A network can be viewed as a general case of a Linear Programming problem. If it is desirable to use the more general network format to express these types of multiobjective optimization problems, LP can be used as a special case of a Linear Programming problem. The keyword "NETWORK" indicates that there is something to look for in the code:

```
6 0 PERIOD
6 0 PERIOD
6 0 PERIOD
1 0 PERIOD
1 0 PERIOD
1 0 PERIOD
```

Here is an example of a General Form of a Generalized Linear Programming problem:

```
1 0 PERIOD
```

The General Form of a Generalized Linear Programming problem is described by the following equation:  

\[ \text{minimize} \quad c^T x \]  

\[ \text{subject to} \quad Ax \leq b \]  

where \( c, A, \) and \( b \) are given matrices and \( x \) is the vector of optimization variables. The General Form of a Generalized Linear Programming problem is a special case of the General Form of a Generalized Linear Programming problem.
If we now have a TIME file as follows, the coupling is done.

**TIME file**

<table>
<thead>
<tr>
<th>TIME</th>
<th>problem name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIODS</td>
<td>MIXED</td>
</tr>
<tr>
<td>COL1</td>
<td>ROW1</td>
</tr>
<tr>
<td>COL6</td>
<td>ROW3</td>
</tr>
<tr>
<td>NODE1</td>
<td>NODE3</td>
</tr>
<tr>
<td>COL18</td>
<td>ROW6</td>
</tr>
<tr>
<td>ENDATA</td>
<td></td>
</tr>
</tbody>
</table>

That the problem is a mixed LP/network is indicated by the keyword MIXED. Here we have that COL1 through COL6 are first stage decision variables, with ROW1 and ROW2 first stage constraints. COL6 through COL17 are second stage decision variables, with ROW3 and ROW4 as constraints. Arcs originating in NODE1 and NODE2 are third stage decision variables, arcs originating in nodes NODE3 through NODE8 belong to stage 4. Finally, all variables corresponding to columns COL18 through whatever the second COLUMNS section dictates are fifth stage decisions with all constraints after ROW5 associated with them.

The last two entries in the first COLUMNS section will take values from the second to the third stage. The amount will be determined by the values of COL16 and COL17 and the corresponding entries in this COLUMNS section.

The last two entries in the ARCS section will bring “row” from NODE8 to the right hand side of ROW5 and ROW6 by entering a number in those rows. The number will be the “multiplier” from the input, and the value ending up on the right hand side will be the product of this multiplier and the flow running out of NODE8 to these rows (which are nodes when viewed from the network).

**Acknowledgements**

The authors are grateful to Dalhousie University and the Natural Sciences and Engineering Research Council of Canada for providing facilities and support for this research.

**References**


---

**Core file sections for networks**

1. **NAME** – starts the input file. The rest of the line can be used for the problem name.

2. **ARCS** section – each data line following the ARCS header specifies input for one arc. In the first name field is the name of the originating node for the arc, in the second name field the name of the terminating node, in the first numeric field the unit cost, in the second numeric field the upper bound on arc flow, in the third numeric field the lower bound on the flow (if not zero) and in the fourth numeric field the arc multiplier (arc gain) if we are dealing with generalized networks. If the word UNCAP follows the ARCS code, upper and lower bounds need not be specified as they are assumed to be +∞ and zero, respectively. Similarly, the keyword UNDIR signals an undirected network with default bounds of +∞ and –∞.

3. **SUPPLY** (optional) – each data line following the SUPPLY header contains a node name in the first name field and the amount supplied in the second numeric field.

4. **DEMAND** (optional) – each data line following the DEMAND header contains a node name in the first name field and the amount demanded in the second numeric field.

5. **ENDATA** – informative. End of problem data.

Note that there is no section naming all nodes (i.e., rows). They are named implicitly by their appearance in the ARCS section. Also note that arcs (i.e., columns) have no names. Hence they cannot be referred to by name, only by a pair of node names. However, if there are parallel arcs, the user must be careful.

