Journal of Global Optimization **17:** 225–234, 2000.

A Dynamic Domain Contraction Algorithm for Nonconvex Piecewise Linear Network Flow Problems*

DUKWON KIM and PANOS M. PARDALOS
Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville

(Received April 2000; Accepted June 2000)

Abstract. We consider the Nonconvex Piecewise Linear Network Flow Problem (NPLNFP) which is known to be $\mathcal{NP}$-hard. Although exact methods such as branch and bound have been developed to solve the NPLNFP, their computational requirements increase exponentially with the size of the problem. Hence, an efficient heuristic approach is in need to solve large scale problems appearing in many practical applications including transportation, production-inventory management, supply chain, facility expansion and location decision, and logistics. In this paper, we present a new approach for solving the general NPLNFP in a continuous formulation by adapting a dynamic domain contraction. A Dynamic Domain Contraction (DDC) algorithm is presented and preliminary computational results on a wide range of test problems are reported. The results show that the proposed algorithm generates solutions within 0 to 0.94 % of optimality in all instances that the exact solutions are available from a branch and bound method.

Key words: Domain contraction; Dynamic slope scaling procedure; Fixed charge; Nonconvex piecewise linear network flow problem

1. Introduction

The Nonconvex Piecewise Linear Network Flow Problem (NPLNFP) is one of the most difficult problems in combinatorial network optimization [11]. In many practical applications dealing with a network structure such as transportation, logistics, supply chain management involving a production-inventory system, and facility location, we usually deal with various cost functions having *economies of scale* and/or *nonconvex piecewise linear* aspects. In most cases, these problems can be formulated as a 0-1 Mixed Integer Programming (MIP) problem which is solved by a branch and bound type method [2].

Although branch and bound algorithms can find exact solutions for small size problems, we cannot avoid exponential time and memory requirements as the problem size increases [10]. Thus, we are in need of developing efficient heuristic methods to produce 'good' quality sub-optimal solutions for practical size prob-

* This paper is dedicated to the memory of Professor P.D. Panagiotopoulos

lems. This can be done by considering the special cost structure and solution behavior of the problem.

In particular, the problem we focus on here is to minimize a sum of discontinuous piecewise linear arc costs with a set of network constraints. This is an extension of the previous study for solving continuous concave Piecewise Linear Network Flow Problems (PLNFP) by reducing them into a fixed charge network model [8]. However, different from the concave case, there is no guarantee of extreme point optimality because of the absence of concavity [3, 5]. Due to the above reason, vertex enumerating type techniques [12] may fail to find exact solutions. As a result, the solution can be attained at the boundary or interior of the feasible domain. This motivates the initial idea of adapting a domain contraction concept [13].

The paper is organized as follows. The following section contains the problem descriptions and formulations, and the domain contraction scheme in the proposed algorithm is discussed in Section 3. The numerical results are shown in Section 4, and Section 5 concludes the paper.

2. The Problem and Formulations

The Nonconvex Piecewise Linear Network Flow Problem (NPLNFP) can be stated as follows:

Given a directed graph $G = (N, A)$ consisting of a set N of m nodes and a set A of n arcs, then solve

[NPLNFP]

$$\min \quad f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij})$$

subject to

$$\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = b_i, \quad \forall\, i \in N \tag{2.1}$$

$$0 \leqslant x_{ij} \leqslant u_{ij}, \quad \forall\, (i, j) \in A \tag{2.2}$$

where f is separable and each f_{ij} is discontinuous piecewise linear. In general, the arc cost $f_{ij}(x_{ij})$ can be defined as follows:

$$f_{ij}(x_{ij}) = \begin{cases} c_{ij}^1 x_{ij} + s_{ij}^1, & 0 \leqslant x_{ij} < \lambda_{ij}^1 \\ c_{ij}^2 x_{ij} + s_{ij}^2, & \lambda_{ij}^1 \leqslant x_{ij} < \lambda_{ij}^2 \\ \vdots & \vdots \\ c_{ij}^{r_{ij}} x_{ij} + s_{ij}^{rij}, & \lambda_{ij}^{r_{ij}-1} \leqslant x_{ij} \leqslant \lambda_{ij}^{r_{ij}} \end{cases}$$

where λ_{ij}^r for $r = 1, 2, \ldots, r_{ij} - 1$ are breakpoints and r_{ij} denotes the number of segments in the given interval $[0, u_{ij}]$. The problem is uncapacitated if $u_{ij} =$

