To appear in *Optimization*
Vol. 00, No. 00, Month 20XX, 1–19

Research Paper

Improving a Primal–Dual Simplex-type Algorithm using Interior Point Methods

Th. Glavelis[a], N. Ploskas[b], and N. Samaras[a]*
[a] *Department of Applied Informatics, School of Information Sciences, University of Macedonia, 156 Egnatia Str., 54006 Thessaloniki, Greece*
[b] *Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Interior point methods and simplex-type algorithms are the most widely-used algorithms for solving linear programming problems. The simplex algorithm has many important applications. Hence, even small improvements in simplex-type algorithms could result in noticeable practical impact. This paper presents a hybrid algorithm that combines the strengths of interior point methods and exterior point simplex algorithms. It applies an interior point method for a few iterations leading to significant improvement of the objective function value. At this point, the proposed algorithm uses an exterior point simplex algorithm to find an optimal solution. A crucial point is the selection of the interior point that will be used by the exterior point simplex algorithm to calculate a direction to the feasible region. The goal of the proposed implementation is twofold: (i) improve the performance of the exterior point algorithm, and (ii) find an optimal basic solution starting from an interior point (purification process). The latter goal is very important since an optimal basic solution can be used to solve closely related linear programming problems (warm-start) and linear programming relaxations of integer programming problems. Computational results on a set of benchmark problems (Netlib, Kennington, Mészáros) are presented to demonstrate the efficiency of the proposed hybrid algorithm. The results show that the proposed algorithm is 1.53× faster than the exterior point simplex algorithm.

Keywords: Linear programming; Simplex algorithm; Interior point methods; Exterior point algorithm; Computational study

AMS Subject Classification: 90C05; 90C51; 65K05

1. Introduction

Linear Programming (LP) is a significant research area in the field of operations research. Linear programs can be found in almost every type of scientific and engineering applications. The first approach for solving linear programming problems (LPs) came from George B. Dantzig who set the fundamental principles. Dantzig proposed the simplex algorithm [1] that starts from a basic feasible solution and moves from one basic feasible solution to an adjacent one until an optimum solution is found. The vast literature of operations research contains an extensive family of simplex-type algorithms. Although the simplex algorithm is widely-used until today, there exist many real-life applications that its performance degrades due to a phenomenon called degeneracy. Simplex-type algorithms may stall at a degenerate vertex for many iterations before moving to another vertex. Degeneracy

*Corresponding author. Email: samaras@uom.gr and redundancy are very common in real-life applications. Consequently, many different approaches were presented in order to overcome degeneracy [2]. Over the last decades, researchers focused on: (i) the reduction of the number of iterations that the simplex algorithm performs [3] [4] [5] (for a review, see [6]), (ii) the reduction of the computational work involved in each iteration [7] [8] [9] [10] (for a review, see [11]), and (iii) new variants of the simplex algorithm [12] [13] [14] [15].

The first polynomial time algorithm for LP is the ellipsoid algorithm which was developed by Khachiyan [16]. The ellipsoid algorithm is impractical for LP. In 1984, a totally new method arose and changed everything in LP [17]; Interior Point Methods (IPMs) revealed that the simplex algorithm was not the only way for solving an optimization problem. Since then, many IPMs have been proposed (for a literature review, see [18] [19]). IPMs have some advantages over the simplex algorithm. Probably, the most important advantage is that the number of iterations is not related to the number of vertices. Nowadays, it is broadly accepted that an infeasible primal-dual IPM is the most efficient algorithm of this category.

Since the development of the simplex method, various papers presented variants of the simplex method that relax feasibility requirements. Al-Sultan and Murty [20] proposed an exterior penalty algorithm for the nearest point problem. Sherali et al. [21] investigated 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. A simplex-type algorithm generating solutions that are not feasible is called an Exterior Point Simplex Algorithm (EPSA) [22]. Dantzig's [1] parametric self-dual algorithm, Kuhn's [23] Hungarian method for the assignment problem, Iri's [24] successive shortest part method for minimum cost flow problems, Zionts' [25] and Terlaky's [26] criss-cross methods are some examples of exterior point methods. However, they are not very efficient in practice for solving LPs. EPSA was proposed by Paparrizos initially for the assignment problem [27] and then for the solution of LPs [22]. Moreover, researchers introduced the primal–dual versions of the algorithm that enhanced its computational behavior [28] [29] [30] (for a literature review of recent advances on EPSA, see [31]).

During the last decades, researchers proposed more efficient implementations of LP algorithms. Other efforts focused on the parallelization of LP algorithms, on CPUs [32] [33] [34] [35] [36] [37] and on GPUs [38] [39] [40] [41] [42] [43], and the combination of different LP algorithms [44] [45] [46] [47] [48]. The proposed hybrid algorithm belongs to the latter category. The idea to combine two types of LP algorithms is not new. Kortanek & Zhu [47] proposed a pivoting procedure from an interior point to a boundary point without worsing the objective value. This procedure can be performed in finite steps but may not be polynomial. Bixby et al. [45] and Bixby & Saltzman [46] proposed a combination of an IPM with the simplex algorithm. The hybrid procedure starts running an IPM first and later switches to the simplex algorithm. Andersen & Ye [44] proposed a combination of an IPM with a pivoting algorithm using a totally different idea from [45] [46]; they construct an artificial linear programming problem, which approximates the original problem, in any iteration of an IPM. Finally, they apply Megiddo's procedure [49] to compute an optimal basis of the approximate problem in n pivot steps. Al-Najjar and Malakotti [50] proposed 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 simplex algorithm can be applied in order to reach an optimal solution. The computational results of the hybrid-LP method are very promising. Pan [51] proposed a pivoting algorithm using the affine-scaling technique. This method produces a sequence of interior points as well as a sequence of vertices, until reaching an optimal vertex. Triantafyllidis [48] proposed a non-monotonic variant of the exterior point algorithmic family by combining EPSA with IPMs.

This paper builds on the work done by Bixby et al. [45] by introducing a hybrid algorithm that combines IPMs and EPSA. The idea of combining these two different types of methods stemmed from the observation that IPMs are able to spot very fast feasible solutions with good objective values, but they need a relatively long time to converge to an optimal solution. In order to take full advantage of EPSA, we use a variation of a Primal-Dual Exterior Point Simplex Algorithm (PDEPSA). Primal-dual algorithms can deal more effectively with the problems of stalling and cycling and as a result improve the performance of EPSA. This variation, which is presented in this paper, is called Primal-Dual Interior Point Simplex Algorithm (PDIPSA) since the algorithm computes a direction to the feasible region according to the interior point that was found by an IPM. The IPM, which we use in our hybrid algorithm, is Mehrotra's Predictor-Corrector method [52], an infeasible primal-dual IPM.

The main advantage of this hybrid algorithm is that it exploits the strengths of both IPM and PDIPSA. In the first iterations, IPM moves from a positive point to a positive point trying to achieve feasibility and optimality, simultaneously. At this point, the proposed hybrid algorithm uses PDIPSA to find an optimal solution in less expensive iterations. The goal of the proposed implementation is twofold: (i) improve the performance of EPSA, and (ii) find an optimal basic solution starting from an interior point (purification process). The latter goal is preferable for a couple of reasons [44]. First of all, a basic solution has generally fewer nonzero elements than a solution in the interior of the optimal face, which is desirable when LP relaxations of integer programming problems are solved. Secondly, an optimal basic solution can be used to warm-start simplex-type algorithms to solve closely related LPs.

