FLIGHT TRANSPORTATION LABORATORY
REPORT R 67-3

OUT-OF-KILTER FLOW (OKF): USER'S GUIDE

Amos Levin

1967

MIT

DEPARTMENT OF AERONAUTICS & ASTRONAUTICS

FLIGHT TRANSPORTATION LABORATORY
Cambridge, Mass. 02139

FLIGHT TRANSPORTATION LABORATORY REPORT R67-3

OUT-OF-KILTER FLOW (OKF):
USER'S GUIDE

Amos Levin

1967

The following few pages are from the book, "Flows in Networks" by Ford and Fulkerson, which is referenced in this report. After these few pages, the report itself commences.

Flows In Networks
Ford + Fulkerson

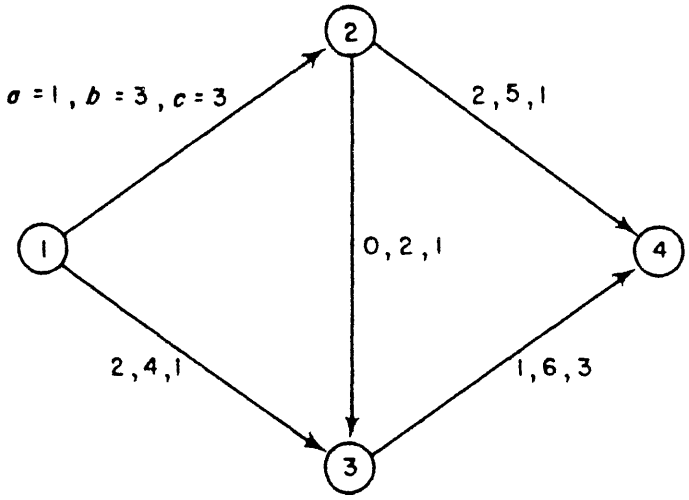


Figure 10.5

may go against one's intuition at first, but a little reflection shows that it is not, after all, surprising.

11. Constructing minimal cost circulations [28]

The method presented here for computing optimal network flows is more general than those described earlier in at least three ways:

(a) lower bounds as well as capacities are assumed for each arc flow, and are dealt with directly;
(b) the cost coefficient for an arc is arbitrary in sign;
(c) the method can be initiated with any circulation, feasible or not, and any set of node numbers.

(It is convenient to describe the computation in terms of circulations, rather than flows from sources to sinks.) The freedom to begin with any circulation and node numbers, instead of starting with particular ones which satisfy certain optimality properties, as has been the case before, is perhaps the most important practical feature of the method. For example, in actual applications, one is often interested in seeing what changes will occur in an optimal solution when some of the given data are altered. This method is tailored for such an examination, since the old optimal primal and dual solutions can be used to start the new problem, thereby greatly decreasing computation time.

An interesting feature of the method is that, loosely speaking, the status of no arc of the network is worsened at any step of the computation. We shall make this statement more precise later on.

§11. CONSTRUCTING MINIMAL COST CIRCULATIONS

We take the problem in circulation form. That is, we want to construct f that satisfies

$$f(x, N) - f(N, x) = 0, \qquad \text{all } x \in N, \tag{11.1}$$

$$l(x, y) \leqslant f(x, y) \leqslant c(x, y), \qquad \text{all } (x, y) \in \mathscr{A}; \tag{11.2}$$

and minimizes the linear cost function

$$\sum_{\mathscr{A}} a(x, y) f(x, y). \tag{11.3}$$

(Here $0 \leqslant l(x, y) \leqslant c(x, y)$, and as usual, we assume integral data.) Thus, if it is desired to construct a feasible flow from s to t of given value v that minimizes (11.3), one can merely add a return flow arc (t, s) with $l(t, s) = c(t, s) = v$, $a(t, s) = 0$, to get the problem in circulation form. Or, if it is desired to construct a maximal feasible flow from s to t that minimizes (11.3), one can take $l(t, s) = 0$, $c(t, s)$ large, $a(t, s)$ negatively large.

Of course feasible circulations may not exist. In this case the algorithm terminates with the location of a subset X of nodes for which the condition of Theorem II.3.1 is violated.

For given node numbers π, we set

$$\bar{a}(x, y) = a(x, y) + \pi(x) - \pi(y). \tag{11.4}$$

Then, for given π and circulation f, an arc (x, y) is in just one of the following states:

$$\begin{aligned}
&(\alpha)\ \bar{a}(x, y) > 0,\ f(x, y) = l(x, y),\\
&(\beta)\ \bar{a}(x, y) = 0,\ l(x, y) \leqslant f(x, y) \leqslant c(x, y),\\
&(\gamma)\ \bar{a}(x, y) < 0,\ f(x, y) = c(x, y),\\
&(\alpha_1)\ \bar{a}(x, y) > 0,\ f(x, y) < l(x, y),\\
&(\beta_1)\ \bar{a}(x, y) = 0,\ f(x, y) < l(x, y),\\
&(\gamma_1)\ \bar{a}(x, y) < 0,\ f(x, y) < c(x, y),\\
&(\alpha_2)\ \bar{a}(x, y) > 0,\ f(x, y) > l(x, y),\\
&(\beta_2)\ \bar{a}(x, y) = 0,\ f(x, y) > c(x, y),\\
&(\gamma_2)\ \bar{a}(x, y) < 0,\ f(x, y) > c(x, y).
\end{aligned}$$

We say that an arc is *in kilter* if it is in one of the states α, β, γ; otherwise the arc is *out of kilter*. Thus to solve the problem, it suffices to get all arcs in kilter, since optimality properties are

$$\bar{a}(x, y) < 0 \Rightarrow f(x, y) = c(x, y), \tag{11.5}$$

$$\bar{a}(x, y) > 0 \Rightarrow f(x, y) = l(x, y). \tag{11.6}$$

With each state that an arc (x, y) can be in, we associate a non-negative number, called the *kilter number* of the arc in the given state. An in-kilter

arc has kilter number 0; the arc kilter numbers corresponding to each out-of-kilter state are listed below:

$$
\begin{aligned}
(\alpha_1) \text{ or } (\beta_1)&: \; l(x, y) - f(x, y), \\
(\gamma_1)&: \; \bar{a}(x, y)[f(x, y) - c(x, y)], \\
(\alpha_2)&: \; \bar{a}(x, y)[f(x, y) - l(x, y)], \\
(\beta_2) \text{ or } (\gamma_2)&: \; f(x, y) - c(x, y).
\end{aligned}
$$

Thus out-of-kilter arcs have positive kilter numbers. The kilter numbers for states α_1, β_1, β_2, γ_2 measure infeasibility for the arc flow $f(x, y)$, while the kilter numbers for states γ_1, α_2 are, in a sense, a measure of the degree to which the optimality properties (11.5), (11.6) fail to be satisfied.