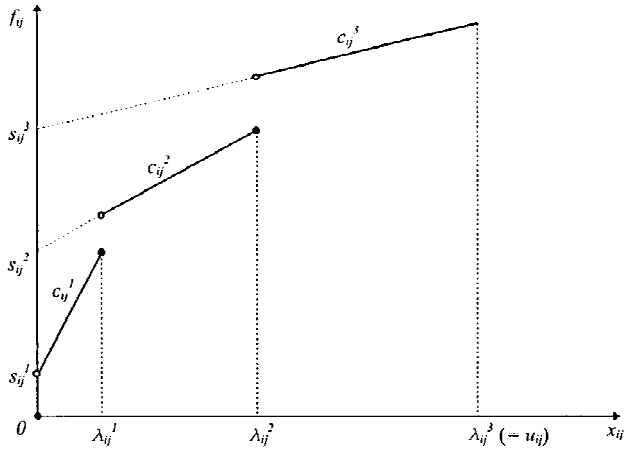

Figure 1. An example of staircase arc cost functions.

∞, $\forall\,(i, j) \in A$. Different from concave piecewise linear cases, it is assumed that each piecewise linear arc cost function can be discontinuous at any breakpoint. Thus, any general concave minimization technique can not be directly implemented to solve this problem.

Nonconvex piecewise linear arc costs can be generalized in two types, the so-called *staircase* [1, 4, 9] and *sawtooth* [9] arc cost functions, respectively. Both arc cost functions have a very similar structure in overall shape, however, they have a different aspect at breakpoints (see Figures 1 and 2). It can be described in mathematical form as follows:

– 'Staircase' arc cost function:

$$f_{ij}^{r-1}(\lambda_{ij}^{r-1}) < f_{ij}^{r}(\lambda_{ij}^{r-1} + \varepsilon), \quad \text{for any } \varepsilon > 0 \text{ and } r = 2, 3, \ldots, r_{ij},$$

and

– 'Sawtooth' arc cost function:

$$f_{ij}^{r-1}(\lambda_{ij}^{r-1} - \varepsilon) > f_{ij}^{r}(\lambda_{ij}^{r-1}), \quad \text{for any } \varepsilon > 0 \text{ and } r = 2, 3, \ldots, r_{ij},$$

where f_{ij}^{k} is an arc cost for the k-th segment in each arc cost function, f_{ij}. Moreover, the slopes in each arc cost function, $c_{ij}^{r} \geqslant 0$, hold a decreasing property, due to the practical reason such as *price discounts*.

In a previous study [8], this problem was reformulated as a 0-1 Mixed Integer Programming (MIP) problem in the fixed charge structure as follows:

Let us define the size of the interval between adjacent breakpoints as

$$\Delta\lambda_{ij}^{r} = \lambda_{ij}^{r} - \lambda_{ij}^{r-1}, \quad \forall\,(i, j) \in A, \quad r = 1, 2, \ldots, r_{ij}, \tag{2.3}$$

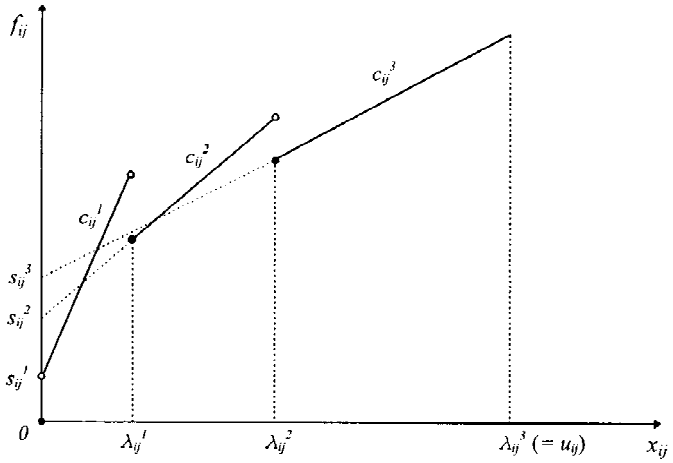

Figure 2. An example of sawtooth arc cost functions.