The paper is organized as follows. Section 2 includes the description of the general framework of the proposed hybrid algorithm. In Section 3, we give the proof of correctness. In order to gain an insight into the practical behavior of the proposed hybrid algorithm, we have performed a computational study on a set of benchmark problems (Netlib, Kennington, Mészáros). These results are presented in Section 4. Finally, the conclusions and possible enhancements of the proposed hybrid algorithm are outlined in Section 5.

2. Description of the hybrid algorithm

Initially, this Section presents the two algorithms that we combine in our implementation. Subsection 2.1 presents PDIPSA, a primal-dual interior point simplex algorithm, while subsection 2.2 gives a brief overview of Mehrotra's interior point method. Finally, the hybrid algorithm, which combines the previously mentioned algorithms, is presented in subsection 2.3.

2.1. *PDIPSA*

In this section, we describe PDIPSA in depth. Consider the following linear programming problem (LP.1) in the standard form:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \qquad (LP.1)$$

where $A \in \mathbb{R}^{m \times n}$, $(c, x) \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and T denotes transposition. We assume that A has full rank, $rank(A) = m, m < n$. Consequently, the linear system $Ax = b$ is consistent.

The dual problem associated with the (LP.1) is presented in (DP.1):

$$\begin{array}{ll} \max & b^T w \\ \text{s.t.} & A^T w + s = c \\ & s \geq 0 \end{array} \qquad (DP.1)$$

where $w \in \mathbb{R}^m$ are the Langrange multipliers and $s \in \mathbb{R}^n$ are the dual slack variables.

Using a basic partition (B, N), where B is the set of basic indices and N is the complementary set of nonbasic indices, the linear programming problem in (LP.1) can be written as shown in (LP.2).

$$\begin{array}{ll} \min & c_B^T x_B + c_N^T x_N \\ \text{s.t.} & A_B x_B + A_N x_N = b \\ & x_B, x_N \geq 0 \end{array} \qquad (LP.2)$$

In (LP.2), A_B is an $m \times m$ nonsingular sub-matrix of A, called basic matrix or basis. The columns of A that belong to subset B are called basic and those that belong to N are called nonbasic. The basic solution corresponding to the basis B is $x_B = (A_B)^{-1} b$ and $x_N = 0$. This solution is feasible iff $x_B \geq 0$. Otherwise, it is infeasible. The solution of (DP.1) is computed by the relation $s_N^T = (c_N)^T - w^T A_N$, where $w^T = (c_B)^T (A_B)^{-1}$. The basis B is dual feasible iff $s \geq 0$. In order to initialize the algorithm, a basic feasible solution must be available. It is well known that a basis for which the primal solution (x_B, x_N) is feasible to (LP.1) and the dual solution (w, s) is feasible to (DP.1), is also an optimal basis.

PDIPSA is initialized with a dual feasible basis. The procedure of a dual feasible basis construction is based on a big–M problem [30]. The solutions of (LP.1) and (DP.1) corresponding to the basic partition (B, N) are denoted by $x = (x_B, x_N)$ and $(w, s) = (w, s_N)$, respectively. We assume that an interior point $y > 0$ to problem (LP.1) is available. We explain how we choose the initial point y later in this Section. The main idea of our algorithm is that the point y must be an interior point. The algorithm generates a sequence of dual feasible bases $B^{(t)}$, $t = 1, 2, \cdots$. The primal solution $x^{(t)}$ and the dual solution $(w^{(t)}, s^{(t)})$ correspond to the basis $B^{(t)}$. Recall that $x^{(t)}$ is not (in general) feasible to (LP.1).

Initially a direction $d_B = y_B - x_B$, where y_B is an initial interior point of (LP.1), is computed. As x_B corresponds to a feasible basis of (DP.1) and y_B is feasible to (LP.1), the direction d_B is an ascent direction for the objective function $c^T x$.

At the next step, the leaving variable x_k is calculated from the following maximum ratio test:

$$a = \frac{x_{B[r]}}{-d_{B[r]}} = max\left\{\frac{x_i}{-d_i} : i \in B, d_i > 0 \wedge x_i < 0\right\} \tag{1}$$

where r is the position in the basic list B where the maximum is found. In case of ties, the rightmost index is selected.

The ray $R = \{x + a'd : a' \geq 0\}$ enters the feasible region of (LP.1) through the boundary point $x + ad$. Then, the algorithm moves to the point y_B, which is inside (interior) the feasible region. This computation is achieved by the relation $y_B = x_B + a'd_B$, where $a' = \frac{a+1}{2}$. PDIPSA can be described formally as shown in Table 1.

Table 1. Primal-Dual Interior Point Simplex Algorithm (PDIPSA)

```
Step 0. (Initialization).
A) Start with a dual feasible basic partition $(B, N)$ and an interior point
$y > 0$ of (LP.1).
Set:
        $P = N, Q = \varnothing$
and compute
        $x_B = (A_B)^{-1} b$, $w^T = (c_B)^T (A_B)^{-1}$, $s_N^T = (c_N)^T - w^T A_N$
B) Compute the direction $d_B$ from the relation: $d_B = y_B - x_B$
Step 1. (Test of optimality and choice of the leaving variable).
If $x \geq 0$ then STOP. (LP.1) is optimal.
else
  Choose the leaving variable $x_k$ from the relation:
        $a = \frac{x_{B[r]}}{-d_{B[r]}} = max\left\{\frac{x_i}{-d_i} : i \in B, d_i > 0 \wedge x_i < 0\right\}$
Step 2. (Computation of the next interior point).
Set:
        $a' = \frac{a+1}{2}$
Compute the interior point from the relation: $y_B = x_B + a'd_B$
Step 3. (Choice of the entering variable).
Set: $H_{rN} = (A_B)^{-1}_{r.} A_N$
Choose the entering variable $x_l$ from the relation:
        $\frac{-s_l}{H_{rN}} = \min\left\{\frac{-s_j}{H_{rj}} : H_{rj} \wedge j \in N\right\}$
Compute the pivoting column: $h_l = (A_B)^{-1} A_{.l}$
if $l \in P$ then
  $P \leftarrow P \setminus \{l\}$
else
  $Q \leftarrow Q \setminus \{l\}$
Step 4. (Pivoting).
Set:
        $B[r] = l$ and $Q \leftarrow Q \cup \{k\}$
Using the new partition $(B, N)$ where $N = (P, Q)$, compute/update the
new basis inverse $A_B^{-1}$ and the variables $x_B$, $w$, and $s_N$.
Go to step $B$.
```

A revised form of the algorithm is implemented in the current paper. Any known technique for updating the basic inverse matrix $(A_B)^{-1}$ and the vectors x_B, w, and s_N can be combined efficiently with the algorithm. Here, our effort is focused on how the direction d can be computed more efficiently. Our results lead to a different calculation of d_B; the component d_B is updated in a way very similar to that of x_B. Moreover, our results lead to the construction of a big–M problem (for solving general LPs).

