Optimal Trees for General Nonlinear Network Flow Problems: A Dynamic Programming Approach

Dalila B. M. M. Fontes
Faculdade de Economia da Universidade do Porto
Adress: Rua Dr. Roberto Frias, 4200-464 Porto, Portugal
Phone: 351-225 571 100, fax: 351-225 505 050
Email: fontes@fep.up.pt

March 2005

Abstract

In this paper, we describe a dynamic programming approach to find optimal trees to the single-source minimum cost network flow problem with general nonlinear costs. This class of problems is known to be NP-*Hard* and there is a scarcity of methods to address them. The algorithms previously developed have considered only two particular types of cost functions: "staircase" and "sawtooth". Here, a dynamic programming approach to find optimal trees, that can be used with any kind of separable and additive cost function, is proposed. Computational experiments were performed using randomly generated problems and the results reported, for small and medium size problems, indicate the effectiveness of the proposed approach.

keywords Dynamic programming, network flows, optimal trees, general nonlinear costs

1 Introduction

The main feature defining the complexity of Minimum Cost Network Flow Problems (MCNFPs) is the type of cost function for each arc. In this sense, MCNFPs can be divided into four categories with increasing complexity, namely: linear, convex, concave, and general nonlinear. Linear MCNFPs, have constant marginal arc costs and can be solved in polynomial time [2]. Convex MCNFPs, which have nondecreasing marginal arc costs, involve the minimization of a convex objective function over a convex feasible region, defined by the network constraints. Therefore, a local optimum is also a global optimum. Although harder than linear MCNFPs, these problems are still "easy to solve" [2]. Concave MCNFPs have nonincreasing marginal arc costs and are much harder than the previous MCNFPs. The complexity of this type of problems arises from minimizing a concave function over a convex feasible region, which implies that a local optimum is not necessarily a global optimum. Although concave MCNFPs are known to be NP-*hard* [8] (even for the simplest version), they do exhibit some special mathematical properties that make them more tractable than general nonlinear MCNFPs [7]. For a recent discussion on general concave MCNFPs, see for example [3, 5] for approximate methods and [4, 6] for exact methods.

General nonlinear MCNFPs have arc costs that are neither convex nor concave such that no convexity or concavity properties can be explored in the determination of an optimal solution. This type of problems is also known as indefinite or discontinuous MCNFPs. Many practical problems involve some sort of discontinuity in cost functions. For example, transport of passengers typically include a fixed cost proportional to the number of vehicles (fuel, drivers, insurance) and thus if the flow is in number of passengers, then the cost function is in the form of a “staircase”. Another type of cost function, known as “sawtooth” may arise in goods distribution as costs are usually made up of a marginal cost, that typically decreases with quantity and a fixed cost that can introduce discontinuities at quantity “breakpoints”. A cost function that is concave up to a certain value and convex afterwards may also appear in production settings, for example due to market reaction for demand of a raw material.

To the best of our knowledge no optimization methods have been reported in the literature for general nonlinear MCNFPs. Existing literature considers only specific types of cost functions, namely staircase [11, 10] and sawtooth [11] cost functions. Lamar in [11] proposes to address the MCNFP with sawtooth or staircase arc costs by transforming the problem into a concave MCNFP on an expanded network. (This transformation is based on the fact that any arbitrary cost function can be rewritten as a summation of convex and concave functions.) However, as the resulting MCNFP is defined over a much larger network, this transformation is only useful for very small problems. Recently, Kim and Pardalos [10] obtained good quality solutions for the nonconvex piecewise linear MCNFP. They solve linear problems that are recursively updated by using the previous solution. At each iteration the feasible domain is reduced by a contraction rule. A recent survey on piecewise linear MCNFPs, including staircase and sawtooth cost functions is given in [9].

