Journal of Global Optimization (2006) 34: 97–125
DOI 10.1007/s10898-005-1657-y

Lower Bounds from State Space Relaxations for Concave Cost Network Flow Problems

DALILA B. M. M. FONTES[1], ELENI HADJICONSTANTINOU[2] and NICOS CHRISTOFIDES[2]
[1]*LIACC, Faculdade de Economia da Universidade do Porto Rua Dr. Roberto Frias, 4200-464 Porto, Portugal (e-mail: fontes@fep.up.pt)*
[2]*Tanaka Business School, Imperial College London South Kensington Campus, London SW7 2AZ, UK (e-mail{e.hconstantinou n.christofides}@imperial.ac.uk)*

(Received 26 January 2005; accepted 27 July 2005)

Abstract. In this paper we obtain Lower Bounds (LBs) to concave cost network flow problems. The LBs are derived from state space relaxations of a dynamic programming formulation, which involve the use of non-injective mapping functions guaranteing a reduction on the cardinality of the state space. The general state space relaxation procedure is extended to address problems involving transitions that go across several stages, as is the case of network flow problems. Applications for these LBs include: estimation of the quality of heuristic solutions; local search methods that use information of the LB solution structure to find initial solutions to restart the search (Fontes et al., 2003, Networks, 41, 221–228); and branch-and-bound (BB) methods having as a bounding procedure a modified version of the LB algorithm developed here, (see Fontes et al., 2005a). These LBs are iteratively improved by penalizing, in a Lagrangian fashion, customers not exactly satisfied or by performing state space modifications. Both the penalties and the state space are updated by using the subgradient method. Additional constraints are developed to improve further the LBs by reducing the searchable space. The computational results provided show that very good bounds can be obtained for concave cost network flow problems, particularly for fixed-charge problems.

Key words: concave cost network flows, dynamic programming, lower bounds, state space relaxation

1. Introduction

The minimum concave cost Network Flow Problem (NFP) is known to be NP-*hard* (Guisewite and Pardalos, 1991a). Its complexity arises from minimizing a concave function over a convex feasible region, defined by the network constraints, which implies that a local optimum is not necessarily a global optimum. The main features defining the complexity of NFPs are the type of cost function for each arc, the number of nonlinear arc costs, and the ratio between variable cost and fixed cost. A discussion of other parameters affecting the complexity of NFPs can be found in Fontes (2000). In this work we concentrate on Single Source Uncapacitated (SSU) concave cost NFPs. This problem class can be used to model more general

NFPs since general nonlinear arc costs can be transformed into concave arc costs on an expanded network (Lamar, 1993) and multiple source and capacitated arcs can be transformed into single source and uncapacitated arcs (Zangwill, 1968). Although the expanded network is generally a much larger network, the growth is polynomial.

Most of the work developed on concave cost NFPs considers problems with Fixed-Charge (FC) cost functions that consist of a fixed cost and a linear variable cost. The latter problem is a particular case of the more general concave cost NFP, although NP-*hard* itself. An extensive survey on concave cost NFPs is given by Guisewite (1994).

LBs have been derived for NFPs with FC costs, e.g. by using Lagrangian relaxation (Hochbaum and Segev, 1989); by linear programming relaxation (Kim and Hooker, 2002; Ortega and Wolsey, 2003). Hochbaum and Segev (1989) developed two Lagrangian relaxations: one by relaxing the constraints involving both the flow and the binary variables and another by relaxing the flow conservation constraints. In the former relaxation, they obtain a problem involving two sets of unrelated variables, which is then separated into two subproblems: a linear NFP and a minimum branching problem. In the second relaxation, the resulting problem can be reduced to a minimum branching problem. The solutions to the relaxed problems are LBs which are feasible to the original problem and thus, upper bounds can be obtained by recomputing their cost using the original FC costs. Kim and Hooker (2002) propose a hybrid Branch-and-Bound (BB) method, which combines constraint programming with linear programming techniques. At each node of the search tree, constraint programming is used to reduce the domain of a discrete variable and thus, the number of branches, while a linear programming relaxation provides a bound on the optimal value of the problem. Ortega and Wolsey (2003) developed a branch-and-cut algorithm that uses simple dicut inequalities and their variants. Bounds are obtained by linear underestimation based on the dynamic slope scaling (Kim and Pardalos, 1999), which recursively updates the objective function to solve a sequence of linear problems.

Burkard et al. (2001) developed a Dynamic Programming (DP) formulation for SSU concave cost NFPs in acyclic networks. As the DP functions are implicitly defined and hard to compute, the authors use successive linear underestimations instead. The solutions to these approximations, which are feasible to the original problem, provide LBs to the optimal value. Uppers bounds can be found by recomputing their cost using the original cost functions. To improve the LBs, the linear approximations are updated by making use of information about the LBs. Guisewite and Pardalos (1991b) give a BB algorithm based on that of Gallo et al. (1980). In the original work (Gallo et al., 1980) the authors enumerate extreme feasible solutions by adding arcs that extend the current subtree. Bounds for

each subproblem were obtained by a linear relaxation of the subproblem that uses linear underestimation of the arc costs for unsatisfied flows. In Guisewite and Pardalos (1991b) the bounding is improved by using linear underestimation functions to project a lower bound on the cost of extending the current path. Recently, Horst and Thoai (1998) developed a BB algorithm where the bounding is performed by linear NFPs on subnetworks. The linear underestimation functions are obtained for each arc with nonlinear arc costs by convex envelops.

In this paper, we develop LBs derived from State Space Relaxations (SSRs) of a DP formulation given in Fontes et al. (2005b. in press). The general SSR procedure, due to Christofides et al. (1981), is extended to address problems involving multistage transitions, i.e. transitions that go across several stages, which is the case for NFPs. LBs are obtained by two known relaxations, namely the cardinality relaxation and the q-set relaxation, that have been adapted to NFPs. Furthermore we propose a new relaxation, the "combined relaxation" that aims at combining the benefits of the two previously mentioned relations. The LBs are sequentially improved by penalizing, in a Lagrangian fashion, customers not exactly satisfied, by performing state space modifications, or both. The subgradient method (Held et al., 1974) is used both to update the penalties and to perform state space modifications. We also propose additional constraints to improve further the LBs by reducing the searchable space. This has been accomplished after analyzing the structure of a feasible solution to the original problem and the structure of a feasible solution to the relaxed problem. Computational experiments are reported for SSU NFPs with cost functions having (i) linear variable costs and fixed costs and (ii) concave variable costs and fixed costs.

To the best of our knowledge problems having both a concave variable cost and a fixed cost have not been addressed before, except for Burkard et al. (2001), Fontes et al. (2003, 2005b). The LBs developed can be used in different ways, including to measure the quality of heuristic solutions, to provide information on the solution structure (Fontes et al., 2003) and to be imbedded into a BB procedure (Fontes et al., 2005a).

2. Problem Definition and Formulation

The network $G = (W, A)$ to be optimized consists of a set W of $n + 1$ vertices (vertices $1, \ldots, n$ denote demand vertices and vertex $n + 1$ denotes the source vertex t) and a set A of m directed arcs, $A \subset \{(i, j) : i \in W, j \in W \setminus \{t\}\}$. Each demand vertex has associated a non-negative integer demand r_i. The supply at the source vertex R matches the total demand required by the n demand vertices. A solution structure is

characterized by the flow r_{ij} on each arc $(i, j) \in A$. A general nondecreasing, nonnegative, and concave cost function g_{ij} is associated with each arc (i, j) and satisfies $g_{ij}(0) = 0$. The objective is to find a subset of arcs and arc flows that satisfy the demand at minimum cost.

SSU concave cost NFPs have a finite optimal solution if and only if there exists a direct path going from the source to every demand vertex and if there are no negative cost cycles. For this problem, a feasible flow is an extreme flow if it contains no positive cycles. For the uncapacitated case a positive cycle is a cycle with all arcs (i, j) satisfying $r_{ij} > 0$. Therefore, for the SSU case, an extreme flow is a tree rooted at the single source spanning all demand vertices (Zangwill, 1968). The objective in solving this class of problems is to find a minimum cost directed tree network that satisfies all customers demand.

In Fontes et al. (2005b) we have developed a DP formulation for the SSU concave cost NFP, that is independent of the type and form of cost functions and also of the number of nonlinear arc costs. In addition, no assumption other than separability and additivity is required.

Consider a set $S \subseteq W$ and a vertex $x \in S$. Let $\{S', \bar{S}'\}$ be one partition of set S, where $S' \subseteq S \setminus \{x\}$ and $\bar{S}' = S \setminus S'$. For each possible set S', let $z \in S'$ be the root vertex of a directed tree spanning the set S'. Let $f(S', z)$ be the minimum cost of supplying all demand vertices in S' with the required commodity available at vertex z through a directed tree rooted at z. The minimum cost of supplying a set S' from vertex $x \notin S'$ with the required commodity made available at some vertex $z \in S'$ is found by determining the best combination of the minimum cost directed tree of S' rooted at vertex $z \in S'$ with the cost of arc (x, z), that is

$$\min_{z \in S'} \left\{ f\left(S', z\right) + g_{xz}\left(\sum_{i \in S'} r_i\right) \right\}.$$

By definition, the minimum cost incurred in supplying the remaining demand vertices of set S not in S' from x is given by $f\left(\bar{S}', x\right)$. Figure 1 shows a possible partition of set S, a possible directed tree in S' rooted at vertex z, and a flow pattern of supplying $\bar{S}'$ from vertex x.

