A Hybrid Genetic Algorithm Approach for Concave Minimum Cost Network Flow Problems

Dalila B. M. M. Fontes and José Fernando Gonçalves
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,jfgoncal}@fep.up.pt

March 2005

Abstract

This paper presents a hybrid approach combining a genetic algorithm with a local search for the single-source uncapacitated minimum cost network flow problem with general concave costs. This class of problems is known to be NP-*Hard* and therefore, exact methods to solve this class of problems in their full generality are only able to address small size instances. Computational experiments were performed using randomly generated problems. The results of the test problems are compared to optimal solutions obtained by dynamic programming method and to upper bounds obtained by a local search method for larger problem instances. From the results reported it can be shown that the hybrid methodology improves upon previous approaches.

keywords Network flows, genetic algorithms, local search, concave-cost optimization

1 Introduction

In this work we focus on the Single Source Uncapacitated (SSU) concave MCNFP, which can be used to address a broader class of MCNFPs. General nonlinear MCNFPs can be transformed into concave MCNFPs on an expanded network. In addition, multiple source and capacitated networks can be transformed into SSU networks. Many practical situations where a product is routed through a network exist. This can be seen from the wide and diverse application areas in which network flow problems have been utilized, see e.g. [8] and the references therein. The complexity of concave MCNFPs 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. Furthermore, there is no simple criterion for deciding whether a local minimum is also a global minimum.

Most of the work developed on concave MCNFPs considers SSU problems with Fixed-Charge (FC) cost functions, i.e. functions having a fixed cost and a linear routing cost. This problem is a particular case of the more general SSU concave MCNFP, although NP-*Hard* itself. A recent work on MCNFPs with FC costs is given in [10], where the authors also discuss other FC problems. Other works [9, 11] consider a routing concave costs but no fixed costs, MCNFPs involving both a concave routing cost and a fixed cost on each arc have only been considered in [3, 4, 5].

Due to the complexity of nonlinear network optimization problems and the fact that efficient methods exist only for highly structured subclasses or small size problems, considerable effort in the research community has been devoted to approximate methods. Burkard *et al.* [3] consider acyclic graphs with

small degree vertices and general concave cost functions. Linear approximations in a Dynamic Programming (DP) approach are obtained. For network flow problems defined on complete graphs, only problems with up to 21 vertices are solved, for which the time requirements are above 13 hours. A local search method has been developed by Fontes *et al.* [5] that uses information from the structure of a lower bound solution to the concave MCNFP in order to generate initial feasible solutions. In [11] Smith and Walters describe a Genetic Algorithm (GA) approach to the problem of finding optimal trees on networks at minimum cost. The initial population is made up of randomly generated feasible trees. A probability, which is proportional to the fitness is computed and assigned to each tree. After two trees T_1 and T_2 have been chosen, the two graphs are superimposed to form a new directed graph G. All arcs in trees T_1 and T_2 will be in graph G and will retain their directions. Mutations are introduced by randomly adding extra arcs to the graph G. The method is tested on SSU concave MCNFP having 15, 20, 25, 30, or 35 vertices. The quality of the solution and the computational time required to find them are not reported.

In this paper, we report a Hybrid Genetic Algorithm (HGA), where a local search strategy is incorporated into a GA, to solve the SSU MCNFP with general concave costs. The local search algorithm tries to improve the solutions in the population by using domain-specific information, while the GA recombines good solutions in order to investigate different regions of the solution space. The computational results are compared with results reported in literature and these comparisons allow for the conclusion that the HGA is not only effective but also efficient for SSU concave MCNFP.