The algorithm concentrates on a particular out-of-kilter arc and attempts to put it in kilter. It does this in such a way that all in-kilter arcs stay in kilter, whereas the kilter number for any out-of-kilter arc either decreases or stays the same. Thus all arc kilter numbers are monotone non-increasing throughout the computation. (This is the interesting feature of the method that was mentioned previously.) However, steps can occur that change no kilter number, and this somewhat complicates the proof of termination. But if the process begins with a feasible circulation, the monotone property is stronger: at least one arc kilter number decreases at each step, thus providing a simpler proof of finiteness in this case.

A basic notion underlying the method is to utilize the labeling process of II.3, modified appropriately, for increasing or decreasing a particular arc flow in a circulation. The appropriate modification this time will not be in terms of the notion of "admissibility" for an arc, used previously, but will rather be more general.

We now state the algorithm.

The out-of-kilter algorithm.† Enter with any integral circulation f and any set of node integers π. Next locate an out-of-kilter arc (s, t) and go on to the appropriate case below.

(α_1) $\bar{a}(s, t) > 0$, $f(s, t) < l(s, t)$. Start a labeling process at t, trying to reach s, first assigning t the label $[s^+, \varepsilon(t) = l(s, t) - f(s, t)]$. The labeling rules are:

(11.7) If x is labeled $[z^{\pm}, \varepsilon(x)]$, y is unlabeled, and if (x, y) is an arc such that either

(a) $\bar{a}(x, y) > 0, f(x, y) < l(x, y)$,

(b) $\bar{a}(x, y) \leqslant 0, f(x, y) < c(x, y)$,

† An IBM 704 code based on this algorithm has been prepared by J. D. Little. A FORTRAN-FAP revision for the IBM 7090 has been written by R. Clasen. This code, identified as RS OKF1, is available through SHARE. A sample problem involving 2900 arcs and 775 nodes required 1139 breakthroughs, 411 non-breakthroughs. Total computing time was 5 minutes, exclusive of an input-output time of 3.2 minutes.

then y receives the label $[x^+, \varepsilon(y)]$, where

$$\varepsilon(y) = \min [\varepsilon(x), l(x, y) - f(x, y)] \text{ in case (a),}$$
$$\varepsilon(y) = \min [\varepsilon(x), c(x, y) - f(x, y)] \text{ in case (b).}$$

(11.8) If x is labeled $[z^{\pm}, \varepsilon(x)]$, y is unlabeled, and if (y, x) is an arc such that either

(a) $\bar{a}(y, x) \geqslant 0, f(y, x) > l(y, x)$,
(b) $\bar{a}(y, x) < 0, f(y, x) > c(y, x)$,

then y receives the label $[x^-, \varepsilon(y)]$, where

$$\varepsilon(y) = \min [\varepsilon(x), f(y, x) - l(y, x)] \text{ in case (a),}$$
$$\varepsilon(y) = \min [\varepsilon(x), f(y, x) - c(y, x)] \text{ in case (b).}$$

If breakthrough occurs (that is, s receives a label), so that a path from t to s has been found, change the circulation f by adding $\varepsilon(s)$ to the flow in forward arcs of this path, subtracting $\varepsilon(s)$ from the flow in reverse arcs, and finally adding $\varepsilon(s)$ to $f(s, t)$. If non-breakthrough, let X and $\overline{X}$ denote labeled and unlabeled sets of nodes, and define two subsets of arcs:

(11.9) $$\mathscr{A}_1 = \{(x, y) | x \in X, y \in \overline{X}, \bar{a}(x, y) > 0, f(x, y) \leqslant c(x, y)\},$$

(11.10) $$\mathscr{A}_2 = \{(y, x) | x \in X, y \in \overline{X}, \bar{a}(y, x) < 0, f(y, x) \geqslant l(y, x)\}.$$

Then let

(11.11) $$\delta_1 = \min_{\mathscr{A}_1} [\bar{a}(x, y)],$$

(11.12) $$\delta_2 = \min_{\mathscr{A}_2} [-\bar{a}(y, x)],$$

(11.13) $$\delta = \min (\delta_1, \delta_2).$$

(Here δ_t is a positive integer or ∞ according as $\mathscr{A}_t$ is non-empty or empty.) Change the node integers by adding δ to all $\pi(x)$ for $x \in \overline{X}$.

(β_1) or (γ_1). $\bar{a}(s, t) = 0$, $f(s, t) < l(s, t)$ or $\bar{a}(s, t) < 0$, $f(s, t) < c(s, t)$. Same as (α_1), except $\varepsilon(t) = c(s, t) - f(s, t)$.

(α_2) or (β_2). $\bar{a}(s, t) > 0$, $f(s, t) > l(s, t)$, or $\bar{a}(s, t) = 0$, $f(s, t) > c(s, t)$. Here the labeling process starts at s, in an attempt to reach t. Node s is assigned the label $[t^-, \varepsilon(s) = f(s, t) - l(s, t)]$. The labeling rules are (11.7) and (11.8) again. If breakthrough, change the circulation by adding and subtracting $\varepsilon(t)$ to arc flows along the path from s to t; then subtract $\varepsilon(t)$ from $f(s, t)$. If non-breakthrough, change the node numbers as above.

(γ_2). $\bar{a}(s, t) < 0$, $f(s, t) > c(s, t)$. Same as (α_2) or (β_2), except $\varepsilon(s) = f(s, t) - c(s, t)$.

The labeling process is repeated for the arc (s, t) until either (s, t) is in kilter, or until a non-breakthrough occurs for which $\delta = \infty$. In the latter

case, stop. (There is no feasible circulation.) In the former case, locate another out-of-kilter arc and continue.

We show that the out-of-kilter algorithm terminates, and that all arc kilter numbers are monotone non-increasing throughout the computation.

Suppose that arc (s, t) is out of kilter, say in state α_1. The origin for labeling is t, the terminal s. The arc (s, t) cannot be used to label s directly since neither (11.8a) nor (11.8b) is applicable. Consequently, if breakthrough occurs, the resulting path from t to s, together with the arc (s, t), is a cycle. Then the flow changes that are made on arcs of this cycle again yield a circulation. Moreover, the labeling rules have been selected in such a way that kilter numbers for arcs of this cycle do not increase, and at least one, namely, for arc (s, t), decreases by a positive integer. Kilter numbers for arcs not in the cycle of course do not change.

Similar remarks apply if (s, t) is in one of the other out-of-kilter states.

We summarize the possible effects of a breakthrough on an arc (x, y) in Fig. 11.1, which shows the state transitions that may occur following breakthrough. If a transition is possible, the number recorded beside the corresponding arrow represents the change in kilter number. (Here ε is the flow change.)