From the above, the minimum cost $f(S, x)$ of supplying all demand vertices in S, with the commodity available at $x \in S$, is obtained by examining all possible subsets $S' \subseteq S \setminus \{x\}$ and is given by

$$f(S, x) = \min_{S' \subseteq S \setminus \{x\}} \left[f(S - S', x) + \min_{z \in S'} \left[f(S', z) + g_{xz}\Big(\sum_{i \in S'} r_i\Big) \right] \right]. \tag{1}$$

Initial conditions for recursion (1) are provided by $f(\{x\}, x) = 0, \forall x \in W$. Hence, the optimal cost of supplying all demand vertices in set W from the

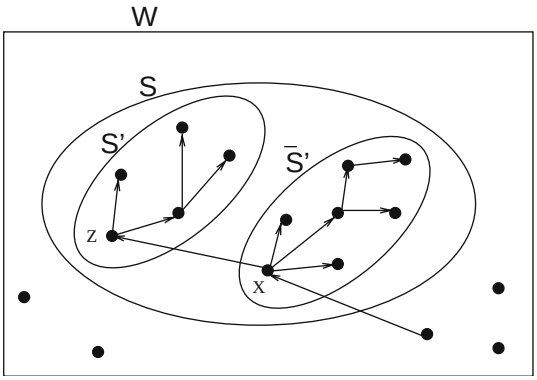

Figure 1. A flow pattern of supplying set S with the commodity available at vertex x.

source vertex t, is given by

$$f^* \equiv f(W,t) = \min_{S' \subseteq W \setminus \{t\}} \left[f\left(W - S', t\right) + \min_{z \in S'} \left[f(S', z) + g_{tz}\Big(\sum_{i \in S'} r_i \Big) \right] \right]. \quad (2)$$

3. State Space Relaxation

Due to the large dimensionality of the state space, few combinatorial problems of large dimension can be solved efficiently by DP alone. This is due to the number of vertices of the state space graph being very large even for small size problems. In our model there are $(n+2)2^{n-1}$ states in the DP formulation and $2^{2n-2}\left(n^2+4n+8\right) - 2^{n-1}\left(3n^2+4n+8\right) + n + 2$ transitions between states. (Details of the state space graph analysis can be found in Fontes et al., 2005b.) Thus, it is quite natural to consider reducing the size of the state space graph. State Space Relaxation (SSR) is a general relaxation procedure in which the state space associated with the DP recursion is relaxed (i.e. the number of states reduced) in such a way that the solution to the relaxed recursion provides a bound to the value of the true optimum. This approach was first proposed by Christofides et al. (1981) and was applied successfully to the travelling salesman problem, to a variety of vehicle routing problems (Christofides et al., 1981; Hadjiconstantinou et al., 1995) and to other combinatorial optimization problems such as cutting problems (Christofides and Hadjiconstantinou, 1995) and set covering problems (Christofides and Paixão, 1993). In Section 3.2 we extend the general SSR procedure to problems where transitions can occur across several stages, as is the case for the concave cost NFP.

SSR in dynamic programming is analogous to Lagrangean relaxation in integer programming. Constraints in an integer programming formulation appear as state variables in a DP recursion and hence constraint relaxation

corresponds to state space relaxation. Moreover, SSR can be thought of as a generalization of the Lagrangean relaxation in integer programming in the sense that it is possible in SSR to use nonlinear mapping functions.

3.1. STATE SPACE RELAXATION FOR ROUTING PROBLEMS

Here, we describe the general SSR procedure for the computation of bounds to problems for which transitions occur only between consecutive stages, (for details see Christofides et al., 1981).

Consider a multistage discrete system where s represents the state space variable. For each stage i let s_i be a generic state taking a value from $\{s_i^1, \ldots, s_i^{m_i}\}$. Defining $f_{0,i}(s_0, s_i)$ as the least cost of changing the system from a state s_0 at stage 0 to a state s_i at stage i, the general forward DP recursion is given by Equation (3), where

1. $\Delta(s_{i-1})$ is the transition set, i.e. set of all possible states s_i (at stage i) that can result from state s_{i-1} (at stage $i-1$),
2. $\Delta^{-1}(s_i) = \{s_{i-1} | \Delta(s_{i-1}) \ni s_i\}$ is the set of all possible states s_{i-1} (at stage $i-1$) that lead to state s_i (at stage i), and
3. $v_i(s_{i-1}, s_i)$ is the cost of moving from state s_{i-1} (at stage $i-1$) to state s_i (at stage i).

$$f_{0,i}(s_0, s_i) = \underset{s_{i-1} \in \Delta^{-1}(s)}{\text{OPT}} [f_{0,i-1}(s_0, s_{i-1}) + v_i(s_{i-1}, s_i)]. \tag{3}$$

The usual computational problem that one encounters when directly applying recursion (3) to a combinatorial problem is the large number of the states $s_i \in \{s^i, \ldots, s^{m_i}\}$ for each stage $i \in \{1, 2, \ldots, n+1\}$. This is particularly relevant in problems where the state variable s involves all subsets of a given set.

The SSR of recursion (3) is obtained by defining a non-injective mapping function $h(\cdot)$ from the state space W^s associated with Equation (3) to a state space H^s. The image of W^s has smaller cardinality, (see Figure 2).

Define $F^{-1}(h(s_i))$ as a set satisfying the condition:

$$\text{if } s_{i-1} \in \Delta^{-1}(s_i) \quad \text{then } h(s_{i-1}) \in F^{-1}(h(s_i)). \tag{4}$$

Let the minimum cost of changing from the "relaxed state" $h(s_0)$ to the "relaxed state" $h(s_i)$ be denoted by $\bar{f}_{0,i}(h(s_0), h(s_i))$. The relaxation of Equation (3) then becomes:

$$\bar{f}_{0,i}(h(s_0), h(s_i)) = \underset{t \in F^{-1}(h(s_i))}{\text{OPT}} [\bar{f}_{0,i-1}(h(s_0), t) + \bar{v}_i(t, h(s_i))], \tag{5}$$

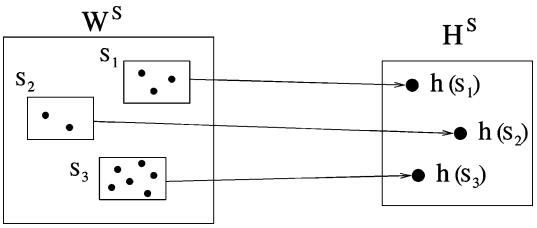

Figure 2. Graphical representation of the SSR by a non-injective mapping function.

where

$$\bar{v}_i\,(t, h\,(s_i)) = \mathrm{OPT}[v_i\,(p, s^*) \mid h\,(p) = t,\ h\,(s_i) = h\,(s^*)]. \tag{6}$$

We then have $\bar{f}_{0,i}\,(h\,(s_0)\,, h\,(s_i)) \leqslant f_{0,i}\,(s_0, s_i)$ for all i, s_0, and s_i.

The choice of $h\,(\cdot)$ plays a major role and it must satisfy the following conditions:

(a) The function $h\,(\cdot)$ is such that the set $F^{-1}\,(\cdot)$ can be computed easily from Equation (4);
(b) The function $h\,(\cdot)$ is such that the optimization in Equation (6) is over a small domain or a good bound on $\bar{v}_i\,(t, h\,(s_i))$ can be obtained.

3.2. STATE SPACE RELAXATION FOR MULTISTAGE TRANSITION PROBLEMS

In this section, we extend the procedure stated in Section 3.1 to handle multistage transitions, i.e. transitions that go across several stages. Recall that s_i is a generic state of stage i and let $f_{1,i}\,(s_1, s_i)$ be the least cost of changing the system from a state s_1 at stage 1 to a state s_i at stage i. The general forward DP recursion is given by Equation (7) where

1. $\Delta\,(s_i)$ is the transition set, i.e. set of all possible states s_j (at stage j, $j \geqslant i+1$) that can result from state s_i (at stage i),
2. $\Delta^{-1}\,(s_j) = \{s_i \mid \Delta\,(s_i) \ni s_j\}$ is the set of all possible states s_i (at stage i) that lead to state s_j (at stage j, $j \geqslant i+1$), and
3. $v_{i,k}\,(s_i, s_k)$ is the cost of state s_i when computed by using state partitions s_{i-k} and s_k.

$$f_{1,i}\,(s_1, s_i) = \underset{\substack{s_k \in \Delta^{-1}(s_i) \\ s_{i-k} \in \Delta^{-1}(s_i)}}{\mathrm{OPT}} [f_{1,i-k}\,(s_1, s_{i-k}) + f_{1,k}\,(s_1, s_k)\, v_{i,k}\,(s_i, s_k)]. \tag{7}$$

As said previously, the direct application of recursion (7) to a combinatorial problem faces the problem of the large dimension of the state variable $s_i \in \{s_i^1, \ldots, s_1^{m_i}\}$ for each stage $i \in \{1, \ldots, n+1\}$. For the NFP

examined here, this is even more relevant than for routing problems, since the possible number of states is even larger. Furthermore, the computation of f involves the computation of two, rather than one, components involving f of smaller stages.

