TR/05/94 April 1994

ALGORITHMS FOR NETWORK PIECEWISE-LINEAR PROGRAMS

F.A.S. Marins
A. Machado
C. Perin

ALGORITHMS FOR NETWORK PIECEWISE_LINEAR PROGRAMS

F.A.S. MARINS^{1,2}, A. MACHADO^{3} andC. PERIN^{3}
fernando.marins@brunel.ac.uk clovis@ime.unicamp.br

ABSTRACT

In this paper a subarea of Piecewise-Linear Programming named network Piecewise-Linear Programming (NPLP) is discussed. Initially the problem formulation, main definitions and related Concepts are presented. In the sequence of the paper, four specialized algorithms for NPLP, as well as the results of a preliminary computational study, are presented.

1. INTRODUCTION

An important area of the Mathematical Programming is the Piecewise-Linear Programming (PLP), which is related with the minimization of a convex separable piecewise-linear objective function, subject to linear constraints. The relevance of this topic is justified by its many theoretical and practical applications in L_1 estimation, determination of initial feasible solution for linear programs, as well as in Goal, Nonlinear or Interval Programming, for instance (see [1]).

In this paper a subarea of PLP named Network Piecewise-Linear Programming NPLP) is explored. Several Real-World applications of NPLP are found in Telephone Network Expansion, Power Distribution Networks and in Planning Operation of Water Multireservoir Systems (see [2]).

The approach adopted by the authors has been to propose and to implemented specialized algorithms for solving NPL Programs directly, so the use of transformations (see [6]) from NPL programs to equivalent LP programs are not necessary. Here are presented four algorithms for NPLP: Primal (Strongly Feasible) Simplex, Dual-Method, Out-Of-Kilter, and Cost-Scaling (Strongly Polynomial).

[1]UNIVERSIDADE ESTADUAL PAULISTA-CAMPUS DE GUARATINGOETÁ-DEPARTAMENTO DE PRODUCÃO-CEP 12500-000-GUARATINGUETÁ-SP-BRAZIL.

[2]BRUNEL UNIVERSITY-DEPARTMENT OF MATHEMATICS & STATISTICS. UXBRIDGE - MIDDLESEX, UB8 3PH - ENGLAND, UK.

[3]UNIVERSIDADE ESTADUAL DE CAMPIMAS - INSTITUTO DE MATEMÁTICA, ESTATISTICA E CIENCIA DA COMPUTÃO. CEP 13061-970 CAMPIMAS - SP - BRAZIL

The final intent of this research is to study the relative effectiveness of these algorithms. Here are only presented preliminary results of algorithms' performance.

2. PROBLEM STATEMENT, DEFINITIONS AND CONCEPTS

In this section, we introduce the Network Piecewise-Linear Programs formulation, relevant definitions and related concepts necessary to algorithms' presentation (see [2], [3], [4], [5], and [6], for details).

Let G = (N,E) be a direct network with |N| = n nodes and |E| = m arcs. The vector b = (b_i) denotes the nodes demands satisfying $\sum_i$ b = 0, and A = (a_{ij}) is the node-arc incidence matrix, defined by $a_{tj,j} = -1$, $a_{hj,j} = +1$ and $a_{i,j} = 0$ if $i \neq t_j, h_j$, where t_j, and h_j are the tail and the head of the arc $j \in E$, respectively.

For each arc j is associated a nonnegative variable flow x as well as a sequence of juxtaposed intervals [$d^k{}_j$, $d^{k+1}{}_j$] ,see figure 1, corresponding to a sequence of increasing cost coefficients $c^k{}_j$. When $x_j \in [d^{fj}{}_j , d^{fj+1}{}_j]$, where f_j is named the current interval of the arc j, the value of the piecewise-linear function C_j (X_j) is given by:

$$c_j(x_j) = \sum_j \left[c^k{}_j(d^{k+1}{}_j - d^k{}_j) \ : \ k = 1..f_j - 1\right] + c^{fj}{}_j(x_j - d^{fj}{}_j).$$

A Network Piecewise-Linear Program is related to find an arc flow vector x = (X_j) in G, optimal solution to MIN {C(x): Ax = b, x ≥ 0}, where $C(x) = \Sigma_j C_j(x_j)$ is a convex separable piecewise-linear objective function.