In this work, we have decided not to restrict the type of cost functions to be considered, but rather the network configuration to be searched for. We describe an algorithm to find the best solution to the general nonlinear MCNFPs among tree shaped solutions. We focus on trees since they are a basic structure in combinatorial problems and represent a fundamental element in a large number of graphic theoretical problems. On the one hand, they are of considerable importance on their own right and have many planning and design applications in a variety of network problems. On the other hand, there are many network flow algorithms which are based on finding optimal trees, both as a final objective or as an intermediate step. A detailed study on optimal trees can be found in [12]. The algorithm described here is based on Dynamic Programming (DP) and is an adaptation of an approach previously developed for concave MCNFPs [4].

2 Problem description and formulation

Our problem consists of finding an optimal tree for the general nonlinear MCNFP. Consider a directed network $G = (W, A)$ where W is a set of $n + 1$ vertices containing the source vertex and n demand vertices and A is a set of m directed arcs. Vertices 1 to n have a nonnegative integer demand r_i and the supply at the source vertex R matches the commodity required by the n demand vertices. Flow on each arc (i, j) has upper u_{ij} and lower l_{ij} limits. A general nonlinear and nonnegative cost function g_{ij} is assigned to each arc. The cost of sending r units of flow through arc (i, j) is given by $g_{ij}(r)$ and satisfies $g_{ij}(0) = 0$.

We developed a DP model to find a tree network and corresponding flows such that demands are satisfied at minimum cost. The formulation proposed is independent of the type of cost functions considered and of the number of nonlinear arc costs. Also, the cost functions may be neither differentiable nor continuous, having only to be separable and additive.

Consider a set $S \subseteq W$ and a vertex $x \in S$. Let $\left\{S', \bar{S}'\right\}$ be one partition of set S, where $S' \subseteq S\backslash\{x\}$ and $\bar{S}'$ is the complement of S' in the set S, that is $\bar{S}' = S - S'$. For each possible set S', let $z \in S'$ be the root vertex of a directed tree spanning the set S' and let $r = \sum_{i \in S'} r_i$ be the total commodity required by the demand vertices in set S'.

Let $f(S', z)$ be the minimum cost of supplying all demand vertices in S' with the required commodity r 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 r made available at some vertex $z \in S'$ satisfying $l_{xz} \leq r \leq u_{xz}$ 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

$$\min_{\substack{z\in S' \\ l_{xz}\le r\le u_{xz}}} \left\{ f\left(S', z\right) + g_{xz}\left(r\right) \right\}.$$

By definition, the minimum cost incurred in supplying, through a tree, the remaining demand vertices of set S not in S' from x is given by $f\left(\bar{S}', x\right)$. Thus, the minimum cost $f(S,x)$ of supplying all demand vertices in S, with the commodity available at $x \in S$ through a directed tree rooted at x, is obtained by examining all possible subsets $S' \subseteq S \setminus \{x\}$, which is given by

$$f(S,x) = \min_{S'\subseteq S\setminus\{x\}} \left[f\left(S - S', x\right) + \min_{\substack{z\in S' \\ l_{xz}\le r\le u_{xz}}} \left[f(S', z) + g_{xz}(r) \right] \right], \qquad (1)$$

where $r = \sum_{i\in S'} r_i$.

Initial conditions are provided by

$$f\left(S,x\right) = \begin{cases} 0, & \text{if } S = \{x\} \\ \infty, & \text{otherwise.} \end{cases} \qquad (2)$$

Recursion (1) applies for all $S \subseteq W$ and all $x \in S$. Hence, the cost of the optimal tree supplying all demand vertices in set W from the source vertex t, is given by $f(W,t)$, if one exists.

An illustration is given in Figure 1, which 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.

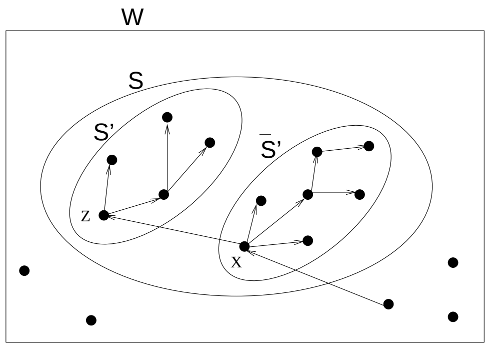

Figure 1: A flow pattern of supplying set S with the commodity available at vertex x through a directed tree.

