Postponing the choice of penalty parameter and step length

Fernando R. Villas-Bôas* and Clovis Perin†

Abstract. We study, in the context of interior-point methods for linear programming, some possible advantages of postponing the choice of the penalty parameter and the steplength, which happens both when we apply Newton's method to the Karush-Kuhn-Tucker system and when we apply a predictor-corrector scheme. We show that for a Newton or a strictly predictor step the next iterate can be expressed as a linear function of the penalty parameter μ, and, in the case of a predictor-corrector step, as a quadratic function of μ. We also show that this parameterization is useful to guarantee either the non-negativity of the next iterate or the proximity to the central path. Initial computational results of these strategies are shown and compared with PCx, an implementation of Mehotra's predictor-corrector method.

RESUMO. No contexto dos métodos de pontos interiores para programação linear, estudamos algumas possveis vantagens em postergar a escolha do parâmetro de penalização e do tamanho de passo quando o método de Newton aplicado no esquema preditor-corretor do sistema das condies de Karush-Kuhn-Tucker. Ns mostramos que o prximo iterando pode ser expresso como uma função linear (quadrática) do parâmetro de centragem μ no esquema preditor puro (preditor-corretor). Também mostramos que esta parametrização é útil para garantir a não-negatividade do próximo iterando ou a proximidade da trajetória central. Resultados computacionais iniciais são apresentados e comparados com o programa PCx, uma implementção autorizada de método preditor-corretor de Mehotra.

1. Introduction

Many primal-dual methods for linear programming use a logarithmic barrier function in order to generate a family of problems penalized by a parameter μ; each problem is solved approximately and the penalty parameter μ is reduced at each iteration, forming a sequence that converges to zero. Some of these methods have to use also a steplength α because of the non-negativity restriction. This approach is attributed to Frisch [4] and is studied in Fiacco & McCormick [3] in the context of nonlinear optimization but was first considered for linear programming by Gill et al. [5].

In practical implementations, such as Mehrotra's [12] predictor-corrector method in PCx [2], the choices of the penalty parameter and of the steplength are done using heuristics that are very effective in practice — but we lack the analytical tools to understand why they work so fine.

In this work we propose to establish an analytical environment that allows us to state the choices of penalty parameter and steplength as a clear optimization subproblem. This is done by expressing the new iterate as a quadratic function of μ for the predictor-corrector method or as a linear function for a strict Newton step, and easily extended for damped Newton steps.

More generally, whatever the *step equations* used, i.e., either the Newton equations or any of the different versions of the predictor-corrector equation for the next step,

*University of Campinas. This research was sponsored by FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo under grant number 95/6979-9

†Department of Applied Mathematics - University of Campinas

damped or not, it is always possible to parameterize the next iterate in terms of μ and the steplength α. Such parameterization is always a polynomial function of μ and α and this happens because step equations are linear.

However, performing the computation of the vectors that form this parameterization of the next iterate presents positive as well as negative side effects.

As a positive consequence, we know by means of a formula involving known vectors how the next iterate will be, without having to choose before hand the values of μ and α. This allows us to postpone such choice and focus our attention on the criteria that a particular algorithm requires of the next iterate in order to converge, i.e. non negative criteria or proximity to the central path. Having postulated and guaranteed that the next iterate will satisfy such convergence criteria, without establishing yet the values of μ and α, we can then choose "optimal" values of μ and α, in the sense of guaranteeing a faster convergence. Such postponed "optimal" choice depends upon locally minimizing a suitable function of the next iterate, such as the *dual gap* or the *complementarity* $x^T z$.

As a negative consequence, we have to perform extra system solves for each vector multiplying a power of μ. For example, in a primal-dual algorithm without corrector step, after obtaining the Cholesky factor, it is necessary to solve only one linear system after choosing μ, whereas if such choice is postponed we have to solve two systems. As another example, in a standard predictor-corrector algorithm it is necessary to solve two systems for each Cholesky factor, as opposed to three system solves in our case. This difficulty is structural and inherent to postponing the choice of μ.

Furthermore, despite we are using the same Cholesky factor, the overhead of the extra system solves is *not* negligible. In a sample of typical Netlib problems that we studied, the time to solve one positive-definite system once we have the Cholesky factor is in average 20% of the time needed to compute the Cholesky factor itself, and this average has a high variance. This result goes against our intuition stemming from dense problems, where such times are order of magnitude smaller than the time to compute one Cholesky factor.

The facts above impose an initial limit to what we could expect from implementations that postpone the choice of μ and α — we should only expect a reduction in CPU times if the reduction in number of iterations is big enough as compared to an algorithm of reference.

But what we aim more to show in this work is that, for a given step equation system based on the logarithmic barrier function, parameterizing the next iterate in terms of a μ and α to be chosen later provides us the means to evaluate and improve the other parts of the given algorithm.

We will show this as follows.

First we will show how the parameterization can be done by explicitly doing it in a few examples of increasing complexity, i.e. we will develop the expressions in a standard primal-dual context, then extend them to include corrector steps, assuming in both cases that an initial feasible point is available.

Second we show how the parameterization itself can be used.

Since proximity to the central path is one of the few criteria available to guarantee convergence, we chose it as the main criterion that the next iterate has to satisfy, and using different proximity criteria we generate a family of algorithms having different polynomial complexities.

We then extend these ideas to a self-dual context in order to relax the requirement of an initial feasible point.

Self-dual methods were originally studied by Goldman and Tucker [6] [18] and were

rediscovered by Todd and Mizuno [22] in the context of interior point methods. An initial implementation by Xu, Hung and Ye [20] showed promising results for Netlib problems, especially for a new class of infeasible problems. Other recent works in self-dual algorithms are those of Xu [19], Xu and Ye [21], and Hung and Ye [8].

We chose to perform iterations in a self-dual space in order to be free from feasibility issues — in this framework given any initial non feasible solution we can always generate a problem that in a sense is equivalent to the original linear program, and where all iterates are feasible. The self-dual problem always has a finite optimal solution that we can use either to find the solution to the original problem or to declare it infeasible or unbounded.

We develop the expressions of the parameterization in a usual primal-dual context assuming that an initial feasible interior point is available. We do so in order to present our ideas in a simpler and clearer way. When developing the expressions for a real implementation we used the self-dual framework with bounded variables. Unfortunately in this case the notation becomes excessively heavy and a little confusing.

To check the practical usefulness of the various algorithms based on the proposed parameterization, we use PCx library, i.e., we use from the original implementation the same routines for input, linear algebra, preconditioning, initial point and stopping criteria. This way all comparisons became fair and meaningful. For each type of neighborhood used we made a different implementation, so that computational results are available for comparison.

Computational results showed that we achieved less iteration count in most of the problems studied, although computational times were bigger.

We believe that these results are positive. Although the CPU times were bigger, the reduction in number of iterations showed that the parameterization actually allowed for a clearer insight of what are the real subproblems to be solved at each iteration.

Besides that, working in a self-dual framework in a predictor-corrector algorithm forces us to solve seven linear systems for each Cholesky factor, which means five extra system solves as compared to Mehrotra's predictor-corrector algorithm. This way, practical implementations using the self-dual framework become impossible for our parameterization.

This paper is organized as follows. In section 2 we develop the linear parameterization for strict Newton step. In section 3 we extend this idea to a predictor-corrector and damped step. We present an overview of the various neighborhoods of the central path in section 4 and our general path-following algorithm in section 5. Some initial computational results are shown in section 8.

2. First degree parameterization in a primal-dual context

We consider the standard linear programming problem and its dual

$$(P)\ \min\left\{c^T x : Ax = b,\ x \geq 0\right\}\ \text{and}\ \ (D)\ \max\left\{b^T y : A^T y + z = c,\ z \geq 0\right\}$$

where A is a $n \times m$ matrix and b and c are vectors of dimension m and n respectively.

