ON IMPLEMENTATION OF A SELF-DUAL EMBEDDING METHOD FOR CONVEX PROGRAMMING*

JOHNNY TAK WAI CHENG and SHUZHONG ZHANG†

Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong

(Received 15 October 2003; Revised 13 August 2004)

In this paper, we implement Zhang's method [22], which transforms a general convex optimization problem with smooth convex constraints into a convex conic optimization problem and then apply the techniques of self-dual embedding and central path following for solving the resulting conic optimization model. A crucial advantage of the approach is that no initial solution is required, and the method is particularly suitable when the feasibility status of the problem is unknown. In our implementation, we use a merit function approach proposed by Andersen and Ye [1] to determine the step size along the search direction. We evaluate the efficiency of the proposed algorithm by observing its performance on some test problems, which include logarithmic functions, exponential functions and quadratic functions in the constraints. Furthermore, we consider in particular the geometric programming and L_p-programming problems. Numerical results of our algorithm on these classes of optimization problems are reported. We conclude that the algorithm is stable, efficient and easy-to-use in general. As the method allows the user to freely select the initial solution if he/she so wishes, it is natural to take advantage of this and apply the so-called warm-start strategy, whenever the data of a new problem is not too much different from a previously solved problem. This strategy turns out to be effective, according to our numerical experience.

Keywords: Convex programs; conic optimization; self-dual embedding; path-following.

2000 Mathematics Subject Classification codes: 90C25; 90C51; 65K05.

---

* This research was supported by Hong Kong RGC Earmarked Grant CUHK4233/01E.

† Corresponding author. E-mail: zhang@se.cuhk.edu.hk.

1 Introduction

Up till this day, it is generally believed that the primal-dual interior point methods, such as the one introduced by Kojima et al. [11], are among the most efficient methods for solving linear programming problems. The general principle of primal-dual interior point method is based on sequentially solving a certain perturbed Karush-Kuhn-Tucker (KKT) system. In its original form, the primal-dual interior point method requires the availability of some initial primal-dual strongly feasible solutions.

In 1994, Ye, Todd and Mizuno [21] introduced the so-called homogeneous self-dual embedding technique to solve linear programs. The main feature of this technique is to construct an artificial problem by embedding a primal-dual problem pair. When this artificial problem is solved, the original primal and dual problems are automatically solved. There are several nice properties of this method. The most important advantage is that no initial solution is required to start the algorithm: trivial initial solution is available for the artificially constructed self-dual embedded problem. It also solves the problem in polynomial time without resorting to any 'Big-M' type constants, regardless the feasibility status of the original problems.

The underlying principles for the interior-point methodology were further extended to solve general convex optimization problems. The most important work along this line is probably the so-call self-concordant barrier theory developed by Nesterov and Nemirovskii [17]. Based on this theory, the interior-point methods can be applied to efficiently solve several important classes of convex optimization problems, where the self-concordant barrier functions are explicitly known.

The idea of self-dual embedding was also extended to solve more general constrained convex optimization problems. Andersen and Ye [1, 2], developed a special type of self-dual embedding model based on the simplified model of Xu, Hung and Ye [20] for linear programming; Luo, Sturm and Zhang [15] proposed a self-dual embedding model for conic optimization.

The purpose of this paper is to specialize an interior-point implemnetation for a new self-dual embedding method, proposed by Zhang [22], for convex programming problems. Moreover, we study the efficiency of the proposed algorithm for solving numerous test problems.

The outline of this paper is as follows. In Section 2, following Zhang [22], we introduce a particular conic formulation for the inequality-constrained convex program, and construct a self-dual embedding model for solving the resulting conic optimization problem. Then we develop an interior-point algorithm to solve this model in Section 3. In Section 4, numerical tests are conducted to solve randomly generated test problems, which include logarithmic, exponential and quadratic constraints.

We continue the study in Section 5 to test the new algorithm on two classes of well-structured optimization problems: geometric programming and L_p-programming. In Section 6, we discuss the role of the algorithmic parameters. In Section 7, we further experiment our new algorithm with two miscellaneous test problems: non-convex optimization, and problems that are not feasible at all. In Section 8, we explore a warm-start strategy, taking advantage of the fact that our method allows us the freedom to select the initial solution. This strategy can be useful if one needs to solve problems with similar structure/datasets repeatedly. A considerable saving is reported using this strategy. Finally, we conclude our paper in Section 9.

2 Self-dual embedding in conic form

In this section we shall sketch the self-dual embedding method proposed in Zhang [22]. Consider a general convex program as follows:

$$\begin{array}{lll} (P) & \min & c^T x \\ & \text{s.t.} & Ax = b \\ & & f_i(x) \leq 0,\ i = 1, ..., m \end{array}$$

where $c \in \Re^n$, $x \in \Re^n$, $b \in \Re^m$, $A \in \Re^{m \times n}$, and $f_i(x)$ is smooth and convex, $i = 1, ..., m$. As is well known in optimization, it is not losing generality to consider the linear objective only: if the objective is to minimize a nonlinear function $f_0(x)$, then one may introduce a new variable x_{n+1}, which is to be minimized instead, together with a new constraint: $f(x) \leq x_{n+1}$. That way, we reformulate the problem into the desired format.