3 Implementation of the DP algorithm

A pure forward DP algorithm is easily derived from the DP recursion by generating all the states of a particular stage one by one. Such implementation may result in considerable waste of computational effort either when complete enumeration of the state space is not required, or when some states are not feasible. In the latter case, the infeasibility of a state is only discovered after it has been generated. Thus, we have implemented the DP formulation based on the idea of gradually expanding the state space graph using a backward-forward procedure on each layer of the state space. Its main advantage is that the expansion of the state space graph is based upon the information relevant to the part of the graph which has already been generated. Therefore, states which are not feasible for the problem are not computed, as only states which are needed for the computation of the solution are considered. The algorithm is dynamic as it detects the needs of a particular problem and behaves accordingly.

States at stage one are either nonexistent or initialized as in equation (2). The algorithm starts from the final state (W, t) and while moving backward visits, without computing, possible states until a state already computed is reached. Then, the procedure is performed in reverse order, i.e. starting from the state last identified in the backward process, it goes forward through computed states until a state (S, x) is found which has not yet been computed. At this point, again it goes backward until a computed state (S', z) is reached. This procedure is repeated until the final state (W, t) is reached with a value that cannot be bettered by any other alternative solution. The main advantage of this backward-forward recursive algorithm is that only intermediate states needed are visited and from these only the feasible ones that may yield a better solution are computed. As will be shown latter only an average of 21% to 25% of the states are computed.

After initialization, which is given by equation (2), the optimal tree cost $f(W, t)$ is obtained by calling the recursive function *Compute*(W, t).

Compute(S, x)

```
If $f(S,x) \neq \infty$ then return $f(S,x)$ to caller
Set $min = \infty$
For each $S' \subseteq S$
    Call $Compute(S \setminus S', x)$
    If $f(S \setminus S', x) \geq min$ then get another subset $S'$
    For each $z \in S'$
        If $(x, z) \notin A$ then get another vertex $z$
        $r = \sum_{i \in S'} r_i$
        If $r > u_{xz}$ or $r < l_{xz}$ then get another vertex $z$
        If $f(S \setminus S', x) + g_{x,z}(r) \geq min$ then get another vertex $z$
        Call $Compute(S', z)$
        If $f(S \setminus S', x) + g_{x,z}(r) + f(S', z) \geq min$ then get another vertex $z$
        $min = f(S \setminus S', x) + g_{x,z}(r) + f(S', z)$
        Store information on subset=$S'$, vertex=$z$, flow=$r$, and $f(S,x) = min$.
    End for
  End for
Return $f(S,x)$
```

At the end of the procedure, if $f(W, t) = \infty$ then no tree network exists satisfying the flow limits; otherwise $f(W, t)$ gives the cost associated with an optimal tree. The solution structure, i.e. the arcs used and the amount of flow routed through these arcs, is obtained by a recursive routine that backtracks through the information stored during the computation of intermediate states.

The complexity of the DP algorithm is, in the worst case, of the order $\mathcal{O}(n2^n)$. As expected, the complexity increases exponentially with the number of demand vertices. On the other hand, it should be noted that the DP model can be applied to MCNFPs with arbitrary cost functions without deteriorating its performance. As it will be shown in Section 4, the algorithm behaviour is independent of the type and number of nonlinear arc costs.