We assume that the sets

$$\mathcal{F}_P = \{x \in \mathbb{R}^n \mid Ax = b,\ x > 0\}\ \text{and}\ \mathcal{F}_D = \left\{(y,z) \in \mathbb{R}^m \times \mathbb{R}^n \mid A^T y + z = c,\ z > 0\right\}$$

are both non empty and define $\mathcal{F} = \mathcal{F}_P \times \mathcal{F}_D$.

We solve (P) using the logarithmic barrier function technique that consists of solving

approximately the family of problems

$$(P_\mu) \; : \; \min \left\{ c^T x - \mu \sum_{j=1}^{n} \ln x_j : Ax = b, \; x > 0 \right\},$$

where $\mu > 0$ is the penalty parameter and should be chosen in a way that forms a sequence converging to zero. For each μ the problem P_μ has an objective function that is strictly convex and so the global minimum is completely characterized by the Karush-Kuhn-Tucker stationary condition

$$\begin{array}{l} Ax - b = 0 \\ A^T y + z - c = 0 \\ xz - \mu e = 0 \\ x > 0, \;\; z > 0, \end{array} \tag{1}$$

where e is the vector of dimension n consisting only of ones and we adopt the convention that if x and z are vectors of same dimension then xz denotes the vector whose components are $x_i z_i$. The internal product will be denoted by $x^T z$.

The above notation xz to indicate the componentwise product of two vectors x and z with same dimension has been established in recent works in interior point methods [9][15] [16], due to the frequency that this concept is used and in order to simplify the notation. In a similar way, if x is a vector and a is a scalar we denote by x^a the vector having components $(x_i)^a$.

In order to solve approximately each problem P_μ and defining $w \equiv (x, y, z)$ we solve also approximately the non linear system

$$H(w) = H(x, y, z) = \begin{pmatrix} Ax - b \\ A^T y + z - c \\ xz - \mu e \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{2}$$

using one iteration of Newton's method, for which we need the Jacobian of H:

$$J = \begin{pmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{pmatrix}.$$

where X and Z are the diagonal matrices whose elements are equal to x and z respectively. In a way similar to what we did for vectors, we extend the power notation for diagonal matrices: if X is a diagonal matrix and a is a scalar then we define X^a as being the diagonal matrix whose nonzero elements are $(X_{i,i})^a$

We assume that the present point (x, y, z) is a feasible interior point, i.e.

$$\begin{array}{rcl} \begin{pmatrix} Ax - b \\ A^T y + z - c \end{pmatrix} & = & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ x & > & 0, \\ z & > & 0. \end{array}$$

The Newton direction $\delta w = (\delta x, \delta y, \delta z)$ that provides the approximate solution $\hat{w} = (\hat{x}, \hat{y}, \hat{z}) = (x + \delta x, y + \delta y, z + \delta z)$ for the system $H(w) = 0$ is given by the solution of the

linear system $J(w)\,\Delta w = -H(w)$, i.e.,

$$\begin{array}{l} A\Delta x = 0 \\ A^T\delta y + \delta z = 0 \\ Z\delta x + X\delta z = \mu e - xz. \end{array} \tag{3}$$

If we define

$$\begin{aligned} \mathcal{P}_x &= Z^{-1} - Z^{-1}XA^T\left(AZ^{-1}XA^T\right)^{-1}AZ^{-1} \\ \mathcal{P}_y &= \left(AZ^{-1}XA^T\right)^{-1}AZ^{-1} \\ \mathcal{P}_z &= A^T\left(AZ^{-1}XA^T\right)^{-1}AZ^{-1} \end{aligned}$$

then by direct calculation we obtain

$$\begin{array}{l} \delta x = \mathcal{P}_x(\mu e - xz) \\ \delta y = -\mathcal{P}_y(\mu e - xz) \\ \delta z = \mathcal{P}_z(\mu e - xz) \end{array} \tag{4}$$

If now we define

$$\left.\begin{array}{l} \gamma_0^x = -\mathcal{P}_x xz \\ \gamma_1^x = \mathcal{P}_x e \\ \gamma_0^y = \mathcal{P}_y xz \\ \gamma_1^y = -\mathcal{P}_y e \\ \gamma_0^z = \mathcal{P}_z xz \\ \gamma_1^z = -\mathcal{P}_z e \end{array}\right\} \tag{5}$$

then we have

$$\left.\begin{array}{l} \delta x = \gamma_1^x\mu + \gamma_0^x \\ \delta y = \gamma_1^y\mu + \gamma_0^y \\ \delta z = \gamma_1^z\mu + \gamma_0^z \end{array}\right\} \tag{6}$$

i.e. we have expressed each new step and therefore each new iterate as linear function of μ. Note that the vectors γ can be calculated without μ.

3. Second degree parameterization

Suppose that we try to determine a single step such that

$$\hat{w} = (\hat{x}, \hat{y}, \hat{z}) = (x + \Delta x, y + \Delta y, z + \Delta z)$$

solves exactly the nonlinear system (2). By direct substitution we see that such step must satisfy the nonlinear system

$$\begin{array}{l} A\Delta x = 0 \\ A^T\Delta y + \Delta z = 0 \\ Z\Delta x + X\Delta z = \mu e - xz - \Delta x\Delta z \end{array}$$

The idea behind a second degree step or predictor-corrector step is to consider that the nonlinear term $\Delta x\Delta z$ can be approximated by $\delta x\delta z$ where δx and δz are the solution of the linear system (3) found in the previous section (as will be shown in section 7, near the central path we have $\|\Delta x\Delta z - \delta x\delta z\| \leq \theta\mu$, i.e., $\Delta x\Delta z$ is close to $\delta x\delta z$ and this agrees with an intuitive idea of approximation).

The system now becomes linear:

$$\left.\begin{array}{l} A\Delta x = 0 \\ A^T\Delta y + \Delta z = 0 \\ Z\Delta x + X\Delta z = \mu e - xz - \delta x\delta z \end{array}\right\} \tag{7}$$

By direct calculation we obtain

$$\left.\begin{array}{l} \Delta x = \mathcal{P}_x\left(\mu e - xz - \delta x\delta z\right) \\ \Delta y = -\mathcal{P}_y\left(\mu e - xz - \delta x\delta z\right) \\ \Delta z = \mathcal{P}_z\left(\mu e - xz - \delta x\delta z\right) \end{array}\right\} \tag{8}$$

But according to (6) we can express δx and δz as a linear function of μ and so we have $\mu e - xz - \delta x\delta z = \mu e - xz - \left(\gamma_1^x\mu + \gamma_0^x\right)\left(\gamma_1^z\mu + \gamma_0^z\right) = -\gamma_1^x\gamma_1^z\mu^2 + \left(e - \gamma_0^x\gamma_1^z - \gamma_1^x\gamma_0^z\right)\mu - xz - \gamma_0^x\gamma_0^z$.

We now define

$$\left.\begin{array}{l} \psi_2^x = -\mathcal{P}_x\gamma_1^x\gamma_1^z \\ \psi_1^x = \mathcal{P}_x\left(e - \gamma_0^x\gamma_1^z - \gamma_1^x\gamma_0^z\right) \\ \psi_0^x = -\mathcal{P}_x\left(xz + \gamma_0^x\gamma_0^z\right) \\ \psi_2^y = \mathcal{P}_y\gamma_1^x\gamma_1^z \\ \psi_1^y = \mathcal{P}_y\left(\gamma_0^x\gamma_1^z + \gamma_1^x\gamma_0^z - e\right) \\ \psi_0^y = \mathcal{P}_y\left(xz + \gamma_0^x\gamma_0^z\right) \\ \psi_2^z = -\mathcal{P}_z\gamma_1^x\gamma_1^z \\ \psi_1^z = \mathcal{P}_z\left(e - \gamma_0^x\gamma_1^z - \gamma_1^x\gamma_0^z\right) \\ \psi_0^z = -\mathcal{P}_z\left(xz + \gamma_0^x\gamma_0^z\right) \end{array}\right\} \tag{9}$$

so that the final step can be expressed as a second degree function of μ

$$\begin{array}{rcl} \Delta x & = & \psi_2^x\mu^2 + \psi_1^x\mu + \psi_0^x \\ \Delta y & = & \psi_2^y\mu^2 + \psi_1^y\mu + \psi_0^y \\ \Delta z & = & \psi_2^z\mu^2 + \psi_1^z\mu + \psi_0^z \end{array}$$

In the case of a damped step the next iterate $\hat{w}$ can be expressed as a function of α and μ since

$$\begin{array}{rcl} \hat{w}\left(\alpha,\mu\right) & = & \left(x,y,z\right) + \alpha\left(\Delta x,\Delta y,\Delta z\right) \\ & = & \left[\begin{array}{l} x + \alpha\left(\psi_2^x\mu^2 + \psi_1^x\mu + \psi_0^x\right) \\ y + \alpha\left(\psi_2^y\mu^2 + \psi_1^y\mu + \psi_0^y\right) \\ z + \alpha\left(\psi_2^z\mu^2 + \psi_1^z\mu + \psi_0^z\right) \end{array}\right] \end{array} \tag{10}$$

The next theorem gives the expression of the dual gap and will be needed for the algorithm. We need a simple lemma.

