TÉCNICO LISBOA

Bi-objective Network Flow Optimization Problem

Pedro Miguel Guedelha Dias

Department of Engineering and Management, Instituto Superior Técnico

June 2016

Abstract

Network flow problems, specifically minimum cost network flow problems, can be used to model many real worl problems, even when a network structure does not arise naturally from the problem. Although the single objective case of the minimum cost flow problem is well studied, work for the multi objective case is scarce and many of the proposed algorithms in literature lack an implementation example or a real worl application. In this paper we present an implementation of a solver capable of solving large bi-objective minimum cost flow problems using the bi-objective network simplex algorithm. To better understand the solver performance capabilities, a test package of multiple randomly generated network problems is used and the results of the tests are then presented to the reader along with an analysis of the results.

***Keywords*—** Bi-Objective Network Simplex, Minimum Cost Flow, Multi-objective Optimization

1 Introduction

Minimum cost flow problems are a special type of optimization problems that can be used to model multiple real worl problems that arise in fields like telecommunication, supply chain management, traffic networks and distribution. The methods used to solve these problems bring many new opportunities to decision makers but with the complexity of world, many of these problems are truly cases of multi-objective problems where there is more than a single objective function for the problem. As such, computational algorithms for the solving of multi-objective minimum cost flow problems have a large set of real world applications.

In this paper we present a developed solver application that uses the bi-objective network simplex algorithm to solve bi-objective minimum cost flow problems. The intractability of these problems present a problem for many algorithms that have their usage limited by current computational power and the developed solver, being able to find the efficient solutions for large

sized network problems with hundreds of arcs presents a great opportunity for demonstrating the application of the algorithm for solving these problems.

The paper starts by introducing several common concepts that are fundamental to understand the algorithms for solving the problem. These concepts are both from the field of network flows and from multi-objective optimization. With these concepts established we present a brief literature review where the current work in the field of multi-objective minimum cost flows is introduced. Both the continuous and the discrete case are analyzed in the literature review.

In the last chapter the reader can find more details about the implementation of the solver. Also, in order to understand the size of the network problems that the solver can deal in an acceptable computational time, a test package was created using ramdomly generated networks and from the results we present to the reader a brief analysis of these results.

2 Concepts and Definitions

A multi-objective optimization problem can be formulated as:

$$\text{"min"} \Big(F(x) = (f_1(x), f_2(x), \ldots, f_p(x)\Big)$$
$$\text{s.t. } x \in X$$

where $p \geqslant 2$ is the number of objective functions and the set X is the feasible set of decision vectors [Ehrgott, 2005].

The set of feasible solutions, also called the decision space, is denoted by $X \subset \mathbb{R}^n$ with n equal to the number of decision variables. The objective space is the space generated by the objective functions of the problem and is denoted $f(X) = \{(f_1(x), f_2(x), \ldots, f_p(x)), x \in X\}$.

The concept of *dominance* and *efficiency* are fundamental as typically does not exists a feasible solution that minimizes all the objective functions simultaneously. The importance of the concepts of dominance and efficiency is justified by [Ehrgott, 2005] with observation that any x which is not efficient cannot represent a better alternative to the decision maker because there exists at least one $x' \in X$ such that $f_k(x') \leqslant f_k(x)$ for all $k = 1, \ldots, p$ where strict inequality holds at least once.

Consider the two objective solution vectors $y', y'' \in \mathbb{R}^p$. Then, y' dominates y'' iff $y' \leqslant y''$ and $y' \neq y''$, i.e., $y'_k \leqslant y''_k$ for all $k = 1, \ldots, p$ with at least one strict inequality. Also, iff there is no other solution $y \in Y$ such that $y \leqq y'$ and $y \neq y'$, the solution y is called a non-dominated solution (ND solution).

We can also make the distinction between *dominance* and *strong dominance*. Consider $y, y' \in \mathbb{R}^p$, two objective solution vectors. Then, y strongly dominates y' iff $y < y''$ and $y \neq y'$, i.e., $y_k < y'_k$ for all $k = 1, \ldots, p$.

