Further Development of Multiple Centrality Correctors for Interior Point Methods

Marco Colombo* Jacek Gondzio†

School of Mathematics
The University of Edinburgh
Mayfield Road, Edinburgh EH9 3JZ
United Kingdom

MS-2005-001

18 October 2005, revised 9 March 2006, 7 September 2006

*Email: M.Colombo@ed.ac.uk
†Email: J.Gondzio@ed.ac.uk, URL: http://maths.ed.ac.uk/~gondzio/

Further Development of Multiple Centrality Correctors for Interior Point Methods

Abstract

This paper addresses the role of centrality in the implementation of interior point methods. We provide theoretical arguments to justify the use of a symmetric neighbourhood, and translate them into computational practice leading to a new insight into the role of recentering in the implementation of interior point methods. Second-order correctors, such as Mehrotra's predictor–corrector, can occasionally fail: we derive a remedy to such difficulties from a new interpretation of multiple centrality correctors. Through extensive numerical experience we show that the proposed centrality correcting scheme leads to noteworthy savings over second-order predictor–corrector technique and previous implementations of multiple centrality correctors.

1 Introduction

Interior point methods (IPMs for short) are well-suited to solving very large scale optimization problems. Their theory is well understood [19] and the techniques used in their implementation are documented in extensive literature (see, for example, [1] and the references therein). Interior point methods require the computation of the Newton direction for the associated barrier problem and make a step along this direction, thus usually reducing primal and dual infeasibilities and complementarity gap; eventually, after a number of iterations, they reach optimality. Since finding the Newton direction is usually a major computational task, a large effort in the theory and practice of IPMs concentrates on reducing the number of Newton steps.

Theoretical developments aim at lowering the upper bound on the number of needed steps. The results provided by such worst-case complexity analysis are informative but exceedingly pessimistic. A common complexity result states that interior point methods (for linear and quadratic programming) converge arbitrarily close to an optimal solution in a number of iterations which is proportional to the problem dimension or to the square root of it. In practice, convergence is much faster: optimality is reached in a number of iterations which is proportional to the logarithm of the problem dimension. Practical developments aim to reduce this number even further. Two techniques have proved particularly successful in this respect: Mehrotra's predictor–corrector algorithm [12] and multiple centrality correctors [8]. These techniques have been implemented in most of commercial and academic interior point solvers for linear and quadratic programming such as BPMPD, Cplex, HOPDM, Mosek, OOPS, OOQP, PCx and Xpress. They have also been used with success in semidefinite programming with IPMs [9].

Both correcting techniques originate from the observation that (when direct methods of linear algebra are used) the computation of the Newton direction requires factoring a sparse symmetric matrix, followed by a backsolve which uses this factorization. The cost of computing the factors is usually significantly larger than that of backsolving: in some cases the ratio between these two computational efforts may even exceed 1000. Consequently, it is worth adding more (cheap) backsolves if this reduces the number of (expensive) factorizations. Mehrotra's predictor–corrector technique [12] uses two backsolves per factorization; the multiple centrality correctors technique [8] allows recursive corrections: a larger number of backsolves per iteration is possible, leading to a further reduction in the number of factorizations. Since these two methods

were developed, there have been a number of attempts to investigate their behaviour rigorously and thus understand them better. Such objectives are difficult to achieve because correctors use heuristics which are successful in practice but hard to analyse theoretically. Besides, both correcting techniques are applied to long-step and infeasible algorithms which have very little in common with the short-step and feasible algorithms that display the best known theoretical complexity. Nevertheless, we would like to mention several of such theoretical attempts as they shed light on some issues which are important in efficient implementations of IPMs.

Mizuno, Todd and Ye [14] analysed the short-step predictor–corrector method. Their algorithm uses two nested neighbourhoods $N_2(\theta^2)$ and $N_2(\theta)$, $\theta \in (0,1)$, and exploits the quadratic convergence property of Newton's method in this type of neighbourhood. The predictor direction gains optimality, possibly at the expense of worsening the centrality, keeping the iterate in a larger neighbourhood $N_2(\theta)$ of the central path. It is then followed with a pure re-centering step which throws the iterate back into a tighter $N_2(\theta^2)$ neighbourhood. Hence, every second step the algorithm produces a point in $N_2(\theta^2)$. This is a clever approach, but the use of the very restrictive N_2 neighbourhood makes it unattractive for practical applications.

Jarre and Wechs [10] took a more pragmatic view and looked for an implementable technique for generating efficient higher-order search directions in a primal–dual interior-point framework. In the Newton system, while it is clear what to consider as right-hand side for primal and dual feasibility constraints (the residual at the current point), the complementarity component leaves more freedom in choosing a target t in the right-hand side. They argue that there exists an optimal choice for which the corresponding Newton system would produce immediately the optimizer; however, it is not obvious how to find it. Therefore, they propose to search a subspace spanned by k different directions $\Delta w_1, \Delta w_2, \ldots, \Delta w_k$ generated from some affinely independent targets $t_1, t_2, \ldots, t_k$. As the quality of a search direction can be measured by the length of the stepsize and the reduction in complementarity gap, they aim to find the combination

$$\Delta w = \Delta w(\rho) = \rho_1 \Delta w_1 + \rho_2 \Delta w_2 + \ldots + \rho_k \Delta w_k$$

that maximizes these measures. This can be formulated as a small linear subproblem which can be solved approximately to produce a search direction Δw that is generally better than Mehrotra's predictor–corrector direction.

Tapia et al. [17] interpreted the Newton step produced by Mehrotra's predictor–corrector algorithm as a perturbed composite Newton method and gave results on the order of convergence. They proved that a level-1 composite Newton method, when applied to the perturbed Karush-Kuhn-Tucker system, produces the same sequence of iterates as Mehrotra's predictor–corrector algorithm. While, in general, a level-m composite Newton method has a Q-convergence rate of $m+2$ [15], the same result does not hold if the stepsize has to be damped to keep non-negativity of the iterates, as is necessary in an interior-point setting. However, under the additional assumptions of strict complementarity and nondegeneracy of the solution and feasibility of the starting point, Mehrotra's predictor–corrector method can be shown to have Q-cubic convergence [17].

Mehrotra's predictor–corrector as it is implemented in optimization solvers [11, 12] is a very aggressive technique. In the vast majority of cases this approach yields excellent results. However, practitioners noticed that this technique may sometimes behave erratically, especially when used for a predictor direction applied from highly infeasible and not well-centered points. This observation was one of the arguments that led to the development of multiple centrality correctors [8]. These correctors are less ambitious: instead of attempting to correct for the whole second-order

error, they concentrate on improving the complementarity pairs which really seem to hinder the progress of the algorithm.

In a recent study, Cartis [3] provided a readable example that Mehrotra's technique may produce a corrector which is larger in magnitude than the predictor and points in the wrong direction, possibly causing the algorithm to fail. We realised that such a failure is less likely to happen when multiple centrality correctors are used because corrector terms are generally smaller in magnitude. This motivated us to revisit this technique and led to a number of changes in the way centrality is treated in the interior point algorithm implemented in the HOPDM solver [7] for linear and quadratic programming.

This paper is organised as follows. In Section 2 we recall the key features of Mehrotra's predictor–corrector algorithm [12], multiple centrality correctors [8], and the recently proposed Krylov subspace searches [13]. In Section 3 we introduce the symmetric neighbourhood and analyse a variant of the feasible long-step path following algorithm based on it. Our analysis, which follows very closely that of Chapter 5 in [19], reveals the surprising property that the presence of the upper bound on the size of complementarity products does not seem to affect the complexity result. In Section 4 we present an algorithm which uses the symmetric neighbourhood and new strategies for computing centrality correctors. In Section 5 we illustrate the performance of the proposed algorithm by applying it to a wide class of linear and quadratic problems reaching tens and hundreds of thousand of variables. Finally, in Section 6 we give our conclusions.

2 Primal–dual methods for linear programming

Consider the following primal–dual pair of linear programming problems in standard form

$$\begin{array}{lll} \text{Primal} & & \\ \min & c^T x & \\ \text{s.t.} & Ax = b, & \\ & x \geq 0; & \end{array} \qquad \begin{array}{lll} \text{Dual} & & \\ \max & b^T y & \\ \text{s.t.} & A^T y + s = c, & \\ & y \text{ free}, \;\; s \geq 0, & \end{array}$$

where $A \in \mathcal{R}^{m \times n}$, $x, s, c \in \mathcal{R}^n$ and $y, b \in \mathcal{R}^m$. The first-order optimality conditions (Karush-Kuhn-Tucker conditions) are

$$\begin{array}{rcl} Ax & = & b \\ A^T y + s & = & c \\ XSe & = & 0 \\ (x, s) & \geq & 0, \end{array} \tag{1}$$

where X and S are diagonal matrices with elements x_i and s_i respectively, and e is a vector of ones. In other words, a solution is characterised by primal feasibility, dual feasibility and complementarity. Path-following interior point methods [19] perturb the above conditions by asking the complementarity pairs to align to a specific barrier parameter μ,

$$XSe = \mu e,$$

while enforcing $(x, s) > 0$. As μ is decreased iteration after iteration, the solution of the perturbed Karush-Kuhn-Tucker conditions traces a unique path toward the optimal set. Path-following

interior point methods seek a solution to the nonlinear system of equations

$$F(x,y,s) = \begin{bmatrix} Ax - b \\ A^T y + s - c \\ XSe - \mu e \end{bmatrix} = 0,$$

where the nonlinearity derives from the complementarity conditions. We use Newton's method to linearise the system according to $\nabla F(x,y,s)\Delta(x,y,s) = -F(x,y,s)$, and obtain the so-called step equations

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = r \tag{2}$$

with

$$r = \begin{bmatrix} b - Ax \\ c - A^T y - s \\ -XSe + \mu e \end{bmatrix}, \tag{3}$$

which need to be solved with a specified μ for a search direction $(\Delta x, \Delta y, \Delta s)$.

