SOLVING SCALARIZED MULTI-OBJECTIVE NETWORK FLOW PROBLEMS WITH AN INTERIOR POINT METHOD

MARGARIDA FONSECA, JOSÉ RUI FIGUEIRA, AND MAURICIO G.C. RESENDE

Abstract. In this paper we present a primal-dual interior-point algorithm to solve a class of multi-objective network flow problems. More precisely, our algorithm is an extension of the single-objective primal infeasible dual feasible inexact interior point method for multi-objective linear network flow problems. Our algorithm is contrasted with standard interior point methods and experimental results on bi-objective instances are reported. The multi-objective instances are converted into single objective problems with the aid of an achievement function, which is particularly adequate for interactive decision-making methods.

1. Introduction

Multi-objective optimization is a branch of mathematical programming dealing with decision-making problems characterized by multiple and conflicting objectives that are to be optimized over a feasible set of decisions or solutions. These problems are important since many real-world applications are by their very nature multi-objective. Such problems are referred to as multi-objective or multiple objective programs, and they are commonly encountered in many areas of human activity, including engineering, management, medicine, biology, and transportation. In these problems, feasible solutions are only described implicitly, through a set of constraints in the form of mathematical functions (equalities and/or inequalities). An optimization problem has to be solved to find the "best" solution or solutions. The main goal of multi-objective mathematical programming is to seek the solutions of multi-objective programs. In such problems, however, there is no single optimal solution (or set of optimal solutions) as is the case when dealing with single objective problems. The notion of *optimal solution* gives place to the concept of *non-dominated solutions*. Non-dominated solutions are characterized by the fact that when moving from one to another, one cannot improve the performance of all of the objectives without the degradation of at least the performance of one objective. Consequently, methods suitable for finding these solutions are considered the most fundamental tools for dealing with multi-objective mathematical problems. For example, see Steuer (1986) and Figueira et al. (2005).

Most of the algorithms for multi-objective linear problems are based on variants of the simplex method for linear programming despite the good performance of interior point methods in solving large scale problems. In fact, interior point methods are not easy to adapt for multi-objective programs since they construct a

Date: Janaury 12, 2009. Revised December 4, 2009.

Key words and phrases. Interior point methods, multi-objective network flows, achievement functions, decision-making .

AT&T Labs Research Technical Report.

sequence of points that converges to a single point on the boundary of the feasible set. Thus, there are few interior point algorithms proposed in the literature for multi-objective problems, and most of them are of an interactive nature. Although the pivoting mechanism to move from one facet to another on the efficient or non-dominated surface, required by simplex-based algorithms, may not pose any special computational difficulty, the number of required operations increases rapidly when the size of the problem increases, making simplex-based algorithms inefficient on large-scale problems. One would expect interior-point-based algorithms to better suited for this class of problems. Moreover, as mentioned by Arbel (1997), the use of interior point algorithms allows new ways of interaction with the decision-maker (DM) during the preference elicitation phase inherent to interactive multi-objective linear problems. Interactive multi-objective optimization methods require interaction with the DM to adapt the progress of the solution process to reflect the local preferences of the DM. All of these methods share one common feature: at each iteration a single objective linear programming problem is formulated to generate a candidate solution or a set of candidate solutions for the examination by the DM. Each of these single objective problems can be solved by a single objective linear programming algorithm.

Interesting applications of interior point algorithms for multi-objective linear problems have been developed (Arbel, 1993; 1994a;b; 1995; 1997; Arbel and Korhonen, 1996a;b; 2001; Arbel and Oren, 1993; 1996; Trafalis et al., 1990; Aghezzaf and Ouaderhman, 2001). The first attempts at using interior point single objective algorithms for multi-objective problems were reported in Arbel (1993; 1994a;b). These approaches used two distinct variants to generate an interior sequence of iterates. Arbel (1993) presents a multi-objective linear programming algorithm based on a variant of Karmarkar's interior point algorithm known as the primal affine-scaling algorithm. The proposed algorithm goes through a series of iterations during which interior step direction vectors are generated to optimize each of the objective functions under consideration. The algorithm developed there and in Arbel (1997) and Arbel and Oren (1993) generates step direction vectors according to the single objective interior algorithm and uses them to yield a single step direction vector for the multi-objective linear programming problem algorithm along which one moves from the current iterate to the next. Motivated by the numerical experiments with single objective linear programming problems that showed primal-dual algorithm to generally perform better than primal algorithm, Arbel extended earlier work (Arbel, 1994b; 1995; 1997), modifying path-following primal-dual algorithms for linear programming problems to solve multi-objective problems according to the ideas developed in his previous studies. The proposed algorithms require interaction with a DM to obtain locally-relevant preference information for the interior directions generated at each step of the iterative process.

The use of scalarizing achievement functions together with aspiration levels, the so-called reference point approach, was first proposed by Wierzbicki (1980; 1986). This approach uses reference points as aspiration levels and provides more flexibility in searching solutions on the non-dominated frontier. It is considered to be one of the best approaches to deal with interactive decision-making and gained some popularity in the multi-objective mathematical programming community. Many authors use reference point based methods as the main tool in their interactive methods. Korhonen and Laakso (1986) show how an unbounded line segment

emanating from the current point in the reference direction can be projected onto the (weakly) non-dominated set. Korhonen and Wallenius (1988) further developed this idea making the specification of the reference direction dynamic. Arbel and Korhonen (1996a;b) combine these ideas and develop an interactive approach which is controlled through aspiration levels. They use one of two interior point variants: a primal variant, the primal affine-scaling algorithm, and a primal-dual variant, the primal-dual path-following algorithm. At each iteration, a DM is asked to specify aspiration levels for the various objectives, and an achievement scalarizing problem is defined to project the aspiration levels onto the non-dominated set. This formulation leads to a single objective linear programming model. An interior point algorithm is then used to draw a path from a starting solution and approach as closely as desired a non-dominated solution corresponding to the optimum of the achievement scalarizing problem. The DM can specify aspiration levels during the solution process and thus steer the interaction solution path toward different areas in the objective space. The interaction with the DM can take place every few, pre-specified, iterations or after the duality gap achieved for the stated aspirations has fallen below a certain threshold.

The above algorithms fall into the class of feasible interior point algorithms, which do not require a feasible starting solution for the interior point algorithm, but an exact determination of the search direction taken in each interior point iteration. This leads to especially hard numerical problems to overcome when the size of the problems increases and when a more precise solution is required. Despite the above cited papers, to the best of our knowledge, no efficient practical implementation has been described to allow the solution of medium- or large-scale multi-objective problems. Moreover, no implementation has been described of an interior point algorithm to solve a special class of multi-objective linear programming problems such as multi-criteria network flow problems, despite their richness of real-world applications.

The purpose of this paper is to present a primal-dual interior point algorithm to solve a class of multi-objective network flow problems and to test this implementation on large instances.

In this paper, by the term "solving multi-objective network flows", we mean that we consider the solution of a Tchebychev-like problem that can further be used in an interactive multi-objective framework (see Steuer (1986) and Figueira et al. (2005)). The Tchebychev transformation is a well-known technique, but we are unaware of any effective implementation to date of interior point methods to solve network flow problems. In this paper we want to investigate the use of interior point methods to solve this type of problem. We propose an extension of the primal-dual interior point method (Kojima et al., 1989) to the infeasible (Kojima et al., 1993) and inexact (Freund et al., 1999) cases. The proposed algorithm, begins with a dual-feasible primal-infeasible starting solution and, at each interior point iteration, the search direction is found by solving the normal equations resulting from the Newton system. Those equations are in the form of a positive definite system with two dense matrix rows. We solve those reduced systems using one of two methods, a direct method based on the Cholesky decomposition of the system matrix or an iterative method based on the pre-conditioned conjugate gradient method with diagonal pre-conditioner (`pcg`). We test the performance of these two methods using instances of bi-objective network flow problems generated with an adaptation of the NETGEN

generator (Klingman et al., 1974) for single objective network flow problems to network flows instances with two objectives. The sizes of the test problems vary from 5 to 1500 nodes and from 7 to 10,000 arcs. We observed a good performance of the variant which applies the `pcg` method to find the search directions. Moreover, in our experiments we used different "weights" for the objectives and different reference points, including the ideal point and an approximation of the nadir point. The algorithm appears to be insensitive to changes in the reference point.

The paper is organized as follow. Section 2 presents the concepts, their definitions, and notation. Section 3 describes the primal-dual algorithm used to solve the multi-objective network flow problem. Section 4 provides discussion of some implementation issues like starting point, stopping criteria for the interior point algorithm, and a reduction of Newton system to the normal equations, and is devoted to the main procedures used in the solution of the normal equations. Section 5 is dedicated to the computational experiments and results. Finally, conclusions are made in Section 6.