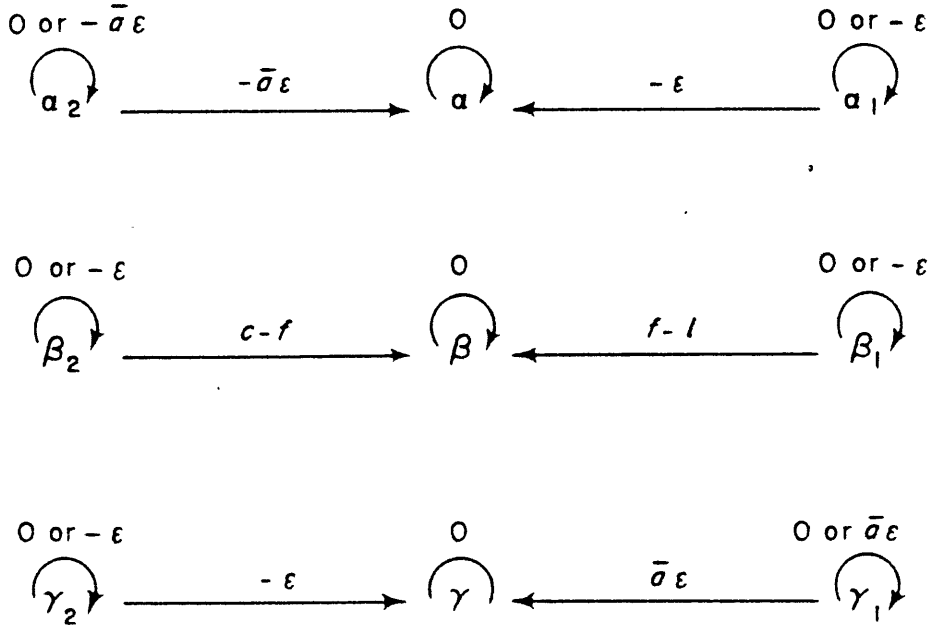


Figure 11.1 Breakthrough diagram

Verification of the breakthrough diagram is straightforward. For example, suppose arc (x, y) is in state α_2, with $\bar{a}(x, y) > 0, f(x, y) > l(x, y)$, and kilter number $\bar{a}(x, y)\,[f(x, y) - l(x, y)] > 0$. If (x, y) is not an arc of the cycle of flow changes, then (x, y) remains in state α_2 with zero change in

kilter number. If the flow in arc (x, y) has changed as a result of the breakthrough, then either (x, y) is the arc (s, t) or, by the labeling rules, (x, y) is a reverse arc of the path from origin to terminal. Specifically, x was labeled from y using (11.8a). In either case, $f(x, y)$ decreases by the positive integer $\varepsilon \leqslant f(x, y) - l(x, y)$, the new state for (x, y) is α_2 or α, and hence the kilter number for (x, y) has decreased by $\varepsilon\bar{a}(x, y) > 0$. The rest of the diagram may be verified similarly.

The state transitions and changes in kilter number that may occur following a non-breakthrough with $\delta < \infty$ are indicated in Fig. 11.2.

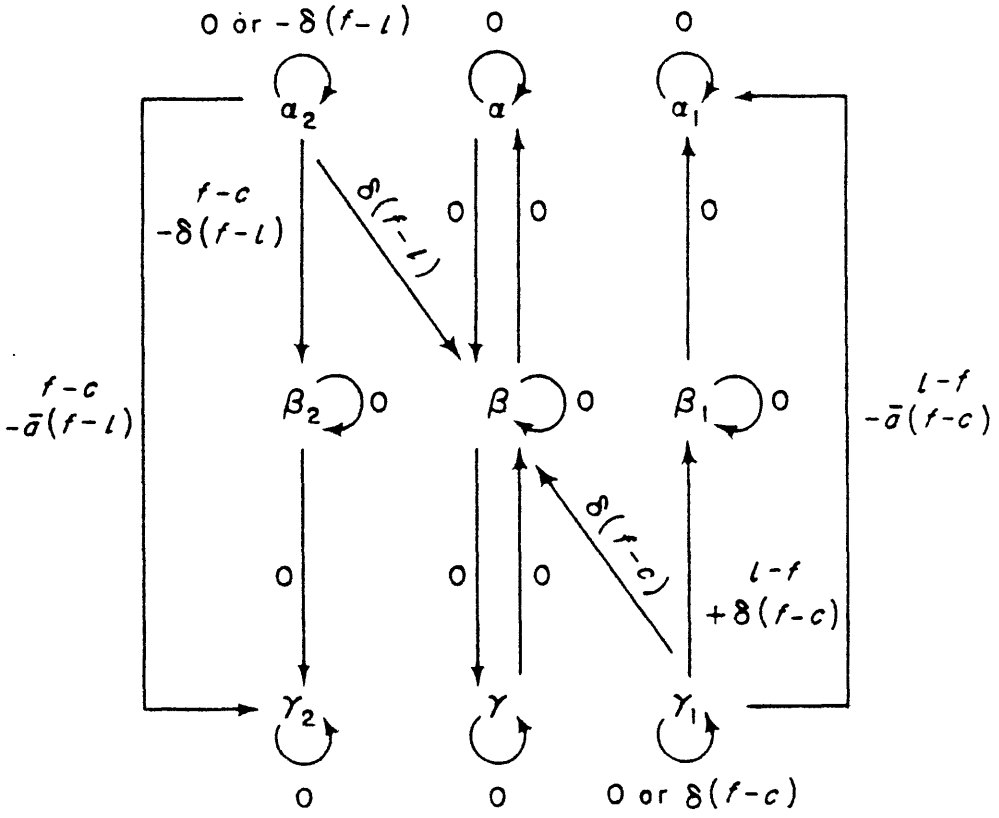


Figure 11.2 Non-breakthrough diagram

Again we omit a detailed verification, but consider, for example, an arc (x, y) in state γ_1, so that $\bar{a}(x, y) < 0$, $f(x, y) < c(x, y)$, having kilter number $\bar{a}(x, y)\,[f(x, y) - c(x, y)] > 0$ before the node number change is made. If both x and y are in X or both in $\overline{X}$, then $\bar{a}(x, y)$ remains the same after the node number change, and consequently (x, y) stays in state γ_1 with no change in kilter number. We cannot have x in X, y in $\overline{X}$ (labeling rule (11.7b)), and hence the remaining possibility is x in $\overline{X}$, y in X. Then $\bar{a}(x, y)$ is increased by $\delta > 0$. Consequently the arc (x, y) either remains in state γ_1 (if $\delta < -\bar{a}(x, y)$), goes into state β (if $\delta = -\bar{a}(x, y)$ and $f(x, y) \geqslant l(x, y)$), into state β_1 (if $\delta = -\bar{a}(x, y)$ and $f(x, y) < l(x, y)$), or into state

α_1 (if $\delta > -\bar{a}(x, y)$ and $f(x, y) < l(x, y)$), and the corresponding changes in kilter number are respectively

$$\begin{aligned} \delta[f(x, y) - c(x, y)] &< 0, \\ \delta[f(x, y) - c(x, y)] &< 0, \\ l(x, y) - f(x, y) + \delta[f(x, y) - c(x, y)] &\leqslant 0, \\ l(x, y) - f(x, y) - \bar{a}(x, y)\,[f(x, y) - c(x, y)] &\leqslant 0. \end{aligned}$$

(The remaining logical possibility $\delta > -\bar{a}(x, y)$, $f(x, y) \geqslant l(x, y)$ cannot occur, since if $f(x, y) \geqslant l(x, y)$, then (x, y) is in $\mathscr{A}_2$ defined by (11.10) and hence $\delta \leqslant -\bar{a}(x, y)$.)

