THE OUT-OF-KILTER ALGORITHM AND SOME OF ITS APPLICATIONS IN WATER RESOURCES

J. Kindler

February 1975 WP-75-19

THE OUT-OF-KILTER ALGORITHM
AND SOME OF ITS APPLICATIONS IN WATER RESOURCES

INTRODUCTION

It has been mentioned on many occasions that the conventional techniques are inadequate to plan and formulate complex water resources systems. Unfortunately it may never be possible to take all the many variables, inputs and outputs fully into account in a wholly systematic manner. Assumptions and simplifications will continue to be necessary. Nonetheless, application of the "system approach" provides water resources planners with a much better set of tools than were available 10 to 15 years ago.

The purpose of this paper is to present a simple water resources allocation model based on the Fulkerson's out-of-kilter algorithm [4]. This algorithm is a special purpose linear programming method which has been efficiently used for the solution of a number of water resources problems, just to mention the Texas Water Plan in the USA [9], the Vistula River Project in Poland [10], and the Trent River System in Canada [8]. The model is presented in form of a complete computer program OKAY written in the Fortran language. Potential applications of the model are illustrated by few computational examples. Although in principle this is an allocation model for a single time period, the possibilities of its extension to multiperiod

analysis are also briefly discussed. For complete description of some of the large scale simulation-optimization packages "driven" by the out-of-kilter algorithm, the interested reader may refer to references [9, 10] given at the end of this paper. The model presented herein is of a general nature and can be considered as a base for development of more complex computer programs designed according to the specific character of a problem subject to analysis. Its original feature is iterative use of the out-of-kilter algorithm to take care of the so-called consumptive losses which as a rule occur in all water resources systems.

The out-of-kilter computer code used in the OKAY program (KILT and PACKUP subroutines), has been developed by the Texas Water Development Board in cooperation with the Water Resources Engineers, Inc., Walnut Creek, California, USA. Although at present there is a number of other codes available (e.g. developed by Boeing, General Motors, Share and most recent very efficient versions by R.S. Barr, F. Glover and D. Klingman), they are not discussed in this paper and the interested reader should refer especially to [1]. The out-of-kilter code presented in this paper was successfully applied for studying water resources development alternatives in the Vistula River Basin, within the framework of the UN sponsored first phase of the "Vistula River Project".

Description of the algorithm is based especially on [1], [3] and [4].

THE OUT-OF-KILTER ALGORITHM

An abstract definition of a network is a collection of nodes and a collection of arcs which connect these nodes. Following [1], the problem which is solved by the out-of-kilter algorithm is that of finding the optimal flow in a circulatory network, and is defined as follows:

$$\text{minimize} \sum_{(i,j)\in M} c_{ij}\, x_{ij} \tag{1}$$

subject to:

$$\sum_j x_{ij} - \sum_j x_{ji} = 0 \qquad i = 1,\ldots,m \tag{2}$$

$$l_{ij} \leq x_{ij} \leq k_{ij} \qquad (i,j) \in M \tag{3}$$

where x_{ij} is the flow from node i to node j, c_{ij} is the cost associated with sending one unit flow from node i to node j, l_{ij} and k_{ij} are, respectively, the lower and upper bounds on the amount of flow in arc (i,j), m is the number of nodes and M is the set of all arcs in the network. The arcs are identified by naming the nodes they connect, for example arc (i,j). It is further assumed that all costs, flows and bounds are integers. In addition flows and bounds are nonnegative values.

For such a network, a feasible circulation is defined as a set of flows satisfying relationships (2) and (3). An optimal circulation satisfies (1), (2) and (3).

Associated with this problem is a dual problem which may be stated as:

$$\text{maximize} \quad \sum_{(i,j)\in M} (l_{ij}\, u'_{ij} - k_{ij}\, u''_{ij}) \qquad (4)$$

subject to:

$$u_i - u_j + (u'_{ij} - u''_{ij}) \leq c_{ij} \qquad (i,j) \in M \qquad (5)$$

$$u'_{ij},\ u''_{ij} \geq 0 \qquad (i,j) \in M \qquad (6)$$

$$u_i \text{ unrestricted} \qquad i = 1,\ldots,m$$

The dual variables u_i are called "node prices", and the expression $\bar{c}_{ij} = c_{ij} + u_i - u_j$ is called "marginal cost" or "net cost" associated with arc (i,j).

Assuming the reader has no knowledge of duality theory, Durbin and Kroenke [3] have described the nature of "node prices" by means of a simple economic example. Let us assume that the total transportation cost of certain commodity in the network depends not only on the distribution charges c_{ij} on each arc (i,j), but also on the prices which consumers located at some or all nodes must pay for a unit of flow commodity. Therefore u_i denotes the price of a unit of flow commodity at node i. The above defined net cost $\bar{c}_{ij}$, represents the total cost to the system (consumer and distributor), of transporting one unit of flow from node i to node j. If $(c_{ij} + u_i)$ is greater than u_j, the $\bar{c}_{ij}$ will

be positive and the flow from node i to node j should be kept as low as possible. For example, let us assume that $c_{ij} = 3$, $u_i = 4$ and $u_j = 6, (\bar{c}_{ij} = 1)$. If we can sell one unit of flow at node i for $u_i = 4$, it does not pay to ship a unit from i to j for $c_{ij} = 3$ and to sell it at node j for $u_j = 6$. In case one unit of flow is sold at node i, the system's profit is equal to 4; moving it and selling at node j means the system's profit is only 3. On the other hand, if $(c_{ij} + u_{ij})$ is smaller than u_j, the $\bar{c}_{ij}$ will be negative and the shipment from i to j is profitable. If the value of $\bar{c}_{ij} = 0$, the system is indifferent to an additional unit flowing from i to j.

It can be proved on the basis of the linear programming theory, that feasible circulation is optimal if and only if one of the following conditions is satisfied by each arc $(i,j) \in M$:

$$\text{If } \bar{c}_{ij} < 0 \text{ then } x_{ij} = k_{ij} \tag{7}$$

$$\text{If } \bar{c}_{ij} = 0 \text{ then } l_{ij} \leq x_{ij} \leq k_{ij} \tag{8}$$

$$\text{If } \bar{c}_{ij} > 0 \text{ then } x_{ij} = l_{ij} \tag{9}$$

Condition (7) states that when net arc cost is negative, flow on the arc ought to be as large as possible. Condition (8) states that when net arc cost is zero, the flow level is unimportant as long as it meets upper and lower bounds. Finally, condition (9) states that when net arc cost is positive, flow on the arc ought to be at the minimum level

possible. The algorithm is designed to construct a circulation meeting these conditions.