Redefine the set F such that for any $j \geqslant i+1$ the following condition is satisfied:

$$\text{if } s_i \in \Delta^{-1}\left(s_j\right) \quad \text{then } h\left(s_i\right) \in F^{-1}\left(h\left(s_j\right)\right). \tag{8}$$

The relaxation of Equation (7) then becomes:

$$\begin{aligned} &\bar{f}_{1,i}\left(h\left(s_1\right), h\left(s_i\right)\right) \\ &\quad = \mathrm{OPT}_{\substack{a \in F^{-1}(h(s_i)) \\ b \in F^{-1}(h(s_i))}} [\bar{f}_{1,i-k}\left(h\left(s_1\right), a\right) + \bar{f}_{1,k}\left(h\left(s_1\right), b\right) + \bar{v}_{i,k}\left(h\left(s_i\right), b\right)], \end{aligned} \tag{9}$$

where

$$\bar{v}_{i,k}\left(h\left(s_i\right), b\right) = \mathrm{OPT}[v_{i,k}\left(p, s^*\right) \mid h\left(p\right) = b,\ h\left(s_i\right) = h\left(s^*\right)]. \tag{10}$$

And thus, a bound (LB for a minimization problem) can be obtained by solving Equation (10).

3.3. STATE SPACE RELAXATION FOR THE CONCAVE COST NFP

We propose relaxations where the state space associated with the DP recursion is relaxed in such a way that the state space dimension is reduced. The solution to the relaxed recursion provides a LB to the optimal solution value, which can be improved either by using penalties in a Lagrangean fashion or by using state space modifications.

Consider the DP formulation of the SSU concave cost NFP given by Equation (1). In the original state space, a state is represented by a pair (S, x). The vertex x, acts as a source to supply the set of vertices S, with x in S. The stage is given by the cardinality of the set S. Let h be a non-injective mapping function from the domain of states (S, x) to the domain of relaxed states $(h\left(S\right), x)$ which in general has smaller cardinality. We then have:

$$\Delta^{-1}\left(S, x\right) = \left\{\left(S', z\right) \mid z \in S',\ S' \subseteq S \setminus \{x\}\right\}, \tag{11}$$

$$F^{-1}\left(h\left(S\right), x\right) = \left\{\left(h\left(S'\right), z\right) \mid z \in S',\ h\left(S'\right) < h\left(S \setminus \{x\}\right)\right\}. \tag{12}$$

The DP recursion, Equation (1) can now be rewritten in the new relaxed space as follows:

$$\bar{f}(h(S), x) = \min_{h(S')<h(S\setminus\{x\})}\Bigg[\bar{f}\left(h\left(S\setminus S'\right), x\right) + \min_{(h(S'),z)\in F^{-1}(h(S),x)}\Bigg[\bar{f}\left(h\left(S'\right), z\right) + g_{xz}\Big(\sum_{i\in S'} r_i\Big)\Bigg]\Bigg]. \tag{13}$$

From the above it is clear that $\bar{f}(h(S), x) \leqslant f(S, x)$ for all $S \subseteq W$ and for all $x \in S$. Therefore, Equation (13) can produce bounds which can be embedded into a BB to solve the original problem to optimality.

The choice of h plays a major role in the SSR. To make the recursion useful it should be chosen to be a separable function, so that given $h(S)$ and x, $h(S\setminus\{x\})$ can be easily computed, the relaxed state space has smaller cardinality, and the relaxed states are "easier" to compute.

3.4. FORMS OF THE MAPPING FUNCTION

The mapping function h can take different forms, in fact it can be any separable function. Several forms of the mapping function were introduced in Christofides et al. (1981). Here, we discuss the three forms of mapping function defining the three relaxations used in this work.

1. The cardinality relaxation (SSR1) – $h(S) = \left(|S|, \sum_{i\in S} r_i\right)$,
2. The q-set relaxation (SSR2) – $h(S) = \left(\sum_{i\in S} q_i, \sum_{i\in S} r_i\right)$,
3. The combined relaxation (SSR3) – $h(S) = \left(|S|, \sum_{i\in S} q_i, \sum_{i\in S} r_i\right)$, where r_i is the demand of vertex $i \in W$ and q_i is a non-negative integer weight associated with vertex $i \in W$. (For the source vertex t $r_t = 0$ and $q_t = 0$.)

SSR1 – The cardinality relaxation

Let $p = |S|$ represent the cardinality of set S, r_i the demand of vertex $i \in W$, and r the total demand of vertices in set S.

$$\text{Defining } h^1(S) = (p, r) = \left(|S|, \sum_{i\in S} r_i\right), \text{ we have } h^1(S\setminus\{x\}) = (p-1, r-r_x).$$

The original state (S, x) is mapped into (p, r, x) and the application of recursion (1) gives[1]

[1]From now on, we use $f(\cdot)$ instead of $\bar{f}(\cdot)$ to denote the least cost function in the relaxed state space.

$$f^1(p,r,x) = \min_{\substack{p' \leqslant p-1 \\ r' \leqslant r-r_x}} \left[f^1\left(p-p', r-r', x\right) + \min_{\substack{z \neq x \\ p' \geqslant 1 \\ r' \geqslant r_z}} \left[f^1\left(p', r', z\right) + g_{xz}\left(r'\right) \right] \right], \tag{14}$$

which is initialized by

$$f^1(p,r,x) = \begin{cases} 0 & \text{if } p=1 \quad \text{and } r=r_x, \\ +\infty & \text{otherwise.} \end{cases} \tag{15}$$

SSR2 – The q-set relaxation

In this relaxation each vertex $i \in W$ is represented by two integer numbers: r_i the demand of vertex $i \in W$ and q_i a weight we assign to each vertex $i \in W$.

Defining $h^2(S) = (q,r) = \left(\sum_{i \in S} q_i, \sum_{i \in S} r_i \right)$,

we have $h^2(S \setminus \{x\}) = (q - q_x, r - r_x)$.

The original state (S,x) is mapped into state (q,r,x) and recursion (1) becomes

$$f^2(q,r,x) = \min_{\substack{q' \leqslant q-q_x \\ r' \leqslant r-r_x}} \left[f^2\left(q-q', r-r', x\right) + \min_{\substack{z \neq x \\ q' \geqslant q_z \\ r' \geqslant r_z}} \left[f^2\left(q', r', z\right) + g_{xz}\left(r'\right) \right] \right], \tag{16}$$

which is initialized by

$$f^2(q,r,x) = \begin{cases} 0 & \text{if } q=q_x \quad \text{and } r=r_x, \\ +\infty & \text{otherwise.} \end{cases} \tag{17}$$

SSR3 – The combined relaxation

We propose a new relaxation that we have called "combined relaxation", which combines features and benefits of the previous two. In general, it is expected that SSR2 performs better than SSR1, since SSR1 can be seen as a particular case of SSR2 when $q_i = 1$ for all $i \in W$. Also, SSR2 allows for a more uniformly mapped state space and thus it can, potentially, give better bounds. For our specific problem, it is worth noticing that SSR1 allows to find LBs that preserve more of the structure of the solution to the original problem. As for SSR1 exactly n arcs are used, even if not all arcs used

are different. In contrast, for SSR2 any positive number of arcs is allowed to be in a solution as long as the total weigh $Q = \sum_{i \in W} q_i$ and the total demand $R = \sum_{i \in W} r_i$ are satisfied. A new relaxation SSR3 combining the ideas of both SSR1 and SSR2 was developed. In this relaxation, the original state (S, x) is mapped into (p, q, r, x), where p, r, and q are as above.

Defining $h^3(S) = (p, q, r) = \left(|S|, \sum_{i \in S} q_i, \sum_{i \in S} r_i \right)$,

we have $h^3(S \setminus \{x\}) = (p-1, q-q_x, r-r_x)$.

Thus, the recursion becomes,

$$f^3(p,q,r,x) = \min_{\substack{p' \leqslant p-1 \\ q' \leqslant q-q_x \\ r' \leqslant r-r_x}} \left[f^3(p-p', q-q', r-r', x) + \min_{\substack{z \neq x \\ p' \geqslant 1 \\ q' \geqslant q_z \\ r' \geqslant r_z}} \left[f^3(p', q', r', z) + g_{xz}(r') \right] \right] \tag{18}$$

and is initialized by

$$f^3(p,q,r,x) = \begin{cases} 0 & \text{if } p=1,\ q=q_x, \text{ and } r=r_x, \\ +\infty & \text{otherwise.} \end{cases} \tag{19}$$

The recursions apply for $p = 1, \ldots, n+1$, $q = 0, \ldots, Q$, $r = 0, \ldots, R$, and $x \in W$, accordingly. Simple bounds can be obtained by solving these recursions:

$$Z^1_{LB} = f^1(n+1, R, t), \; Z^2_{LB} = f^2(Q, R, t), \text{ and } Z^3_{LB} = f^3(n+1, Q, R, t).$$

A LB represents the least cost of supplying the total amount of commodity R available at the source vertex t to a *group*[2] of customers (not necessarily all distinct). In addition, in the case of Z^1_{LB} this *group* must have n elements taken from the customers set V, while in the case of Z^2_{LB} the *group* must have a total weight given by Q. For Z^3_{LB} the *group* must contain n customers, with a total weight Q and whose requirements add up to R. The partial knowledge of the structure of the solution given by this third relaxation enables us to impose some additional constraints that improve the quality of the bound, (see Section 4.2).

It can easily be argued that the quality of the bound given by SSR3 has to be better than the bound provided by any of the other two relaxations. This can be illustrated in terms of the mathematical abstraction we are trying to solve. Consider the following conditions:

[2]Here and hereafter, we use the word *group* to denote a collection of vertices that may include some vertices more than once. The elements in the *group are* not necessarily all different: actually if they are all different then an optimal solution to the original problem has been found.