It follows from the breakthrough and non-breakthrough diagrams that kilter numbers are monotone non-increasing throughout the computation. Moreover, if breakthrough occurs, at least one arc kilter number decreases by a positive integer. Thus to establish termination, it suffices to show that an infinite sequence of consecutive non-breakthroughs, each with $\delta < \infty$, is impossible. To show this, let us suppose that a labeling resulting in non-breakthrough with $\delta < \infty$ has occurred, and let X, $\overline{X}$ denote labeled and unlabeled sets of nodes. After changing the node numbers, the new function $\bar{a}'(x, y)$ is given by

$$\bar{a}'(x, y) = \begin{cases} \bar{a}(x, y) - \delta & \text{for } x \text{ in } X,\ y \text{ in } \overline{X}, \\ \bar{a}(x, y) + \delta & \text{for } x \text{ in } \overline{X},\ y \text{ in } X, \\ \bar{a}(x, y) & \text{otherwise.} \end{cases} \tag{11.14}$$

If the arc (s, t) is still out of kilter, then the origin is the same for the next labeling, and it follows from (11.14) and the labeling rules that every node of X will again be labeled. Thus if the new labeling results in non-breakthrough with labeled set X', we have $X \subseteq X'$. Let $\mathscr{A}_1'$, $\mathscr{A}_2'$ denote the new sets defined in terms of X', $\bar{a}'$, and f by (11.9), (11.10), and suppose $X = X'$. Then from (11.14) we have $\mathscr{A}_1' \subseteq \mathscr{A}_1$, $\mathscr{A}_2' \subseteq \mathscr{A}_2$, and at least one of these inclusions is proper by (11.11), (11.12), (11.13). Hence the new labeling either assigns a label to at least one more node, or failing this, an arc is removed from one of the sets $\mathscr{A}_1$ or $\mathscr{A}_2$. It follows that, after finitely many non-breakthroughs with $\delta < \infty$, we either get the arc (s, t) in kilter, obtain a breakthrough, or obtain a non-breakthrough with $\delta = \infty$.

If a non-breakthrough with $\delta = \infty$ occurs, there is no feasible circulation. For if $\delta = \infty$, then from the definitions of $\mathscr{A}_1$, $\mathscr{A}_2$ and the labeling rules, we have $f(x, y) \geqslant c(x, y)$, $f(y, x) \leqslant l(y, x)$ for $x \in X$, $y \in \overline{X}$. Moreover, for the arc (s, t), either t is in X, s in $\overline{X}$ with $f(s, t) < l(s, t)$, or s is in X, t in $\overline{X}$ with $f(s, t) > c(s, t)$. (This is immediate for cases $\alpha_1, \beta_1, \beta_2, \gamma_2$ of the algorithm, and follows from (11.9) and the assumption $\delta = \infty$ for case α_2, from (11.10) and the assumption $\delta = \infty$ for case γ_1.) Hence,

summing the conservation equations (11.1) over x in X, we obtain in all cases

$$0 = f(X, \overline{X}) - f(\overline{X}, X) > c(X, \overline{X}) - l(\overline{X}, X).$$

But this violates the feasibility condition of Theorem II.3.1. Thus $\delta = \infty$ implies there is no feasible circulation.

THEOREM 11.1. *The out-of-kilter algorithm either solves the problem* (11.1), (11.2), (11.3) *in finitely many applications of the labeling process or terminates with the conclusion that no feasible circulation exists. All arc kilter numbers are monotone non-increasing throughout the computation. In addition, if the algorithm is initiated with a feasible circulation, at least one arc kilter number decreases with each labeling.*

The only part of Theorem 11.1 that remains to be checked is the last assertion. If the computation begins with a feasible circulation, the states α_1, β_1, β_2, γ_2 are empty to begin with, and consequently remain empty through the computation. Hence, at each non-breakthrough (as well as each breakthrough), the kilter number for at least one arc, namely (s, t), decreases by a positive integer.

It is worth while to point out how the out-of-kilter algorithm generalizes the method of § 3 for constructing a maximal flow from source s to sink t that minimizes cost over all maximal flows. Here we suppose $l = 0, a \geqslant 0$, as in § 3. Now add the arc (t, s) to the network with $l(t, s) = 0$, $c(t, s)$ large, and $a(t, s)$ negatively large. If we start with the zero circulation and all node numbers zero, as in § 3, then the only out-of-kilter arc is (t, s) (it is in state γ_1) and hence it remains the only out-of-kilter arc throughout the computation. Then the origin for the labeling process is always s, the terminal t, and the labeling rules, flow change, and node number change all reduce to those of § 3.

References

1. R. Bellman, "On a Routing Problem," *Quart. Appl. Math.* 16 (1958), 87–90.
2. C. Berge, *Theorie des Graphes et ses Applications*, Dunod, Paris, 1958.
3. R. G. Busacker and P. J. Gowen, "A Procedure for Determining a Family of Minimal-Cost Network Flow Patterns," O.R.O. Technical Paper 15, 1961.
4. A. S. Cahn, "The Warehouse Problem," *Bull. Amer. Math. Soc.* 54 (1948), 1073 (abstract).
5. T. F. Cartaino and S. E. Dreyfus, "Application of Dynamic Programming to the Airplane Minimum Time-to-climb Problem," *Aero. Engr. Rev.* 16 (1957), 74–77.
6. A. Charnes and W. W. Cooper, "Generalizations of the Warehousing Model," *Op. Res. Quart.* 6 (1955), 131–172.

FLIGHT TRANSPORTATION LABORATORY

DEPARTMENT OF AERONAUTICS AND ASTRONAUTICS
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS 02139

20 May 1976

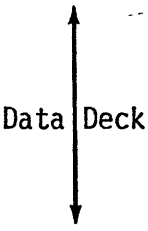
MEMORANDUM

Re: OKF Control Cards

From: Bob Mann, x3-7571

The new control cards for OKF runs are now as follows:

```
// 'program name',CLASS=B,REGION=128K
/*MITID USER=(M3865,Pnnnnn,,,password)
/*MAIN TIME=time, LINES=lines,CARDS=cards
/*SRI LOW
//  EXEC  PGM=GO
//STEPLIB  DD  DSNAME=U.M3865.P11724.OKFLOAD.COMMAND,DISP=SHR
//FT05F001  DD  DDNAME=SYSIN
//FT06F001  DD  SYSOUT=A
//FT07F001  DD  SYSOUT=B
//SYSIN  DD *
```




```
/*
```

This where:

'program name' is any identifying name

Pnnnnn is your programmer number

password is your password

time is the estimated CPU run time in minutes (usually less than 1)

lines is the estimated number of lines output (usually less than 2 times the number of arcs per job step) in thousands

cards are the estimated number of cards output in hundreds (only used if the OUTPUT PUNCH option is to be used)

Flight Transportation
Laboratory, M.I.T.