**Core file – example**

```
NAME     problem name
ARCS
    NODE1   NODE2   cost   upper   lower   multiplier
    NODE1   NODE3   cost   upper   lower   multiplier
    ......    ......    ......    ......    ......    ......    ......
    NODEk   NODEn   cost   upper   lower   multiplier
SUPPLY
    ......    ......    ......    ......    ......    ......    ......
    NODE1
DEMAND
    ......    ......    ......    ......    ......    ......    ......
ENDATA
```

---

**3.2 Time File**

It is normally assumed that in a NETGEN file, all arcs originating in a given node are given before we start giving arcs originating in the next node. This rule should be followed. We shall further assume that FROM-nodes are given in node order, in the sense that if the arcs originating in NODEi occur before those originating in NODEj, then NODEj belongs to the same or a later time period. However,
ENDA

3.1 Core File

- column 1-60: fourth numeric field
- column 41-50: third numeric field
- column 31-40: second numeric field
- column 21-30: first numeric field
- column 11-20: second name field
- column 2-10: code field
- column 1-10: code field

Data for networks

Netgen we have the following operation of the data:...
AN ALGORITHM FOR DISJUNCTIVE PROGRAMMING PROBLEMS

by Nicholas Beaumont
Monash University
Clayton, Vic. 3168, Australia

1. INTRODUCTION

In some mathematical programming problems binary variables (hereafter called logical variables) are introduced to express logical relationships amongst constraints (see Williams 1978, p.159). The commonest example is either a setup cost being incurred or production being zero. The resulting MIP problem is usually solved by relaxing the MIP problem and forming a tree based on the arbitration of the logical variables - the Branch and Bound algorithm.

This method has a number of disadvantages:

1. Matrix generation is complicated by the need to re-express logical relationships in terms of logical variables and "simple" constraints.

2. It enlarges the problem by introducing extra rows, columns and coefficients.

3. It seems inelegant to express, e.g. a binary disjunction of constraints as two rows knowing that in any integer feasible solution at least one of the rows will be slack.

These considerations lead to a re-examination of the idea (Rado 1966) of arbitrating logical constraints (a "logical constraint" is a logical combination of "simple" constraints) instead of logical variables. For brevity only disjunctions of constraints will be discussed, that is groups of constraints of which at least one must be true.

2. THE ALGORITHM

2.1 The Surrogate

The relaxation of the condition \( \delta = 0 \) OR \( \delta = 1 \) is \( 0 < \delta < 1 \). We define a surrogate of a disjunction as a simple constraint which is implied by the variables' bounds and any term of the disjunction. We can then relax a disjunctive programming (DP) problem by replacing each disjunction by a surrogate.

A surrogate for the logical condition that at least \( k \) of the \( n \) constraints

\[ \sum_{j=1}^{n} a_{ij} x_j < b_i \quad \forall i \in \{1, 2, \ldots, m\} \quad (1) \]

is a single "scenario". The nodes visited by the path correspond to certain values assumed by certain entries of the matrices in the core file. Thus a scenario is completely specified by a list of column/row names and values, and a probability value. Once a single given scenario is described, then other scenarios that branch from it may be described by indicating in which period the branch has occurred, and then listing the subsequent column/row names and values. It is best to work through the example of Figure 2.

Figure 2. Scenarios example - event tree

There are two types of data lines. The first, signified by SC in the code field, gives the name of the scenario in the first name field and its probability in the first numeric field; and then gives the name of the scenario from which the branch occurred and the name of the period in which the branch occurred—i.e., the first period in which the two scenarios differ—in the second name field and third name field, respectively. A scenario that originates in period one is indicated by ROOT in the name field. The next data lines give the column/row values assumed by the scenario.
In a language more specific to our application, a "path" in the tree is defined as a sequence of nodes where each node is connected to the next by an edge. The set of all paths in the tree represents the set of all possible solutions. A solution is considered to be a path from root to leaf, where each node in the path represents a decision made at that point.

