Annals of Operations Research manuscript No.
(will be inserted by the editor)

Exterior Point Simplex-type Algorithms for Linear and Network Optimization Problems

Konstantinos Paparrizos · Nikolaos Samaras · Angelo Sifaleras

Received: date / Accepted: date

Abstract Two decades of research led to the development of a number of efficient algorithms that can be classified as exterior point simplex-type. This type of algorithms can cross over the infeasible region of the primal (dual) problem and find an optimal solution reducing the number of iterations needed. The main idea of exterior point simplex-type algorithms is to compute two paths/flows. Primal (dual) exterior point simplex-type algorithms compute one path/flow which is basic but not always primal (dual) feasible and the other is primal (dual) feasible but not always basic. The aim of this paper is to explain to the general OR audience, for the first time, the developments in exterior point simplex-type algorithms for linear and network optimization problems, over the recent years. We also present other approaches that, in a similar way, do not preserve primal or dual feasibility at each iteration such as the monotonic build-up Simplex algorithms and the criss-cross methods.

Keywords Mathematical Programming · Exterior Point Simplex-type Algorithms · Linear Programming · Network Flows

K. Paparrizos
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
E-mail: paparriz@uom.gr

N. Samaras
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
Tel.: +30-2310-891866 / Fax: +30-2310-891879
E-mail: samaras@uom.gr

A. Sifaleras
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
Tel.: +30-2310-891884 / Fax: +30-2310-891881
E-mail: sifalera@uom.gr

Mathematics Subject Classification (2000) 90C27 · 65K05 · 90C05 · 90B10 · 91A90

1 Introduction

The Linear Problem is one of the most useful and well-studied optimization problems, which is widely used in several areas of science. Lots of real world problems can be formulated as Linear Programs (LPs). The popularity of linear programming can be attributed to many factors such as the ability to model large problems, and the ability to solve large problems in a reasonable amount of time. Many algorithms have been invented for the solution of a Linear Program (LP). The majority of these algorithms belong to two main categories: (i) Simplex-type or pivoting algorithms and (ii) Interior-Point Methods (IPMs). All the algorithms presented in this paper belong to the first category, except one that belongs to both categories.

The classical Primal Simplex Algorithm (PSA) [36,37] had been the most efficient method for solving LPs until the mid 80's. It performs sufficiently well in practice, particularly on LPs of small or medium size. PSA starts with a feasible basis of the polyhedron $P = \{x|Ax \leq b, x \geq 0\}$ where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and use pivot operations in order to preserve feasibility of the basis and guarantee monotonicity of the objective value. Klee and Minty in their classical paper [74] showed that the Dantzig's largest coefficient pivoting rule performs exponentially on some specially constructed LPs. The first polynomial time algorithm for linear programming is the ellipsoid or russian algorithm which was developed by Khachyan [71]. In 1984, Karmarkar [70] developed the first Interior Point Method (IPM). Subsequent research has led to the development of efficient IPMs which outperform PSA on large-scale LPs, practically as well. IPMs seek to approach the optimal solution through a sequence of points inside the feasible region.

Despite this fact, the computational improvement of simplex-type algorithms remained of great interest. Many pivoting rules are known for Simplex-type algorithms. A complete presentation of them can be found in [119]. Pan [88] offers computational results with large-scale sparse problems, demonstrating superiority of nested Dantzig and Devex rules to commonly used pivoting rules. None of the existing simplex-type algorithms admits polynomial worst case complexity. Borgwardt [23,24] proved that the average case behavior of a shadow-vertex algorithm is polynomial under a probabilistic model. Specifically, the number of iterations needed to solve an LP by the simplex-type algorithms depends upon the pivot columns used. Fukuda and Terlaky [47] using the existence of a known solution for the LP, proved that starting from any basis, a short admissible pivot sequence exists, that contains at most n pivot steps leading to an optimal basis. It remains an open question whether a strongly polynomial algorithm for linear programming exists, whose running time depends only on m and n.

A common feature of almost all simplex-type algorithms is that they can be interpreted as a procedure following simplex-type paths, that lead to the optimal solution. An efficient way to improve a pivoting algorithm is by avoiding moving from one feasible vertex to an adjacent one. This can be accomplished by avoiding the boundary of the polyhedron $P = \{x|Ax \leq b, x \geq 0\}$ and constructing two paths to converge to the optimal solution. One path is exterior to the feasible region while the other is feasible. Such an algorithm is called an Exterior Point Simplex Algorithm (EPSA).

The first EPSA was originally developed by Paparrizos [94], for the Assignment Problem (AP). Later, Paparrizos [96] generalized EPSA to the general LP. Dosios and Paparrizos [42] showed that EPSA is closely related to successive shortest path algorithms and Lemke's algorithm and in [104] they showed that every cycling free pivoting rule of the PSA is also cycling free for the dual EPSA. EPSA constructs two paths to the optimal solution. One path consists of basic but not feasible solutions, while the second path is feasible [43]. The key idea behind EPSA is that making steps in directions that are linear combinations of attractive descent direction can lead to faster convergence than that achieved by PSA. Paparrizos et al. [105] have pointed out that the geometry of EPSA reveals that this algorithm is faster than PSA. This conclusion was partially verified by preliminary computational experiments on randomly generated dense and sparse LPs [101, 103]. However, an intensive computational study on NETLIB LPs [81] is needed to show the practical value of EPSA. Furthermore, in [103] new variations of EPSA using three different strategies to enter Phase II are presented.

Several other approaches have exploited the above key idea. Mitra et al. [82] give a survey of such methods including those within and outside simplex framework. Another similar method is the algorithm presented by Cardoso and Clímaco [26]. Their algorithm starts from an easily found face of the polytope of high dimension. At every iteration the algorithm chooses a new direction among those belonging to one of the faces of immediately higher dimension. This algorithm could be considered as belonging to a group of approaches designated as non-vertex simplex-type algorithms. Gondzio [57] presented an active set algorithm that makes steps in the ascent directions that are the approximate projections on the objective function gradient onto the face defined by the current set of active constraints or the linear combinations of such directions. Al-Sultan and Murty [8] proposed an exterior penalty algorithm for the nearest point problem. Sherali et al. [116] examined three exterior point approaches for solving LPs. These algorithms have the advantage that they can be initialized at arbitrary starting solutions and present a great degree of flexibility in designing particular algorithmic variants.

Recently, Malakooti and Al-Najjar [80] proposed the hybrid-LP, a method for solving LPs using both interior and boundary paths. Their method uses an interior direction to pass to an improved basic feasible solution. Then, the PSA can be applied in order to reach an optimal solution. As a consequence, the proposed method bypass several extreme (boundary) points. Other simplex-type algorithms, that in one way or other, avoid the feasible region can be found in [12, 16, 17, 28, 35, 68, 79, 83, 87, 124].

A well established way of avoiding the previous computational disadvantages, is the transformation of exterior path of EPSA into a dual feasible simplex path. The algorithm that results from this transformation is called Primal-Dual Exterior Point Simplex Algorithm (PDEPSA) [97, 99]. The revised form of the PDEPSA can be found in [102]. This algorithm requires an initial dual feasible basic solution. Since, such solution does not exist for a generic LP, a modified big-M scheme is applied. The main advantage of PDEPSA is its computational performance. Experimental investigation on randomly generated sparse and dense LPs show the practical value of PDEPSA [102, 115]. The computational results are very promising.

Recently, Guerrero-Garcia and Santos-Palomo [59] presented a deficient-basis dual counterpart of Paparrizos, Samaras, and Stephanides' primal-dual simplex-type algorithm [102]. In that paper they address what would be thought of as its deficient-basis dual counterpart by showing how to take advantage of the knowledge of a dual feasible point in a deficient-basis simplex-type environment.

PDEPSA requires an initial dual basic feasible solution and a feasible point of an LP, as it is depicted in Fig. 1. This means that at every iteration the dual slack variables $s = c - A^T w \geq 0$ holds. If point x^1 is feasible to dual LP and point y^1 is feasible to primal LP then the direction $d^1 = x^1 - y^1$ is ascent for the objective function $c^T x$. The direction d^1 creates a ray $R^1 = \{x^1 + td^1 : t \geq 0\}$, which enters the feasible region from the boundary point y^2. The hyperplane from which ray R^1 enters the feasible region corresponds to the basic variable x_k. Then a dual pivot operation on which basic variable x_k exits the basis, is performed. The fact that the points $y^t,\ t = 1, 2, ...$ are boundary, leads to a serious computational disadvantage. This disadvantage occurs because the point y belongs to more than one hyperplanes. This means that ties (degenerate vertices) exist in the choice of leaving variable, which in turn can lead to stalling and/or cycling. Degeneracy degrades efficiency as the simplex-type algorithms could stall at a degenerate vertex for many iterations before it moves to another vertex.

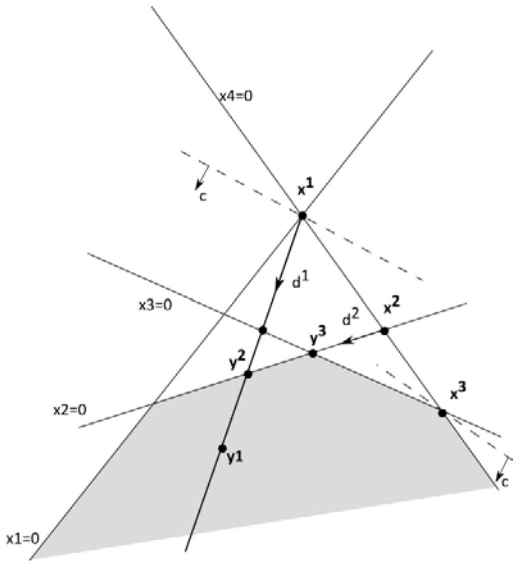

Fig. 1 PDEPSA geometric interpretation.

In order to overcome this disadvantage, PDEPSA has to be modified. An important modification of the PDEPSA is to traverse across the interior of the feasible region, at every iteration, in an attempt to avoid combinatorial complexities and degenerate vertices of vertex-following algorithms. This new algorithm is called Primal-Dual Interior Point Simplex Algorithm (PDIPSA) and developed by Samaras [113]. PDIPSA can be also seen as a procedure to move from any interior point to an optimal basic solution. It can be combined with IPMs, in order to develop a hybrid algorithm consisting of two stages. At the first stage, an IPM is applied and at the second stage PDIPSA is applied to compute an optimal basic solution. Computational studies on randomly generated sparse LPs [51] and on a set of LPs (in mps format) available through NETLIB [113] are used to establish the practical value of the PDIPSA. The computational results are very encouraging.

The idea of an exterior point path was the motivation to try to develop similar algorithms for network optimization problems. Network optimization [3] consists a special type of linear optimization, that has certain computational benefits related to linear optimization. Network flow problems not only can be solved very quickly even with a large number of variables, but also their solution usually leads to integer solutions due to the total unimodularity property. Moreover, a large number of real-world applications in communications, informatics and transportation can be modeled as network flow problems [4, 106].

A non-dual signature method for the AP has been developed by Paparrizos [93] with $O(n^4)$ complexity. This algorithm visits only strong trees that are obtained from strongly feasible trees by dropping the feasibility requirement. A relaxation column signature method for the solution of an AP has been reported in [95]. This algorithm solves a sequence of relaxed transportation problems and generates strong dual feasible bases. This algorithm has an $O(n^3)$ complexity and the average number of iterations is bounded by $n \log(n)$.

Later, Paparrizos developed also EPSA for the Transportation Problem (TP) [98]. The research on initialization methods for EPSAs in TP led to the development of three methods. The first method uses a Balinski's dual feasible tree [94], the second method starts from a simple start forest [1] that is neither primal nor dual feasible, and the third method is based on a dual feasible forest [2]. These initialization methods afterwards have been specialized for the solution of the AP. It is noteworthy, that each one initialization method was proved to be faster than the classic PSA for the TP in computational studies.