2. Concepts: Definitions and Notation

Consider a directed graph $G = (V, E)$, where V is a set of m vertices and E represents a set of n edges; let (i, j) denote the arc from vertex i to vertex j. The multi-objective "minimum cost" network flow problem, with p objectives, can be formulated as

$$
\begin{aligned}
&\min && \sum_{(i,j)\in E} {c^1}_{ij} x_{ij} \\
&\min && \sum_{(i,j)\in E} {c^2}_{ij} x_{ij} \\
&\vdots && \vdots \\
&\min && \sum_{(i,j)\in E} {c^p}_{ij} x_{ij}
\end{aligned} \tag{1}
$$

subject to:

$$\sum_{(i,j)\in E} x_{ij} - \sum_{(j,i)\in E} x_{ji} = b_i, \quad \forall i \in V \tag{2}$$

$$l_{ij} \leq x_{ij} \leq u_{ij}, \quad \forall (i,j) \in E. \tag{3}$$

In this formulation, x_{ij} denotes the flow on arc (i, j) and ${c^q}_{ij}$, $q = 1, ..., p$, are the costs to transport one unit of flow on arc (i, j). For each $i \in V$, b_i denotes the flow produced or consumed at vertex i. If $b_i > 0$, vertex i is called a source vertex. If $b_i < 0$, vertex i is called a sink vertex. Otherwise (i.e. $b_i = 0$), vertex i is called a transshipment vertex. For each arc $(i, j) \in E$, u_{ij} (l_{ij}) denotes the upper (lower) bound on the flow on arc (i, j). Most often, data are assumed to be integer, and many implementations of network flow algorithms adopt this assumption. However, there can exist applications where the data are real numbers, and algorithms should ideally handle problems with real data. Constraints of type (2) are referred to as the flow conservation equations, while constraints of type (3) are called the flow capacity constraints. In the remainder of this paper we assume, without loss of generality, a lower bound $l_{ij} = 0$ for all arcs $(i, j) \in E$. A simple change of variables can transform the corresponding problem with upper and lower bounds

into an equivalent one only having lower bounds, $l_{ij} = 0$, for all $(i,j) \in E$. In matrix notation, the problem can be stated as

$$\text{“min”} \quad \{Cx : \quad Ax = b, x + s = u, x \geq 0\}, \tag{4}$$

where A is the $m \times n$ vertex-arc incidence matrix of graph G=(V,E), vectors x, s, and u are real n-vectors, and b is a demand/supply m-dimensional real vector. The dense real matrix C with the p cost vectors with respect to the n arcs $(i,j) \in E$, denoted by one index i, $i = 1, \ldots, n$, is defined as

$$C = \begin{pmatrix} {c^1}^{\mathrm{T}} \\ {c^2}^{\mathrm{T}} \\ \vdots \\ {c^p}^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} {c^1}_1 & {c^1}_2 & \ldots & {c^1}_n \\ {c^2}_1 & {c^2}_2 & \ldots & {c^2}_n \\ \vdots & \vdots & & \vdots \\ {c^p}_1 & {c^p}_2 & \ldots & {c^p}_n \end{pmatrix}.$$

Scalarizing functions, such as Tchebychev-like augmented weighted functions, can be used to convert the multi-objective problem into one with a single-objective. Scalarizing weighted and augmented Tchebychev-like problems as well as more general functions, such as the achievement functions presented in Wierzbicki (1980; 1986), have many interesting properties. One of the families[1] of achievement functions, particularly well adapted to work with reference point approaches is

$$\begin{array}{l} \rho(Cx, \pi^0, \overline{\lambda}, \sigma) = \min_{q=1,2,\ldots,p} \left\{ \overline{\lambda}_q (\pi^0_q - {c^q}^T x) \right\} + \\ \sigma \sum_{q=1}^{p} \overline{\lambda}_q (\pi^0_q - {c^q}^T x) \end{array} \tag{5}$$

where $\rho(\cdot,\cdot,\cdot,\cdot)$ is an application of the feasible set of solutions in the objective space into $\mathbb{R}$; $\overline{\lambda}_1, \overline{\lambda}_2, \ldots, \overline{\lambda}_p$ are the non-negative “weights” associated with each of the p objectives so that $\sum_{q=1}^{p} \overline{\lambda}_q = 1$; $\pi^0 = \begin{pmatrix} \pi^0_1 & \pi^0_2 & \ldots & \pi^0_p \end{pmatrix}^{\mathrm{T}}$ is the vector which represents the reference point considered in the objective space and σ is an arbitrary small positive number ($0 < \sigma \ll 1$).

The achievement problem

$$\begin{array}{rl} \max & \rho(Cx, \pi^0, \overline{\lambda}, \sigma) \\ \text{subject to:} & x \in X, \end{array} \tag{6}$$

can be formulated, where X is the set of feasible solutions in the decision space. For a given reference point π^0, two fundamental properties can be proved:

i) If $x^\star = \text{argmax} \{\rho(Cx, \pi^0, \overline{\lambda}, \sigma)$ subject to $x \in X\}$, then $x^\star$ is an efficient solution.

ii) If $x^\star$ is an efficient solution, then there exists a function $\rho(z, \pi^0, \overline{\lambda}, \sigma)$ such that $x^\star$ is a (global) optimal solution of $\max \rho(Cx, \pi^0, \overline{\lambda}, \sigma)$ subject to $x \in X$.

To convert this single objective non-linear problem into one that is linear, a free variable α needs to be introduced. If we assume that the reference point is the *ideal point* (the one corresponding to the minimum of each objective), α needs to be positive. Later we consider the adaptation of the problem to the case where the reference point is any point (in general, a point that represents an aspiration level for the DM) in the objective space, and thus α will be a free variable. We use the

[1]The word *family* is used here to state that several functions can be built according to the variability of “weighted” coefficients and reference points.

common approach of decomposing the free variable α by $\alpha = \alpha^+ - \alpha^-$. Using the above notation, we can then formulate (4) as either the single-objective program

$$\begin{aligned} &\min && \alpha + \sigma \textstyle\sum_{q=1}^{p} c^q x - \sigma \sum_{q=1}^{p} \pi_q \qquad (7)\\ &\text{subject to:} \end{aligned}$$

$$\begin{aligned} &\alpha \geq \overline{\lambda}_q({c^q}^{\mathrm{T}} x - \pi_q), \quad q = 1, \ldots, p\\ &Ax = b\\ &x + s = u\\ &x \geq 0, \quad s \geq 0, \quad \alpha \geq 0 \end{aligned} \qquad (8)$$

or

$$\begin{aligned} &\min && \alpha + \sigma \textstyle\sum_{q=1}^{p} c^q x - \sigma \sum_{q=1}^{q} \pi_q \qquad (9)\\ &\text{subject to:} \end{aligned}$$

$$\begin{aligned} &{c^q}^{\mathrm{T}} x - \lambda_q \alpha + r_q = \pi_q, \quad q = 1, \ldots, p\\ &Ax = b\\ &x + s = u\\ &x \geq 0, \quad s \geq 0, \quad \alpha \geq 0 \end{aligned} \qquad (10)$$

where r is a slack p-vector and $\lambda_q = 1/\overline{\lambda}_q$, with $q = 1, \ldots, p$. Let us assume for the moment that our reference point is the ideal one. In this case, $\alpha \geq 0$.

Formulation (9-10) is presented for subsequent development through the application of the interior point algorithm. Of course, the fact that many programs can be written in the form of linear programs with equality and non-negativity constraints does not necessarily imply that these programs are algebraically similar. Furthermore, the strategies followed in the application of the interior point algorithm should not necessarily be the same, since the constraint matrix as well as the algebraic structures can be quite different. The constraint which is present in (9-10), associated with the cost matrix C, as well as with the relative "weights" related to the objective functions, and to the chosen reference point, introduce numerical difficulties which are dealt with in the implementation of the interior point algorithm. A primal-dual method is proposed to solve this program. The corresponding dual problem is

$$\max b^{\mathrm{T}} y - \Pi^{\mathrm{T}} v - u^{\mathrm{T}} w \qquad (11)$$

subject to

$$A^{\mathrm{T}} y - C^{\mathrm{T}} v - w + z = \sigma \sum_{q=1}^{p} c^q \qquad (12)$$

$$\lambda_q^{\mathrm{T}} v + z_\alpha = 1 \qquad (13)$$

$$v \geq 0, \quad w \geq 0, \quad z \geq 0, \quad z_\alpha \geq 0. \qquad (14)$$

A vector $\overline{x}$ is a solution of problem (9-10) if the optimality conditions

$$
\begin{array}{l}
Ax = b \\
x + s = u \\
Cx - \alpha\lambda + r = \Pi \\
A^{\mathrm{T}}y - C^{\mathrm{T}}v - w + z = \rho \sum_{q=1}^{p} c^q \\
\lambda^{\mathrm{T}}v + z_\alpha = 1 \\
x^{\mathrm{T}}z = 0, \quad r^{\mathrm{T}}v = 0, \quad s^{\mathrm{T}}w = 0, \quad \alpha z_\alpha = 0 \\
x \geq 0, \alpha \geq 0, r \geq 0, s \geq 0, \\
v \geq 0, w \geq 0, z \geq 0, z_\alpha \geq 0
\end{array}
\tag{15}
$$

are satisfied.

The solution of the achievement problems presented in this section is important when dealing with interactive procedures. The DM can provide an aspiration level, treated as a reference point, and a non-dominated solution will be obtained after solving the linear network flow problem stated above.

3. A truncated primal-infeasible dual-feasible interior-point algorithm

Primal-dual interior-point algorithms (for example, see Monteiro and Adler (1989), Kojima et al. (1989), Megiddo (1989), and Tanabe (1988)) have been supported by many sophisticated codes to solve large scale linear problems, including network flow problems. They have suitable theoretical properties and were shown to have good performance in practice. A feasible primal-dual algorithm operates simultaneously on the primal and dual problems. It requires feasible interior primal and dual starting solutions and iterates in the interior of the feasible region of the primal-dual pair of problems. It was proved that the iterates generated by the feasible primal-dual algorithm are feasible if the step lengths in the primal and the dual spaces of the interior point algorithm are appropriately chosen and the direction components are computed exactly (for example, see Kojima et al. (1989)). Usually, practical implementations of a feasible primal-dual algorithm are slightly different from their corresponding theoretical polynomial-time variant. The main differences at each iteration are related to the update procedure for the central parameter and the selection of the interior step lengths, which are not easy to compute directly from theory.

A major drawback of feasible algorithms is that they require the solution of a Newton system with a high degree of accuracy, in general, computed through direct factorization. Moreover, obtaining a feasible starting point can be very expensive. Although the better theoretical properties of primal-dual interior point algorithms are reached when feasible and exact variants of these algorithms are used, many practical implementations of primal-dual methods only use infeasible and/or inexact variants.