and define the gap of function values at each breakpoint in the arc cost functions as

$$\begin{aligned} \Delta d_{ij}^r &= (c_{ij}^r \lambda_{ij}^{r-1} + s_{ij}^r) - (c_{ij}^{r-1} \lambda_{ij}^{r-1} + s_{ij}^{r-1}) \\ &= s_{ij}^r - s_{ij}^{r-1} + (c_{ij}^r - c_{ij}^{r-1}) \lambda_{ij}^{r-1}, \quad \forall (i, j) \in A, \ \forall r, \end{aligned} \tag{2.4}$$

where $s_{ij}^0 = 0$ and $c_{ij}^0 = 0$ (also clearly $\Delta d_{ij}^1 = s_{ij}^1$). We now let x_{ij}^r be the part of x_{ij} that lies within level r (i.e. r-th segment), in the following sense:

$$x_{ij}^r = \begin{cases} 0 & \text{if } x_{ij} \leqslant \lambda_{ij}^{r-1} \\ x_{ij} - \lambda_{ij}^{r-1} & \text{if } \lambda_{ij}^{r-1} < x_{ij} \leqslant \lambda_{ij}^r \\ \Delta \lambda_{ij}^r & \text{if } x_{ij} \geqslant \lambda_{ij}^r, \end{cases} \tag{2.5}$$

and introduce new binary variables defined by

$$y_{ij}^r = \begin{cases} 1 & \text{if } \lambda_{ij}^{r-1} < x_{ij} \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

Then, we have

$[\text{NPLNFP}]_{\text{MIP}}$

$$\min \quad \sum_{(i,j) \in A} \sum_{r=1}^{r_{ij}} (c_{ij}^r x_{ij}^r + \Delta d_{ij}^r y_{ij}^r)$$

subject to constraints in (1) and

$$x_{ij} = \textstyle\sum_{r=1}^{r_{ij}} x_{ij}^r, \quad \forall (i, j) \tag{2.7}$$

$$x_{ij}^r \leqslant \Delta \lambda_{ij}^r y_{ij}^r, \quad \forall (i, j), \ r = 1, \ldots, r_{ij} \tag{2.8}$$

$$x_{ij}^{r-1} \geqslant \Delta \lambda_{ij}^{r-1} y_{ij}^r, \ \forall (i, j), \ r = 2, \ldots, r_{ij} \tag{2.9}$$

$$x_{ij}^r \geqslant 0, \quad \forall (i, j), \ r = 1, \ldots, r_{ij} \tag{2.10}$$

$$y_{ij}^r \in \{0, 1\}, \quad \forall (i, j), \ r = 1, \ldots, r_{ij}. \tag{2.11}$$

It is noticed that combining one constraint from (8) and one from (9) yields $\Delta\lambda_{ij}^{r-1} y_{ij}^{r} \leqslant x_{ij}^{r-1} \leqslant \Delta\lambda_{ij}^{r-1} y_{ij}^{r-1}$, which implies $y_{ij}^{r} \leqslant y_{ij}^{r-1}$, $\forall\ (i, j), r > 1$.

There is another approach to formulate the problem as a concave Minimum Cost Network Flow Problem model. In [9], Lamar described an equivalent formulation of MCNFP with general nonlinear arc costs (including the problem considered in this section) as a concave MCNFP on an extended network. The equivalence between the problems is based on converting each arc with an arbitrary cost function in the original problem into an arc with a concave piecewise linear cost function in series with a set of parallel arcs, each with a linear arc cost function (see [9] for details). Thus, the resulting problem is a concave MCNFP, which is different from the FCNFP formulation model shown above.

However, since both approaches are limited by problem size, we propose a new Linear Programming (LP)-based heuristic approach by employing the *Dynamic Slope Scaling Procedure* [6], and *Domain Contraction* technique [13].

3. Dynamic Domain Contraction Algorithm

Next we consider successive LP problems which linearize the original version of NPLNFP described in Section 2 with a linearized objective function as follows. For each iteration k, we solve the following LP problem to find a solution, $\bar{x}^k$:

$[\text{NPLNFP}]_{\text{LP}}^{\text{k}}$

$$\min \quad \bar{f}^k(x) = \sum_{(i,j)\in A} \bar{c}_{ij}^{k} x_{ij} \tag{3.12}$$

subject to the same constraints in (1) and (2), where the dynamic slope scaling factor, $\bar{c}_{ij}^{k}$ is obtained and updated by the following scheme:

- For the initial iteration $k = 0$,

$$\bar{c}_{ij}^{k} = c_{ij}^{r_{ij}} + \frac{s_{ij}^{r_{ij}}}{u_{ij}}\ .$$

- For $k = 1, 2, 3, \ldots$, if the previous solution $\bar{x}_{ij}^{k-1}$ occurs in the r-th segment, then

$$\bar{c}_{ij}^{k} = c_{ij}^{r} + \frac{s_{ij}^{r}}{\bar{x}_{ij}^{k-1}}, \quad \text{for } \bar{x}_{ij}^{k-1} > 0.$$

The detail updating schemes when $\bar{x}_{ij}^{k-1} = 0$ can be found in [6].