After the development of EPSAs for TP and AP, focus was given in generalizing these results in other classic network optimization problems. Towards this direction, the first primal EPSA for the Minimum Cost Network Flow Problem (MCNFP) was recently presented in [100].

Section 2 discusses recent developments in strongly related research approaches such as Monotonic Build-Up Simplex Algorithms (MBUSAs) and Criss-Cross Algorithms (CCAs). Section 3 presents two EPSAs for LPs. EPSAs for network optimization problems are presented in the following three sections. More specifically, Sects. 4, 5, and 6, present EPSAs for the solution of the MCNFP, TP, and AP respectively. Section 7 presents some educational tools for the teaching of EPSAs, using visualization and animation techniques. Finally, Sect. 8 provides some conclusions and open research problems regarding the improvement of EPSAs.

2 Related work

Since the sixties there was a strong interest in finding an efficient way to get to an optimal basis via a series of infeasible ones. Algorithms, like criss-cross, monotonic build-up Simplex algorithms, and others all belong to this exterior point “simplex-like” category. A review paper summarizing the pros and cons of pivot, ellipsoid, and interior point methods was presented by Illeś and Terlaky in [67]. The authors summarized in their findings that, although the theoretical criteria (e.g., worst case complexity) clearly favor IPMs, the pivot methods are keep competing in practice. Furthermore, pivot algorithms are favorable for integer linear problems while the IPMs win for very large, sparse LPs.

2.1 Criss-Cross Algorithms (CCAs)

2.1.1 Zionts’ criss-cross algorithm

A large number of solution approaches has been presented in the last decades, that did not preserve primal or dual feasibility at each iteration. Zionts [126] first proposed in 1969 a criss-cross method for the solution of LPs. The efficiency of this method stems from the fact that this type of algorithms do not require an initial feasible solution, as does the classic Simplex algorithm. Therefore, since there is no need to solve either an auxiliary problem, or use any artificial variable, the criss-cross method starts to work towards the solution of the LP from the beginning (one phase). A formal description of the Ziont’s method, as presented in their paper, is as follows:

Step 0 (Initialization). Partition the initial LP to the corresponding primal and dual problem and start using any primal and/or dual infeasible basis.

Step 1 (Alternate pivoting in the primal and dual problem). Select any of the primal or dual infeasible variables (since the basis is not optimal) as follows:

(i) If a dual infeasible variable is selected, then we have found the entering variable. Afterward, we choose the leaving variable among the primal feasible variables, in a manner that these variables preserve primal feasibility.
(ii) If a primal infeasible variable is selected, then we have identified the leaving variable. Afterward, we choose the entering variable among the dual feasible variables, so that dual feasibility of these variables is preserved.

Step 2 (Termination test). If either the primal or the dual basis is feasible, then the criss-cross method reduces to the classic Simplex algorithm as follows:

(i) If the primal problem is feasible, STOP. Continue with the primal Simplex algorithm.
(ii) If the dual problem is feasible, STOP. Continue with the dual Simplex algorithm.

In this case, the computations are carried out until an optimal solution is achieved, if there is one. Else, if neither the primal or the dual basis is feasible but a basis exchange is possible, then select another infeasible variable and continue with *Step*

1. Otherwise, if the current basis is infeasible and none of the infeasible variables allows a pivot as described above, then no optimal solution exists for the LP.

A few years later in 1972, Zionts presented an improved version of his procedure that accelerates the detection of unbounded and infeasible solutions. Computational results on randomly-generated LPs were also presented in the same paper [127], that clearly shown an improvement both in terms of computational time and also in the number of total iterations.

2.1.2 Terlaky's criss-cross algorithm

A new version of a criss-cross method, quite different from Ziont's procedure was developed by Terlaky in 1985 [118]. The main difference is that although Terlaky's criss-cross method starts from neither a primal or dual feasible solution and alternates between primal or dual steps, it does not continue with the primal or dual Simplex method in case of primal or dual feasibility (i.e., as in *Step 2*, of the following Ziont's method).

Furthermore, a worst-case example of Terlaky's pivoting rule for the criss-cross simplex method was presented by Roos in 1990 [111] who showed that the required number of iterations may be exponential in the number of variables and constraints of the LP. His example used the Klee-Minty polytope [74].

Although, the finiteness of the Terlaky's criss-cross method was initially proved by showing that it cannot cycle using the Bland's pivoting rule [22], Fukuda and Matsui [44] presented a new proof in 1991 by showing that at each iteration an explicit improvement in the sense of lexicographic ordering is made by the algorithm. Fukuda and Terlaky analytically presented the mathematical ideas and proof techniques behind finite criss-cross pivot methods in [46]. Moreover, the authors presented an elegant algorithmic proof of the LP strong duality theorem using the finiteness result of the least-index criss-cross method, as also a series of open problems.

2.1.3 New variants of Criss-Cross Algorithm (CCA)

Klafszky and Terlaky in 1992 [73] generalized the CCA of Terlaky [118] to Quadratic Programming (QP) problem. Valiaho in 1992 [121] gave a new and alternative way of proving the finiteness of CCA for QP. Zhang [125] changed the minimal index selection rule into "last-in-first-out" (LIFO) rule and "most-often-selected-variable" (MOSV) rule and proved the finiteness of CCA for LP with these rules. Ills and Mszros in 2001 [62] simplified the finiteness proves of Zhang [125] for CCA and showed that Farkas lemma (published in 1894 in Hungarian and 1901 in German, see [62] for more details) can be proved using a variant of CCA presented in Klafszky and Terlaky in 1991 [72] using LIFO and MOSV rules, too. Paper of Akkeles et al. in 2004 [7] includes all of the ideas mentioned in these previous papers and show that CCA with LIFO and MOSV anti-cycling rules guarantee finiteness of CCA for QP problems.

Since then, criss-cross methods have been extended for different type of problems (e.g., oriented matroid programming), and also attracted the interest of several researchers who seek to find if there is a polynomial criss-cross algorithm. Recently, Li [78] in 2010 introduced a new practical variant of the finite criss-cross algorithm. Li's method used a generalization of a known heuristic method about optimal basic variables and optimal nonbasic variables of an LP in standard form, that tend to be characterized by bearing lower and higher pivoting indices, respectively. Thus, by using reordering variable indices, Li's new least pivoting index criss-cross algorithm was proved to be efficient in a preliminary computational study using randomly constructed problems and sparse NETLIB problems [48].

2.2 MBUSAs

The first version of EPSA introduced by Paparrizos [94] in 1991 is almost the same as the Monotonic Build-Up Simplex Algorithm (MBUSA) that was independently developed by Anstreicher and Terlaky [13], and published a few years later [14]. The most distinctive characteristic of both EPSA and the MBUSA, is that they produce a sequence of basic solutions, some of which are neither primal nor dual feasible. Thus, MBUSA can be also viewed as an exterior point method.

Bilen et al. [21] followed the main idea of the MBUSA of Anstreicher and Terlaky [14] and presented a finite MBUSA for linear feasibility problems and a recursive anti-degeneracy procedure. The same authors also studied new monotone Simplex algorithms for feasibility problems in their recent work [31].

Csizmadia [30] studied pivot based solution techniques of LPs, linear complementarity problems, and linear feasibility problems. He also computed, under a weaker degeneracy concept, a new upper bound for the number of iterations of the MBUSA developed by Anstreicher and Terlaky [14] for LPs. An interesting real-world application of MBUSAs regarding a blending problem of the Hungarian oil company (MOL Plc.), was also presented in the same PhD Thesis.

A new concept of s-monotone index selection rule for LPs was recently introduced by Csizmadia et al. [33]. The authors proved the finitness of several known methods such as the primal (dual) Simplex, MBUSAs, and criss-cross algorithm with s-monotone pivot rules. Moreover, the practical efficiency of the MBUSAs with various s-monotone pivoting rules was also shown by numerical experiments using test problems from the NETLIB data set.

Additionally, computational aspects of Simplex and MBUSAs using different anti-cycling pivot rules have also been recently studied by Illés and Nagy in [64]. The authors evaluated the flexibility of several variations of index selection rules (e.g., minimal index/Bland, last-in-first-out, most-often-selected-variable) for Simplex type algorithms using the publicly available benchmark collections NETLIB and MIPLIB [75] (the linear relaxation of the problems). Their computational study used 108 test problems in total and it was shown that, the MBUSA can be a practically competitive alternative to the classic PSA. More specifically, the hybrid most-often-selected-variable MBUSA version was proved to be the most efficient among the theoretically finite index selection rules. Furthermore, although the MBUSA requires fewer iterations in average, it spends more time per iteration.

An interesting new variant of the MBUSA for the maximum flow problem with non-zero lower bounds, have also been presented by Illés and Molnár-Szipai in [63].

The authors showed that this new variant runs in strongly polynomial time on the original network, without any transformation.

Finally, it is worth mentioning that, EPSAs and MBUSAs are quite different from the criss-cross methods. This is due to the fact that, the trace of a criss-cross method is not monotonic with respect to the objective function. Also, both EPSA and MBUSA differ from criss-cross methods, since the formers require a basic feasible solution for their initialization.

3 EPSAs for LPs

The algorithms presented in this section belong to a special "exterior point simplex-type" category. These algorithms differ radically from the classical simplex algorithm. The distinguished characteristic of these algorithms is their flexibility of generating two paths to the optimal solution. One path consists of basic solutions but not feasible while the other is feasible. As a consequence these algorithms avoid the movement between adjacent vertices and, in fact, follows shortcuts to the optimal solution. Although, IPMs are currently the fastest approach for the solution of LPs, simplex-type algorithms remain a good choice for solving LPs. Computational results on randomly generated sparse and dense LPs have shown that EPSA is up to ten times faster than the classical PSA. This computational improvement is due to the essential reduction on the number of iterations.

3.1 Notations and definitions about LP

Consider the following LP in the standard form:

$$\begin{aligned} &\min c^T x \\ &s.t.\ Ax = b \\ &\quad x \geq 0 \end{aligned} \tag{L.P.1}$$

where $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ and T denotes transposition. Moreover, by e we denote a unit column vector, (e.g., the unit row vector is $e^T = (1, 1, ..., 1)$). Without loss of generality, we assume that the coefficient matrix A is of full row rank (rank(A) $= m$, $m < n$). The dual problem associated with (L.P.1) is:

$$\begin{aligned} &\max b^T w \\ &s.t.\ A^T w + S = c \\ &s \geq 0 \end{aligned} \tag{D.P.}$$

where $w \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$. Also, the cardinality of a set P will be denoted by $|P|$.

3.2 Primal EPSA for LPs

Let B be a subset of the index set {1,2,...,n} with $|B| = m$. The partition of the coefficient matrix A as (B, N), where $N = \{1,2,...,n\} \setminus B$ is called a basic partition. The ith row and the jth column of A will be denoted by $A_{i.}$ and $A_{.j}$ respectively. If A is a matrix, A_B denotes the submatrix of A containing the columns $A_{.j}$ such that $j \in B$. Similarly, A_N denotes the submatrix of A containing the columns $A_{.j}$ such that $j \in N$. By partitioning and ordering the vectors x and c, (L.P.1) can be written as:

$$\begin{aligned} &\min\ c_B^T x_B + c_N^N x_N \\ s.t.\ &A_B x_B + A_N x_N = b \\ &x_B, x_N \geq 0 \end{aligned} \tag{L.P.2}$$

If A_B is an $m \times m$ non-singular submatrix of A, then B is called a basis and A_B is a basic matrix. The submatrix A_N is called a non-basic matrix. Consequently, the columns of A which belong to B are called basic and the remaining ones are called non-basic. Given a basis B, the associated solution $x_B = (A_B^{-1})b$, $x_N = 0$ is called a basic solution. Clearly, a basic solution is feasible iff $x_B \geq 0$. Otherwise, it is called infeasible. Geometrically, a basic feasible solution of (L.P.2) corresponds to a vertex of the polyhedral set of the feasible region. The solution of the (D.P.) is given by $s = c - A^T w$, where $w^T = (c_B)^T (A_B)^{-1}$ are the simplex multipliers and s are the dual slack variables. The solution w is feasible to (D.P.) iff $s \geq 0$. The current basis B is optimal if the primal solution (x_B,x_N) is feasible to (L.P.1) and the dual solution (w,s) is feasible to (D.P.).