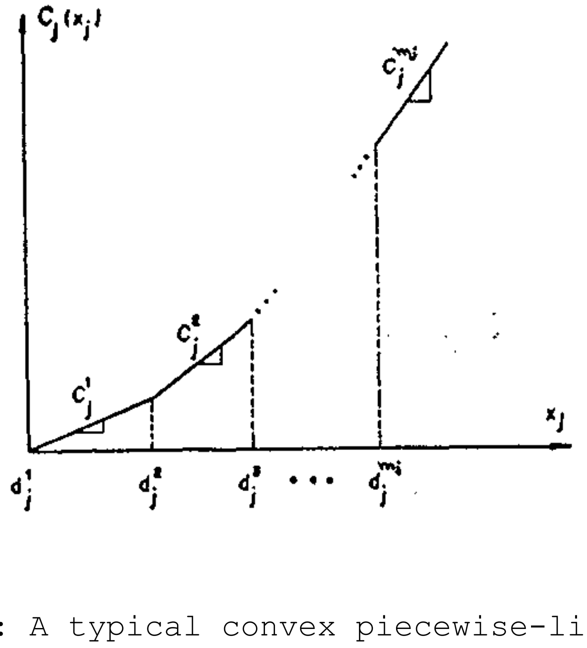


Figure 1: A typical convex piecewise-linear function associated with arc j ∈ E.

The Dual Program related to this NPL Program is to find a node price vector $y = (y_j)$ and an arc current cost vector $z = (z_j)$ optimal solution to the problem

MAX $(yb - D(z):\ y^A - z \leq o,\ z^j \geq c^i_j,\ j \varepsilon E\}$,

where $D(z) = \sum_j D'_j(z_j)$, and

$$D'_j(z_j) = \sum_j \left[d^{k+}{}_j(c^{k+}{}_j - c^k_j) : k = 1..f_j - 1\right] + d^{fj+1}{}_j(z_j - c^{fj}_j).$$

Observe that f_j is the current interval of z_j and satisfies $z_j\, \varepsilon \left[c^{fj}{}_j\, , c^{fj+1}{}_j\right]$ with $f_j \in \{1..\ m_j\ \}$, where $m_j$is the total number of juxtaposed intervals related to arc j.

Every basis of the NPL Program is associated with a spanning tree T and an arc current interval vector $f = (f_j)$.

Consider a distinguished and fixed node to be the root of T. For each node i there is a unique path p_i, in T that connects node i to the root. A basic arc $j \in T$ is said to be down (up) if it is directed away (to) the root in T.

The addition of a nonbasic arc s creates, in T U {s}, a cycle $Q_s = Q^+_s \cup Q^-_s$ defined by $Q^+_s = \{j \in Q_s$: j, s have the same direction in $Q_s\}$ and $Q^-_s = Q \setminus Q^+_s$.

The removal of a basic arc r creates a subtree that defines a node partition $(S_r, N \setminus S_r)$, where $S_r = \{i \in N: r \in P_i\}$.

The cut-set $K_r = K^+_r \cup K^-_r$, associated with the basic arc r, is defined by the subsets of arcs $K^+_r = \{j \in E \mid t_j\, \varepsilon\, s_{r,}\ h_j \in N \setminus s_r\}$ and $K^-_r = \{j \in E \mid t_j \in N \setminus s_r,\ h_j \in s_r\}$.

Given a spanning tree T and an arc current interval vector $f = (f_j)$, the unique associated basic solution (x,y,z), consisting of an arc flow vector $x = (X_j)$, a node price vector $y = (Y_j)$, and a current interval vector $z = (Z_j)$,is computed as follows:

$Y_i = 0$ if i is the root,

$Y_i = \sum_j [c^{fj}_j : j \in$ T is an up arc in $P_i] - \sum [c^{fj}_j : j \in$ T is a down arc in P_i] if $i \neq$ root,

$z_j = c^{fj}_j j$ if $j \in T$,

$z_j = \max\{Y_{hj} - Y_{tj}\ ,\ c^1{}_j\}$ if $j \in E \setminus T$,

$X_j = d^{fj}_j$ if $j \in E \setminus T$, and

X_j $j \in T$ are computed by back substitution in $Ax = b$.

NOTES: (a) There is no need to consider other dual variables besides Y in order to work with the dual method;
(b) A vector x' is a primal feasible flow if Ax' = b, and $x' \geq 0$;
(c) A primal feasible flow x is optimal to the NPL Program

if $C(x) \leq C(x')$ for every primal feasible flow x'.

For each $j \in E$ define the left and the right reduced costs as below:

for $j \in T$, let $c^-_j = c^+_j = 0$,

for $j \in E \setminus T$, let $c^-_j = c^{fj-1}{}_j - y_{hj} + y_{tj}$, and $c^+_j = c^{fj}{}_j - y_{hj} + y_{tj}$.

It is possible to show that a basic solution is dual feasible when $c^-_j \leq 0$ and $c^+_j \geq 0$ for $j \in E$.

A basic dual feasible solution is called optimal if for every arc $j \in T$, $x_j \in [d^{fj}_j, d^{fj+1}_j]$, assume $c^{fj-1}_j = d^{fj-1}_j] = -\infty$, and $c^{fj+1}_j = d^{fj+1}_j = \infty$, if they are not defined.