(i) $\sum_{i \in V} \alpha_i = n$,
(ii) $\sum_{i \in V} \alpha_i r_i = R$, and
(iii) $\sum_{i \in V} \alpha_i q_i = Q$,

where $\alpha_i \in \mathbb{N}_0$ for all $i \in W$, $R = \sum_{i \in V} r_i$, and $Q = \sum_{i \in V} q_i$ ($\alpha_t = 1$, $r_t = 0$, and $q_t = 0$).

In each relaxation we are looking for an integer linear combination of

- SSR1 – pairs of numbers of the form $(1, r_i)$ satisfying conditions (i) and (ii),
- SSR2 – pairs of numbers of the form (q_i, r_i) satisfying conditions (ii) and (iii),
- SSR3 – triplets of numbers of the form $(1, q_i, r_i)$ satisfying conditions (i)–(iii).

The fact that there are less combinations satisfying the conditions stated for SSR3 than for SSR1 and SSR2 brings about two major benefits to this relaxation. On the one hand, the search for a solution among a reduced set of combinations (although the whole state space is larger) should provide a better LB. On the other hand, the number of local optima is also expected to be reduced, increasing the chance of obtaining a better LB or even the global optimum.

For further details on these relaxations, on the structure and size of the state space graph, on the procedures implementing them, and the computational results obtained (see Fontes, 2000).

4. The Lower Bounds

The LBs obtained by solving the relaxed recursions are sequentially improved by a State Space Ascent (SSA) procedure which considers Lagrangean penalties for SSR1, state space modifications for SSR2, and a combination of both for SSR3.

Further improvements are achieved by restricting the searchable space. The space reduction is accomplished by using additional constraints derived after analyzing the structure of feasible solutions to the original problem and to the relaxed problem.

4.1. ITERATIVE COMPUTATION OF THE LBS

The LB solution is a graph containing a path (or paths) from the source vertex to demand vertices and its value corresponds to $f^1(P, R, t)$, $f^2(Q, R, t)$, or $f^3(P, Q, R, t)$ depending on the relaxation being used. The graph structure may represent a tree or may have loops and is obtained by backtracking, using the information stored during the computation of states. Although all demand is distributed, it is not ensured that all vertices

get the required amount. Therefore, some customers may not have been supplied while others may have been oversupplied.

SSR1 – The cardinality relaxation

In SSR1, forcing a partial solution "closer" to feasibility is achieved by penalizing vertices i not exactly satisfied using a penalty λ_i. A new LB can be obtained by resolving recursion (14) with updated cost functions $g'_{ij}(r) = g_{ij}(r) + \lambda_j$ and it is given as

$$B(\lambda) = \min_{\substack{p' \leqslant P-1 \\ r' \leqslant R-r_t}} \left[f^1\left(P-p', R-r', t\right) + \min_{\substack{z \neq t \\ 1 \leqslant p' \\ r_z \leqslant r'}} \left[f^1\left(p', r', z\right) + g'_{tz}\left(r'\right) \right] \right] - \sum_{i \in V} \lambda_i. \tag{20}$$

We then wish to choose λ^* to be the maximizer of B. Several heuristic ways to initialize the penalties λ_i (cost and demand related, random, and the zero vector) were tried. The analysis of the results obtained allowed us to conclude that not much correlation exists between the initial values of the penalties and either the quality of the optimality gap or the iterations/time needed to find it. Thus, we decided to use $\lambda^0 = 0$ in the computational experiments. The subgradient optimization method (Held et al., 1974) starts with $\lambda = \lambda^0 = 0$ and at the kth iteration λ^k is updated as in Equation (21), where ψ_i^k denotes the net flow supplied to customer i at iteration k, $\lceil \mu \rceil$ the smallest integer greater than or equal to μ, and t^k the *step size*, which is given by Equation (22).

$$\lambda_i^k = \max\left\{\lambda_i^0, \lambda_i^{k-1} + \left\lceil t^k \cdot \left(\psi_i^k - r_i\right)\right\rceil\right\} \quad \forall i \in V, \tag{21}$$

$$t^k = \alpha \cdot \frac{Z_{LB}^{1k}}{\sum_{i \in V}\left(\psi_i^k - r_i\right)^2} \cdot \frac{Z_{UB} - Z_{LB}^{1k}}{Z_{UB}}. \tag{22}$$

In the computation of t^k, α is positive, Z_{UB} is the best solution found so far, and Z_{LB}^{1k} the LB at iteration k. The upper bound is obtained using the local search procedure described in Fontes et al. (2003). It can be shown that the method converges if $\sum_{k=1}^{\infty} t^k = \infty$ and $\lim_{k \to \infty} t^k = 0$. These conditions are satisfied if one starts with $\alpha > 0$ and periodically reduces α by some factor (Held et al., 1974). Note that Equations (21) and (22) are not the standard Lagrangean equations as the penalties can not be negative and a reduction factor is applied to t^k to prevent large changes.

SSR2 – The q-set relaxation

In SSR2, state space modifications are used. If the maximum value for the total weight is set to $\widehat{Q}$, we have to maximize B in the following equation subject to $Q \leqslant \widehat{Q}$.

$$B(q) = \min_{\substack{q' \leqslant Q - q_t \\ r' \leqslant R - r_t}} \left[f^2(Q - q', R - r', t) + \min_{\substack{z \neq t \\ q_z \leqslant q' \\ r_z \leqslant r'}} \left[f^2(q', r', z) + g_{tz}(r') \right] \right]. \quad (23)$$

The maximization of B is, in general, difficult since it is a discontinuous function of q. However, for some problems simple heuristic rules have worked quite well (Christofides and Hadijconstantinou, 1995, Christofides and Paixão, 1993). From the initializations tried we concluded that the quality of the bounds is roughly independent of the initial values of the weights. Thus, we set $q_i^0 = 0$, $i \in W$, which correspond to the smallest state space. At a very early stage of the computational experiments, we have verified that subgradient optimization gave better results. Let Z_{LB}^{2k} be the LB obtained at iteration k. The weights are updated as in Equation (24) and the *step size* t^k computed as in Equation (25).

$$q_i^k = \max \left\{ q_i^0, q_i^{k-1} + \left\lceil t^k \cdot \frac{\psi_i^k - r_i}{\max_j \{r_j\}} \right\rceil \right\}, \quad i \in V. \quad (24)$$

$$t^k = \alpha \cdot \frac{n^2}{\sum_{i=1}^{n} \left(\frac{\psi_i^k - r_i}{r_i} \right)^2} \cdot \frac{Z_{UB} - Z_{LB}^{2k}}{Z_{UB}}. \quad (25)$$

SSR3 – The combined relaxation

In SSR3, we are free to choose between using penalties in a Lagrangian fashion or state space modifications, or both. The LB is obtained by maximizing B subject to $Q \leqslant \hat{Q}$.

$$B(\lambda, q) = \min_{\substack{p' \leqslant P-1 \\ q' \leqslant Q - q_t \\ r' \leqslant R - r_t}} \left[f^3(P - p', Q - q', R - r', t) + \min_{\substack{z \neq t \\ 1 \leqslant p' \\ q_z \leqslant q' \\ r_z \leqslant r'}} \left[f^3(p', q', r', z) + g'_{tz}(r') \right] \right] - \sum_{i \in V} \lambda_i. \quad (26)$$

There is no foreseeable advantage in changing both λ_i and q_i simultaneously as their individual benefits could be cancelled out. After some experimentation we decided upon the use of a three-phase procedure to improve the LB, as follows.

1. In the first phase, the penalties and weights are initialized as $\lambda_i^0 = 0$ and $q_i^0 = 0$, respectively. The penalties are updated as given for SSR1, while the weights remain unchanged. Phase I is performed for a pre-specified number of iterations.
2. The second phase picks-up from the best LB found during phase I. The weights and the penalties are initialized at the values corresponding to the best LB. State space modifications, as given for SSR2, are applied over a pre-specified time period, while the penalties remain unchanged.
3. In the third phase, the penalties and weights are initialized at the values corresponding to the best LB found so far. Then it is improved by updating the penalties as in phase I, for a pre-specified number of iterations.

During phase I iterations are performed quickly and the time taken is almost constant for each iteration. After some iterations have been made, typically, no further improvement is achieved (a local optima may have been found). The procedure moves to phase II with the λ-vector corresponding to the best Z_{LB}. As the weights q_i's are modified, both the value of the total weight Q and the time per iteration increase. Thus, only a relatively small number of iterations can be performed before the computational time per iteration becomes prohibitively large. In the third phase, the penalties are updated again in an attempt to improve the LB further.

4.2. IMPROVING THE LBS BY ADDITIONAL CONSTRAINTS

The introduction of the mapping function brought the advantage of reducing the dimensionality of the state space. It also brought loss of knowledge in the sense that we no longer have information on the vertices of a set but only on their image. Thus, a feasible solution to the original problem may not be obtained.Certain specific restrictions can be imposed without increasing the dimensionality of the state space. The potential advantages are: (i) reduction of the space to search and thus, of the computational time; (ii) elimination of some infeasible states to the original problem; and (iii) elimination of potential local optima.

We studied the use of additional constraints to eliminate loops, to prevent the usage of repeated arcs, and to reduce the space to search. The first two types have not proven to be advantageous as the increase of the state space would make its computation too time consuming. The latter type originated three sets of constrains: constraints enforcing the use of sets, constraints to supply only reachable vertices, and constraints using the cardinality of the partition.