Each iteration of a simplex-type algorithm requires the computation of the basis inverse $(A_B)^{-1}$. This can be done efficiently if $(A_B)^{-1}$ is maintained in some factorized form. At every iteration its factors have to be updated. The simplest updating scheme is the Product Form of the Inverse (PFI). The current inverse $(A_{\overline{B}})^{-1}$ can be computed from the previous inverse $(A_B)^{-1}$ with a simple pivoting operation. It is well known that:

$$(A_{\overline{B}})^{-1} = E^{-1}(A_B)^{-1} \tag{1}$$

where E^{-1} is the inverse of the eta-matrix. Eta-matrix is an $m \times m$ identity matrix by replacing the r-th column with the column vector $h_l = (A_B^{-1})A_{.l}$. A formal description of EPSA follows:

Step 0 (Initialization). Start with a primal feasible basic partition (B,N). Compute the basis inverse $(A_B)^{-1}$ and the vectors x_B, w, and s_N. Find the sets of indices $P = \{j \in N : s_j < 0\}$ and $Q = \{j \in N : s_j \geq 0\}$. Define an arbitrary vector $\lambda = (\lambda_1, \lambda_2, ..., \lambda_{|P|}) > 0$ and compute $s_0 = \sum_{j \in P} \lambda_j s_j$. Compute the improving feasible direction d_B using the relation:

$$d_B = -\sum_{j \in P} \lambda_j h_j$$

where $h_j = B^{-1}A_{.j}$.

Step 1 (Termination test).

(i) *(Optimality test).* If $P = \emptyset$, STOP. Problem (L.P.2) is optimal.

(ii) *(Choice of leaving variable).* If $d_B \geq 0$, STOP. If $s_0 = 0$, problem (L.P.2) is optimal. Otherwise, choose the leaving variable $x_{B[r]} = x_k$ using the relation:

$$\alpha = \frac{x_{B[r]}}{-d_{B[r]}} = \min\left\{\frac{x_{B[i]}}{-d_{B[i]}} : d_{B[i]} < 0\right\}$$

If $\alpha = +\infty$, problem (L.P.2) is unbounded.

Step 2 (Choice of entering variable). Compute the row vectors $H_{rP} = (A_B)^{-1}_{r.}A_P$ and $H_{rQ} = (A_B)^{-1}_{r.}A_Q$. Also, compute the ratios θ_1 and θ_2 using the relations:

$$\theta_1 = \frac{-s_p}{h_{rp}} = \min\left\{\frac{-s_j}{h_{rj}} : h_{rj} > 0 \wedge j \in P\right\}$$

$$\theta_2 = \frac{-s_q}{h_{rq}} = \min\left\{\frac{-s_j}{h_{rj}} : h_{rj} < 0 \wedge j \in Q\right\}$$

Determine the indices t_1 and t_2 such that $P(t_1) = p$ and $Q(t_2) = q$. If $\theta_1 \leq \theta_2$, set $l = p$. Otherwise, set $l = q$. The non basic variable x_l enters in the basis.

Step 3 (Pivoting). Set $B[r] = l$. If $\theta_1 \leq \theta_2$ set $P = P \backslash \{l\}$ and $Q = Q \cup \{k\}$. Otherwise, set $Q[t_2] = k$. Using the new partition (B, N) where $N = (P, Q)$ update the basis inverse $(A_B)^{-1}$ using relation (1) and compute the vectors x_B, w, and s_N. Also, update the new direction $\overline{d}_B$ using the relation $\overline{d}_B = E^{-1}d_B$. If $l \in P$, set $\overline{d}_{B[r]} = d_{B[r]} + \lambda_l$. Go to Step 1.

A complete proof of correctness of the above algorithm can be found in [97] and [105]. In order to solve general LPs, a modified big-M method with one artificial variable is applied. A presentation of this method can be found in [101]. Furthermore, EPSA can be initiated by any basis (not necessarily feasible) iff the initial d_B direction crosses the feasible region. This can be checked using the following relation:

$$\beta = max\left\{\frac{x_{B[i]}}{-d_{B[i]}} : x_{B[i]} < 0\right\} \leq \alpha = \min\left\{\frac{x_{B[i]}}{-d_{B[i]}} : d_{B[i]} < 0\right\}$$

where $1 \leq i \leq m$. If the above relation holds, then Primal EPSA can be applied directly to (LP.2) even when the initial basic partition (B, N) is not feasible [120].

Computational results on randomly generated sparse and dense LPs show that EPSA is up to 10 times faster than PSA [101]. This translates into corresponding savings in CPU time. The computational results show that EPSA works well in practice. These results indicate that as the problem size increases and the density of problem decreases, the frequency of EPSA outperforming the PSA increases. In [103] new variations of EPSA using three different strategies to enter Phase II are presented. These variations were tested on randomly generated sparse LPs of various sizes and densities. In particular, the best variation of EPSA is about up to 12 times faster than PSA on LPs of size 1,500 $\times$ 1,500. Additionally, for the same LPs, EPSA is about up to 120 times faster than the built-in implementation of PSA in Matlab. Parallelizing simplex-type algorithms is one of the most challenging problems. Recently, a parallel implementation of the revised EPSA [108] on a multicore machine was presented. In this approach the basis inverse is computed in parallel. The parallel implementation uses the Parallel Computing Toolbox of the Matlab R2009a x86 environment. The speed-up gained from the parallelization of the basis inverse is of average 4.72.

3.3 Primal-Dual EPSA for LPs

Forcing the exterior path of an EPSA to become a dual feasible simplex path results in an algorithm free of the following two computational disadvantages: i) It is difficult to construct good moving directions and ii) there is no known way of moving into the interior of the feasible region. This algorithm, PDEPSA, geometrically moves from an exterior dual feasible point towards the interior of the feasible region using either a boundary or an interior starting point. PDEPSA can be described formally as follows.

Step 0 (Initialization). Start with a primal feasible solution y for problem (L.P.2) and a dual feasible basic partition (B, N) to problem (L.P.2). Compute the basis inverse $(A_B)^{-1}$ and the vectors x_B, w, and s_N. Set $P = N$ and $Q = \emptyset$. Compute the direction $d_B = y_B - x_B$.

Step 1 (Termination test).

(i) *(Test of optimality)*. If $x_B \geq 0$, STOP. Problem (L.P.2) is optimal.
(ii) *(Choice of leaving arc)*. Choose the leaving variable $x_{B[r]} = x_k$ using the relation:

$$\alpha = \frac{x_{B[r]}}{-d_{B[r]}} = \max\left\{\frac{x_{B[i]}}{-d_{B[i]}} : d_{B[i]} > 0 \wedge x_{B[i]} < 0\right\}$$

Step 2 (Choice of entering variable). Compute the pivoting row $H_{rN} = (A_B)^{-1}_{r.}A_N$. Choose the entering variable x_l from the relation:

$$\mu = \frac{-s_l}{H_{rl}} = \min\left\{\frac{-s_j}{H_{rj}} : H_{rj} < 0 \wedge j \in N\right\}$$

Step 3 (Pivoting). Set $B[r] = l$ and $Q = Q \cup \{k\}$. If $l \in P$, set $P = P \backslash \{l\}$. Otherwise, set $Q = Q \backslash \{l\}$. Using the new partition (B, N), where $N = (P, Q)$ update the basis inverse $(A_B)^{-1}$ using relation (1) and compute the vectors x_B, w, and s_N. Update the new direction $\overline{d}_B$ using the relation $\overline{d}_B = E^{-1}d_B$. If $l \in P$ and $y_l > 0$, set $\overline{d}_B[r] = \overline{d}_B[r] + y_l$ and $y_l = 0$. Go to Step 1.

A complete proof of correctness of the above algorithm in tableau format can be found in [97,99] and in revised form in [102]. In order to solve general LPs, a modified big-M method with one artificial variable is applied. A presentation of this method can be found in [102]. An efficient modification of PDEPSA, namely PDIPSA [113], traverses across the interior of the feasible region in an attempt to avoid combinatorial complexities of vertex-following algorithms. Also, moving into the interior of the feasible region is translated into an important reduction of number of iterations and CPU time, particularly in LPs, that are degenerate.

Numerical experiments on randomly generated sparse LPs are presented in [102, 115] to verify the practical value of PDEPSA. Compared to the classical PSA, PDEPSA found to be faster both in terms of number of iterations and in terms of CPU time on randomly generated sparse LPs. Also, computational experiments on randomly generated sparse LPs [51] and on a subset of LPs from Netlib [113] have been used to establish the practical value of the PDIPSA. Ploskas and Samaras [107] recently proposed an efficient parallel implementation of a primal-dual exterior point algorithm (PDEPSA) on GPUs in order to take advantage of the GPUs increasing computational capabilities. This is the first parallel exterior point algorithm developed for the solution of a linear programming problem. This algorithm shows superior computational results compared to the sequential implementation of the same algorithm and the sequential MATLABs interior point method on randomly generated optimal sparse and dense linear programming problems and on a set of benchmark problems (netlib, kennington, Meszaros). The maximum speedup gained over MATLABs interior point method is on average 143 on dense LPs and 15 on sparse LPs. The GPU-based PDEPSA is on average 2.30 times faster than MATLAB's interior point method and 32.25 times faster than the CPU-based PDEPSA on a set of benchmark problems (netlib, kennington, Meszaros).

4 EPSAs for the MCNFP

A wide category of network optimization problems is the MCNFP [117]. Furthermore, other well known problems like the TP and the AP are special cases of MCNFP. This section presents a primal and a dual EPSA in Sects. 4.3 and 4.4 respectively. Both algorithms are well defined under the assumption that all the problems, which they are applied to, are not degenerate. However, a method similar to the rule of Bland [22] for the general LP, was applied in their implementation in order to avoid stalling or cycling.

4.1 Efficient solution algorithms for the MCNFP

Several variants of pivoting algorithms for various subclasses of network optimization problems with polynomial number of iterations and with polynomial number of arithmetic operations exist. Regarding the MCNFP, Ahuja and Orlin presented a scaling network Simplex algorithm [5]. The first polynomial time specialization of the Network Primal Simplex Algorithm (NPSA) for the MCNFP using cost-scaling techniques was proposed by Orlin [85]. Other polynomial time dual network simplex algorithms for the MCNP include the works by Orlin et al. [86] and Armstrong and Jin [15]. Currently, the fastest strongly polynomial time algorithms for the capacitated MCNFP are the algorithms by Orlin [84] and Vygen [122]. Although, both algorithms obtain a running time of $O(m \log m(m + n \log n))$, the algorithm by Vygen works directly with the capacitated network without any transformation to transshipment problem.