Some specific definitions useful to the Out-of-Kilter algorithm description are presented in the section 3.3.

Finally, additional concepts (generalizations of the concepts introduced in [7]) are necessary due to Cost-Scaling algorithm presentation in section 3.4.

Let a primal feasible flow x such that $x_j \in [d^{fj}_j, d^{fj+1}_j]$. Given $\alpha > 0$, x is called α-optimal if it satisfies:

$$c'_j < -\alpha \quad \Rightarrow \quad X_j = d^k_j,$$

$$d^k_j < X_j < d^{k+1}_j \quad \Rightarrow \quad \alpha < c'_j < +\alpha,$$

$$c'_j > +\alpha \quad \Rightarrow \quad X_j = d^{k+1}_j,$$

where $k = f_j$, and $c'_j = c^k{}_j - y_{hj} + y_{tj}$.

A pseudo flow is an arc flow vector x that does not necessarily satisfy $Ax = b$. It is said to be α-optimal when it satisfies the above conditions.

For a given pseudo flow x define the node flow excess vector $e = (e_i)$, with $e_i = b_i - \sum_j [x_j : t_j = i] + \sum_j [x_j : h_j = i]$. Observe that if $e = (e_i) = 0$ then the pseudo flow x is a primal feasible flow.

In the next section we described the four implemented algorithms, Primal Simplex, Dual Method, Out-of-Kilter and Cost-Scaling, with its main operational characteristics.

3. ALGORITHMS FOR NETWORK PIECEWISE-LINEAR PROGRAMS

Now let us introduce each one of the specialized developed algorithms for solving NPL Programs.

3.1 PRIMAL SIMPLEX ALGORITHM

For this study a network specialization of a well-known simplex method for PLP (see [1]) was implemented.

Denote:

$\hat{C}_s = c_s^+$ if $c_s^+ < 0$; $\hat{C}_s = c_s^-$ if $c_s^- > 0$, $\hat{C}_s = 0$ otherwise.

The basic steps of the algorithm may be written as below.

```
0. Let T, f = $(f_j)$ be, respectively, a spanning tree and an arc
   current interval vector associated with a basic primal
   feasible solution (x,y).

1. Select an entering nonbasic arc $s \notin T$ such that $\hat{C}_s \neq 0$.
   If there is no such an arc then STOP:CURRENT SOLUTI ONOPTIMAL.
   Otherwise set $s_0 = s$ and compute the cycle $Q = Q^+ \cup Q^-$ formed
   in $T \cup \{s_0\}$.

2. Perform the ratio test on the cycle Q
   If $\hat{C}_s < 0$ then

      compute $\delta_0 = d^{fs+1}{}_s - d^{fs}{}_s$, $\delta_1 = \min\{d^{fj+1}{}_s - X_j : j \in Q^+ \setminus \{s_0\}\}$,

      and $\delta_2 = \min\{x_j - d^{fj}{}_j : j \in Q^- \setminus \{s_0\}\}$.
           else
      compute $\delta_2 = d^{fs}{}_s - d^{fs-1}{}_s$, $\delta_1 = \min\{x_j - d^{fj}{}_j : j \in Q^+ \setminus \{s_0\}\}$,

      and $\delta_2 = \min\{d^{fj+1}{}_j - X_j : J \varepsilon Q^- \setminus \{s_0\}\}$.

   Let $\delta = \min\{\delta_0, \delta_1, \delta_2\}$.
   Select a leaving arc $r \in Q$ that minimizes $\delta$.
   If there is no such an arc then STOP:NO FINITE SOLUTION.
   Otherwise if $\hat{C}_s < 0$ set $f_s = f_s - 1$.
   Update the flow vector x (arcs in Q).
   Update the price vector y (in the subtree $T \cup \{s_0\} \setminus \{r\}$).
   If the new $\hat{C}_r \neq 0$ then set s = r and GO TO 2.
   If $r \neq s_0$ then update the spanning tree $T = T \cup \{r\} \setminus \{s_0\}$.
   GO TO 1.
```

In order to obtain an initial arc flow basic vector x, it is enough to consider any spanning tree and additional intervals with values $d = -\infty$, $c = -\infty$ for each basic arc with negative flow. Therefore, phase I of the Simplex Method does not need to be considered as a separated phase.

In NPL Programs we performed a Pivot operation each time that it is updated the flow vector x, and we performed an Iteration when it is chosen a nonbasic entering arc. The main characteristic of the last pivot of an iteration is that the leaving arc is not a candidate to enter in the basis (immediately).

Note that all pivoting associated to a same iteration have entering and leaving arcs that belong to the same cycle Q. In order to be efficient, the algorithm is implemented with just one update of the spanning tree and nodes price per iteration.

In fact, the implemented algorithm has the property of avoiding both Cycling (cyclic degenerate pivot sequence) and Stalling (exponentially long degenerate pivot sequence) phenomena.