2.1 Mehrotra's predictor–corrector technique

A number of advantages can be obtained by splitting the computation of the Newton direction into two steps, corresponding to solving the linear system (2) independently for the two right-hand sides

$$r_1 = \begin{bmatrix} b - Ax \\ c - A^T y - s \\ -XSe \end{bmatrix} \quad \text{and} \quad r_2 = \begin{bmatrix} 0 \\ 0 \\ \mu e \end{bmatrix}. \tag{4}$$

First, we can postpone the choice of μ and base it on the assessment of the quality of the affine-scaling direction; second, the error made by the affine-scaling direction may be taken into account and corrected. Mehrotra's predictor–corrector technique [12] translates these observations into a powerful computational method. We recall its key features.

The affine-scaling *predictor* direction $\Delta_a = (\Delta_a x, \Delta_a y, \Delta_a s)$ is obtained by solving system (2) with right-hand side r_1 defined above. This direction is used to evaluate a predicted complementarity gap after maximum feasible step

$$g_a = (x + \alpha_P \Delta_a x)^T (s + \alpha_D \Delta_a s).$$

The ratio $g_a / x^T s$ measures the quality of the predictor. If it is close to one then very little progress is achievable in direction Δ_a and a strong centering component should be used. Otherwise, if the ratio is small then less centering is needed and a more aggressive optimization is possible. In [12] the following choice of the new barrier parameter is suggested

$$\mu = \left(\frac{g_a}{x^T s}\right)^2 \frac{g_a}{n} = \left(\frac{g_a}{x^T s}\right)^3 \frac{x^T s}{n}, \tag{5}$$

corresponding to the choice of $\sigma = (g_a / x^T s)^3$ for the centering parameter.

If a full step in the affine-scaling direction is made, then the new complementarity products are equal to

$$(X + \Delta_a X)(S + \Delta_a S)e \;=\; XSe + (S\Delta_a x + X\Delta_a s) + \Delta_a X \Delta_a S e \;=\; \Delta_a X \Delta_a S e,$$

as the third equation in the Newton system satisfies $S\Delta_a x + X\Delta_a s = -XSe$. The term $\Delta_a X \Delta_a S e$ corresponds to the error introduced by Newton's method in linearising the perturbed complementarity condition. Ideally, we would like the new complementarity products to be equal to μe. Mehrotra's second-order corrector is obtained by solving the system (2) with right-hand side

$$r = \begin{bmatrix} 0 \\ 0 \\ \mu e - \Delta_a X \Delta_a S e \end{bmatrix} \tag{6}$$

for the direction Δ_c. Such corrector direction does not only add the centrality term but corrects for the error made by the predictor as well. Once the predictor and corrector terms are computed they are added to produce the final direction

$$\Delta = \Delta_a + \Delta_c.$$

The cost of a single iteration in the predictor–corrector method is only slightly larger than that of the standard method because two backsolves per iteration have to be executed, one for the predictor and one for the corrector. The use of the predictor–corrector technique leads to significant savings in the number of IPM iterations and, for all non-trivial problems, translate into significant CPU time savings [11, 12]. Indeed, Mehrotra's predictor–corrector technique is advantageous in all interior point implementations which use direct solvers to compute the Newton direction.

2.2 Multiple centrality correctors

Mehrotra's predictor–corrector technique is based on the optimistic assumption that a *full* step in the corrected direction will be possible. Moreover, an attempt to correct all complementarity products to the same value μ is also very demanding and occasionally too aggressive. Finally, Mehrotra's corrector does not provide CPU time savings when used recursively [2]. The multiple centrality correctors technique [8] removes these drawbacks.

Assume that a predictor direction Δ_p is given and the corresponding feasible stepsizes α_P and α_D in the primal and dual spaces are determined. We look for a centrality corrector Δ_m such that larger steps will be made in the composite direction $\Delta = \Delta_p + \Delta_m$. We want to enlarge the stepsizes to

$$\tilde{\alpha}_P = \min(\alpha_P + \delta,\, 1) \quad \text{and} \quad \tilde{\alpha}_D = \min(\alpha_D + \delta,\, 1),$$

for some aspiration level $\delta \in (0, 1)$. We compute a trial point

$$\tilde{x} = x + \tilde{\alpha}_P \Delta_p x, \quad \tilde{s} = s + \tilde{\alpha}_D \Delta_p s,$$

and the corresponding complementarity products $\tilde{v} = \tilde{X}\tilde{S}e \in \mathcal{R}^n$.

The complementarity products $\tilde{v}$ are very unlikely to align to the same value μ. Some of them are significantly smaller than μ, including cases of negative components in $\tilde{v}$, and some exceed

μ. Instead of trying to correct them all to the value of μ, we correct only the *outliers*. Namely, we try to move small products ($\tilde{x}_j\tilde{s}_j \leq \gamma\mu$) to $\gamma\mu$ and move large products ($\tilde{x}_j\tilde{s}_j \geq \gamma^{-1}\mu$) to $\gamma^{-1}\mu$, where $\gamma \in (0,1)$; complementarity products which satisfy $\gamma\mu \leq x_j s_j \leq \gamma^{-1}\mu$ are already reasonably close to their target values, and do not need to be changed.

Therefore, the corrector term Δ_m is computed by solving the usual system of equations (2) for a special right-hand side $(0,\ 0,\ t)^T$, where the target t is defined as follows:

$$t_j = \begin{cases} \gamma\mu - \tilde{x}_j\tilde{s}_j & \text{if} \quad \tilde{x}_j\tilde{s}_j \leq \gamma\mu \\ \gamma^{-1}\mu - \tilde{x}_j\tilde{s}_j & \text{if} \quad \tilde{x}_j\tilde{s}_j \geq \gamma^{-1}\mu \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

The computational experience presented in [8] confirmed that this strategy is effective and the stepsizes in the primal and dual spaces computed for the composite direction are larger than those corresponding to the predictor direction. Moreover, this technique can be applied recursively on the direction $\Delta_p := \Delta_p + \Delta_m$. Indeed, we use it as long as the stepsizes increase at least by a fraction of the aspiration level δ, up to a maximum number of times determined at the beginning of the solution of the problem according to the ratio between factorization cost and backsolve cost, as detailed in Section 3.3 of [8].

2.3 Krylov subspace searches

While the approach presented above generates a series of correctors that are evaluated and applied recursively, Mehrotra and Li [13] propose a scheme in which a collection of linearly independent directions is combined through a small linear subproblem.

Following the approach explored by Jarre and Wechs [10], they express the requirements for a good search direction as a linear program. In particular, they impose conditions aimed at ensuring global convergence of the algorithm when using generic search directions. The directions considered in the subspace search can include all sorts of linearly independent directions: affine-scaling direction, Mehrotra's corrector, multiple centrality correctors, Jarre–Wechs directions. In the recently proposed approach, Mehrotra and Li [13] generate directions using a Krylov subspace mechanism.

At the k-th iteration of interior point method we have to solve the Newton system $H_k\Delta_k = \xi_k$, where

$$\xi_k = \begin{bmatrix} b - Ax^k \\ c - A^T y^k - s^k \\ \mu e - X^k S^k e \end{bmatrix}$$

is the right-hand side evaluated at the current iterate and H_k is the corresponding Jacobian matrix. The direction Δ_k is used to compute a trial point:

$$\tilde{x} = x^k + \alpha_P \Delta_k x, \quad \tilde{y} = y^k + \alpha_D \Delta_k y, \quad \tilde{s} = s^k + \alpha_D \Delta_k s.$$

At the trial point $(\tilde{x}, \tilde{y}, \tilde{s})$, a usual interior point method would have to solve the system $\tilde{H}\tilde{\Delta} = \tilde{\xi}$ in order to find the next search direction. Instead, Mehrotra and Li [13] generate a Krylov subspace for $\tilde{H}\tilde{\Delta} = \tilde{\xi}$. The Krylov subspace of dimension j is defined as

$$K_j(H_k, \tilde{H},\ \tilde{\xi}) = \text{span}\{\xi_H, G\xi_H, G^2\xi_H, \ldots, G^{j-1}\xi_H\},$$

where $\xi_H = H_k^{-1}\tilde{\xi}$, and $G = I - H_k^{-1}\tilde{H}$. Note that for stability reasons $\tilde{H}$ is preconditioned with H_k, the factors of which have already been computed. The subspace thus generated contains j linearly independent directions.

In the algorithm of [13], the affine-scaling direction Δ_a, Mehrotra's corrector Δ_0, the first j directions $\Delta_1, \Delta_2, \ldots, \Delta_j$ from $K_j(H_k, \tilde{H}, \tilde{\xi})$ and, but only under some circumstances, a pure recentering direction Δ_{cen} are combined with appropriate weights ρ:

$$\Delta(\rho) = \rho_a \Delta_a + \sum_{i=0}^{j} \rho_i \Delta_i + \rho_{cen} \Delta_{cen}.$$

The choice of the best set of weights ρ in the combined search direction is obtained by solving an auxiliary linear programming subproblem. The subproblem maximizes the rate of decrease in duality gap whilst satisfying a series of requirements: non-negativity of the new iterate, upper bounds on the magnitude of the search direction, upper bounds on infeasibilities, decrease in the average complementarity gap, and closeness to the central path.

3 Symmetric neighbourhood

Practical experience with the primal–dual algorithm in HOPDM [7] suggests that one of the features responsible for its efficiency is the way in which the quality of centrality is assessed. By "centrality" we understand here the spread of complementarity products $x_i s_i$, $i = 1, \ldots, n$. Large discrepancies within the complementarity pairs, and therefore bad centering, create problems for the search directions: an unsuccessful iteration is caused not only by small complementarity products, but also by very large ones. This can be explained by the fact that Newton's direction tries to compensate for very large products, as they provide the largest gain in complementarity gap when a full step is taken. However, the direction thus generated may not properly consider the presence of very small products, which then become blocking components when the stepsizes are computed.