These drawbacks are significant when solving large scale network flow problems. One of the first implementations of a primal-dual feasible interior point method for network flow problems is due to Portugal et al. (1996). The authors investigated the use of an iterative solver to obtain the interior search direction. Based on the computational results by Portugal et al. (1996) and on the conclusions of their paper, Portugal et al. (2000) proposed a primal-dual interior point algorithm, that

is primal infeasible and dual feasible, called the *truncated primal-infeasible dual-feasible* (TPIDF) algorithm. In addition, the algorithm works with inexact search directions. It was applied to solve network flow problems, and the `PDNET` code (which implements the TRIDF algorithm) is considered today to be a competitive code for solving large-scale network flow problems (Portugal et al., 2008).

Kojima et al. (1993) proved that the primal-dual infeasible-interior-point method converges if the interior step lengths are computed by a specific rule they proposed. This algorithm was shown to have several practical implementable variants. A primal-dual infeasible-interior-point algorithm starts with an infeasible-interior point, that is, an initial point whose non-negative variables are strictly positive but the primal and dual residuals are not necessarily zero. In that case, the generated sequence of iterates is not restricted to the interior of the feasible region. Consequently, optimal solutions are approached by moving through not only the interior but also the exterior of the feasible region of the primal-dual pair of problems. Despite the simplicity of these methods, the theoretical constraints needed to control the central parameter and step lengths are not practical or easy to implement for large scale problems. Consequently, most implementations of infeasible primal-dual algorithms relax these constraints in one way or another. The possibility of using arbitrary starting points and long step sizes that can be different in the primal and dual spaces are not the only explanation for the popularity of infeasible primal-dual algorithms. Perhaps not less important is the fact that, because of its simple structure, primal-dual algorithms can easily be modified to handle inexact search directions. The use of inexact search directions is a major difference of most interior-point algorithms (feasible or infeasible) whose convergence is proved under the assumption that the search directions are calculated exactly. The so-called inexact or truncated interior point methods are those where the search directions are calculated only to moderate accuracy. Since inexact interior point algorithms follow the central path in a less rigorous way than *exact* ones, an increase in the number of iterations is expected. However, the use of inexact search directions can nevertheless result in a decrease in the total processing time, because the inexact search directions can sometimes be calculated very efficiently. Algorithms featuring similar search directions for network flow problems were proposed by Resende and Veiga (1993) and Portugal et al. (2000). Also, in standard linear programming problems, inexact directions are studied by Freund and Jarre (1997), Oliveira and Sorensen (1997), Freund et al. (1999), Mizuno and Jarre (1999), Baryamureeba et al. (1999), Korzak (2000), and Monteiro and O'Neal (2003).

Several authors have used iterative linear solvers to compute Newton search directions. For example, Resende and Veiga (1993) and Oliveira and Sorensen (1997) used the pre-conditioned conjugate gradient method in conjunction with some specific pre-conditioners to solve the normal systems of equations, to calculate the interior search direction. Their computational results show that iterative interior point methods can be extremely useful in practice.

Korzak (2000) presents a convergence analysis of inexact variants of the Kojima et al. (1993) infeasible algorithm for linear programming, where the iterates are bounded, and shows that the (polynomial) convergence of such variants can be proved in almost the same way as the convergence of the (exact) original algorithm by Kojima et al. (1993). Contrary to what occurs with the infeasible exact algorithm, when infeasible problems are processed, the iterates are unbounded and the

infeasibility of the given problems cannot be proved. The convergence analysis of inexact methods for linear programming usually assumes that no primal or dual feasible starting solutions are known, and it is based on the convergence rules of Kojima et al. (1993). However, standard network flow problems and standard network flow problems with multiple objectives, have the nice property that it is easy to compute a feasible starting dual solution. In those particular cases, Freund et al. (1999) recommend the use the algorithm of Portugal et al. (2000).

In the implementation of Portugal et al. (2000; 2008), strategies are designed to explore the very nature of network flow problems as a particular way of defining a starting solution and two network-specific stopping criteria for the interior point algorithm are proposed. Although they do not use the theoretical rules of convergence of Kojima et al. (1993), the pre-conditioning techniques used in the iterative method applied to solve the system of normal equations, as well as other techniques related to algebraic efficient procedures to manipulate network matrices, allow us to observe in practice the efficiency of their algorithm. More recently, Monteiro and O'Neal (2003), obtained some complexity properties for the modified versions of the long-step primal-dual infeasible algorithms for solving linear programs, where the search directions are computed by means of an iterative linear solver applied to a pre-conditioner normal system of equations, with a *maximum weight basis* preconditioner. For network flows these correspond to a class of pre-conditioners based on spanning trees. Such pre-conditioners were introduced by Resende and Veiga (1993) in the context of the minimum cost network flow problem, and later generalized by Oliveira and Sorensen (1997) for general linear programming problems. They make the condition number of the preconditioned normal equations system uniformly bounded regardless of the values of the diagonal elements of the scaled matrix. Nice complexity properties can be reached with an iterative variant of the infeasible primal-dual interior point method of Kojima et al. (1993) if that class of pre-conditioners is used together with a careful selection of the stopping criterion of the iterative solver. In the Monteiro and O'Neal (2003) approach, as well as in the approach of Portugal et al. (2000), only the search direction components calculated by the normal equations are computed approximately. The remaining components of the search direction are calculated in a way that the equations of the Newton system that correspond to primal and dual feasibilities are satisfied exactly, while the equation of the Newton system corresponding to the centrality condition is violated. This way of choosing the search direction is crucial to establish that the number of outer iterations of the proposed methods are polynomially bounded.

For the above reasons, we choose to use the TPIDF algorithm to solve (9).

Consider the sets

$$R_p = \{(x, \alpha, r, s) \in \mathcal{R}^{n+1+p+n} : x \geq 0, \quad \alpha \geq 0, \quad r \geq 0, s \geq 0\}, \tag{16}$$

$$R_d = \{(y, z, z_\alpha, v, w) \in \mathcal{R}^{m+n+1+p+n} : z \geq 0, \quad z_\alpha \geq 0, \quad v \geq 0, \quad w \geq 0\}, \tag{17}$$

$$\overline{R} = \{(x, \alpha, r, s, y, z, z_\alpha, v, w) : (x, \alpha, r, s) \in R_p, (y, z, z_\alpha, v, w) \in R_d\}, \tag{18}$$

and $R_{p_+}, R_{d+}, \overline{R}_+$, the interior of R_p, R_d, and $\overline{R}$. The primal feasible set $\mathcal{F}_p$ and the dual feasible set $\mathcal{F}_d$ of (9-10) are defined, respectively, as

$$\mathcal{F}_p = \{(x, \alpha, r, s) \in R_{p_+} : Ax = b, \quad x + s = u,\ Cx - \alpha\lambda + r = \pi\},$$

$$\mathcal{F}_d = \{(y, z, z_\alpha, v, w) R_{d+} : A^{\mathrm{T}} y - C^T v - w + z = \rho \sum_{q=1}^{p} c_q^{\mathrm{T}}, \lambda^{\mathrm{T}} v + z_\alpha = 1\}.$$

In network flow problems, the primal condition $Ax = b$ is the most difficult to satisfy. Consider the subset $\tilde{\mathcal{F}}_p$ of the primal feasible set defined by

$$\tilde{\mathcal{F}}_p = \{(x, \alpha, r, s) \in R_{p_+} : x + s = u,\ Cx - \alpha\lambda + r = \pi\}.$$

A primal-infeasible dual-feasible algorithm operates on the dual feasible set $\mathcal{F}_d$ and on the partial primal feasible set $\tilde{\mathcal{F}}_p$, i.e. on the set $\tilde{\mathcal{F}}$ defined by

$$\tilde{\mathcal{F}} = \{(x, \alpha, r, s, y, z, z_\alpha, v, w) \in \overline{R}_+ : (x, \alpha, r, s) \in \tilde{\mathcal{F}}_p, (y, z, z_\alpha, v, w) \in \mathcal{F}_d\}.$$

When an infeasible primal-dual algorithm is used, the primal feasibility residuals ξ_1, ξ_2, ξ_3, the dual feasibility residuals ζ_1, ζ_2, and the complementary feasibility residuals $\varsigma_1, \varsigma_2, \varsigma_3, \varsigma_4$, in each iteration k, can be defined as

$$\xi_1^k = u - x^k - s^k \tag{19}$$

$$\xi_2^k = -Ax^k + b \tag{20}$$

$$\xi_3^k = \Pi - Cx^k + \lambda\alpha^k - r^k \tag{21}$$

$${\zeta_1}^k = \sigma \sum_{k=1}^{p} c_k^{\mathrm{T}} - A^{\mathrm{T}} y^k + C^{\mathrm{T}} v^k + w^k - z^k \tag{22}$$

$$\zeta_2^k = 1 - \lambda^{\mathrm{T}} v^k - z_\alpha^k \tag{23}$$

$$\varsigma_1^k = \mu^k e - X^k Z^k e \tag{24}$$

$$\varsigma_2^k = \mu^k - z_\alpha^k \alpha^k \tag{25}$$

$$\varsigma_3^k = \mu^k e - R^k V^k e \tag{26}$$

$$\varsigma_4^k = \mu^k e - S^k W^k e. \tag{27}$$

These residuals are possibly not null. However, if the algorithm is dual feasible, then the dual feasibility residuals ${\zeta_1}^k$ and ${\zeta_2}^k$ are null at each iteration. Moreover, it is possible to force ${\xi_1}^k$ to be null in each iteration k.

In spite of the fact that single objective and multi-objective network flow problems formulated as linear programs with the same form, each is algebraically processed by interior point algorithms in a different way. The case of the multi-objective network flow problem introduces particular numerical difficulties. The major difficulty arises from the constraint matrix. It is not simply a node-arc adjacency matrix but assumes a more complex structure and is numerically more unstable. It was necessary to develop appropriate strategies for which the implementation of the interior point algorithm proposed by Portugal et al. (2000) for single objective network flows could be adapted to solve multi-objective problems. Thus, the interior point algorithm implemented is specialized for network flow problems with multiple objectives and not only for single objective problems, since the structure of the matrices involved between the network flow problems constraints with multiple objectives and the adjacency matrices of single objective are significantly different.

The TPIDF algorithm applied to (9 - 10) can then be described as follows.