4.2 Notations and definitions about MCNFP

Let $G = (V, E)$ be a directed network with m nodes and n arcs, where V and E are the sets of nodes and arcs respectively. Each arc $(i, j) \in E$ has a cost c_{ij} which denotes the unit shipping cost along arc (i, j). Associated with each arc (i, j) is also an amount x_{ij} of flow on the arc (contrary to NPSA, x_{ij} might become negative). EPSA algorithms for the MCNFP have been applied to uncapacitated MCNFPs. This means that $l_{ij} \leq x_{ij} \leq u_{ij}$, where $l_{ij} = 0$ and $u_{ij} = +\infty$. We associate with each node $i \in V$ a number b_i which indicates its available amount of supply or demand. Node i will be called a source, sink or transshipment node, depending upon whether $b_i > 0$, $b_i < 0$, or $b_i = 0$ respectively. Source nodes produce more flow than they consume. Sink nodes consume more flow than they produce and transshipment nodes produce as much flow as they consume. We make the assumption that G is a balanced network, that is $\sum\limits_{i \in V} b_i = 0$. The MCNFP can be formulated as the following LP:

$$\begin{aligned} &\min \sum_{(i,j)\in E} c_{ij}x_{ij} \\ &s.t \sum_{k:(i,k)\in E} x_{ik} - \sum_{j:(j,i)\in E} x_{ji} = b_i, i \in V \qquad \text{(N.P.)} \\ &x_{ij} \geq 0, (i, j) \in E \end{aligned}$$

In the above formulation, constraints of type $\sum x_{ki} - \sum x_{jk} = b_k$ are known as the flow conservation equations, while constraints of type $0 \leq x_{ij} \leq +\infty$ are called the flow capacity constraints. In matrix notation MCNFP can be formulated as a linear program of the form $\min\left\{c^T x : Ax = b, l \leq x \leq u\right\}$ where $A \in \mathbb{R}^{m\times n}$, $c, x, l, u \in \mathbb{R}^n$, $b \in \mathbb{R}^m$.

Furthermore, in this section, notations such as paths (directed or not), cycles (directed or not) and trees, are defined as in [20]. The basic solution for the MCNFP can be represented using a tree. Each basis tree of a MCNFP consists of a flow vector x, which is the unique solution of problem. Henceforth, the basis tree of a MCNFP will be denoted by T, while the basic flow of each T will be denoted by $x(T)$. We say that T is a primal feasible tree, iff $x(T) \geq 0$. Otherwise, it will be called a primal infeasible tree. A starting primal or dual feasible tree solution T consists of $n - 1$ directed arcs that form a tree. These arcs and the corresponding flows are called *basic arcs* and *basic variables*, respectively. Similarly, if for a tree-solution T, $s_{ij} \geq 0$ for every arc $(i, j) \notin T$ then it is said to be *dual feasible*. Network Simplex-type algorithms compute basic solutions x and (w, s) which are complementary.

Since a basic solution in Simplex-type algorithms for the MCNFP are represented as tree solutions, we will denote by C the cycle that is created when an arc enters the basic tree T. The notation $\uparrow\uparrow$ or $\uparrow\downarrow$ will used for arcs that have the same or the opposite orientation to each other in the cycle C, respectively. For simplicity reasons, from now on the leaving arc and the entering arc will be denoted by (k, l) and (g, h) respectively. Moreover, each non basic arc h_{ij} can be represented in terms of the basic arcs as a column vector of the classical network simplex tableau. An element $h_{gh}(k, l)$ of this column vector h_{gh} is equal to -1, 1, or 0, depending whether for the non basic arc (g,h) it holds $(g,h) \uparrow\uparrow (k,l)$ in C, or $(g,h) \uparrow\downarrow (k,l)$ in C, or $(k,l) \notin C$ respectively. Detailed description of these computations can be found in [20].

Finally, by T^* we will denote the tree that contains the nodes that are "cut off" from the current tree solution, when the leaving arc is discarded. In simplex-type algorithms only the reduced cost variables of arcs with one node belonging to the tree T^* and the other to the subtree $T \backslash T^*$ usually need to be updated.

4.3 Primal EPSA for the MCNFP

A Primal Network Exterior Point Simplex-type Algorithm (NEPSA) was presented in [100]. The main difference between NEPSA and the classical Network Primal Simplex Algorithm (NPSA) is that the basic flow of the former does not always remain feasible. NEPSA constructs two flows to the optimal solution. One flow is basic but not always feasible; so this is an "exterior flow". The other flow is feasible but not always basic. Contrary to NPSA, NEPSA first selects a leaving arc and afterwards selects an entering arc. A description of NEPSA follows.

Step 0 (Initialization). Start with a primal feasible tree T, which can be constructed using the well known big M method [20]. Compute the flows x_{ij}, the node potentials (dual variables) w_i, and the reduced costs s_{ij}, as described in [3]. Continue with a partition of the initial arc set E into three subsets ($E = T \cup P \cup Q$), as follows: $P = \{(i,j) : s_{ij} < 0\}$, $Q = \{(i,j) : s_{ij} \geq 0\}$, $\forall\ (i,j) \notin T$. Next, compute the column vectors h_{ij} and the "complemental" flow vector d, which assists NEPSA to compute the second feasible but not always basic flow. In a similar way, vector d is partitioned as $d = \{d(T), d(P), d(Q)\}$. The elements of d are computed using the following relations:

$$d(T) = -\sum_{(i,j)\in P} h_{ij}, \quad d(P) = (\lambda_1, \lambda_2, ..., \lambda_{|P|}), \quad d(Q) = (0, 0, ..., 0).$$

Step 1 (Termination test). If $P = \emptyset$, then an optimal solution has been found for problem (N.P.) and the algorithm terminates. Otherwise, the algorithm continues. If $\exists (i,j) \in T : d_{ij} < 0$, then the algorithm proceeds to the next step. Otherwise the problem (N.P.) is unbounded and therefore negative cost cycles have been identified.

Step 2 (Choice of leaving arc). Choose the leaving arc (k,l) using the following minimum ratio test:

$$a = \frac{x_{kl}}{-d_{kl}} = \min\left\{\frac{x_{ij}}{-d_{ij}} : (i,j) \in T,\ d_{ij} < 0\right\}$$

Compute the second flow $y = x + ad$. This second flow remains feasible, but not always basic, since y corresponds to a network and not to a tree. Elements of vector $\lambda = (\lambda_1, \lambda_2, ..., \lambda_{|P|})$ can have any value, so long as vector d has appropriate values in order to compute a feasible flow y. In our implementation, vector λ is initialized using the values $\lambda = (1, 1, ..., 1)$.

Step 3 (Choice of entering arc). Find the minimum ratios:

$$\theta_1 = -s_{p_1 p_2} = \min\left\{-s_{ij} : h_{ij}(k,l) = 1 \wedge (i,j) \in P\right\}$$

$$\theta_2 = s_{q_1 q_2} \min\left\{s_{ij} : h_{ij}(k,l) = -1 \wedge (i,j) \in Q\right\}$$

Choose the entering arc $(g,h) = (p_1, p_2)$ or (q_1, q_2), in case that $\theta_1 \leq \theta_2$ or not respectively.

Step 4 (Pivoting). If $\theta_1 \leq \theta_2$, then the pivot will be called “Type A iteration”. In this case since the entering arc will be $(g,h) = (p_1, p_2)$, set $P = P \setminus (p_1, p_2)$. Otherwise, the pivot will be called “Type B iteration”. In the latter case since the entering arc will be $(g,h) = (q_1, q_2)$, set $Q = Q \setminus (q_1, q_2)$. Using the new partition $(T\ \ P\ \ Q)$, where $T = T \setminus (k,l) \cup (g,h)$ and $Q = Q \cup (k,l)$ the values x_{ij}, s_{ij}, h_{ij} and d_{ij} are updated. The new $\overline{d}$ vector can be updated using the relation $\overline{d} = y - \overline{x}$. However, if $(g,h) \in P$, $\overline{d}_{gh} = \overline{d}_{gh} + \lambda_{gh}$. A faster update method for the d vector, is described in [100].

4.4 Dual EPSA for the MCNFP

A Dual Exterior Point Simplex-type Algorithm for the MCNFP (DNEPSA) have been recently developed. A preliminary geometrical interpretation of DNEPSA was described in [49]. Furthermore, the mathematical proof of correctness of DNEPSA, a detailed comparative computational study of DNEPSA and DNSA on sparse and dense random problem instances, a statistical analysis of the experimental results, and finally some new results on the empirical complexity of DNEPSA, were published in [50]. The analysis proves the superiority of DNEPSA compared to DNSA in terms of cpu time and iterations. The main difference between DNEPSA and the classical Dual Network Simplex Algorithm (DNSA), is the fact that the tree-solutions formed during the iterations of DNEPSA are not necessarily always dual feasible. In other words, DNEPSA starts from a dual feasible tree solution T and reaches an optimal solution by following a route consisting of solutions that do not always belong to the feasible area of the dual solution. Furthermore, DNEPSA contrary to the DNSA first selects the entering arc and afterwards selects the leaving arc. Finally, both entering and leaving arcs are selected using different rules for the DNEPSA and DNSA algorithms respectively. Various methods can be used to find a starting dual feasible tree solution. An algorithm that can construct a dual feasible tree-solution for the generalized network problem (and also for pure networks) is described in [53], and an improved version of the algorithm is presented in [60], which gives a dual feasible solution that is closer to an optimal solution.

Step 0 (Initialization). Compute the flows x_{ij}, the node potentials (dual variables) w_i, and the reduced costs s_{ij}, as described in [3]. Start with a partition of the basic arcs into two subsets as follows $I_- = \{(i,j) \in T : x_{ij} < 0\}$ and $I_+ = \{(i,j) \in T : x_{ij} \geq 0\}$. Compute a direction flow vector d, towards to the feasible region of the dual problem, using the following relations:

$$d(I_-) = 1,\ d(I_+) = 0,\ d_{ij} = \sum_{(u,v)\in I_-} h_{uv}, \forall (i,j) \notin T$$

Step 1 (Termination test). If $I_- = \emptyset$, this means that the current tree-solution is optimal. Otherwise, create a set $J_- = \{(i, j) \notin T : s_{ij} > 0 \wedge d_{ij} < 0\}$. If $(J_- = \emptyset) \wedge (I_- \neq \emptyset)$, then the problem is infeasible.

Step 2 (Choice of entering arc). Choose the entering arc (g, h) using the following minimum ratio test:

$$\alpha = \frac{s_{gh}}{-d_{gh}} = \min\left\{ \frac{s_{ij}}{-d_{ij}} : (i, j) \in J_- \right\}$$

Step 3 (Choice of leaving arc). Find the minimum ratios:

$$\theta_1 = -x_{k_1 l_1} = \min\left\{-x_{ij} : (i, j) \in I_- \wedge (i, j) \uparrow\uparrow (g, h)\right\}$$

$$\theta_2 = x_{k_2 l_2} = \min\left\{x_{ij} : (i, j) \in I_+ \wedge (i, j) \uparrow\downarrow (g, h)\right\}$$

Choose the leaving arc $(g, h) = (k_1, l_1)$ or (k_2, l_2), in case that $\theta_1 \leq \theta_2$ or not respectively.

Step 4 (Pivoting). If $\theta_1 \leq \theta_2$, the pivot will be called “Type A iteration”. In this case, set $I_- = I_- \setminus (k, l)$ and $I_+ = I_+ \cup (g, h)$. Otherwise, the pivot will be called “Type B iteration”. In this case, set $I_+ = I_+ \cup (g, h) \setminus (k, l)$. It is not necessary for the algorithm in every iteration to compute the values x_{ij}, s_{ij}, and d_{ij} or to create sets I_- and I_+ from scratch. These variables and sets can be efficiently updated from iteration to iteration as described in [49].

Preliminary computational results [50] have shown that DNEPSA is about 1.22 times faster than DNSA in terms of the number of iterations and about 1.57 times faster in terms of the CPU time. This computational study was based on several random generated MCNFP instances by NETGEN with varying density ranging from sparse problems (2%) to dense problems (40%), varying number of nodes from 200 to 700, and varying number of arcs approximately from 800 to 195,000. Although, these results have been based only on random problems, they clearly illustrate a very promising computational performance. However, more computational results that are: i) based on standard benchmark problems (e.g., NETLIB), and ii) use the polynomial version of DNSA, will help in order to derive more general conclusions.