Control Cards for Out-of-Kilter Runs

```
// 'name',REGION=128K,CLASS=B
/*MITID USER=(M3865,progno, password)
/*MAIN TIME=time,LINES=lines,CARDS=cards
/*SRI LOW
//  EXEC FORG,PROG='U.M3865.8856.LOADOK.OK(OKF)'
//G.SYSIN DD *
```

```
      data cards
```



```
/*
```

name= your name
progno= your programmer number
time= estimated time for the run in minutes
lines= number of lines printed in thousands
cards= number of cards punched in hundreds

OKF currently can handle up to 500 nodes and 2000 arcs.

MIT Flight Transportation Laboratory

IBM /360 OUT OF KILTER NETWORK FLOW ROUTINE

DESCRIPTION FOR THE USER

Table of Contents

1. Introductory Notes

This writeup is intended for the user of the "Out of Kilter" program which has been written for the IBM system 360 model 65. The program has been successfully run at the MIT Computation Center.

Both the program and the writeup are based on the SHARE routine RS OKF1 and its corresponding writeup.

The FORTRAN subprograms are written in FORTRAN IV (G level). The assembly language subprograms use the extended mnemonic branching instruction codes and the macros SAVE and RETURN.

2. Formulation

A Computer routine for the solution of "network flow" programs -- problems of finding those flows of an homogeneous commodity through a capacitated network minimizing the sum of the linear costs of flow through each arc -- is herein described. The computational algorithm employed is described in the book "Flows in Networks",L.R. Ford and D.R. Fulkerson, Princeton University Press, 1962, pp.162-169.

The network in question consists of nodes designated by i or j, and a certain collection of arcs joining pairs of nodes. The arc $\widehat{ij}$ is thought of as directed from i to j. With each arc in the network is associated the following four integer quantities.

c_{ij} the cost of one unit of flow from i to j along arc $\widehat{ij}$;

u_{ij} the upper bound on the amount of flow along the arc $\widehat{ij}$;

ℓ_{ij} the lower bound on the amount of flow along the arc $\widehat{ij}$;

x_{ij} the quantity of flow along the arc $\widehat{ij}$

The network flow problem is that of determining x_{ij} (for all arcs $\widehat{ij}$ of the network) such that

(1) $\ell_{ij} \leq x_{ij} \leq u_{ij}$ (all arcs $\widehat{ij}$),

(2) the net flow into any node (generally zero) remains fixed throughout the solution of the problem, and

(3) $\sum_{\widehat{ij}} c_{ij}x_{ij}$ is minimized

3. Data

Data Format

A node may be represented by any combination of six Hollerith characters (at least one of which is neither zero nor blank); i and j below are such combinations. (Note that for node names a blank is a character, and different from a zero.) The numerical data above are represented as right-justified integers in the appropriate fields. All data pertaining to one arc are entered on one card as follows:

1..6	7..12	13..18	19,20	21..30	31..40	41..50	51..60	61..80
blank	i	j	free to use	c_{ij}	u_{ij}	ℓ_{ij}	x_{ij}	free to use

Leading zeros in the numeric fields need not be entered, nor need any figures where zero is desired. Node names must be left justified and must differ in the first four characters.

Of course, fields 7-50 contain constants for the stated problem. Entry of the "x_{ij}" is optional, constituting only an initial guess at the solution.

An optional initial set of node prices π_i may be entered. These are entered one per card as follows:

1 .. 6	7 .. 12	13 ... 20	21 ... 30	31 80
blank	i	free to use	π_i	free to use

Assembly of Data

The data just described is put together in the following way:

1) All arcs $\widehat{ij}$ having a given first node i must be adjacent in the deck. (No other requirement on their order is made.)

2) The arc cards are preceded by two cards, the first being the title card and the second bearing the word "ARCS" in the field 1-4. The title card should be blank in column 1 and may have any Hollerith punches in columns 2-80.

3) If no node prices are given, the arc cards are followed by a card bearing "END" in 1-3.

4) If node prices are given,the arcs are followed by a card bearing "NODES" in 1-5; the node cards follow this, and all the cards are followed by the END card of (3).

4. Control Cards for Standard Run

Input, computation, and output are effected by control cards whose punching in the field 1-12 controls the operation of the routine. Punching always begins in column 1, and there is one blank between English words. The first card of the deck which follows the program deck must be the control card

READY.

Following the "READY" card must be one of the two control cards

CARDS or TAPE.

If "TAPE", the assembled data described in the previous section should be on the reserved input tape. If "CARDS", the assembled data should immediately follow this control card.

Next may be placed any combination of the three output control cards

OUTPUT TAPE
OUTPUT PRINTER
OUTPUT PUNCH

which will cause the types of output described in Section 8. At least one OUTPUT control card must be included in the data set.

Next is placed the card

COMPUTE

which causes computation to begin.

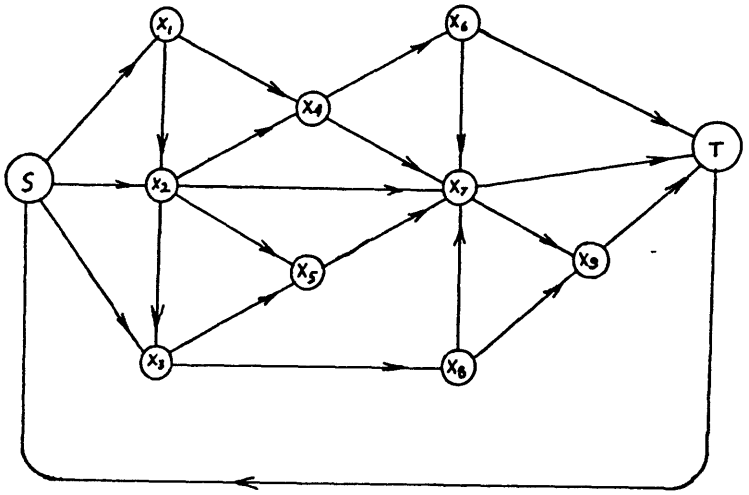
The last card in the deck must be the control card

PAUSE

which terminates the job.