This is possible by starting the algorithm with a Strongly Feasible Basic Flow vector x, with a modified ratio test (which breaks ties adequately) maintaining Strong Feasibility of all basis, and using a smart selection of the entering arc in each iteration. For details see reference [8], here we present the main related ideas.

A basic flow vector x is called strongly feasible if every degenerate basic arc $J \in T$ with $x_j = d^{f_j+1}{}_j$ is up in T, and every degenerate basic arc $J \in T$ with $x_j = d^{f_j}{}_j$ is down in T. This is an extension of the concept of strongly feasible spanning trees introduced by Cunningham ([9]) and Barr, Glover and Klingman ([10]) for Linear Networks.

An initial strongly feasible basic flow vector x, and an associated current interval arc vector f, can be obtained from any feasible spanning tree by increasing or decreasing by 1 (which is adequate) the current interval f_j associated each Degenerate basic arc j that do not satisfy the strong feasibility property.

The rule to maintain strong feasibility throughout the algorithm is as follows: Let s be the entering arc and let w be the last common node of the paths in T from the root to the end nodes of arc s, t_s and n_s. Consider the cycle Q formed in $T \cup \{s\}$. Traverse this cycle starting at node w with the same direction of s if $\hat{C}_s < 0$, or with the opposite direction of s if $\hat{C}_s > 0$. Choose as the leaving arc r the first one of the cycle that satisfies the ratio test.

With these two modifications the proposed algorithm avoid cycling. Furthermore, if is adopted a refinement in the selection of the entering arc stalling does not occur. There is several entering rules (see [8]) with this characteristics.

We have implemented the rule known by Least Recently Considered (LRC) Rule: Let 1..m be any fixed ordering of the arcs and suppose that in the last iteration arc j was the entering arc. So, in the next iteration should be select the arc s as the entering one the first arc of the sequence j+1, j+2,..., m, m+1,...,j which is a candidate ($\hat{C}_s \neq 0$).

3.2. DUAL ALGORITHM

The second optimizer code, utilized in this research, is based on the dual method on a graph proposed by Ali, Padman and Thiagarajan (see [11]), and its description is given below.

```
0. Let $T$, $f = (f_j)$ be a spanning tree and an arc current nterval
   vector associated with a basic dual feasible solution $x,y)$.

1. Select a leaving basic arc $r \in T$ such that:

               $X_r < d^{fr}_r$ or $x_r > d^{fr+1}_r$ .

   If there is no such an arc then STOP:CURRENT SOLUTION OPTIMAL.
   Otherwise compute the cut-set $K_r = K^+_r \cup K^-_r$ associated with r.

2. Perform the ratio test on the cut-set $K_r$.

   If $x_r > d^{fr+1}_r$

     then set $\delta_0 = c^{fr+1}_r - y_{hr} + y_{tr}$ and $f = r = f_r + 1$.

     else set $\delta_0 = c^{fr-1}_r - y_{hr} + y_{tr}$.

If ($x_r > d^{fr+1}_r$ and r is up) or ($x_r < d^{fr}_r$ and r down)
   then [case 1]
          compute $\delta_1 = \min \{ + (c^{fj}_j - y_{hj} + y_{tj}) : j \in K^+_r \}$,
                  $\delta_2 = \min \{ - (c^{fj-1}_j - y_{hj} + y_{tj}) : j \in K^-_r \}$.
else [case 2]
          compute $\delta_1 = \min \{ - (c^{fj-1}_j - y_{hj} + y_{tj}) : j \in K^{++}_r \}$,
                  $\delta_2 = \min \{ + (c^{fj}_j - y_{hj} + y_{tj}) : j \in K^-_r \}$.
Compute $\delta = \min \{ \delta_0 , \delta_1 , \delta_2 \}$.
Select an entering arc $s \in K_r$ that minimizes $\delta$.

If there is no such an arc then STOP:PRIMAL IS INFEASIBLE.
Otherwise if ($s \in K^-_r$ in case 1) or ($s \in K^+_r$ in case 2)
               then set $f_s = f_s - 1$.

Update the flow $x_j$ in the arcs j in the cycle $T \cup \{s\}$.
Update the nodes prices $y_1$ in the subtree $T \setminus \{r\}$.
If $r \neq s$ then update the spanning tree $T \cup \{s\} \setminus \{r\}$.
If ($x_s < d^{fs}_s$ or $x_s > d^{fs+1}_s$) then set $r = s$ and GO TO 2.
GO TO 1.
```

Observe that an initial basic dual feasible solution can be obtained by introducing an artificial root node 0 with demand $b_0 = 0$ and n artificial arcs from the root to every other node with a unique feasible interval [0,0] with cost 0.

