# MINET
# FAST NETWORK LP SOLVER
# Description and User's Guide for V2.00

*István Maros*

January 1988
WP-88-6

# M I N E T
# FAST NETWORK LP SOLVER
# Description and User's Guide for V2.00

*István Maros*

January 1988
WP-88-6

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

## FOREWORD

This paper extends the WP-87-50 in the direction to a detailed discussion of pricing and anti-degeneracy strategies applicable in solutions of network LP solvers. Besides the theoretical part it also serves as a users guide to the existing software.

Alexander B. Kurzhanski
Chairman
System and Decision Sciences Program

## CONTENTS

# MINET
# FAST NETWORK LP SOLVER
# Description and User's Guide for V2.00

*István Maros*

Computer and Automation Institute of
the Hungarian Academy of Sciences,
H-1395 Budapest, P.O.B. 408.

## 1. INTRODUCTION

This User's Guide gives information on the new (V2.00) version of MINET, the fast solver of network type linear programming (LP) problems. The statement of the problem, the theoretical backgrounds, the data structures, the main ideas of the solution algorithm, and the input/output formats of the first version are described in IIASA Working Paper "MINET a Fast Network LP Solver" WP-87-50, June 1987, by I. Maros.

Since the preparation of the first version some changes have been made on MINET. The purpose of these changes is to improve the user friendliness of MINET on IBM PC/XT, AT and compatible computers, and to enable the use of an anti-degeneracy strategy in column selection. The development and incorporation of the anti-degeneracy strategy was suggested by the fact that the number of degenerate iterations (when the objective function does not change) is very large with network LP algorithms.

In the input file MINET V2.00 allows the use of free format. This may be convenient in some cases. However, it is our experience that using some more regular format (for instance the one which was used in V1.00) gives a better overview of the problem and possible errors or wrong data can more easily be detected than in the case of free format.

The use of the very efficient sectional pricing strategy has also been changed.

The intermediate output in trace 1 mode has been redesigned and also the output lines at the end of Phase-I and Phase-II, where the number of degenerate iterations appears as well.

At the start of MINET the user gets a menu-like information on the possible choices of the run parameters. Each run parameter, except for the file name of the input data, has a default value. Usually they need not be changed since they have been established on the basis of long experience with different classes of problems.

One thing is to be noted. The new anti-degeneracy strategy requires some extra computations which increases the time per iteration. Its use is justified only when the overall computational effort is reduced by the use of this strategy. The activation of the anti-degeneracy column selection strategy is optional and the default interpretation is not to use it.

For convenience we repeat the problem statement to enable better understanding of the present documentation.

## 2. PROBLEM STATEMENT

The minimal cost network flow (MCNF) problem or capacitated transshipment problem can be stated as follows:

$$\text{minimize } z = \sum_{(i,j)\in E} c(i,j)^* x(i,j) \tag{1}$$

$$\text{subject to } \sum_{i\in I(k)} x(i,k) - \sum_{j\in O(k)} x(k,j) = b(k) \quad \text{for } k \in M \tag{2}$$

$$0 \le x(i,j) \le u(i,j) \quad \text{for } (i,j) \in E \ , \tag{3}$$

where

$$I(k) = \{i \in M : (i,k) \in E\} \ ,$$

$$O(k) = \{j \in M : (k,j) \in E\} \ ,$$

$E$ is a set of arcs $(i, j)$ of the network $G(M, E)$,

$M$ is the set of nodes.

The cardinality of $M$ is denoted by $m$, and that of $E$ is denoted by $n$. The constant $b(k)$ represents the requirement at node $k$. A node $k \in M$, for which

$b(k) < 0$ is called a supply node,

$b(k) > 0$ is called a demand node,

$b(k) = 0$ is called a pure transshipment node.

It is assumed that

$$V = \sum_{k \in M} b(k) = 0 \ . \tag{4}$$

If $V > 0$ then there is no feasible solution. If $V < 0$ then the problem can be converted to the prescribed form by adding a dummy destination with a requirement of $-V$ and slack arcs from each source to the dummy destination. The slack arcs are given unit costs of zero and infinite arc capacities.

It should be noted that (3) is not a restriction of generality because individual lower bounds of the variables different from zero can always be moved to zero by a simple transformation.

Associated with each node $k \in M$ is a dual variable $\pi(k)$ called its node potential or simplex multiplier. An arc $(i, j)$ is directed from node $i$ to node $j$. An arc $(i, j)$ is said to be out-directed from node $i$ and in-directed in node $j$. In this sense $I(k)$ is the set of tail nodes of arcs that are in-directed to node $k$, and $O(k)$ is the set of head nodes of arcs that are out-directed from node $k$.