4 Computational results

The algorithm presented in this paper was implemented in Fortran and computationally evaluated on a 200MHz Pentium PC with 64 MB of RAM by solving a set of randomly generated test problems. The problems considered are amongst the most difficult problems as all arcs have cost functions that are neither convex nor concave. Three different types of cost functions are considered: type G1 and type G2 are variations of the fixed-charge cost function where discontinuities other than at the origin are introduced and type G3, which have arc costs that are initially concave and then convex having a discontinuity at the break point. Types G1 and G2 correspond to the so called staircase and sawtooth cost functions, see [9], in our case with two segments.

$$g_{ij}(r) = \begin{cases} 0, & \text{if } r = 0, \\ -a_{ij}r^2 + b_{ij}r + c_{ij} & \text{if } r \leq \hat{R}, \\ a_{ij}r^2 + b_{ij}r + c_{ij} + k & \text{otherwise,} \end{cases}$$

where $a_{ij} = 0$ for G1 and G2, $k = b_{ij}$ for G1, and $k = -b_{ij}$ for G2 and G3.

A graphical representation of the three cost functions considered is given in Figure 2.

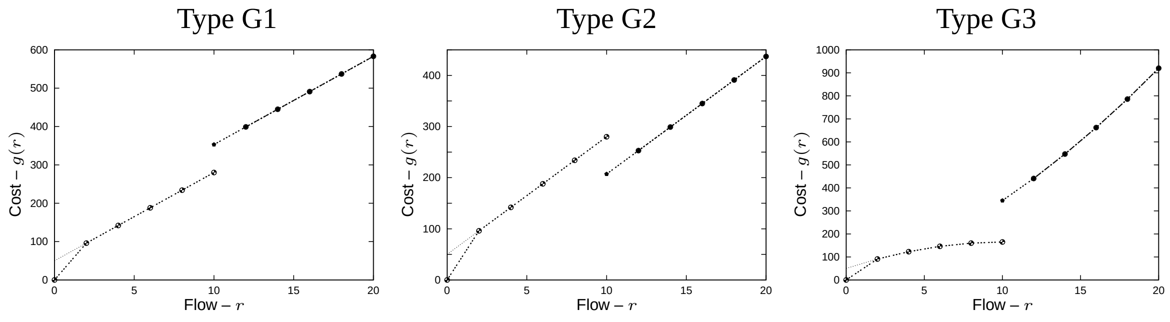

Figure 2: Types of general nonlinear cost functions considered.

The problems data can be downloaded from the OR-Library [1] and a thorough description of the generation procedure is provided in [5]. For each type of cost function five groups (groups 1 to 5 in [1]) of problems were considered, each group containing three problem instances of the same size. Problem group is mainly defined by the expected ratio between the variable cost and the fixed cost (V/F), which was set to 0.01, 0.1, 1, 2, and 10. The value of $\hat{R}$ was set to 50% of the total demand R. Problem size, given by the number of vertices $n+1$, was set to 10, 12, 15, 17, and 19.

Overall, computational experiments were carried out on 225 unconstrained problem instances (15 instances were solved for each problem size and each type of cost function) and the computational results are presented in the following section.

Table 1 summarizes the results for all test problems solved using the DP algorithm described in this paper. The figures shown in this table were obtained as averages over 15 problem instances of a given problem size and cost function type. In order to show even stronger evidence that the methods performance is independent of cost function type we also give the results obtained for the same set of problems considering linear cost functions ($b_{ij}r$).

Two measures of performance were computed for each problem: Time-the computational time (h:mm:ss) required to find an optimal solution; and S_{comp}-[1]the percentage of the state space that is actually computed by the DP algorithm.

The results reported show that a significant reduction in the state space enumeration has been achieved for all problems (only an average of 21% to 25% of states have to be computed). On the other hand, the computational time increases exponentially with problem size. Nevertheless, as expected, the computational time is independent of the cost function type, even when comparing general nonlinear and linear costs. The algorithms performance, can also be observed in the graph of Figure 3, which displays the computational time versus problem size for each type of cost function.