The starting solution can be obtained by setting the price vector $y = 0$. The flow vector x and the current interval vector f are set to $x_j = d^{fj}_j$, in such a way that $c^{fj-1}_j \leq 0 \leq c^{fj}_j$ for all $j \in E$ and then the artificial arc flows are computed by solving the system of constraints by back substitution.

In order to reduce the computational effort, we force the leaving arc r to be the previous entering arc s whenever such an arc is candidate to leave the spanning tree. In this way the cut-set and the spanning tree do not have to be update for one or more iterations. We call a pivoting every time that the ratio test is executed and we call an iteration every time the cut-set is determined.

An efficient data structure to maintain the spanning tree and the cut-set is used, and consist of four node vectors, predecessor, thread, reverse thread and node partition indices, and one arc vector to store the cut-set.

3.3. OUT-OF-KILTER ALGORITHM

The next implemented algorithm is a specialized version for NPL Programs of the Out-of-Kilter method for Network Linear Programs presented in reference [12].

Before to exhibit the algorithm, let us to introduce useful specific concepts and notations to this approach.

From the general theory of Linear Programming is possible (see [2]) to write the Complementary Slackness Conditions (CSC) associated with the Primal and Dual NPL Programs as below, where x, y, z as defined before:

$$d^{fj}_j < x_j < d^{fj+1}_j \quad \Rightarrow \quad y_{hj} - y_{tj} = c^{fj}_j,$$
$$c^{fj-1}_j < y_{hj} - y_{tj} < c^{fj}_j \quad \Rightarrow \quad x_j = d^{fj}_j.$$
$$(\text{ and } z_j = \max \{c^1_j , y_{hj} - y_{tj} \}).$$

Given an arc $j \in E$, if the CSC are satisfied then j is said to be In Kilter (IK), otherwise j is Out-of-Kilter (OK). The Kilter Number (NK ≥ 0) associated withthe arc j is how much the flow x may be alter in order to put (or to remain, if arc j is IK yet) the arc j in IK status.

So for each solution (x,y,z) is possible to associate a Solution Kilter Number (NK), computed by NK = $\sum_j$ NK, $j \in E$.

Now, to find the current interval $f_j \in \{1..m_j\}$, where m_j is the total number of juxtaposed intervals related to arc j, in order to satisfy $c^{fj}_j \leq Z_j < c^{fj+1}_j$, for each arc $j \in E$. It is possible to demonstrate (see [4], [8], and [12]) that:

if $c^{fj}_j < Zj < c^{fj+1}_j$ then $NK_j = |x_j - d^{fj}_j|$,
if $z_j = c^{fj}_j = y_{hj} - y_{tj}$ then $NK_j = d^{fj}_j - x_j$ for $x_j < d^{fj}_j$,
$NKj = 0$ for $d^{fj}_j \leq x_j \leq d^{fj+1}_j$,
$NK_j = X_j - d^{fj+1}_j$ for $d^{fj+1}_j < x_j$.
if $z_j = c^1_j > y_{hj} - y_{tj}$ then $NK_j = x_j - d^1_j$.

Besides this, define c_j" by:

if $d^{fj}_j < x_j + d^{fj+1}_j$ then $c_j" = c^{fj}_j - y_{hj} + y_{tj}$.

if $x_j = d^k_j$ with $k \in \{1..m_j\}$ then $c_j" = c^k_j - z_j$ for $Z_j < c^k_j$,
$c_j" = 0$ for $c^k_j \leq z_j \leq c^{k+1}_j$,
$c_j" = c^{k+1}_j - z_j$ for $c^{k+1}_j < z_j$.

Now we are able to describe the algorithm's phases.

0. Let (x,y,z) be a solution to a NPL Program satisfying Ax = b, $x \geq 0$, $Z_j = \max \{c^1_j\ y_{hj} - y_{tj}$ and $c^{fj}_j \leq z_j \leq c^{fj+1}_j$ with $f_j \in \{1.\ .m_j\}$.

1. Compute NK associated with the current solution.

   If NK = 0 then STOP:CURRENT SOLUTION OPTIMAL.
   Otherwise let $s \in E$ be an arc with $NK_s > 0$.

   Let $T = \theta$ and $\Delta_1 = 0$, for each $i \in N$.
   Compute $c"_s$.
   If $c_s > 0$ then let $V = \{h_s$ and $\Delta_{hs} = NK_s$.
   else let $V = \{t_s\}$ and $\Delta_{ts} = NK_s$.

2. Let $\Psi_1 = \{r \neq s,\ c_r" < 0,\ t_r \notin V,\ h_r \in V$ and $x_r < d^{fr}_r\}$,
   $\Psi_2 = \{r \neq s,\ c_r" = 0,\ t_r \notin V,\ h_r \in V$ and $x_r < d^{fr}_r\}$,
   $\Psi_3 = \{r \neq s,\ c_r" > 0,\ t_r \in V,\ h_r \notin V$ and $x_r < d^{fr}_r\}$, and
   $\Psi_4 = \{r \neq s,\ c_r" = 0,\ t_r, \in, V,\ h_r \notin V$ and $x, > d^{fr-1}_r\}$.