Lemma 1. *Vectors δx and δz are orthogonal; vectors Δx and Δz are orthogonal*

Proof: We reproduce the proof in [13]. δx and δz satisfy system (3). Multiply the first equation by δy^T and the second by δx and combine them to see that $\delta x^T\delta z = 0$. Use the same idea in system (7).

Theorem 2. *Let $\hat{g}$ be the dual gap after a damped second degree step and let g be the previous gap. Then $\hat{g} = \alpha n\mu + \left(1-\alpha\right)g$*

Proof:

$$\begin{aligned}\hat{g} &= \sum_{i=1}^{n} \hat{x}_i \hat{z}_i = \sum_{i=1}^{n} (x_i + \alpha \Delta x_i)(z_i + \alpha \Delta z_i) \\ &= \sum_{i=1}^{n} x_i z_i + \alpha \sum_{i=1}^{n} (x_i \Delta z_i + z_i \Delta x_i) + \alpha^2 \sum_{i=1}^{n} \Delta x_i \Delta z_i \\ &= g + \alpha \sum_{i=1}^{n} (\mu - x_i z_i - \delta x_i \delta z_i) = g + \alpha (n\mu - g)\end{aligned}$$

Note that the same theorem holds for first degree steps.

4. Neighborhoods of the central path

The central path is defined as

$$\mathcal{C} = \{(x, y, z) \in \mathcal{F} : xz = \mu e\}$$

i.e., it is a continuous path parameterized by the real variable μ — for each μ the point in the path is completely characterized as the unique solution of system (1). It was studied by Bayer and Lagarias [1], Sonnevend [17] and Megiddo [11], among others. Since the vector y is not needed in the implementations of interior point methods (IPM), we will omit them in some contexts and consider only the pair (x, z).

Quoting Hung and Ye [8], path-following algorithms generate a sequence of points within certain neighborhoods of the central path $\mathcal{C}$, which prevents iterates from prematurely getting too close to the boundary of the feasible region.

Now suppose we are given a current iterate (x, z). This iterate is a solution of a linear system involving some μ, therefore there is a unique point in the central path characterized by μ, which we call a μ-center. In order to define a neighborhood we must be able to quantify the distance from the vector xz to the μ-center. Following the ideas of Peng et al. [15], this quantification can be done defining the following proximity measures.

$$\begin{aligned}\delta_F(xz, \mu) &= \sum_{i=1}^{n} \left(\frac{x_i z_i}{\mu} - 1 - \log\left(\frac{x_i z_i}{\mu}\right) \right), \\ \delta_K(xz, \mu) &= \left\| \frac{xz}{\mu} - e \right\|, \\ \delta_J(xz, \mu) &= \frac{1}{2} \left\| \sqrt{\frac{xz}{\mu}} - \sqrt{\frac{\mu}{xz}} \right\|\end{aligned}$$

In δ_J the square root of the vector is to be understood as the vector whose components are the square roots of each component. Note that for all three measures its value is zero if xz is the point in the central path.

The first measure, known as the *logarithmic barrier function* with respect to the *barrier parameter* μ, was introduced by Frisch [4] and historically it was the first to appear. Although it is not practical for implementations, it is useful in the complexity analysis of many IPMs because it has the *barrier property*: it becomes unbounded if xz approaches the boundary of the nonnegative orthant. This measure is specially useful for *large-update*

methods, that at each iteration drastically reduce μ and then use several damped Newton steps to recenter.

The second measure δ_K was introduced by Kojima et al. [10] and is the most used measure for complexity analysis but it doesn't have the barrier property.

The third one is due to Jansen et al. [9] and seems to lead to more concise proofs of complexity, including that of damped steps algorithms. The motivation behind this measure is to be able to work in a scaled space, where the notations and proofs become more simple and where this measure is equivalent to the classical Euclidean norm. Recent works from Peng et al. [16] propose new algorithms based on this measure that might have their practical performance confirmed — but although the measure is promising, it has not produced yet outstanding results regarding complexity or practical implementations.

We will focus only on measures δ_K and δ_J and for implementation purposes we will use only δ_K.

We can now define the neighborhoods, initially using the Euclidean norm and later we can extend them using other norms.

Associated with the first proximity measure we have the classical definition

$$\mathcal{N}_2(\beta) = \left\{(x,y,z) \in \mathcal{F} : \left\| \frac{xz}{\mu} - e \right\| \leq 1 - \beta \right\}$$

where $\beta \in (0,1)$. This definition has another equivalent form

$$\mathcal{N}_2(\beta) = \{(x,y,z) \in \mathcal{F} : \|xz - \mu e\| \leq (1-\beta)\mu\}$$

and we will freely use either form as convenient.

The second neighborhood, associated with δ_J, is defined as

$$\mathcal{N}_J(\beta) = \left\{(x,y,z) \in \mathcal{F} : \left\| \sqrt{\frac{xz}{\mu}} - \sqrt{\frac{\mu}{xz}} \right\| \leq \beta \right\}$$

where $\beta > 0$

The first neighborhood has two natural extensions, $\mathcal{N}_\infty(\beta)$ where the definition is the same but the norm is l_∞, and

$$\mathcal{N}_\infty^-(\beta) = \{(x,y,z) \in \mathcal{F} : xz \geq \beta\mu\}$$

We can see that

$$\mathcal{C} \subset \mathcal{N}_2(\beta) \subset \mathcal{N}_\infty(\beta) \subset \mathcal{N}_\infty^-(\beta) \subset \mathcal{F}$$

The extension of the second one is similar.

We say that a point is close to the central path under a certain proximity criterion $\mathcal{N}$ if it belongs to any of the neighborhoods above for a given $\beta > 0$

We can now state a general path-following algorithm. However, we are not simply seeking generality for it's own sake. Rather we will restrict the general algorithm to the "smallest" framework that encompasses all the algorithms that we actually implemented and for which computational results are available.

5. Algorithm

We present here the schema for the algorithm that includes all implementations we did — this schema permits first and second order steps, and various types of neighborhoods.

Step 0. Assume that a given proximity criterion $\mathcal{N}$ is given and let $w_0 = (x_0, y_0, z_0)$ be an initial feasible point close to the central path under this proximity criterion. Let $\varepsilon > 0$ be a tolerance for the duality gap and define $\mu_0 = x_0^T z_0/n$.

Step 1. If $g_k = x_k^T z_k \leq \varepsilon$ stop.

Step 2. Compute the function $\Delta w_k(\mu)$, i.e., compute according to (5) the vectors γ that allow the new step Δw_k to be expressed as a function of μ (or in the case of a second degree parameterization compute according to (9) the vectors ψ).

Step 3. minimize (approximately) $g_{k+1}(\alpha, \mu_k) = \alpha n \mu_k + (1-\alpha) g_k$ subject to $w_{k+1} = w_k + \alpha \Delta w_k(\mu_k)$ being close to the central path.

Step 4. Set $w_{k+1} = w_k + \alpha \Delta w_k$ and $k := k+1$ and go to step 1.