The decision tree is constructed by recursively splitting the set of all possible solutions into smaller subsets based on the decisions made at each node. The split is determined by a decision function that evaluates the current state of the problem and chooses the best decision to make next.

In order to construct the tree, we start with the root node, which represents the initial state of the problem. At each node, we evaluate the decision function to determine the best decision to make next. This decision is represented by an edge that leads to a new node, which becomes the parent of the current node.

The decision function takes into account the current state of the problem, which includes the current node and the remaining nodes in the tree. It evaluates the possible outcomes of each decision and selects the decision that leads to the best outcome, according to some criteria such as minimizing the risk or maximizing the expected value.

The tree is grown by recursively applying the decision function to each node, until all nodes have been evaluated and the tree is fully constructed. The resulting tree provides a clear and structured representation of the problem, making it easier to understand and analyze.

For example, consider a decision tree representing a series of decisions to be made in a business setting. The root node represents the initial state of the business, and each subsequent node represents a decision that needs to be made. The decision function at each node evaluates the possible outcomes of each decision, and selects the decision that leads to the best outcome, according to some criteria such as maximizing profits or minimizing risk.

The tree is grown by recursively applying the decision function to each node, until all nodes have been evaluated and the tree is fully constructed. The resulting tree provides a clear and structured representation of the problem, making it easier to understand and analyze.

In conclusion, the use of decision trees can be a powerful tool for solving complex problems in a structured and systematic way. By breaking down the problem into smaller, more manageable pieces, we can more easily understand the possible outcomes of each decision and make informed choices that lead to the best possible outcome.
Step 4: The optimal feasible solution is the last solution stored in Step 2. If no solution has been stored, the problem is infeasible.

3. CRITERIA

As in the conventional algorithm the total number of nodes which have to be examined is much affected by the way in which the tree is built and traversed. The questions arising in the disjunctive approach are:

1. Which problem to select from the set of unresolved problems.
2. Which disjunction to arbitrate.
3. Which term of the disjunction is to replace the surrogate.

Obvious criteria are penalties obtained by anticipating the first dual iteration, pseudocosts and user supplied priorities.

3.1 Other Criteria

Rado (1966) suggests a maximin criterion. For each disjunction, note the term which has the smallest "infeasibility" and let this be (by definition) the infeasibility of the disjunction. Arbitrate on the disjunction which has the largest infeasibility. The infeasibility is defined as $\sum_{j=1}^{n} a_j x_j - b$ for a constraint of form $\sum_{j=1}^{n} a_j x_j < b$ and analogously for other forms.

We suggest a surrogate criterion.

1. Arbitrate on the infeasible disjunction whose surrogate has the highest non-zero shadow price.
2. If no disjunction is chosen in (1), arbitrate on the infeasible disjunction whose surrogate has the smallest slack.
3. In either case select the term with the smallest infeasibility gap.

For some problems (1 and 2) this criterion seems to be effective, evidently giving a good measure of the importance of a disjunction. This criterion appears to dominate the maximin criterion. Dimensional considerations suggested multiplying the shadow price by the infeasibility, but this worsened performance.

4. IMPLEMENTATION AND RESULTS

The conventional and disjunctive algorithms were compared by coding (in FORTRAN 77 on a VAX/750) the Land-Doig algorithm (Land and Powell 1973) with MINOS 4.0 (Murtagh and Saunders 1977) being used to solve subproblems. Both programs use MPS input format except that the disjunctive program reads row cards of type "OR" and "AN" to define logical constraints.

2.3.4 Scenarios

To describe scenarios one needs a data structure that expresses inter-period dependencies. This is best developed as a description of the distribution of a process vector in the periods, $t = 1, \ldots, T$, just as one may describe a stochastic process in probability theory. We consider the random entries of $(c_t, b_t, A_{ts}, s \leq t)$ as states of a process vector $\varphi_t$, for each time $t = 1, \ldots, T$. Given the corresponding (finite dimensional) joint distributions of this process $\varphi$, Kolmogorov's construction yields a probability measure $P_t$ termed the process distribution, on the space $\Omega$ of trajectories. Thus, in general, we have the alternatives of describing the distribution of the stochastic process $\varphi$ in joint or conditional state distribution form, or as a process distribution over trajectories.