[1] $S_{comp} = \frac{\text{no. of states computed}}{\text{total no. of states}} \times 100\%.$

Size	Linear		Cost Type G1		Cost Type G2		Cost Type G3	
N	Time	S_{comp}	Time	S_{comp}	Time	S_{comp}	Time	S_{comp}
10	0:00:01	22.72	0:00:01	24.10	0:00:01	24.15	0:00:01	23.85
12	0:00:07	22.66	0:00:05	22.18	0:00:08	23.37	0:00:08	23.18
15	0:03:24	22.14	0:03:49	22.60	0:04:02	22.61	0:03:52	21.51
17	0:35:50	22.19	0:38:30	22.69	0:40:32	22.69	0:39:15	22.63
19	5:15:41	21.39	5:44:45	21.05	5:45:14	21.06	5:45:36	21.3

Table 1: Computational performance for each problem size and cost function type.

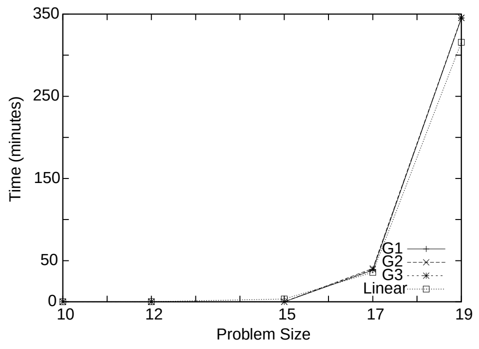

Figure 3: The effect of problem size on computational time for each cost function type.

5 Conclusions

In this paper we presented a DP methodology for finding optimal tree-networks to general nonlinear MCNFPs. In fact, the cost functions do not have to be differentiable nor continuous. Also, they might be neither convex nor concave having only to be separable and additive.

Not many work has been reported in literature involving MCNFPs with nonlinear arc costs that are neither convex nor concave. The works found, although searching for any solution structure, address only staircase and sawtooth cost functions.

Optimal trees are of importance since they constitute the simplest form of network which can be used for distribution. Furthermore, even when a more complex network is being thought of, a tree is still of great importance either as a starting point or as building block.

The algorithm implementation is based on the idea of gradually expanding the state space graph using a backward-forward procedure on each layer of the state space. One of its main advantages is that the expansion of the state space graph requires information relating only to the part of the graph which has already been generated. A large number of randomly generated test problems of varying size and complexity was used to evaluate the algorithms performance and the results have shown it to be effective at solving small and medium size problems only (as time requirements grow exponentially).

References

[1] J. E. Beasley. Or-Library. http://www.brunel.ac.uk/depts/ma/research/jeb/info.html.

[2] D. P. Bertesekas. *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[3] R. E. Burkard, H. Dollani, and P. H. Thach. 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, 2001.

[4] D. B. M. M. Fontes, E. Hadjiconstantinou, and N. Christofides. A dynamic programming approach for solving single-source uncapacitated concave minimum cost network flow problems. 2003. Under Revision.

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

[6] D. B. M. M. Fontes, E. Hadjiconstantinou, and N. Christofides. A branch-and-bound algorithm for concave network flow problems. 2004. Submitted.

[7] G. M. Guisewite. Network problems. In R. Horst and P. M. Pardalos, editors, *Handbook in Global Optimization*, pages 506–648. Kluwer Academic, 1994.

[8] G. M. Guisewite and P. M. Pardalos. Algorithms for the single-source uncapacitated minimum concave-cost network flow problem. *Journal of Global Optimization*, 3:245–265, 1991.

[9] D. Kim. Piecewise linear network flow problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*. Kluwer Academic Publisher, 2003.

[10] D. Kim and P. M. Pardalos. A dynamic domain contraction algorithm for nonconvex piecewise linear network flow problems. *Journal of Global Optimization*, 17:225–234, 2000.

[11] B. W. Lamar. A method for solving network flow problems with general nonlinear arc costs. In D.-Z. Du and P. M. Pardalos, editors, *Network optimization Problems*. World Scientific, 1993.

[12] T. L. Magnanti and L. A. Wolsey. Optimal trees. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, chapter 9. Elsevier, 1995.