5. Example

The example which follows is a modification of the one given in the book " Flows In Networks", L.R. Ford and D.R. Fulkerson, Princeton University Press, 1962, pp. 123-127. Costs and bounds for the arcs can be found in the data listing on the next page. Since the cost on the arc $\overset{\frown}{T\ S}$ is very low (negative) compared to the costs on the other arcs, the routine finds the maximal flow that minimizes costs from S to T.



```
READY
CARDS
   F. AND F. EXAMPLE 1
ARCS   i      j          Cij          Uij          lij          Xij
       S      X1           3           50           35            0
       S      X2           6           30            0            0
       S      X3           8           15            0            0
       X1     X2           2           50            0            0
       X1     X4           2           25            0            0
       X2     X3           2           15            0            0
       X2     X4           1           45            0            0
       X2     X5           3           10           10            0
       X2     X7           8           15            0            0
       X3     X5           1           10            0            0
       X3     X8           3           20            0            0
       X4     X6           9           90            0            0
       X4     X7           8           10            0            0
       X5     X7           5           60            0            0
       X6     X7           1           10            7            0
       X6     T            2           10            0            0
       X7     X9           1           10            0            0
       X7     T            4           80            0            0
       X8     X7           2           20            0            0
       X8     X9           3           10            0            0
       X9     T            3           10            0            0
       T      S       -10000           85           25            0
END
OUTPUT PRINTER
OUTPUT TAPE
COMPUTE
PAUSE
```

6. Jobs with More than One Run.

The control card setup described in Section 3 applies to jobs with only one run. By a "job", we mean all that is done in one pass at the computer; that is, any work that can be done without manual interference with the computer and, in addition, without inputting the program instructions into the computer more than once. By a "run", we mean that which is involved in the solution of one problem.

For multiple run jobs, the standard input for each run is as described in Section 3 with the "PAUSE" card removed. Runs may be stacked one after another. Only one "PAUSE" card may be used, and it is always placed after the "COMPUTE" card of the final run.

Each run begins with a "READY" card or a "SAVE" card as described in Section 6. Each run ends with a "COMPUTE" card. The job ends with a "PAUSE" card.

7. Save and Alter Run

In Section 3, the standard run beginning with the " READY" card was described. In Section 5, it was noted that these runs may be stacked, one after another. Frequently it is desired to execute a run in which only relatively few c_{ij}, u_{ij} or ℓ_{ij} are changed, but in which the arc configuration remains the same. In this event, a "Save and Alter" precedure may be followed. A "Save and Alter" run may be any run except the first. The control card setup for this type of run is as follows.

The first card of the run must be the control card

SAVE

which initiates a new run without destroying the results of the previous run.

The second card is the title card, which may have any Hollerith punches in it, except that column 1 should be blank.

Next are placed the "OUTPUT" cards as mentioned in Section 3 and described in Section 8.

Next are placed any number of "ALTER" cards. Each "ALTER" card has the following format:

1...6	7..12	13..18	19,20	21..30	31..40	41..50	51..60	61..80
ALTER	i	j	n_{ij}	c_{ij}	u_{ij}	ℓ_{ij}	Δf_{ij}	free to use

i and j are the source and sink nodes of an arc which is in core storage; that is, one which was used on the preceding "READY" run. n_{ij} may be left blank if there is only one arc $\overset{\frown}{ij}$. If there is more than one arc $\overset{\frown}{ij}$, then n_{ij} gives the number of this arc as to whether it was the 1st, 2nd, 3rd, etc. arc $\overset{\frown}{ij}$ which was read into memory in the applicable "READY" run. c_{ij}, u_{ij} and ℓ_{ij} are the new values of these same quantities for this arc.

Δf_{ij} is usually zero (or blank). It is the change in the flow out of node i and into node j. Note that inputting a new x_{ij} is meaningless, since x_{ij} on input is a guess, and guessing a different value of x_{ij} on an alter run would only upset the conservation of flow from the nodes. Hence inputting a non-zero Δf_{ij} is a means of deliberately upsetting the flow conservation. It will change x_{ij} to $x_{ij} + \Delta f_{ij}$.

The last card of the run must be the control card

COMPUTE

which causes computation to begin.

Note that any number of "Save and Alter" runs may follow one "READY" run. The effects of each "Save and Alter" are cumulative.

The program also allows "ALTER" cards to be placed after the "OUTPUT" cards and before the "COMPUTE" card on a "READY" run. This "Ready and Alter" run is useful when data is on tape and a few changes in the value of c, u, and ℓ are needed before the run is to be executed.

8. Other Program Options

In the standard run, the program requires that every node be a first node for some arc and be a second node for some other arc. This is the standard network problem. Another type of problem allows arcs to end at nodes at which no arcs begin. These arcs are desginated by the program as "dead end arcs." Also we may have nodes on which no arc terminates. The type of problem for which the requirement that each node begin some arc and end some arc is not enforced by the program is designated a "transportation" problem.

The reserved input tape may have data for several jobs stacked on it. There are no ends of file on this tape except at the end of all data; the program knows when it is at the end of the data for one run by sensing the "END" card record. In certain cases, it may be desirable to pass over some data packages while processing a job. In this event, the control card "SKIP" is used.

The general "READY" type run is now described.
The first card must be the control card

READY.

An optional card which must follow the "READY" card if this is a transportation problem, is the control card

TRANSPORTATION.

Also optional is the control card

SKIP

which is used to cause the reserved input tape to skip one package of assembled data. As many "SKIP" cards are used as are needed to skip the desired number of packages of assembled data. The "SKIP" cards and the "TRANSPORTATION" card may be in any order immediately following the "READY" card.

Following the above cards must be one of the two control cards

CARDS or TAPE.

These cards are as described in Section 3.

The data package follows the "CARDS" control card. Following the data package, or the control card "TAPE" where there is no data package with the control cards, may be an optional title card. If this is included, it supersedes the title card on the data package.

Next may be placed any number of "OUTPUT" cards as described in Section 8.

Next may be placed any number of "ALTER" cards as described in Section 6.

The last card in the run must be the control card

COMPUTE

which causes computation to begin.

9. Output

The type of output is controlled by one or more of four control cards. The four control cards are

a) OUTPUT PRINTER
b) OUTPUT TAPE
c) OUTPUT PUNCH
d) OUTPUT NODES

The "OUTPUT PRINTER" control card causes output to be written on the system output device.This output is written for printing on the peripheral printer under program control. The system output device is denoted in the program by the symbol "KO", and KO has the value 6 in the version of the program submitted. The data for each arc are printed horizontally on the page. The data for one arc, $\widehat{ij}$, are printed in the following order:

1) node name i
2) node name j
3) c_{ij}, the unit cost of arc $\widehat{ij}$
4) u_{ij}, the upper bound of the quantity of flow through arc $\widehat{ij}$
5) ℓ_{ij}, the lower bound of the qunatity of flow through arc $\widehat{ij}$
6) x_{ij}, the quantity of flow in the arc $\widehat{ij}$
7) "FLOW" $= c_{ij}\, x_{ij}$, the total cost of x_{ij} units at the cost c_{ij}
8) π_i, the node price of node i
9) π_j, the node price of node j
10) $\overline{c}_{ij}$, the quantity $\pi_i + c_{ij} - \pi_j$
11) The letter "K", the letter "N" or nothing. The letter "K" is printed if all the arcs are in kilter. The letter "N" is printed if this arc could not be brought into kilter,

indicating that the problem has
no feasible solution. Nothing is
printed in all other cases.