![Event tree representation of scenarios](image)

Figure 1. Event tree representation of scenarios

More specifically, with the stochastic process $\varphi$ is associated a filtration $\{\mathcal{F}_t : t = 1, \ldots, T\}$, where for each $t$ the sigma algebra $\mathcal{F}_t$ consists of subsets of $\Omega$ termed events determined by the history of the process $\varphi$ up to time $t$, and $\mathcal{F}_t \subset \mathcal{F}_{t+1}$ for
Problem 3 demonstrates that the disjunction approach works well with the combination approach. When done, the disjunction approach can also be used to determine whether a disjunction of clauses is satisfied.

The formulation is correct and perhaps more natural, but does not give the same results.

\[
\begin{align*}
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij} \\
\text{c} & = \sum_{i=1}^{L} \sum_{j=1}^{K} \lambda_i \lambda_j \beta_{ij} \theta_{ij}
\end{align*}
\]

This example illustrates the structure in this case.
CONCLUSION

The disjunctive algorithm shows some advantages (simpler formulation, a smaller matrix, a smaller tree and less CPU time) on some kinds of problem. Comparisons of performance might be affected, e.g. by tolerances.

The disadvantages are:

1. In general, nodal problems are formed by altering the matrix. This implies that each subproblem must start with a re-inversion. If, as is common, the terms of a disjunction are bounds, arbitration can be (and is) expressed by altering variable bounds in which case re-inversion is not needed. It should be noted that all subproblems in both programs start with a re-inversion.

2. Generating, storing and altering the constraints is much more difficult and time consuming than retaining the values of arbitrated variables.

It is intended to speed up both programs (especially by removing unnecessary re-inversions) and do more benchmarking. Suitable test problems would be much appreciated.

A longer paper which outlines some extensions is available from the author.

REFERENCES


...in a similar fashion to the normal, using the standard descriptors as presented in, for example, Raiffa and Schlaifer [16]. An adjustment to other intervals could be effected within the framework of the linear transformations described below.

Subroutine. Some random entries may have distributions that are computed by subroutines, for example, empirical distributions which are discretely distributed but whose values may be randomly generated by user-supplied computer codes.

```
INDEP SUB
COL1 ROW8 blank PERIOD2
```

This example indicates that the user must access a subroutine to generate an appropriate distribution for the entry COL1/ROW8.

2.3.3 Blocks

Blocks may be regarded as mutually independent random vectors. We provide for three distribution types: discrete, subroutine, or linear transformation. As in the independent case, blocks with common distribution types are grouped in the same section under a header line.

Discrete. The "values" of a block are actually vectors of values of the entries that make up the block, and to each value of a block there corresponds a probability. We need two sorts of data lines to describe a block. The first line, distinguished by a BL in the code field, gives the name of the block, the name of the period in which the block is realized, and the probability that the block assumes a given vector value; the following lines identify which entries of the block assume which value.

```
<table>
<thead>
<tr>
<th>BLOCKS</th>
<th>DISCRETE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL BLOCK1 PERIOD2 0.5</td>
<td></td>
</tr>
<tr>
<td>COL1 ROW6 83.0</td>
<td></td>
</tr>
<tr>
<td>COL2 ROW8 1.2</td>
<td></td>
</tr>
<tr>
<td>BL BLOCK1 PERIOD2 0.2</td>
<td></td>
</tr>
<tr>
<td>COL2 ROW8 1.3</td>
<td></td>
</tr>
<tr>
<td>BL BLOCK1 PERIOD2 0.3</td>
<td></td>
</tr>
<tr>
<td>COL1 ROW6 84.0</td>
<td></td>
</tr>
</tbody>
</table>
```

One needs to record only those values that change. We adopt the convention that the first statement of the block is the basis from which all changes are computed. (Thus zero values must be stated explicitly.) In this example the block, called BLOCK1, is the 2-vector made up of the entries COL1/ROW6 and COL2/ROW8. It takes values (83.0, 1.2) with probability 0.5, (83.0, 1.3) with probability 0.2, and (84.0, 1.2) with probability 0.3.