5 EPSAs for the transportation problem (TP)

5.1 Efficient solution algorithms for the TP

The simplex algorithm for solving the TP was adapted by Dantzig in [38]. A few years later Charnes and Cooper presented a more easily comprehended approach, known as the stepping-stone method [27]. Also, several methods have been proposed in the literature for finding an near-optimal basis for the TP, such as the method by Russel [112]. The importance of finding such quality initial solutions for the TP (as also for any optimization problem) is commonly accepted, since it permits us to reduce the time required to find the optimal solution. Arshom and Khan [18] proposed a simple and fast method for solving general TPs, as an alternative to the previous methods [27], [38]. Also, Portugal et al. [109] presented a computational investigation of interior-point algorithms for the Linear TP. However, since the TP is a special case of the MCNFP, several polynomial-time algorithms for the MCNFP can also be used for efficiently solving the TP.

5.2 Notations and definitions about the TP

A TP is a special case of the MCNFP and can be represented by a bipartite graph $G(S, D, E) = G(V, E)$, where S, D are two disjoint sets of nodes such $|S| = m$, $|D| = n$ and $V = S \cup D$. Notations $|S|$ and $|D|$ stand for the cardinality number of the sets S and D respectively. The supply and demand nodes are denoted with $i \in S$ and $j \in D$ respectively. Here, E is the arc set. An arc $(i, j) \in E$ is directed from nodes of S to nodes of D. The mathematical formulation of the TP with an $m \times n$ cost matrix c, an $m \times 1$ supply vector a and an $n \times 1$ demand vector b such as $\sum\limits_{i \in S} a_i = \sum\limits_{j \in D} b_i$ is as follows:

$$\begin{aligned} &\min \sum_{(i,j)\in E} c_{ij} x_{ij} \\ &s.t. \sum_{(i,j)\in E} x_{ij} = a_i, i \in S \\ &\quad \sum_{(i,j)\in E} x_{ij} = b_j, j \in D \\ &x_{ij} \geq 0, i \in S, j \in D \end{aligned} \qquad \text{(T.P.)}$$

The decision variables x_{ij} denote the number of product units transported from the supply node $i \in S$ to the demand node $j \in D$. Each basic solution to TP corresponds to a spanning tree of the bipartite network $G = (S, D, E)$. Each basic solution T is associated to the decision variables $x_{ij}(T), \forall (i, j) \in E$, the dual row (supply) variables $u_i(T), \forall i \in S$, the dual column (demand) variables $\upsilon_j(T), \forall j \in D$, and the reduced cost variables $s_{ij}(T) = c_{ij} - u_i(T) - \upsilon_j(T), \forall (i, j) \in E$.

Furthermore, by $s_{i.}$ we denote a row vector corresponding to the i row of that ($m \times n$) matrix. For example, by $s_{5.}$ we denote a row vector containing all the reduced cost variables which corresponds to arcs leaving from the fifth supply node. Therefore, $s_{5.} = (s_{51}, s_{52}, s_{53}, ..., s_{5n})$. Similarly, by $s_{.j}$ we denote a column vector corresponding to the j column of that ($m \times n$) matrix. For example, by $s_{.5}$ we denote a column vector containing all the reduced cost variables which corresponds to arcs coming to the fifth demand node. Therefore, $s_{.5}^T = (s_{15}, s_{25}, s_{35}, ..., s_{m5})$.

5.3 Dual EPSAs for the TP

An exterior point algorithm has been developed for the TP by Paparrizos [98]. Furthermore, signature methods for the TP have also been reported in [93, 41]. EPSA for the TP may be initialized either with (1) a Balinski's dual feasible tree [94], (2) a simple start forest [1] that is neither primal nor dual feasible, and (3) a dual feasible forest [2]. The latter algorithm requires at most $\mu D - \frac{\mu(\mu-1)}{2}$ iterations, and has a

$O(\mu(m+n)D)$ worst case complexity, where $\mu = \min\{m,n\}$ and D is the total demand. A description of the EPSA for the TP, follows:

Step 0 (Initialization). Start with a special solution T. The set V of nodes is partitioned into two subsets F and $T \setminus F$, that depend on the initialization method. Determine the subsets $F, T \setminus F$ and compute the reduced cost variables s_{ij} according to the initialization method.

Step 1 (Test of optimality). If $F = \emptyset$, this means that the current solution T is optimal for problem (T.P.).

Step 2 (Choice of entering arc). Choose the entering arc (g,h) by $s_{gh} = \delta = \min\left\{s_{ij} : i \in F \wedge j \in T \backslash F\right\}$

Step 3 (Choice of leaving arc). Choose the leaving arc (k,l) according to the special rules of each initialization method.

Step 4 (Pivoting). Set $\overline{T} = T \cup (g,h) \backslash (k,l)$. Then if $h \notin T^*$, for every row node $i \in T^*$ and column node $j \in T^*$ set $s_{i.}(\overline{T}) = s_{i.}(T) + \delta e^T$, and $s_{.j}(\overline{T}) = s_{.j}(T) + \delta e$ respectively. Finally, update F, $T \backslash F$ and set $\overline{T} = T$.

The differences of these three different initialization methods, are due to the way their starting tree structures T are initialized, the way subset F is constructed, and the way the leaving arc is chosen. Moreover, a computational study of a dual exterior point algorithm for the TP, using randomly generated TPs, was presented in [89]. In that study it was shown that, the initialization method of a dual feasible forest [2] had the best performance, followed by the Balinski's dual feasible tree [94]. The third initialization method of simple start forest [1] was the worst among the three compared methods on all test instances. However, each one initialization method was proved to be faster than the classic PSA for the TP. More precisely, the dual feasible forest [2] initialization in TPs of size 300×300, was about 4.5 and 4.86 times faster than the classic PSA for the TP, in terms of number of iterations and CPU time respectively.

5.3.1 EPSA with a Balinski's dual feasible tree for the TP

Balinski tree is dual feasible. Its root is row–node 1 and all column nodes lie below the root at depth 1. The remaining row nodes are connected to the column nodes of the tree and thus lie at depth 2. Let T denote the Balinski tree. The standard arcs of the tree are the arcs of type $(1, j)$, where j is a column node. Initially, we set $u_1(T) = 0$ and $\upsilon_j(T) = c_{1j}$, $j = 1,2,...,n$. Thus, for each arc $(1, j)$ we have $s_{1j}(T) = c_{1j} - u_1(T) - \upsilon_j(T) = 0$. Given a row index i, the column index associated with the minimum $c_{ij} - \upsilon_j$ value of row i is $j(i) = argmin\{c_{ij} - \upsilon_j : j = 1,2,...,n\}$. The remaining dual variables $u_i(T)$ can be computed by setting $u_i(T) = \min\left\{c_{ij} - \upsilon_j(T) : j = 1,...,n\right\}, i = 1,...,m$.

Let now $u_i(T) = c_{ij(i)} - \upsilon_{j(i)}, i = 2,3,...,n$. Then $(i, j(i)) \in T$. At this point all arcs belonging to the Balinski tree have been determined and the reduced costs of the non-basic variables can now be easily computed. The decision variables x_{ij} corresponding to basic arcs (i, j) of the tree are computed easily by setting $x_{ij} = \alpha(i), i = 2,...,n$ and $x_{1j} = b(j)$, if j is a leaf of the tree. For the remaining arcs of type $(1, j)$ (where column node j has at least one child), we set $x_{1j} = b(j) - \sum_{(i,j)\in T, i\neq 1} x_{ij}$. Finally, the forest F is the set of trees $\{T_j : x_{1j} < 0\}$, where T_j is the subtree rooted on column node j. Obviously, $T \backslash F$ is a subtree of T.

5.3.2 EPSA with a simple start forest for the TP

Let F denote this forest. We partition the nodes into two sub-forests F^S and F^D. The set F^S consists of all the row-nodes, while the set F^D of all the column nodes. Additionally, we set $u_i(F^S) = \upsilon_j(F^D) = 0, i \in S, j \in D$. Thus, we have $s_{ij}(F) = c_{ij}$. To adjust the forest to the needs of our algorithm, we need to transform the forest F to a tree T. This can easily be done by inserting an artificial node 0 (which is the root of tree T) and adding $m + n$ artificial arcs. We also set $u_0(T) = 0$, for the artificial transshipment node. For each row node i an artificial arc $(i, 0)$ with unit cost $c_{i0} = 0$ is introduced. Similarly, for each column node j an artificial arc $(0, j)$ with unit cost $c_{0j} = 0$ is introduced. All the basic decision variables of type $x_{i0}, i = 1, 2, ..., n$ are initially set equal to $x_{i0} = \alpha(i) > 0$. Similarly, all the basic decision variables of type $x_{0j}, j = 1, 2, ..., n$ are initially set equal to $x_{0j} = b(j) > 0$. Moreover, the dual exterior point Simplex-type algorithm for the AP updates the sets F^S, F^D in such a way that $F = \{T_i : i \in S\}$ and $F^D = \{T_j : j \in D\}$.

5.3.3 EPSA with a dual feasible forest for the TP

The initial forest F consists of small trees, each one of them rooted on a column node. By T_j we denote the tree belonging in forest F, that is rooted on column node j. We partition the trees T_j into two sets F^S and F^D. The set F^S consists of all the row-nodes, while the set F^D of all the column nodes. Additionally, we set $\upsilon_j(F) = 0, j \in D$ and $u_i(F) = \min\left\{c_{ij} : j = 1, ..., n\right\}, i \in S$. The forest will include basic arcs (i, t), where t is a column node computed from the equation $u_i(F) = c_{it}$. Each such arc (i, t) will have $x_{ij} = \alpha(i)$. Since $s_{ij}(F) \geq 0$, the forest F is dual feasible. Initially we set: $F^S = \{T_j : b_j < \sum\limits_{i \in T_j} a_i\}$ and $F^D = \{T_j : \sum\limits_{i \in T_j} a_i \leq b_j\}$. To adjust the forest to the needs of our algorithm, we need to transform the forest F to a tree T. This can easily be done by inserting an artificial transshipment node 0 (which is the root of tree T) and adding n artificial arcs. If $T_j \in F^S$, then T includes an artificial arc $(j, 0)$ with unit cost $c_{j0} = 0$, such that $x_{j0} = \sum\limits_{i \in T_j} \alpha(i) - b(j) > 0$. If on the contrary, $T_j \in F^D$, then T includes an artificial arc $(0, j)$ with unit cost $c_{0j} = 0$, such that $x_{0j} = b(j) - \sum\limits_{i \in T_j} \alpha(i) \geq 0$.

6 EPSAs for the assignment problem (AP)

6.1 Efficient solution algorithms for the AP

Several algorithm categories exist for the AP, such as the primal-dual, simplex type, cost operator, recursive, forest, and interior point algorithms. Kuhn presented the first non-simplex algorithm for the AP, known as the Hungarian method. Goldfarb [54] presented special variants of the dual simplex method, that are based upon Balinski's signature method [19], and have computational bounds of $O(n^3)$ for solving the AP. Other polynomial algorithms for the AP include the works of Hung [61] and Glover et al. [52]. Also, Akgül in [6] presented a genuinely polynomial primal simplex algorithm for solving the AP in at most $\frac{1}{2}n(n + 3) - 4$ pivots. Furthermore, this algorithm can be implemented to run in $O(n^3)$ time for dense graphs and $O(n^2 \log n + nm)$ for sparse graphs.

6.2 Notations and definitions about the AP

The AP is a special case of the TP, where $m = n$, $\sum_{i \in S} a_i = 1$, and $\sum_{j \in D} b_i = 1$, [25]. The mathematical formulation of the AP is as follows:

$$
\begin{aligned}
&\min \sum_{(i,j)\in E} c_{ij}x_{ij} \\
&s.t. \sum_{(i,j)\in E} x_{ij} = 1, i \in S \qquad\qquad \text{(A.P.)} \\
&\quad \sum_{(i,j)\in E} x_{ij} = 1, j \in D \\
&x_{ij} \in \{0,1\}, (i,j) \in E
\end{aligned}
$$

The AP can be represented by a bipartite graph $G(S, D, E) = G(V, E)$, where S, D are two disjoint sets of nodes such that $|S| = |D| = n$ and $V = S \cup D$.