As it is mentioned before, PDIPSA belongs to the family of EPSA. EPSA construct two paths to an optimal solution. The first path is a sequence of basic but not feasible solutions and this path is called exterior path. The other path is a feasible path, its points move on the boundary of the feasible region. Despite their promising computational performance, EPSA have two significant computational disadvantages. The first weakness stems from the difficulty of constructing "good" moving directions, which could lead the algorithm close to an optimal solution. The creation of a direction with these features is a difficult process. The computational performance of EPSA is strongly connected to this moving direction. The second disadvantage is the fact that there is no known method, which can reveal the path that leads to the interior of the feasible region; something that would make easier the search of a computational good direction. These disadvantages can be avoided if the exterior path is replaced with a dual feasible simplex path.

A good implementation of this type of EPSA is described by Paparrizos et al. [53]. The main idea of the Revised Primal Dual Simplex Algorithm (RPDSA) is based on the process of moving from any interior point to an optimal basic solution. Although this algorithm is better from the exterior point simplex algorithm and it deals very well with the two disadvantages, which were described above, it also has some weaknesses. RPDSA begins with an interior point and at each iteration a boundary point is used to compute the leaving variable. It has been observed that the problem of stalling and cycling can arise very often at this stage. This weakness can be overcome if the boundary point is replaced by an interior point. The transfer into the interior of the feasible region makes the algorithm to avoid the problem of stalling and cycling.

In this paper, we propose an alternative approach of RPDSA. This approach is much more efficient since it can overcome these two significant drawbacks, of stalling and cycling. Furthermore, this algorithm is a primal–dual algorithm, meaning that it simultaneously solves the primal and the dual problem. In contrast to RPDSA, we use an interior point at each iteration to compute the leaving variable and this is the key factor that enhances the algorithm and thus, we can avoid the problem of stalling and cycling.

A geometrical representation is necessary to clarify the reasons that our algorithm can deal quite satisfactory with these two drawbacks. In Figure 1, we present an LP problem where the problem of staling arises. We assume that our algorithm is at vertex A at the current iteration. According to RPDSA, the direction d' computes the boundary point y' from which it enters the feasible region. This point is used to choose the leaving variable (constraint); at this point, there are three possible options, constraints (1), (2) or (3). In other words, there exist bonds in the specific LP problem. If the next leaving variable is constraint (1), then our algorithm will move to vertex B. In this case, the new direction enters again the feasible region from point y'; keeping the point y' boundary, the algorithm will move to vertices C and D consecutively until it reaches vertex E, which is an optimal solution. Consequently, it will compute the optimal value after four iterations; this phenomenon of pivoting between degenerated vertices of the feasible region is called stalling. The problem of stalling can be overcome if the boundary point y' is replaced from an interior point y''. Now, using this interior point the direction

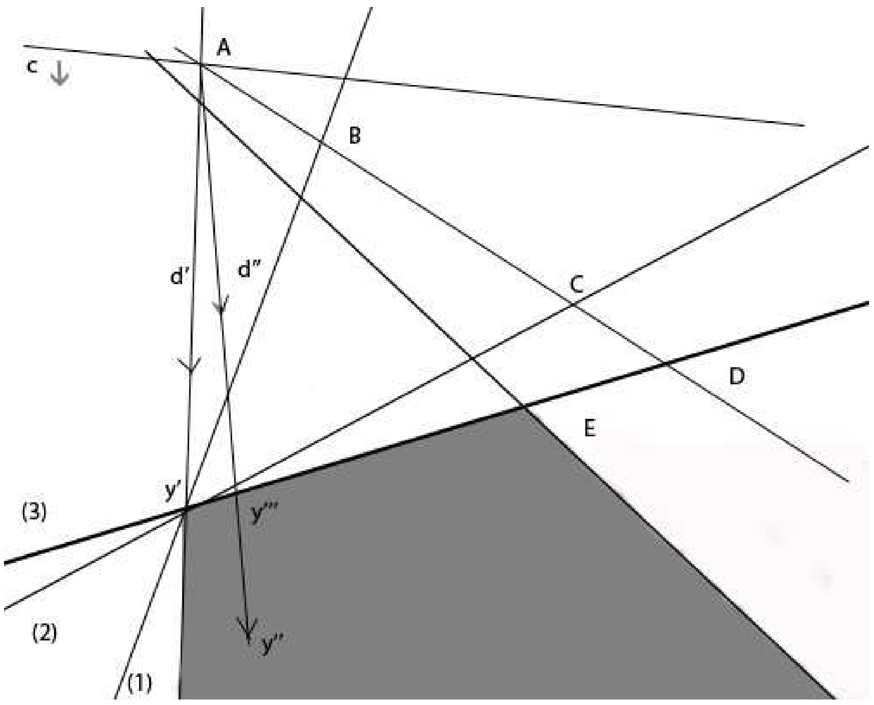

Figure 1. Stalling

d'' enters the feasible region from the boundary point point y'''. Only constraint (3) comes through this point and PDIPSA will move to vertex D without visiting vertices B and C. Consequently, an optimal solution will be calculated in two iterations in contrast to RPDSA that needs four iterations.

Apart from the problem of stalling, RPDSA has another significant drawback, it is vulnerable to cycling. In order to clarify this specific situation, we assume in the above LP problem that the objective function is parallel to line $\varepsilon 1$ (see Figure 2). According to RPDSA, the algorithm's pivot from vertex A to vertex D cannot lead to any change of the objective function value. Furthermore, in such cases, the algorithm may continue cycling from one vertex to another. This weakness can be also overcome if we replace the boundary point with an interior point as it was described previously.

2.2. *Mehrotra's predictor-corrector method*

Since Karmarkar's algorithm [17], many improvements have been made both in theory and in practice of IPMs. An infeasible IPM moves from a positive point to a positive point trying to achieve feasibility and optimality, simultaneously; this is the big difference with the simplex algorithm, which follows a sequence of adjacent boundary points to an optimal solution. It has been observed that IPMs can deal much better than the simplex algorithm in large-scale sparse LPs [18]; these problems are very common in transportation and scheduling applications that have network models at their core. IPMs are also of interest from a theoretical point of view, because they have polynomial complexity. There are three main categories of IPMs: (i) affine-scaling methods, (ii) potential reduction methods, and (iii) central trajectory methods. The affine-scaling algorithm is an attractive choice due to its simplicity and its relative good performance in practice. However, its performance is sensitive to the starting point. Potential reduction methods do