TPIDF-MNFP - Truncated primal-infeasible dual-feasible interior-point algorithm applied to scalarized multi-objective linear network flow problem:

(1) Take an arbitrary point

$$(x^0, \alpha^0, r^0, s^0, y^0, z^0, z_\alpha^0, v^0, w^0) \in \widetilde{\mathcal{F}}.$$

(2) For each $k = 0, 2, \ldots,$ do

(a) Set

$$\varpi^k = (x^k)^{\mathrm{T}} z^k + \alpha^k {z_\alpha}^k + (r^k)^{\mathrm{T}} v^k + (s^k)^{\mathrm{T}} w^k$$

and

$$\mu^k = \beta_1 \frac{\varpi^k}{2n+p}$$

where $0 < \beta_1 < 1$. In a practical implementation we use $\beta_1 = 0.1$.

(b) Compute the primal residuals ξ_1^k, ξ_2^k, ξ_3^k, the dual residuals ζ_1^k, ζ_2^k, and the complementarity residuals ς_1^k, ς_2^k, ς_3^k, ς_4^k, with the formulae (19-21), (22-23), and (24-27), respectively. If ξ_1^k, ζ_1^k, and ζ_2^k are not null, then STOP: send an error message.

(c) Compute the auxiliary diagonal matrix

$$F^k = (Z^k S^k + W^k X^k)$$

and auxiliary vector

$$\chi^k = (F^k)^{-1}[XS(z-w) + \mu(x-s)].$$

Compute the scaling matrix

$$\Theta^k = ((X^k)^{-1} Z^k + (S^k)^{-1} W^k)^{-1}$$

as

$$\Theta^k = X^k S^k (F^k)^{-1}$$

Find $(\Delta y^k, -\Delta v^k)$ as the solution of the positive definite system with the main matrix

$$\begin{pmatrix} A\Theta^k A^{\mathrm{T}} & A\Theta^k C^{\mathrm{T}} \\ C\Theta^k A^{\mathrm{T}} & C\Theta^k C^{\mathrm{T}} + \lambda \frac{\alpha^k}{z_\alpha^k} \lambda^T + (V^k)^{-1} R^k \end{pmatrix} \tag{28}$$

and the right-hand-vector

$$\begin{pmatrix} A\chi^k + \xi_2^k \\ C\chi^k + \pi - Cx^k + \frac{\mu^k}{z_\alpha^k}\lambda - \mu^k (V^k)^{-1} e \end{pmatrix} \tag{29}$$

(d) Recover the remaining components of the Newton search direction by the formulae

$$
\begin{aligned}
\Delta x^k &= -\chi^k + \Theta^k(A^{\mathrm{T}}\Delta y^k - C^{\mathrm{T}}\Delta v^k) \\
\Delta z_\alpha^k &= -\lambda^{\mathrm{T}}\Delta v^k \\
\Delta s^k &= -\Delta x^k \\
\Delta z^k &= \mu_k(X^k)^{-1}e - Z^k e - (X^k)^{-1}Z^k\Delta x^k \\
\Delta r^k &= \mu_k(V^k)^{-1}e - R^k e - (V^k)^{-1}R^k\Delta v^k \\
\Delta w^k &= \mu_k(S^k)^{-1}e - W^k e - (S^k)^{-1}W^k\Delta x^k \\
\Delta\alpha^k &= \frac{\mu_k}{z_\alpha^k} - \alpha^k + \frac{\alpha^k}{z_\alpha^k}\lambda^T\Delta v^k.
\end{aligned}
\tag{30}
$$

(e) Choose a primal step length and a dual step length according to

$$
\begin{aligned}
\delta_p^k = \varrho_p \max\{\delta : & \; x^k + \delta\Delta x^k \geq 0, \\
& \; \alpha^k + \delta\Delta\alpha^k \geq 0, \\
& \; r^k + \delta\Delta r^k \geq 0, \\
& \; s^k + \delta\Delta s^k \geq 0\},
\end{aligned}
\tag{31}
$$

$$
\begin{aligned}
\delta_d^k = \varrho_d \max\{\delta : & \; z^k + \delta\Delta z^k \geq 0, \\
& \; z_\alpha^k + \delta\Delta z_\alpha^k \geq 0, \\
& \; v^k + \delta\Delta v^k \geq 0, \\
& \; w^k + \delta\Delta w^k \geq 0\}.
\end{aligned}
$$

(We use $\varrho_p = \varrho_d = 0.9995$, as suggested in McShane et al. (1989)).

(f) Form the new iterate:

$$
\begin{aligned}
x^{k+1} &= x^k + \delta_p^k\Delta x^k \\
\alpha^{k+1} &= \alpha^k + \delta_p^k\Delta\alpha^k \\
r^{k+1} &= r^k + \delta_p^k\Delta r^k \\
s^{k+1} &= s^k + \delta_p^k\Delta s^k \\
& \\
y^{k+1} &= y^k + \delta_d^k\Delta y^k \\
z_\alpha^{k+1} &= z_\alpha^k + \delta_d^k\Delta z_\alpha^k \\
v^{k+1} &= x^k + \delta_d^k\Delta v^k \\
w^{k+1} &= x^k + \delta_d^k\Delta w^k.
\end{aligned}
\tag{32}
$$

(g) Go to 2 if an appropriate stopping criterion is not satisfied.

If the system defined in step 2c is only approximately solved, then the above-described infeasible algorithm is truncated or inexact. Let $\epsilon > 0$ and $\epsilon_p > 0$ be the tolerances for the total complementarity gap and the primal feasibility error, respectively. The TPIDF algorithm tries to calculate an element of the set

$$\{(x, \alpha, r, s, y, z, z_\alpha, v, w) \in \overline{R} : \| (\xi_2, \xi_3) \| < \epsilon_p, \| (\varpi) \| < \epsilon\},$$

where (ξ_2, ξ_3) and ϖ are the non-null primal feasibility residuals and the complementarity gap residual, respectively) and uses it as an approximation of a solution of (9 - 10), the so-called (ϵ, ϵ_p)-solution.

To ensure the convergence towards an (ϵ, ϵ_p)-solution, it is customary to force the iterates to lie within a neighborhood of the central path. We can use the same type of neighborhoods originally proposed by Kojima et al. (1993) and used in Portugal et al. (2000) and Korzak (2000). Let $\gamma \in (0, 1)$ and $\gamma_p > 0$, and consider the primal

error tolerance $\epsilon_p > 0$. A neighborhood $\mathcal{N}_{-\infty}(\gamma, \gamma_p)$ of the central path for (9 - 10) can be defined by as

$$\begin{aligned}\mathcal{N}_{-\infty}(\gamma,\gamma_p) = \quad & \{(x,\alpha,r,s,y,z,z_\alpha,v,w) \in \overline{R}_+ : \\ & \varpi = (x^{\mathrm{T}}z + \alpha z_\alpha + w^{\mathrm{T}}s), \\ & x_i z_i \geq \gamma\varpi/(2n+1), \quad i = 1,\ldots,n, \\ & w_i s_i \geq \gamma\varpi/(2n+1), \quad i = 1,\ldots,n, \\ & \alpha z_\alpha \geq \gamma\varpi/(2n+1), \\ & \varpi \geq \gamma_p \|(Ax-b, \Pi - Cx + \lambda\alpha - r)\| \ \text{ or} \\ & \|(Ax-b, \Pi - Cx + \lambda\alpha - r)\| \leq \epsilon_p\}\end{aligned}$$

The following trivial lemma gives a connection between $\mathcal{N}_{-\infty}(\gamma, \gamma_p)$ and an (ϵ, ϵ_p)-solution.

Lemma 1. *If $(x,\alpha,r,s,y,z,z_\alpha,v,w) \in \mathcal{N}_{-\infty}(\gamma,\gamma_p)$ and $\varpi \leq min\{\epsilon, \epsilon_p\gamma_p\}$ then $(x,\alpha,r,s,y,z,z_\alpha,v,w)$ is an $(\epsilon,\epsilon_p)-$solution of (9 - 10).*

Any primal-dual interior point algorithm, applied to (9 - 10), has to calculate, at each iteration k, a solution

$$\overline{\Delta} = \left(\Delta x^k, \Delta\alpha^k, \Delta r^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta z_\alpha^k, \Delta v^k, \Delta w^k\right),$$

for a system with the matrix

$$\overline{M} = \left(\begin{array}{ccccccccc}
0 & 0 & 0 & 0 & A^{\mathrm{T}} & I & 0 & -C^{\mathrm{T}} & -I \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & \lambda^{\mathrm{T}} & 0 \\
\hline
I & 0 & 0 & I & 0 & 0 & 0 & 0 & 0 \\
A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & -\lambda & I & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
Z^k & 0 & 0 & 0 & 0 & X^k & 0 & 0 & 0 \\
0 & {z_\alpha}^k & 0 & 0 & 0 & 0 & \alpha^k & 0 & 0 \\
0 & 0 & V^k & 0 & 0 & 0 & 0 & R^k & 0 \\
0 & 0 & 0 & W^k & 0 & 0 & 0 & 0 & S^k
\end{array}\right) \tag{33}$$

and the right-hand-vector

$$\overline{h} = \left({\zeta_1}^k, {\zeta_2}^k, \xi_1^k, \ \ \xi_2^k, \ \ \xi_3^k, \ \ \varsigma_1^k, \ \ \varsigma_2^k, \ \ \varsigma_3^k, \ \ \varsigma_4^k\right).$$

In inexact methods, such systems are only approximately solved. An inexact (Newton) search direction is accepted in the TPIDF algorithm if

$$\overline{M\Delta} = \overline{h} + \overline{\epsilon},$$

where the residual vector

$$\overline{\epsilon} = (0,0,0,\epsilon_2^k, \epsilon_3^k, 0,0,0,0)$$

satisfies some appropriate conditions, in order to bound the feasibility problem error. Following Portugal et al. (2000) we consider that ϵ_2^k and ϵ_3^k are such that

$$\|(\epsilon_2^k, \epsilon_3^k)\| \leq \beta_0 \|(Ax^k - b, \ \ \Pi - Cx^k + \lambda\alpha^k - r^k)\|, \tag{34}$$

with $0 \leq \beta_0 \leq \beta_1$.