The "OUTPUT TAPE" control card causes output to be written on the reserved output tape. This output may be printed peripherally using single space (or double space) control. It may also be punched peripherally, and the cards gotten thereby will be substantially the same as the cards gotten from the "OUTPUT PUNCH" option described below. The information from the "OUTPUT TAPE" option is the same as that from the "OUTPUT PRINTER" option, except that items 8), 9), and 10), are not output. This output is compatible with the input "TAPE" option.

The "OUTPUT PUNCH" option gives items 1) through 7) on the on-line punch. This option is generally very time consuming except on short problems.

Any of the above three options may be used in combination on any one problem. At least one OUTPUT control card must be included in each data set.

The "OUTPUT NODES" option will output a list of node prices in addition to the arc information on the tape or punch options. This option will have no effect on the printer output option.

All of the output on the reserved output tape and on the punch is compatible with the input to the problem. The "OUTPUT PRINTER" output is not compatible with the input.

In addition to the above, all control card information is written on the peripheral printer device, with the execption of the

"COMPUTE" control card for which is substituted a count of the arcs and the nodes. The messages in Section 9 are all written on the system output device also.

On the following two pages are shown the "OUTPUT PRINTER" results of the example given in Section 4. "Flow" is $c_{ij}\ x_{ij}$. "Total system contribution" is the optimal value of the objective function $\sum c_{ij}\ x_{ij}$. Note that the first page contains information that would be on the system output device regardless of whether "OUTPUT PRINTER" is requested.

```
READY
CARDS
   F. AND F. EXAMPLE 1
ARCS
OUTPUT PRINTER
OUTPUT TAPE
NO OF ARCS=    22 NO OF NODES=    11
THIS RUN OUTPUT TO TAPE
```

F. AND F. EXAMPLE 1

ARCS		COST	UPPER	LOWER	X	FLOW	PI1	PI2	CBAR
S	X1	3	50	35	50	150	13	17	-1 K
S	X2	6	30	0	20	120	13	19	0 K
S	X3	8	15	0	15	120	13	24	-3 K
X1	X2	2	50	0	25	50	17	19	0 K
X1	X4	2	25	0	25	50	17	20	-1 K
X2	X3	2	15	0	15	30	19	24	-3 K
X2	X4	1	45	0	5	5	19	20	0 K
X2	X5	3	10	10	10	30	19	25	-3 K
X2	X7	8	15	0	15	120	19	30	-3 K
X3	X5	1	10	0	10	10	24	25	0 K
X3	X8	3	20	0	20	60	24	28	-1 K
X4	X6	9	90	0	20	180	20	29	0 K
X4	X7	8	10	0	10	80	20	30	-2 K
X5	X7	5	60	0	20	100	25	30	0 K
X6	X7	1	10	7	10	10	29	30	0 K
X6	T	2	10	0	10	20	29	34	-3 K
X7	X9	1	10	0	0	0	30	31	0 K
X7	T	4	80	0	75	300	30	34	0 K
X8	X7	2	20	0	20	40	28	30	0 K
X8	X9	3	10	0	0	0	28	31	0 K
X9	T	3	10	0	0	0	31	34	0 K
T	S	-10000	85	25	85	-850000	34	13	-9979 K

END

TOTAL SYSTEM CONTRIBUTION = -848525

NO OF BREAKTHRUS= 12, NO OF NONBREAKTHRUS= 11, NO OF X CHANGES= 67
NO OF NODES FROM WHICH LABELING WAS DONE= 131

PAUSE

RESERVED TAPE HAS BEEN WRITTEN

10. Program Messages

One exception to the previous formats is permitted. If the "READY" of "SAVE" card is not the first card in a run this is not considered to be an error, but it is assumed that these are comment cards. The contents of columns 7-72 of all cards in a run (if any) which precede the "READY" or "SAVE" card plus columns 7-72 of the "READY" or "SAVE" card itself are written on the system output device.Thus only columns 1-6 of the "READY" and the "SAVE" card are fixed in format, the rest of the card may be used for comments. The above is also applicable to the "PAUSE" card.

Below is given a list of comments which may be written on the system output device.

Comments 3),4), 5), 6), 7), 8), 9), 12), and 13), denote errors in data set-up that were caught by the pre-processing routines. Conditions 10) and 11) are considered to be errors only if no "TRANSPORTATION" control card was present. Whenever any of the above error conditions are present, the run is terminated.

Comment 18) is given to convey information but is not regarded as an error.

Comment 17) denotes a trivial infeasibility--in this case the algorithm is not executed.

Comment 2) is written if the algorithm computation was started but not finished. Comment 1) will be present when comment 2) is written.

OFF LINE PROGRAM COMMENTS

1) OVERFLOW IN NODE PRICES	A node price is greater than 100,000,000. Costs should be rescaled to run job.
2) RUN TERMINATED AT ARC___ ___	Gives the arc at which run was terminated due to the reason stated above the comment.
3) RUN TERMINATED DUE TO ERRORS IN THE DATA	Self - explanatory
4) TOO MANY NODES IN THIS RUN	
5) TOO MANY ARCS IN THIS RUN	
6) CARD PUNCHING ERROR IN ARC CARD NO. _____	These comments are self-explanatory
7) CARD PUNCHING ERROR IN NODE CARD NO. _____	
8) THE ARC IN THE ABOVE ALTER CARD IS NOT IN CORE	
9) SOURCE NODES ARE NOT ADJACENT, ARC ___ ___	All arcs having similar first nodes must be adjacent. This comment gives an arc which is separated from another arc having the same first node.
10) ARC ___ ___ IS A DEAD END ARC	The second node of this arc does not appear anywhere as a first node.
11) NO ARC ENDS AT NODE___ ___	Self-explanatory
12) CARD ___NODE___NOT IN ARCS	A node card appears on which the node is not represented in any arc.
13) ILLEGAL CONTROL CARD(___ ___ ___)	The control card just read into core is not able to be interpreted by the program.

14) OUTPUT CONTROL CARD MISSING OR OUR OF SEQUENCE	Self - explanatory
15) RESERVED TAPE HAS BEEN WRITTEN 16) NO RESERVED TAPE HAS BEEN WRITTEN	This comment states whether an output has been written on a tape other than the system device (as requested by an " OUTPUT TAPE" control card).
17) ARC ___ HAS LOWER BOUND GREATER THAN UPPER BOUND	Self-explanatory
18) NODE ___ NON-CONSERVATIVE, NET FLOW=____	Node has a finite net flow. Negative flow denotes source node.
19) THIS RUN OUTPUT TO TAPE 20) THIS RUN OUTPUT PUNCH	These comments state where the output to this run may be found.
21) _____ARCS ARE OUT OF KILTER	This run was completed, but there is no feasible solution. As many as 100 arcs are marked with an "N" on the output. "N" denotes that these arcs are not in kilter.