Constraints enforcing the use of sets

Let (S, x) be the state being solved and (S', z) the partition under consideration. It is obvious that set S' cannot contain vertex x and that set $S \setminus S'$ cannot contain vertex z. Furthermore, neither vertex x nor vertex z can be in set $S' \setminus \{z\}$ or in set $S \setminus \big(S' \cup \{x\}\big)$. Nevertheless, in the relaxed state space, these conditions are not necessarily forbidden to occur. To impose these conditions, for SSR1 we use a three-dimensional binary matrix $\sigma_1(p, r, x)$, initially set to zero. Then, for each subset S in V and each vertex x in $W \setminus S$ with $p = |S|$ and $r = \sum_{i \in S} r_i$, we set $\sigma_1(p, r, x) = 1$. Thus, when computing state (p, r, x) we only consider partitions satisfying

$$\begin{aligned}
&\sigma_1(p', r', x) = 1,\\
&\sigma_1(p' - 1, r' - r_z, z) = 1,\\
&\sigma_1(p' - 1, r' - r_z, x) = 1,\\
&\sigma_1(p - p', r - r', z) = 1,\\
&\sigma_1(p - p' - 1, r - r' - r_x, x) = 1,\\
&\sigma_1(p - p' - 1, r - r' - r_x, z) = 1.
\end{aligned} \tag{27}$$

For SSR2 a three-dimensional matrix $\sigma_2(q, r, x)$, is defined and initialized in a similar way. Then, for each subset S in V and each vertex x in $W \setminus S$ with $q = \sum_{i \in S} q_i$ and $r = \sum_{i \in S} r_i$ we set $\sigma_2(q, r, x) = 1$. A state (q, r, x) must therefore, satisfy conditions (28).

$$\begin{aligned}
&\sigma_2(q', r', x) = 1,\\
&\sigma_2(q' - 1, r' - r_z, z) = 1,\\
&\sigma_2(q' - 1, r' - r_z, x) = 1,\\
&\sigma_2(q - q', r - r', z) = 1,\\
&\sigma_2(q - q' - 1, r - r' - r_x, x) = 1,\\
&\sigma_2(q - q' - 1, r - r' - r_x, z) = 1.
\end{aligned} \tag{28}$$

Although these constraints do not prevent arc repetitions or loops, they do reduce the likelihood of them happening. It should be noticed that in the definition of these matrices sets, rather than groups, are used. And also that matrix σ_1 is computed only once, whereas matrix σ_2 needs to be computed at every iteration during phase II due to weight modifications. For SSR3 these conditions are strengthened as both conditions (27) and (28) must be satisfied.

Constraints to supply only reachable vertices

Further reduction of the searchable space may be achieved by tightening the possible values of the flow that can be routed on each arc. An obvious lower limit is given by the demand of its end vertex, i.e. if arc (x, z) is used then the flow on this arc must satisfy $r_{xz} \geqslant r_z$. An upper limit may also be found. Let V_z be the set of vertices reachable from vertex z using at most n arcs if vertex z is the

source vertex and $n-1$ arcs otherwise. The maximum amount of flow that can be routed through arc (x, z) is then given by $r_z + \sum_{i \in V_z} r_i$. Similar reasoning can be applied for the weights and number of vertices to be considered. These limits are given by Equations (29) – (31), which can be applied to all three relaxations (i.e. Equations (29) and (31) for SSR1, Equations (30) and (31) for SSR2, and Equations (29) – (31) for SSR3).

$$1 \leqslant p' \leqslant 1 + |V_z|, \tag{29}$$

$$q_z \leqslant q' \leqslant q_z + \sum_{i \in V_z} q_i, \tag{30}$$

$$r_z \leqslant r' \leqslant r_z + \sum_{i \in V_z} r_i. \tag{31}$$

The set of reachable vertices V_z is computed only once for each problem, regardless of the relaxation being used. If SSR1 is being used there is no need to store V_z, as the values of $r_{V_z} = \sum_{i \in V_z} r_i$ and $|V_z|$ for each vertex z remain unchanged. If SSR2 or SSR3 is being used we must keep the values of the sets V_z since, as a result of changing the weights, the values of $q_{V_z} = \sum_{i \in V_z} q_i$ must be recomputed whenever state space modifications are applied.

Constraints using the cardinality of the partition

The computation of state (p, q, r, x) involves four cycles, one for each state variable (p', q', r', and z). The vertex cycle, where the variable z is decided, is performed first. This allows us to impose the constraints discussed previously. If the second cycle is chosen to be the one associated with variable p' (cardinality cycle), we are then able to reduce further the range of values that the variables q' and r' can take. Let $R_{max}(x, p)$ and $R_{min}(x, p)$ be the maximum and minimum flow, respectively, that a set of p vertices not containing vertex x may require. Let $Q_{max}(x, p)$ and $Q_{min}(x, p)$ be defined in the same way. These matrices provide upper and lower limits on the values of flow and weight to be supplied by vertex x to a set of p vertices.

Let (p, q, r, x) be the state to be computed and (p', q', r', z) be the partition under consideration. Conditions on the maximum and minimum values both for flow and weights can be imposed as follows:

$$\begin{aligned} &r' \geqslant r - R_{max}(x, p - p' - 1) - r_x, \\ &r' \leqslant r - R_{min}(x, p - p' - 1) - r_x, \\ &r' \geqslant R_{min}(z, p' - 1) + r_z, \\ &r' \leqslant R_{max}(z, p' - 1) + r_z. \end{aligned} \tag{32}$$

$$\begin{aligned} &q' \geqslant q - Q_{max}(x, p - p' - 1) - q_x, \\ &q' \leqslant q - Q_{min}(x, p - p' - 1) - q_x, \\ &q' \geqslant Q_{min}(z, p' - 1) + q_z, \\ &q' \leqslant Q_{max}(z, p' - 1) + q_z. \end{aligned} \tag{33}$$

Only partitions satisfying conditions (32) and (33) are considered for the computation of state (p, q, r, x). Note that if SSR1 is being considered only constraints (32) that restrict the values of the flow can be imposed, while if SSR2 is being used none can be imposed as the variable associated with the cardinality is not being used.

4.3. IMPLEMENTATION OF THE LB ALGORITHM

The State Space Ascent (SSA) uses State Space Relaxation (SSR) iteratively to maximize the LB. The procedure implementing SSR3 was chosen since this is the relaxation that gives better results and also because the procedure for SSR1 or SSR2 can easily be deduced by eliminating the state variable q in the first case, or the state variable p in the second case. The bound used $B(\lambda, q)$ is given by Equation (26) and satisfies the additional constraints (27)–(33).

As explained before, the SSA algorithm implemented consists of three phases. During phase I the LB is improved by using penalties in a Lagrangian fashion. In phase II the algorithm picks-up from the best LB found during phase I and improves upon it by using state space modifications. The final phase uses again penalties to improve the LB and starts from the best LB obtained during the previous phases. At the end of the SSA algorithm an optimal solution to the NFP may have been found or a good LB on the problem has been obtained. The flow diagram for the LB algorithm is given in Figure 3.

In the implementation of the SSR procedure, which is used by the SSA, memory requirements are less restrictive than the time requirements. Thus, we choose to represent the relaxed state space as a matrix: a four-dimensional matrix if SSR3 is used and a three-dimensional matrix if either SSR1 or SSR2 is being used. For SSR3 the size of the matrix is $(n+1)^2 \cdot (Q+1) \cdot (R+1)$. There is some memory waste since only part of the matrix represents existing states and from these we are only interested in a small percentage. However, searching for the computed states, both to compute other states and backtracking to find the solution structure, is avoided.

States at stage 1 are initialized as in Equation (19). The algorithm is then developed in a backward–forward recursive way, starting at the final state, $(n+1, Q, R, t)$ and going backward (just visiting states and not computing them) until a state already computed is found. The procedure moves then forward computing all visited states by using states already computed. When a state is reached such that its computation involves the computation of other states, the procedure moves backward again. Such backward–forward development is repeated until a value for the final state that cannot be bettered is found. A major feature of this backward–forward procedure is that only a small part of the relaxed state space needs