Although, as referred by Baryamureeba and Steihaug (2006), satisfying this termination criterion for the iterative linear solver is computationally expensive once the iterates become almost primal feasible or primal-dual feasible, it is one of the most practical and computable termination criteria. For example, Freund et al. (1999) suggest to control the errors using other tolerance variables that are updated at every iteration, but are not known *a priori*. The variables reach zero faster than $\|\epsilon_2^k\|$ and $\|\epsilon_3^k\|$, respectively, thus requiring high accuracy, very close to a solution. Furthermore, Mizuno and Jarre (1999) control the error using a seminorm. This norm is not computable, but the accuracy requirement is shown to be slightly weaker than the relative error with respect to the right-hand-side in a modified linear system. This system requires a QR factorization of the matrix A and the accuracy requirement is thus not suitable for computation. Moreover, it was shown that this class of termination criterion has suitable theoretical properties.

In studying the global convergence for the TPIDF algorithm, Portugal et al. (2000) make use of the theory of Kojima et al. (1993), and show that if the steps length δ_p and δ_d in the primal and dual spaces are chosen in the same way as in Kojima et al. (1993) then, because of primal error feasibility is forced to be bounded, the sequence of iterates generated by the TPIDF algorithm is restricted to the appropriated neighborhood and the complementarity gap is reduced. However `PDNET` follows the usual choice of McShane et al. (1989) for the step length δ_p and δ_d. Furthermore, the initial point in `PDNET` is not necessarily in the neighborhood of convergence. Despite this fact the authors of `PDNET` did not observe divergence in their computational experiences.

4. Implementation issues

The main difficulties arising in the implementation of the TPIDF-MNFP algorithm are related to the solution of the system of normal equations (28-29) used to compute the Newton search direction. While in the application of the interior point algorithm for network flow problems with single objective, we solve at each iteration k, systems of matrices of the form $A\Theta^k A^{\mathrm{T}}$, where Θ^k is a diagonal matrix with diagonal positive elements and A is the adjacency matrix of the network, in TPIDF-MNFP (network flow problems with multiple objectives) the systems are solved at each iteration k with the matrices (28) that are poorly conditioned.

Other implementation aspects are the determination of an initial interior point solution to start the algorithm and the description of the stopping criteria.

4.1. **Starting point.**

In network flow problems it is customary to consider costs and capacities as non-negative values. We thus assume that

$$\sum_{t=1}^{p} c_t > 0$$

and $u > 0$. The starting point $(x^0, \alpha^0, r^0, s^0, y^0, z^0, {z_\alpha}^0, v^0, w^0)$ needs to be dual feasible. It is easy to verify that if conditions

$$\begin{aligned} w^0 &> 0 \\ y^0 &= (0, 0, \ldots, 0) \\ v^0 &= (\nu, \nu, \ldots, \nu) \\ {z_\alpha}^0 &= 1 - \nu \sum_{t=1}^{p} \lambda_t \\ z^0 &= w^0 + (\rho + \nu) \sum_{t=1}^{p} c^t \end{aligned}$$

are satisfied, where

$$0 < \nu \leq \left(\sum_{t=1}^{p} \lambda_t \right)^{-1},$$

then $(y^0, z^0, {z_\alpha}^0, v^0, w^0) \in \mathcal{F}_d$; i.e. this starting point is interior dual feasible. Moreover, it is not difficult to define the primal components $(x^0, \alpha^0, s^0, r^0)$ that satisfy the primal constraints of the MNFP algorithm, except, possibly, the primal condition $Ax = b$. In fact, if $(x^0, \alpha^0, s^0, r^0)$ verifies the conditions

$$\begin{aligned} x^0 &> 0 \\ s^0 &= u - x^0 \\ \alpha^0 &\geq \max_i \{ \frac{|{c_i}^{\mathrm{T}} x^0 - \pi_i|}{\lambda_i} \} \\ r_t^0 &= \pi_t - {c_t}^{\mathrm{T}} x^0 + \lambda_t \alpha^0, \quad \text{for } t = 1, \ldots, p, \end{aligned}$$

then the starting point belongs to $\tilde{\mathcal{F}}_p$. In our implementation, we compute a starting point in $\tilde{\mathcal{F}}$, using the above formulae with x^0, s^0, α^0, w^0, and ν computed as

$$\begin{aligned} \nu &= \left(2 \sum_{t=1}^{p} \lambda_t \right)^{-1} \\ x^0 &= u/2 \\ s^0 &= u/2 \\ \alpha^0 &= 1 + \sum_{t=1}^{p} \frac{|{c_t}^{\mathrm{T}} - \pi_t|}{\lambda_t} \\ w^0 &= \frac{{r^0}^{\mathrm{T}} v^0 + \alpha^0 {z_\alpha}^0 + x^0}{s^0(2n + m + 2)} \sum_{t=1}^{p} c_t. \end{aligned}$$

4.2. **Stopping criterion.** Let the error of the current iterate be defined as

$$\epsilon = \varpi^k + {\zeta_2}^k + \frac{\zeta_1^k}{\omega_c} + \frac{\xi_1^k}{\omega_u} + \frac{\xi_2^k}{\omega_b} + \frac{\xi_3^k}{\omega_\pi}, \tag{35}$$

where, ζ_1^k, ζ_2^k, ξ_1^k, ξ_2^k, ξ_3^k, are the primal and dual residues at iteration k, given by (19-21) and (22-23), and

$$\begin{aligned}\omega_b &= 1 + \|b\| \\ \omega_\pi &= 1 + \|\pi\| \\ \omega_u &= 1 + \|u\| \\ \omega_c &= 1 + \|\rho \sum_{q=1}^{p} c_q\| \\ \omega_g &= \omega_b + \omega_\pi + \omega_u + \omega_c - 3.\end{aligned}$$

An iterate is accepted as a solution if the error falls below some given tolerance, e.g. if $\epsilon < 10^{-6}$.

4.3. **Solving the Newton normal systems of equations.** System (28) has a positive definite matrix, so the solution of the normal equations exists and it is unique in each interior point iteration k. However, as the duality measure tends to zero, the matrices of the reduced systems tend to be poorly conditioned, which causes serious numerical drawbacks. To solve the normal equations we implemented two methods: a direct method based on the Cholesky decomposition of the matrix in (28), and an iterative method, the preconditioned conjugate gradient method (`pcg`) with a diagonal preconditioner. Although the Cholesky method is one of the most used to solve positive definite systems and one of the most used in interior-point methods (particularly in feasible interior-point algorithms), the iterative methods, as the `pcg` method, have been used to solve network flow problems (Portugal et al., 2000; Castro, 2005). We compared the performance of these two methods to solve the normal equations (28-29) and discuss computational results in the next section. In this section we describe the procedures used to implement both methods for solving system (28-29). We begin by explicitly defining the matrices and vectors used in the computation.

To compute the right-hand-side vector of each system (28-29), we consider, in each iteration k, the auxiliary vector g^k, defined as

$$g^k = \Gamma^{-1}[X^k S^k (Z^k - W^k)e + (X^k - S^k)e\mu^k],$$

where $\Gamma = (Z^k S^k + W^k X^k)$. Then, the right-hand-side vector $\overline{b}^k = (\overline{b}_1, \overline{b}_2)$ of each system (28-29) is computed as

$$\begin{aligned}\overline{b}_1 &= b - Ax^k + Ag^k \\ \overline{b}_2 &= \overline{\xi_3}^k + Cg^k.\end{aligned}$$

To compute $\overline{b^k}$, we only need to calculate vector g^k, which is defined by applying algebraic operations between diagonal matrices, and performing matrix-vector operations, with the network matrix A and with the dense, but usually small, cost matrix C. If x is an n-vector, the vector $\omega = (\omega_t)_{i=1,\ldots,p}$ resulting from the product of x and the matrix C, can be trivially calculated as

$$\omega_t = \sum_{i=1}^{n} c_i^t x_i.$$

On the other hand, to perform the matrix-vector product with the network matrix A we have the following routine:

Computing the matrix-vector product $\mathbf{y} = \mathbf{Ax}$.

(1) Initialize $y_i = 0$, for all $i = 1, \ldots, m$.
(2) For $i = 1, \ldots, n$ do
$snod =$ "head" of the arc i
$enod =$ "tail" of the arc i
$y_{snod} = y_{snod} + x_i$
$y_{enod} = y_{enod} - x_i$
end for;
(3) $y_{root} = 0$, where *root* is the root node.

The network matrix A has a redundant row which we eliminate considering null the row corresponding to the *root* node, as in Portugal et al. (2000).

