grant ARO DAAL03-86-K-0171 (Center for Intelligent Control Systems), and by the National Science Foundation under grant NSF DDM-8903385. Thanks are due to J. Kennington for providing us with an updated version of the NETFLO code.

<sup>2</sup>Center for Intelligent Control Systems and Laboratory for Information and Decision Systems, M.I.T., Cambridge, MA 02139.

<sup>3</sup>Center for Intelligent Control Systems and Laboratory for Information and Decision Systems, M.I.T., Cambridge, MA 02139.

## 1. INTRODUCTION

Linear network optimization problems such as shortest path, assignment, max-flow, transportation, and transshipment, are undoubtedly the most common optimization problems in practice. Extremely large problems of this type, involving thousands and even millions of variables, can now be solved routinely, thanks to recent algorithmic and technological advances.

Up to the late seventies, there were basically two types of algorithms for linear network optimization: the *simplex* method and its variations, and the *primal-dual* method and its close relative, the *out-of-kilter* method. There was some controversy regarding the relative merit of these methods, but thanks to the development of efficient implementation ideas, the simplex method emerged as the fastest of the two for most types of network problems.

A number of algorithmic developments in the eighties have changed significantly the situation. New methods were invented that challenged the old ones, both in terms of practical efficiency, and theoretical worst-case performance. One of these methods, called *relaxation* [Ber82], [BeT85], is a dual ascent method resembling the coordinate ascent method of unconstrained nonlinear optimization. It significantly outperforms in practice both the simplex and the primal-dual methods for many types of problems, as evidenced by extensive computational experiments with codes of the RELAX family developed by the authors. These codes are described in [BeT88] and are in the public domain; they are distributed by the authors as a service to the research community at no cost.

The present note describes briefly a new release of RELAX, called RELAXT-III, and compares it computationally with two state-of-the-art primal simplex codes RNET and NETFLO.

## 2. THE RELAXATION METHOD

We first introduce the minimum cost flow problem that the relaxation method solves. Consider a directed graph with node set  $\mathcal{N} = \{1, 2, \dots, N\}$  and arc set  $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$ , with each arc  $(i, j)$  having a cost coefficient  $a_{ij}$ . Letting  $x_{ij}$  be the flow of the arc  $(i, j)$ , the problem is

$$\begin{aligned} & \text{minimize} && \sum_{(i,j) \in \mathcal{A}} a_{ij} x_{ij} && (MCF) \\ & \text{subject to} && \sum_{\{j|(i,j) \in \mathcal{A}\}} x_{ij} - \sum_{\{j|(j,i) \in \mathcal{A}\}} x_{ji} = s_i, && \forall i \in \mathcal{N}, \\ & && 0 \leq x_{ij} \leq c_{ij}, && \forall (i, j) \in \mathcal{A}, \end{aligned} \tag{1}$$

$$\tag{2}$$

where  $a_{ij}$ ,  $c_{ij}$ , and  $s_i$  are given integers.

We refer to  $c_{ij}$  as the *capacity* of arc  $(i, j)$ , we refer to  $s_i$  as the *supply* of node  $i$ , and we refer to the constraints (1) and (2) as the *conservation of flow constraints* and the *capacity constraints* respectively. A flow vector satisfying both of these constraints is called *feasible*. We remark that the relaxation method can handle multiple arcs in each direction between any pair of nodes.

A dual problem to (MCF) is obtained by associating a Lagrange multiplier  $p_i$  with the conservation of flow constraint (1) at node  $i$ . This dual problem is

$$\begin{aligned} & \text{maximize } q(p) \\ & \text{subject to no constraint on } p, \end{aligned} \tag{3}$$

where  $q$  is the dual functional given by

$$q(p_1, \dots, p_N) = \sum_{(i,j) \in \mathcal{A}} \min_{0 \leq x_{ij} \leq c_{ij}} (a_{ij} + p_j - p_i)x_{ij} + \sum_{i \in \mathcal{N}} s_i p_i. \tag{4}$$

This formulation of the dual problem has been used in many prior works dealing with primal-dual and relaxation methods; see e.g. [Roc84], [BeT89]. We henceforth refer to (MCF) as the *primal problem*, and note that standard duality results imply that the optimal primal cost equals the optimal dual cost. We refer to the multiplier  $p_i$  as the *price of node  $i$* .