Any arc that meets the optimality conditions (7), (8) or (9) is said to be "in-kilter"; otherwise, the arc is said to be "out-of-kilter". An "out-of-kilter" arc must then satisfy one of the following conditions:

I.	$\bar{c}_{ij} < 0$	and	$x_{ij} < k_{ij}$
II.	$\bar{c}_{ij} > 0$	and	$x_{ij} > l_{ij}$
III.	$c_{ij} > 0$	and	$x_{ij} < l_{ij}$
IV.	$\bar{c}_{ij} = 0$	and	$x_{ij} < l_{ij}$
V.	$\bar{c}_{ij} = 0$	and	$x_{ij} > k_{ij}$
VI.	$\bar{c}_{ij} < 0$	and	$x_{ij} > k_{ij}$

The general thrust of the algorithm is to bring each out-of-kilter arc in-kilter by adjusting its flow or by changing appropriately the node prices. In order to change the flow of an out-of-kilter arc (s,t), a suitable path must be found from note t to node s which, in conjuction with (s,t), forms a cycle. Flow is then adjusted on each arc of the cycle by amounts which maintain node conservation and contribute towards bringing (s,t) in kilter. A path is being searched by alternative use of labelling rules and a node price changing rules. Excellent description of these rules may be found in [3] and [7]. It should be noted that the algorithm arbitrarily selects an

out-of-kilter arc, and tries to bring that arc into kilter while not forcing any other arc farther out-of-kilter. If the selected arc can be brought into kilter, the algorithm selects another out-of-kilter arc and repeats the procedure. The procedure terminates when all arcs are found to be in kilter. If any arc cannot be brought into kilter, the problem cannot be solved.

THE NETWORK MODEL OF WATER RESOURCES SYSTEM

The network representation is one of the most natural ways of describing a water resources system. Nodes represent all fixed points in the system, such as reservoir, water withdrawal and waste discharge points, water use points, control and balance profiles, gauging profiles, etc. Arcs are inserted for all river reaches, pipelines and other water transfer facilities, demands, supplies, storage quantitites, etc. It should be noted that all arcs must have compatible units (m^3/sec or m^3).

In principle the attached program can be used for optimization of the water resources allocation problem in a single time period. It is well known, however, that in reservoir systems it is normally desirable to optimize allocation (and at the same time operation of the reservoirs), taking into account more than one time period. Otherwise, the final storage will always be depleted to meet the demands of the period at hand, with no hedging against future requirements. For multiperiod analysis the final storage from one period becomes the initial storage for the second period and so on.

Although such analysis can also be made with the help of the attached program (by proper formulation of the network), in such case the usual way is to implement a special program for multiperiod analysis (e.g. the Water Resources Management Model developed for the Vistula Project study which uses KILT and PACKUP as its basic subroutines). The one period network may be thought then as being expanded into the third dimension with interconnection of time-period planes by the storage arcs.

In all cases the objective is to minimize the penalties associated with not meeting the water user target demands or in-stream target flows, or to minimize the sum of these penalties and the system operating costs (e.g. pumping costs). Determination of these penalties or in other words, development of loss functions associated with water shortages, which is one of the fundamental problems in most of the water resources studies, is outside of the scope of this paper.

Referring to the second paragraph of this paper, it should be noted that if arc cost c_{ij} expresses unit operating costs of one of the system's elements, it must be a positive value. Otherwise, if arc cost c_{ij} expresses unit penalties associated with not meeting the target value on this arc (upper bound), it must be taken with a negative sign.

An additional advantage of the networks solved by the out-of-kilter algorithm is the possibility of considering linearized nonlinear loss (penalty) or cost functions, providing these functions are convex. As far as the loss functions are concerned, this restriction is met in most of the real problems. Unfortunately, this in not always the case with the cost functions.

Target demands and flows, as well as discharge and storage capacities make upper bounds on the appropriate arcs. The supply arcs have lower and upper bounds equal to the actual supply rate.

Adequate representation of a water resources system requires also insertion of the consumptive loss arcs. The consumptive losses are assumed here to be proportional to water use, in other words they are expressed as a certain percentage of the amount actually delivered to the water user.

DESCRIPTION OF THE PROGRAM OKAY

The attached program OKAY uses the out-of-kilter algorithm to allocate flows in a network to minimize the total cost of flow in the network. Subroutine PACKUP constructs a packed list of arcs entering each node. Multiple arcs between any pair of nodes are allowed, however, the total number of arcs entering and leaving any single node should be less than 14 (see TEMP in PACKUP). DIMENSION must also be changed in case of networks containing more than 800 arcs or 250 nodes. Subroutine KILT is the proper out-of-kilter algorithm. Subroutine CONSOL is provided to check if the consumptive loss percentages are satisfied.

As an input, the program requires the follwoing information:

A. TITLE CARDS (13A6) 3 cards

   Col.1 - 78 - TITL

B. CONTROL CARD (4I10)

   Col. 1 - 10 - ARCS No. of arcs in the network

   11 - 20 - NODES No. of nodes in the network

21 - 30 - CONS No. of consumptive loss arcs

31 - 40 - ITER Max. number of iterations on the subroutine KILT (at least 1)

C. NETWORK CARDS (7I10) 1 card for each arc

Col. 1 - 10 - N Arc number

11 - 20 - NF(N) Source node for arc N

21 - 30 - NT(N) Sink node for arc N

31 - 40 - LO(N)N Lower bound of flow in arc N

41 - 50 - HI(N) Upper bound of flow in arc N

51 - 60 - COST(N) Cost per unit flow in arc N

61 - 70 (if positive) or penalty for deficit (if negative)

61 - 70 FLOW(N) Initial flow in arc N (usually zero)

D. CONSUMPTIVE LOSS CARDS (3I10) 1 card for each consumptive loss card

Col. 1 - 10 - MAR(I) No. of arc entering the consumptive loss node (supply arc)

11 - 20 - NAR(I) No. of consumptive loss arc

21 - 30 - LAR(I) Consumptive loss expressed as precentage of water supply rate

In application to water resources allocation problems usually an extra node must be created. This is the so-called balance node to and from which all demands and supplies are routed (see the main program).