Based on the Dynamic Slope Scaling Procedure (DSSP) briefly shown above, we develop a contraction rule to reduce the given feasible domain (polyhedron) for this problem. For each iteration k, the feasible domain to be considered can be

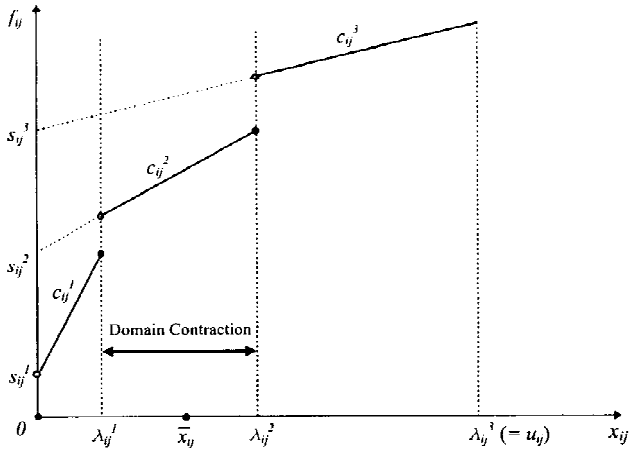

Figure 3. An example of Domain Contraction.

reduced depending on which segments contain the current LP solution from the DSSP. If the current LP solution for arc (i, j), x_{ij}^k occurs in the r-th segment, then we can solve this problem as a fixed charge problem with an arc cost function as:

$$f_{ij}(x_{ij}) = c_{ij}^r x_{ij} + s_{ij}^r, \quad \text{for } (i, j) \in A.$$

However, due to the absence of concavity in the arc cost functions, we cannot directly decompose and reduce this problem into a simple fixed charge problem [7]. This implies that a set of constraints to specify which segment of the arc cost function to be considered must be added to the original set of constraints.

Let us define the set of additional constraints by *contraction constraints* as follows:
For iteration k, if $\bar{x}_{ij}^{k-1}$ occurs in the r-th segment, then

$$\lambda_{ij}^{r-1} \leqslant x_{ij} < \lambda_{ij}^r, \quad \text{for } (i, j) \in A.$$

In addition, we define a *contraction* for iteration k by

$$(\bar{l}_{ij}^k, \bar{u}_{ij}^k) \quad \text{for } (i, j) \in A.$$

where $\bar{l}_{ij}^k = \lambda_{ij}^{r-1}$ and $\bar{u}_{ij}^k = \lambda_{ij}^r$ if $\bar{x}_{ij}^{k-1} \in (\lambda_{ij}^{r-1}, \lambda_{ij}^r]$. Clearly, these contraction constraints replace the original redundant capacity constraints, $0 \leqslant x_{ij} \leqslant u_{ij}$ at each iteration.

As a result, we solve the following recursive LP problems incorporating the Dynamic Domain Contraction (DDC) scheme described above.

$[\text{NPLNFP}]_{\text{LP}}^{\text{k}} - \text{DDC}$

$$\min \quad \bar{f}^k(x) = (\bar{c}^k)^T x$$

subject to

$$x \in \mathcal{D}_k = \mathcal{S} \cap \mathcal{C}_k$$

where

$$\mathcal{D}_k \subset \mathbf{R}^n, \text{ where } n = |A|, \tag{3.13}$$

$$\mathcal{S} = \{ x \mid Ax = b, \ A \text{ is a node-arc incidence matrix} \} \text{ and} \tag{3.14}$$

$$\mathcal{C}_k = \{ x \mid \bar{l}^k \leqslant x \leqslant \bar{u}^k \}. \tag{3.15}$$

In this model, we solve LP problems over the reduced domain $\mathcal{D}_k$ which is dynamically updated based on the previous LP solution $\bar{x}^{k-1}$. A series of reduced domains $\mathcal{D}_k$ may produce different types of solutions (i.e. interior point solution, boundary (not vertex) point solution, or extreme point solution) for the original problem [NPLNFP], according to the relation between $\mathcal{S}$ and $\mathcal{C}_k$.