2 Problem definition

The network $G = (W, A)$ to be optimized consists of a set W of $n + 1$ vertices (vertex $n + 1$ denotes the source vertex t and vertices $1, \ldots, n$ denote demand vertices) and a set A of m directed arcs, $A \subset \{(i, j) : i, j \in W\}$. Each demand vertex has an associated nonnegative integer value r_i. The supply at the source vertex matches the total demand required by the n demand vertices. 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 S of arcs to be used and the level of flow routed through them x_{ij} for all $(i, j) \in S$, in order to satisfy the demand at minimum cost. The problem can be cast in an intuitive way in the form of a nonlinear mathematical programming model, given by:

$$(\mathcal{P}) \quad \text{Minimize} \quad \sum_{(i,j)\in A} g_{ij}(x_{ij}) \tag{1}$$

subject to

$$\sum_{(k,i)\in A} x_{ki} - \sum_{(i,k)\in A} x_{ik} = r_i \qquad \text{for all } i \in W \setminus \{t\} \tag{2}$$

$$x_{ij} \geq 0 \qquad \text{for all } (i, j) \in A. \tag{3}$$

Equation (1) states the minimum cost nature of the problem, while equations (2) represent the flow conservation constraints.

Concave MCNFPs have the combinatorial property that if a finite solution exists, then an optimal solution occurs at a vertex (extreme point) of the corresponding feasible domain (defined by the network constraints). SSU concave MCNFPs have a finite 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 the SSU concave MCNFP a feasible flow is an extreme flow if it contains no cycles. Therefore, an extreme flow is a tree rooted at the single source spanning all demand vertices [13]. Thus, our objective is to find an optimal tree rooted at the source vertex that satisfies all customers demand at minimum cost.

3 Methodology

In our Hybrid Genetic Algorithm (HGA), we hybridize the GA described below with local search method. Local Search (LS) methods take advantage of the knowledge that an optimal solution to the SSU concave MCNFP is an extreme flow, i.e. a tree rooted at the single source spanning all demand vertices. The main advantage is that the search is confined to extreme flows. The main drawback is that there are many solutions that are local optima but not global optima where the method can get trapped. Thus, by combining the LS with the GA we take advantage of the "local" improvement capabilities of LS and also of the "global" nature of the GA. Our approach comprises a GA, a Tree-Constructor, and a LS. In Figure 1 the architecture of the proposed approach is illustrated. The GA is responsible for evolving

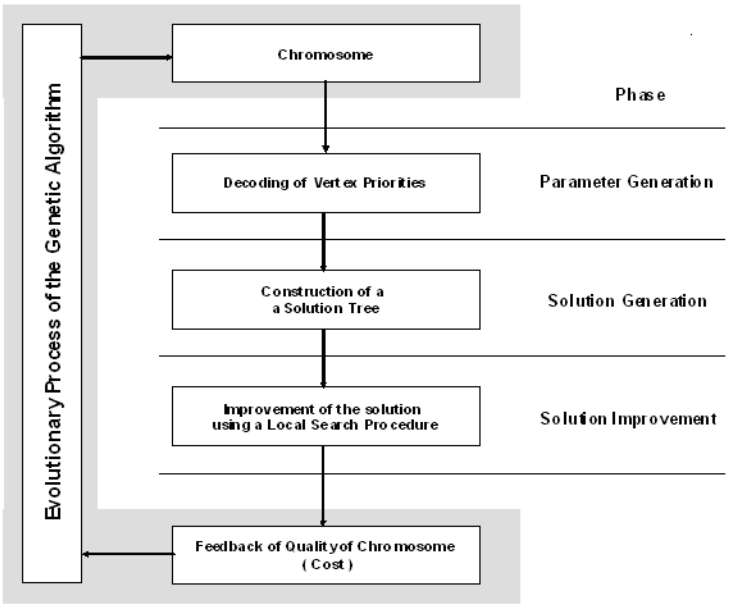

Figure 1: Architecture of the proposed approach.

the chromosomes, which represent the parameters (vertex priorities) used by the Tree-Constructor. For each chromosome the following are applied, in turn: *Decoding of priorities* that transforms the chromosome supplied by the GA into vertex priorities; a *Tree-Constructor* that constructs a feasible tree based on vertex priorities; and a *Local search* procedure which tries to improve the current solution by means of small perturbations. After a solution is obtained the corresponding quality (cost) is measured and fed back to the GA.