The relaxation method belongs to a special class of iterative ascent methods for solving the dual problem (3) in which ascents are made along *elementary* directions. In these methods one starts with a price vector and tries to successively obtain new price vectors with improved dual cost. Each price change entails changing the price of a subset of nodes, say  $\mathcal{L}$ , by the *same* amount, while the price of the remaining nodes are held fixed, that is, the prices are iterated according to

$$p_i := \begin{cases} p_i + \alpha & \text{if } i \in \mathcal{L}, \\ p_i & \text{if } i \notin \mathcal{L}, \end{cases}$$

where  $\alpha$  is some positive scalar, which is small enough to ensure that the new price vector has an improved dual cost.

Different methods correspond to different schemes for determining the elementary directions (i.e., the node sets  $\mathcal{L}$ ). Classical methods in this class include the primal-dual method, the dual-simplex method, and the relaxation or coordinate ascent method. What distinguishes the relaxation method from the other methods is that each elementary direction is computed so that it has a *small number of nonzero elements* (i.e., the corresponding set  $\mathcal{L}$  has few nodes). Such a direction may not be optimal in terms of rate of dual cost improvement, but can typically be computed much faster than those generated by the other methods. In fact, we found that often the direction generated by an iteration of the relaxation method has just a single nonzero element, in which case the price of only one node is changed; this motivates the name “coordinate ascent”.

#### 4. Code Comparisons

Like most dual ascent methods, the relaxation method simultaneously with changing  $p$  along a direction of dual cost improvement, also iterates on a flow vector  $x$  satisfying *complementary slackness* together with  $p$ , that is,  $x$  satisfies the capacity constraints (2) and, for each arc  $(i, j)$ ,  $x_{ij}$  is at the lower bound 0 (respectively, the upper bound  $c_{ij}$ ) if the *reduced cost* of that arc, namely  $a_{ij} - p_i + p_j$ , is positive (respectively, negative). This flow vector guides the search for a dual ascent direction and helps to determine when the price vector reaches optimality.

### 3. THE RELAXT-III CODE

RELAXT-III is a new release of the RELAX codes, implementing the relaxation method. It differs from previous releases, namely RELAX-II and RELAXT-II described in [BeT85] and [BeT88], primarily in the initialization of the node prices. In RELAX-II and RELAXT-II no attempt is made to use “favorable” initial prices (although the user can supply them based, for example, on the results of an earlier optimization). In RELAXT-III the prices are initialized using a special *shortest path* procedure between the sources and the sinks (nodes with nonzero supply). Not all arcs are taken into account in the shortest path initialization; only those arcs that have relatively large capacity. Also, the initialization is done only for problems with relatively few sources and sinks. Such problems often tend to have a shortest path-like structure and can benefit greatly from the new initialization procedure. RELAXT-III also contains a number of other minor improvements in coding that improve its efficiency under certain circumstances.

RELAXT-III is most similar to the RELAXT-II code. Like the latter, it stores reduced arc costs rather than node prices, and it maintains the set of arcs with zero reduced cost in two linked arc lists that facilitate the efficient retrieval of these arcs for various operations.

### 4. CODE COMPARISONS

The main competitor of the relaxation method seems to be the primal simplex method as specialized to the minimum cost flow problem (see, for example, [Dan63], [HeK80], [Chv83], [AMO89]). RNET and NETFLO are two state-of-the art implementations of this method, developed by Grigoriadis and Hsu [GrH80], and Kennington and Helgason [KeH80], respectively. We have compared RELAXT-III with RNET and NETFLO on a  $\mu$ VAX Workstation 2000 running VMS 4.6. We have

also compared RELAXT-III with NETFLO on a Macintosh II using the Absoft FORTRAN compiler. We note that our version of RNET was compiled under an older version (3.7) of VMS. We estimate that approximately, a 15% reduction in the solution time of RNET could be obtained if it were compiled under VMS 4.6.