Subroutine. The user accesses a subroutine to compute the distribution of the block consisting of the listed entries.
normal on [0, oo) are two-parameter families of distributions and may be handled
belonging to the standard family on [0,1] the normal

Normal. The distribution is specified by mean \mu \text{ and variance } \sigma^2

In this example the random entity COL1/ROW2 is uniformly distributed over the in-

Independent. The endpoints of the interval are the only relevant parameters for our

are only mutually exclusive if

TABLE 1

<table>
<thead>
<tr>
<th>PROBLEM NUMBER</th>
<th>VARIABLES</th>
<th>CONSTRAINTS</th>
<th>COEFFICIENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

In this example the entity COL1/ROW2 takes values 6 with probability 0.2 and 0.4 with

0.4 PERIOD 2

0.3 PERIOD 1

0.2 PERIOD 0

0.1 PERIOD 0

0.6 PERIOD 2

0.2 PERIOD 0

0.4 PERIOD 2

TABLE 2

<table>
<thead>
<tr>
<th>PROBLEM NUMBER</th>
<th>VARIABLES</th>
<th>CONSTRAINTS</th>
<th>COEFFICIENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

The flight and second numeric values.

...
RUNNING THE NATO WORKSHOP FOR COAL
Sverre Storøy
Department of Informatics, University of Bergen, Norway
Stein W. Wallace
Chr. Michelsen Institute, Bergen, Norway

Now, as the workshop is history, and our file has been closed by NATO, it seems to be
time to share some of our experiences with the readers of COAL Newsletter. This is not
going to be an encyclopaedia of how to run a workshop, but hopefully we can give some
useful advice.

For those who did not take part in the event, let us briefly mention that we are talking
about a workshop with limited participation (about 40), and where the participants were
invited based on submitted abstracts. The workshop ended up being supported as a NATO
Advanced Research Workshop.

Below are three sections. The first indicates the amount of time needed to plan and run
the workshop, the second which things went wrong, and the third discusses aspects of the
workshop that we think worked nicely. Hopefully all three sections can be of some help to
people in a situation similar to ours.

How much time did it take to plan and run the workshop? Since the workshop was run at
Chr. Michelsen Institute, a private independent, non-profit research institute, some of the
procedures of such an institution help us in evaluating the actual time spent on planning.
Based on very detailed time-sheets, we are able to estimate the total number of hours
spent on planning and running the workshop to about 600, or about 1/3 of a man-year.
This includes everything from making an initial contact with COAL, via applying twice
to NATO, sending out the organization of the event itself and cleaning it up afterwards. It
is our firm belief that if we had to do it again, we would still use about the same amount
of time. There are some errors we would not do again (see below), but new errors would
certainly show up.

Errors we made in the planning. There are clearly several errors of minor importance that
were made during the planning process. Below follows a list of errors that we hope it will
be useful to think about also for other who wish to run a workshop.

Call-for-papers. As pointed out above, participants were invited based on submitted ex-
tended abstracts. Hence, in the call-for-papers we asked people to send us such abstracts
together with a registration form. At this point we made what we believe to be our major
error.

In this example: Columns COL1 through COL5 are PERIOD1 decision variables
and COL6 and COL7 are PERIOD2 variables; rows ROW1 and ROW2 are PERIOD1 con-
straints and ROW3 through ROW18 are PERIOD2 constraints. All remaining rows and
columns belong to PERIOD3.

Other possible keywords on the PERIODS line are NETWORK for pure network
problems and MIXED for coupled LP/network problems. These are explained in
some more detail in sections 3.2 and 4, respectively.