It is clear that there are two types of reduced domains $\mathcal{D}_k$ which characterize the obtained solution in this domain contraction procedure. The first case is that if $\mathcal{C}_k \subset \mathcal{S}$ which implies that $\mathcal{D}_k = \mathcal{C}_k$, then we obtain an *interior point solution* for the original nonconvex problem, although the LP solution occurs at a vertex of $\mathcal{D}_k$. Notice that the reduced feasible domain $\mathcal{D}_k$ is a n-dimensional polytope generated by n contraction constraints. The second case is that if $\mathcal{C}_k \not\subset \mathcal{S}$, so that the reduced domain $\mathcal{D}_k$ contains some boundary points and/or extreme points of $\mathcal{S}$, then we may have boundary or extreme point solution for the original problem.

As a result, this dynamic domain contraction algorithm produces a series of reduced feasible domains $\mathcal{D}_k$ to search the sub-domain which contains the solution, and the algorithm terminates when there is no more improvement in successive LP solutions.

4. Numerical Experiments

Due to the difficulty of finding a set of known test problems, we generate five groups of pseudo-random problems, whose numbers of arcs are ranged from 35 to 335. Each group has two sets of problems, and each set contains 20 problems. The first set and second set in each group consist of problems with the fixed number of linear segments $r_{ij} = 3$ and $r_{ij} = 5$, respectively. Note that the actual size of problems (the number of binary variables) in the 0-1 mixed integer programming formulation, $[\text{NPLNFP}]_{\text{MIP}}$ is given by $|A|r_{ij}$ when r_{ij} is a constant for all arcs. This estimation is based on an extended network [9] from applying the Arc Separation Procedure (ASP) for the original network. Thus, more precisely, the total number of arcs in the extended network structure when each arc cost has a different r_{ij} is given by

$$n_e = \sum_{(i,j) \in A} r_{ij} \tag{4.16}$$

where n_e denotes the number of arcs, $|A_e|$ in the extended network, $G_e = (A_e, N)$. This means that the actual number of binary variables in the $[\text{NPLNFP}]_{\text{MIP}}$ formulation ranges from 105 up to 1675 in our test problems.

The algorithm has two parts, the domain contraction subroutine (DDC) to generate $\mathcal{D}_k$ and the dynamic slope scaling subroutine (DSSP). The algorithm is implemented in standard C with CPLEX 4.0 (callable library) and executed on a SUN workstation running UNIX. In addition, we apply the CPLEX branch and bound algorithm to the test problems for the exact solution, in order to assess the effectiveness and efficiency of the proposed DDC algorithm.

It is observed that the branch and bound algorithm has difficulty to obtain the exact solutions for the problems having more that 80 arcs with 5 segments (i.e. $n_e = 80 \times 5 = 400$), due to the memory limitation.

For each set of 10 problems, we report the minimum, average and maximum relative errors, and the computational time(sec). The relative errors are computed as follows:

$$\text{RE } (\%) = \left[\frac{f_{\text{DDC}} - f_{\text{exact}}}{f_{\text{exact}}} \right] \times 100,$$

and the computational time is measured by the total CPU time excluding I/O operations. For example, the average relative error in problem set #4 is 0.628%, the average CPU time is 0.073(sec), and the maximum relative error is 0.82% from implementing the proposed DDC algorithm on 20 instances. For problem set #5 to #10, the average relative errors are not available since the branch and bound algorithm fails to find exact solutions for most instances.

In order to see the efficiency of the proposed algorithm, we may compare the solution times between DDC and B&B algorithms. The average CPU times from implementing the DDC algorithm are ranging from 0.014 (sec) up to 0.475 (sec) over the test problems with up to $n_e = 400$, while the average CPU times from the branch and bound algorithm are ranging from 6.091 (sec) to 965.355 (sec). The maximum relative error and the maximum CPU time occur in Problem set # 4 as of 0.94% and 1.02 (sec), respectively, while the maximum CPU time of the branch and bound is 1799.32 (sec) for the same size of test problems.

The computational results are summarized in Table 1. The results show that the proposed solution approach can produce 'high' quality near-optimal solutions efficiently for such a difficult class of nonconvex piecewise linear network flow problems described in Section 2, and the performance of the algorithm is stable regarding time and quality.