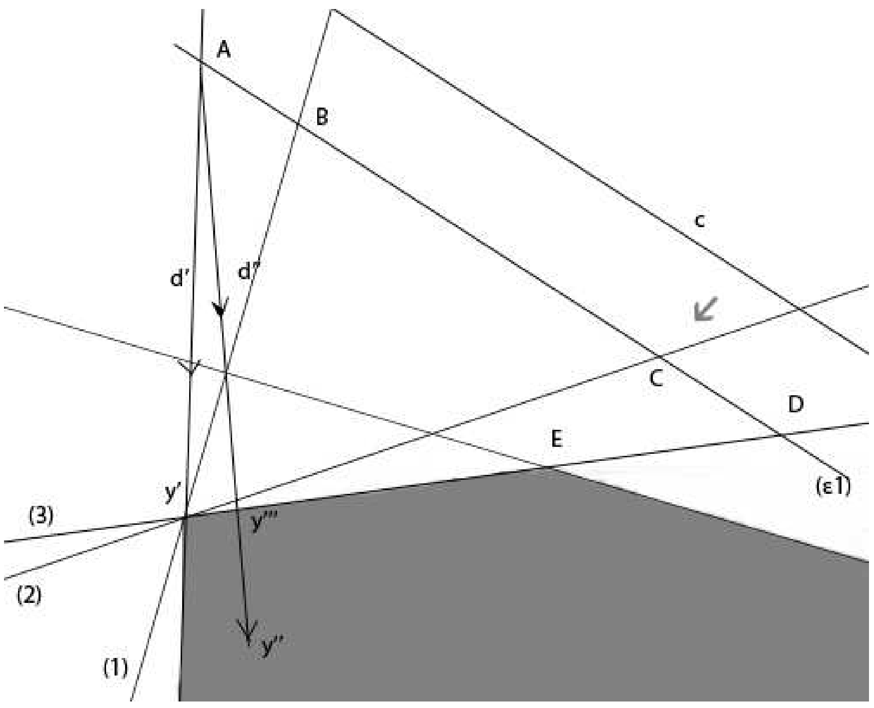

Figure 2. Cycling

not have the simplicity of affine-scaling methods, but they are more attractive than affine-scaling methods. IPMs based on the central trajectory are the most useful in theory and the most used in practice.

The main advantages of IPMs in comparison to the simplex algorithm are: (i) the number of iterations is not related with the number of vertices, and (ii) IPMs are not influenced by degeneracies. On the other hand, IPMs have some significant weaknesses: (i) it has been observed that IPMs are not very effective to detect infeasibility or unboundedness in some cases, and (ii) numerical issues and bad starting points may lead IPMs to slow convergence in the later iterations.

The primal–dual path following algorithm is an example of an IPM that operates simultaneously on the primal and dual LPs. Moreover, the primal–dual algorithms that incorporate predictor and corrector steps are the most efficient IPMs. This is the reason that we chose to implement Mehrotra's Predictor-Corrector method [52] in order to calculate a "good" interior point for PDIPSA. Although some primal-dual IPMs need a strictly feasible interior point as a starting point, which is difficult to calculate, Mehrotra's Predictor-Corrector is an infeasible primal-dual IPM and it just requires that $(x, s) > 0$ for the starting point. Mehrotra also proposed an efficient heuristic to obtain a starting point [52].

2.3. *Combining IPM–PDIPSA (Hybrid IPM–PDIPSA)*

A hybrid algorithm is proposed in this paper. This algorithm combines an IPM with an EPSA and more specifically Mehrotra's Predictor-Corrector method with PDIPSA; the most efficient IPM and EPSA, respectively. The main goal of this combination is to adopt the strengths of each algorithm and to eliminate their disadvantages. According to this thought, the hybrid algorithm executes Mehrotra's Predictor-Corrector method for a few iterations in order to start PDIPSA from a "good" interior point. Then, PDIPSA completes the calculations and solves the

LP problem.

PDIPSA demands a starting interior point; this point is computed by Mehrotra's Predictor-Corrector method. Moreover, the interior point is necessary for the calculation of the direction d, which reveals the leaving variable. Another significant issue is that a "good" initial interior point can lead to significant less iterations of PDIPSA. Consequently, if the initial point is closer to optimal vertex, then an optimal solution will be sooner spotted by PDIPSA. This is the main reason for using an IPM at the first stage. IPM is able to move to an interior point close to the optimal vertex at the first iterations. In this step, PDIPSA receives the interior point and continues finding the solution. Taking under consideration IPM's relative large computational cost per iteration and late convergence at the last iterations (due to numerical issues or a bad starting point), our hybrid algorithm takes full advantage of IPMs at the same time of giving a "good" interior point to PDIPSA.

The hybrid algorithm is described formally as follows:

Table 2. Hybrid approach combining Mehrotra's predictor-corrector method and PDIPSA

```
**Step 1.** *(IPM).*
Perform a few iterations with Mehrotra's Predictor-Corrector method in order to compute a "good" interior point $y$.
**Step 2.** *(PDIPSA).*
A) Initialize PDIPSA with a dual feasible basic partition $(B, N)$ and the interior point $y$ taken from Step 1.
B) Iteratively, PDIPSA continues until it computes an optimal solution as it was described in Section 2.1.
```

3. Proof of correctness

The geometrical representation, which was presented in Section 2.1, and the similarity of the proposed hybrid algorithm to the dual simplex method, reveal immediately its correctness. When the algorithm terminates, the basic solution is both primal and dual feasible. To complete the proof of correctness of the algorithm, it suffices to show that every basic partition, which is constructed by PDIPSA, is dual feasible and the computation of the maximum ratio test (Equation (1)) is well defined.

Theorem 1: If the initial basic partition of PDIPSA is dual feasible, then every consecutive partition is dual feasible.

Proof: The proof is by induction on the number of iterations. Denote by t the number of iterations. It is obvious from Step A of PDIPSA that for $t = 1$ the relations $S_j^{(1)} \geq 0\,, j \in N^{(1)}$, and $S_j^{(1)} = 0, j \in B^{(1)}$, hold. Suppose now that the relation $S_j^{(t)} \geq 0$, $j \in N^{(t)}$, holds. Let $(B^{(t+1)}, N^{(t+1)})$ be the new basic partition and $S_j^{(t+1)}$, $j \in N^{(t+1)}$, the corresponding dual slack variables. The dual slack variables can be computed by the relation

$$S_j^{(t+1)} = S_j^{(t)} - \frac{S_l^{(t)}}{H_{rl}} H_{rj}, j \in N^{(t+1)}, \tag{2}$$

where H_{rj}, $j \in N^{(t+1)}$, is the pivot row. From the choice of the entering variable

x_l

$$\frac{-S_l^{(t)}}{H_{rl}} = min\{\frac{-S_j^{(t)}}{H_{rj}} : H_{rj} < 0 \wedge j \in N^{(t)}\} \tag{3}$$

we conclude that $\frac{-S_l^{(t)}}{H_{rl}} \geq 0$. If $H_{rj} \leq 0, j \in N^{(t+1)}$, then $S_j^{(t+1)} \geq 0$ holds as the summation of two vectors with positive entries. If $H_{rj} > 0, j \in N^{(t+1)}$, then relation (2) is equivalent to

$$\frac{S_j^{(t)}}{H_{rj}} \geq \frac{S_l^{(t)}}{H_{rl}}, \tag{4}$$

which is true according to relation (3) and consequently $S_j^{(t+1)} \geq 0$. Hence, if the initial basic partition is dual feasible, then PDIPSA constructs dual feasible partitions at every iteration.