In order to show that the above algorithm is well defined we must show that it terminates properly and that all steps can be executed. There is no problem for steps 1 and 4. For step 0 the problem of finding an initial feasible point close to the central will be dealt with when we extend our ideas to a self-dual framework. Sections 2 and 3 showed that step 2 presents no problem. Step 3 originates by itself a nonlinear subproblem that is well defined — we will see that we can state and solve this problem approximately (or even exactly for some proximity criteria), provided that the restriction set is non-empty. We will show that this is the case depending on the proximity criterion chosen.

Also depending on the proximity criterion there will be different polynomial complexities associated. In the next section we will elaborate more on the restriction set for step 3 and on the polynomial complexities.

6. Convergence results

The results in this section guarantee the convergence of the algorithm described in the previous section for the neighborhoods $\mathcal{N}_2(\beta)$ and $\mathcal{N}_\infty^-(\beta)$, for specific values of β that include those used in our implementations. Note that the complexity of the algorithm is polynomial.

Theorem 3. *Let $\mathcal{N}$ be the criterion in the algorithm defined by $\beta \in (0, 1/2)$ and neighborhood $\mathcal{N}_2(\beta)$ and assume that only first degree steps are taken. Then step 3 admits an approximate minimization. Furthermore the algorithm stops in $\mathcal{O}(\sqrt{n}L)$ steps.*

Proof: This is the classical primal-dual algorithm of Monteiro and Adler and the proof can be found in [13].

In order to prove convergence results for predictor-corrector steps we will need some technical lemmas. The following lemma, due to Ye [23], is a variation with tighter bounds of a lemma first presented by Monteiro [13].

Lemma 4. *Let p and q be two n-dimensional real vectors such that $p^T q \geq 0$ and $p+q = r$. Then*

$$\|pq\| \leq \frac{\|r\|^2}{\sqrt{8}}$$

Proof

$$\begin{aligned}
\|pq\|^2 &= \sum_{i=1}^{n} (p_i q_i)^2 \\
&\leq \left(\sum_{p_i q_i > 0} p_i q_i \right)^2 + \left(\sum_{p_i q_i < 0} p_i q_i \right)^2 \\
&\leq 2 \left(\sum_{p_i q_i > 0} p_i q_i \right)^2 \\
&\leq 2 \left(\sum_{p_i q_i > 0} (p_i + q_i)^2 / 4 \right)^2 \\
&\leq \|r\|^4 / 8
\end{aligned}$$

Lemma 5. *Suppose that* $\|xz - \mu e\| \leq \theta\mu$ *and that* $\hat{\mu} = (1 - \tau)\,\mu$. *Then*

$$\|\delta x \delta z\| \leq \frac{(\theta + \sqrt{n}\tau)^2}{\sqrt{8}\,(1 - \theta)} \mu$$

Proof Define $D = Z^{1/2} X^{-1/2}$, $p = D\delta x$, $q = D^{-1}\delta z$. Multiply the last equation of system (3) by $(XZ)^{-1/2}$ so that $p + q = D\delta x + D^{-1}\delta z = r = (xz)^{-1/2}(\hat{\mu}e - xz)$. Noting that $\|xz - \mu e\| \leq \theta\mu$ implies $x_i z_i \geq (1 - \theta)\,\mu$ and using lemma 4 we have

$$\begin{aligned}
\|\delta x \delta z\| &= \|pq\| \leq \|r\|^2 / \sqrt{8} = \left\| (xz)^{-1/2} (\hat{\mu}e - xz) \right\|^2 / \sqrt{8} \\
&= \frac{1}{\sqrt{8}} \sum_{i=1}^{n} \frac{(x_i z_i - \hat{\mu})^2}{x_i z_i} \\
&\leq \frac{1}{\sqrt{8}} \sum_{i=1}^{n} \frac{(x_i z_i - \hat{\mu})^2}{(1 - \theta)\,\mu} \\
&= \frac{1}{\sqrt{8}\,(1 - \theta)\,\mu} \|xz - \hat{\mu}e\|^2 \\
&= \frac{1}{\sqrt{8}\,(1 - \theta)\,\mu} \|(xz - \mu e) + (\mu - \hat{\mu})\,e\|^2 \\
&\leq \frac{1}{\sqrt{8}\,(1 - \theta)\,\mu} \left(\|xz - \mu e\| + \|(\mu - \hat{\mu})\,e\| \right)^2 \\
&\leq \frac{1}{\sqrt{8}\,(1 - \theta)\,\mu} \left(\theta\mu + \sqrt{n}\tau\mu \right)^2 \\
&= \frac{(\theta + \sqrt{n}\tau)^2}{\sqrt{8}\,(1 - \theta)} \mu
\end{aligned}$$

Lemma 6. *Suppose that* $\|xz - \mu e\| \leq \theta\mu$ *and that* $\hat{\mu} = (1 - \tau)\,\mu$. *Then*

$$\|\Delta x \Delta z\| \leq \frac{1}{\sqrt{8}\,(1 - \theta)} \left(\theta + \sqrt{n}\tau + \frac{(\theta + \sqrt{n}\tau)^2}{\sqrt{8}\,(1 - \theta)} \right)^2 \mu$$

Proof Again define $D = Z^{1/2}X^{-1/2}$, $p = D\Delta x$, $q = D^{-1}\Delta z$. Multiply the last equation of system (7) by $(XZ)^{-1/2}$ so that $p+q = D\Delta x+D^{-1}\Delta z = r = (xz)^{-1/2}(\hat{\mu}e - xz - \delta x\delta z)$. Note that $-\theta\mu \leq x_i z_i - \mu \leq \theta\mu$ and so by lemma 4 and lemma 5 we have

$$\begin{aligned}
\|\Delta x\Delta z\| &= \|pq\| \leq \|r\|^2/\sqrt{8} = \left\|(xz)^{-1/2}(\hat{\mu}e - xz - \delta x\delta z)\right\|^2/\sqrt{8} \\
&= \frac{1}{\sqrt{8}}\sum_{i=1}^{n}\frac{(x_i z_i - \hat{\mu} + \delta x_i \delta z_i)^2}{x_i z_i} \\
&\leq \frac{1}{\sqrt{8}}\sum_{i=1}^{n}\frac{(x_i z_i - \hat{\mu} + \delta x_i \delta z_i)^2}{(1-\theta)\,\mu} \\
&= \frac{1}{\sqrt{8}(1-\theta)\,\mu}\|xz - \hat{\mu}e + \delta x\delta z\|^2 \\
&= \frac{1}{\sqrt{8}(1-\theta)\,\mu}\|(xz-\mu e) + (\mu-\hat{\mu})\,e + \delta x\delta z\|^2 \\
&\leq \frac{1}{\sqrt{8}(1-\theta)\,\mu}\left(\|xz-\mu e\| + \|(\mu-\hat{\mu})\,e\| + \|\delta x\delta z\|\right)^2 \\
&\leq \frac{1}{\sqrt{8}(1-\theta)\,\mu}\left(\theta\mu + \sqrt{n}\tau\mu + \frac{(\theta+\sqrt{n}\tau)^2}{\sqrt{8}(1-\theta)}\mu\right)^2 \\
&= \frac{1}{\sqrt{8}(1-\theta)}\left(\theta + \sqrt{n}\tau + \frac{(\theta+\sqrt{n}\tau)^2}{\sqrt{8}(1-\theta)}\right)^2\mu
\end{aligned}$$

Theorem 7. *Let τ and θ satisfy*

$$\frac{1}{\sqrt{8}(1-\theta)}\left(\theta + \sqrt{n}\tau + \frac{(\theta+\sqrt{n}\tau)^2}{\sqrt{8}(1-\theta)}\right)^2 + \frac{(\theta+\sqrt{n}\tau)^2}{\sqrt{8}(1-\theta)} \leq \theta(1-\tau).$$

If $\|xz-\mu e\| \leq \theta\mu$ and $\hat{\mu} = (1-\tau)\,\mu$ then $\|\hat{x}\hat{z} - \hat{\mu}e\| \leq \theta\hat{\mu}$