6.3 Dual EPSAs for the AP

Paparrizos [94] developed the first exterior point algorithm algorithm for the AP. This algorithm has a $O(n^3)$ worst case complexity and solves an $n \times n$ AP in at most $\frac{n(n-1)}{2}$ iterations. A systematic procedure to construct worst case examples for this algorithm was presented in [92]. EPSA for the AP may be initialized either with (1) a Balinski's dual feasible tree [94], (2) a simple start forest [1] that is neither primal nor dual feasible, and (3) a dual feasible forest [2]. A description of the EPSA for the AP, follows:

Step 0 (Initialization). Start with a special solution T. The set V of nodes is partitioned into two subsets F and $T \setminus F$, that depend on the initialization method. Determine the subsets $F, T \setminus F$ and compute the reduced cost variables s_{ij} according to the initialization method.

Step 1 (Test of optimality). If $F = \emptyset$, this means that the current solution T is optimal for problem (A.P.).

Step 2 (Choice of entering arc). Choose the entering arc (g, h) by $s_{gh} = \delta = \min\left\{s_{ij} : i \in F \wedge j \in T\backslash F\right\}$

Step 3 (Choice of leaving arc). Choose the leaving arc (k, l) according to the special rules of each initialization method.

Step 4 (Pivoting). Set $\overline{T} = T \cup (g, h)\backslash(k, l)$. Then set $q = -\delta$ or $q = \delta$ depending whether $h \in T^*$ or $h \notin T^*$ respectively. Then for each row node $i \in T^*$ and for each column node $j \in T^*$, set $s_{i.}(\overline{T}) = s_{i.}(T) - qe^T$ and $s_{.j}(\overline{T}) = s_{.j}(T) + qe$ respectively. Finally, update F, $T\backslash F$ and set $\overline{T} = T$.

The differences of these three different initialization methods, are due to the way their starting tree structures T are initialized, the way subset F is constructed, and the way the leaving arc is chosen.

Two computational studies using benchmark and randomly generated APs, on the initialization methods of the EPSA for the AP were presented in [91] and [114] respectively. These experimental evaluations of the three initialization methods showed the same ranking. More precisely, the initialization method of a dual feasible forest [2] showed the best performance, followed by the simple start forest [1]. The third initialization method of Balinski's dual feasible tree [94] was the worst among the three compared methods on all test instances. More precisely, the computational study using randomly generated APs, showed that the initialization method of a dual feasible forest was constantly faster independently of the problem dimension. This constant factor varied between 1.58 and 1.72 in terms of number of iterations and between 1.66 and 1.58 in terms of CPU time. Additionally, the computational study using benchmark problems from TSPLib95 [110] showed that the third initialization method of a dual feasible forest was about 1.940 times faster than the second initialization method of a simple start forest in terms of the number of iterations and about 2.105 times faster in terms of the CPU time. In the same study it was shown that, the third initialization method of a dual feasible forest was about 3.176 times faster than the second initialization method of the Balinski's dual feasible tree in terms of the number of iterations and about 3.238 times faster in terms of the CPU time.

The following three initialization methods, consist specializations of the three initialization methods for the TP, especially for the AP.

6.3.1 EPSA with a Balinski's dual feasible tree for the AP

Balinski tree is dual feasible. Its root is row - node 1 and all column nodes lie below the root at depth 1. The remaining row nodes are connected to the column nodes of the tree and thus lie at depth 2. Let T denote the Balinski tree. The standard arcs of the tree are the arcs of type $(1, j)$, where j is a column node. Initially, we set $u_1(T) = 0$ and $\upsilon_j(T) = c_{1j}$, $j = 1, 2, ..., n$. Thus, for each arc $(1, j)$ we have $s_{1j}(T) = c_{1j} - u_1(T) - \upsilon_j(T) = 0$. Given a row index i, the column index associated with the minimum $c_{ij} - \upsilon_j$ value of row i is $j(i) = argmin\{c_{ij} - \upsilon_j : j = 1, 2, ..., n\}$. The remaining dual variables $u_i(T)$ can be computed by setting $u_i(T) = \min\left\{c_{ij} - \upsilon_j(T) : j = 1, ..., n\right\}, i = 1, ..., n$.

Let now $u_i(T) = c_{ij(i)} - \upsilon_{j(i)}, i = 2, 3, ..., n$. Then $(i, j(i)) \in T$. At this point all arcs belonging to the Balinski tree have been determined and the reduced costs of the nonbasic variables can now be easily computed. The decision variables x_{ij} corresponding to basic arcs (i, j) of the tree are computed easily by setting $x_{ij} = 1, i = 2, ..., n$ and $x_{ij} = 1$, if j is a leaf of the tree. For the remaining arcs of type $(1, j)$ (where column node j has at least one child), we set $x_{1j} = 1 - m_j \leq 0$, where m_j is the number of children of column node j. Finally, the forest F is the set of trees $\{T_j : x_{1j} < 0\}$, where T_j is the subtree rooted on column node j. Obviously, $T \backslash F$ is a subtree of T.

6.3.2 EPSA with a simple start forest for the AP

Let Q be the simple start forest. Forest Q consists of the $2n$ isolated nodes. The set F consists of all the row-nodes, while the set $T\backslash F$ of all the column nodes. Thus, $F = \{i : i \in S\}$ and $T\backslash F = \{j : j \in D\}$. Additionally, we set $u_i(Q) = \upsilon_j(Q) = 0, i \in S, j \in D$. Thus, we have $s_{ij}(Q) = c_{ij}$. To adjust the forest to the needs of our algorithm, we need to transform the forest Q to a tree T. This can easily be done by inserting an artificial node 0 (which is the root of tree T) and adding $2n$ artificial arcs. For each row node i an artificial arc $(i, 0)$ with unit cost $c_{i0} = 0$ is introduced. Similarly, for each column node j an artificial arc $(0, j)$ with unit cost $c_{0j} = 0$ is introduced. All the basic decision variables of type $x_{i0}, i = 1, 2, ..., n$ and $x_{0j}, j = 1, 2, ..., n$ are initially set equal to 1. Moreover, the dual exterior point Simplex-type algorithm for the AP updates the sets F, $T\backslash F$ in such a way that $F = \{T_i : i \in S, (1, 0) \in T\}$ and $T\backslash F = \{T_j : j \in D\}$.

6.3.3 EPSA with a dual feasible forest for the AP

Let Q denote this forest. Forest Q consists of n subtrees T_j and is computed as follows: Let $u_i(Q), \upsilon_j(Q), i, j = 1, 2, ..., n$, denote the dual variables that correspond to the column and the row nodes of the forest respectively. Initially we set $\upsilon_j(Q) = 0, \forall j \in D$. In succession, we compute the dual variables of the column nodes $u_i(Q)$ by setting $u_i(Q) = \min\{c_{ij} : j = 1, ..., n\}, \forall i \in S$.

Let t be the column node such that $u_i(Q) = c_{it}$. Then, the arc (i, t) is a basic arc of the forest Q. For each arc (i, t) we set $x_{it} = 1$. The reduced costs s_{ij} can now easily be computed. Note that $s_{ij}(Q) \geq 0$, which means that forest Q is dual feasible. The initial set F, is $F = T_j : d(j) \geq 2$ where $d(j)$ is the degree of column node j. Accordingly, $T\backslash F = T_j : d(j) < 2$. For the same reason referred at the simple start method, we need to transform the forest Q to a tree T. This can easily be done by inserting an artificial node 0 (root of the tree T) and n artificial arcs. For each column node j such that $T_j \in F$ (j is the root of subtree T_j) an artificial arc $(j, 0)$ with unit cost $c_{j0} = 0$ is introduced. Similarly, for each column node $j \notin F$ an artificial arc $(0, j)$ with unit cost $c_{0j} = 0$ is introduced. Finally, it is set $x_{0j}(T) = |d(j) - 2|$ and $x_{j0}(T) = |d(j) - 2|$, where $d(j)$ is the degree of node j after the insertion of the artificial arcs.

7 Web-based educational software packages for EPSAs

It is well known that the traditional face-to-face teaching of algorithms, for the solution of optimization problems is considered difficult by most students. Since the improvement of OR teaching is always a goal for any OR instructor [9,29,77], e-learning technologies have become a very active research area for many OR scientists [40,58,69,76].

Visualization software tool for NEPSA has been implemented by Andreou et al. [11] (available at: `http://users.uom.gr/~sifalera/ORIJ`). Furthermore other didactic applications, utilizing animation and visualization techniques, have been developed for EPSAs for the AP by Andreou et al. [10] and by Papamanthou et al. [90]

(available online at: `http://www.assignmentproblems.com/linearAPdidactic.htm#links` and `http://users.uom.gr/~samaras/gr/yujor/yujor.htm`).

8 Conclusions and open research problems in EPSAs

The importance of the results of the papers on the new variants of the criss-cross algorithms is that, an algorithm, i.e. CCA, has such variants which are finite for the whole, quite general class of problems with several different anti-cycling rules. This leads to an interesting question: *Could such variants of EPSA or MBUSA be defined that solve QP or some more general classes of Linear Complementarity Problems (LCP) in finite number of iterations*? An equally important open research question is *whether the DNEPSA variant has a polynomial variant or not*.

Additionally, the finiteness of the original quadratic simplex method developed by Wolfe [123] was ensured by the means of perturbation. Recently, Illés & Nagy have proved that the quadratic primal simplex algorithm is finite when s-monotone index selection rules are applied [65]. Thus, another interesting research direction, as it is illustrated in several papers [7], [32], [34], [45], [46], [39], is *whether a variant of EPSA (MBUSA) exist that solves QP (or sufficient LCP) in finite number of steps?*

Goldfarb et al. [56] analyzed two pivot rules, the Inward Most Negative (IMN) and the Altered IMN (AIMN) rules, for specializing the primal simplex algorithm for the shortest path problem. Both versions require at most $O(n^2)$ pivots and $O(n^3)$ time. Also, Goldfarb & Hao [55], presented the first strongly polynomial primal simplex algorithm for solving the maximum flow problem. Thus, another important question that deserves to be investigated is *whether an EPSA (or MBUSA) variant exist for maximal flow problem (or MCNFP) with polynomial worst case complexity?*. Some preliminary work on this question has been already reported in [63], for feasibility sub-problem of maximal flow problem with non-zero lower bounds. Also, other interesting future research direction is the practical and theoretical comparison of these polynomial variants of Simplex Algorithms (SAs) with the best variants of EPSAs (or MBUSAs).

Also, fractional linear programming problems constitute another important class of optimization problems that can be solved by applying pivot algorithms. This problem class and a variant of CCA for fractional linear programming problems have been studied by Illés et al. in [66]. Following the ideas presented in this paper, one more interesting question that deserves further investigation is *whether an EPSA (or MBUSA) variant exist for solving fractional linear programming problems in finite number of iterations?*

It is well known that simplex-type algorithms' computational behavior can be improved by modifying: (1) the initial solution and (2) the pivoting rule. Although EPSA outperforms clearly the PSA, it has two major computational disadvantages. These are:

– It is difficult to construct "good moving directions". The two paths generated by EPSA depend on the initial feasible direction, that is closely related to the initial feasible vertex.

– There is no known way of moving into the interior of the feasible region. This movement will provide more flexibility in the searching of computationally good directions.