The flow, cost, and upper bound of arc $(i, j)$ are represented, respectively by $x(i, j)$, $c(i, j)$, and $u(i, j)$. The objective is to determine a set of arc flows which satisfies the node requirements and capacity restrictions at a minimum total cost.

Let us denote the matrix of constraints on the left hand side of (2) by $A$. It is easy to see that each column of $A$ is associated with an arc $(i, j)$ and contains one coefficient with value of -1 and one with +1 corresponding to the starting node and the ending node, respectively. In addition to this usual case we allow in MINET such arcs which correspond to loops (self-loops). In this case the "from" and the "to" nodes coincide. The matrix column of such an arc contains one -1 coefficient.

Using a more concise notation we can write problem (1)-(3) in the following form (with evident interpretation of the correspondence):

$$\text{minimize } z = c'x \tag{5}$$

$$\text{subject to } Ax = b \tag{6}$$

$$0 \leq x \leq u \ , \tag{7}$$

where $A$ is an m by $n$ matrix, $c$, $x$, and $u$ are $n$-vectors, $b$ is an $m$-vector and $'$ (prime) denotes the transpose.

The dual of this problem is the following:

$$\text{maximize } Z = b'w - u'v \tag{8}$$

$$\text{subject to } A'w - v \leq c \tag{9}$$

$$w \quad \text{unrestricted} \;, \tag{10}$$

$$v \geq 0 \;,$$

where $w$ is an $m$-vector and $v$ is an $n$-vector of dual variables.

It is a rather characteristic feature of the network LP problems that usually there are much more arcs than nodes, i.e. the number of variables is much larger than the number of node constraints, $m \ll n$.

## 3. COLUMN SELECTION STRATEGY OF MINET

Though column selection (also called pricing) in network LP algorithms is a very simple operation (see e.g. [2]) still it is the experience that high percent (up to 95) of the solution effort (and time) is spent in this operation. This is especially true if in each iteration all the nonbasic variables are interrogated (remember that due to $m \ll n$ this can really be a great deal of work). Therefore if would be essential to design an "optimal" pricing strategy. So far it was impossible to produce such a strategy which proved to be equally efficient to a wide class of network LP problems.

MINET is equipped with several different column selection strategies. Some of them are variants of known pricing methods (standard pricing ST), while the applied sectional pricing (SC) is new. The choice between ST and SC is controlled by a user accessible parameter (see section 7.).

### 3.1. Standard Pricing

MINET offers four options in standard pricing. The selection is controlled by parameter NPA. In this section this identifier will be written in capitals, for better readability. This is equivalent with parameter npa of section 7.

If standard pricing is chosen then the value of NPA defines the actual strategy.

NPA = 0 Scanning of the matrix always starts at the first nonbasic variable of A in (6). In Phase-I scanning stops at the first improving variable (one with positive reduced cost $d(j)$). In Phase-II the whole matrix is interrogated and a vector with the largest $d(j)$ is selected.

| | |
|---|---|
| NPA = 1 | Both in Phase-I and Phase-II the whole matrix is interrogated and a vector with the largest $d(j)$ is selected. |
| NPA = 2 | Scanning of the matrix always starts at the first nonbasic variable of $A$ and stops in both phases at the first improving vector. |
| NPA = 3 | In Phase-I scanning of the matrix always starts at the last nonbasic variable of $A$, goes in the decreasing order of indices, and stops at the first improving vector. In Phase-II the whole matrix is interrogated and a vector with the largest $d(j)$ is selected. |

## 3.2. Sectional Pricing

Sectional pricing is controlled by three parameters (for better readability in this section they will be written in capitals)

NSC
KSC
KVC

With proper setting of the parameters we can obtain most of the known pricing methods as special cases of SC.

In sectional pricing matrix $A$ of the constraints in (6) is divided by nearly equally spaced vertical lines into NSC parts, called sections.

Parameter KSC defines the number of sections to be scanned during one pricing operation in the normal case.

Parameter KVC defines the number of improving vectors (the ones with positive reduced cost $d(j)$) to be scanned in one section.

In a general pricing step SC scans only a subset of nonbasic variables and the one with the largest reduced cost is selected to enter the basis.