The Genetic Algorithm

The proposed GA uses a random key alphabet U(0,1) and an evolutionary strategy identical to the one proposed by [1]. The important feature of random keys is that all offspring formed by crossover are feasible solutions. Through the dynamics of the GA, the system learns the relationship between random key vectors and solutions with good objective function values. A chromosome, which is made of n genes (where n is the number of vertices) is encoded as a vector of random keys (random numbers). The n genes are used to obtain the priorities, which are then used by the Tree-Constructor. Each chromosome is decoded into priorities by sorting the genes and constructing the tree in ascending order, see Figure 2.

The reproduction and crossover operators determine which parents will have offspring, and how genetic material is exchanged between parents. Reproduction and crossover operators tend to increase the quality of the populations and force convergence. Mutation opposes convergence and replaces genetic

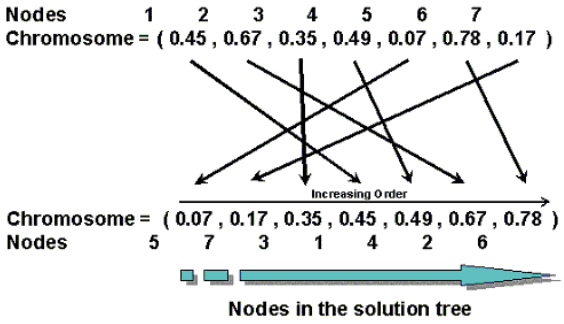

Figure 2: Chromosome decoding procedure.

material lost during reproduction and crossover. Reproduction is accomplished by first copying some of the best individuals from one generation to the next, in what is called an elitist strategy [7]. Parameterized uniform crossovers [12] are employed in place of the traditional one-point or two-point crossover. After choosing two parents randomly from the full, old population (including chromosomes copied to the next generation in the elitist pass), at each gene we toss a biased coin to select which parent will contribute the allele. It assumes that a coin toss of heads selects the gene from the first parent and of tails chooses the gene from the second parent. The probability value of tossing heads is determined empirically. Rather than the traditional gene-by-gene mutation with very small probability at each generation, one or more new members of the population are randomly generated from the same distribution as the original population. This process prevents premature convergence of the population, as in a mutation operator, and leads to a simple statement of convergence.

The local search

For concave MCNFPs, the standard marginal definition of local optimality, i.e., rerouting some flow, is not satisfactory, as for strictly concave costs it results in all extreme flows, i.e. trees, being locally optimal. Thus, we use the following $\mathcal{N}(X) = \{X' \mid X' \text{ is an extreme flow adjacent to the extreme flow } X\}$, where extreme flows X and X' are adjacent if $X \cup X'$ contains a single undirected path [6].

The LS improves on a given solution by comparing it with all adjacent extreme solutions, which are obtained by considering all augmenting paths connecting every pair of vertices. As no transshipment vertices exist, this corresponds to replacing an arc in the solution by an arc not in the solution such that the new solution is still a tree. This is repeated until no further cost reductions can be achieved.

The local search attempts to improve every single solution, i.e. tree, obtained by the Tree-Constructor.

4 Computational results

To test the efficiency and effectiveness of the HGA we use the problem set described in [5]. The authors consider two groups of problems: Eucledian problems, group A and random problems, group B. In this work only problems in group A are used and the input data can be downloaded from [2]. The results obtained by the HGA are compared to optimal solutions found by a DP approach [4] for problem instances with up to 19 vertices and to heuristic solutions found by a local search algorithm [5], for larger size problems. The results obtained so far show evidence of the efficiency of the proposed algorithm.

The following four types of cost function were used: polynomials of degree 0 (linear problems), degree 1 (fixed-charge problems), and degree 2 (problems having a concave routing component) both with and without a fixed cost component. The cost functions g_{ij} are nondecreasing and a_{ij}, b_{ij}, and c_{ij} are nonnegative.

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

More specifically, we consider: *Type I:* linear[1], obtained by setting $a_{ij} = 0$ and $c_{ij} = 0$; *Type II:* linear fixed-charge, obtained by setting $a_{ij} = 0$; *Type III:* concave, obtained by setting $c_{ij} = 0$; and *Type IV:* concave with fixed-charge.