If $\Psi_1 U \Psi_2 U\ \Psi_3 U \Psi_4 = \varnothing$ then GO TO 3.
Let $r \in \Psi_1 U \Psi_2 U\ \Psi_3 U \Psi_4$.
Case $r \in \Psi_1$:
   $\text{let}\,\Delta_{tr} = \min\{\Delta_{nr}, d^{fr}{}_r - X_r\}, V = V\,U\{t_r\}\,\text{and}\,T = T\,U\{r\}$.
Case $r \in \Psi_2$:
   $\text{let}\,\Delta_{tr} = \min\{\Delta_{nr}, d^{fr}{}_r - X_r\}, V = V\,U\{t_r\}\,\text{and}\,T = T\,U\{r\}$.

Case $r \in \Psi_3$:
   $\text{let}\,\Delta_{tr} = \min\{\Delta_{nr}, d^{fr}{}_r - X_r\}, V = V\,U\{h_r\}\,\text{and}\,T = T\,U\{r\}$.
Case $r \in \Psi_3$:
   $\text{let}\,\Delta_{tr} = \min\{\Delta_{nr}, d^{fr}{}_r - X_r\}, V = V\,U\{h_r\}\,\text{and}\,T = T\,U\{r\}$.
If $\{t_s, h_s\} \not\subset V$ then GO TO 2.

```
    Increase by min $\{\Delta_{hs},\Delta_{ts}\}$ the flow in the arcs of the cycle Q
    formed in the subgraph (V,T).

    GO TO 1.
3. Let $\psi_1 = \{r : h_r \in V, t_r \notin V \text{ and } c_r'' > 0\}$,
       $\psi_2 = \{r : h_r \notin V, t_r \in V \text{ and } c_r'' < 0\}$, and
       $\theta = \min \{|c_r''| : r \in \psi_1 \cup \psi_2\}$
    If $\theta = \infty$ then STOP:NO FINITE SOLUTION.
    Otherwise update $y_i = y_i + \theta$ and $z_j = \max \{c_j^1, y_{hj} - y_{tj}\}$, for
    all $i \in N$ and $j \in E$.

    GO TO 2.
```

Note that this algorithm can be initialized with any solution (x,y,z) that satisfies Ax = b, x ≥ 0, and for j ∈ E Zj = max $\{c_j^1, y_{hj} - y_{tj}\}$.

3.4. COST-SCALING ALGORITHM

In this section is presented a extension of the Minimum Cost Flow Circulation Algorithm proposed by Goldberg and Tarjan (see [7]) to solve NPL Programs with integer data. It is possible to prove that this specialized algorithm (see [5] and [6]) is a strongly polynomial one, and runs in 0(n.m'.log n.min(log nC , m' .log n)) time, where n is the number of nodes, m' is the total number of intervals, and C is the largest integer arc cost.

The implemented algorithm is described below, and it utilizes a cost scaling technique and a maximum flow routine to solve the subproblems generated by the scaling.

Denote: $d^-{}_j = d_j^{fj}$, $d^+{}_j = d^{fj+1}{}_j$, $c_j^- = c^{fj-1}{}_j - y_{hj} + y_{tj}$, and
$c^+{}_j = c^{fj+1}{}_j - y_{hj} + y_{tj}$.

```
0. Let x,y be the starting feasible flow and prices vector,
   respectively. Let α = max $\{c^k{}_h\}$, all k = 1..$m_j$,and j ∈ E.
1. Set $\alpha = 2^{[\log \alpha]}$.
   While α ≥ 1 do
     For j ∈ E do
       while $c'_j \geq +\alpha$ do $f_j = f_j - 1$, and $x_j = d^+{}_j$.
       while $c'_j \leq -\alpha$ do $f_j = f_j + 1$, and $x_j = d^-{}_j$.
     For (i ∈ E and $e_1 > 0$) do
       Repeat
     If exists j such that $(t_j = i \text{ and } x_j < d^+{}_j)$ then $x_j = d^+{}_j$.
```

```
        If exists j such that $(h_j = i \text{ and } x_j > d^-_j)$ then $x_j = d^-_j$.
        If exists j such that $(t_j = i, x_j = d_j \text{ and } c_j > -\alpha)$
           then $f_j = f_j - 1$.
        If exists j such that $(t_j = i, x_j = d^+_j \text{ and } c^+_j < +\alpha)$
           then $f_j = f_j + 1$.
        If none of the above then $y_i = y_i + \alpha$.
       Until $e_i = 0$.
     Set $\alpha = \alpha / 2$.
```

This algorithm can be written as a strongly polynomial one,see reference [6] for details.