At the very beginning pricing starts in section 1 and stops when KVC vectors with $d(j) > 0$ have been encountered. The last scanned vector in the section is marked. Then pricing proceeds to the next section where exactly the same procedure is applied. Altogether KSC sectors are visited in this way. If in a section no improving vector was found then one more section will be searched if there is any section left. The last scanned section is marked.

In a subsequent iteration scanning starts in the section next to the marked section (if it was the last then SC cycically goes back to the first section). The first scanned variable in this section will be the one next to the marked variable. Scanning is carried out in the same way as described above. If the end of the section is encountered then pricing cyclically goes back to the beginning of the section.

It is easy to see that the above procedure provides a great deal of flexibility in the pricing operation.

## 4. ANTI-DEGENERACY STRATEGY

It is well known that network LP algorithms make a very large number of degenerate steps (when the value of the objective function does not change). In theory, the danger of cycling is present, however, in practice these algorithms never fall in a cycle. At the same time it is true that considerable improvement could be expected if the number of degenerate iterations could be reduced. Such efforts have been made in general purpose LP algorithms [1], [3]. Some special pricing procedures have been designed to reduce the number of degenerate iterations. Each of them requires additional computations per (major) iterations. The actual rate of improvement would depend on the balance between the extra computations per iteration and the reduction in the number of iterations. In single-user PC terms it means the reduction of solution time.

In this section we try to take the advantage of some results in the field and exploit them in favor of improving the performance of network LP algorithms.

First we present some observations regarding degeneracy and column selection.

For simplicity we will denote the vector of basic solution by $\beta$, with components $(\beta_1, \cdots \beta_m)$, and the vector of upper bounds of basic variables by $\mathbf{u}$. $H$ is the index set of basic variables which are at one of their bounds.

$H_1$ denotes the index set of basic variables at their lower bound, and $H_2$ is the index set of basic variables at their upper bound.

$$H_1 = \{i : \beta_i = 0\}$$

$$H_2 = \{i : \beta_i = u_i\}$$

Clearly $H = H_1 \cup H_2$. A basis is degenerate if $H$ is non-empty. One column of matrix $\mathbf{A}$ is denoted by $\mathbf{a}_j$, and its updated form by $\alpha_j$ ($\alpha_j = \mathbf{B}^{-1}\mathbf{a}_j$). It is easy to see [3] that if column $j$ enters the basis then it will change the value of the objective function, if the following holds for this column:

$$\sum_{i \in H} |\alpha_{ij}| = 0 \tag{11}$$

However, checking this condition is rather expensive, especially if we use the revised simplex method which is the case with network simplex algorithms. There is a possibility to approximate this sum by:

$$|\sum_{i \in H} \alpha_{ij}| \tag{12}$$

This sum can easily be computed if we define vector $\mathbf{c}$ as

$$c_i = \begin{cases} 1, & \text{if } i \in H \\ 0, & \text{otherwise} \end{cases}$$

With this $\mathbf{c}$

$$|\sum_{i \in H} \alpha_{ij}| = |\mathbf{c}'\boldsymbol{\alpha}_j| = |\mathbf{c}'\mathbf{B}^{-1}\mathbf{a}_j| \ .$$

Since vector $\mathbf{p} = \mathbf{c}'\mathbf{B}^{-1}$ is independent of $j$, it can be computed by a BTRAN operation [4]. Having computed $\mathbf{p}$, we consider the potential improving vectors and compute $s_j = \mathbf{p}'\mathbf{a}_j$. In this case vectors with smaller $|s_j|$ can be selected.

There is another possibility. Again it is easy to see [3] that if

$$\sum_{i \in H_1} \alpha_{ij} - \sum_{i \in H_2} \alpha_{ij} > 0 \tag{13}$$

then in Phase-II we only can make a degenerate step. This observation can be utilized in the following way. Let us define vector $\mathbf{v}$ such that

$$v_i = \begin{cases} 1, & \text{if } i \in H_1 \\ -1, & \text{if } i \in H_2 \\ 0, & \text{otherwise} \end{cases}$$

With this vector we can easily compute the sum in (13):

$$\mathbf{v}'\boldsymbol{\alpha}_j = \mathbf{v}'\mathbf{B}^{-1}\mathbf{a}_j \ .$$

Here $\mathbf{q} = \mathbf{v}'\mathbf{B}^{-1}$ is independent of $j$ and can be computed by a BTRAN operation. The use of this vector can be the following. If there are several candidates on the basis of reduced cost we can drop those candidates which evidently will cause a degenerate step. Unfortunately, the opposite is not true: If we select a candidate for which (13) does not hold it still may happen that only a degenerate step can be made. In any case this criterion still can be used as a filter.