Proof:

$\hat{x}_i\hat{z}_i - \hat{\mu} = (x_i+\Delta x_i)(z_i+\Delta z_i) - \hat{\mu} = x_i z_i + x_i\Delta z_i + z_i\Delta x_i + \Delta x_i\Delta z_i - \hat{\mu} = \Delta x_i\Delta z_i - \delta x_i\delta z_i$

therefore

$$\begin{aligned}
\|\hat{x}\hat{z} - \hat{\mu}e\| &= \|\Delta x\Delta z - \delta x\delta z\| \leq \|\Delta x\Delta z\| + \|\delta x\delta z\| \\
&\leq \left(\frac{1}{\sqrt{8}(1-\theta)}\left(\theta + \sqrt{n}\tau + \frac{(\theta+\sqrt{n}\tau)^2}{\sqrt{8}(1-\theta)}\right)^2 + \frac{(\theta+\sqrt{n}\tau)^2}{\sqrt{8}(1-\theta)}\right)\mu \\
&\leq \theta(1-\tau)\,\mu = \theta\hat{\mu}
\end{aligned}$$

Note that if $\theta = \frac{1}{4}$ and $\tau = \min(\frac{1}{5\sqrt{n}}, \frac{1}{5\sqrt{13}})$ then the condition in theorem 7 is satisfied.

The theorem above just proved that step 3 in the general algorithm is well defined if the neighborhood used is $\mathcal{N}_2\left(\frac{1}{4}\right)$. The choice $\tau = \frac{1}{5\sqrt{n}}$ guarantees that

$$\frac{g_{k+1}}{g_k} \leq \left(1 - \frac{1}{5\sqrt{n}}\right)$$

which in turn guarantees that the algorithm finishes in $\mathcal{O}(\sqrt{n}L)$ steps.

When neighborhood $\mathcal{N}_2$ is used, the underlying minimization in step 3 is

$$\begin{array}{rcl} \min \hat{g} & = & \alpha n\mu + (1-\alpha)\, g \\ \text{s.t.} & & \\ \hat{w} & \in & \mathcal{N}_2\left(\frac{1}{4}\right) \end{array}$$

but in this case

$$\hat{w} \in \mathcal{N}_2\left(\frac{1}{4}\right) \Leftrightarrow$$
$$\left\|\hat{x}\hat{z} - \hat{\mu}e\right\|^2 \leq \frac{9}{16}\hat{\mu}^2 \Leftrightarrow$$
$$\sum_{i=1}^{n}\left(\hat{x}_i\hat{z}_i - \hat{\mu}\right)^2 \leq \frac{9}{16}\hat{\mu}^2 \Leftrightarrow$$
$$\sum_{i=1}^{n}\left(\left(x_i + \alpha\Delta x_i\right)\left(z_i + \alpha\Delta z_i\right) - \hat{\mu}\right)^2 \leq \frac{9}{16}\hat{\mu}^2 \Leftrightarrow$$
$$\sum_{i=1}^{n}\left(\left(x_i + \alpha\left(\psi_2^{x_i}\hat{\mu}^2 + \psi_1^{x_i}\hat{\mu} + \psi_0^{x_i}\right)\right)\left(z_i + \alpha\left(\psi_2^{z_i}\hat{\mu}^2 + \psi_1^{z_i}\hat{\mu} + \psi_0^{z_i}\right)\right) - \hat{\mu}\right)^2 - \frac{9}{16}\hat{\mu}^2 \leq 0$$

This last inequation can be regrouped into a 8th degree real polynomial in the variables α and $\hat{\mu}$, because the vectors ψ do not depend upon $\hat{\mu}$ and α and were previously calculated. For each fixed $\alpha \in (0,1]$ the optimization problem above can be solved exactly because the solution $\hat{\mu}$ is the smallest positive root of a polynomial. Since it is computationally cheap to find roots of a polynomial for each α, we can solve this optimization subproblem quite accurately — and we did so in one of our implementations described in section 8.

We now proceed to generalize the previous result to the wider neighborhood $\mathcal{N}_\infty^-$.

Lemma 8. *Let p and q be two n-dimensional real vectors such that $p^Tq = 0$ and $p+q = r$. Then*

$$-\frac{\|r\|^2}{4} \leq p_iq_i \leq \frac{r_i^2}{4}$$

Proof We reproduce the proof by Ye [23]. The right hand inequality follows from $p_i + q_i = r_i$. For the left hand inequality

$$\begin{array}{rcll} p_iq_i & \geq & \displaystyle\sum_{p_iq_i<0} p_iq_i & \\ & = & \displaystyle -\sum_{p_iq_i>0} p_iq_i & \quad (\text{since } p^Tq = 0) \\ & \geq & \displaystyle -\sum_{p_iq_i>0} \frac{r_i^2}{4} & \quad (\text{using the right hand inequality}) \\ & \geq & -\|r\|^2/4 & \end{array}$$

Theorem 9. *Let τ and θ satisfy*

$$-\frac{\left(\theta+\tau\sqrt{n}+\frac{(\theta+\tau)^2}{4\theta}\right)^2}{\theta}+\frac{(\theta+\tau)^2}{4\theta}+(1-\tau)\geq\theta(1-\tau).$$

If $xz\geq\theta\mu$ and $\hat{\mu}=(1-\tau)\mu$ then $\hat{x}\hat{z}\geq\theta\hat{\mu}$

Proof According to the previous lemma we have

$$\begin{aligned}
\delta x_i\delta z_i &\leq \frac{(x_iz_i-\hat{\mu})^2}{4(x_iz_i)}\leq\frac{(x_iz_i-\hat{\mu})^2}{4\theta\mu}\\
&= \frac{((x_iz_i-\mu)+(\mu-\hat{\mu}))^2}{4\theta\mu}\\
&\leq \frac{(\theta\mu+\tau\mu)^2}{4\theta\mu}\\
&= \frac{(\theta+\tau)^2}{4\theta}\mu
\end{aligned}$$

and

$$\begin{aligned}
\Delta x_i\Delta z_i &\geq -\frac{\left\|(xz)^{-1/2}(\hat{\mu}e-xz-\delta x\delta z)\right\|^2}{4}\\
&= -\sum_{i=1}^{n}\frac{(\hat{\mu}-x_iz_i-\delta x_i\delta z_i)^2}{x_iz_i}\\
&\geq -\sum_{i=1}^{n}\frac{(\hat{\mu}-x_iz_i-\delta x_i\delta z_i)^2}{\theta\mu}\\
&= -\frac{\|(\mu e-xz)+(\hat{\mu}-\mu)e-\delta x\delta z\|^2}{\theta\mu}\\
&\geq -\frac{(\|\mu e-xz\|+\|\tau\mu e\|+\|\delta x\delta z\|)^2}{\theta\mu}\\
&\geq -\frac{\left(\theta\mu+\tau\sqrt{n}\mu+\frac{(\theta+\tau)^2}{4\theta}\mu\right)^2}{\theta\mu}\\
&= -\frac{\left(\theta+\tau\sqrt{n}+\frac{(\theta+\tau)^2}{4\theta}\right)^2}{\theta\mu}\mu
\end{aligned}$$

and so

$$\begin{aligned}
\hat{x}_i\hat{z}_i &= \Delta x_i\Delta z_i-\delta x_i\delta z_i+\hat{\mu}\\
&\geq \left(-\frac{\left(\theta+\tau\sqrt{n}+\frac{(\theta+\tau)^2}{4\theta}\right)^2}{\theta}+\frac{(\theta+\tau)^2}{4\theta}+(1-\tau)\right)\mu\\
&\geq \theta(1-\tau)\mu\\
&= \theta\hat{\mu}
\end{aligned}$$

Note that a possible choice to satisfy the condition in theorem 9 is

$$\begin{aligned} \theta &= \frac{1}{5} \\ \tau &= \min\left(\frac{1}{2n}, \frac{1}{30}\right). \end{aligned}$$

Again this theorem proved that step 3 in the general algorithm is well defined if the neighborhood used is $\mathcal{N}_\infty^-\left(\frac{1}{5}\right)$. The choice $\tau = \frac{1}{2n}$ guarantees that

$$\frac{g_{k+1}}{g_k} \leq \left(1 - \frac{1}{2n}\right),$$

which in turn guarantees that the algorithm finishes in $\mathcal{O}(nL)$ steps.