Lemma 1: At every iteration of PDIPSA, the maximum ratio test yields $a \in (0, 1)$.

Proof: The condition $x_{B[i]} < 0$ combined with the relation $d_B = y_B - x_B$ and the facts that x_B is dual feasible and y_B is primal feasible, implies the relation $x_i < 0 \Rightarrow d_i > 0, i \in B$.

From the maximum ratio test we have

$$a = max\{\frac{-x_i}{d_i} : i \in B, d_i > 0 \wedge x_i < 0\} =$$

$$max\{\frac{|x_i|}{y_i - x_i} : i \in B, d_i > 0 \wedge x_i < 0\} =$$

$$max\{\frac{|x_i|}{|y_i| + |x_i|} : i \in B, d_i > 0 \wedge x_i < 0\}$$

It is obvious from the above relation that $0 < a < 1$.

4. Computational results

In this Section, we present the results from a computational study that we conducted to demonstrate the efficiency of the proposed hybrid algorithm. The computational comparison has been performed on a quad-processor Intel Core i7 3.4 GHz with 32 Gbyte of main memory and 8 cores, a clock of 3.7 GHz, an L1 code cache of 32 KB per core, an L1 data cache of 32 KB per core, an L2 cache of 256 KB per core, an L3 cache of 8 MB and a memory bandwidth of 21 GB/s, running under Microsoft Windows 8 64-bit. All algorithms have been implemented using MATLAB Professional R2015b. Some linear algebra built-in functions were also used to code the algorithms (e.g., inverse of an array, multiplication of two arrays, multiplication of array and vector, and the mldivide operator for solving systems of linear equations). Execution times have been measured in seconds using *tic* and *toc* MATLAB's built-in functions. For each instance, we averaged times over 10 runs. All runs were executed as a batch job.

Totally, 83 LPs were considered from the Netlib set (Optimal and Kennington

LPs) [54] [55] and the problematic, misc, and stochlp sections of Mészáros collection [56]. The Netlib library is a well known suite containing many real world LPs. Ordóñez and Freund [57] have shown that 71% of the Netlib LPs are ill-conditioned. Hence, numerical difficulties may occur. We implemented an MPS reader to read MPS files and convert data into MATLAB mat files. All runs terminated with correct optimal objective values. Table 3 presents some useful information about the test bed, which was used in the computational study. The first column includes the name of the problem, the second the number of constraints, the third the number of variables, the fourth the nonzero elements of matrix A, and the fifth the optimal objective value.

Table 3.: Statistics of the Netlib (optimal and Kennington LPs) and Mészáros LPs

Name	**Constraints**	**Variables**	**Nonzeros A**	**Optimal objective value**
aa4	426	7,195	52,121	2.59E+04
aa5	801	8,308	65,953	5.37E+04
aa6	646	7,292	51,728	2.70E+04
adlittle	56	97	383	2.25E+05
afiro	27	32	83	-4.65E+02
agg	488	163	2,410	-3.60E+07
agg2	516	302	4,284	-2.02E+07
agg3	516	302	4,300	1.03E+07
aircraft	3,754	7,517	20,267	1.57E+03
beaconfd	173	262	3,375	3.36E+04
blend	74	83	491	-3.08E+01
bnl2	2,324	3,489	13,999	1.81E+03
car4	16,384	33,052	63,724	3.55E+01
cari	400	1,200	152,800	5.82E+02
cr42	905	1,513	6,614	2.80E+01
cre-a	3,516	4,067	14,987	2.36E+07
d6cube	415	6,184	37,704	3.15E+02
fffff800	524	854	6,227	5.56E+05
fit1d	24	1,026	13,404	-9.15E+03
forplan	161	421	4,563	-6.64E+02
fxm2-6	3,900	5,602	32,239	1.84E+04
fxm3_6	6,200	9,492	54,589	1.86E+04
gen	769	2,560	63,085	0.00E+00
gen1	769	2,560	63,085	0.00E+00
gfrd-pnc	616	1,092	2,377	6.90E+06
iiasa	669	2,970	6,648	2.63E+08
israel	174	142	2,269	-8.97E+05
jendrec1	2,109	4,228	89,608	7.03E+03
lotfi	153	308	1,078	-2.53E+01
maros-r7	3,136	9,408	144,848	1.50E+06
nsic1	451	463	2,853	-9.17E+06
nsic2	465	463	3,015	-8.20E+06
nsir1	4,407	5,717	138,955	-2.89E+07
nsir2	4,453	5,717	150,599	-2.72E+07
osa-07	1,118	23,949	143,694	5.36E+05
osa-14	2,337	52,460	314,760	1.11E+06

osa-30	4,350	100,024	600,138	2.14E+06
p05	5,090	9,500	58,955	3.15E+02
p010	10,090	19,000	117,910	1.12E+06
pgp2	4,034	9,220	18,440	4.47E+02
primagaz	1,554	10,836	21,665	1.07E+09
r05	5,190	9,500	103,955	5.58E+05
rail507	507	63,009	409,349	1.72E+02
rail516	516	47,311	314,896	1.82E+02
rail582	582	55,515	401,708	2.10E+02
rat1	3,136	9,408	88,267	2.00E+06
rat5	3,136	9,408	137,413	3.08E+06
rat7a	3,136	9,408	268,908	2.07E+06
recipe	91	180	663	-2.67E+02
rosen2	1,032	2,048	46,504	-5.44E+04
rosen7	264	512	7,770	-2.03E+04
rosen8	520	1,024	15,538	-4.21E+04
rosen10	2,056	4,096	62,136	-1.74E+05
sc105	105	103	280	-5.22E+01
sc205	205	203	551	-5.22E+01
sc205-2r-400	8,813	8,814	24,030	-1.01E+01
sc205-2r-800	17,613	17,614	48,030	-1.01E+01
sc205-2r-1600	35,213	35,214	96,030	0.00E+00
sc50a	50	48	130	-6.46E+01
sc50b	50	48	118	-7.00E+01
scagr25	471	500	1,554	-1.48E+07
scagr7	129	140	420	-2.33E+06
scagr7-2b-64	9,743	10,260	32,298	-8.33E+05
scagr7-2r-216	8,223	8,660	27,042	-8.34E+05
scagr7-2r-432	16,431	17,300	54,042	-8.34E+05
scfxm1	330	457	2,589	1.84E+04
scfxm1-2b-64	19,036	28,914	106,919	2.88E+03
scfxm3	990	1,371	7,777	5.49E+04
scrs8	490	1,169	3,182	9.04E+02
sctap1	300	480	1,692	1.41E+03
sctap2	1,090	1,880	6,714	1.72E+03
sctap3	1,480	2,480	8,874	1.42E+03
share1b	117	225	1,151	-7.66E+04
share2b	96	79	694	-4.16E+02
ship12l	1,151	5,427	16,170	1.47E+06
ship12s	1,151	2,763	8,178	1.49E+06
slptsk	2,861	3,347	72,465	2.99E+01
standata	359	1,075	3,031	1.26E+03
stocfor1	117	111	447	-4.11E+04
stocfor2	2,157	2,031	8,343	-3.90E+04
stocfor3	16,675	15,695	64,875	-4.00E+04
testbig	17,613	31,223	61,639	-6.04E+01
zed	116	43	567	-1.51E+04