While the concept of *dominance* is made for the objective space, we can also define the concept of *efficiency* for the decision space. Consider a feasible solution $x' \in X$. The solution is said to be efficient iff it is impossible to find another solution $x \in X$ with a better evaluation of a given objective function without deteriorat-

ing the evaluation of at least another objective, i.e., $x' \in X$ is said to be efficient iff it is impossible to find another solution $x \in X$ such that $y = f(x) \leqq y' = f(x')$ and $y \neq y'$.

While the previous concepts were related with the field of multi-objective optimization, several concepts related to graph theory are also used in the solving of minimum cost network flows. Let $G = (V, E)$ be a directed and connected graph where the set V is a finite set of vertices or nodes with cardinality $|V| = m$ and the set E is a collection of ordered pairs of elements of V called edges or arcs, with cardinality $|E| = n$. Also, let a *path* be a set of alternating arcs and nodes. In a directed graph, all the arcs in any given path must have the same direction. With the concept of path we can now present the definition of *cycle* as a closed path where the only repeated node is the starting and ending point, that is coincident. The concept of cycle is fundamental to the solving of minimum cost flow problems, as is the concept of *tree*. Let a tree $T = (N, A)$ be a subgraph of G that contains no cycle and $N \subseteq V$ and $A \subseteq E$. A tree is called a *spanning tree* when it spans the set of vertices of the graph G, i.e., $N = V$.

Minimum cost flow problems are optimization problems where the objective is to find the best combination of flows through the arcs of a given network in respect to the costs associated with each given arc. Let us consider a network that is a graph $G = (V, E)$ with the set *V* of nodes and the set *E* of arcs. The formulation of the minimum cost flow problem can be stated as

$$\begin{aligned}
&\text{min } f(x) = \sum_{(i,j)\in E} c_{ij}x_{ij} \\
&\text{s.t.} \sum_{j|(i,j)\in E} x_{ij} - \sum_{k|(k,i)\in E} x_{ki} = b_i, \forall i \in V \\
&l_{ij} \leqslant x_{ij} \leqslant u_{ij}, \forall (i,j) \in E
\end{aligned}$$

where the vector c of costs, with c_{ij} for all $(i, j) \in E$, is the vector that gives the costs associated with each arc in the network, the vectors l and u are the lower and upper bound vectors of the network and the vector b is the *supply* of the network, this is, the total amount of flow that is flowig through the arcs of the network.

The bi-objective formulation of the minimum cost flow problem has the same from of the single objective minimum cost flow problem with the addition of a second objective function that used a second vector of costs c^2. The exact formulation of the bi-objective minimum cost flow problem is given as

$$\begin{aligned}
&\text{min } f_1(x) = \sum_{(i,j)\in E} c^1_{ij}x_{ij} \\
&\text{min } f_2(x) = \sum_{(i,j)\in E} c^2_{ij}x_{ij} \\
&\text{s.t.} \sum_{j|(i,j)\in E} x_{ij} - \sum_{k|(k,i)\in E} x_{ki} = b_i, \forall i \in V \\
&l_{ij} \leqslant x_{ij} \leqslant u_{ij}, \forall (i,j) \in E
\end{aligned}$$

3 Literate Review

The reader can now find a review of the scientific literature of the theory and algorithms used in the solving of multi-objective minimum

cost flow problems. We divide the review in two different cases: the continuous case of the multi-objective minimum cost flow problem and the discrete case of the multi-objective minimum cost flow problem.

3.1 Continuous case

The continuous case deals with problems where the objective variables are not restricted to any value that is inside the constraint set. This means that the objective variables can assume as a value any real number.

The authors [Malhotra and Puri, 1984] present a generalization of the out-of-kilter method for solving bi-objective minimum cost flows with uniform capacity for all arcs, with the efficient frontier built in a left-to-right fashion starting with the lexicographical minimum in respect to the first objective function. The set of flows is in-kilter with the first objective function but because the multi-objective problem does not have an optimal point, some of the arcs will be out-of-kilter for the second objective function. These arcs are called *eligible arc set* and for each arc in this set, Pareto cycles are found in the incremental graph of the current flow and added to the current flow. The authors argue that these flows represent efficient solutions for the problem.