To gain some perspective on how RELAXT-III compares with primal-dual and sequential shortest codes, we have conducted a limited comparison with the following codes on a Macintosh II:

- (a) SSP: This is our straightforward implementation of the sequential shortest path (or primal-dual) method, using the same data structures and coding techniques as for RELAXT-III.
- (b) RELAX-SSP: This is a combination of the relaxation method and the sequential shortest path method. It use several cycles of coordinate ascent iterations to initialize the preceding code SSP.

Table 1 shows the times required for solution of the first 35 NETGEN benchmark problems [KNS74], which are the benchmarks most commonly used to compare network flow codes. They are a mixture of transportation problems (1-10), assignment problems (11-15), and lightly capacitated transshipment problems (16-35). Detailed specifications of these problems may be found in many sources, including [KNS74], [KeH80], [BeT85], and [BeT88].

Table 2 shows the times required for solution of large assignment and transportation problems. These are the same problems as the ones of Table VI of [BeT85]. Table 3 gives their characteristics.

## REFERENCES

[AMO89] Ahuja, R. K., Magnanti, T. L., and Orlin, J. B., "Network Flows", Sloan W. P. No. 2059-88, M.I.T., Cambridge, MA, March 1989 (also in Handbooks in Operations Research and Management Science, Vol. 1, Optimization, G. L. Nemhauser, A. H. G. Rinnooy-Kan, and M. J. Todd (eds.), North-Holland, Amsterdam, 1989).

[BeT85] 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; also Operations Research J., Vol. 36, 1988, pp. 93-114.

[BeT88] Bertsekas, D. P., and Tseng, P., "RELAX: A Computer Code for Minimum Cost Network

Prob. No.	RELAXT-III $\mu$ VAX	RELAXT-III MAC-II	RNET $\mu$ VAX	NETFLO $\mu$ VAX	NETFLO MAC-II	SSP MAC-II	RELAX-SSP MAC-II
1	1.22	1.21	2.94	2.47	2.78	9.00	4.07
2	0.97	1.17	3.32	2.18	2.35	7.53	3.83
3	1.54	1.57	4.80	3.22	3.27	10.02	5.52
4	1.28	1.58	5.16	3.12	3.40	11.88	6.67
5	1.79	2.18	5.73	3.82	4.30	13.92	6.27
6	2.03	2.48	8.20	6.09	6.67	26.53	12.32
7	2.76	3.12	11.23	7.76	8.48	35.93	13.37
8	2.91	3.60	13.58	7.90	9.03	30.42	14.42
9	2.83	3.23	16.69	9.42	9.23	24.00	14.48
10	2.08	3.67	14.57	8.08	10.25	40.67	16.00
11	0.82	1.18	4.32	4.47	4.83	2.82	1.13
12	0.98	1.25	6.23	4.91	5.83	2.58	1.10
13	1.36	1.18	6.70	4.36	6.33	4.30	1.35
14	1.42	1.83	8.79	6.37	7.88	4.18	2.13
15	1.88	2.17	8.65	6.34	8.20	4.50	2.98
16	1.76	2.40	3.26	2.75	2.92	6.18	4.10
17	1.98	2.70	3.40	3.06	3.13	10.43	3.38
18	1.25	1.58	2.57	2.36	2.53	5.95	3.23
19	3.08	4.02	3.30	2.64	3.05	9.30	7.03
20	2.47	2.93	3.33	3.60	3.25	6.90	6.38
21	2.77	3.43	4.53	4.29	4.82	9.28	7.23
22	2.58	2.90	3.08	2.58	2.67	5.65	5.28
23	2.53	2.68	3.41	2.45	2.82	7.78	7.37
24	1.11	1.28	2.59	3.24	3.75	3.58	1.75
25	1.73	2.08	5.11	4.71	5.25	6.38	3.32
26	1.35	1.52	2.26	2.19	2.37	1.40	1.13
27	1.06	1.97	2.69	1.87	3.65	3.75	1.93
28	4.61	5.25	6.07	9.72	10.15	17.67	13.28
29	4.16	4.10	8.12	11.32	10.65	18.98	13.08
30	4.77	5.47	8.16	12.53	13.37	24.08	10.28
31	3.13	3.70	9.84	14.02	15.22	29.88	15.38
32	6.82	7.70	12.97	17.02	17.87	52.72	32.47
33	5.47	6.25	14.22	22.21	23.28	36.13	31.92
34	6.09	6.90	14.74	25.07	26.82	34.83	26.48
35	6.13	6.38	15.40	22.31	28.92	43.57	32.30