Referring to the FLOW variable, at the start of solution all flows in the network must satisfy continuity either by a consistent initialization or setting them equal to zero.

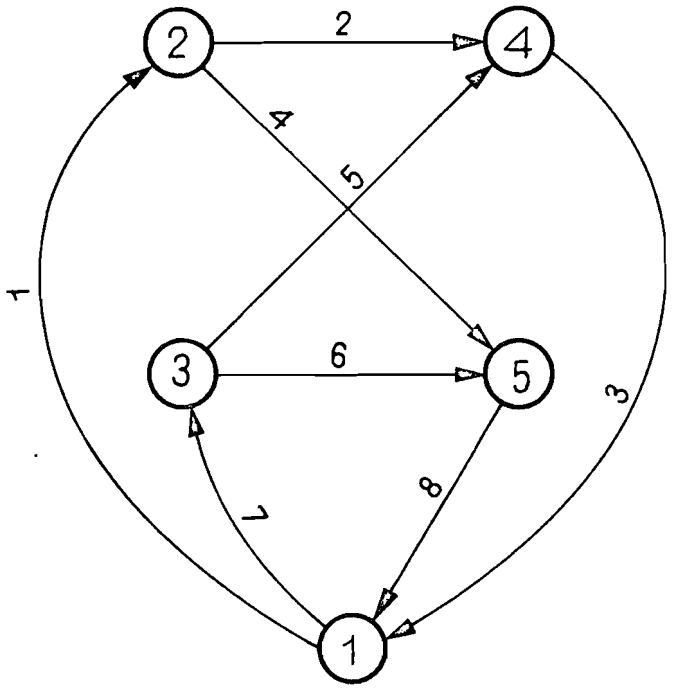
The program requires also that initially the consumptive loss arcs have lower and upper bounds equal to the consumptive loss associated with delivery of the target water demand.

Output from the program consists of a set of optimal flows in the network, value of the objective function corresponding to the optimal solution, final values of node prices and the actual number of iterations performed (iterative use of Subroutine KILT).

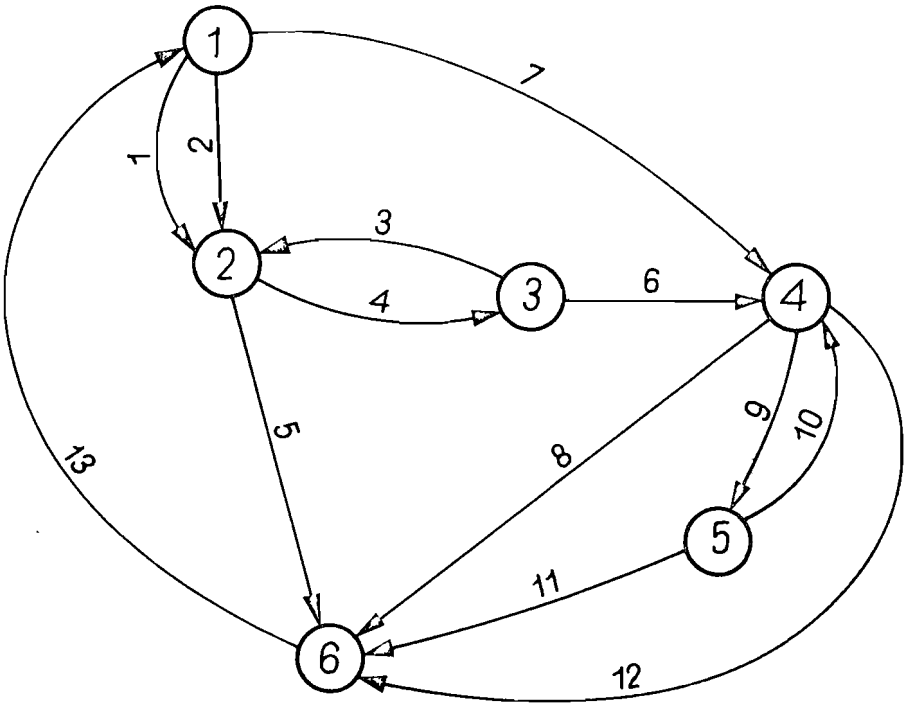
The iterative application of the out-of-kilter algorithm converges very quickly and produces the optimum solution, what has been computationally tested by the parallel application of the standard linear programming code.

Although some experience is called for in formulating the network to describe a given problem, it is hoped that the enclosed examples will give the reader a "feel" for the operation of the out-of-kilter algorithm and the OKAY program.

Example 1



Example 2



EXAMPLE 1
NETWORK EXAMPLE TAKEN FROM *DISCRETE OPTIMIZATION* BY
PLANE D.R., AND MC MILLAN C.JR.
ARCS = 8 NODES = 5 CONS. LOSS DEMANDS = 0

NETWORK DATA

N	NF(N)	NT(N)	LO(N)	HI(N)	COST(N)	FLOW(N)
1	1	2	6	6	-0	-0
2	2	4	0	6	1	-0
3	4	1	3	10	-0	-0
4	2	5	-0	6	2	-0
5	3	4	-0	4	4	-0
6	3	5	-0	4	3	-0
7	1	3	4	4	-0	-0
8	5	1	7	10	-0	-0

EXAMPLE 1
NETWORK EXAMPLE TAKEN FROM *DISCRETE OPTIMIZATION* BY
PLANE D.R., AND MC MILLAN C.JR.

NUMBER OF ITERATIONS = 1

OPTIMAL FLOWS IN NETWORK

1	6
2	3
3	3
4	3
5	-0
6	4
7	4
8	7

TOTAL PENALTY COST = 21

NODE PRICES

1	4
2	3
3	2
4	4
5	5

ARCS = 13 NODES = 6 CONS. LOSS DEMANDS = 0

NETWORK DATA

N	NF(N)	NT(N)	LO(N)	HI(N)	COST(N)	FLOW(N)
1	1	2	0	260	-0	-0
2	1	2	460	460	-0	-0
3	3	2	0	260	715	-0
4	2	3	260	260	-0	-0
5	2	6	0	720	27	-0
6	3	4	0	260	-0	-0
7	1	4	1100	1100	-0	-0
8	4	6	0	1500	642	-0
9	4	5	1200	1200	-0	-0
10	5	4	0	1200	1300	-0
11	5	6	0	1200	-0	-0
12	6	1	1560	1560	-0	-0
13	4	6	0	260	-0	-0