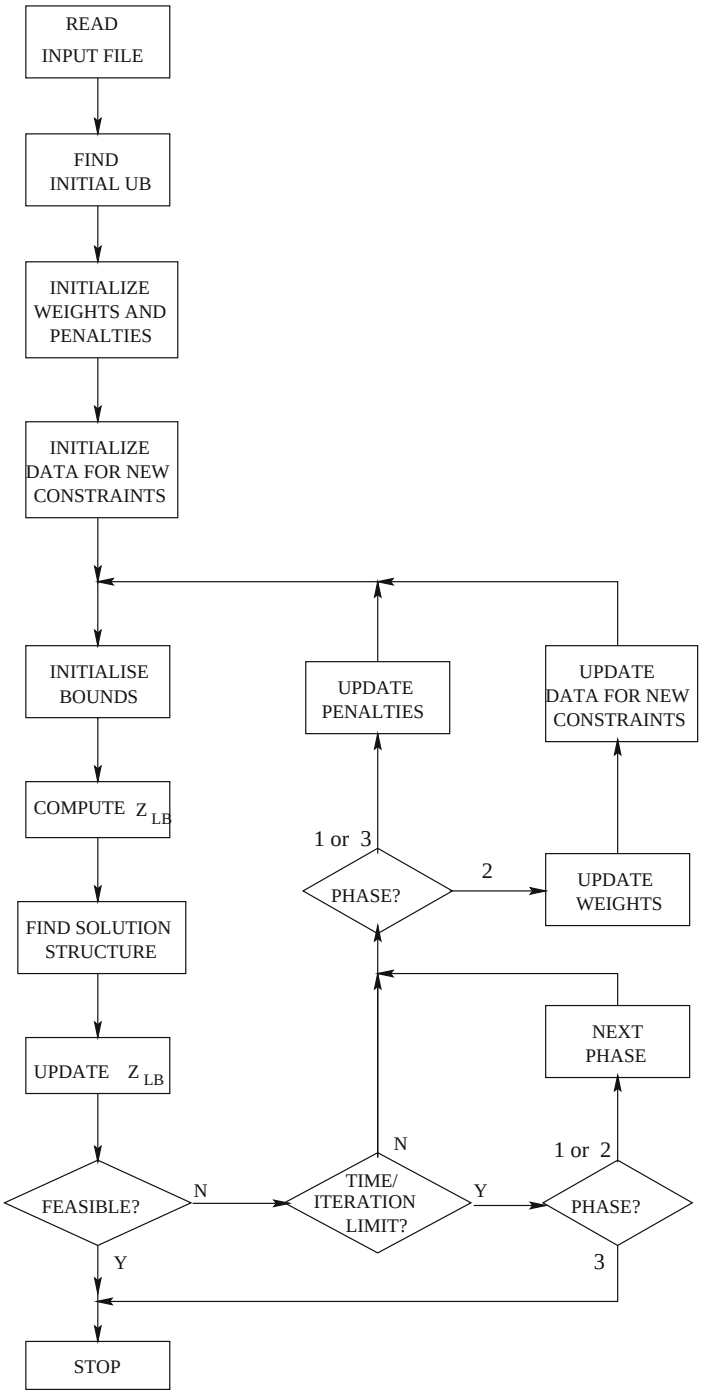

Figure 3. Flow diagram of the LB algorithm (using SSR3).

to be visited and from it just some states are actually computed. This is achieved, because the state space is being expanded using the information of the part of the graph already generated. Since a potential value for the state being computed is obtained at an early stage, a very large number of partitions are eliminated by comparing intermediate values with the current best value. Recall that, to compute a state there are three components that have to be computed and added: $g_{xz}(r')$, $f^3(p-p', q-q', r-r', x)$, and $f^3(p', q', r', z)$. (A detailed description is given in Appendix A.)

5. Computational Experiments

The algorithms presented in this paper were implemented in Fortran and computationally evaluated on a 200 MHz Pentium PC with 64 MB of RAM.

5.1. TEST PROBLEMS

The set of test problems used here has been previously used in Fontes et al. (2003, 2005b) and is available for download from the OR-Library (Beasley, OR-L). Two types of cost function were considered: polynomials of degree 1 and polynomials of degree 2. (We decided to choose polynomial functions, since any function can be easily approximated by a Taylor series.) The former include a linear variable cost and a fixed-cost, while the latter include a concave variable cost and a fixed-cost. For ease of notation we denote these problems as linear FC and concave FC, respectively.

Type I: linear FC

$$g_{ij}(x) = \begin{cases} b_{ij} \cdot x + c_{ij} & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Type II: concave FC

$$g_{ij}(x) = \begin{cases} -a_{ij} \cdot x^2 + b_{ij} \cdot x + c_{ij} & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$$

The cost function g_{ij} is nondecreasing and a_{ij}, b_{ij}, and c_{ij} are non-negative.

To the best of our knowledge NFPs with cost functions of type II have only been considered in Fontes (2000), Burkard et al. (2001), Fontes et al. (2003, 2005b).

For each problem n points were drawn uniformly from the unit square and for existing arcs b_{ij} was set to the integer nearest hundred times the Euclidean distance between points i and j. The source vertex is located at an extreme point of the unit square, as this makes the problems harder. The concavity coefficient a_{ij} for concave FC costs takes the maximum value that still guarantees the cost function to be nondecreasing. This means that a_{ij} increases with the increase of b_{ij} and as these two parameters are used with opposite sign in the cost function a measure of equilibrium for all arc costs is achieved. This equilibrium results in having much harder problems as there will be many more local optima solutions. The expected ratio

Table 1. Parameters for the 10 groups of problem instances

Groups	1	2	3	4	5	6	7	8	9	10
M	5	5	5	5	5	10	10	10	10	10
β	300	30	3	1.5	0.3	550	55	5.5	2.75	0.55
V/F	0.01	0.1	1	2	10	0.01	0.1	1	2	10

between the variable cost and the fixed cost V/F is an important measure of the problem difficulty (Hochbaum and Segev, 1989). In our computational experiments it can take five different values and the fixed-cost components were derived such that c_{ij}/b_{ij} (defined as β) was given by $(M+1)/(2V/F)$, as shown in Table 1. The demands are integer numbers uniformly drawn from $\{1, 2, \ldots, M\}$, where M was set to either 5 or 10. The number of arcs m was set to the nearest integer to a random number uniformly drawn between 3 and 4 times the number of vertices.

For problems with 10, 12, 15, 17, and 19 vertices all ten groups were considered for each size, while for problems with 25 and 30 vertices the first five groups were considered for each size. Each group contains three problem instances of the same type. Overall, computational experiments were carried out on 360 problem instances.

5.2. COMPUTATIONAL RESULTS

In Tables 2–4 we report results for phases I–III, respectively. All tables give the average optimality gaps and the average number of iterations and computational time needed to find them. We also report on the number of problem instances for which an optimal solution has been found at the end of phase I and II. During phase III no additional optimal solutions have been found. In Tables 2 (a), 3 (a), and 4 (a) the results reported are for linear FC NFPs, while in Tables 2 (b), 3 (b), and 4 (b) the results reported are for concave FC NFPs.

For linear FC NFPs very good quality LBs can be obtained at the end of phase I, while for concave FC NFPs, within similar computational time, the LBs have much larger optimality gaps. This, does not come as a surprise since concave FC problems are in general much "harder" than linear FC problems.

In Table 3 we provide the results obtained at the end of phase II. Comparing Tables 2 and 3 one can observe improvements in the optimality gaps achieved at the expense of a greater number of iterations and higher computational time. The magnitude of the improvements obtained decreases with problem size, specially for concave FC NFPs.

In Table 4 the final results are presented. As it can be seen, there is some improvement in the quality of the LBs during the last phase. However, the

Table 2. Results for (a) linear and (b) concave FC NFPs (phase I)

	n	Iters.	Time	Gap(%)	#opt.
(a)	10	392	00:00:09	0.68	6
	12	586	00:00:43	0.94	4
	15	628	00:02:51	0.75	0
	17	720	00:08:23	1.32	0
	19	772	00:22:19	1.45	0
	25	917	00:29:02	1.92	0
	30	918	01:03:26	1.70	0
(b)	10	673	0:00:20	3.41	2
	12	768	0:01:20	5.43	0
	15	788	0:04:52	6.51	0
	17	814	0:12:45	7.03	0
	19	831	0:27:20	7.71	0
	25	1014	0:25:06	9.49	0
	30	854	1:18:13	9.56	0

Table 3. Results for (a) linear and (b) concave FC NFPs (phase II)

	n	Iters.	Time	Gap(%)	#opt.
(a)	10	399	00:00:12	0.0004	24
	12	655	00:16:30	0.06	19
	15	694	00:44:32	0.36	5
	17	743	00:40:24	0.82	3
	19	799	00:55:53	0.97	2
	25	929	01:28:28	1.96	0
	30	925	02:04:25	1.69	0
(b)	10	758	00:01:02	0.0002	29
	12	894	00:32:27	0.71	19
	15	847	01:14:20	4.05	4
	17	865	01:18:13	4.45	1
	19	863	01:55:27	5.32	1
	25	1021	00:63:13	8.76	0
	30	862	03:15:12	9.30	0

increase in the computational time required is too large for such a small quality improvement. Therefore, at least for larger size problems it may not worth to perform all three phases, see Figure 4 for a comparison of phase I and phase III performances.

In Figures 5 and 6, we plot the methods performance against problem group. Although in Hochbaum and Segev (1989) and Ortega and Wolsey (2003) it has been concluded that problems are "easier" for extreme values of the ratio V/F and "harder" for middle ones, we are not able to find a performance changing pattern with problem group since the solution

Table 4. Results for (a) linear and (b) concave FC NFPs (all phases)

	Size	Iters.	Time	Gap(%)
(a)	10	413	00:00:12	0.0004
	12	673	00:20:00	0.04
	15	735	01:37:42	0.25
	17	776	02:10:23	0.56
	19	844	06:12:45	0.76
	25	967	04:23:04	1.61
	30	970	13:48:38	1.55
(b)	10	716	00:00:57	0.0002
	12	904	00:39:44	0.36
	15	886	02:47:32	3.42
	17	914	03:57:43	3.61
	19	914	06:38:36	4.36
	25	1083	02:29:16	7.94
	30	910	11:50:10	8.24

quality improves with the V/F ratio for linear FC NFPs, while it deteriorates for concave FC NFPs.

No result comparisons are possible since the only authors addressing problems as general as ours (Burkard et al., 2001) do not report LB results. Any other comparison, if possible, would be unfair because either the methods would be tailored for a specific type of problems, e.g. (Hochbaum and Segev 1989) for linear FC problems, or the problems addressed have concave costs with no fixed costs (Guisewite and Pardalos, 1991b), or have only some concave arc costs the remainder being linear (Horst and Thoai, 1998), or have only some vertices with demand (Gallo et al., 1980; Guisewite and Pardalos, 1991b).