The authors [Lee and Pulat, 1991] present a similar algorithm, introducing a revised out-of-kilter method which initially assigns uniform weights on both objective functions and uses the out-of-kilter method to solve the resulting single objective problem. The solution is adjusted in order to be basic and using this solution, the efficient frontier is searched to the left by considering arcs that are out-of-kilter with respect to the first objective function and to the right by considering arcs that are out-of-kilter for the second objective function. To perform a move from one basic flow to another, the selection of the entering arc is made using a ratio of the reduced costs for the two objectives and the arc with the smallest ratio is chosen. The algorithm proposed by [Pulat et al., 1992] is conceptually similar, with the latter using the network simplex algorithm to solve the bi-objective minimum cost flow problem in its parametric formulation.

Also an algorithm is presented by [Sedeño Noda and González-Martín, 2000], based in the previous where the parametric formulation of the bi-objective minimum cost flow problem is solved in a left-to-right fashion using the network simplex method where in each iteration, from a list S of arcs yielding the minimum ratio of the reduced costs one arc is selected to enter the basis tree of the current efficient basic feasible flow. The same authors [Sedeño Noda and González-Martín, 2003] present another method, modifying a method proposed by [Aneja and Nair, 1979]. Instead of exploiting the topological connectivity of the efficient set, applies iteratively the weighted-sum method to the problem.

While the previous methods allowed the computation of the whole efficient frontier, we now introduce methods that present a representative subset of the efficient frontier. All of these methods are applicable only to the bi-objective minimum cost flow problem only and use L and U which "sandwich" the efficient frontier, i.e.,

$Y_n \subseteq ((L + \mathbb{R}^2_{\geqq}) \cap (U + (-\mathbb{R}^2_{\geqq})))$. Articles that use the sandwich method for bi-objective minimum cost flows use the algorithm presented in [Burkard et al., 1991].

The author [Ruhe, 1988] introduces the Hausdorff distance between the upper and lower approximation for measure of the error of the aproximation, which in contrast to [Burkard et al., 1991], does not favor one objective function over the other. The authors [Fruhwirth et al., 1989] introduce two new rules for generating breakpoints, called *angle bisection* and *slope bisection rule* in which error decreases quadratically with the number of breakpoints. The rules can be used to establish an upper bound on the number of evaluations can be given to achieve a specific accuracy level.

the authors [Ruhe and Fruhwirth, 1990] use the sandwich algorithm to compute an ε-optimal approximation for the bi-objective minimum cost flow problem. In this method, a modification to the rule for determining additional breakpoints is applied and instead of solving one minimum cost flow, two problems are solved each iteration.

3.2 Discrete case

The discrete case addresses problems with only two objective functions and make use of two phases in the finding of all integer efficient solutions, with the supported integer efficient flows found in the first phase and the unsupported integer efficient flows in the second phase.

The authors [Lee and Pulat, 1993] present a method which elaborates on the cycle relationship between basic feasible flows and the structure of solutions of bi-objective minimum cost integer flows. The method uses a specific method used to compute candidate points located inside the triangles defined by two consecutive supported non-dominated points.

The claim that the previous algorithm is incorrect because some efficient flows are missed since the algorithm only introduces two non-basic arcs, when more than two are in general needed is made by the authors [Sedeño Noda and González-Martín, 2001] . The authors propose a left-to-right approach starting with the efficient solution that is optimal with respect to objective function c^1.

In [Przybylski et al., 2006], the authors show the incorrectness of the approach suggest by the previous authors. The authors provide an example showing that it is not possible to find all non-dominated objective vectors for bi-objective minimum cost integer flow problems by a straightforward application of the network simplex algorithm and a different adjancy graph is needed.

The author [Figueira, 2002] proposes a branch-and-bound approach to find all non-dominated objective vectors for the problem. For each pair of extreme non-dominated objective vectors that define a triangle in the objective space, the approach allows to find unsupported non-dominated objective vectors that may be located inside the triangle.