EXAMPLE 2
PROBLEM FORMULATED BY ERIC WOOD FROM IIASA

NUMBER OF ITERATIONS = 1

OPTIMAL FLOWS IN NETWORK

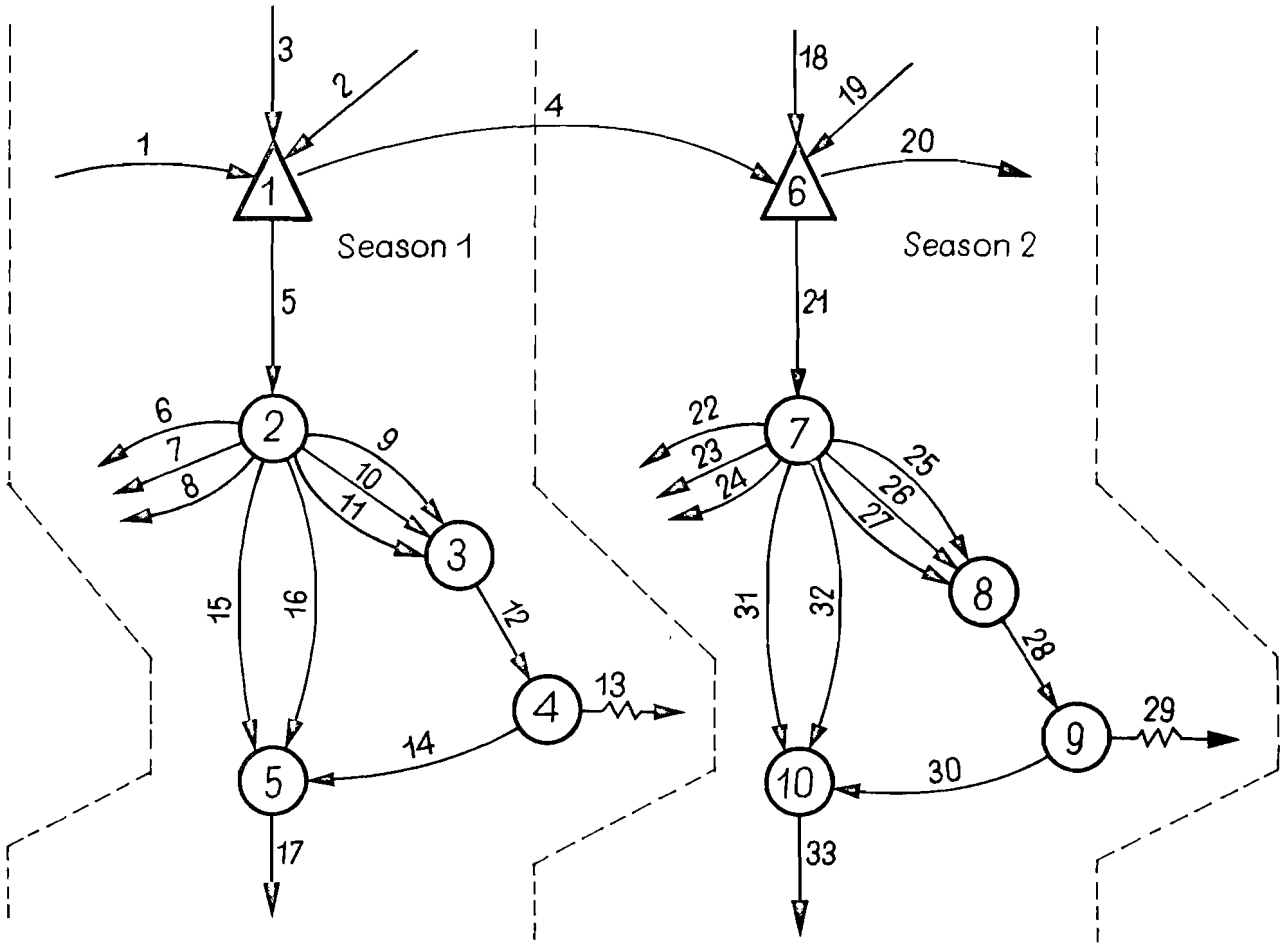
1	-0
2	460
3	-0
4	260
5	200
6	260
7	1100
8	-0
9	1200
10	-0
11	1200
12	1560
13	160