The big question mark with the network simplex algorithm is how expensive these criteria are. From a purely theoretical point of view the answer is not simple because the gain cannot be predicted. It is easier to see that extra computations per nonbasic column is necessary only when the reduced cost of the variable is positive, that is the column is a candidate. The maintenance of **p** and/or **q** does require extra work. To justify the use of them some well designed experimentation is necessary.

Presently the **p** vector approach is implemented in MINET.

## 5. THE STRUCTURE OF THE PROBLEM (INPUT) FILE

Just as the previous version, MINET V2.00 is not equipped with an own data handling facility. The idea is that network LP problems are usually formulated at a higher level of hierarchy, most probably by user programs or data base systems. MINET expects that the problem to be solved is provided in an ASCII (text) file. All the data are integers, decimal point is not allowed anywhere. Within a record the format is free. However, it is recommended to use some regular format (for instance the one which was used in V1.00) because it gives a better overview of the problem and possible errors or wrong data can more easily be detected than in the case of free format.

The structure of the file is as follows (in square brackets we give the original formats of MINET V1.00):

| | |
|---|---|
| 1. record: | $m$ (the number of nodes), $n$ (the number of arcs) [format: 2I5], |
| 2. record: | 'from' node, 'to' node, upper bound, cost coefficient of the arc [format: 2I5, 2I10], |
| . | |
| . | |
| . | |
| N+2 record: | right-hand-side elements (node requirements), five in one record, zero requirements must explicitly be given [format: 5I10], |

•

•

•

## 6. THE STRUCTURE OF THE SOLUTION FILE

The detailed solution output provides information on the arc variables and on the nodes as well. The tabular form is conventional. First the arc variables are listed in a sequential order, one record (line) for each of them. The meaning of the fields are as follows:

a) Serial number of the arc.

b) 'from' node.

c) 'to' node.

d) Status of the arc

| | |
|---|---|
| B | basic, |
| U | non-basic at upper bound, |
| _ (blank) | non-basic at lower bound. |

e) Value of the arc variable (the actual flow in the solution).

f) Capacity of the arc.

g) Unit cost of flow along the arc.

h) Relative cost (shadow price) of the arc variable in the solution, $v(j)$ of (9) and (10).

The above structure is repeated $n$ (the number of arc variables) times.

The information for the nodes is somewhat different. Again, one record per node is produced.

a) Sequential number of the node preceded by a $-$ (minus) sign.

b) Type of the node as follows

| | |
|---|---|
| 1 | if supply node, |
| $-1$ | if demand node. |

c) Status of the node variable

|  |  |
|---|---|
| B | basic, |
| __ (blank) | non-basic. |

d) Value of the node variable. If the solution is feasible this value is 0, if infeasible then this value gives the unsatisfied node requirement.

e) Node requirement (demand or supply).

f) Relative cost (shadow price) of the node variable in the solution, $w(i)$ of (9) and (10).

This structure is repeated m (the number of nodes) times.

## 7. USE OF MINET

MINET can run on IBM PC/XT, AT and compatible computers under DOS operating system. It is a fully in-core program meaning that after reading the input data it works fully within the memory. The program does not use arithmetic coprocessor so it can run equally efficiently on all computers in the above range.

The executable program is contained in file MINET.EXE. It must not be renamed and can only be loaded from the default driver and directory. The present version of MINET requires approximately 200 KByte of available memory. With this memory it can solve problems up to 1500 nodes and 8000 arcs.

To invoke MINET we simply type as a response to the prompt of the operating system:

```
MINET <Enter>
```

After this the programs gets loaded and on a clear screen the following appears:

```
MINET network LP optimizer V2.00 (c) MTA SZTAKI, 1987
Max. problem size: M <= 1500 & N <= 8000

Authorized user: IIASA
Date: 1987 Nov 11

INPUT FILE..........................................................................:
```

The user must provide the file name containing the problem. In our example it will be

```
INPUT FILE..........................................................................: med2 <Enter>
```

Full path names are supported.If the file does not exist then the program reports the error and repeats the request.