In [Lee and Pulat, 1991], the authors extend their own algorithm for determining all the efficient extreme points in the decision space for the continuous case in order to also find all integer efficient flows in the efficient frontier.

4 Methodology

As already stated, the main topic of this paper is the development of an efficient solver capable of solving large bi-objective minimum cost flow problems. In this section the reader can find the description of the algorithm that was used, the details of the implementation of the solver and the results from the test package that was used to find the capabilities of the developed solver.

4.1 Bi-Objective Network Simplex Algorithm

In the single objective network simplex, each basic feasible set is represented by a tree given by the set of basic arcs that have a flow $l_{ij} \leqslant x_{ij} \leqslant u_{ij}$. All the other non-basic arcs have a flow with value of either $x_{ij} = l_{ij}$ or $x_{ij} = u_{ij}$. In each step of the algorithm, a non-basic arc is chosen to enter the basis, resulting in a cycle that can be used to determine the arc that leaves the basis.

In the bi-objective case of the problem, the algorithm starts by finding the optimal solution for one of the objective functions and uses that solution as the initial solution for the multi-objective problem. Since there are two cost components associated with each arc (i, j) in the network, in the network simplex algorithm the reduced costs for each arc also consists of the two components $\bar{c}^1_{ij}$ and $\bar{c}^2_{ij}$.

To find the entering arc, it is computed for each non-basic arc, the ratio of the reduced costs is calculated, with the ratio given by $\mu = \min\{\frac{\bar{c}^2_{ij}}{\bar{c}^1_{ij}} : (i,j) \in L$ with $\bar{c}^2_{ij} < 0$ and $\bar{c}^1_{ij} > 0, \frac{\bar{c}^2_{ij}}{\bar{c}^1_{ij}} : (i,j) \in U$ with $\bar{c}^2_{ij} > 0$ and $\bar{c}^1_{ij} < 0\}$. The non-basic arcs for each the ratio of its reduced costs is equal to the calculated μ are said to be in the set of candidate basic-entering arcs.

The algorithm performs a simplex pivot operation and one of the candidate arcs is removed from the candidate set and enters the basis, with an arc leaving the basis. By performing multiple iterations the algorithm can iteratively find the set of efficient solutions in the efficient frontier, until the optimal solution for the second objective function is reached, that being the criteria for the algorithm to stop.

4.2 Implementation

The solver developed for the bi-objective minimum cost flow problem, using the algorithm described in the previous section, was written in the C programming language and uses the CPLEX C Callable Library that is provided with the IBM CPLEX software. The CPLEX C Callable Library was selected to provide multiple methods and functions that can be used by the solver to solve the single objective part of the larger multi-objective problem. The functions used by the program to solve the multi-objective problem were implemented by the author.

Although the CPLEX library provides several mechanisms to improve single objective optimization, these mechanisms could not be used in the solver. Both the presolver and the aggregator are mechanisms that aggressively try to reduce the size of the problem to improve performance but also destroy the network structure of the problem that is needed by the algorithm.

The solver starts finding the optimal solution for the first objective function using the

weighted-sum of the two objective functions, in order to find an initial solution to the multi-objective problem. This new objective functions is given by $z = w * z_1 + (w - 1) * z_2$, where $w = 0.9999$. The solver also finds the optimal solution for the second objective function as that solution is the stopping criteria for the algorithm.

For every iteration of the algorithm, the solver computes the reduced cost ratio for every non-basic arc using the formulation already present at the previous section. To insert the non-basic arc that was selected as the entering arc, the solver then performs a single simplex pivot operation, requesting that the arc that shall enter the new basis is the previously found arc. This procedure is then repeated for every iteration, with each iteration returning a new efficient solution until the optimal solution for the second objective function is reached.

4.3 Results and Comments

In order to understand the capabilities of the solver described in the previous section, a test package consisting of multiple network flow problems was created. The test package problem instances were created using an altered version of NETGEN to generate bi-objective minimum cost flow problems. The test package consisted of a set of 32 classes that used different parameters. Since we wanted to study the impact of the number of nodes, arcs and the upper capacity in the performance of the solver, only these parameters are different between each test class and all other parameters were kept constant in order to not interfere with the results.