When neighborhood $\mathcal{N}_\infty^-$ is used, the underlying minimization in step 3 is

$$\begin{aligned} \min\ \hat{g} &= \alpha n\mu + (1-\alpha)\, g \\ \text{s.t.} & \\ \hat{w} &\in \mathcal{N}_\infty^-(0.2) \end{aligned}$$

and similarly the condition $\hat{w} \in \mathcal{N}_\infty^-(0.2)$ is equivalent to n inequations

$$\hat{x}_i \hat{z}_i \geq 0.2\hat{\mu}.$$

Using the second degree parameterization each of these inequations are equivalent to the 4th degree polynomial inequation

$$\left(x_i + \alpha\left(\psi_2^{x_i}\hat{\mu}^2 + \psi_1^{x_i}\hat{\mu} + \psi_0^{x_i}\right)\right)\left(z_i + \alpha\left(\psi_2^{z_i}\hat{\mu}^2 + \psi_1^{z_i}\hat{\mu} + \psi_0^{z_i}\right)\right) - 0.2\hat{\mu} \geq 0.$$

The minimization in this case is much harder computationally. Fixing $\alpha \in (0, 1]$, each inequation has up to two intervals as solution set, associated with the roots of the polynomial, and the overall solution set is obtained via interval calculation using up to $2n$ intervals, in a $\mathcal{O}(n)$ computation for each fixed α. The associated optimal $\hat{\mu}$ is then available as the smallest positive element of the overall solution set — so we have to limit the number of different α inspected to obtain the approximate minimization. The resulting algorithm for neighborhood $\mathcal{N}_\infty^-(0.2)$ was implemented and the computational results are shown in section 8.

7. Self-dual framework

We now present the extension to a self-dual framework that was actually used for the implementations, to account for the initial feasible point. In order to treat bounded variables we chose to explicitly separate bounded variables from non bounded, as this way the implementation is straightforward.

We define the primal problem

$$\begin{aligned} & \min\ c_1^T v + c_2^T x \\ & \text{s.t.} \\ \text{(LP)} \quad & A_1 v + A_2 x = b \\ & 0 \leq v \leq u \\ & 0 \leq x \end{aligned}$$

where

$$\begin{aligned} c_1, v, u &\in \mathbb{R}^k \\ c_2, x &\in \mathbb{R}^{n-k} \\ b &\in \mathbb{R}^m \end{aligned}$$

$$\begin{aligned} A_1 &\in \mathbb{R}^{m\text{x}k} \\ A_2 &\in \mathbb{R}^{m\text{x}(n-k)} \end{aligned}$$

and we define the corresponding dual problem

$$\begin{array}{cc} & \max\ b^T y - u^T w \\ & \text{s.t.} \\ \text{(LD)} & A_1^T y - w \leq c_1 \\ & A_2^T y \leq c_2 \\ & w \geq 0 \end{array}$$

where

$$\begin{aligned} y &\in \mathbb{R}^m \\ w &\in \mathbb{R}^k \end{aligned}$$

and we define

$$\begin{array}{cc} & \min\ c_0\theta \\ & \text{s.t.} \end{array}$$

$$\begin{array}{cccccccccc} & (1) & & & A_1 v & +A_2 x & -b\tau & -r_1\theta & = & 0 \\ & (2) & & & -v & & +u\tau & -r_2\theta & \geq & 0 \\ \text{(HLP)} & (3) & -A_1^T y & +w & & & +c_1\tau & -r_3\theta & \geq & 0 \\ & (4) & -A_2^T y & & & & +c_2\tau & -r_4\theta & \geq & 0 \\ & (5) & b^T y & -u^T w & -c_1^T v & -c_2^T x & & -r_5\theta & \geq & 0 \\ & (6) & r_1^T y & +r_2^T w & +r_3^T v & +r_4^T x & +r_5\tau & & = & -c_0 \\ & & & w, & v, & x, & \tau, & & \geq & 0 \end{array}$$

where we denote by s, h, z and κ the slack variable of inequations (2), (3), (4) and (5) respectively and

$$\begin{aligned} r_1 &= (A_1 v_0 + A_2 x_0 - b\tau_0)/\theta_0 \\ r_2 &= (u_0\tau_0 - v_0 - s_0)/\theta_0 \\ r_3 &= \left(c_1\tau_0 - A_1^T y_0 - h_0 + w_0\right)/\theta_0 \\ r_4 &= \left(c_2\tau_0 - A_2^T y_0 - z_0\right)/\theta_0 \\ r_5 &= \left(b^T y_0 - u_0^T w_0 - c_1^T v_0 - c_2^T x_0 - \kappa_0\right)/\theta_0 \\ c_0 &= \left(v_0^T h_0 + x_0^T z_0 + s_0^T w_0 + \tau_0\kappa_0\right)/\theta_0 \end{aligned}$$

where $v_0 > 0$, $x_0 > 0$, $s_0 > 0$, $z_0 > 0$, $w_0 > 0$ and y_0 are an arbitrary initial point and where $\tau_0 = \kappa_0 = \theta_0 = 1$, so that $(y_0, w_0, v_0, x_0, \tau_0, \theta_0, s_0, h_0, z_0, \kappa_0)$ satisfy (HLP). We also define $\mathcal{F}_h^0$ as the set of all points $(y_0, w_0, v_0, x_0, \tau_0, \theta_0, s_0, h_0, z_0, \kappa_0)$ satisfying (HLP) and such that $(w, v, x, \tau, s, h, z, \kappa) > 0$.

This way the (self-) central path for (HLP) is defined as

$$\mathcal{C} = \left\{ (y, w, v, x, \tau, \theta, s, h, z, \kappa) \in \mathcal{F}_h^0 : \begin{pmatrix} vh \\ xz \\ ws \\ \tau\kappa \end{pmatrix} = \mu e \right\}$$

and we define the euclidean neighborhood of the central path as

$$\mathcal{N}_2(\beta) = \left\{ (y, w, v, x, \tau, \theta, s, h, z, \kappa) \in \mathcal{F}_h^0 : \left\| \begin{pmatrix} vh \\ xz \\ ws \\ \tau\kappa \end{pmatrix} - \mu e \right\| \leq \beta\mu \right\}.$$

Extension to other neighborhoods is done in a way similar to what we did in the primal-dual context.

Using exactly the same arguments used in [22] but now in the context of bounded variables it can be proved that (HLP):

- is self-dual in the sense that it is a linear problem whose dual is identical to (HLP);
- has a feasible interior point $(y_0, w_0, v_0, x_0, \tau_0, \theta_0, s_0, h_0, z_0, \kappa_0)$ that lies in the central path for $\mu_0 = \dfrac{v_0^T h_0 + x_0^T z_0 + w_0^T s_0 + \tau_0\kappa_0}{n + k + 1}$;
- has an optimal solution and its solution set is bounded;
- has a strictly self-complementary solution, i.e., a solution

  $$(y^*, w^*, v^*, x^*, \tau^*, \theta^* = 0, s^*, h^*, z^*, \kappa^*)$$

  such that

  $$\begin{pmatrix} v^* + h^* \\ x^* + z^* \\ w^* + s^* \\ \tau^* + \kappa^* \end{pmatrix} > 0.$$

Furthermore,

- (LP) has a feasible and bounded solution if and only if $\tau^* > 0$. In this case, $(v^*/\tau^*, x^*/\tau^*)$ is an optimal solution for (LP) and $(y^*/\tau^*, w^*/\tau^*)$ is an optimal solution for (LD).
- If $\tau^* = 0$ then:
  a) if $c_1^T v^* + c_2^T x^* < 0$ then (LP) is infeasible;
  b) if $b^T y^* - u^T w^* < 0$ then (LD) is infeasible.