6. Conclusions

In this work we reviewed the main ideas behind state space relaxation and expand its application to multistage problems, in particular to NFPs. The LBs developed can be used to evaluate the quality of heuristic solutions and to give information on the solution structure, as reported in Fontes et al (2003). An alternative, and perhaps more common use is the incorporation into a BB scheme as in the paper by Fontes et al. (2005a).

We provide three distinct relaxations of the DP formulation Fontes et al. (2005b) of the concave cost NFP, based on three forms of mapping function. The cardinality relaxation, for concave cost NFPs, has the advantage that the correct number of arcs, in a feasible solution to the original problem, must be used in any solution to the relaxed problem. In general, it has the advantage of relaxing the original state space into a very

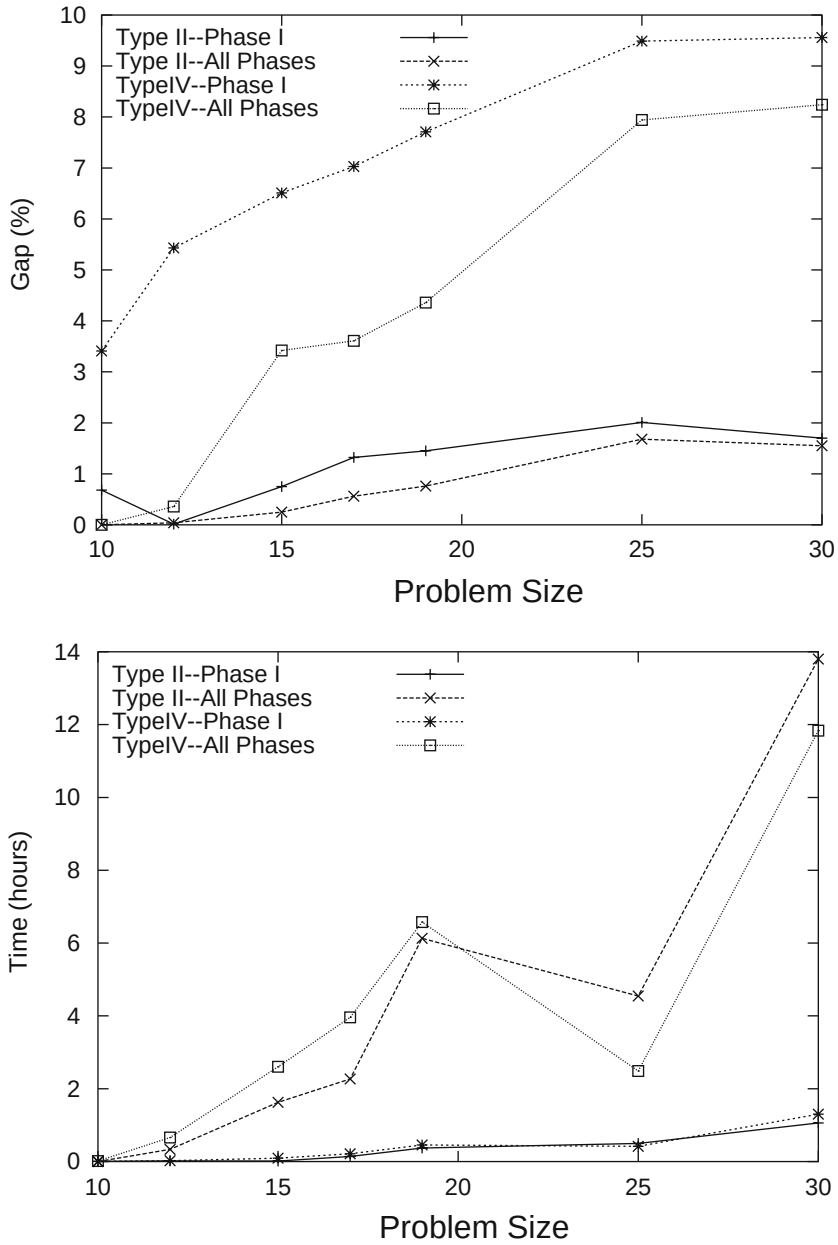

Figure 4. Average gap and time per problem size for both cost functions, phases I and III.

small relaxed state space, which is responsible for modest computational time and memory requirements. This can also be seen as its major drawback since too much information is lost as too many original states may be mapped into the same single relaxed state. The q-set relaxation, a generalization of the cardinality relaxation, gives better bounds as the state space

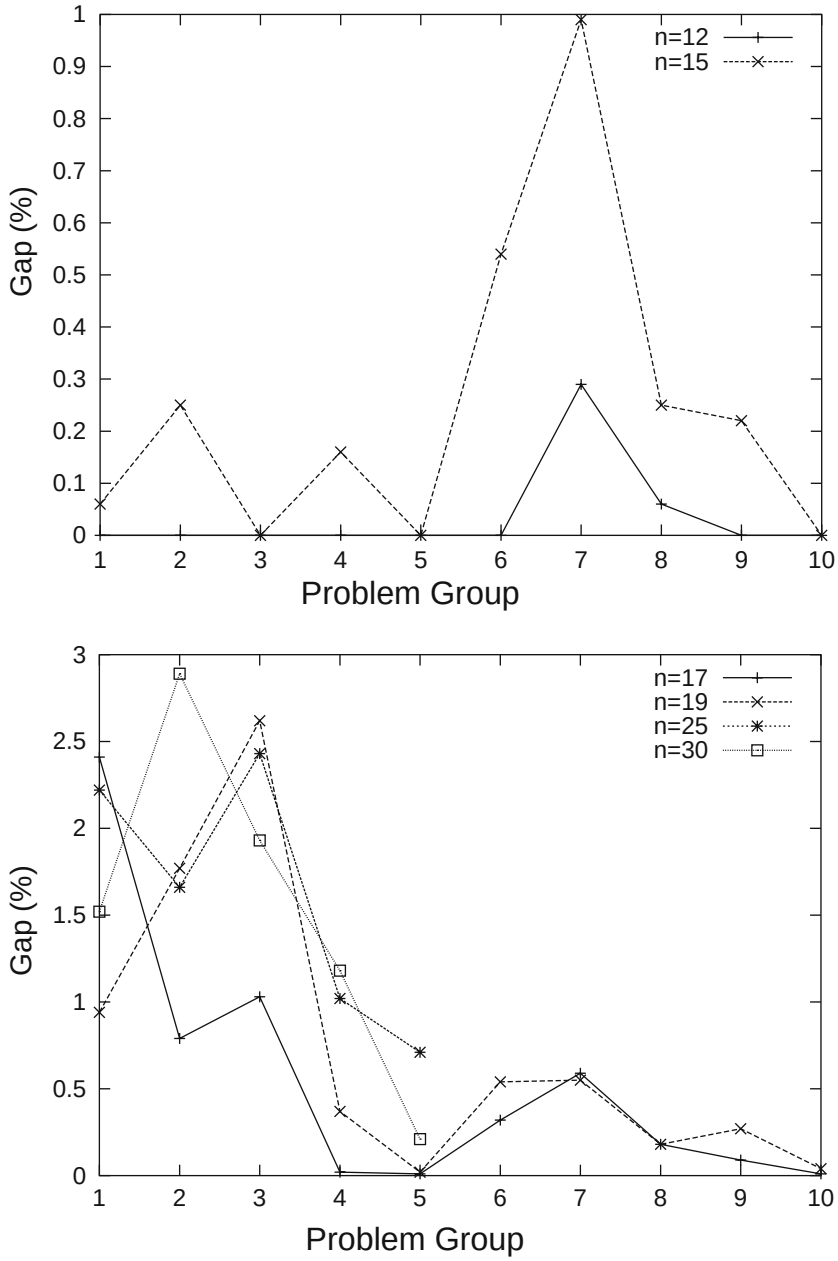

Figure 5. Average gap per group for FC NFPs.

is mapped more uniformly. Given that the state space obtained is usually larger, more computational burden is expected. A relaxation combining the advantages of both previous relaxations has been proposed. The combined relaxation corresponds to the q-set relaxation with the extra constraint that exactly n arcs must be used in the solution of a problem with $n+1$ vertices.