To make explicit the elements of matrix (28) considered in each interior-point iteration, we use the following notation. Let $\widehat{m}$ and $\widehat{p}$ be the indices defined, respectively, by

$$\widehat{m} = \frac{m(m-1)}{2} \quad \text{and} \quad \widehat{p} = \frac{p(p-1)}{2}.$$

The sub-matrices in (28) are of the form

$$A\Theta^k A^\top = \begin{pmatrix} d_1 & q_1 & q_2 & \cdots & q_{\widehat{m}-(m-2)} \\ q_1 & d_2 & q_3 & \cdots & q_{\widehat{m}-(m-3)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ q_{\widehat{m}-(m-2)} & q_{\widehat{m}-(m-3)} & \cdots & q_{\widehat{m}} & d_m \end{pmatrix},$$

$$C\Theta^k A^\top = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1(m-1)} & g_{1m} \\ g_{21} & g_{22} & \cdots & g_{2(m-1)} & g_{2m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ g_{p1} & g_{p2} & \cdots & g_{p(m-1)} & g_{pm} \end{pmatrix},$$

$$C\Theta^k C^\top + \lambda \frac{\alpha^k}{z_\alpha^k} \lambda^T + (V^k)^{-1} R^k = \begin{pmatrix} d_{m+1} & f_1 & f_2 & \cdots & f_{\widehat{p}-(p-2)} \\ f_1 & d_{m+2} & f_3 & \cdots & f_{\widehat{p}-(p-3)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ f_{\widehat{p}-(p-2)} & f_{\widehat{p}-(p-3)} & \cdots & f_{\widehat{p}} & d_{m+p} \end{pmatrix}.$$

In each iteration, the positive definite matrix of (28) is symmetric. We can thus store it into two vectors, $\overline{d}$ and $\overline{s}$, respectively,

$$\overline{d} = (d_1, \ldots, d_m, d_{m+1}, \ldots, d_{m+p}),$$

and

$$\begin{aligned} \overline{s} = \; & (q_1, \ \ldots, \ q_{\widehat{m}}, \\ & g_{11}, \ \ldots, \ g_{1m}, \\ & g_{21}, \ \ldots, \ g_{2m}, \ f_1, \\ & g_{31}, \ \cdots \ g_{3m}, \ f_2, f_3, \\ & \quad \cdots \\ & g_{p1}, \ \ldots, \ g_{pm}, \ f_{\widehat{p}-(p-2)}, \ \ldots, \ f_{\widehat{p}}). \end{aligned}$$

The first m diagonal elements stored in $\overline{d}$, are obtained from $diag(A\Theta^k A^{\mathrm{T}})$, and thus we can explore the network structure to improve the algorithm, as we can see in the next procedure.

Computing the diagonal elements $\mathbf{d_j}$, $\mathbf{j = 1, \ldots, m}$.

(1) Initialize $d_j = 0$, for all $j = 1, ..., m$.
(2) For $i = 1, ..., n$ do
$snod =$ "head" of the arc i
$enod =$ "tail" of the arc i
$d_{snod} = d_{snod} + \Theta_i^k$
$d_{enod} = d_{enod} + \Theta_i^k$
end for;
(3) $d_{root} = 0$, where *root* is the root node.

The remaining p diagonal elements, d_{m+t}, $t = 1, \ldots, p$, stored in $\overline{d}$, are the diagonal elements of the dense, but small, matrix $C\Theta^k C^{\mathrm{T}} + \lambda \frac{\alpha^k}{z_\alpha^k} \lambda^T + (V^k)^{-1} R^k$, and they can be simply computed with

$$d_{m+t} = \left(\sum_{i=1}^{n} \Theta_i (c^t{}_i)^2 \right) + (r_t / v_t) + \frac{\alpha}{z_\alpha} (\lambda_t)^2.$$

Next we describe the procedures used to compute the $(\widehat{m} + pm + \widehat{p})$ elements of the vector $\overline{s}$. First, we observe that we can again explore the network structure when we calculate the elements $q(t)$, $t = 1, \ldots, \widehat{m}$, of the lower triangular submatrix of $A\Theta^k A^{\mathrm{T}}$.

Computing the non-diagonal elements, $\mathbf{q(t)}$, $\mathbf{t = 1, \ldots, \widehat{m}}$ (storage by rows).

(1) Initialize $t = 1$.
(2) For $i = 1, \ldots, m$ and $i \neq root$ do
For $j = 1, \ldots, i-1$ do
$q(t) = 0$
If $i \neq root$ then
For $arc = 1, \ldots, n$ do
$snod =$ "head" of the arc i
$enod =$ "tail" of the arc i
If (($snod = i$ and $enod = j$)
or ($enod = i$ and $snod = j$)) then
$q(t) = q(t) - \Theta(arc)$
end if;
end for;
end if;
$t = t + 1$
end for;
end for;
(3) Initialize $index(1) = 1$ and $q(root) = 0$.
(4) For $i = 1, \ldots, n-1$ and $i \neq root$ do
$index(i) = i - 1 + index(i-1)$
$t = index(i)$

$q(t) = 0$
end for;

To compute the elements of $C\Theta^k A^{\mathrm{T}}$ we can transpose the p vector-columns, $\overline{u}^t$, $t = 1, \ldots, p$, which are computed by multiplying A by each vector $\Theta^k c^t$, that is,

$$\overline{u}^t = A(\Theta^k c^t).$$

Moreover, to compute the remaining elements $f_1, f_2, \ldots, f_{\widehat{p}}$, of $\overline{s}$, from the matrix $C\Theta^k C^{\mathrm{T}} + \lambda \frac{\alpha^k}{z_\alpha^k} \lambda^T + (V^k)^{-1} R^k$ we used the following procedure.

Computing the non-diagonal elements $\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_{\widehat{p}}}$ (storage by rows).

(1) Initialize $t = 1$.
(2) For $i = 2, \ldots, p$ do
  For $j = 1, \ldots, i-1$ do
    $f_t = 0$
    For $k = 1, \ldots, n$ do
      $f_t = f_t + \lambda_i \lambda_j \frac{\alpha}{z_\alpha} + c_k^i c_k^j \theta_k$
    end for;
    $t = t + 1$
  end for;
end for;

In the next two sub-sections we described the methods used to solve system (28-29). The Newton search components $(\Delta y^k, \Delta v^k)$, in each iteration k, are obtained by the solution (dy, dv) of each system (28-29), as

$$\Delta y^k = dy$$

and

$$\Delta v^k = -dv.$$

4.3.1. *Cholesky method.* The Cholesky method method requires that the elements of the system matrix be given explicitly. However, if the system is symmetric, it is only necessary to have two vectors to store its elements. To solve system (28-29), in each interior point iteration k, consider the above described vectors $\overline{d}$, and $\overline{s}$, respectively, with the diagonal and non-diagonal lower part of the matrix of the system, that need to be first computed, and the right-hand-side vector $\overline{b^k}$. Then we can use the following procedure to obtain the solution of (28-29). This routine explores the symmetric structure of the matrix. To simplify the exposition, in the next description we use b, d and s for $\overline{b^k}$, $\overline{d}$, and $\overline{s}$.

Solve the symmetric system (28-29) by a direct method.

(1) For $k = 1, 2, \ldots, (m+p) - 1$ do
  For $i = k+1, \ldots, (m+p)$ do
    $l = \frac{(i-1)(i-2)}{2} + k$
    $aux = s_l$

$s_l = \frac{s_l}{d_k}$
$d_i = d_i - s_l aux$
end for;
For $j = k+1. \ldots, i-1$ do
$\alpha = l + j - k$
$\gamma = \frac{(j-1)(j-2)}{2} + k$
$s_\alpha = s_\alpha - s_\gamma aux$
end for;
end for;

(2) For $k = 2, \ldots, (m+p)$ do
For $j = 1, \ldots, k-1$ do
$l = \frac{(k-1)(k-2)}{2} + j$
$b_k = b_k - s_l b_j$
end for;
end for;

(3) For $k = 1, \ldots, (m+p)$ do
$b_k = \frac{b_k}{d_k}$
end for;

(4) For $k = (m+p) - 1, \ldots, 1$ do
For $j = k+1, \ldots, (m+p)$ do
$l = \frac{(j-1)(j-2)}{2} + k$
$b_k = b_k - s_l b_j$
end for;
end for;

4.3.2. *Preconditioned Conjugate Gradient Method.* When an iterative method is used to find the Newton direction, there are no guarantees that the primal-dual interior point method converges. However, it is possible to define a stopping criterion for the preconditioned conjugate gradient algorithm that guarantees sufficient accuracy (see Portugal et al. (2000) and Baryamureeba and Steihaug (2006)). The choice of a good preconditioner is another important point to consider. We only considered the diagonal pre-conditioner

$$D^k = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix},$$

where $D_1 = diag(d_1, \ldots, d_m)$ and $D_2 = diag(d_{m+1}, \ldots, d_{m+p})$ are the diagonal matrices obtained from the diagonal vector $\overline{d}$ containing the diagonal elements of matrix (28). Note that all those elements are strictly positive. Given a starting solution, for example $(dy, dv) = (0, 0)$, the pseudo-code is as follows.

Solve the symmetric system (28) by the pcg method.

(1) $\begin{pmatrix} dy^0 \\ dv^0 \end{pmatrix} = \begin{pmatrix} dy \\ dv \end{pmatrix}$

(2) Compute:

$$r_1{}^0 = \overline{b_1} - (A\Theta^k A^{\mathrm{T}} dy^0 + A\Theta^k C^{\mathrm{T}} dv^0)$$

$$
\begin{aligned}
{r_2}^0 &= \overline{b_2} - (C\Theta^k A^{\mathrm{T}} dy^0 + (C\Theta^k C^{\mathrm{T}} + \\
&\quad \lambda \tfrac{\alpha^k}{z_\alpha^k} \lambda^T + (V^k)^{-1} R^k) dv^0)
\end{aligned}
$$

(3) $\begin{pmatrix} {z_1}^0 \\ {z_2}^0 \end{pmatrix} = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}^{-1} \begin{pmatrix} {r_1}^0 \\ {r_2}^0 \end{pmatrix}$

(4) $\begin{pmatrix} {p_1}^0 \\ {p_2}^0 \end{pmatrix} = \begin{pmatrix} {z_1}^0 \\ {z_2}^0 \end{pmatrix}$

(5) $i := 0;$

(6) do stopping criterion not satisfied $\rightarrow$

(7) call *main* `pcg` *cycle* procedure

(8) $i := i + 1$

(9) od;

(10) $\begin{pmatrix} dy \\ dv \end{pmatrix} = \begin{pmatrix} dy^i \\ dv^i \end{pmatrix}$

The main procedure of `pcg` is next described.

procedure `pcg` *cycle*

Compute:

(1) ${q_1}^i = A\Theta^k A^{\mathrm{T}} {p_1}^i + A\Theta^k C^{\mathrm{T}} {p_2}^i$