Since our primary aim is to improve the computational performance of PDIPSA, we compare the proposed hybrid algorithm, HYBRID, with PDIPSA. As described in Section 2, HYBRID uses Mehrotra's Predictor-Corrector method to calculate an interior point. In addition, we have implemented the primal Revised Simplex Algorithm (RSA) and we use it in the computational study as a reference point for the comparison. All algorithms use the same preprocessing, scaling, and basis update methods. Their major difference is how they select the entering and leaving variable. In the simplex implementation, we use Dantzig's pivoting rule; however, if degeneracy is detected during the algorithm's execution, then simplex will automatically switch to the steepest-edge pivoting rule and the problem will be perturbed. We selected to use Dantzig's pivoting rule since the steepest edge variant that we implemented is quite expensive and thus, we use it only when degeneracy is detected. When stalling occurs, our algorithm automatically perturbs the upper and lower bounds by adding a small positive number (we use a value of $1e-6$) to the bounds. After the solution of the perturbed problem, we remove the perturbation by resetting the problem to its original values. All algorithms share the same data structures and sparse linear algebra routines. All LPs have been presolved and scaled (using the equilibration scaling technique [58]) prior to the execution of each algorithm. The basis update method used in all algorithms is the Product Form of the Inverse [59]. In order to guarantee the accuracy, we compute from scratch the inverse of the basis every 80 iterations.

Table 4 presents the execution time and the number of iterations of each algorithm over the Netlib and Mészáros set of LPs, while Figure 3 presents the performance profile based on the execution time of the algorithms. The performance profile is displayed in logarithmic scale with base 2 using a tool developed in [60]. We also report the number of iterations performed by Mehrotra's Predictor-Corrector method ("Interior Iter") in order to initialize the proposed hybrid algorithm (HYBRID). A limit of $1,000$ seconds was set, so symbol "-" denotes that this algorithm did not find an optimal solution in the specific time interval. HYBRID is able to solve all instances, while PDIPSA did not solve three instances (rail507, rail516, and rail582) and RSA did not solve eleven instances (aa5, aa6, d6cube, jendrec1, nsir2, p010, rail507, rail516, rail582, scfxm1-2b-64, and slptsk). In addition, we report the geometric mean of the execution time and the number of iterations for all algorithms. We also report the geometric mean of the execution time and the number of iterations for the instances solved by all three algorithms (shown in parentheses in the last row of the table).

When considering all problems, HYBRID is 1.53× faster than PDIPSA and 2.1× faster than RSA. Moreover, HYBRID performs 1.36× less iterations than PDIPSA and 1.69× less iterations than RSA. When considering only the instances that all three algorithms can solve, HYBRID is 1.49× faster than PDIPSA and 1.73× faster than RSA. Moreover, HYBRID performs 1.34× less iterations than PDIPSA and 1.57× less iterations than RSA. Taking also into account that PDIPSA and RSA fail to solve some instances, HYBRID is superior to PDIPSA and RSA on these benchmark instances. Finally, HYBRID has better or equal performance than PDIPSA and RSA on 70 (84.3%) and 65 (78.3%) instances, respectively.

Table 4.: Execution time and number of iterations

Problem	**PDIPSA**		**HYBRID**			**RSA**	
	Time	**Iter**	**Time**	**Iter**	**Interior Iter**	**Time**	**Iter**
aa4	17.31	4,565	15.49	4,430	2	49.70	14,833
aa5	102.29	8,096	77.24	6,826	2	-	-
aa6	46.26	5,377	40.54	5,053	2	-	-
adlittle	0.03	77	0.03	89	2	0.03	97

afiro	0.01	17	0.01	14	3	0.01	14
agg	0.10	149	0.06	77	7	0.06	83
agg2	0.38	296	0.18	173	5	0.10	138
agg3	0.36	291	0.20	195	5	0.10	138
aircraft	5.44	1,989	5.27	1,675	1	10.84	4,034
beaconfd	0.04	86	0.04	83	4	0.02	47
blend	0.02	59	0.01	37	5	0.02	76
bnl2	12.90	2,149	11.37	1,630	20	71.19	3,921
car4	3.86	2,798	3.50	2,163	10	17.21	10,349
cari	0.87	459	1.62	494	1	1.69	1,116
cr42	1.06	625	0.87	571	5	1.13	581
cre-a	11.93	2,993	10.34	2,892	1	72.01	4,899
d6cube	146.95	7,528	115.43	6,243	15	-	-
fffff800	0.20	170	0.21	191	1	0.25	399
fit1d	4.54	626	2.53	651	3	2.24	1,773
forplan	0.28	244	0.22	200	3	0.07	180
fxm2-6	4.19	1,303	2.59	895	3	8.01	1,868
fxm3_6	98.51	5,168	71.56	2,673	10	128.05	6,532
gen	33.06	8,162	16.83	5,278	1	96.48	15,304
gen1	35.75	8,162	19.47	6,434	2	210.56	20,112
gfrd-pnc	0.64	335	0.53	331	6	1.61	550
iiasa	2.97	2,448	2.19	1,953	8	2.83	1,966
israel	0.16	313	0.11	166	8	0.10	262
jendrec1	333.20	9,230	113.76	4,005	5	-	-
lotfi	0.09	195	0.07	180	3	0.17	123
maros-r7	50.87	2,631	43.84	2,419	10	82.11	3,310
nsic1	0.31	552	0.13	307	1	0.14	405
nsic2	0.19	283	0.18	270	6	0.16	432
nsir1	113.88	5,206	21.22	2,631	2	32.89	3,547
nsir2	46.48	2,845	32.86	2,676	5	-	-
osa-07	7.60	897	6.67	631	15	5.58	719
osa-14	31.19	1,879	34.30	1,421	10	52.83	2,512
osa-30	138.23	3,881	133.10	2,813	10	284.17	4,889
p05	27.97	1,829	34.60	1,820	1	536.28	3,118
p010	175.27	3,648	190.61	3,539	1	-	-
pgp2	10.17	5,138	9.62	4,855	2	17.91	6,024
primagaz	12.30	2,253	11.95	2,208	1	109.09	6,098
r05	34.33	1,763	35.56	1,761	1	506.04	3,101
rail507	-	-	266.36	3,635	5	-	-
rail516	-	-	222.02	4,734	5	-	-
rail582	-	-	258.71	3,231	5	-	-
rat1	7.70	1,613	7.05	1,589	10	44.71	2,901
rat5	23.80	2,015	21.83	1,928	12	35.51	3,107
rat7a	81.57	2,866	66.17	2,475	15	127.08	4,221
recipe	0.02	32	0.02	23	6	0.02	48
rosen2	3.14	990	2.21	734	1	17.61	4,161
rosen7	0.21	195	0.11	159	3	0.30	517
rosen8	1.09	484	0.45	338	2	1.13	988
rosen10	14.91	1,828	7.04	1,329	4	36.15	4,777
sc105	0.03	76	0.03	66	5	0.02	68
sc205	0.17	180	0.10	159	2	0.09	166