The notion of spread in complementarity products is not well characterised by either of the two neighbourhoods N_2 or $N_{-\infty}$ commonly used in theoretical developments of IPMs. To overcome this disadvantage, here we formalise a variation on the usual $\mathcal{N}_{-\infty}(\gamma)$ neighbourhood, in which we introduce an upper bound on the complementary pairs. This neighbourhood was implicitly used in the previous section to define an achievable target for multiple centrality correctors. We define the symmetric neighbourhood to be the set

$$\mathcal{N}_s(\gamma) = \{(x, y, s) \in \mathcal{F}^0 : \gamma\mu \leq x_i s_i \leq \frac{1}{\gamma}\mu,\ i = 1, \ldots, n\},$$

where $\mathcal{F}^0 = \{(x, y, s) : Ax = b, A^T y + s = c, (x, s) > 0\}$ is the set of strictly feasible primal–dual points, $\mu = x^T s/n$, and $\gamma \in (0, 1)$.

While the $\mathcal{N}_{-\infty}$ neighbourhood ensures that some products do not approach zero too early, it does not prevent products from becoming too large with respect to the average. In other words, it does not provide a complete picture of the centrality of the iterate. The symmetric neighbourhood $\mathcal{N}_s$, on the other hand, promotes the decrease of complementarity pairs which are too large, thus taking better care of centrality.

The analysis is done for the long-step feasible path-following algorithm and follows closely the presentation of [19, Chapter 5]. In this context, the search direction $(\Delta x, \Delta y, \Delta s)$ is found by solving system (2) with $r = (0,\ 0, -XSe + \sigma\mu e)^T$, $\sigma \in (0,1)$, $\mu = x^T s/n$.

First we need a technical result, the proof of which can be found in [19, Lemma 5.10] and is unchanged by the use of $\mathcal{N}_s$ rather than $\mathcal{N}_{-\infty}$.

Lemma 1 *If* $(x,y,s) \in \mathcal{N}_s(\gamma)$, *then* $\|\Delta X \Delta S e\| \leq 2^{-3/2}\left(1+\frac{1}{\gamma}\right) n\mu$.

Our main result is presented in Theorem 2. We prove that it is possible to find a strictly positive stepsize α such that the new iterate $(x(\alpha), y(\alpha), s(\alpha)) = (x,y,s) + \alpha(\Delta x, \Delta y, \Delta s)$ will not leave the symmetric neighbourhood, and thus this neighbourhood is well defined. This result extends Theorem 5.11 in [19].

Theorem 2 *If* $(x,y,s) \in \mathcal{N}_s(\gamma)$, *then* $(x(\alpha), y(\alpha), s(\alpha)) \in \mathcal{N}_s(\gamma)$ *for all*

$$\alpha \in \left[0, 2^{3/2}\gamma \frac{1-\gamma}{1+\gamma}\frac{\sigma}{n}\right].$$

Proof: Let us express the complementarity product in terms of the stepsize α along the direction $(\Delta x, \Delta y, \Delta s)$:

$$\begin{aligned} x_i(\alpha)s_i(\alpha) &= (x_i + \alpha\Delta x_i)(s_i + \alpha\Delta s_i) \\ &= x_i s_i + \alpha(x_i\Delta s_i + s_i\Delta x_i) + \alpha^2 \Delta x_i \Delta s_i \\ &= (1-\alpha)x_i s_i + \alpha\sigma\mu + \alpha^2\Delta x_i\Delta s_i. \end{aligned} \tag{8}$$

We need to study what happens to this complementarity product with respect to both bounds of the symmetric neighbourhood. Let us first consider the bound $x_i s_i \leq \frac{1}{\gamma}\mu$. By Lemma 1, equation (8) implies

$$x_i(\alpha)s_i(\alpha) \leq (1-\alpha)\frac{1}{\gamma}\mu + \alpha\sigma\mu + \alpha^2 2^{-3/2}\left(1+\frac{1}{\gamma}\right)n\mu.$$

At the new point $(x(\alpha), y(\alpha), s(\alpha))$, the duality gap is $x(\alpha)^T s(\alpha) = n\mu(\alpha) = n(1-\alpha+\alpha\sigma)\mu$, as can be obtained by summing up both sides of equation (8) and remembering that $\sum_i \Delta x_i \Delta s_i = 0$ in a feasible algorithm. The relation $x_i(\alpha)s_i(\alpha) \leq \frac{1}{\gamma}\mu(\alpha)$ holds provided that

$$(1-\alpha)\frac{1}{\gamma}\mu + \alpha\sigma\mu + \alpha^2 2^{-3/2}\left(1+\frac{1}{\gamma}\right)n\mu \leq \frac{1}{\gamma}(1-\alpha+\alpha\sigma)\mu,$$

from which we derive a first bound on the stepsize:

$$\alpha \leq 2^{3/2}\frac{1-\gamma}{1+\gamma}\frac{\sigma}{n} = \bar{\alpha}_1.$$

Considering now the bound $x_i s_i \geq \gamma\mu$ and proceeding as before, we derive a second bound on the stepsize:

$$\alpha \leq 2^{3/2}\gamma\frac{1-\gamma}{1+\gamma}\frac{\sigma}{n} = \bar{\alpha}_2.$$

Therefore, we satisfy both bounds and guarantee that $(x(\alpha), y(\alpha), s(\alpha)) \in \mathcal{N}_s(\gamma)$ if

$$\alpha \in [0, \min(\bar{\alpha}_1, \bar{\alpha}_2)],$$

which proves the claim, as $\gamma \in (0, 1)$. ■

It is interesting to note that the introduction of the upper bound on the complementarity pairs does not change the polynomial complexity result proved for the $\mathcal{N}_{-\infty}(\gamma)$ neighbourhood (Theorem 5.12 in [19]). Therefore, the symmetric neighbourhood provides a better practical environment without any theoretical loss. This understanding provides some additional insight into the desired characteristics of a well-behaved iterate. With this theoretical guidance, in the next section we will proceed to discuss the practical implications derived from here.

4 Weighted correctors

Newton's method applied to the primal–dual path-following algorithm provides a first-order approximation of the central path, in which the nonlinear KKT system corresponding to the barrier problem is linearised around the current point (x^k, y^k, s^k). Consistently with the standard analysis of Newton's method, this linear approximation is valid locally, in a small neighbourhood of the point where it is computed. Depending on the specific characteristics of the point, such an approximation may not be a good direction at all if used outside this neighbourhood.

Mehrotra's algorithm adds a second-order correction to the search direction in order to construct a quadratic approximation of the central path. This technique works extremely well, and the practical superiority of a second-order algorithm over a first-order one is broadly recognised [8, 11, 12]. However, the central path is a highly nonlinear curve that, according to Vavasis and Ye [18], is composed by $O(n^2)$ turns of a high degree and segments in which it is approximately straight. Given the complexity of this curve, it is unrealistic to be able to approximate it everywhere with a second-order curve.

Failures of Mehrotra's corrector have been known by practitioners since its introduction. In practical implementations, it was noticed that Mehrotra's corrector would sporadically produce a stepsize shorter than the one obtained in the predictor direction. In such situations, it is common to reject the corrector, then try to use some multiple centrality correctors or move along the predictor direction alone. This issue has recently been analysed by Cartis [3], who provided an example in which the second-order corrector does not behave well. Cartis' analysis is based on an algorithm that combines a standard primal–dual path-following method with a second-order correction. Despite not being exactly Mehrotra's predictor–corrector algorithm, both are very close in spirit. The example shows that for certain starting points the corrector is always orders of magnitude larger than the predictor, in both primal and dual spaces. Whilst the predictor points towards the optimum, the second-order corrector points away from it. As the final direction is given by

$$\Delta = \Delta_a + \Delta_c,$$

the combined direction is influenced almost exclusively by the corrector, hence it is not accurate. The solution outlined by Cartis in [3], and then further developed in [4], is to reduce the influence exerted by the corrector by weighting it by the square of the stepsize. In a similar way, Salahi

et al. [16] propose to find the corrector by weighting the term $\Delta_a X \Delta_a S e$ in (6) by the allowed stepsize for the affine-scaling direction.

The theoretical findings outlined above give rise to the following generalisation

$$\Delta = \Delta_a + \omega \Delta_c,$$

where we weight the corrector by a parameter $\omega \in (0, 1]$ independent of α.

In our implementation the weight is chosen independently at each iteration such that the stepsize in the composite direction is maximized. This gives us the freedom to find the optimal weight $\hat{\omega}$ in the interval $(0, 1]$. This generalisation allows for the possibility of using Mehrotra's corrector with a small weight, if that helps in producing a better stepsize; on the other hand, the choice $\hat{\omega} = 1$ yields Mehrotra's corrector again.

We have applied the weighting strategy to multiple centrality correctors as well. The justification in this case comes from the following argument. In Section 2.2 we have seen that the target point in the multiple centrality correctors technique depends on a parameter δ which measures the greediness of the centrality corrector. In the previous implementations, this parameter was fixed at coding time to a value determined after tuning to a series of representative test problems. However, for a specific problem such a value may be too conservative or too aggressive; moreover, the same value may not be optimal throughout a problem. Hence, it makes sense to provide a mechanism to adaptively change these correctors in order to increase their effectiveness.

Below we formalize the weighted correctors algorithm.