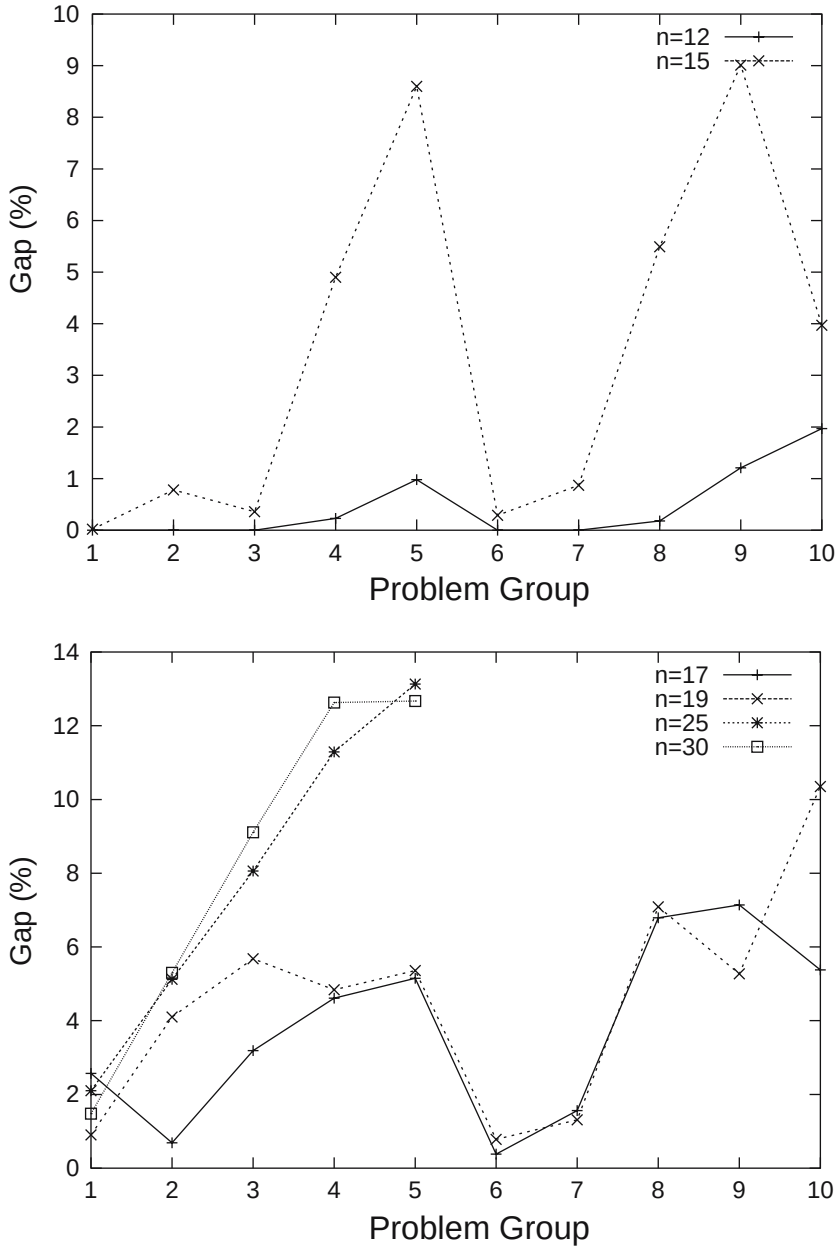

Figure 6. Average gap per group for concave FC NFPs.

Furthermore, this constraint is also valid for each subproblem. This allows the search to be quicker and more effective, as getting trapped into a local optima is more unlikely and more structure of the solution to the original problem is kept.

The bounds obtained have been improved by forcing the solution "closer" to feasibility, which is achieved by penalizing the customers that are not exactly satisfied and/or by using state space modifications. We also have developed three different sets of constraints, that although redundant to the original problem can be used to reduce the searchable state space by enforcing the use of sets and by reducing the partitions to be considered by limiting the range of values that the cardinality, flow, and weight variables can take.

An efficient implementation of the SSA algorithm was described in which the state space graph is gradually expanded using a backward–forward procedure on each layer of the state space. The expansion process requires information only on the part of the graph which has already been generated, resulting in a powerful implementation that is capable of detecting the needs of a particular problem and behaving accordingly. Computational experiments on 360 randomly generated test problems of varying size and complexity have shown the effectiveness of our methodology. For some test problems the optimal solution is found. Actually we find the optimal solution for almost all (53 out of the 60 solved) of the 10-vertex problems, the majority (forty) of the 12-vertex problems and, for some (nine) of the 15-vertex problems. In order to be able to find the optimal solution for all problems a BB procedure can be developed that incorporates these LBs. This procedure is the subject of the paper (Fontes et al., 2005a).

For linear FC cost NFPs the quality of the LBs is particularly good and, basically it does not deteriorate with problem size. Unfortunately, this is no longer the case for concave FC cost NFPs. Nonetheless, for the larger size problems considered (25 and 30 vertices) the average optimality gaps found are quite similar.

Furthermore, the concave problems we consider are amongst the most difficult ones since all arcs have a concave variable cost and a fixed cost. To the best of our knowledge, these type of problems have only been addressed by Burkard et al. (2001), Fontes et al. (2003, 2005b) . An unavoidable drawback of our methodology is that time requirements grow rapidly with problem size.

Appendix A: Description of the SSR Procedure

The procedure consists of three modules: *Initialize*, *Calculate*, and *UsedArcs*. The first module, *Initialize* attributes the initial values to the function f as in Equation (19). The module *Calculate* is a recursive function that computes the value of the final state $f(P, Q, R, t)$ by computing all the necessary intermediate states. This module is the core module of the algorithm, and thus a detailed description of this recursive function is given below. The last module, *UsedArcs* is also recursive and by using the

backtrack information finds the arcs used to supply the customers as well as the amount of flow routed through each used arc.

Recursive Function Calculate(p, q, r, x)

1. If already calculated, Goto 4.
2. $min = Z_{UB} + 1$.[3]
3. For each vertex z satisfying: $q_z \leqslant q - q_x$, $r_z \leqslant r - r_x$, and $z \neq x$.
   (a) If arc (x, z) does not exist, Goto 3.
   (b) If $g_{xz}(r_z) \geqslant min$, Goto 3.
   (c) For each triplet (p', q', r') satisfying Equations (27)–(33).
      i If $g_{xz}(r') \geqslant min$, Goto 3.
      ii aux $= g_{xz}(r') + Calculate(p - p', q - q', r - r', x)$.
      iii If aux $\geqslant min$, Goto (c).
      iv aux = aux + Calculate (p', q', r', z).
      v If (aux $\geqslant$ min), Goto (c).
      vi Update min = aux and store backtrack information (p', q', r', and z).
4. Return to caller.

References

Beasley, J.E. (OR-L), 'Or-Library', http://www.brunel.ac.uk/depts/ma/research/jeb/ info.html.

Burkard, R.E., Dollani, H. and Thach, P.H. (2001), Linear approximations in a dynamic programming approach for the uncapacitated single-source minimum concave cost network flow problem in acyclic networks, *Journal of Global Optimization*, 19, 121–139.

Christofides, N. and Hadjiconstantinou, E. (1995), An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts, *European Journal of Operational Research*, 83, 21–38.

Christofides, N., Mingozzi, A. and Toth, P. (1981), State space relaxation procedures for the computation of bounds to routing problems, *Networks*, 11, 145–164.

Christofides, N. and Paixão, J. (1993), Algorithms for large scale set covering problems, *Annals of Operations Research*, 43, 261–277.

Fontes, D.B.M.M. (2000), Optimal Network Design Using Nonlinear Cost Flows, PhD thesis, The Management School, Imperial College of Science Technology and Medicine, London, U.K.

Fontes, D.B.M.M., Hadjiconstantinou, E. and Christofides, N. (2003), Upper bounds for single source uncapacitated minimum concave-cost network flow problems, *Networks*, 41, 221–228.

Fontes, D.B.M.M., Hadjiconstantinou, E. and Christofides, N. (2005a), A branch-and-bound for the uncapacitated single source minimum concave cost network flow problem. *This Journal*.

[3]The upper bound (Z_{UB}) is obtained as in fontes et al. (2003). If only one optimal solution is required then min is set to Z_{UB} instead.

Fontes, D.B.M.M., Hadjiconstantinou, E. and Christofides, N. (2005b), A dynamic programming approach for solving single-source uncapacitated concave minimum cost network flow problems, *European Journal of Operational Research*, in Press.

Gallo, G., Sandi, C. and Sodini, C. (1980), An algorithm for the min concave cost flow problem, *European Journal of Operational Research*, 4, 249–255.

Guisewite, G.M. (1994), Network problems. In Horst, R. and Pardalos, P.M. (Eds.), *Handbook of Global Optimization*, Kluwer Academic, Dordrecht, pp. 609–648.

Guisewite, G.M. and Pardalos, P.M. (1991a), Algorithms for the single-source uncapacitated minimum concave-cost network flow problem', *Journal of Global Optimization*, 3, 245–265.

Guisewite, G.M. and Pardalos, P.M. (1991b), Global search algorithms for minimum concave-cost network flow problems, *Journal of Global Optimization*, 1, 309–330.

Hadjiconstantinou, E., Christofides, N. and Mingozzi, A. (1995), A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations, *Annals of Operations Research*, 61, 21–43.

Held, M., Karp, R.M. and Crowder, H.P. (1974), Validation of subgradient optimization, *Mathematical Programming*, 6, 62–88.

Hochbaum, D.S. and Segev, A. (1989), Analysis of a flow problem with fixed charges, *Networks*, 19, 291–312.

Horst, R. and Thoai, N.V. (1998), An integer concave minimization approach for the minimum concave cost capacitated flow problem on networks, *OR Spectrum*, 20, 45–53.

Kim, D. and Pardalos, P.M. (1999), A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure, *Operations Research Letters*, 24, 195–203.

Kim, H.-J. and Hooker, J. (2002), Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach, *Annals of Operations Research*, 115, 95–124.

Lamar, B.W. (1993), A method for solving network flow problems with general nonlinear arc costs. In Du, D.-Z. and Pardalos, P.M. (Eds.), *Network Optimization Problems*, World Scientific, Singapore.

Ortega, F. and Wolsey, L.A. (2003), A branch-and-cut algorithm for the single-commodity, uncapacitated, fixed-charge network flow problem, *Networks*, 41, 143–158.

Zangwill, W.I. (1968), Minimum concave cost flows in certain networks, *Management Science*, 14, 429–450.