11. Program Operation Notes

The I/O device reference numbers the program uses are given below. Of course, these numbers may vary from installation to installation.

I/O Device	Reference Numbers
System input device - all control cards and data packages of the "CARDS" variety	5
System output device- general editing output and "OUTPUT PRINTER" option	6
Reserved input tape- data packages of "TAPE" variety	2
Reserved output tape- "OUTPUT TAPE" output	3
Card punch	7

The current capacity of the routine is 4800 nodes and 14200 arcs. The numbers can be changed, as indicated in section 13.

System control cards must be included in the deck whenever the reserved tapes are used. The numbers 2 and 3 for the reserved input and output tapes respectively were arbitrarily chosen. These numbers can be changed, but they must correspond to the tape numbers specified on the system control cards.

For a reserved output tape the following two control cards must be included:

```
//G.FT03F001 DD UNIT=TAPE9,LABEL=(1,NL),                          X
//     VOLUME=SER=tapeid,DCB=(RECFM=FB,LRECL=80,BLKSIZE=8000)
```

These cards should immediately precede the data. When the job is run under the ASP system (at the MIT Computation Center), the following card must also be included:

```
/*SETUP DDNAME=FT03F001,DEVICE=2400-9,ID=(tapeid,RING,SAVE,NL)
```

This card should immediately follow the job card. Note that tapeid is an identification number assigned to the tape by the MIT Computation Center. Three similar control cards must be included whenever a reserved input tape is used, but FT03 should be changed to FT02. The OS/360 user's manual contains more details concerning the use of reserved tapes.

The sequence of operations by the computer when it is doing one problem is as follows:

First the "READY" card is looked for.

Next the data package is read.

Next comes the generation of the output. When outputting is finished, the next run (if any) will be started.

The running time for this program, of course, varies considerably from problem to problem. The input and output time will be roughly proportional to the number of arcs. The execution of the algorithm is the most variable part of the problem, and its duration will depend on the type of problem considered. At the end of "PRINTER" output, the number of

non-breakthroughs that were obtained are written. Also it writes the "number of X changes," which is the sum of the number of arcs in each breakthrough chain, and "number of nodes from which labeling was done," which is the sum of the number of nodes scanned on each labeling operation.

As an example, a problem was run that gave the following statistics:

Number of arcs	414
Number of nodes	348
Number of breakthroughs	40
Number of non-breakthroughs	179
Number of X-changes	1915
Number of nodes from which labeling was done	7550

The upper bounds on the elapsed times were:

Program compilation	2.2 min.
Data preprocessing	3.3 sec.
Algorithm computations	3.6 sec.

12. Structure of the Program

A. The main routine (MAINE)

1) Sets up I/O device numbers and dimensions

2) Calls the preprocessing routines

PREDAT
ARCASY
MAKEJL
NODASY
READER
TRANSL

3) Calls the subroutine KILTER once for each arc.

4) Calls the postprocessing routine OUTPUT.

The routine also processes certain error and infeasiblility conditions.

B. Subroutine PREDAT looks for a control card of the type " READY", " SAVE", or "PAUSE". If it finds a "READY" card, core is cleared and it looks for a control card of the type "CARDS", "TAPE", "SKIP", or "TRANSPORTATION". After if finds a "CARD" or "TAPE" control card it then looks for the control card "ARCS" on the appropriate input device.

If a "SAVE" card is found the program returns control to the main program and control is passed next to the subroutine READER.

If a "PAUSE" card is found, the end-of-job instructions are executed.

C. Subroutine ARCASY reads arc record after arc record into storage until it comes to a record with "END" or one with "NODES"

The ℓ_{ij}, u_{ij}, c_{ij}, and x_{ij} information is stored in the KL, KU, KC, and KX blocks, respectively. The BCD names of the first nodes are stored in NN, and the BCD names of the second nodes are stored in IJ.

D. Subroutine MAKEJL sets up lists in IL and JL storage. These lists are cumulative counts of the arcs beginning and ending at the nodes. The subroutine also replaces the IJ names by numbers.

E. Subroutine NODASY reads in the node prices, if any.

F. Subroutine READER reads the OUTPUT, ALTER, and COMPUTE control cards.

G. Subroutine TRANSL performs the final operations before going to the Out of Kilter algorithm.

H. Subroutine KILTER tests the arc presented to see if it is in kilter. If it is not in kilter, the assembly language subroutine LABELIN is called. Depending on a flag set in LABELN, the KILTER subroutine then calls either UPNOPR OR BREAKT. When the arc has been brought into kilter or when it is determined that the arc cannot be brought into kilter, the control passes back to MAINE.

I. Subroutine OUTPUT generates the output required for the run.

J. Assembly language subroutine LABELN performs the labeling operation. If a breakthrough results, the next subroutine called by KILTER will be BREAKT. If a non-breakthrough results, the next subroutine called by KILTER will be UPNOPR.

K. Assembly language subroutine BREAKT alters the quantities of flow in the cycle generated by LABELN.

L. Assembly languate subroutine UPNOPR raises the node prices of the labeled nodes by the appropriate amount.

M. Assembly language function NODENO returns the number of the node that has the name presented.

N. Assembly language function LADDR returns the rightmost 16 bits of the word presented as a 32-bits FORTRAN integer.

O. Assembly language function LDECR returns the leftmost 16 bits of the word presented as a 32-bits FORTRAN integer.

P. Assembly language subroutine PLACE stores the rightmost 16 bits of the first full-word argument in the leftmost 16 bits of the second full-word argument.

13. Compiling the Program

In order to change the I/O device numbers of the program, only the main routine need be compiled. The I/O device numbers are the first items to be defined by the program. The symbols assigned to the tapes are as follows:

KI	= System input device
KO	= System output device
KQ(1)	= Punch card device
KQ(2)	= Reserved output tape
KQ(3)	= Reserved input tape

In order to change the dimensions of the program, it is necessary to change the dimensions of all the FORTRAN subprograms and also the numeric values of the symbols KQ(4) and KQ(5). The assembly language subprograms should not be changed since they do not contain dimensions information.

Let a be the maximum of arcs allowed in the program and n the maximum number of nodes allowed. Then the storage which must be allocated for each symbol is as follows:

SYMBOL	DIMENSION
KL	a
KC	a
KU	a
KX	a
NL	n
NN	2n
NP	n
IJ	n
IL	n +1
JL	maximum of n+1 and a - 2n - 1
JI	a

a = arcs
n = nodes

Total storage for above symbols $5a + 4n + \max(a, 3n + 2)$.

Also KQ(4) = a, and KQ(5) = n in the main routine.

A total of 108,000 full words were available for dimensions when the program was last tested on the IBM 360 model 65 computer. One can choose a and n to be any positive integers as long as $5a + 4n + \max(a, 3n + 2) \leq$ full-word storage available for dimensions.