2.3 Stoch file

In the stoch file the distributions of the random variables are specified. As
mentioned, we consider three varieties of distributions: independent, blocks, and
scenarios. Each type will be treated in separate sections of this file; each section
consists of a header line followed by data lines.

Stoch file – header lines
1. STOCH – informative. Identifies a new problem with a give name in the second
   word field.
2. INDEP section – specifies the distribution of all independent random entries in
   separate sections for each type.
3. BLOCKS section – specifies the joint distribution of all dependent random entries
   in separate sections for each type.
4. SCENARIOS section – specifies the scenarios.
5. ENDATA – informative. End of problem data.

2.3.1 A note on distributions

The purpose of the stoch file is to give the user the information needed to com-
pute with the random variables. In many applications the distributions are discrete
or discrete approximations of (absolutely) continuous distributions; thus the user
needs, ultimately, to know what value the random variable takes and with what
probability. The discrete case is straightforward—this information may be explic-
itly provided in the stoch file and then stored in appropriate data structures by the
user. In the continuous case users may have their own discretization scheme and
may need only to know the parameters and type of the continuously distributed
random variables. Such data is easily provided; however, users must then process
it themselves to obtain a discrete approximation. Finally there are cases where the
random variables may be accessed only through a user-supplied subroutine—for ex-
ample the output of a random number generator of nonstandard type. Alternatively,
the user may be able to compute directly with certain continuous distributions and
may build approximations to more general distributions based on them—for exam-
ple the piecewise linear and piecewise quadratic distributions investigated by Wets
[17] and Birge and Wets [2]. This information is easily transmitted using the various
data structures described below.
The flow of the page is difficult to follow due to the fragmented sections and the lack of clear headings. The text appears to be discussing the structure of a process or system, but it is not organized in a coherent manner.

**Time - The Example**

Core fix - Example

**The Flow of the Process**

A detailed description of the process is provided, but it is not easy to follow due to the fragmented sections and the lack of clear headings.

**Time Fix - The Example**

A more detailed description of the process is provided, with clear headings and sections. The text is easier to follow and understand.

**Time Fix - The Example**

A more detailed description of the process is provided, with clear headings and sections. The text is easier to follow and understand.
trust that many problems will become available using the basic elements to the fullest extent—with tailored modifications only when absolutely necessary.

2. Standard Format for Linear Programs

2.1 Core File

The core file is sketched only briefly since it closely follows the MPSX standard [6]. The core file consists of sections introduced by header lines. Data lines follow the headers. We assume that all appropriate dimensioning has been done before the files are read (for example in a file similar to the SPECS file for the MINOS program [12]).

A data line is divided into six fields: three name fields, two numeric fields and one code field.

Data line for LP

- columns 2 and 3: code field
- columns 5–12: first name field
- columns 15–22: second name field
- columns 25–36: first numeric field
- columns 40–47: third name field
- columns 50–61: second numeric field

A name is treated as a character string and may contain any ASCII symbol. Only numbers with decimal point, or in scientific notation, may appear in numeric fields.

Core file sections for LP

1. NAME — starts the input file. The second word field is used to identify the problem.
2. ROWS section — each data line specifies the names of the objective c and rows of the matrix A in the first name field, and the type of constraint (E, L, G, H) in the code field. The list of row names must be in order from first period to last, preceded by the objective name(s).
3. COLUMNS section — each data line specifies the column names and the nonzero values of c and A. This must be done in column order.
4. RHS section — data lines specify nonzero entries of the righthand side b.
5. BOUNDS (optional) — data lines specify the bound, u and ς, using the codes LO or UP in the code field.
6. RANGES (optional) — cf. the MPSX standard [6].
7. ENDDATA — informative. End of problem data.
Rather we hope that the proposal offers sufficient heuristics to be useful and we expect further elaboration in our almost complete draft of our proposed \textit{specific} section of the paper. This draft of our proposal section is in the process of being completed. However, one should not expect it to include a formal presentation of our ideas. We do not propose to include a formal presentation of our ideas.