TOTAL PENALTY COST = 5400

NODE PRICES

1	27
2	0
3	27
4	27
5	27
6	27

Example 3



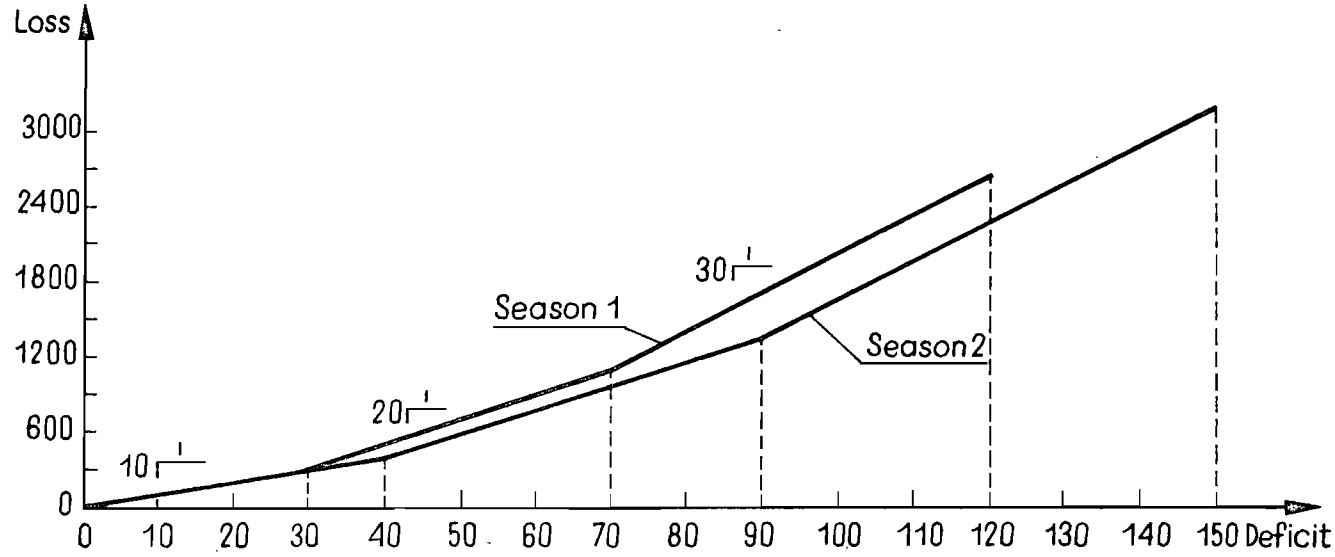
Input Data

-Reservoir capacity	1000
-Discharge capacity of water transfer	300
-Demand I /season 1/, 30+40+50	120
-Demand I /season 2/, 40+50+60	150
-Demand II /season 1/, 80+100+300	480
-Demand II /season 2/, 200+120+340	660
-Consumptive loss assoc. with Demand I	5%
-Consumptive loss assoc. with Demand II	8%
-Minimum acceptable flow /season 1/	50
-Minimum acceptable flow /season 2/	40
-Reservoir inflow /season 1/	50
-Reservoir inflow /season 2/	150
-Initial storage /season 1/	200

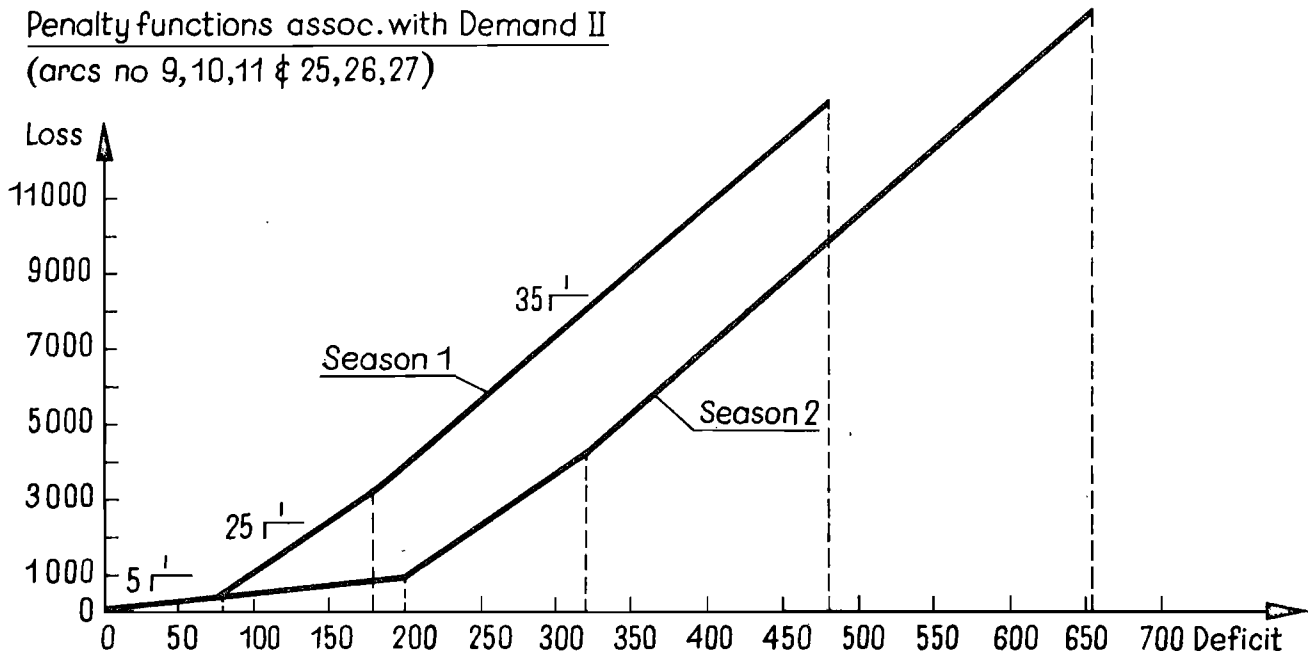
Network Representation /arc nos./	season 1	season 2
- Reservoir inflow	3	18
- Initial storage	1	4
- Final storage	4	20
- Water transfer /pumping/	2	19
- River channel	5,16,17	21,32,33
- Demand I	6, 7, 8	22,23,24
- Demand II	9,10,11	25,26,27
- Auxiliary arcs	12	28
- Consumptive loss	13	29
- Wastewater discharge	14	30
- Minimum acceptable flow	15	31

Note: Pumping and penalty costs indicated on the next page and on the computer printout /page 16/.

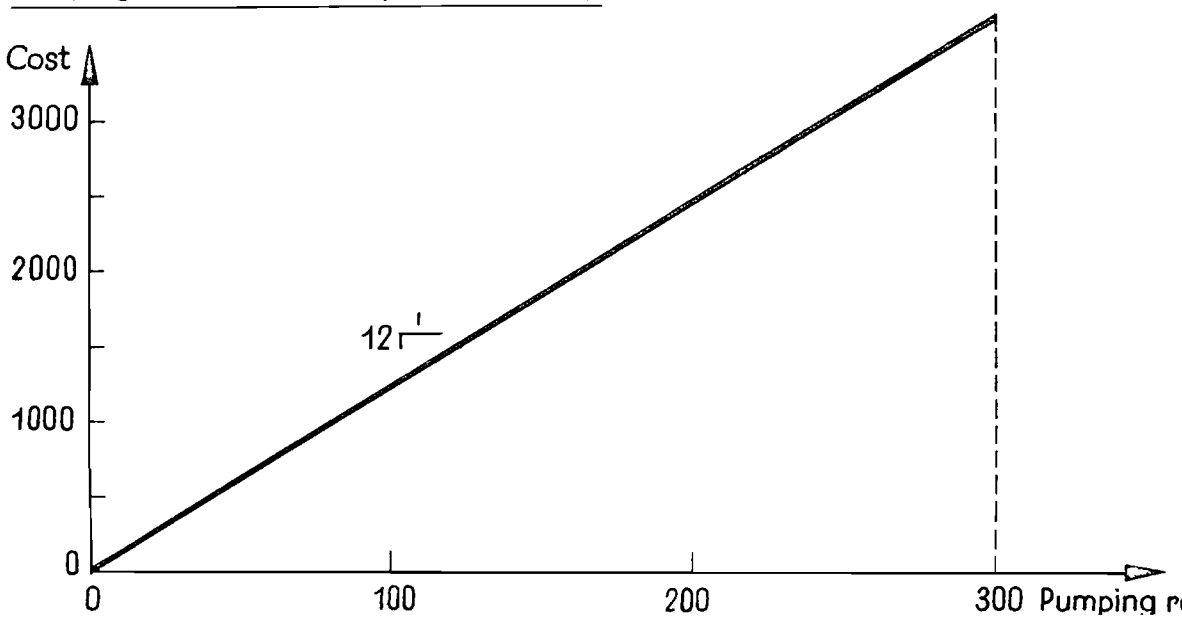
Penalty functions assoc. with Demand I
(arcs no 6,7,8 ¢ 22, 23, 24)