(2)
$$
\begin{aligned}
{q_2}^i &= C\Theta^k A^{\mathrm{T}} {p_1}^i + \\
&\quad (C\Theta^k C^{\mathrm{T}} + \lambda \tfrac{\alpha^k}{z_\alpha^k} \lambda^T + (V^k)^{-1} R^k) {p_2}^i
\end{aligned}
$$

(3) $\epsilon^i = (z_1^i r_1^i + z_2^i r_2^i)$

(4) $\gamma^i := \epsilon^i / (p_1^i q_1^i + p_2^i q_2^i)$

(5) $\begin{pmatrix} dy^{i+1} \\ dv^{i+1} \end{pmatrix} = \begin{pmatrix} dy^i \\ dv^i \end{pmatrix} + \gamma^i \begin{pmatrix} {p_1}^i \\ {p_2}^i \end{pmatrix}$

(6) $\begin{pmatrix} {r_1}^{i+1} \\ {r_2}^{i+1} \end{pmatrix} = \begin{pmatrix} {r_1}^i \\ {r_2}^i \end{pmatrix} - \gamma^i \begin{pmatrix} {q_1}^i \\ {q_2}^i \end{pmatrix}$

(7) $\begin{pmatrix} {z_1}^{i+1} \\ {z_2}^{i+1} \end{pmatrix} = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}^{-1} \begin{pmatrix} {r_1}^{i+1} \\ {r_2}^{i+1} \end{pmatrix}$

(8) $\beta^i := (z_1^{i+1} r_1^{i+1} + z_2^{i+1} r_2^{i+1}) / \epsilon^i$

(9) $\begin{pmatrix} {p_1}^{i+1} \\ {p_2}^{i+1} \end{pmatrix} = \begin{pmatrix} {z_1}^{i+1} \\ {z_2}^{i+1} \end{pmatrix} + \beta^i \begin{pmatrix} {p_1}^i \\ {p_2}^i \end{pmatrix}$

A measure for the accuracy of the `pcg` solution at iteration i can be given by the residual $(r_1, r_2) = ({r_1}^i, {r_2}^i)$. To guarantee sufficient accuracy of the interior point search direction, at the interior point iteration k, the condition

$$\|(r_1, r_2) \leq \beta_0 \|(\xi_2^k, \xi_3^k)\| \tag{36}$$

should be satisfied (see Portugal et al. (2000) and Baryamureeba and Steihaug (2006)). This condition is used as the *stopping criterion* for the `pcg` algorithm.

TABLE 1. Data dimension of test problems

Problem	Vertices	Arcs	Rows	Columns	Nonzeros
P1	5	7	14	17	46
P2	10	30	42	63	184
P3	40	600	642	1203	3604
P4	100	1000	1102	2003	6004
P5	300	900	1202	1803	5404
P6	500	2500	3002	5003	15004
P7	600	8000	8602	16003	48004
P8	1500	10000	11502	20003	60004

There is a computational advantage of the `pcg` algorithm over the Cholesky method, applied to solve (28-29), because it does not require the explicit computation of the matrix of the system. Using the associative propriety, only matrix-vector multiplications are needed. We can use the procedure already described to perform operations with the network matrix A. Moreover, matrix-vector products with the transpose of the network matrix, A^T, can be easily computed by the next procedure.

Computing the matrix-vector product, $\mathbf{x} = \mathbf{A^T y}$.

(1) Initialize $x_i = 0$, for all $i = 1, ..., n$
(2) For $i = 1, ..., n$ do

$snod$ = "head" of the arc i
$enod$ = "tail" of the arc i
If $snod \neq root$ and $enod \neq root$ then
  $x_i = y_{snod} - y_{enod}$
If $snod = root$ then
  $x_i = -y_{enod}$
If $enod = root$ then
  $x_i = +y_{snod}$
end for;

5. Computational experiments and results

In this section we report on our computational experience with the above algorithm for solving multi-objective problems, by using a set of multi-objective problems with two objectives. The algorithms were coded in FORTRAN. Parts of the code were adapted from the FORTRAN subroutines in Portugal et al. (2008). All the experiments were performed on a 1.69 GHz Personal Computer with 256 MB of RAM. The set of test problems was generated with a modification of the NETGEN generator (Klingman et al., 1974) to obtain network flow problems with two objectives. Table 1 lists the names and dimensions of the test problems.

The results of the computational experiments are summarized in Tables 2 to 10, where the CPU time is measured in seconds.

TABLE 2. Direct Variant of TPIDF-algorithm ($\pi = (70, 30); \lambda_1 = 0.9, \lambda_2 = 0.1$): Iterations and time (in seconds) for three levels of precision

	$\epsilon < 5 \times 10^{-3}$		$\epsilon < 10^{-6}$		$\epsilon < 10^{-8}$	
Problem	Iters	Time	Iters	Time	Iters	Time
P1	5	0	9	0	10	0
P2	8	0	10	0	12	0
P3	16	0	19	0	21	0
P4	19	2	22	2	24	2
P5	17	10	22	13	24	14
P6	19	3660	25	3647	27	7200
P7	25	530	32	666	—	> 7200
P8	23	4767	31	6378	—	> 7200

TABLE 3. `pcg` Variant of TPIDF-algorithm ($\pi = (70, 30), \lambda_1 = 0.9, \lambda_2 = 0.1$): Iterations and times (in seconds) for three levels of precision

	$\epsilon < 5 \times 10^{-3}$		$\epsilon < 10^{-6}$		$\epsilon < 10^{-8}$	
Problem	Iters	Time	Iters	Time	Iters	Time
P1	7	0	9	0	10	0
P2	8	0	12	0	13	0
P3	25	0	29	0	30	0
P4	34	0	39	0	40	0
P5	31	3	35	6	41	9
P6	41	12	46	21	53	35
P7	61	58	65	91	75	244
P8	71	319	83	964	87	1019

We implemented and compared two variants of the TPIDF-algorithm that differ with respect to the method used to solve the system which defines the Newton direction. One variant uses a direct method based on Cholesky factorization while the other one uses an iterative method, the preconditioned conjugate gradient method with diagonal preconditioner.

With the direct-solver variant, we obtained the following results for three precision levels referred in the table by ϵ. We are able to observe (Table 2) how difficult it was for this *direct* variant to reach the solution in the larger problems.

Instead, if the `pcg` method is used to compute the search direction, much better times are obtained, although more interior point iterations are needed, as can be seen in Table 3.

Moreover, the closer the interior-point algorithm is to the optimal solution, the more time it takes to find an admissible iterate. In the direct-solver variant, one can observe that the closer one is from a primal-dual solution, the more poorly conditioned is the system that defines the search direction, so more time is needed to obtain an interior admissible solution. On the other hand, one observes a similar

TABLE 4. Iterations in pcg Variant of TPIDF-algorithm ($\pi = (70, 30), \lambda_1 = 0.9, \lambda_2 = 0.1$): Interior point method (IPM) iterations and preconditioned conjugate gradient (PCG) iterations for three levels of precision.

	$\epsilon < 5 \times 10^{-3}$		$\epsilon < 10^{-6}$		$\epsilon < 10^{-8}$	
Problem	IPM	PCG	IPM	PCG	IPM	PCG
P1	7	39	9	52	10	57
P2	8	59	12	98	13	109
P3	25	450	29	591	30	633
P4	34	1774	39	2463	40	2630
P5	31	3656	35	6845	41	10477
P6	41	6926	46	12484	53	21953
P7	61	13481	65	21389	75	60420
P8	71	46481	83	149945	87	152546

behavior when the pcg-method is used to compute the search direction, but less time is required. The pcg variant of the TPIDF algorithm was by far preferred over the variant where a direct method was used to compute the search direction.

The increase in time needed for the pcg-method to compute the search direction was related to the increase in the number of iterations required to find a search direction, as we can see in Table 4.

An excessive increase in the pcg iterations to obtain more precise results, from $\epsilon < 5 \times 10^{-3}$ to $\epsilon < 5 \times 10^{-8}$, can be observed. In practice, this is responsible for the excessive amount of time taken by the algorithm to find the solution of the problem. This point could be potentially improved with a more appropriate preconditioner.

In the experiments, we also ran the pcg variant with different bi-criteria reference points, namely, the ideal point or the nadir point of the multi-objective problem. As we can observe from the results in Tables 5 and 6, when the CPU time of the algorithm was compared to the number of pcg iterations obtained for the ideal point and for the nadir point, better results were obtained for the ideal point, as expected. Table 6 shows the results obtained when the nadir point was used instead of the ideal point.

Concluding the experiments, we also have considered different values for the objective weights, using the pcg variant of the TPIDF-algorithm for the ideal point as the multi-objective reference point. The TPIDF algorithm did not seem to be very sensitive to changes in the objective weights for the multi-objective problem. The results presented in Tables 7-10 were stated for a precision of $\epsilon = 10^{-8}$.

6. CONCLUSIONS

In spite of the fact that the single- and multi-objective network flows problems are quite different from an algebraic as well as numerical point of view, we observe that the techniques described in Portugal et al. (2000; 2008), with the proper adaptations, can be used to solve network flow problems with several objectives. The main contribution of this paper was to show that the inexact infeasible primal-dual

TABLE 5. Interior point method iterations and time in seconds and number of PCG iterations for the `pcg` variant when π is the ideal point ($\lambda_1 = 0.9, \lambda_2 = 0.1$) with precision $\epsilon = 10^{-8}$.

Problem	IPM iters	Time	PCG iters
P1	10	0	57
P2	16	0	133
P3	30	0	685
P4	43	1	3277
P5	34	5	6291
P6	41	38	23824
P7	84	135	32253
P8	97	947	140161

TABLE 6. Interior point method iterations and time in seconds and number of PCG iterations for the `pcg` variant when π is the nadir point ($\lambda_1 = 0.9, \lambda_2 = 0.1$) with precision $\epsilon = 10^{-8}$.