Cases a) and b) are not exclusive.

This all means that when we work in a self-dual framework we can always find a finite solution to (HLP) with which we can completely solve the original problem.

There is a very practical consequence of system (HLP) being self-dual. Analyzing the optimality condition for (HLP) we can see that it is almost identical to that of the original

(LP), in the sense that we have only to consider the primal form of the original problem, especially in all linear algebra routines.

As an immediate consequence, all results in the previous sections remain valid, but the first and second degree parameterizations have to be re-stated in this framework in order to allow for a proper implementation. Unfortunately, the formulas and the notation in this case become too heavy to be presented here — but they are straightforward.

It might seem that the system involved in (HLP) is too big. However, the system that provides the Newton direction for the associated the Karush_Kuhn-Tucker condition for (HLP) is

$$\begin{aligned}
A_1 d_v + A_2 d_x - b d_\tau - r_1 d_\theta &= 0 && (1)\\
-d_v + u d_\tau - r_2 d_\theta - d_s &= 0 && (2)\\
-A_1^T d_y + d_w + c_1 d_\tau - r_3 d_\theta - d_h &= 0 && (3)\\
-A_2^T d_y + c_2 d_\tau - r_4 d_\theta - d_z &= 0 && (4)\\
b^T d_y - u^T d_w - c_1^T d_v - c_2^T d_x - r_5 d_\theta - d_\kappa &= 0 && (5)\\
r_1^T d_y + r_2^T d_w + r_3^T d_v + r_4^T d_x + r_5 d_\tau &= 0 && (6)\\
S d_w + W d_s &= \mu e_1 - ws && (7)\\
H d_v + V d_h &= \mu e_2 - vh && (8)\\
Z d_x + X d_z &= \mu e_3 - xz && (9)\\
\kappa d_\tau + \tau d_\kappa &= \mu - \tau\kappa && (10)
\end{aligned}$$

and we can, with some algebra manipulation, show that this system can be solved using the solution of a smaller system similar to $AA^T x = \vec{b}$ where $A = [A_1 : A_2]$. Therefore the computational effort is of the same magnitude of standard primal-dual algorithms.

8. Computational results

As previously mentioned in the introduction, we were interested in being able to make fair comparisons between actual implementations. To do so, we took as basis Mehrotra's [12] predictor-corrector implementation PCx [2], for which the code in C is available. But we did some slight changes. First, Mehrotra's heuristics for the initial point does not generate a centered point, but by minimally tinkering with the parameters in his heuristic we generated initial points that are self-centered with respect to $\mathcal{N}_2$. Second, PCx allows for higher order Gondzio [7] corrections — we inhibited this feature in the configuration file but allowed the original feature of conjugated gradient refinement. We call this modified implementation PCx-r.

We then introduced into the original code, bypassing their main algorithm, the code for our algorithms, therefore inheriting the same routines for pre-processing (input reading, pre-conditioning via scaling, re-ordering of rows and columns, dense column handling and symbolic Cholesky factorization), starting point, all linear algebra (Ng e Peyton [14] routines for solving sparse linear systems as well as conjugated gradient refinement), termination criteria and output writing.

In the development of our work we made many implementations, for different choices of neighborhoods and parameter β, using strictly predictor steps or predictor-corrector steps, allowing or not the steplength α to be optimized, and using different strategies for approximately solving the nonlinear subproblem. The computational results of all these implementations are quite extensive and presenting them here would possible bring more

confusion then light in our understanding. We chose instead to pick three implementations that illustrate better some possible conclusions.

The implementations not presented here were excluded for the following reasons.

- Algorithms with strictly predictor steps were excluded. We noticed that this family of simpler implementations present, as is already known, iteration count and CPU times that are bigger then its predictor-corrector counterparts.
- Jansen neighborhoods didn't have a remarkable performance regarding iteration count and CPU times, as compared with euclidean neighborhoods.

All codes were compiled with the same compiling options and in the same computer. With this strategy, whatever difference in number of iterations and CPU times can only be attributed to the implementation of the algorithms themselves. Also, this way CPU times are not important in absolute terms, but only their relation with one another.

At each iteration of our algorithms we made a transformation that took the iterate in PCx *primal-dual* environment and transformed it into a *self-dual* iterate, in which we computed the next iterate and then transformed back to the primal-dual environment.

We made three implementations.

- .The first, which we call PCx-N2a, uses $\mathcal{N}_2(0.25)$ as neighborhood of the central path and fixes $\alpha = 1$, optimizing only in the variable μ. This way in step 3 of the algorithm the objective function and the restriction set become

$$\min_{\alpha,\mu} n\mu$$
$$\text{s.t.}$$
$$\sum_{i=1}^{n} (\hat{x}_i \hat{z}_i - \mu)^2 \leq \frac{9}{16}\mu^2$$
$$\text{or}$$
$$\sum_{i=1}^{n} \left(\left(x_i + \psi^x_{2,i}\mu^2 + \psi^x_{1,i}\mu + \psi^x_{0,i}\right) \left(z_i + \psi^z_{2,i}\mu^2 + \psi^z_{1,i}\mu + \psi^z_{0,i}\right) - \mu \right)^2 \leq \frac{9}{16}\mu^2$$

  using the notation for the primal-dual framework (in the self-dual framework, that was actually used, this notation is much more extensive).

- The second, PCx-N2b, allows both α and μ to be optimized, so that now the underlying optimization subproblem is

$$\min_{\alpha,\mu} \alpha n\mu + (1-\alpha)\, g$$
$$\text{s.t.}$$
$$\sum_{i=1}^{n} \left(\left(x_i + \alpha\left(\psi^x_{2,i}\mu^2 + \psi^x_{1,i}\mu + \psi^x_{0,i}\right)\right) \left(z_i + \alpha\left(\psi^z_{2,i}\mu^2 + \psi^z_{1,i}\mu + \psi^z_{0,i}\right)\right) - \mu \right)^2 \leq \frac{9}{16}\mu^2$$

- The third, PCx-SN uses $\mathcal{N}_\infty^-(0.2)$ and also fixes $\alpha = 1$. In this case the optimization is

$$\begin{gathered}\min_{\alpha,\mu} n\mu \\ \text{s.t.} \\ \left(x_i + \psi_{2,i}^x \mu^2 + \psi_{1,i}^x \mu + \psi_{0,i}^x\right)\left(z_i + \psi_{2,i}^z \mu^2 + \psi_{1,i}^z \mu + \psi_{0,i}^z\right) \leq 0.2\mu \\ \text{for } i = 1, ..., n\end{gathered}$$

  This optimization involves a set of n restrictions in μ; each restriction has a solution a set of up to two real intervals that μ must belong to, and so the global feasible set is the intersection of n sets of up to two intervals. In terms of computational effort, it is complex and expensive to generate this feasible set, since we have to find n times all the roots of a polynomial of degree four.

Note that convergence results were previously proved for all these algorithms.

Some further details: the problems we studied are a subset of the Netlib problems — medium size problems, bounded and unbounded, and all feasible. We chose not to study infeasible problems because we wanted to use the same original termination criteria of PCx. Actually, because of the self-dual framework, we could have used a different termination criteria that has been shown to be more efficient for infeasible problems [20].

The number of iterations and CPU times in seconds for each implementation are shown in the table below. Column L indicates the number of non-zero elements in the Cholesky factor L. All tests were performed using a Pentium 3 processor at 500 MHz running under Windows NT, using Borland 5.0 C compiler.