Given an initial iterate (x^0, y^0, s^0) such that $(x^0, s^0) > 0$, and the number M of corrections allowed at each iteration;

Repeat for $k = 0, 1, 2, \ldots$ until some convergence criteria are met:

- Solve system (2) with right-hand side r_1 (4) for a predictor direction Δ_a.
- Set μ according to (5) and find Mehrothra's corrector direction Δ_c by solving system (2) with right-hand side (6).
- Do a linesearch to find the optimal $\hat{\omega}$ that maximizes the stepsize α in $\Delta^\omega = \Delta_a + \omega \Delta_c$.
- Set $\Delta_p = \Delta_a + \hat{\omega} \Delta_c$.

  Do
  - Solve system (2) with right-hand side $(0, 0, t)$, t given by (7) for a centrality corrector direction Δ_m.
  - Perform a linesearch to find the optimal $\hat{\omega}$ that maximizes the stepsize α in $\Delta^\omega = \Delta_p + \omega \Delta_m$.
  - Set $\Delta_p := \Delta_p + \hat{\omega} \Delta_m$.

  While the maximum number of correctors M has not been reached and the stepsize has increased by at least a fraction of the aspiration level δ;
- Update the iterate $(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha_k \Delta_p (x^k, y^k, s^k)$.

End

5 Numerical results

We have implemented our proposal within the HOPDM interior point solver [7]. In our implementation we generate a sequence of multiple centrality correctors, and for each of them we choose the optimal weight $\hat{\omega}$ which maximizes the stepsizes in primal and dual spaces for a combined direction of form

$$\Delta = \Delta_p + \omega \Delta_c.$$

The composite direction $\Delta = \Delta_p + \hat{\omega}\Delta_c$ becomes a predictor for the next centrality corrector, hence the correcting process is recursive, and can be interrupted at any stage.

We use $\gamma = 0.1$ in the definition of symmetric neighbourhood and define aspiration levels for the stepsizes using the rule

$$\tilde{\alpha}_P = \min(1.5\alpha_P + 0.3,\ 1) \quad \text{and} \quad \tilde{\alpha}_D = \min(1.5\alpha_D + 0.3,\ 1).$$

These values are larger than the ones suggested in [8], $\tilde{\alpha} = \min(\alpha + 0.1,\ 1)$, because the weighting mechanism allows us to control the contribution of the corrector in an adaptive way. Centrality correctors are accepted in the primal and/or dual space if $\alpha_P^{new} \geq 1.01\alpha_P$ and/or $\alpha_D^{new} \geq 1.01\alpha_D$, respectively.

Concerning the choice of ω, we implemented a linesearch in the interval $[\omega_{\min}, \omega_{\max}] = [\alpha_P\alpha_D, 1]$. There are two reasons for using $\omega_{\min} = \alpha_P\alpha_D$. First, using the stepsizes α_P and α_D for the predictor direction gives

$$(X + \alpha_P \Delta X)(S + \alpha_D \Delta S)e \ = \ XSe + \alpha_P S\Delta X e + \alpha_D X \Delta S e + \alpha_P\alpha_D \Delta X \Delta S e,$$

and the term $\alpha_P\alpha_D$ appears with the second-order error. Secondly, the study of Cartis [3] suggests squaring the stepsize for the corrector. Our computational experience indicates that the straight use of $\omega = \omega_{\min} = \alpha_P\alpha_D$ is too conservative. Still, such $\omega_{\min}$ is a reliable lower bound for attractive weights ω.

In our crude linesearch procedure we choose 9 points uniformly distributed in the interval $[\alpha_P\alpha_D, 1]$ and evaluate, for each of these points, the stepsizes in both spaces. When a larger stepsize α_P or α_D is obtained, the corresponding ω is stored as ω_P or ω_D respectively. Hence, we allow two different weightings for directions in the primal and dual spaces.

The ultimate objective in choosing ω is to increase the stepsizes α_P and α_D. These stepsizes depend on ω in a complex way. Examples corresponding to a common behaviour are given in Figure 1, where we show how the product $\alpha_P\alpha_D$ varies depending on the choice of ω for Mehrotra's corrector at two different iterations of problem `capri` of the Netlib set. On the left, $\omega \in [0.4, 1]$ and $\hat{\omega} = 0.475$ gives a product $\alpha_P\alpha_D = 0.583$, better than a value of 0.477 that would have been obtained by using a full weight on Mehrotra's corrector. On the right, $\omega \in [0.178, 1]$ and the choice of $\omega \in (0.6, 0.7)$ leads to the best product $\alpha_P\alpha_D$ of about 0.375.

We tested our implementation in a series of computational experiments using test problems from different collections. Computations were performed on a Linux PC with a 3GHz Intel Pentium processor and 1GB of RAM. For the purpose of consistency, we decided to implement in HOPDM the criteria used in the study performed by Mehrotra and Li [13]. Therefore, optimal termination occurs when the following conditions are met:

$$\frac{\mu}{1 + |c^T x|} \leq 10^{-10}, \qquad \frac{\|b - Ax\|}{1 + \|b\|} \leq 10^{-8}, \qquad \frac{\|c - A^T y - s\|}{1 + \|c\|} \leq 10^{-8}. \tag{9}$$

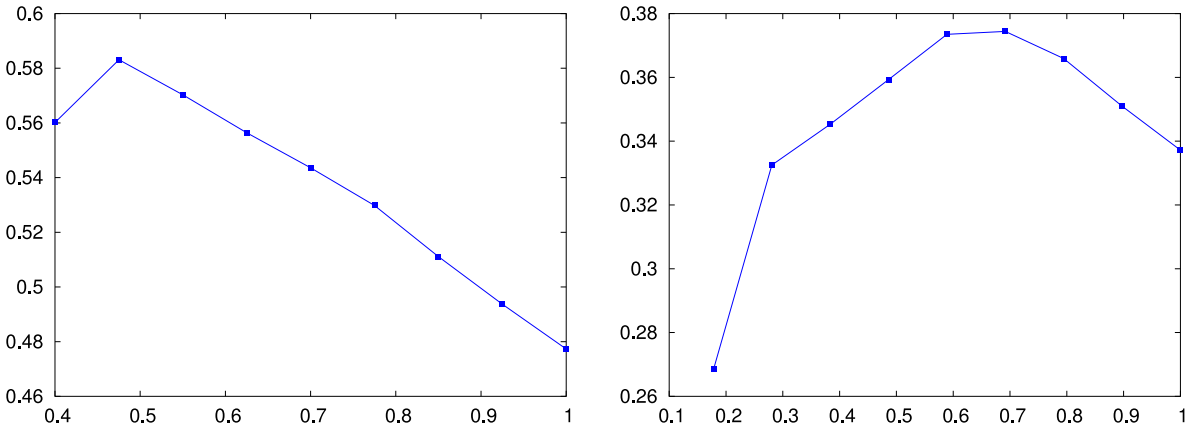

Figure 1: Relationship between ω and $\alpha_P \alpha_D$ in two iterations of problem `capri`.

5.1 Mehrotra-Li test collection

First we considered the test set used in [13]: it contains 101 problems from both Netlib and Kennington collections. We present our results in terms of number of iterations and number of backsolve operations. The rationale behind this decision is that the multiple centrality correctors technique determines the number of allowed correctors on the basis of the ratio between factorization cost and backsolve cost [8]. This ratio can be very different across implementations and is mainly influenced by the linear algebra routines used. For example, HOPDM [7] comes with an in-house linear algebra implementation, while PCx [5] relies on the more sophisticated sparse Cholesky solver of Ng and Peyton. Therefore, the PCx code tends to use less correctors per iteration.

In Table 1, column HO displays the results obtained by the previous implementation, while column dHO reports the results obtained by the current implementation of weighted correctors. The last column presents the relative change between the two versions of HOPDM tested. As a reference, we also report in this table the overall statistics of PCx (release 1.1) on these problems. Also for PCx we adopted the termination criteria (9). We found the number of backsolves by counting the number of calls to the functions `IRSOLV()` and `EnhanceSolve()`, for HOPDM and PCx respectively.

	PCx	HO	dHO	Change
Iterations	2114	1871	1445	-22.77%
Backsolves	4849	6043	5717	-5.39%
Backsolves/iter.	2.29	3.23	3.95	+22.29%

Table 1: Overall results obtained on Mehrotra and Li's test collection.

From Table 1 we first observe the very small number of backsolves per iteration needed by PCx. This is due to the fact that PCx allows the use of Gondzio's multiple centrality correctors only in 4 problems: `dfl001`, `maros-r7`, `pds-10` and `pds-20`. Also we notice that when we allow an adaptive weighting of the correctors there is a tendency to use more correctors per iteration than previously. This happens because the weighting mechanism makes it more likely to accept some correctors that otherwise would have been rejected as too aggressive. While this usually leads to a decrease in iteration count, it also makes each iteration more expensive.

In Table 2 we detail the problems for which we obtained savings in computational time. Given the small dimension of most of the problems in the Netlib collection, we did not expect major savings. However, as the problem sizes increase, we can see that the proposed way of evaluating and weighting the correctors becomes effective. This led us to investigate further the performance of the proposed implementation, which we will discuss in Section 5.2.

Problem	HO	dHO	Problem	HO	dHO
bnl1	0.36	0.25	pilot87	12.62	11.88
dfl001	150.63	114.80	pds-06	24.59	21.31
maros-r7	7.76	7.52	pds-10	96.57	79.29
pilot	5.23	4.35	pds-20	923.71	633.64

Table 2: Problems that showed time savings (times are in seconds).

We were particularly interested in comparing the results produced by our weighted correctors approach with those published in [13]. The computation of Krylov subspace directions in Mehrotra and Li's approach does involve considerable computational cost, as the computation of each Krylov direction requires a backsolve operation. This can be seen from the definition of the power basis matrix

$$G = I - H_k^{-1}\tilde{H},$$

which involves an inverse matrix. In fact, calling u the starting vector in the Krylov sequence, the computation of the vector $H_k^{-1}\tilde{H}u$ requires first to compute $v = \tilde{H}u$ (matrix–vector multiplication) and then to determine $t = H_k^{-1}v$ (backsolve on the Cholesky factors).