Problem	IPM iters	Time	PCG iters
P1	12	0	63
P2	15	0	107
P3	25	0	496
P4	29	1	1704
P5	40	15	18303
P6	34	40	24975
P7	45	165	39339
P8	66	2008	290450

TABLE 7. Interior point method iterations and time in seconds and number of PCG iterations for the `pcg` variant with objective weight values $\lambda_1 = 0.7$ and $\lambda_2 = 0.3$.

Problem	IPM iters	Time	PCG iters
P1	10	0	51
P2	14	0	112
P3	26	0	5222
P4	31	1	1740
P5	34	4	4688
P6	43	28	17113
P7	55	123	29077
P8	102	1029	150399

TABLE 8. Interior point method iterations and time in seconds and number of PCG iterations for the `pcg` variant with objective weight values $\lambda_1 = 0.5$ and $\lambda_2 = 0.5$.

Problem	IPM iters	Time	PCG iters
P1	9	0	50
P2	14	0	113
P3	27	0	557
P4	32	1	1550
P5	41	8	9484
P6	41	13	7550
P7	59	116	27439
P8	75	1105	163670

TABLE 9. Interior point method iterations and time in seconds and number of PCG iterations for the `pcg` variant with objective weight values $\lambda_1 = 0.3$ and $\lambda_2 = 0.7$.

Problem	IPM iters	Time	PCG iters
P1	9	0	52
P2	15	0	118
P3	31	0	768
P4	31	1	1740
P5	39	7	9169
P6	45	20	10727
P7	61	114	27448
P8	69	876	129287

TABLE 10. Interior point method iterations and time in seconds and number of PCG iterations for the `pcg` variant with objective weight values $\lambda_1 = 0.1$ and $\lambda_2 = 0.9$.

Problem	IPM iters	Time	PCG iters
P1	9	0	6
P2	14	0	101
P3	29	0	671
P4	35	1	1594
P5	42	7	8696
P6	46	17	10375
P7	70	94	21724
P8	69	873	123429

interior point algorithm is a useful tool to solve multi-objective problems. Unfortunately, our TPIDF algorithm implementation suffers from the performance of the standard diagonal preconditioner used in the `pcg` to compute the Newton direction. More investigation needs to be done to derive improved preconditioners for the multi-objective network flow problem.

We limited our computational study to bi-objective instances, but the method can accommodate more objectives, which is not the case with some classical techniques for network flow problems. We are unaware of any other practical (effective and efficient) algorithm for solving multi-objective network flow problems with several objectives. Furthermore, our computational experiments show that the interior point algorithm implemented is not very sensitive to the variation of the reference points as well as of the "weights" assigned to the different functions. Therefore, we think that these algorithms can be easily used from an interactive point of view, which is one of the lines for our future research.

Acknowledgement

The two first authors would like to acknowledge the financial support of the MONET research project (POCTI/GES/37707/2001). The second author also acknowledges DIMACS Research Center at Rutgers University for his sabbatical year when starting this work in 2003.

References

B. Aghezzaf and T. Ouaderhman. An interactive interior point algorithm for multiobjective linear programming problems. *Operations Research Letters*, 29(4): 163–170, 2001.

A. Arbel. An interior multiobjective linear programming algorithm. *Computers & Operational Research*, 20(7):723–735, 1993.

A. Arbel. Anchoring points and cones of opportunities in interior multiobjective linear programming. *Journal of the Operational Research Society*, 45(1):83–96, 1994a.

A. Arbel. A multiobjective interior primal-dual linear programming algorithm. *Computers & Operations Research*, 21(4):433–445, 1994b.

A. Arbel. An interior multiple objective primal-dual linear programming algorithm using efficient anchoring points. *Journal of the Operational Research Society*, 46 (9):1121–1132, 1995.

A. Arbel. An interior multiobjective primal-dual linear programming algorithm based on approximated gradients and efficient anchoring points. *Computers & Operations Research*, 24(4):353–365, 1997.

A. Arbel and P. Korhonen. Using aspiration levels in an interactive interior multiobjective linear programming algorithm. *European Journal of Operational Research*, 89(1):193–201, 1996a.

A. Arbel and P. Korhonen. Using aspiration levels in an interior primal-dual multiobjective linear programming algorithm. *Journal of Multi-Criteria Decision Analysis*, 5(2):61–71, 1996b.

A. Arbel and P. Korhonen. Using objective values to start multiple objective linear programming algorithms. *European Journal of Operational Research*, 128(3): 587–596, 2001.

A. Arbel and S. Oren. Generating interior search directions for multiobjective linear programming. *Journal of Multi-Criteria Decision Analysis*, 2(2):73–86, 1993.

A. Arbel and S. Oren. Using approximate gradients in developing an interactive interior primal-dual multiobjective linear programming algorithm. *European Journal of Operational Research*, 89(1):202–211, 1996.

V. Baryamureeba and T. Steihaug. On the convergence of an inexact primal-dual interior point method for linear programming. In *Large-Scale Scientific Computing: Lecture Notes in Computer Science*, volume 3743, pages 629–637. Springer Berlin / Heidelberg, 2006.

V. Baryamureeba, T. Steihaug, and Y. Zhang. Properties of a class of preconditioners for weighted least squares problems. Technical report, Department of Informatics, University of Bergen, Norway, 1999. No. 170.

J. Castro. An interior-point approach for primal block-angular problems. Technical report, Dept. of Statistics and Operations Research, Universitat Politecnica de Catalunya, Barcelona, Spain., September 2005. No. 20.

J. Figueira, S. Greco, and M. Ehrgott. *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*. Springer Science + Business Media, Inc., New York, 2005.

R. Freund and F. Jarre. A QMR-based interior-point algorithm for solving linear programs. *Mathematical Programming*, 76:183–210, 1997.

R. Freund, F. Jarre, and S. Mizuno. Convergence of a class of inexact interior-point algorithms for linear programs. *Mathematics of Operations Research*, 24 (1):50–71, 1999.

A. Klingman, A. Napier, and J. Stutz. Netgen A program for generating large scale capacitated assignement, transportation, and minimum cost network flow problems. *Management Science*, 20(5):814–821, 1974.

M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior-point algorithm for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming, Interior-Point and Related Methods*, pages 29–47. Spring-Verlag, New York, 1989.

M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61(3):263–280, 1993.

P. Korhonen and J. Laakso. A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research*, 24(2):277–287, 1986.

P. Korhonen and J. Wallenius. A Pareto Race. *Naval Research Logistics*, 35(6): 615–623, 1988.

J. Korzak. Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems. *SIAM Journal on Optimization*, 11(1): 133–148, 2000.

K. McShane, C. Monma, and D. Shanno. An implementation of a primal-dual interior point method for linear programming. *ORSA Journal on Computing*, 1: 70–83, 1989.

D. Megiddo. Pathways to the optimal set in linear programming. In *Progress in Mathematical Programming, Interior-Point and Related Methods*, pages 131–158, New York, 1989. Spring-Verlag.

S. Mizuno and F. Jarre. Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation. *Mathematical Programming*,

84:357–373, 1999.

R. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part I: Linear programming. *Mathematical Programming*, 44(1):27–41, 1989.

R. Monteiro and J. O'Neal. Convergence analysis of a long-step primal-dual infeasible interior-point lp algorithm based on iterative linear solvers. Technical report, School of ISyE, Georgia Tech, USA, October 2003.

A. Oliveira and D. Sorensen. Computational experience with a preconditioner for interior point methods for linear programming. Technical report, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1997. No. 28.

L. Portugal, F. Bastos, J. Júdice, J. Paixão, and T. Terlaky. An investigation of interior point algorithms for the linear transportation problem. *SIAM Journal on Scientific Computing*, 17:1202–1223, 1996.

L. Portugal, M. Resende, G. Veiga, and J. Júdice. A truncated primal-infeasible dual-feasible network interior point method. *Networks*, 35:91–108, 2000.

L. F. Portugal, M. G. C. Resende, G. Veiga, J. Patrício, and J. J. Júdice. Fortran subroutines for network flow optimization using an interior point algorithm. *Pesquisa Operacional*, 28:243–261, 2008.

M. Resende and G. Veiga. An implementation of the dual affine scaling algorithm for the minimum cost flow on bipartite uncapacited networks. *SIAM Journal on Optimization*, 3:516–537, 1993.

R. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application.* John Wiley & Sons, New York, 1986.

K. Tanabe. Centered newton method for mathematical programming. In *System Modeling and Optimization: Proceedings of the 13th IFIP Conference, Tokyo, Japan, Aug./Sept. interiorpoint 1987*, volume 113, pages 197–206. M. Iri and K. Yajima (eds.), Springer-Verlag, 1988.

T. Trafalis, T. Morin, and S. Abhyankar. Efficient faces of polytopes : Interior point algorithms, parameterization of algebraic varieties, and multiple objective optimization. In J. C. Lagarias and M. J. Todd, editors, *Mathematical Developments Arising from Linear Programming: Proceedings of a Joint Summer Research Conference held at Bowdoin College, Brunswick, Maine, USA, June/July 1988*, volume 114 of *Contemporary Mathematics*, pages 319–341. American Mathematical Society, Providence, Rhode Island, USA, 1990.

A. Wierzbicki. The use of reference objectives in multiobjective optimisation. In Fandel G. and Gal T., editors, *Multiple Criteria Decision Making, Theory and Application, Proceedings*, number 177 in LNEMS, pages 468–486, Hagen, 1980. Springer-Verlag.

A. Wierzbicki. On the completeness and constructiveness of parametric characterizations to vector optimization problems. *OR-Spectrum*, 8(2):73–87, 1986.

(M. Fonseca) INESC-Coimbra, National Institute of Engineering of Systems and Computers of Coimbra, Rua Antero de Quental, 199, 3000-033 Coimbra, Portugal
E-mail address: margarid@inescc.pt

(J.R. Figueira) CEG-IST, Center for Management Studies, Instituto Superior Técnico, Tagus Park, Av. Cavaco Silva, 2780-990 Porto Salvo, Portugal
E-mail address: figueira@ist.utl.pt

(M.G.C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address, M.G.C. Resende: mgcr@research.att.com