The test package was designed with a growing number of nodes and arcs, starting with an hundred arcs and finishing with 4000 arcs. For each class we generated a set of 30 randomly generated networks with similar parameters. A total of 960 network problems were used to test the solver performance.

The CPU time for the complete set of problems that compose the test package is plotted in figure 1. For the first ten classes of the test package, the solver never required more than 1 second to solve the problem. For the classes from 11 to 26, the solver never used more than 10 seconds of CPU time and only the classes higher than 26 required higher amounts of CPU time, with a sharp increase in the time required by the solver for these problems.

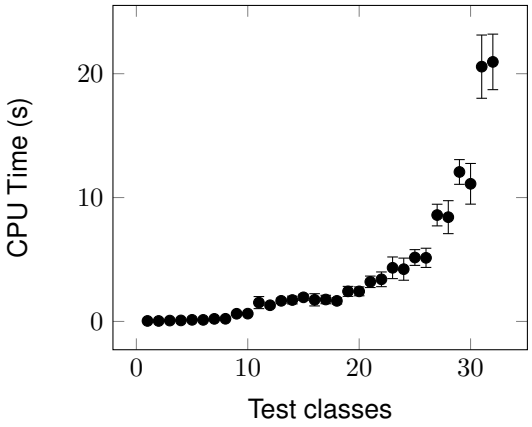

Figure 1: Average CPU time for the 32 test classes

The increasing amount of CPU time can be explained by two different factors: the cost of the computation of the reduced costs ratio for the non-basic arcs and the cost of the simplex pivot operation that inserts a new arc into the basis. As the number of arcs increases, also does the

time required by the solver to compute all the ratios as the calculation implies a linear scan of all the arcs of the network. Also, the cost for the insertion of the entering arc also increases as the higher amount of arcs implies greater computation needed to find the cycle and determine the leaving arc.

Since we also wanted to study the amount of solutions found by the solver, the average number of solutions was calculated for each class and the results are presented in figure 2. The behaviour of this variable is similar to the behaviour of the CPU time, with an increasing amount of solutions found by the solver for test classes that had higher number of arcs. The reader can also take note that for classes in range 9 to 12, a much larger of solutions was found than for classes of similar size. But while the number of solutions found is higher, figure 1 shows that the CPU time for these classes was similar to the results of similar classes.

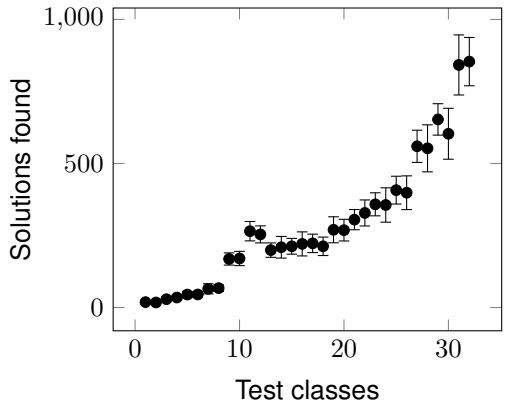

Figure 2: Average number of solutions found for the 32 test classes

Finally, the average number of solutions and the average CPU time are shown in figure 3. The figure shows that the increase of the number of solutions also implies an increase in the CPU time for the problem. Since for each iteration, the algorithm returns a new solution, and larger networks yield larger number of solutions, the total cost in CPU time tends to increase because more solutions imply more iterations and, as already explained, larger networks with more arcs require more CPU time for each iteration.

If the computational time required for each iteration is decreased, even for large problems that have an high number of solutions, the total time required by the solver for these problems will also decrease. Decreasing the computational time cost of each iteration of the algorithm could be a topic for future work.