In the tables of results presented in [13], the best performance in terms of iteration count is obtained by PCx4, which uses 4 Krylov subspace vectors. These directions are combined with an affine-scaling predictor direction and Mehrotra's second-order correction, leading to 6 backsolves per iteration. This number increases when the linear subproblem produces an optimal objective value smaller than a specified threshold or the new iterate fails to satisfy some neighbourhood condition: in these cases the pure centering direction Δ_{cen} also needs to be computed, and a seventh backsolve is performed.

Table 4 in the Appendix presents a full comparison, in terms of iterations (Its) and backsolves (Bks), between the results obtained in [13] and the weighted correctors technique proposed in this paper. Again, we compare the two strategies according to the number of iterations and backsolves, as we do not have access to this version of PCx and therefore we cannot report CPU times. Columns PCx0, PCx2 and PCx4 repeat the results reported in [13] for 0, 2 and 4 Krylov directions, respectively. As we understand the paper [13], PCx0 uses exactly 2 backsolves per iteration: one to compute the affine-scaling direction, another to compute Mehrotra's corrector; PCx2 and PCx4 compute two and four additional Krylov vectors, hence they use 4 and 6 backsolves per iteration, respectively. In columns HO-0, HO-2 and HO-4, we present the results obtained by HOPDM when forcing the use of 0, 2 and 4 multiple centrality correctors. In the column called HO-∞ we report the results obtained when an unlimited number of correctors is allowed (in practice we allow no more than 20 correctors). The last column, labelled dHO, presents the results obtained by choosing the number of correctors allowed according to [8].

Consequently, up to 2, 4 and 6 backsolves per iteration are allowed in PCx0, PCx2 and PCx4 and in HO-0, HO-2 and HO-4 runs, respectively. The number of backsolves reported for HOPDM includes two needed by the initialisation procedure: the number of backsolves should not exceed $2 \times \text{Its} + 2$, $4 \times \text{Its} + 2$ and $6 \times \text{Its} + 2$ respectively for HO-0, H0-2 and H0-4. The observed

number of backsolves is often much smaller than these bounds because the correcting mechanism switches off when the stepsizes are equal to 1 or when the corrector does not improve the stepsize. Problem `afiro` solved by HO-4 needs 24 backsolves, 22 of which compute different components of directions, hence the average number of backsolves per iteration is only 22/6 and is much smaller than 6. Occasionally, as a consequence of numerical errors, certain components of direction are rejected on the grounds of insufficient accuracy: in such case the number of backsolves may exceed the stated upper bounds. The reader may observe for example that `pilot4` is solved by HO-4 in 16 iterations, but the number of backsolves is equal to 100 and exceeds $6 \times 16 + 2 = 98$.

The results presented in Table 4 allow us to conclude that the new implementation of multiple centrality correctors leads to significant savings in the number of iterations compared with Mehrotra and Li's approach. HO-2 needs 1418 iterations as opposed to 1752 needed by PCx2, a saving of 334 iterations, that is 19%. HO-4 saves 149 iterations (10%) over PCx4.

The version HO-∞ requires 1139 iterations to solve the collection of 101 problems, an average of just above 11 iterations per problem. This version has only an academic interest, yet it reveals a spectacular efficiency of interior point methods which can solve difficult linear programs of medium sizes (reaching a couple of hundred thousand variables) in just a few iterations. In particular, it suggests that if we had a cheap way of generating search directions, then it would be beneficial to have as many as possible.

5.2 Beyond Netlib

We have applied our algorithm to examples from other test collections besides Netlib. These include other medium to large linear programming problems, stochastic problems and quadratic programming problems.

We used a collection of medium to large problems taken from different sources: problems `CH` through `CO9` are MARKAL (Market Allocation) models; `mod2` through `worldl` are agricultural models used earlier in [8]; problems `route` through `rlfdual` can be retrieved from `http://www.sztaki.hu/~meszaros/public_ftp/lptestset/misc/`, problems `neos1` through `fome13` can be retrieved from `ftp://plato.asu.edu/pub/lptestset/`. Complete statistics can be found in Table 5 in the Appendix, where we compare the performance of different versions of the algorithm when forcing a specified number of multiple centrality corrector directions.

In Table 3 we provide a time comparison between our previous implementation (shown in column HO), and the current one based on weighted correctors (column dHO). This test collection contains problems large enough to show a consistent improvement in CPU time: in only 4 problems (`mod2`, `dbc1`, `watson-1`, `sgpf5y6`) we recorded a degradation of the performance by more than 1 second. The improvements are significant on problems with a large number of nonzero elements. In these cases, dHO produces savings from about 10% to 30%, with the remarkable results in `rail2586` and `rail4284`, for which the relative savings reach 45% and 65%, respectively.

In Figure 2, we show the CPU-time based performance profile [6] for the two algorithms. This graph shows the proportion of problems that each algorithm has solved within τ times of the best. Hence, for $\tau = 1$ it indicates that dHO has been the best solver on 72% of the problems, against 28% for HO. For larger values of τ, the performance profile for dHO stays above the one

Problem	HO	dHO	Change
CH	1.03	1.23	19.4%
GE	5.72	5.46	-4.5%
NL	4.37	3.95	-9.6%
BL	2.15	2.14	-0.5%
BL2	2.35	2.31	-1.7%
UK	2.48	3.21	29.4%
CQ5	2.54	2.60	2.4%
CQ9	9.67	8.84	-8.6%
CO5	3.16	3.59	13.6%
CO9	21.10	15.35	-27.3%
mod2	20.59	21.68	5.3%
world	26.35	23.41	-11.2%
world3	31.13	27.49	-11.7%
world4	73.21	56.14	-23.3%
world6	39.33	32.79	-16.6%
world7	43.14	36.02	-16.5%
worldl	43.95	36.82	-16.2%
route	40.92	33.78	-17.4%
ulevi	9.04	9.55	5.6%
ulevimin	16.52	16.46	-0.4%
dbir1	162.18	146.51	-9.7%

Problem	HO	dHO	Change
dbir2	208.93	156.11	-25.3%
dbic1	72.96	77.31	5.9%
pcb1000	0.26	0.33	26.9%
pcb3000	1.13	1.16	2.7%
rlfprim	15.63	15.08	-3.5%
rlfdual	71.17	49.79	-30.0%
neos1	169.11	141.89	-16.1%
neos2	113.86	86.13	-24.4%
neos3	132.02	120.59	-8.7%
neos	1785.80	1386.58	-22.4%
watson-1	138.60	166.21	19.9%
sgpf5y6	49.58	64.45	30.0%
storm-1000	1661.54	1623.19	-2.3%
rail507	9.77	10.10	3.4%
rail516	7.59	5.89	-22.4%
rail582	9.67	9.60	-0.7%
rail2586	1029.36	566.82	-44.9%
rail4284	2779.63	978.48	-64.8%
fome11	407.20	265.21	-34.9%
fome12	766.96	508.61	-33.7%
fome13	1545.05	990.62	-35.9%

Table 3: Time comparison on large problems (times are in seconds).

for HO, thus confirming its superiority. In particular, it solves all problems within 1.3 times of the best.

The collection of stochastic programming problems contains 119 examples and comes from `http://www.sztaki.hu/~meszaros/public_ftp/lptestset/stochlp/`. The full set of results on these problems is shown in Table 6 in the Appendix. We see again that the dHO version solves this set of problems in 1468 iterations as opposed to HO-0 which needs 2098 iterations. The HO-∞ version (of academic interest only) solves this set of problems in 1181 iterations, which gives an astonishing average of merely 10 iterations per problem.

We have also tested the implementation on a collection of 29 quadratic programming problems, available from `http://www.sztaki.hu/~meszaros/public_ftp/qpdata/`. Normally, HOPDM automatically chooses between direct and iterative approaches for computing directions. A higher-order correcting scheme makes much more sense with a direct method when the back-solve is significantly less expensive than the factorization. In order to maintain consistency, we forced HOPDM to use a direct approach rather than an iterative scheme when solving this class of problems. Complete results are presented in Table 7. The analysis of these results confirms that the use of multiple centrality correctors applied to quadratic programs leads to remarkable savings in the number of iterations. It is often the case that due to the presence of complicated sparsity structure in the quadratic terms, the factorizations in quadratic programming get very expensive. Such a situation favours the use of many correctors, and the good performance of dHO can be noticed on large instances such as `aug3dc`, `aug3dcqp` and `boyd2`.

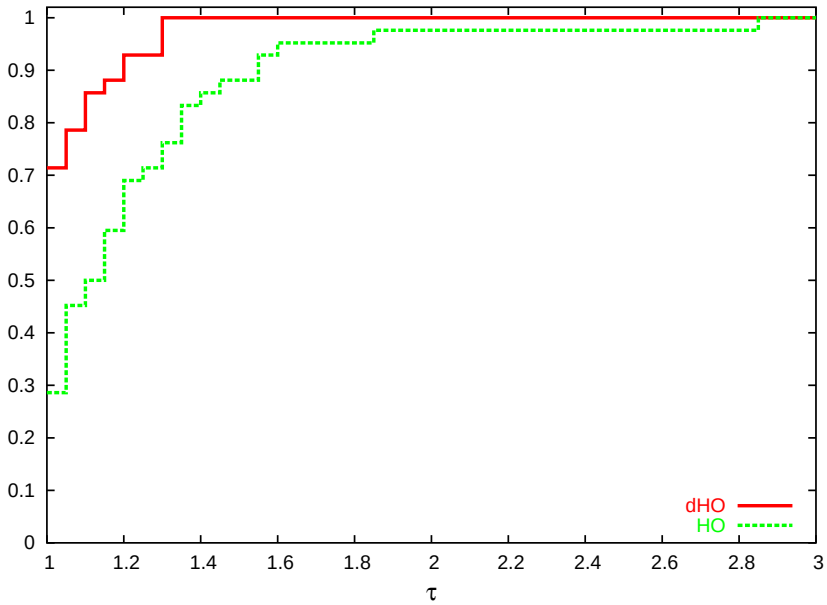

Figure 2: Performance profile for HO and dHO on the set of problems of Table 3.