The idea is to start with any 2 α-optimal feasible flow for some α equal to a power of two, then this flow is transformed into a pseudo flow that is α-optimal, by increasing or decreasing the flow through the arcs that do not satisfy the α-optimality conditions (see section 2).

Such an operation creates some node excess flows. With a sequence of pushing and relabelling operations that reduce the excess flow $|e_1|$ of each node, this α-optimal pseudo flow is transformed into an α-optimal feasible flow. The process is repeated until $\alpha < 1$.

If no feasible flow is known at start, it is possible to begin with x = 0 and y = 0, however a check for feasibility in the first iteration should be introduced in order to detect infeasible problems.

4. SOME COMMENTS AND PRELIMINARY COMPUTATIONAL RESULTS

In order to obtain initial exploratory data about the relative performance of the four presented algorithms, some experiments were run with PASCAL implementations of these algorithms on an IBM compatible 386 microcomputer.

These experiments were performed by solving randomly generated NPL Programs (details of this generator in [8]), with up to 1200 nodes, 32000 arcs, and 9 intervals per arc, and the CPU time (excluding Input/Output times) spent by each algorithm for solving these problems was observed.

In figures 2, 3 and 4 are presented the CPU times - mean of 10 randomly generated NPL problems - for the three best detected algorithms. It is possible to observed the influence of the number of nodes, number of arcs, and number of intervals per arc, respectively, in algorithms' performance, for the generated problems. The results of the Dual Algorithm are not presented because it was the worst algorithm every time.

It is interesting to note the good performance of the Out-of-Kilter relative to CPU times spent by the Simplex, practically these algorithms had identical behaviour, and both shown great superiority over the Dual and Cost-Scaling algorithms, for solving the generated NPL Programs.

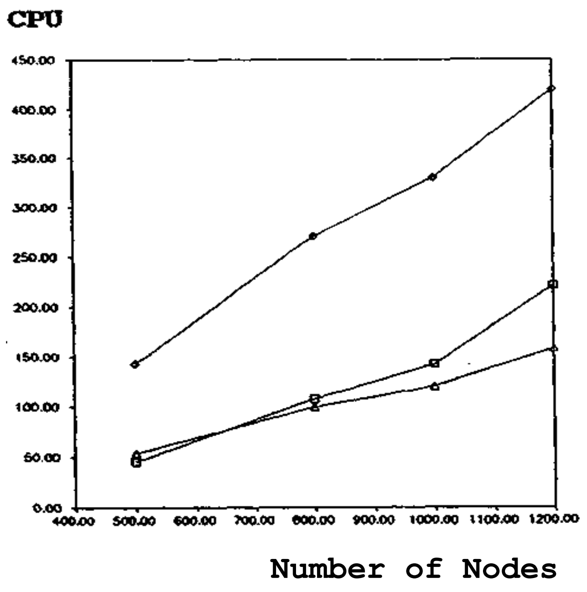
The main objective of this research is to develop a statistically designed experiments for studying the relative effectiveness of these four algorithms, and for identifying the effect on solution time when are changed (singly or in combinations) the two factors: problem class and problem size.

The work's idea is based on the paper published by Amini and Barr (see [13]), where the authors employed rigorous statistical procedure in order to compare the three best-known network reoptimization algorithms.

Therefore, at the moment, it is being devised a statistical experimental design to evaluate the relative efficiencies of the four algorithms in question, and a NPL optimization testing system to generate the data for the statistical proposed study.

Below are presented the preliminary computational results observed for the Primal, the Out-of-Kilter, and the Cost-Scaling algorithms to solve the randomly generated NPL problems (transsshipment networks). As mentioned above the cpu times related to the Dual algorithm's performance were excluded, because the values were very higher than that ones related to the others three algorithms. In the next figures the CPU times are given in seconds, and the algorithm identification is made using the symbols:

□ ↔ Primal, ◊ ↔ Cost-Scaling, and Δ ↔ Out-of-Kilter.