5. Concluding Remarks

Different from the traditional solution methods including branch and bound and cutting planes, the proposed DDC algorithm with DSSP consists of simply solving recursively updated LP problems (DSSP) and dynamically reducing the feasible

Table I. The summary of computational results for 200 problems

Problem		Size				DSSP		B & B
Group	set no.	m	n	r_{ij}	n_e	RE[†](%) (min, max)	CPU (sec) (min, max)	CPU (sec) (min, max)
G1	1	12	35	3	105	0.102 (0.00, 0.35)	0.014 (0.01, 0.03)	6.091 (1.21, 9.13)
	2	12	35	5	175	0.391 (0.02, 0.74)	0.026 (0.01, 0.04)	43.745 (8.90, 67.25)
G2	3	18	80	3	240	0.628 (0.14, 0.82)	0.073 (0.03, 0.13)	191.547 (24.74, 560.24)
	4	18	80	5	400	0.707 (0.16, 0.94)	0.475 (0.09, 1.02)	965.355 (65.46, 1799.32)
G3	5	27	175	3	525	n/a[‡]	0.533 (0.16, 1.29)	n/a
	6	27	175	5	875	n/a	0.959 (0.47, 1.93)	n/a
G4	7	32	225	3	675	n/a	0.692 (0.29, 1.54)	n/a
	8	32	225	5	1125	n/a	1.304 (0.57, 3.80)	n/a
G5	9	37	335	3	1005	n/a	1.198 (0.59, 3.21)	n/a
	10	37	335	5	1675	n/a	1.892 (0.77, 4.90)	n/a

[†]The average Relative Error.
[‡]Exact solutions are not available from B&B due to the memory limitation.

domain (DDC). Thus, the implementation of the DDC algorithm is fairly simple. However, the results show that the algorithm generates high quality solutions (with less than 1% error) efficiently. In addition, due to its simple and flexible structure, the proposed method is robust.

For the future work, the quality of solutions obtained by the DCC algorithm can be improved by incorporating a local search procedure.

References

1. Bornstein, C.T. and Rust, R. (1988), Minimizing a sum of staircase functions under linear constraints, *Optimization* 19: pp. 181–190.
2. Guisewite, G.M. and Pardalos, P.M. (1990), Minimum concave-cost network flow problems: Applications, complexity, and algorithms, *Annals of Operations Research* 25: 75–100.
3. Guisewite, G.M. and Pardalos, P.M. (1991), Global search algorithms for minimum concave cost network flow problems, *Journal of Global Optimization* 1: 309–330.

4. Holmberg, K. (1994), Solving the staircase cost facility location problem with decomposition and piecewise linearization, *European Journal of Operational Research* 75: 41–61.
5. Horst, R. and Pardalos, P.M. (*eds*.), (1995), *Handbook of Global Optimization*, Kluwer Academic Publishers.
6. Kim, D. and Pardalos, P.M. (2000), A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure, *Operations Research Letters* 24(4): 195–203.
7. Kim, D. and Pardalos, P.M. (2000), Piecewise linear network flow problems, in: A. Migdalas, P.M. Pardalos and P. Varbrand (eds.), *Local to Global Optimization*, Kluwer Academic Publishers.
8. Kim, D. and Pardalos, P.M. (2000), Dynamic slope scaling and trust interval techniques for concave piecewise linear network flow problems, *Networks* 35: 216–222.
9. Lamar, B.W. (1993), A method for solving network flow problems with general nonlinear arc costs, in: D.-Z. Du, and P. M. Pardalos, (eds.), *Network Optimization Problems*, World Scientific Publishing Co., 147–167.
10. Nemhauser, G.L. and Wolsey, L.A. (1988), *Integer and Combinatorial Optimization*, John Wiley and Sons Inc., New York.
11. Pardalos, P.M. and Rosen, J.B. (1987), *Constrained Global Optimization: Algorithms and Applications*, Lecture Notes in Computer Science 268, Springer-Verlag, Berlin.
12. Pardalos, P.M. (1988), Enumerative techniques for solving some nonconvex global optimization problems, *OR Spectrum* 10: 29–35.
13. Thakur, L.S. (1991), Domain contraction in nonlinear programming: Minimizing a quadratic concave objective over a polyhedron, *Mathematics of Operations Research* 16(2) 390–407.