6 Conclusions

In this paper we have revisited the technique of multiple centrality correctors [8] and added a new degree of freedom to it. Instead of computing the corrected direction from $\Delta = \Delta_p + \Delta_c$ where Δ_p and Δ_c are the predictor and corrector terms, we allow a choice of weight $\omega \in (0, 1]$ for the corrector term and compute $\Delta = \Delta_p + \omega\Delta_c$. We combined this modification with the use of a symmetric neighbourhood of the central path. We have shown that the use of this neighbourhood does not cause any loss in the worst-case complexity of the algorithm.

The computational results presented for different classes of problems demonstrate the potential of the proposed scheme. The use of new centrality correctors reduces the number of IPM iterations needed to solve a standard set of 101 small to medium scale linear problems from 1871 iterations to 1445, and similar savings are produced for other classes of problems including quadratic programs. Further savings of the number of iterations are possible after adding more correctors: the number of iterations on the same test set is reduced to 1139. Tested on a collection of 220 problems, this version needs 2320 iterations, an average of 11 iterations per problem. It should be noted, however, that the use of too many correctors does not minimize the CPU time.

We have compared our algorithm against the recently introduced Krylov subspace scheme [13]. The two approaches have similarities: they look for a set of attractive independent terms from which the final direction is constructed. Mehrotra and Li's approach uses the first few elements from the basis of the Krylov space; our approach generates direction terms using centrality correctors of [8]. Mehrotra and Li's approach solves an auxiliary linear program to find an optimal combination of all available direction terms; our approach repeatedly chooses the best weight for each newly constructed corrector term (and switches off if the use of the corrector does not offer sufficient improvement). Eventually, after adding k corrector terms, the directions used in our approach have form

$$\Delta = \Delta_a + \omega_1\Delta_1 + \ldots + \omega_k\Delta_k,$$

and the affine-scaling term Δ_a contributes to it without any reduction. Hence, the larger the stepsize, the more progress we make towards the optimizer.

The comparison presented in Section 5.1 shows a clear advantage of our scheme over that of [13]. Indeed, with the same number of direction terms allowed, our scheme outperforms Krylov subspace correctors by a wide margin. Multiple centrality correctors show consistent excellent performance on other classes of problems including medium to large scale linear programs beyond the Netlib collection and medium scale quadratic programs.

A Tables of results

Problem	PCx0	HO-0		PCx2	HO-2		PCx4	HO-4		HO-∞		dHO	
	Its	Its	Bks	Its	Its	Bks	Its	Its	Bks	Its	Bks	Its	Bks
25fv47	25	27	55	18	18	75	15	15	95	14	184	18	76
80bau3b	36	31	64	26	19	79	22	15	93	13	195	18	78
adlittle	11	12	25	10	9	33	8	9	47	9	91	10	29
afiro	7	8	13	6	6	18	6	6	24	6	35	7	15
agg	18	21	43	15	14	57	10	13	79	12	143	14	57
agg2	22	22	46	18	15	61	15	14	86	12	151	15	61
agg3	21	20	41	17	15	62	14	15	84	14	147	15	62
bandm	17	14	30	13	10	40	11	10	58	9	106	11	34
beaconfd	10	10	19	9	8	30	7	8	40	9	82	8	23
blend	9	10	20	8	8	30	7	8	42	8	61	8	23
bnl1	36	39	80	35	22	93	27	17	107	17	162	25	91
bnl2	32	25	51	24	16	67	19	16	93	13	171	16	93
boeing1	20	20	40	16	14	61	12	13	76	11	142	15	47
boeing2	12	14	28	11	11	41	10	11	61	10	117	12	34
bore3d	15	15	28	12	12	48	11	11	60	10	102	12	33
brandy	19	19	38	16	14	58	13	14	82	12	160	17	50
capri	18	19	38	15	13	49	12	11	63	11	121	14	43
cycle	23	27	55	15	19	81	13	18	104	15	165	19	83
czprob	27	23	46	17	17	67	16	14	82	13	176	19	60
d2q06c	29	31	64	22	20	83	18	17	105	14	201	17	106
d6cube	19	17	35	14	14	54	11	12	64	11	139	12	64
degen2	12	14	29	10	11	44	8	10	62	9	100	11	44
degen3	16	20	42	12	14	57	10	14	84	11	121	14	84
dfl001	47	46	97	39	31	125	31	26	152	25	329	22	222
e226	17	19	40	14	15	64	11	14	87	12	132	16	49
etamacro	23	20	42	17	13	52	14	12	73	12	141	13	52
fffff800	27	30	62	19	20	84	16	17	108	17	244	19	84
finnis	23	21	42	18	18	60	14	29	169	17	197	26	81
fit1d	17	16	34	13	14	60	11	12	75	11	148	15	47
fit1p	17	18	38	13	13	56	12	12	72	11	156	15	47
fit2d	22	29	60	14	16	66	12	17	107	12	160	24	83
fit2p	20	24	49	15	16	66	12	15	90	12	150	17	54
forplan	22	19	39	16	16	66	13	13	80	13	145	15	46
ganges	19	12	26	13	9	40	11	8	50	8	93	10	32
gfrd-pnc	18	14	27	12	10	36	11	9	53	8	91	11	32
greenbea	36	37	76	32	23	94	28	23	146	20	263	22	98
greenbeb	32	41	84	28	26	108	25	22	139	19	242	28	99
grow15	18	11	22	19	9	33	13	8	50	7	94	9	26

Table 4: Comparison with Mehrotra and Li's algorithm.

Problem	PCx0	HO-0		PCx2	HO-2		PCx4	HO-4		HO-∞		dHO	
	Its	Its	Bks	Its	Its	Bks	Its	Its	Bks	Its	Bks	Its	Bks
grow22	21	11	22	21	9	36	15	8	49	7	66	9	28
grow7	15	11	22	15	8	29	11	8	46	7	57	9	26
israel	19	21	44	15	14	59	12	13	81	11	135	16	52
kb2	12	12	24	9	10	37	8	9	47	8	75	10	29
lotfi	14	14	28	11	10	38	9	9	47	9	77	11	31
maros-r7	17	15	30	13	11	43	11	11	62	9	114	10	68
maros	19	20	41	15	15	64	12	13	79	11	136	16	51
nesm	25	27	56	20	17	70	17	14	87	12	153	17	72
perold	32	24	49	25	17	70	21	14	87	13	172	16	69
pilot	36	27	56	25	19	78	22	15	95	14	170	15	95
pilot4	68	30	62	61	19	79	53	16	100	17	228	19	84
pilotja	29	30	62	23	18	77	21	15	95	14	179	16	84
pilotnov	17	15	30	14	10	39	11	10	53	9	87	10	46
pilotwe	48	30	62	31	19	79	27	15	94	15	204	19	83
pilot87	34	31	64	25	18	76	19	16	99	14	220	15	125
recipe	9	8	15	8	7	25	7	7	35	7	49	7	18
scagr25	16	16	32	13	10	40	11	9	52	8	92	11	35
scagr7	14	12	23	11	10	40	9	10	56	9	82	11	32
scfxm1	18	17	35	14	12	51	11	10	63	9	112	14	43
scfxm2	19	19	40	15	13	55	12	11	70	10	140	15	46
scfxm3	21	19	40	15	13	54	12	12	73	10	128	15	47
scrs8	21	17	35	15	12	48	13	11	72	10	112	13	40
scsd1	9	8	15	8	8	25	7	7	30	7	54	7	18
scsd6	12	10	20	10	9	29	9	8	41	8	68	9	24
scsd8	12	11	21	10	8	28	8	7	33	7	60	9	23
sctap1	16	17	33	12	12	44	9	11	58	11	114	13	37
sctap2	13	15	29	10	11	38	9	10	51	8	86	12	33
sctap3	13	15	30	11	11	34	10	11	51	8	87	12	33
seba	14	9	15	11	7	23	9	7	33	7	114	8	19
share1b	19	21	43	15	15	61	12	13	82	12	129	17	54
share2b	17	15	29	14	11	41	12	9	43	8	72	12	35
shell	20	21	42	16	12	50	13	12	73	10	128	14	44
ship04l	12	11	24	10	10	40	8	8	49	8	118	10	31
ship04s	13	12	26	10	9	41	8	9	53	8	104	10	31
ship12l	14	13	28	13	9	36	11	8	47	8	103	11	34
ship12s	12	14	29	10	10	41	8	9	51	8	102	11	34
sierra	19	18	38	15	12	48	11	11	66	10	104	14	43
stair	14	18	36	12	12	46	11	16	91	11	145	12	46
standata	13	16	30	10	12	43	8	12	61	10	117	13	35
standmps	22	21	40	20	16	63	16	14	78	13	147	16	46
stocfor1	11	12	22	9	9	32	7	8	43	8	77	9	26
stocfor2	20	19	38	15	14	58	13	12	74	10	151	15	46
truss	20	17	35	15	11	46	12	10	61	9	113	11	46
tuff	19	15	30	15	10	39	12	9	51	9	98	10	39
vtpbase	11	11	20	9	9	34	7	9	40	8	103	9	25
wood1p	19	22	43	16	16	65	14	14	85	13	162	16	65
woodw	30	29	59	21	18	74	19	16	99	14	167	18	77
cre-a	24	25	52	18	18	74	15	15	97	12	143	18	59
cre-b	40	42	86	28	23	96	23	24	148	16	219	22	99
cre-c	25	28	57	19	19	80	15	16	98	14	184	19	63
cre-d	39	43	88	29	23	96	26	22	137	17	233	22	100
ken-07	15	12	25	12	10	39	10	8	48	7	107	10	31

Table 4: Comparison with Mehrotra and Li's algorithm.