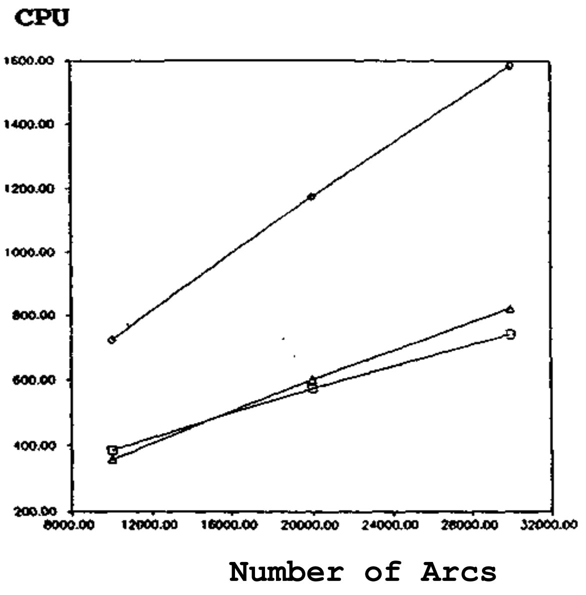
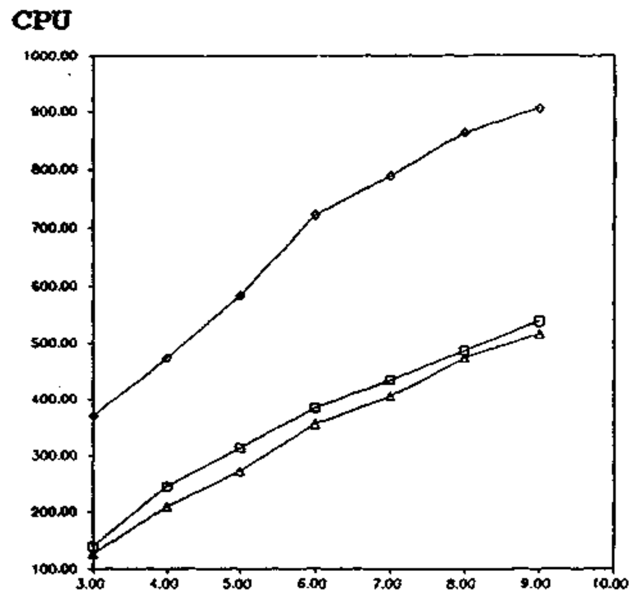


Figure 2.CPU as a Function of the number of Nodes

Figure 2.CPU as a Function of the number of Arcs



Number of Intervals per Arc

Figure 4. CPU as a function of the Number of Intervals per Arc

5. REFERENCES

[1] Fourer, R. (1985). A Simplex Method for Piecewise-Linear Programming I: Derivation and Proof. Mathematical Programming 13, 1-13.

[2] Perin, C. and Marins, F.A.S. (1988). Strong Feasibility in Network Piecewise-Linear Programs. In: Proceedings of the IV Latin-Iberian-American Congress on Operations Research and System Engineering,Rio de Janeiro-RJ, Brazil.

[3] Perin, C. and Marins, F.A.S. (1991). Computational Experience with Dual Simplex Algorithm for Network Piecewise-Linear Programs. In: Proceedings of the XIII World Congress on Computation and Applied Mathematics, Dublin, Ireland.

[4] Perin, C. and Marins, F.A.S. (1989). An Out-of-Kilter Algorithm for Network Piecewise-Linear Programs. In: Proceedings of the XII National Congress on Computational and Applied Mathematics, Sao Jose do Rio Preto-SP, Brazil.

[5] Machado, A., Perin, C. and Marins, F.A.S. (1992). A Strongly Polynomial Algorithm for Network Piecewise-Linear Programs. In: Proceedings of the XXIV Brazilian Symposium of Operations Research, Salvador-BA, Brazil.

[6] Machado, A., Perin, C. and Marins, F.A.S. (1993). An O(n.m'.log n.min(log nC , m'logn)) Algorithm for Network Piecewise-Linear Programs. In: Proceedings of the XXV Brazilian Symposium of Operations Research, Campinas-SP, Brazil.

[7] Goldberg, A.V. and Tarjan, R. (1990). Finding Minimum - Cost Circulations by Successive Approximation. Mathematics of Operations Research 15:3 August, 430-466.

[8] Marins, F.A.S. (1987). Studies of Network Piecewise-Linear Programs. Library of the IMECC-UNICAMP, Campinas-SP, Brazil (PhD Dissertation, Electrical Engineering College, State University of Campinas, Campinas-SP-Brazil).

[9] Cunningham, W.H. (1976). A Network Simplex Method. Mathematical Programming 11, 105-116.

[10] Barr, R.S., Glover, F. and Klingman, D. (1977). The Alternating Basis Algorithm for Assignment Problems. Mathematical Programming 13, 1-13.

[11] Ali, A. , Padman, R. and Thiagarajan, H. (1989). Dual Algorithm for Pure Network Problems. Operations Research 37:1, 159-171.

[12] Kennington, J.L. and Helgason, R.V. (1980). Algorithms for Network Programming. John Wiley & Sons, New York.

[13] Amini, M.M. and Barr, R.S. (1993). Network Reoptimization Algorithms: A Statistically Designed Comparison. ORSA Journal on Computing 5:4.

Acknowledgements: This work was partially supported by FAPESP-Fundacão de Amparo à Pesquisa do Estado de São Paulo (Brazil) and CNPq - Conselho Nacional de Desenvolvimento Cientifico e Tecnol6gico (Brazil).