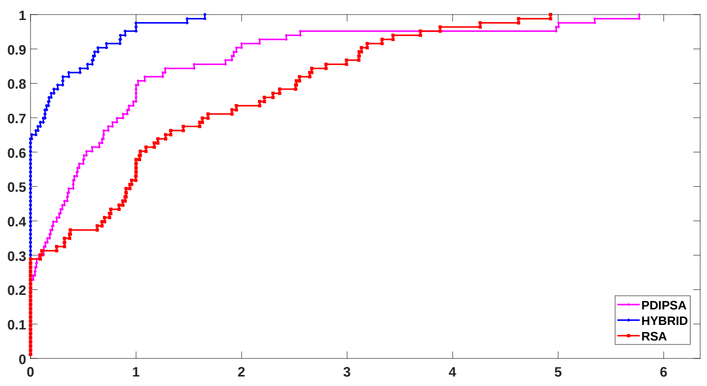

Figure 3. Performance profile based on the execution time of the three algorithms

sc205-2r-400	5.37	924	0.28	47	1	0.17	51
sc205-2r-800	39.88	1,685	1.48	87	1	0.98	91
sc205-2r-1600	314.74	3,364	11.44	167	1	9.81	171
sc50a	0.02	39	0.01	32	5	0.02	27
sc50b	0.01	32	0.01	31	5	0.01	29
scagr25	0.3	215	0.17	149	12	0.79	462
scagr7	0.04	82	0.03	76	2	0.03	82
scagr7-2b-64	126.85	2,617	21.56	2,937	3	36.23	4,278
scagr7-2r-216	13.32	2,616	13.43	2,503	3	24.95	4,653
scagr7-2r-432	98.28	5,238	94.37	5,248	3	188.36	9,629
scfxm1	0.36	344	0.32	248	10	0.21	349
scfxm1-2b-64	828.32	4,820	625.38	3,878	1	-	-
scfxm3	4.57	1,310	2.83	829	9	3.36	1090
scrs8	0.56	414	0.63	450	3	0.55	583
sctap1	0.12	282	0.12	241	1	0.22	387
sctap2	0.36	362	0.65	429	1	3.11	1042
sctap3	0.95	623	1.48	636	1	5.56	1155
share1b	0.11	148	0.09	112	1	0.07	155
share2b	0.05	97	0.04	84	5	0.05	136
ship12l	0.22	176	0.32	303	2	0.45	204
ship12s	0.15	326	0.16	335	1	0.16	311
slptsk	143.47	1,301	99.37	1,193	5	-	-
standata	0.26	362	0.18	228	1	0.13	216
stocfor1	0.01	17	0.01	22	2	0.02	30
stocfor2	4.29	784	5.3	914	2	9.9	1214
stocfor3	330.85	4,955	259.15	2,984	2	584.23	10,142
testbig	43.07	804	2.21	803	1	0.79	804
zed	0.01	29	0.01	25	3	0.02	50
Geometric mean	2.94 (1.49)	791.58 (585.85)	1.92 (1.01)	583.10 (438.67)	3 (3)	4.04 (1.74)	984.16 (690.60)

5. Conclusions

Some combinations of LP algorithms have been already proposed in the literature. In this paper, we study the combination of an IPM and an EPSA algorithm. More specifically, we used Mehrotra's Predictor-Corrector method and PDIPSA, a primal-dual interior point simplex algorithm. Our hybrid approach starts running Mehrotra's Predictor-Corrector method for a number of iterations in order to calculate a "good" interior point. Then, it initializes PDIPSA with a dual feasible basic partition and the interior point. Finally, PDIPSA continues solving the problem. Our aim is to take full advantage of both LP algorithms; use Mehrotra's Predictor-Corrector method at the first iterations which lead to significant enhancement of the objective function's value and then, use PDIPSA at the latter iterations which lead to fast convergence to an optimal solution. PDIPSA was utilized because of its behavior to the problem of stalling and cycling which enhances its computational performance and makes it one of the most efficient variations of EPSA.

A computational study was also presented with experiments over the Netlib (optimal and Kennington) and the Mészáros collection. Computational results showed that the proposed hybrid algorithm can improve PDIPSA's execution time significantly. More specifically, the proposed hybrid algorithm is $1.53\times$ faster than PDIPSA and it performs $1.36\times\%$ less iterations than PDIPSA. In addition, the proposed hybrid algorithm is on average $2.1\times$ faster than the primal revised simplex algorithm.

References

[1] Dantzig, G.B.: Linear programming and extensions. Princeton, NJ: Princeton University Press (1963)

[2] Elhallaoui, I., Metrane, A., Desaulniers, G., Soumis, F.: An improved primal simplex algorithm for degenerate linear programs. INFORMS Journal on Computing 23(4), 569-577 (2011)

[3] Bland, R.G.: New finite pivoting rules for the simplex method. Mathematics of Operations Research 2, 103-107 (1977)

[4] Forrest, J.J., Goldfarb, D.: Steepest-edge simplex algorithms for linear programming. Mathematical Programming 57, 341-374 (1992)

[5] Harris, P.M.J.: Pivot selection methods of the Devex LP code. Mathematical Programming Study 4, 30-57 (1975)

[6] Ploskas, N., Samaras, N.: Pivoting rules for the revised simplex algorithm. Yugoslav Journal of Operations Research 24(3), 321-332 (2014)

[7] Bartels, R.H., Golub, G.H.: The simplex method of linear programming using LU decomposition. Communications of the ACM 12(5), 266-268 (1969)

[8] Forrest, J.J., Tomlin, J.A.: Updated triangular factors of the basis to maintain sparsity in the product form simplex method. Mathematical Programming 2(1), 263-278 (1972)

[9] Reid, J.K.: A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. Mathematical Programming 24(1), 55-69 (1982)

[10] Suhl, L.M., Suhl, U.H.: A fast LU update for linear programming. Annals of Operations Research 43(1), 33-47 (1993)

[11] Elble, J. M., Sahinidis, N. V.: A review of the LU update in the simplex algorithm. International Journal of Mathematics in Operational Research 4(4), 366-399 (2012)

[12] Hager, W.W.: The LP dual active set algorithm. In High Performance Algorithms and Software in Nonlinear Optimization (pp. 243-254). Springer US (1998)

[13] Hager, W.W.: The dual active set algorithm and its application to linear programming. Computational Optimization and Applications 21(3), 263-275 (2002)