Problem	PCx0	HO-0		PCx2	HO-2		PCx4	HO-4		HO-∞		dHO	
	Its	Its	Bks	Its	Its	Bks	Its	Its	Bks	Its	Bks	Its	Bks
ken-11	21	15	32	15	11	47	13	10	63	9	133	13	40
ken-13	28	19	40	21	13	55	16	11	69	11	138	15	47
ken-18	30	24	49	23	15	63	21	13	78	9	139	15	63
osa-07	18	23	47	20	12	52	12	13	73	11	111	13	41
osa-14	19	19	39	21	15	61	18	12	73	11	106	15	47
osa-30	24	20	40	24	17	73	18	15	93	13	147	17	60
osa-60	22	20	41	30	18	76	15	12	70	11	118	14	46
pds-02	25	19	38	19	15	60	15	14	77	11	124	15	60
pds-06	35	27	56	27	19	78	23	17	100	15	177	17	118
pds-10	41	32	66	31	24	100	27	20	120	18	199	18	155
pds-20	58	48	98	42	26	105	34	23	136	21	253	21	198
Totals	2194	2047	4169	1752	1418	5719	1438	1289	7608	1139	13699	1445	5717

Table 4: Comparison with Mehrotra and Li's algorithm.

Problem	HO-0		HO-2		HO-4		HO-∞		dHO	
	Its	Bks	Its	Bks	Its	Bks	Its	Bks	Its	Bks
CH	26	54	17	70	15	94	12	155	16	83
GE	39	80	24	100	20	127	23	307	20	133
NL	32	66	19	79	15	93	13	192	15	93
BL	28	58	17	70	15	93	13	182	16	85
BL2	32	66	17	70	15	94	13	189	16	85
UK	30	62	19	78	15	93	14	191	19	82
CQ5	41	84	26	107	22	135	18	253	24	109
CQ9	49	100	29	119	24	154	22	299	26	152
CO5	50	102	31	128	28	179	36	472	30	142
CO9	58	118	44	147	62	357	52	627	41	237
mod2	38	78	25	102	21	128	15	248	22	121
world	42	86	24	98	21	128	16	230	22	121
world3	52	106	35	144	28	172	19	291	28	158
world4	51	104	31	128	27	164	21	281	26	170
world6	44	90	29	120	23	144	19	261	22	145
world7	40	82	28	115	22	135	17	267	21	137
worldl	52	106	34	139	25	156	20	282	24	157
route	21	42	14	56	14	80	11	114	14	80
ulevi	26	51	18	65	17	103	15	185	17	103
ulevimin	68	138	35	146	27	172	22	306	27	186
dbir1	22	44	15	61	13	80	9	112	12	104
dbir2	25	52	15	60	13	81	12	167	11	104
dbic1	54	109	26	108	20	128	17	213	24	109
pcb1000	20	42	15	62	12	74	11	138	15	62
pcb3000	23	48	15	63	13	80	12	165	15	63
rlfprim	13	25	8	30	8	40	7	65	8	40
rlfdual	13	25	10	37	9	49	9	80	9	63
neos1	55	111	33	139	29	179	21	275	24	195
neos2	40	81	24	103	24	140	19	210	19	125
neos3	49	98	32	183	35	191	30	326	28	195
neos	109	220	48	196	41	260	41	591	49	430
watson-1	112	226	45	187	30	192	46	655	49	195
sgpf5y6	45	92	26	112	24	157	19	214	30	109
stormG2-1000	130	261	78	331	66	398	53	687	73	316
rail507	35	71	20	82	15	91	14	199	19	81

Table 5: Comparison on large problems.

Problem	HO-0		HO-2		HO-4		HO-∞		dHO	
	Its	Bks	Its	Bks	Its	Bks	Its	Bks	Its	Bks
rail516	30	61	15	56	12	67	10	142	15	56
rail582	45	92	20	84	16	102	19	229	19	84
rail2586	99	200	30	125	24	147	20	287	24	149
rail4284	65	132	30	122	25	158	22	361	19	213
fome11	44	92	31	129	25	150	23	294	24	224
fome12	44	91	29	125	25	147	20	286	22	215
fome13	43	91	29	123	24	145	20	289	20	215
Totals	1934	3937	1110	4599	959	5857	845	11317	974	5926

Table 5: Comparison on large problems.

Problem	HO-0		HO-2		HO-4		HO-∞		dHO	
	Its	Bks	Its	Bks	Its	Bks	Its	Bks	Its	Bks
aircraft	10	19	7	28	7	36	7	47	8	21
cep1	18	37	14	54	12	69	10	125	14	43
deter0	16	33	11	46	11	67	10	115	11	46
deter1	20	40	12	48	11	68	9	120	11	68
deter2	18	38	12	49	10	60	9	121	9	64
deter3	18	37	14	54	12	75	10	129	11	73
deter4	16	33	11	44	10	62	9	108	9	68
deter5	18	37	14	56	12	71	11	116	12	71
deter6	17	34	12	48	10	59	9	116	10	59
deter7	18	37	13	56	12	72	11	112	11	77
deter8	19	38	12	49	12	71	10	135	12	71
fxm2-16	24	50	17	71	15	94	12	182	17	73
fxm2-6	22	45	14	59	13	81	11	128	16	52
fxm3-16	40	82	22	91	20	124	18	244	26	97
fxm3-6	27	56	17	70	16	100	14	205	21	72
fxm4-6	30	62	18	75	17	104	15	176	22	75
pgp2	25	52	16	67	14	85	13	157	18	59
pltexpA2-16	17	34	11	43	11	62	11	133	11	62
pltexpA2-6	15	30	12	46	11	62	10	110	12	46
pltexpA3-16	36	74	21	89	19	119	16	203	19	124
pltexpA3-6	23	47	16	69	15	90	12	147	16	69
pltexpA4-6	40	82	25	105	20	126	19	270	23	105
sc205-2r-100	13	25	10	38	9	47	8	56	11	30
sc205-2r-16	10	20	8	30	8	40	7	79	9	24
sc205-2r-1600	17	34	17	65	11	48	11	79	14	44
sc205-2r-200	11	21	11	40	9	43	8	62	11	30
sc205-2r-27	11	20	8	26	7	31	7	54	8	26
sc205-2r-32	14	27	9	31	8	38	8	117	9	31
sc205-2r-4	9	17	7	24	7	32	7	69	7	18
sc205-2r-400	15	29	11	42	10	49	10	70	12	35
sc205-2r-50	18	36	10	38	9	50	9	78	13	38
sc205-2r-64	13	26	9	36	8	45	8	74	10	28
sc205-2r-8	9	17	7	25	7	37	7	51	8	20
sc205-2r-800	17	35	11	40	10	48	11	86	12	37
scagr7-2b-16	13	28	12	49	10	59	10	106	14	43
scagr7-2b-4	13	25	11	45	11	63	10	118	11	32
scagr7-2b-64	23	48	15	60	12	71	12	141	16	50
scagr7-2c-16	14	29	12	49	11	60	12	186	13	38
scagr7-2c-4	12	24	13	55	11	67	10	108	10	30

Table 6: Comparison on stochastic programming problems.

Problem	HO-0		HO-2		HO-4		HO-∞		dHO	
	Its	Bks	Its	Bks	Its	Bks	Its	Bks	Its	Bks
scagr7-2c-64	19	40	13	53	12	67	13	144	13	39
scagr7-2r-108	20	42	14	57	13	78	11	132	16	51
scagr7-2r-16	14	29	12	48	10	61	10	94	13	40
scagr7-2r-216	20	42	15	61	14	83	11	116	15	46
scagr7-2r-27	19	40	12	48	11	69	11	137	13	40
scagr7-2r-32	18	38	11	46	12	68	11	109	12	37
scagr7-2r-4	13	27	10	43	10	58	10	108	10	31
scagr7-2r-432	23	48	16	65	13	79	14	170	17	54
scagr7-2r-54	18	38	12	49	11	65	10	114	14	43
scagr7-2r-64	20	42	12	48	13	75	11	128	14	43
scagr7-2r-8	13	27	11	47	10	60	11	127	11	34
scagr7-2r-864	25	52	16	65	14	85	12	155	18	58
scfxm1-2b-16	25	51	15	61	15	86	14	152	15	61
scfxm1-2b-4	20	40	13	52	14	80	11	124	15	44
scfxm1-2b-64	37	75	25	103	20	118	17	175	25	89
scfxm1-2c-4	20	40	13	50	13	72	12	121	15	44
scfxm1-2r-128	34	69	26	108	22	137	16	176	25	89
scfxm1-2r-16	25	51	17	66	15	89	13	144	17	67
scfxm1-2r-256	49	99	27	114	28	181	21	230	27	99
scfxm1-2r-27	29	60	19	75	16	92	14	176	18	76
scfxm1-2r-32	30	61	18	73	16	93	16	159	17	71
scfxm1-2r-4	20	41	13	50	13	76	12	145	15	45
scfxm1-2r-64	33	67	19	77	18	110	16	168	22	77
scfxm1-2r-8	22	45	15	60	13	77	13	132	17	52
scfxm1-2r-96	37	75	22	90	18	116	17	187	25	91
scrs8-2b-16	7	11	7	17	7	17	6	35	7	13
scrs8-2b-4	8	12	6	15	6	17	6	50	6	12
scrs8-2b-64	10	18	8	28	8	37	7	72	8	21
scrs8-2c-16	8	13	6	16	6	18	6	35	6	13
scrs8-2c-32	7	12	6	21	6	26	6	56	6	15
scrs8-2c-4	8	12	6	15	6	17	6	52	6	12
scrs8-2c-64	8	16	6	28	6	34	6	61	6	19
scrs8-2c-8	8	12	6	15	6	17	6	38	7	13
scrs8-2r-128	10	20	8	33	8	38	7	60	9	25
scrs8-2r-16	7	12	6	17	6	19	6	34	6	16
scrs8-2r-256	10	21	8	34	8	39	7	80	9	26
scrs8-2r-27	8	14	6	20	6	32	6	56	6	15
scrs8-2r-32	6	11	6	13	6	15	6	31	6	14
scrs8-2r-4	7	11	6	15	6	17	6	32	6	12
scrs8-2r-512	13	27	9	36	8	45	7	81	9	26
scrs8-2r-64	6	11	6	15	6	17	6	33	6	14
scrs8-2r-64b	9	18	8	30	7	32	7	73	8	24
scrs8-2r-8	9	16	8	25	8	35	7	61	8	20
scsd8-2b-16	7	13	6	17	6	23	5	49	6	17
scsd8-2b-4	7	12	6	18	6	22	5	48	6	14
scsd8-2b-64	7	13	5	17	5	26	5	38	5	14
scsd8-2c-16	6	11	5	14	5	18	5	34	5	14
scsd8-2c-4	7	12	6	19	6	25	6	53	6	14
scsd8-2c-64	6	12	5	19	5	26	5	37	5	14
scsd8-2r-108	12	25	8	35	7	42	7	83	10	33
scsd8-2r-16	7	13	6	20	6	23	5	37	6	20
scsd8-2r-216	13	27	8	34	7	43	7	89	10	33