We do not expect this section will be detailed. However, the first column and possibly also the second column of this section will be detailed. This section will be detailed in the appropriate sections.

There are different approaches for handling programs. This is not enough to develop programs. These are different approaches for handling programs.
CALL FOR PAPERS

A special issue of Mathematical Programming Series B devoted to large scale problems is being prepared under the joint editorship of A. R. Conn, N. I. M. Gould and Ph. L. Toint.

The emphasis is intended to be towards large scale nonlinear programming rather than linear programming. However, articles that emphasize applications, algorithms and/or theory that is not primarily linear programming oriented could be suitable and are solicited.

Mathematical Programming Series B is now on a similar schedule to Series A with similar standards for publication and the same circulation. The advantage of a special issue, besides collecting articles on a coherent theme is that the refereeing process can often be expedited.

If you have a suitable paper for such an issue you are invited to submit it to one of the editors:

A. R. Conn
Department of Combinatorics and Optimization
Faculty of Mathematics
University of Waterloo
Waterloo, Ontario N2L 3G1
Canada
Tel: 519 888 4458
E-mail: arconn@water.bitnet
arconn@water.waterloo.edu

N. I. M. Gould
Computer Science and Systems Division
A.E.R.E. Harwell
Oxfordshire OX11 ORA
U.K.
Tel: 235-24141 ext. 3357
E-mail: nimgould@water.bitnet
nimgould@water.waterloo.edu

Ph. L. Toint
Department of Mathematics
Facultes Universitaires Notre Dame de la Paix
Rue de Bruxelles, 61
B-5000 Namur, Belgium
Tel: 32-81-22061 ext. 2592
E-mail: phtoint@bnandp10.bitnet

The deadline for submission is August 1, 1988 for the issue that is to be published in 1989. Late submissions, an excess of submissions etc. would automatically be considered for Mathematical Programming Series A, unless the author wishes otherwise.

where the functions $Q_1, Q_2, \ldots, Q_T$ are defined recursively:

\[
Q_1(z_1) = \min \{ c_1 z_1 + E_1 Q_1(z_1, z_2) \}
\]

subject to

\[
l_t \leq z_1 \leq u_t
\]

\[
A_{11} z_1 + A_{12} z_2 = b_1
\]

\[
Q_2(z_1, z_2) = \min \{ c_2 z_2 + E_2 Q_2(z_1, z_3) \}
\]

subject to

\[
l_t \leq z_2 \leq u_t
\]

\[
A_{21} z_1 + A_{22} z_2 = b_2
\]

\[
A_{21} z_1 + A_{22} z_2 + A_{23} z_3 = b_3
\]

and so forth, for $t = 4, \ldots, T - 1$, where "$E_t$" represents expectation with respect to the random variables in period $t$, until finally

\[
Q_T(z_1, \ldots, z_{T-1}) = \min \{ c_T z_T \}
\]

subject to

\[
l_T \leq z_T \leq u_T
\]

\[
A_{T1} z_1 + \cdots + A_{TT} z_T = b_T
\]

The data defining this problem may be conveniently arranged in an LP formulation for a single realization of the random variables:

objective: $c_1 z_1 + c_2 z_2 + \ldots + c_T z_T$

constraints: $z_t \in \mathbb{R}^{m_t}$, $t = 1, \ldots, T$

$t_t \leq z_t \leq u_t$, $t = 1, \ldots, T$

\[
A_{11} z_1 = b_1 \in \mathbb{R}^{m_1}
\]

\[
A_{21} z_1 + A_{22} z_1 = b_2 \in \mathbb{R}^{m_2}
\]

\[
A_{T1} z_1 + A_{T2} z_T + \ldots + A_{TT} z_T = b_T \in \mathbb{R}^{m_T}
\]

(MP)

All entries of the matrices $A_{it}$ and vectors $c_t, t_t, u_t, b_t$ may be random (although in practice all but a few entries will be deterministic). The indices $t_t = 1, \ldots, T$ signify the periods of the problem; to each period $t$ there corresponds a decision vector $s_t \in \mathbb{R}^{m_t}$. The lower block-triangular constraint system expresses the typical feature of these problems—the decisions of the prior periods constrain the decision of the current period explicitly (since those decisions are known) but the decision of the current period is affected only implicitly by the feasibility and costs of possible future recourse decisions.