[14] Pan, P.Q.: The most-obtuse-angle row pivot rule for achieving dual feasibility: a computational study. European Journal of Operational Research 101(1), 164-176 (1997)
[15] Pan, P.Q.: A revised dual projective pivot algorithm for linear programming. SIAM Journal on Optimization 16(1), 49-68 (2005)
[16] Khachiyan, L.G.: Polynomial algorithms in linear programming. USSR Computational Mathematics and Mathematical Physics 20(1), 53-72 (1980)
[17] Karmarkar, N.K.: A new polynomial time algorithm for linear programming. Combinatorica 4, 373-395 (1984)
[18] Gondzio, J.: Interior point methods 25 years later. European Journal of Operational Research 218(3), 587-601 (2012)
[19] Wright, S.J.: Primal-dual interior-point methods. Siam (1997)
[20] Al-Sultan, K.S., Murty, K.G.: Exterior point algorithms for nearest points and convex quadratic programs. Mathematical Programming 57(1-3), 145-161 (1992)
[21] Sherali, H.D., Ozdaryal, B., Adams, W.P., Attia, N.: On using exterior penalty approaches for solving linear programming problems. Computers & Operations Research 28(11), 1049-1074 (2001)
[22] Paparrizos, K.: An exterior point simplex algorithm for (general) linear programming problems. Annals of Operations Research 47, 497-508 (1993)
[23] Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Research Logistic Quarterly 2(1-2), 83-97 (1955)
[24] Iri, M.: A new method of solving transportation-network problems. Journal of the Operations Research Society of Japan 3(1), 2 (1960)
[25] Zionts, S.: The criss-cross method for solving linear programming problems. Management Science 15(7), 426-445 (1969)
[26] Terlaky, T.: A convergent criss-cross method. Optimization 16(5), 683-690 (1985)
[27] Paparrizos, K.: An infeasible exterior point simplex algorithm for assignment problems. Mathematical Programming 51(1-3), 45-54 (1991)
[28] Glavelis, T., Samaras, N.: An experimental investigation of a primal-dual exterior point simplex algorithm, Oprimization: A Journal of Mathematical Programming and Operations Research 62(8), 1143-1152 (2013)
[29] 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)
[30] Samaras, N.: Computational improvements and efficient implementation of two path pivoting algorithms. Ph.D. dissertation, Department of Applied Informatics, University of Macedonia (2001)
[31] Paparrizos, K., Samaras, N., Sifaleras, A.: Exterior point simplex-type algorithms for linear and network optimization problems. Annals of Operations Research 229(1), 607-633 (2015)
[32] Hall, J.A.J.: Towards a practical parallelisation of the simplex method. Computational Management Science 7(2), 139-170 (2010)
[33] Hall, J.A.J., McKinnon, K.I.M.: PARSMI, a parallel revised simplex algorithm incorporating minor iterations and Devex pricing. In Wasniewski, J., Dongarra, J., Madsen, K., Olesen, D. (eds.) Applied Parallel Computing, Lecture Notes in Computer Science, Springer, 1184, 67-76 (1996)
[34] Hall, J.A.J., McKinnon, K.I.M.: ASYNPLEX, an asynchronous parallel revised simplex algorithm. Annals of Operations Research 81, 27-49 (1998)
[35] 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), 2-6 September, Constanta, Romania (2009)
[36] Ploskas, N., Samaras, N., Margaritis, K.: A parallel implementation of the revised simplex algorithm using OpenMP: some preliminary results. In Optimization Theory, Decision Making, and Operations Research Applications (pp. 163-175). Springer New York (2013)
[37] Thomadakis, M.E., Liu, J.C.: An efficient steepest-edge simplex algorithm for SIMD

computers. In Proceedings of the 10th International Conference on Supercomputing (ICS 1996), Philadelphia, Pennsylvania, USA, 286-293 (1996)

[38] Lalami, M.E., El-Baz, D., Boyer, V.: Multi GPU implementation of the simplex algorithm. In Proceedings of the 2011 IEEE 13th International Conference on High Performance Computing and Communications (HPCC), Banff, Canada, 179-186 (2011)

[39] Mamalis, B., Pantziou, G.: Advances in the Parallelization of the Simplex Method. In Algorithms, Probability, Networks, and Games (pp. 281-307). Springer International Publishing (2015)

[40] Meyer, X., Albuquerque, P., Chopard, B.: A multi-GPU implementation and performance model for the standard simplex method. In Proceedings of the 1st International Symposium and 10th Balkan Conference on Operational Research, Thessaloniki, Greece, 312-319 (2011)

[41] Ploskas, N., Samaras, N.: GPU accelerated pivoting rules for the simplex algorithm. Journal of Systems and Software 96, 1-9 (2014)

[42] Ploskas, N., Samaras, N.: Efficient GPU-based implementations of simplex type algorithms. Applied Mathematics and Computation 250, 552-570 (2015)

[43] Ploskas, N., Samaras, N.: A computational comparison of scaling techniques for linear optimization problems on a graphical processing unit. International Journal of Computer Mathematics 92(2), 319-336 (2015)

[44] Andersen, E.D., Ye, Y.: Combining interior–point and pivoting algorithms for linear programming. Management Science 42(12), 1719-1731 (1996)

[45] Bixby, R.E., Gregory, J.W., Lustig, I.J., Marsten, R.E., Shanno, D.F.: Very large-scale linear programming: a case study in combining interior point and simplex methods. Operations Research 10(5), 885-897 (1992)

[46] Bixby, R.E., Saltzman, M.J.: Recovering an optimal basis from an interior point solution. Operations Research Letters 15(4), 169-178 (1993)

[47] Kortanek, K.O., Zhu, J.: New purification algorithms for linear programming. Naval Research Logistic Quarterly 35, 571–583 (1988)

[48] Triantafyllidis, C.P.: A non-monotonic infeasible interior–exterior point algorithm for linear programming. PhD thesis, University of Macedonia, Greece (2014)

[49] Megiddo, N.: On finding primal- and dual-optimal Bases. ORSA Journal on Computing 3(1), 63-65 (1991)

[50] Al-Najjar, C., Malakooti, B.: Hybrid–LP: Finding advanced starting points for simplex and pivoting LP methods. Computers & Operations Research 38(2), 427-434 (2011)

[51] Pan, P.Q.: An affine-scaling pivot algorithm for linear programming. Optimization 62(4), 431-445 (2013)

[52] Mehrotra, S.: On the implementation of a primal-dual interior point method. SIAM Journal on optimization 2, 575-601 (1992)

[53] Paparrizos, K., Samaras, N., Stephanides, G.: A new efficient primal dual simplex algorithm. Computers & Operations Research 30(9), 1383-1399 (2003)

[54] Carolan, W.J., Hill, J.E., Kennington, J.L., Niemi, S., Wichmann, S.J.: An empirical evaluation of the KORBX algorithms for military airlift applications. Operations Research 38(2), 240-248 (1990)

[55] Gay, D.M.: Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Newsletter 13, 10-12 (1985)

[56] Mészáros, C.: Linear programming test problems. `http://www.sztaki.hu/~meszaros/public_ftp/lptestset/` (2015). Accessed 18 October 2017.

[57] Ordóñez, F., Freund, R.: Computational experience and the explanatory value of condition measures for linear optimization. SIAM Journal on Optimization 14(2), 307-333 (2003)

[58] Ploskas, N., Samaras, N.: The impact of scaling on simplex type algorithms. In Proceedings of the 6th Balkan Conference in Informatics (pp. 17-22). ACM (2013)

[59] Ploskas, N., Samaras, N.: A computational comparison of basis updating schemes for the simplex algorithm on a CPU-GPU system. American Journal of Operations Research 3, 497-505 (2013)

[60] Dolan, E. D., Moré, J. J.: Benchmarking optimization software with performance

profiles. Mathematical Programming 91(2), 201-213 (2002)