If the file is OK then the input starts and the problem size is echoed in the following way:

```
Input started:
Problem size:
        number of nodes    (M)............   74
        number of arcs     (N)............ 1399
```

When the input is completed the the user-accessible run parameters together with their default values appear in a menu-like fashion (the acceptable values are given in parenthesis):

```
Input file: ......................med2

Run parameters
msp=   1    (1,2)......pricing: 1=sectional, 2=sequential
nsc=   1    (1-n)......number of sectors
ksc=   1    (1-nsc)....number of sectors to be scanned per iteration
kvc=   1    (1-n)......number of candidates per sector
tra=   0    (0-3)......level of iteration report
npa=   3    (0-3)......strategic parameter of sequential pricing
deg=   0    (0-1)......1=anti-degeneracy column selection, 0=no

Give command (1:Display, 2:Modify, 3:Run):
```

The meaning of the parameters are as follows:

| | |
|---|---|
| msp | controls the pricing strategy.<br>If msp=1 then sectional pricing is selected,<br>if msp=2 then sequential pricing is selected.<br>In the case of sectional pricing three further parameters<br><br>nsc<br>ksc<br>kvc<br><br>are active, while the different choices of sequential pricing are controlled by parameter<br><br>npa. |
| nsc | number of sectors in sectional pricing. |
| ksc | number of sectors to be scanned at each iteration. |
| kvc | number of candidates in each scanned sector. |
| tra | level of iteration report, |

| | |
|---|---|
| tra = 0 | produces no iteration report, but at the end of Phase-I and Phase-II one iteration line is sent to the screen, |
| tra = 1 | produces one line on the screen per iteration. |

Higher trace levels are reserved for software maintenance purposes and are not explained here.

npa controls the sequential pricing if msp = 2.

| | |
|---|---|
| If npa = 0 | then scanning stops at the first improving vector (first positive $d(j)$) in Phase-I, and selects the vector with largest reduced cost $d(j)$ in Phase-II, |
| if npa = 1 | then the largest $d(j)$ is searched in both phases, |
| if npa = 2 | then the first improving vector is selected in both phases, |
| if npa = 3 | then the first improving vector is selected in Phase-I but scanning starts at the last column and goes backwards, largest $d(j)$ in Phase-II. |

deg selects the anti degeneracy column selection strategy.

To change the value of any of the parameters we have to enter the "Modify" mode by answering 2 to the command request:

```
Give command (1:Display, 2:Modify, 3:Run): 2 <Enter>
```

The answer is:

```
Modification: parameter=value (type "ok" to escape)
```

that is the explanation how a modification must be given. First the identifier of the desired parameter is to be typed followed by the = (equal) sign and the new value of the parameter. For example if we want to select sequential pricing then we type:

```
msp=2 <Enter>
```

If we give an invalid value or non numerical characters then different error messages are produced by MINET and the process can be repeated.

If we entered the "Modify" mode inadvertently then we simply can return by:

```
ok <Enter>
```

After a number of changes we may be interested in the current values of the parameters. We can display them from the "Command" mode by typing 1:

```
Give command (1:Display, 2:Modify, 3:Run): 1 <Enter>
```

When we are satisfied with the values of the parameters (in most cases it will be the default setting) then we can let the program run by:

```
Give command (1:Display, 2:Modify, 3:Run): 3 <Enter>
```

At the end of the run the qualification of the solution (OPTIMAL SOLUTION, or NO FEASIBLE SOLUTION) appears on the screen, followed by the solution time in seconds. A detailed solution is not automatically produced. If we answer Y or y to the question

```
DO YOU WANT A DETAILED SOLUTION? (Y/N) Y <Enter>
```

then a further question must be answered:

```
OUTPUT FILE.......................................................................:
```

Here we can give the name of the file where the detailed result is to be written. This will be an ASCII (text) file which can easily be processed by a user program. Full path names are supported, like:

```
OUTPUT FILE.............................................................: a:\m\med2.res <Enter>
```

The final possibility offered by MINET is:

```
CONT? (1:NEW PROBLEM, 2:NEW RUN, 3:END)
```

If we want to solve the same problem with different value(s) of the run parameters we select NEW RUN by 2. In this case the program remains loaded and the problem need not be read in again. If we want to solve another problem without reloading the program then we select NEW PROBLEM by 1. To stop run we select END by 3.

## 8. REFERENCES

[1] Greenberg, H.J., "Pivot section tactics", in H.J. Greenberg (ed.) "Design and implementation of optimization software", Sijthoff and Nordhoff, 1978, pp. 143-174.

[2] Maros, I., "MINET a Fast Network LP Solver", IIASA WP-87-50, June 1987.

[3] Maros, I., "Adaptivity in linear programming, II", (in Hungarian) Alkalmazott Matematikai Lapok 7(1981), pp. 1-71.

[4] Orchard-Hays, W., "Advanced Linear Programming Computing Techniques", McGraw-Hill, 1968.