				PCx-r		PCx-SN		PCx-N2a		PCx-N2b	
Problem	Rows	Cols	L	iter.	CPU	iter.	CPU	iter.	CPU	iter.	CPU
25FV47	821	1571	33809	30	1.83	24	3.19	44	4.10	44	4.22
BANDM	305	472	3936	19	0.14	16	0.33	30	0.42	30	0.49
BNL2	2324	3489	81275	40	8.94	34	13.32	67	19.19	67	19.33
BOEING1	351	384	5725	21	0.25	29	1.11	54	1.26	54	1.39
BOEING2	166	143	1912	18	0.03	18	0.24	34	0.26	34	0.32
BORE3D	233	315	1034	18	0.03	15	0.11	28	0.09	28	0.17
CAPRI	271	353	3962	24	0.17	20	2.65	42	0.56	42	0.64
CYCLE	1903	2857	56102	45	4.19	22	16.72	51	8.47	51	8.6
CZPROB	929	3523	3520	32	0.58	21	13.77	41	2.18	41	2.27
FIT2P	3000	13525	3000	21	4.57	22	125.72	59	47.31	59	47.61
FFFFF800	524	854	9573	33	0.55	26	5.95	51	1.56	51	1.69
FORPLAN	161	421	3304	28	0.25	23	3.06	47	0.76	47	0.89
PILOT	1441	3652	200812	43	32.56	62	130.16	135	122.88	135	123.24
PILOT.WE	722	2789	15605	45	1.70	38	30.30	79	6.77	79	7.00

As mentioned before, we achieved less iteration count in most of the problems studied, although computational times were bigger. There are two reasons for this. First, at each iteration we have to solve seven linear systems using the same Cholesky factor in order to calculate the predictor-corrector parameterization, as opposed to two system solves in the usual predictor-corrector algorithm. Second, in this first version of our implementation there are still some important parts of the code that need to be re-written for better speed,

such as the interval calculations routines and the routines that find roots of polynomials. This is corroborated by the difference in CPU times between PCx_N2a and PCx-SN — the former does not use interval calculation and has smaller CPU times although having larger iteration count.

It is interesting to note that PCx-N2a and PCx-N2b achieved exactly the same number of iterations — although optimization was also allowed for the steplength α, the optimum was always found at $\alpha = 1$. We believe that this happens because we are dealing with feasible iterates algorithms — for instance for infeasible iterates algorithms using potential reduction functions this does not happen.

9. Conclusions

As we had proposed in section in section 1, by parameterizing the next iterate in terms of α and μ we established an analytical environment where the choices of these parameters can be stated as a clear optimization subproblem — which precludes the use of heuristics for such choices. Although Mehrotra's heuristics are very good — to the point that practical implementations merely substitute them for similar ones — yet the need exists to replace them for more analytical tools.

More then simply introducing an analytical resource, we showed that the type of parameterization we made is useful as a means to evaluate and improve the other parts a given algorithm, since these parts can also be expressed in terms of the parameterization. This had not explicitly been done before and this might be the main contribution of our work.

We can conclude that the use of a self-dual framework — although allowing for an elegant treatment of the issues of initial point and infeasibility detection — presents the great disadvantage of requiring a greater number of system solves at each iteration, and the price we pay in practical implementations is so big as to become prohibitive. By itself, the self-dual framework can not reduce the number of iteration count to the point of compensating the extra system solves. This is particularly so for Netlib problems, where the CPU times for computing each Cholesky factor is not remarkably greater than each system solve, because of sparsity.

We also conclude that the optimization involving large neighborhoods should be performed with extreme care, considering its greater computational complexity — but research in this direction deserves further investigation since the reduction in iteration count is significant.

There are two possible future developments using the idea of parameterizing the next iterate the way we did.

One is to force the parameterization for the predictor-corrector algorithm to use $\mu = 0$ in the predictor step and then calculate μ only at the corrector step. This way the parameterization becomes of first order in μ — therefore computationally cheaper as it requires less system solves — and is more similar to the one used in Mehrotra's algorithm.

Another one is to work out of the self-dual framework, directly using an infeasible-iterates algorithm, in order so save system solves. In this framework it would also be possible to use potential reduction algorithms instead of the central path.

References

[1] D.A. Bayer and J.C. Lagarias. The nonlinear geometry of linear programming. I: Affine and projective scaling trajectories. II: Legendre transform coordinates and

central trajectories. *Transactions of the American Mathematical Society*, 314:499–581, 1989.

[2] J. Czyzyc, S. Mehrotra, M. Wagner, and S.J. Wright. *PCx User Guide (Version 1.1)*. Optimization Technology Center, 1997.

[3] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, 1968. Reprint : Volume 4 of *SIAM Classics in Applied Mathematics*, SIAM Publications, Philadelphia, PA 19104–2688, USA, 1990.

[4] K. R. Frisch. The logarithmic potential method for convex programming. Unpublished manuscript, Institute of Economics, University of Oslo, Oslo, Norway, May 1955.

[5] P.E. Gill, W. Murray, M.A. Saunders, J.A. Tomlin, and M.H. Wright. On projected Newton barrier methods for linear programming and an equivalence to Karmakar's projective method. *Mathematical Programming*, 36:183–209, 1986.

[6] A.J. Goldman and A.W. Tucker. Polyedral convex cones. In H.W. Kuhn and A.W. Tucker, editors, *Linear Inequalities and Related Systems*, pages 19–40. Princeton University Press, Princeton, NJ, 1956.

[7] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6:137–156, 1996.

[8] P. Hung and Y. Ye. An asymptotical $\mathcal{O}(\sqrt{n}L)$-iteration path-following linear programming algorithm that uses wide neighbourhoods. *SIAM Journal of Optimization*, 6:570–586, 1996.

[9] B. Jansen, C. Roos, T. Terlaky, and J.-Ph. Vial. Primal-dual algorithms for linear programming based on the logarithmic barrier methods. *Journal of Optimization Theory and Applications*, 83:1–26, 1994.

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

[11] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming : Interior Point and Related Methods*, pages 131–158. Springer Verlag, New York, 1989. Identical version in : *Proceedings of the 6th Mathematical Programming Symposium of Japan, Nagoya, Japan*, pages 1–35, 1986.

[12] S. Mehrotra. On the implementation of a primal–dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[13] R.D.C. Monteiro and I. Adler. Interior path following primal–dual algorithms : Part I : Linear programming. *Mathematical Programming*, 44:27–41, 1989.

[14] E. Ng and B.W. Peyton. Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM Journal on Scientific Conputing*, 14:1034–1056, 1993.

[15] J. Peng, C. Roos, and T. Terlaky. New complexity analysis of the primal-dual newton method for linear optimization. Technical Report 98-05, Faculty of Technical Mathematics and Informatics, Delft University of Technology, 1998. To appear in *Annals of Operation Research*.

[16] J. Peng, C. Roos, and T. Terlaky. A new class of polynomial primal-dual methods for linear and semidefinite optimization. Technical report, Faculty of Information Technology and Systems, Delft University of Technology, December 1999.

[17] G. Sonnevend. An "analytic center" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. In A. Prekopa, J. Szelezsan, and B. Strazicky, editors, *System Modelling and Optimization : Proceedings of the 12th IFIP–Conference held in Budapest, Hungary, September 1985*, volume 84 of *Lecture Notes in Control and Information Sciences*, pages 866–876. Springer Verlag, Berlin, West–Germany, 1986.

[18] A.W. Tucker. Dual systems of homogeneous linear relations. In H.W. Kuhn and A.W. Tucker, editors, *Linear Inequalities and Related Systems*, pages 3–18. Princeton University Press, Princeton, NJ, 1956.

[19] X. Xu. An $\mathcal{O}(\sqrt{n}L)$-iteration large-step infeasible path-following algorithm for linear programming. Technical report, Department of Management Sciences, The University of Iowa, Iowa City, Iowa 52242, USA, 1994.

[20] X. Xu, P. Hung, and Y. Ye. A simplification of the homogeneous and self-dual linear programming algorithm and its implementation. *Annals of Operations Research*, 62:151–172, 1996.

[21] X. Xu and Y. Ye. A generalized homogeneous and self-dual algorithm for linear programming. *Operations Research Letters*, 17(94-3), 1995.

[22] Y. Ye, M.J. Todd, and S. Mizuno. An $\mathcal{O}(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm. *Mathematics of Operations Research*, 19:53–67, 1994.

[23] Yinyu Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley and Sons, New York, 1997.