Penalty functions assoc. with Demand II
(arcs no 9,10,11 ¢ 25,26,27)



Pumping costs function (arcs no 2 ¢ 19)



EXAMPLE 3
AN EXAMPLE HOW OKAY PROGRAM CAN BE USED FOR SOLUTION OF
SIMPLE MULTIPERIOD PROBLEMS /2 TIME PERIODS/
ARCS = 33 NODES = 10 CONS. LOSS DEMANDS = 2

NETWORK DATA

N	NF(N)	NT(N)	LO(N)	HI(N)	COST(N)	FLOW(N)
1	11	1	200	200	-0	-0
2	11	1	-0	300	12	-0
3	11	1	50	50	-0	-0
4	1	6	-0	1000	-0	-0
5	1	2	-0	99999	-0	-0
6	2	11	-0	30	-10	-0
7	2	11	-0	40	-20	-0
8	2	11	-0	50	-30	-0
9	2	3	-0	80	-5	-0
10	2	3	-0	100	-25	-0
11	2	3	-0	300	-35	-0
12	3	4	-0	480	-0	-0
13	4	11	24	24	-0	-0
14	4	5	-0	456	-0	-0
15	2	5	-0	50	-40	-0
16	2	5	-0	99999	-0	-0
17	5	11	-0	99999	-0	-0
18	11	6	150	150	-0	-0
19	11	6	-0	300	12	-0
20	6	11	-0	1000	-0	-0
21	6	7	-0	99999	-0	-0
22	7	11	-0	40	-10	-0
23	7	11	-0	50	-20	-0
24	7	11	-0	60	-30	-0
25	7	8	-0	200	-5	-0
26	7	8	-0	120	-25	-0
27	7	8	-0	340	-35	-0
28	8	9	-0	660	-0	-0
29	9	11	53	53	-0	-0
30	9	10	-0	607	-0	-0
31	7	10	-0	40	-40	-0
32	7	10	-0	99999	-0	-0
33	10	11	-0	99999	-0	-0

CONS. LOSS DATA

12	13	5
28	29	8

EXAMPLE 3
AN EXAMPLE HOW OKAY PROGRAM CAN BE USED FOR SOLUTION OF SIMPLE MULTIPERIOD PROBLEMS /2 TIME PERIODS/

NUMBER OF ITERATIONS = 2

OPTIMAL FLOWS IN NETWORK

1	200
2	300
3	50
4	90
5	460
6	0
7	0
8	50
9	0
10	60
11	300
12	360
13	18
14	342
15	50
16	0
17	392
18	150
19	300
20	0
21	540
22	0
23	0
24	60
25	-0
26	100
27	340
28	440
29	35
30	405
31	40
32	-0
33	445

TOTAL PENALTY COST = 12600

NODE PRICES

1	25
2	25
3	0
4	0
5	0
6	25
7	25
8	0
9	0
10	0
11	0

BIBLIOGRAPHY

[1] Barr, R.S., Glover, F., Klingman, D., An improved version of the out-of-kilter method and a comparative study of computer codes, Mathematical Programming (7), 1974.

[2] Dantzig, G.B., Notes on network flow, linear programming and stochastic programming models for a river basin, IIASA Memorandum, 1974.

[3] Durbin, E.P., D.M. Kroenke, The out-of-kilter algorithm: a primer, Rand Co., Memorandum RM-5472-PR, 1967.

[4] Fulkerson, D.R., An out-of-kilter method for minimal cost flow problems, Journal of Society of Applied Industrial Mathematics, Vol. 9, No. 1, 1961.

[5] Glover, F., Karney D., Klingman, D., Implementations and computational comparisons of primal, dual primal-dual computer codes for minimum costs network flow problems, Networks (4), 1974.

[6] King, I.P., Filimowski, J., Kindler, J., The out-of-kilter algorithm as a single-step method for simulation and optimization of Vistula River planning alternatives, Proceedings of the IASH Int, Symposium on Mathematical Models in Hydrology, Warsaw, 1971.

[7] Plane, D.R., McMillan, C.Jr., Discrete optimization. Integer programming and network analysis for management decisions, Prentice Hall, Inc., 1971.

[8] Sigivaldason, O.T., A simulation model for operating a multipurpose multireservoir system, Typed report, 1974.

[9] Texas Water Development Board and Water Resoruces Engineers, Inc., Systems simulation for management of a total water resource, Report 118, 1970.

[10] UNDP/UN and Hydroproject, Planning comprehensive development of the Vistula River System. Draft Final Report, 1972.