Table 1: Solution times in secs for the first 35 problems standard NETGEN benchmark problems.

Problem Number	RELAXT-III $\mu$ VAX	RNET $\mu$ VAX	NETFLO $\mu$ VAX
1	3.27	65.28	62.52
2	5.67	124.14	131.72
3	7.02	279.08	226.24
4	10.56	64.64	75.20
5	11.79	193.29	165.67
6	17.68	96.63	100.29
7	36.84	202.52	213.23
8	43.51	378.49	398.51
9	86.44	731.52	677.12
10	102.67	745.10	891.40
11	21.32	118.18	114.31
12	33.71	255.07	244.63
13	44.06	426.97	443.11
14	62.17	650.37	685.78
15	110.38	973.72	991.75
16	9.05	60.64	42.29
17	16.71	119.31	138.13
18	29.65	210.31	214.34
19	46.38	313.09	354.53
20	51.12	485.55	303.67

Table 2: Solution times in secs for the assignment and transportation problems described in Table 3.

Flow Problems", Annals of Operations Research, Vol. 13, 1988, pp. 127-190.

[Ber82] Bertsekas, D. P., "A Unified Framework for Minimum Cost Network Flow Problems", Laboratory for Information and Decision Systems Report LIDS-P-1245-A, M.I.T., Cambridge, MA, 1982; also in Math. Programming, 1985, pp. 125-145.

[Chv83] Chvatal, V., Linear Programming, Freeman, N. Y., 1983.

[Dan63] Dantzig, G. B., Linear Programming and Extensions, Princeton Univ. Press, Princeton, N. J., 1963.

[GrH80] Grigoriadis, M. D., and Hsu, T., The Rutgers Minimum Cost Network Flow Subroutines (RNET documentation), Dept. of Computer Science, Rutgers Univ., N. J., Nov. 1980.

[KNS74] Klingman, D., Napier, A., and Stutz, "NETGEN - A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems", Man-

Problem No.	No. of Sources	No. of Sinks	No. of Arcs	Cost Range	Total Supply
1	1000	1000	8000	1-10	1000
2	1500	1500	12000	1-10	1500
3	2000	2000	16000	1-10	2000
4	1000	1000	8000	1-1000	1000
5	1500	1500	12000	1-1000	1000
6	1000	1000	8000	1-10	100000
7	1500	1500	12000	1-10	153000
8	2000	2000	16000	1-10	220000
9	2500	2500	20000	1-10	275000
10	3000	3000	24000	1-10	330000
11	1000	1000	8000	1-1000	100000
12	1500	1500	12000	1-1000	153000
13	2000	2000	16000	1-1000	220000
14	2500	2500	20000	1-1000	275000
15	3000	3000	24000	1-1000	330000
16	500	500	10000	1-100	15000
17	750	750	15000	1-100	15000
18	1000	1000	20000	1-100	30000
19	1250	1250	25000	1-100	37500
20	1500	1500	30000	1-100	45000

**Table 3:** Characteristics of the transportation problems solved. These problems were randomly generated using NETGEN. The first five are symmetric assignment problems.

agement Science, Vol. 20, 1974, pp. 814-822.

[KeH80] Kennington, J., and Helgason, R., Algorithms for Network Programming, J. Wiley, N. Y., 1980.

[PaS82] Papadimitriou, C. H., and Steiglitz, K., Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Englewood Cliffs, N. J., 1982.

[Roc84] Rockafellar, R. T., Network Flows and Monotropic Optimization, Wiley-Interscience, N. Y., 1984.