Table 6: Comparison on stochastic programming problems.

Problem	HO-0		HO-2		HO-4		HO-∞		dHO	
	Its	Bks	Its	Bks	Its	Bks	Its	Bks	Its	Bks
scsd8-2r-27	11	23	8	35	7	42	7	79	7	36
scsd8-2r-32	7	13	6	20	5	22	5	37	6	20
scsd8-2r-4	7	12	6	19	6	25	5	52	6	14
scsd8-2r-432	13	26	8	34	7	43	6	69	10	31
scsd8-2r-54	11	22	8	32	7	42	6	71	9	27
scsd8-2r-64	7	13	5	16	5	22	5	38	6	15
scsd8-2r-8	7	13	6	17	6	23	5	34	6	15
scsd8-2r-8b	7	13	6	17	6	23	5	34	6	15
sctap1-2b-16	14	28	10	35	9	44	9	90	9	37
sctap1-2b-4	12	23	9	32	9	37	8	88	10	27
sctap1-2b-64	23	46	14	58	12	67	12	139	17	53
sctap1-2c-16	15	30	10	38	9	44	9	102	10	42
sctap1-2c-4	12	23	9	30	9	42	8	88	10	27
sctap1-2c-64	18	36	12	45	10	53	10	109	13	38
sctap1-2r-108	19	38	13	52	11	69	10	113	13	41
sctap1-2r-16	12	24	10	38	9	49	8	76	10	42
sctap1-2r-216	20	40	14	56	12	72	11	121	13	40
sctap1-2r-27	15	30	10	39	10	53	9	98	10	53
sctap1-2r-32	15	29	10	39	10	57	8	107	9	54
sctap1-2r-4	12	22	9	33	9	41	8	81	10	26
sctap1-2r-480	24	48	17	70	11	70	12	143	17	52
sctap1-2r-54	18	36	11	44	10	57	9	118	13	38
sctap1-2r-64	15	30	10	37	9	48	9	107	12	35
sctap1-2r-8	12	23	9	33	8	38	8	83	9	33
sctap1-2r-8b	13	24	10	38	9	43	8	77	10	38
stormG2-125	93	188	58	240	43	263	34	459	53	218
stormG2-27	76	154	40	170	32	198	26	350	31	267
stormG2-8	47	95	27	112	23	142	19	275	23	127
Totals	2098	4204	1440	5634	1298	7209	1181	13002	1468	5414

Table 6: Comparison on stochastic programming problems.

Problem	HO-0		HO-2		HO-4		HO-∞		dHO	
	Its	Bks	Its	Bks	Its	Bks	Its	Bks	Its	Bks
qp500-1	13	25	12	48	13	75	11	142	13	93
qp500-2	18	35	16	65	15	95	12	133	15	116
qp500-3	8	12	7	22	7	24	6	50	7	27
qp1000-1	14	27	12	43	14	81	10	128	10	58
sqp2500-1	15	24	12	39	12	55	12	145	12	71
sqp2500-2	20	38	15	58	14	78	12	179	14	128
sqp2500-3	19	35	14	55	13	66	13	186	13	119
aug2d	12	21	10	41	10	62	9	93	11	54
aug2dc	13	22	11	45	10	57	9	87	10	52
aug2dcqp	7	12	8	25	8	31	7	107	7	26
aug2dqp	8	12	8	12	8	12	8	65	8	12
aug3d	10	15	9	27	9	38	9	108	9	53
aug3dc	35	71	37	149	39	234	29	408	23	161
aug3dcqp	41	81	34	154	24	150	17	219	20	184
aug3dqp	10	13	10	15	10	17	10	52	10	20
powell20	9	18	9	35	7	38	7	53	9	36
yao	15	28	14	57	15	79	22	179	24	103
cvxqp1	11	20	7	31	7	43	7	96	7	69

Table 7: Comparison on quadratic programming problems.

Problem	HO-0		HO-2		HO-4		HO-∞		dHO	
	Its	Bks	Its	Bks	Its	Bks	Its	Bks	Its	Bks
cvxqp2	12	20	7	31	7	43	7	92	7	52
cvxqp3	22	45	14	57	11	67	11	134	13	125
boyd1	27	53	25	93	23	129	18	327	24	100
boyd2	56	113	37	155	40	243	25	268	32	156
cont-100	12	23	20	74	19	102	20	228	19	122
cont-101	9	17	8	26	9	43	10	116	9	50
cont-200	11	22	19	71	19	100	19	187	19	117
cont-201	10	19	9	32	8	45	8	67	8	45
cont-300	10	20	10	42	8	43	9	88	7	49
exdata	11	21	9	40	9	49	8	69	9	61
selfqmin	22	46	14	58	13	80	11	137	12	110
Totals	480	908	417	1600	401	2179	356	4152	381	2369

Table 7: Comparison on quadratic programming problems.

References

[1] E. D. Andersen, J. Gondzio, C. Mészáros, and X. Xu, *Implementation of interior point methods for large scale linear programming*, in Interior Point Methods in Mathematical Programming, T. Terlaky, ed., Kluwer Academic Publishers, 1996, pp. 189–252.

[2] T. J. Carpenter, I. J. Lustig, J. M. Mulvey, and D. F. Shanno, *Higher-order predictor-corrector interior point methods with application to quadratic objectives*, SIAM Journal on Optimization, 3 (1993), pp. 696–725.

[3] C. Cartis, *Some disadvantages of a Mehrotra-type primal-dual corrector interior-point algorithm for linear programming*, Technical Report 04/27, Numerical Analysis Group, Computing Laboratory, Oxford University, 2004.

[4] ——, *On the convergence of a primal-dual second-order corrector interior point algorithm for linear programming*, Technical Report 05/04, Numerical Analysis Group, Computing Laboratory, Oxford University, 2005.

[5] J. Czyzyk, S. Mehrotra, and S. Wright, *PCx user guide*, Technical Report OTC 96/01, Optimization Technology Center, May 1996.

[6] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[7] J. Gondzio, *HOPDM (version 2.12) – A fast LP solver based on a primal-dual interior point method*, European Journal of Operational Research, 85 (1995), pp. 221–225.

[8] ——, *Multiple centrality corrections in a primal-dual method for linear programming*, Computational optimization and applications, 6 (1996), pp. 137–156.

[9] J.-P. Haeberly, M. Nayakkankuppam, and M. Overton, *Extending Mehrotra and Gondzio higher order methods to mixed semidefinite-quadratic-linear programming*, Optimization Methods and Software, 11 (1999), pp. 67–90.

[10] F. Jarre and M. Wechs, *Extending Merhotra's corrector for linear programs*, Advanced Modeling and Optimization, 1 (1999), pp. 38–60.

[11] I. J. Lustig, R. E. Marsten, and D. F. Shanno, *On implementing Mehrotra's predictor–corrector interior-point method for linear programming*, SIAM Journal on Optimization, 2 (1992), pp. 435–449.

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

[13] S. Mehrotra and Z. Li, *Convergence conditions and Krylov subspace-based corrections for primal-dual interior-point method*, SIAM Journal on Optimization, 15 (2005), pp. 635–653.

[14] S. Mizuno, M. Todd, and Y. Ye, *On adaptive step primal-dual interior-point algorithms for linear programming*, Mathematics of Operations Research, 18 (1993), pp. 964–981.

[15] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, 1970.

[16] M. Salahi, J. Peng, and T. Terlaky, *On Mehrotra-type predictor-corrector algorithms*, AdvOl Report 2005/4, McMaster University, 2005.

[17] R. Tapia, Y. Zhang, M. Saltzman, and A. Weiser, *The Mehrotra predictor-corrector interior-point method as a perturbed composite Newton method*, SIAM Journal on Optimization, 6 (1996), pp. 47–56.

[18] S. A. Vavasis and Y. Ye, *A primal–dual interior point method whose running time depends only on the constraint matrix*, Mathematical Programming, 74 (1996), pp. 79–120.

[19] S. J. Wright, *Primal-dual interior-point methods*, SIAM, Philadelphia, 1997.