PROGRAM OKAY

```
      PROGRAM OKAY(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      INTEGER FLOW,HI,COST,ARCS,PI,DEFCIT,CONS,PER
      LOGICAL INFES
      COMMON /ADATA/ NF(800),NT(800),FLOW(800),HI(800),LO(800)
     1,COST(800),PI(250),ARCS,NODES,INFES
     2,MAR(50),LAR(50),NAR(50)
      DIMENSION TITL(39)
      READ(5,1) TITL
      WRITE(6,100) TITL
      READ(5,2) ARCS,NODES,CONS,ITER
      WRITE(6,101) ARCS,NODES,CONS
      READ(5,3) (N,NF(N),NT(N),LO(N),HI(N),COST(N),FLOW(N), L=1,ARCS)
      WRITE(6,102)
      NODES=NODES+1
      KEY = 1
      DO 200 N=1,ARCS
      IF(NF(N).EQ.0) NF(N)=NODES
      IF(NT(N).EQ.0) NT(N)=NODES
      IF(NF(N).EQ.NODES.OR.NT(N).EQ.NODES) KEY=2
  200 CONTINUE
      IF(KEY.EQ.1) NODES=NODES-1
      WRITE(6,112)
      WRITE(6,103) (N,NF(N),NT(N),LO(N),HI(N),COST(N),FLOW(N),N=1,ARCS)
      IF(CONS.EQ.0) GO TO 800
      WRITE(6,109)
      DO 300 I=1,CONS
      READ(5,2) MAR(I),NAR(I),LAR(I)
  300 WRITE(6,110) MAR(I),NAR(I),LAR(I)
  800 DO 400 NITER=1,ITER
      CALL KILT
      IF(CONS.EQ.0) GO TO 500
      IFLAG=0
      DO 900 IK=1,CONS
      I = MAR(IK)
      J = NAR(IK)
      PER = LAR(IK)
      CALL CONSOL (I,J,PER,JCOR)
      IFLAG=IFLAG + JCOR
  900 CONTINUE
      IF(IFLAG.EQ.0) GO TO 500
  400 CONTINUE
  500 CONTINUE
      WRITE(6,100) TITL
      WRITE(6,111) NITER
      WRITE(6,104)
      WRITE(6,105) (N,FLOW(N), N=1,ARCS)
      ITOT = 0
      DO 202 N=1,ARCS
      IF(COST(N)) 675,680,685
  675 DEFCIT=FLOW(N)-HI(N)
      GO TO 688
  680 DEFCIT=0
      GO TO 688
  685 DEFCIT=FLOW(N)
  688 ITOT=ITOT+COST(N)*DEFCIT
  202 CONTINUE
      WRITE(6,106) ITOT
```

```
      WRITE(6,107)
      DO 203 N=1,NODES
  203 WRITE(6,108) N,PI(N)
    1 FORMAT(13A6)
    2 FORMAT(4I10)
    3 FORMAT(7I10)
  100 FORMAT(1H1/(1X,13A6))
  101 FORMAT(1X,6HARCS =,I5,12H      NODES =,I5,26H       CONS. LOSS DEMAN
     1DS =,I5)
  102 FORMAT(1H0,10X,12HNETWORK DATA/)
  103 FORMAT(1X,7I10)
  104 FORMAT(1H0,1X,24HOPTIMAL FLOWS IN NETWORK//)
  105 FORMAT(1X,I5,I8)
  106 FORMAT(///21X,20HTOTAL PENALTY COST =,I10)
  107 FORMAT(///21X,11HNODE PRICES//)
  108 FORMAT(21X,I5,5X,I10)
  109 FORMAT(//10X,15HCONS. LOSS DATA/)
  110 FORMAT(3I10)
  111 FORMAT(1H0,1X,22HNUMBER OF ITERATIONS =,I5)
  112 FORMAT(9X,1HN,5X,5HNF(N),5X,5HNT(N),5X,5HLO(N),5X,5HHI(N),3X,
     17HCOST(N),3X,7HFLOW(N)/)
      STOP
      END
```

```
      SUBROUTINE KILT
      INTEGER FLOW,HI,COST,ARCS,PI,DEFCIT,CONS,PER
      INTEGER AIK,C,E,AIN,COK,EPS,SNK,AL,SRC,AOK,CUT,DEL,A
      LOGICAL INFES
      COMMON /ADATA/ NF(800),NT(800),FLOW(800),HI(800),LO(800)
     1,COST(800),PI(250),ARCS,NODES,INFES
     2,MAR(50),LAR(50),NAR(50)
      COMMON IPT(250),LARCS(250),LLN,NARCS,LIST(1600),LN(250),CUT(800)
     1,NA(250),NB(250)
      CALL PACKUP
      JKL = 0
      INFES=.TRUE.
      DO 200 A=1,ARCS
      IF( HI(A) - LO(A) ) 340,200,200
  200 CONTINUE
      INFES = .FALSE.
      INF=999999
      AOK   = 0
      AIK   = 0
  220 AIN=AIK + 1
      DO 320 A=AIN,ARCS
      IA = NF(A)
      JA = NT(A)
      AIK = A  - 1
      IF( LO(A) - FLOW(A) )  240,240,360
  240 IF( HI(A) - FLOW(A) ) 380,260,260
  260 IF( COST(A) + PI(IA) - PI(JA) ) 280,320,300
  280 IF( HI(A) - FLOW(A) ) 320,320,360
  300 IF( LO(A) - FLOW(A) ) 380,320,320
  320 CONTINUE
  340 CONTINUE
      RETURN
  360 SRC = NT(A)
      SNK = NF(A)
      E   =+1
      GO TO 400
  380 SRC = NF(A)
      SNK = NT(A)
      E   =-1
  400 IF( A - AOK ) 440,420,440
  420 IF( NA(SRC) ) 480,440,480
  440 AOK = A
      DO 460 N=1,NODES
      NA(N) = 0
  460 CONTINUE
      NA(SRC) = SNK*E
      NB(SRC) = AOK*E
      K  = 0
      LU = 1
      LN(LU) = SRC
  480 COK = COST(A) + PI(IA) - PI(JA)
  500 LLN = LN(LU)
      LN(LU) = 0
      LU     = LU - 1
      IADD = LARCS(LLN)
      MAX = IPT(LLN)
      DO 740 AL = IADD, MAX
```

```
      JKL = JKL + 1
      A = LIST(AL)
      IA = NF(A)
      JA = NT(A)
      IF(NA(IA)) 520,620,520
  520 IF(NA(JA)) 740,540,740
  540 CONTINUE
      IF( HI(A) - FLOW(A) ) 720,720,560
  560 IF( LO(A) - FLOW(A) ) 580,580,600
  580 IF( COST(A) + PI(IA) - PI(JA) ) 600,600,720
  600 CONTINUE
      NA(JA) = IA
      NB(JA) = A
      LO     = LO+1
      LN(LO) = JA
      GO TO 700