The proposed format is most easily understood by considering the problem (MP) in stages of gradually increasing levels of detail. First we regard (MP) as a
A general form of the multiperiod stochastic linear program is:

\[ \min \{ c_1^T x_1 + c_2^T x_2 \} \]

subject to:

\[ \begin{align*}
    & A_1 x_1 + A_2 x_2 + b_1 = y_1 \\
    & x_1 \leq x_2
\end{align*} \]

1. **Problem Statement**

   - **Accessible Form**: The accessible form of the standard problem format are available in computer

   - **Mixed Stochastic Network and Linear Program**: The mixed stochastic network and linear program.

   - **Problem Formulation**: The problem formulation.

   - **Parameters**:

     - The problem parameters.

   - **Program Committee**:

     - The program committee.

   - **Application Areas**: The application areas.

   - **Nonlinear Optimization**: The nonlinear optimization.

   - **Main Themes**: The main themes.

   - **October 10-14, 1988 Workshop on Mathematical Programming**
COAL OBJECTIVES

The Committee on Algorithms is involved in computational developments in mathematical programming. There are three major goals: (1) ensuring a suitable basis for comparing algorithms, (2) acting as a focal point for computer programs that are available for general calculations and for test problems, and (3) encouraging those who distribute programs to meet certain standards of portability, testing, ease of use and documentation.

NEWSLETTER OBJECTIVES

The newsletter's primary objective is to provide a vehicle for the rapid dissemination of new results in computational mathematical programming. To date, our profession has not developed a clear understanding of the issues of how computational tests should be carried out, how the results of these tests should be presented in the literature, or how mathematical programming algorithms should be properly evaluated and compared. These issues will be addressed in the newsletter.

A STANDARD INPUT FORMAT FOR MULTIPERIOD STOCHASTIC LINEAR PROGRAMS

J.R. Birge
Department of Industrial and Operations Engineering
University of Michigan, Ann Arbor, USA

M.A.H. Dempster
Department of Mathematics, Statistics and Computing Sciences
Dalhousie University, Halifax, Canada
and Balliol College, Oxford, England

H.I. Gassmann
School of Business Administration
Dalhousie University, Halifax, Canada

E.A. Gunn
Department of Industrial Engineering
Technical University of Nova Scotia, Halifax, Canada

A.J. King
International Institute for Applied Systems Analysis
Laxenburg, Austria

S.W. Wallace
Chr. Michelsen Institute, Bergen, Norway

ABSTRACT

Data conventions for the automatic input of multiperiod stochastic linear programs are described. The input format is based on the MPSX standard and is designed to promote the efficient conversion of originally deterministic problems by introducing stochastic variants in separate files. A flexible "header" syntax generates a useful variety of stochastic dependencies. An extension using the NETGEN format is proposed for stochastic network programs.
## Mathematical Programming Society

### Committee on Algorithms Newsletter

<table>
<thead>
<tr>
<th>NO. 17</th>
<th>JENS CLAUSEN</th>
<th>EDITORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECEMBER 1987</td>
<td>ROBERT R. MEYER</td>
<td></td>
</tr>
</tbody>
</table>

### CONTENTS

- A Standard Input Format for Multiperiod Stochastic Linear Programs  
  J. R. BIRGE, M. A. H. DEMFSTER, H. I. GASSMANN, E. A. GUNN, A. J. KING, S. W. WALLACE  
  1
- An Algorithm for Disjunctive Programming Problems  
  N. BEAUMONT  
  20
- Running the NATO Workshop for COAL  
  S. STORØY, S. W. WALLACE  
  26
- Calendar  
  28
- Call for Papers  
  30
- Workshop on Mathematical Programming  
  31