However, more comprehensive computational investigations of EPSAs are still required to be done on LP problems selected from NETLIB repository [48]. Furthermore, the performance of exterior-point algorithms for the MCNFP (i.e., NEPSA and DNEPSA) can be improved by using special data structures for storing and updating the necessary variables, (e.g., Fibonacci heaps). Moreover, it would be interesting to develop a capacitated version of NEPSA and DNEPSA. Finally, another future research direction is the parallel implementation of EPSAs. Parallelizing Simplex-type algorithms is one of the most challenging problems.

Acknowledgements It is with sorrow that we note that, during the preparation of this paper, the first author Prof. Konstantinos Paparrizos, suddenly and unexpectedly passed away. Prof. Konstantinos Paparrizos was a well respected and loved colleague for the Operations Research community, at the international level, for his excellent scientific activity and sincerity. He was a very active researcher and he had successfully stimulated many young scientists. The main research results of his career, together with contributions from the other two co-authors who were honored to be supervised by him during their Ph.D. studies, are presented in this paper, which we dedicate to his memory. Finally, we would also like to express our gratitude to the referees, whose their instructive comments greatly improved the quality of the paper.

References

1. Achatz, A., Paparrizos, K., Samaras, N., Tsiplidis, K.: A forest exterior point algorithm for assignment problems. In: M.P. Pardalos, A. Midgalas, R. Buckard (Eds.) Combinatorial and Global Optimization, chap. Combinatorial and Global Optimization, pp. 1–10. Singapore: Word Scientific (2002)
2. Achatz, H., Kleinschmidt, P., Paparrizos, K.: A dual forest algorithm for the assignment problem. In: Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift, vol. 4, pp. 1–10. London: AMS & ACM (1991)
3. Ahuja, R. K., Magnanti, T. L., Orlin, J. B.: Network Flows: Theory, Algorithms and Applications. Englewood Cliffs, NJ: Prentice Hall (1993)
4. Ahuja, R. K., Magnanti, T. L., Orlin, J. B., Reddy, M. R.: Applications of network optimization. In: M.O. Ball, T.L.M.C.L. Monma, G.L. Nemhauser (Eds.) Network Models, vol. 7, chap. Handbooks of Operations Research and Management Science, pp. 1–83. Amsterdam: Elsevier, North-Holland (1995)
5. Ahuja, R. K., Orlin, J. B.: The scaling network simplex algorithm. Operations Research **40**(Supp. No. 1), S5–S13 (1992)
6. Akgül, M.: A genuinely polynomial primal simplex algorithm for the assignment problem. Discrete Applied Mathematics **45**(2), 93–115 (1993)
7. Akkeleş, A. A., Balogh, L., Illeś, T.: New variants of the criss-cross method for linearly constrained convex quadratic programming. European Journal of Operational Research **157**(1), 74–86 (2004)
8. Al-Sultan, K.S., Murty, K. G.: Exterior point algorithms for nearest points and convex quadratic programs. Mathematical Programming **57**, 145–161 (1992)
9. Alpers, A., Trotter, L. E.: Teaching computational discrete optimization at the undergraduate level. INFORMS Transactions on Education **9**(2), 63–69 (2009)
10. Andreou, D., Paparrizos, K., Samaras, N., Sifaleras, A.: Application of a new network-enabled solver for the assignment problem in computer-aided education. Journal of Computer Science **1**(1), 19–23 (2005)
11. Andreou, D., Paparrizos, K., Samaras, N., Sifaleras, A.: Visualization of the network exterior primal simplex algorithm for the minimum cost network flow problem. Operational Research. An International Journal **7**(3), 449–464 (2007)

12. Andrus, J., Schaferkotter, M.: An exterior-point method for linear programming problems. Journal of Optimization Theory and Applications **91**, 561–583 (1996)
13. Anstreicher, K. M., Terlaky, T.: A monotonic build-up simplex algorithm for linear programming. Tech. Rep. 91-82, Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands (1991)
14. Anstreicher, K. M., Terlaky, T.: A monotonic build-up simplex algorithm for linear programming. Operations Research **42**(3), 556–561 (1994)
15. Armstrong, R. D., Jin, Z.: A new strongly polynomial dual network simplex algorithm. Mathematical programming **78**(2), 131–148 (1997)
16. Arsham, H.: A hybrid gradient and feasible direction pivotal solution algorithm for general linear programs. Applied Mathematics and Computation **188**(1), 596–611 (2007)
17. Arsham, H., Cimperman, G., Damij, N., Damij, T., Grad, J.: A computer implementation of the push-and-pull algorithm and its computational comparison with LP simplex method. Applied Mathematics and Computation **170**(1), 36–63 (2005)
18. Arsham, H., Kahn, A.: A simplex-type algorithm for general transportation problems: An alternative to stepping-stone. Journal of the Operational Research Society **40**(6), 581–590 (1989)
19. Balinski, M. L.: Signature methods for the assignment problem. Operations Research **33**(3), 527–536 (1985)
20. Bazaraa, M. S., Jarvis, J. J., Sherali, H. D.: Linear Programming and Network Flows (4th ed.). Hoboken, New Jersey: Wiley (2009)
21. Bilen, F., Csizmadia, Z., Illés, T.: Anstreicher - Terlaky type monotonic simplex algorithms for linear feasibility problems. Optimization Methods and Software **22**(4), 679–695 (2007)
22. Bland, R. G.: New finite pivoting rules for the simplex method. Mathematics of Operations Research **2**(2), 103–107 (1977)
23. Borgwardt, K.: The average number of pivot steps required by the simplex-method is polynomial. Mathematical Methods of Operations Research **26**(1), 157–177 (1982)
24. Borgwardt, K.: Some distribution-independent results about the asymptotic order of the average number of pivot steps of the simplex method. Mathematics of Operations Research **7**(3), 441–462 (1982)
25. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. Philadephia, PA: Society for Industrial and Applied Mathematics (2009)
26. Cardoso, D. M., Clímaco, J. C.: The generalized simplex method. Operations Research Letters **12**(5), 337–348 (1992)
27. Charnes, A., Cooper, W. W.: The stepping stone method of explaining linear programming calculations in transportation problems. Management Science **1**(1), 49–69 (1954)
28. Chen, H. D., Pardalos, P.M., Saunders, M. A.: The simplex algorithm with a new primal and dual pivot rule. Operations Research Letters **16**(3), 121–127 (1994)
29. Cochran, J. J.: You want them to remember? then make it memorable! means for enhancing operations research education. European Journal of Operational Research **219**(3), 659–670 (2012)
30. Csizmadia, Z.: New pivot based methods in linear optimization, and an application in the petroleum industry. Ph.D. thesis, Eötvös Loránd University of Sciences, Applied Mathematics (2007)
31. Csizmadia, Z., Bilen, F., Illés, T.: A new analysis for monotonic type simplex algorithms for feasibility problems. Alkalmazott Matematikai Lapok **24**(2), 163–185 (2007).
32. Csizmadia, Z., Illeś, T.: New criss-cross type algorithms for linear complementarity problems with sufficient matrices. Optimization Methods and Software **21**(2), 247–266 (2006)
33. Csizmadia, Z., Illés, T., Nagy, A.: The s-monotone index selection rules for pivot algorithms of linear programming. European Journal of Operational Research **221**(3), 491–500 (2012)
34. Csizmadia, Z., Illés, T., Nagy, A.: The s-monotone index selection rule for criss-cross algorithms of linear complementarity problems. Acta Univ. Sapientiae **5**(1), 103–139 (2013)
35. Curet, N. D.: A primal-dual simplex method for linear programs. Operations Research Letters **13**(4), 233–237 (1993)
36. Dantzig, G. B.: Programming in a linear structure. Tech. rep., Comptroller, US Air Force, Washington, D.C. (1948)
37. Dantzig, G. B.: Programming of interdependent activities: II mathematical model. Econometrica **17**(3/4), 200–211 (1949)
38. Dantzig, G.B.: Application of the simplex method to a transportation problem. In: T. C. Koopmans (Ed.) Activity analysis of production and allocation, pp. 359–373. New York: Wiley (1951)
39. den Hertog, D., Roos, C., Terlaky, T.: The linear complimentarity problem, sufficient matrices, and the criss-cross method. Linear Algebra and its Applications **187**, 1–14 (1993)

40. Dobson, G., Shumsky, R.: Web-based simulations for teaching queueing, little's law, and inventory management. INFORMS Transactions on Education **7**(1), 106–123 (2006)
41. Dosios, K., Notopoulos, P., Paparrizos, K.: Generalization of a signature method to transportation problems. Yugoslav Journal of Operations Research **6**(1), 55–71 (1996)
42. Dosios, K., Paparrizos, K.: A new exterior point algorithm for linear problems. Yugoslav Journal of Operations Research **4**(2), 137–148 (1994)
43. Dosios, K., Paparrizos, K., Samaras, N., Tsiplidis, K.: Simplex type algorithms generating two paths to the optimal solution. In: Proceedings of the 2nd Scandinavian Workshop on Linear Programming, pp. 35–39. Copenhagen, Denmark (1996)
44. Fukuda, K., Matsui, T.: On the finiteness of the criss-cross method. European Journal of Operational Research **52**(1), 119–124 (1991)
45. Fukuda, K., Namiki, M., Tamura, A.: EP theorems and linear complementarity problems. Discrete Applied Mathematics **84**(13), 107–119 (1998)
46. Fukuda, K., Terlaky, T.: Criss-cross methods: A fresh view on pivot algorithms. Mathematical Programming **79**(1-3), 369–395 (1997)
47. Fukuda, K., Terlaky, T.: On the existence of a short admissible pivot sequence for feasibility and linear optimization problems. Pure Mathematics and Applications **10**(4), 431–447 (1999)
48. Gay, D. M.: Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Newsletter **13**, 10–12 (1985)
49. Geranis, G., Paparrizos, K., Sifaleras, A.: A dual exterior point simplex type algorithm for the minimum cost network flow problem. Yugoslav Journal of Operations Research **19**(1), 157–170 (2009)
50. Geranis, G., Paparrizos, K., Sifaleras, A.: On a dual network exterior point simplex type algorithm and its computational behavior. RAIRO - Operations Research **46**(3), 211–234 (2012)
51. Glavelis, T., Samaras, N., Paparrizos, K.: An experimental investigation of a primal-dual exterior point simplex algorithm. In: Proceedings of the 1st International Symposium and 10th Balkan Conference on Operational Research (BALCOR 2011), vol. 2, pp. 240–247. Thessaloniki, Greece (2011)
52. Glover, F., Glover, R., Klingman, D.: Threshold assignment algorithm. In: G. Gallo, C. Sandi (Eds.) Netflow at Pisa, *Mathematical Programming Studies*, vol. 26, pp. 12–37. Berlin: Springer (1986)
53. Glover, F., Klingman, D., Napier, A.: Basic dual feasible solutions for a class of generalized networks. Operations research **20**(1), 126–136 (1972)
54. Goldfarb, D.: Efficient dual simplex algorithms for the assignment problem. Mathematical Programming **33**(2), 187–203 (1985)
55. Goldfarb, D., Hao, J.: A primal simplex algorithm that solves the maximum flow problem in at most nm pivots and $O(n^2m)$ time. Mathematical Programming **47**(3), 353–365 (1990)
56. Goldfarb, D., Hao, J., Kai, S. R.: Efficient shortest path simplex algorithms. Operations Research **38**(4), 624–628 (1990)
57. Gondzio, J.: Another simplex-type method for large scale linear programming. Control and Cybernetics **25**(4), 739–760 (1996)
58. Gregoriou, G., Kirytopoulos, K., Kiriklidis, C.: Project management educational software (ProMES). Computer Applications in Engineering Education **21**(1), 46–59 (2013)
59. Guerrero-Garcia, P., Santos-Palomo, A.: A deficient-basis dual counterpart of paparrizos, samaras and stephanides' primal-dual simplex-type algorithm. Optimization Methods Software **24**, 187–204 (2009)
60. Hultz, J., Klingman, D., Russell, R.: Advanced dual basic feasible solution for a class of capacitated generalized networks. Operations research **24**(2), 301–313 (1976)
61. Hung, M. S.: A polynomial simplex method for the assignment problem. Operations Research **31**(3), 595–600 (1983)
62. Illeś, T., , Mèszêros, K.: A new and constructive proof of two basic results of linear programming. Yugoslav Journal of Operations Research **11**(1), 15–30 (2001)
63. Illés, T., Molnár-Szipai, R.: On strongly polynomial variants of the MBU-simplex algorithm for a maximum flow problem with non-zero lower bounds. Optimization **63**(1), 39–47 (2013)
64. Illés, T., Nagy, A.: Computational aspects of simplex and MBU-simplex algorithms using different anti-cycling pivot rules. Optimization **63**(1), 49–66 (2014)
65. Illés, T., Nagy, A.: Finiteness of the quadratic primal simplex method when s-monotone index selection rules are applied. Tech. Rep. 2014-01, Department of Operations Research, Eötvös Loránd University of Sciences, Budapest, Hungary (2014)
66. Illés, T., Szirmai, A., Terlaky, T.: The finite criss-cross method for hyperbolic programming. European Journal of Operational Research **114**(1), 198–214 (1999)