For problems with 10 up to 19 vertices we consider all four cost functions and the results obtained are compared to the optimal solution values reported in [4]. For each problem size and cost type 30 problem instances have been solved (600 in total) and the average results are summarized in Table 1. From these, only for one (type II with 10 vertices) an optimal solution was not obtained. Furthermore, the largest computational time needed was 6.63 seconds (for a type IV with 19 vertices problem).

	Type I		Type II		Type III		Type IV	
size	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)	Gap(%)	Time(s)
10	0	0.713	0	0.823	0	0.844	0	0.895
12	0	1.071	0	1.232	0	1.318	0	1.424
15	0	1.802	0	2.109	0	2.237	0	2.501
17	0	2.644	0	3.146	0	3.267	0	3.735
19	0	3.241	0	4.004	0	4.097	0	4.625

Table 1: Computational performance for smaller size instances.

For larger size problems, with 25 up to 50 vertices, we solved fifteen problem instances for each problem size and cost type. The results obtained are compared to the CPLEX optimal solutions reported in [5] for type II problems and to the upper bounds reported in [5] for type IV problems. These problem instances have been solved for all four cost function types. However, in Table 2 we only report the results for types II and IV since only these are comparable to the results in [5].

	Type II				Type IV		
size	Gap†(%)	Time†(s)	Gap(%)	Time(s)	UB/UB†(%)	Time†(s)	Time(s)
25	0.017	5.27	0	9.513	100.72	14.22	10.278
30	0.001	20.64	0	14.613	99.13	24.77	18.392
40	0.002	113.02	0.005	31.667	99.90	210.61	42.697
50	0	144.26	0	59.222	99.94	1000.00	77.616

Table 2: Computational performance for larger size instances († results reported in [5]).

The conclusion that can be drawn from the comparison of our results with those of Fontes et al. [5] is that, although the quality of the results are quite similar, the computational time required to find such results in our method is better. Furthermore, the magnitude of the computational time improvement seems to grow with problem size.

5 Conclusions

In this paper we presented a Hybrid Genetic Algorithm (HGA) that combines a genetic algorithm and a local search algorithm, to solve the SSU concave MCNFP.

The results obtained have been compared with existing literature and the comparisons have shown the HGA to improve upon the efficiency of existing methods. For all but one of the six hundred problems with sizes ranging from 10 to 19 we were able to obtain an optimal solution. For larger size problems, having from 25 up to 50 vertices, we found the optimal solution for all fixed-charge problems. For concave problems we do not have optimal solutions to compare our solutions with. Thus, they have

[1]Linear problems are only considered to show strong evidence on the fact that the method performance does not depend either on the type of cost functions or on the number of nonlinear arc costs.

been compared with upper bound values reported in literature. The quality of the solutions obtained by the HGA algorithm are quite similar to the ones reported by Fontes et al. [5]. However, the HGA is quicker. Thus, the HGA is capable of efficiently finding heuristic solutions for SSU concave MCNFPs, which are NP-hard problems.

References

[1] J. C. Bean. Genetics and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6:154–160, 1994.

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

[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 new dynamic programming approach for solving single-source uncapacitated concave minimum cost network flow problems. 2003. Submitted.

[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] G. Gallo and C. Sodini. Adjacent extreme flows and application to min concave cost flow problems. *Networks*, 9:95–121, 1979.

[7] D. E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, Reading, Masschusetts, 1989.

[8] G. M. Guisewite. Network problems. In R. Horst and P. M. Pardalos, editors, *Handbook of Global Optimization*, pages 609–648. Kluwer Academic, Dordrecht, 1994.

[9] 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.

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

[11] D. K. Smith and G. A. Walters. Evolutionary approach for finding optimal trees in undirected networks. *European Journal of Operational Research*, 120(3):593–602, 2000.

[12] W. M. Spears and K. A. Dejong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

[13] W. I. Zangwill. Minimum concave cost flows in certain networks. *Management Science*, 14:429–450, 1968.