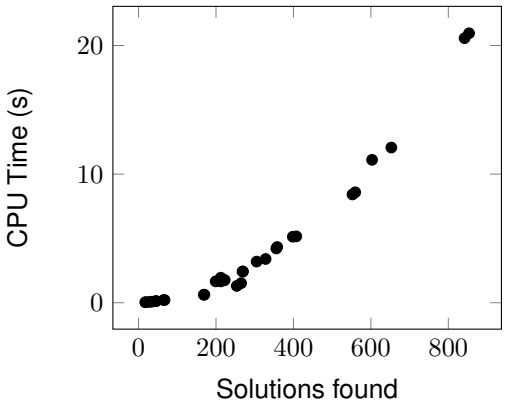

Figure 3: Average number of solutions versus average CPU time for the 32 test classes

5 Conclusions

In this paper it was presented an implementation of the bi-objective network simplex algorithm for solving the bi-objective minimum cost flow problem. The reader is introduced to the

fundamental concepts of multi-objective optimization and network flows and a short literature review of the current work for multi-objective minimum cost flow problems is presented to the reader. The bi-objective network simplex is also described in greater detail.

The solver that was developed uses this algorithm to build the efficient frontier of the problem and was shown to be able to solve network problems having an high number of arcs, with the larger networks that make up the test package having 4000 arcs and 2000 nodes while not requiring more than 26 seconds for these networks.

References

[Aneja and Nair, 1979] Aneja, Y. and Nair, K. (1979). Bicriteria transportation problem. *Management Science*, 25:73–78.

[Burkard et al., 1991] Burkard, R., Hamacher, H., and Rote, G. (1991). Sandwich approximation of the univariate convex functions with an application to separable convex programming. *Naval Research Logistics Quarterly*, 38(6):911–924.

[Ehrgott, 2005] Ehrgott, M. (2005). *Multicriteria Optimization*. Springer, second edition.

[Figueira, 2002] Figueira, J. R. (2002). On the integer bi-criteria network flow problem: A branch-and-bound approach. *Cahier du LAMSADE*, 191.

[Fruhwirth et al., 1989] Fruhwirth, B., Burkard, R., and Rote, G. (1989). Approximation of convex curves with application to the bicriterial minimum cost flow problem. *European Journal of Operational Research*, 42:326–338.

[Lee and Pulat, 1991] Lee, H. and Pulat, P. (1991). Bicriteria network flow problems: Continuous case. *European Journal of Operations Research*, 51:119–126.

[Lee and Pulat, 1993] Lee, H. and Pulat, P. (1993). Bicriteria network flow problems: Integer case. *European Journal of Operational Research*, 66:148–157.

[Malhotra and Puri, 1984] Malhotra, R. and Puri, M. (1984). Bi-criteria network problems. *Cahiers du Centre d' Études Recherche Opérationelle*, 1(26):95–102.

[Przybylski et al., 2006] Przybylski, A., Gandibleux, X., and Ehrgott, M. (2006). The biojective integer minimum cost flow problem - incorrectness of sedeño-noda and gonzález-martín algorithm. *Computers and Operations Research*, 33:1459–1463.

[Pulat et al., 1992] Pulat, P., Huarng, F., and Lee, H. (1992). Efficient solutions for the bicriteria network flow problem. *Computers and Operations Research*, 19:649–655.

[Ruhe, 1988] Ruhe, G. (1988). Complexity results for multicriterial and parametric network flows using a pathological graph of zadeh. *Zeitschrift fur Operations Research*, 32:9–27.

[Ruhe and Fruhwirth, 1990] Ruhe, G. and Fruhwirth, B. (1990). ϵ-optimality for bicriteria programs and its application to minimum cost flows. *Computing*, 44:21–34.

[Sedeño Noda and González-Martín, 2000] Sedeño Noda, A. and González-Martín, C. (2000). The biobjective minimum cost flow problem. *European Journal of Operations Research*, 124:591–600.

[Sedeño Noda and González-Martín, 2001] Sedeño Noda, A. and González-Martín, C. (2001). An algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 28:139–156.

[Sedeño Noda and González-Martín, 2003] Sedeño Noda, A. and González-Martín, C. (2003). An alternative method to solve the biobjective minimum cost flow problem. *Asia-Pacific Journal of Operations Research*, 20:241–260.