67. Illeś, T., Terlaky, T.: Pivot versus interior point methods: Pros and cons. European Journal of Operational Research **140**(2), 170–190 (2002)
68. Jurík, T.: A nearest point approach algorithm for a class of linear programming problems. Journal of Applied Mathematics, Statistics and Informatics **4**(2), 129–138 (2008)
69. Karagiannis, P., Markelis, I., Paparrizos, K., Samaras, N., Sifaleras, A.: E-learning technologies: Employing matlab web server to facilitate the education of mathematical programming. International Journal of Mathematical Education in Science and Technology **37**(7), 765–782 (2006)
70. Karmarkar, N.: A new polynomial time algorithm for linear programming. Combinatorica **4**(4), 373–395 (1984)
71. Khachian, L. G.: A polynomial algorithm in linear programming. Soviet Mathematics Doklady **20**, 191–194 (1979)
72. Klafszky, E., Terlaky, T.: The role of pivoting in proving some fundamental theorems of linear algebra. Linear Algebra and its Applications **151**, 97–118 (1991)
73. Klafszky, E., Terlaky, T.: Some generalizations of the criss-cross method for quadratic programming. Optimization **24**(1-2), 127–139 (1992)
74. Klee, V., Minty, G.J.: How good is the simplex algorithm? In: O. Shisha (ed.) Inequalities, vol. III, pp. 159–175. New York: Academic Press (1972)
75. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., et al.: MIPLIB 2010. Mathematical Programming Computation **3**(2), 103–163 (2011)
76. Lazaridis, V., Paparrizos, K., Samaras, N., Sifaleras, A.: Visual linprog: A web-based educational software for linear programming. Computer Applications in Engineering Education **15**(1), 1–14 (2007)
77. Lee, J., Raffensperger, J.: Using AMPL for teaching the TSP. INFORMS Transactions on Education **7**(1), 37–69 (2006)
78. Li, W.: Practical criss-cross method for linear programming. In: L. Zhang, B.L. Lu, J. Kwok (Eds.) Advances in Neural Networks - ISNN 2010, *Lecture Notes in Computer Science*, vol. 6063, pp. 223–229. Berlin: Springer (2010)
79. Li, W., Guerrero-García, P., Santos-Palomo, A.: A basis-deficiency-allowing primal phase-I algorithm using the most-obtuse-angle column rule. Computers & Mathematics with Applications **51**, 903–914 (2006)
80. Malakooti, B., Al-Najjar, C.: The complex interior-boundary method for linear and nonlinear programming with linear constraints. Applied Mathematics and Computation **216**(7), 1903–1917 (2010)
81. Maros, I.: Computational techniques of the simplex method, vol. 61. Massachusetts: Springer (2003)
82. Mitra, G., Tamiz, M., Yadegar, J.: Experimental investigation of an interior search method within a simplex framework. Communications of the ACM **31**, 1474–1482 (1988)
83. Murty, K., Fathi, Y.: A feasible direction method for linear programming. Operations Research Letters **3**(3), 121–127 (1984)
84. Orlin, J. B.: A faster strongly polynomial minimum cost flow algorithm. Operations research **41**(2), 338–350 (1993)
85. Orlin, J. B.: A polynomial time primal network simplex algorithm for minimum cost flows. Mathematical Programming **78**(2), 109–129 (1997)
86. Orlin, J. B., Plotkin, S. A., Tardos, É.: Polynomial dual network simplex algorithms. Mathematical programming **60**(1-3), 255–276 (1993)
87. Pan, P. Q.: A largest-distance pivot rule for the simplex algorithm. European Journal of Operational Research **187**(2), 393–402 (2008)
88. Pan, P. Q.: A fast algorithm for linear programming. Journal of Computational Mathematics **28**(6), 837–847 (2010)
89. Papamanthou, C., Paparrizos, K., Samaras, N.: Computational experience with exterior point algorithms for the transportation problem. Applied Mathematics And Computation **158**(2), 459–475 (2004)
90. Papamanthou, C., Paparrizos, K., Samaras, N.: A parametric visualization software for the assignment problem. Yugoslav Journal of Operations Research **15**(1), 1–12 (2005)
91. Papamanthou, C., Paparrizos, K., Samaras, N., Sifaleras, A.: On the initialization methods of an exterior point algorithm for the assignment problem. International Journal of Computer Mathematics **87**(8), 1831–1846 (2010)
92. Papamanthou, C., Paparrizos, K., Samaras, N., Stergiou, K.: Worst case examples of an exterior point algorithm for the assignment problem. Discrete Optimization **5**(3), 605–614 (2008)

93. Paparrizos, K.: A non-dual signature method for the assignment problem and a generalization of the dual simplex-method for the transportation problem. RAIRO **22**(3), 269–289 (1988)
94. Paparrizos, K.: An infeasible (exterior point) simplex algorithm for assignment problems. Mathematical Programming **51**(1), 45–54 (1991)
95. Paparrizos, K.: A relaxation column signature method for assignment problems. European Journal of Operational Research **50**(2), 211–219 (1991)
96. Paparrizos, K.: An exterior point simplex algorithm for (general) linear programming problems. Annals of Operations Research **46-47**(2), 497–508 (1993)
97. Paparrizos, K.: A new primal and dual pivoting rule for the simplex algorithm. In: Proceedings of the SYMOPIS' 96, pp. 448–453. Zlatibor (1996)
98. Paparrizos, K.: A non improving simplex algorithm for transportation problems. RAIRO **30**(1), 1–15 (1996)
99. Paparrizos, K.: Pivoting algorithms generating two paths. In: International Symposium on Mathematical Programming (ISMP' 97), p. 207 (book of abstracts). Laussane: EPFL Switzerald (1997)
100. Paparrizos, K., Samaras, N., Sifaleras, A.: A new exterior simplex type algorithm for the minimum cost network flow problem. Computers & Operations Research **36**(4), 1176–1190 (2009)
101. Paparrizos, K., Samaras, N., Stephanides, G.: An efficient simplex type algorithm for sparse and dense linear programs. European Journal of Operational Research **148**(2), 323–334 (2003)
102. Paparrizos, K., Samaras, N., Stephanides, G.: A new efficient primal dual simplex algorithm. Computers & Operations Research **30**(9), 1383–1399 (2003)
103. Paparrizos, K., Samaras, N., Triantafyllidis, C.: A computational study of exterior point simplex algorithm variations. In: Proceedings of 20th Hellenic Operational Research Society, pp. 777–785. Spetses, Greece (2008)
104. Paparrizos, K., Samaras, N., Tsiplidis, K.: Some results on the finiteness of an exterior point simplex algorithm. In: Proceedings of the 3rd Balkan Conference on Operations Research, vol. 1, pp. 402–414. Thessaloniki, Greece (1995)
105. Paparrizos, K., Samaras, N., Tsiplidis, K.: Pivoting algorithms for linear programming generating two paths. In: C.A. Floudas, P.M. Pardalos (Eds.) Encyclopedia of Optimization, 2nd ed., pp. 2965–2969. New York: Springer (2009)
106. Phillips, N. V., Glover, F., Klingman, D.: Network Models in Optimization and Their Applications in Practice. New York: Wiley (1992)
107. Ploskas, N., Samaras, N.: Efficient GPU-based implementations of simplex type algorithms. Applied Mathematics and Computation **250**, 552–570 (2015).
108. Ploskas, N., Samaras, N., Sifaleras, A.: A parallel implementation of an exterior point algorithm for linear programming problems. In: Proceedings of the 9th Balkan Conference on Operational Research (BALCOR 2009). Constanta, Romania (2009)
109. Portugal, L., Bastos, F., Júdice, J., Paixão, J., Terlaky, T.: An investigation of interior-point algorithms for the linear transportation problem. SIAM Journal on Scientific Computing **17**(5), 1202–1223 (1996)
110. Reinelt, G.: TSPLIB - a traveling salesman problem library. INFORMS Journal on Computing **3**(4), 376–384 (1991)
111. Roos, C.: An exponential example for Terlaky's pivoting rule for the criss-cross simplex method. Mathematical Programming **46**(1), 79–84 (1990)
112. Russell, E. J.: Letters to the editor - extension of dantzig's algorithm to finding an initial near-optimal basis for the transportation problem. Operations Research **17**(1), 187–191 (1969)
113. Samaras, N.: Computational improvements and efficient implementation of two path pivoting algorithms. Ph.D. thesis, Dept. of Applied Informatics, University of Macedonia (2001)
114. Samaras, N., Sifaleras, A.: A comparative computational study of exterior point algorithms for the assignment problem. In: Proceedings of 19th National Conference of Hellenic Operational Research Society (HELORS). Arta, Greece (2007)
115. Samaras, N., Sifaleras, A., Triantafyllidis, C.: A primal - dual exterior point algorithm for linear programming problems. Yugoslav Journal of Operations Research **19**(1), 123–132 (2009)
116. Sherali, H. D., Özdaryal, B., Adams, W. P., Attia, N.: On using exterior penalty approaches for solving linear programming problems. Computers and Operations Research **28**, 1049–1074 (2001)
117. Sifaleras, A.: Minimum cost network flows: Problems, algorithms, and software. Yugoslav Journal of Operations Research **23**(1), 3–17 (2013)
118. Terlaky, T.: A convergent criss-cross method. Optimization **16**(5), 683–690 (1985)
119. Terlaky, T., Zhang, S.: Pivot rules for linear programming: a survey on recent theoretical developments. Annals of Operations Research **46**(1), 203–233 (1993)

120. Triantafyllidis, C., Samaras, N.: Three nearly scaling invariant versions of an exterior point algorithm for linear programming. Optimization: A Journal of Mathematical Programming and Operations Research (2014). To appear
121. Väliaho, H.: A new proof for the criss-cross method for quadratic programming. Optimization **25**(4), 391–400 (1992)
122. Vygen, J.: On dual minimum cost flow algorithms. Mathematical Methods of Operations Research **56**(1), 101–126 (2002)
123. Wolfe, P.: The simplex method for quadratic programming. Econometrica: Journal of the Econometric Society **27**, 382–398 (1959)
124. Yeh, W. C., Corley, H.: A simple direct cosine simplex algorithm. Applied Mathematics and Computation **214**(1), 178–186 (2009)
125. Zhang, S.: New variants of finite criss-cross pivot algorithms for linear programming. European Journal of Operational Research **116**(3), 607–614 (1999)
126. Zionts, S.: The criss-cross method for solving linear programming problems. Management Science **15**(7), 426–445 (1969)
127. Zionts, S.: Some empirical tests of the criss-cross method. Management Science **19**(4-Part-1), 406–410 (1972)