  620 IF( LO(A) - FLOW(A) ) 640,720,720
  640 IF( HI(A) - FLOW(A) ) 680,660,660
  660 IF( COST(A) + PI(IA) - PI(JA) ) 720,680,680
  680 CONTINUE
      NA(IA) = -JA
      NB(IA) = -A
      LO     = LO+1
      LN(LO) = IA
  700 IF(NA(SNK)) 760,740,760
  720 K      = K + 1
      CUT(K) = A
  740 CONTINUE
      IF( LO ) 1060,1060,500
  760 EPS = INF
      NI  = SRC
  780 NJ  = IABS(NA(NI))
      A   = IABS(NB(NI))
      IA=NF(A)
      JA=NT(A)
      C=COST(A)+PI(IA)-PI(JA)
      IF( NB(NI) ) 900,800,800
  800 IF( C ) 860,860,820
  820 IF( LO(A) - FLOW(A) ) 1000,1000,840
  840 EPS = MIN0( EPS, LO(A) - FLOW(A) )
      GO TO 1000
  860 IF( HI(A) - FLOW(A))1000,1000,880
  880 EPS = MIN0( EPS, HI(A) - FLOW(A) )
      GO TO 1000
  900 IF( C ) 960,920,920
  920 IF( LO(A) - FLOW(A) ) 940,1000,1000
  940 EPS = MIN0( EPS, FLOW(A) - LO(A) )
      GO TO 1000
  960 IF( HI(A) - FLOW(A) ) 980,1000,1000
  980 EPS = MIN0( EPS, FLOW(A) - HI(A) )
 1000 NI = NJ
      IF( NI - SRC ) 780,1020,780
 1020 NJ = IABS(NA(NI))
      A  = IABS(NB(NI))
      FLOW(A) = FLOW(A) + ISIGN(EPS,NB(NI))
      NI = NJ
      IF( NI - SRC ) 1020,1040,1020
```

```
1040 AOK=0
     GO TO 220
1060 DEL = INF
     IC = 0
     DO 1200 I=1,K
     A  = CUT(I)
     IA = NF(A)
     JA = NT(A)
     C = COST(A) + PI(IA) - PI(JA)
     IF(NA(JA)) 1140,1080,1140
1080 CONTINUE
     IF( HI(A) - FLOW(A) ) 1140,1140,1100
1100 IF( DEL - C ) 1200,1200,1120
1120 DEL = C
     IC = I
     NSN = IA
     GO TO 1200
1140 IF(NA(IA)) 1200,1160,1200
1160 CONTINUE
     IF( LO(A) - FLOW(A) ) 1180,1200,1200
1180 IF (-C.GE.DEL) GO TO 1200
     DEL = -C
     IC = I
     NSN = JA
1200 CONTINUE
     LU = 1
     LN(LU) = NSN
     IF( DEL - INF ) 1300,1220,1300
1220 IF( HI(AOK) - FLOW(AOK) ) 1240,1260,1240
1240 IF( LO(AOK) - FLOW(AOK) ) 1280,1260,1280
1260 DEL =IABS(COK)
     NSN = SRC
     LN(LU) = NSN
     GO TO 1300
1280 INFES = .TRUE.
     WRITE(6,2000) AOK,NF(AOK),NT(AOK),LO(AOK),HI(AOK),FLOW(AOK)
    1,COST(AOK)
2000 FORMAT(5H0 ARC,I5,58H  CANNOT BE BROUGHT INTO KILTER. DATA FOR ARC
    I LISTED BELOW/6I10)
     RETURN
1300 DO 1340 N=1,NODES
     IF(NA(N)) 1340,1320,1340
1320 CONTINUE
     PI(N) = PI(N) + DEL
1340 CONTINUE
     IF(  IC .EQ. 0 ) GO TO 220
     CUT(IC) = CUT(K)
     K = K - 1
     GO TO 220
     END
```

```
      SUBROUTINE PACKUP
      INTEGER FLOW,HI,COST,ARCS,PI,DEFCIT,CONS,PER
      INTEGER OVERA,OVERN,OVERSZ,TEMP,DIMEN
      LOGICAL INFES
      DIMENSION TEMP(14,250)
      COMMON /ADATA/ NF(800),NT(800),FLOW(800),HI(800),LO(800)
     1,COST(800),PI(250),ARCS,NODES,INFES
     2,MAR(50),LAR(50),NAR(50)
      COMMON IPT(250),LARCS(250),LLN,NARCS,LIST(3600)
     1,OVERA(500),OVERN(500)
      EQUIVALENCE(TEMP(1),LIST(101))
      DIMEN=14
      OVERSZ=500
      LOC=0
      MAX = 0
      DO 200 J=1,NODES
      IPT(J)=0
  200 CONTINUE
      DO 360 L=1,ARCS
      IF(HI(L).EQ.LO(L).AND.HI(L).EQ.FLOW(L)) GO TO 360
      J=NF(L)
      DO 340 I=1,2
      IPT(J)=IPT(J)+1
      M=IPT(J)
      IF(M-DIMEN) 310,310,300
  300 CONTINUE
      LOC=LOC+1
      IF(LOC.GT.OVERSZ)  GO TO 700
      OVERN(LOC)=J
      OVERA(LOC)=L
      GO TO 320
  310 CONTINUE
      TEMP(M,J)=L
  320 CONTINUE
      J=NT(L)
  340 CONTINUE
  360 CONTINUE
      IADD=0
      DO 420 J=1,NODES
      LARCS(J)=IADD+1
      LIMIT=IPT(J)
      IF(LIMIT-DIMEN)376,376,362
  362 LIM=DIMEN+1
      L=1
      DO 370 M=LIM,LIMIT
  364 IF(OVERN(L)-J) 368,366,368
  366 IADD=IADD+1
      LIST(IADD)=OVERA(L)
      L=L+1
      GO TO 370
  368 L=L+1
      IF(L-LOC) 364,364,370
  370 CONTINUE
      LIM=DIMEN
      GO TO 378
  376 LIM=LIMIT
  378 DO 380 M=1,LIM
```

```
      IADD=IADD+1
      LIST(IADD)=TEMP(M,J)
  380 CONTINUE
  400 IPT(J)=IADD
  420 CONTINUE
      RETURN
  700 WRITE(6,9001)
 9001 FORMAT(55H1    SIZE LIMIT OF OVERSZ EXCEEDED  EXECUTION TERMINATED)
      STOP
      END
```

```
      SUBROUTINE CONSOL (I,J,PER,JCOR)
      INTEGER FLOW,HI,COST,ARCS,PI,DEFCIT,CONS,PER
      COMMON /ADATA/ NF(800),NT(800),FLOW(800),HI(800),LO(800)
     1,COST(800),PI(250),ARCS,NODES,INFES
     2,MAR(50),LAR(50),NAR(50)
      JF=(FLOW(I)*PER)/100
      JCOR=FLOW(J)-JF
      IF(JCOR.EQ.0) RETURN
      HI(J)=JF
      LO(J)=JF
      RETURN
      END
```