Let the decision vector be

$$\bar{x} := \begin{bmatrix} p \\ q \\ x \end{bmatrix} \in \Re^1 \times \Re^1 \times \Re^n,$$

and the problem data be

$$\bar{c} := \begin{bmatrix} 0 \\ 0 \\ c \end{bmatrix} \in \Re^1 \times \Re^1 \times \Re^n,\ \bar{b} := \begin{bmatrix} 1 \\ 0 \\ b \end{bmatrix} \in \Re^1 \times \Re^1 \times \Re^m \text{ and } \bar{A} := \begin{bmatrix} 1 & 0 & 0^T \\ 0 & 1 & 0^T \\ 0 & 0 & A \end{bmatrix} \in \Re^{(m+2)\times(n+2)}.$$

We have an equivalent conic formulation for (P) as given by

$$\begin{array}{lll} (KP) & \min & \bar{c}^T \bar{x} \\ & \text{s.t.} & \bar{A}\bar{x} = \bar{b} \\ & & \bar{x} \in \mathcal{K} \end{array}$$

where

$$\mathcal{K} = \bigcap_{i=1}^{m} \mathcal{K}_i,$$

with

$$\mathcal{K}_i = \text{cl } \{\bar{x} \mid \ p > 0, \ q - pf_i(x/p) \geq 0\} \subseteq \Re^{n+2}, \quad i = 1, ..., m.$$

The natural $2m$-logarithmically homogeneous barrier function for $\mathcal{K}$ is

$$F(\bar{x}) = -m \log p - \sum_{i=1}^{m} \log(q - pf_i(x/p)).$$

Related to the duality of convex cones, the conjuate of the convex function $f(x)$ is defined as

$$f^*(s) = \sup\{(-s)^T x - f(x) \mid x \in \text{dom } f\}.$$

Therefore, we obtain the dual of conic problem:

$$\begin{array}{lll} (KD) & \max & \bar{b}^T \bar{y} \\ & \text{s.t.} & \bar{A}^T \bar{y} + \bar{s} = c \\ & & \bar{s} \in \mathcal{K}^* \end{array}$$

where

$$\mathcal{K}^* = \text{cl } (\mathcal{K}_1^* \oplus \cdots \oplus \mathcal{K}_m^*) = \text{cl} \left\{ \sum_{i=1}^{m} \bar{s}_i = \sum_{i=1}^{m} \begin{bmatrix} u_i \\ v_i \\ s_i \end{bmatrix} : \ v_i > 0, u_i - v_i f_i^*(s_i/v_i) \geq 0, i = 1, ..., m \right\}$$

and its $2m$-logarithmically homogeneous barrier function is given as

$$F^*(\bar{s}_1, \cdots, \bar{s}_m) = -\sum_{i=1}^{m} [\log v_i + \log(u_i - v_i f^*(s_i/v_i))].$$

The self-dual embedding model is as follows

$$\begin{array}{lccccccc} \min & & & & \beta\theta & & & \\ \text{s.t.} & & \bar{A}\bar{x} & -\bar{b}\tau & +\bar{r}_p\theta & & = & 0 \\ & -\bar{A}^T\bar{y} & & +\bar{c}\tau & +\bar{r}_d\theta & -\bar{s} & = & 0 \\ & \bar{b}^T\bar{y} & -\bar{c}^T\bar{x} & & +\bar{r}_g\theta & -\kappa & = & 0 \\ & -\bar{r}_p^T\bar{y} & -\bar{r}_d^T\bar{x} & -\bar{r}_g\tau & & & = & -\beta \\ & & \bar{x} \in \mathcal{K}, & \tau \geq 0, & \bar{s} \in \mathcal{K}^*, & \kappa \geq 0 & & \end{array} \tag{1}$$

where $q^0 - p_0 f_i(x^0/p_0) > 0$, $i = 1, .., m$, and $v_i^0 > 0$, $u_i^0 - v_i^0 f_i^*(s_i^0/v_i^0) > 0$, $i = 1, ..., m$, are the initial solutions, and the residuals are defined as

$$\bar{r}_p := \bar{b} - \bar{A}\bar{x}^0, \ \bar{r}_d := \bar{s}^0 - \bar{c} + \bar{A}^T\bar{y}^0, \ \bar{r}_g := 1 + \bar{c}^T\bar{x}^0 - \bar{b}^T\bar{y}^0, \text{ and } \beta := 1 + (\bar{x}^0)^T\bar{s}^0 > 1.$$

We consider the following barrier approach for solving (1) with $\mu > 0$ as the barrier parameter

$$\begin{array}{lccccccc}
\min & \mu F(\bar{x}) & -\mu\log\tau & +\beta\theta & +\mu F^*(\bar{s}_1,\cdots,\bar{s}_m) & -\mu\log\kappa & & \\
\text{s.t.} & \bar{A}\bar{x} & -\bar{b}\tau & +\bar{r}_p\theta & & & = & 0 \\
-\bar{A}^T\bar{y} & & +\bar{c}\tau & +\bar{r}_d\theta & -\sum_{i=1}^m \bar{s}_i & & = & 0 \\
\bar{b}^T\bar{y} & -\bar{c}^T\bar{x} & & +\bar{r}_g\theta & & -\kappa & = & 0 \\
-\bar{r}_p^T\bar{y} & -\bar{r}_d^T\bar{x} & -\bar{r}_g\tau & & & & = & -\beta.
\end{array}$$

Due to the self-duality we derive the following KKT optimality condition, where the solution is denoted by $(\bar{y}(\mu),\bar{x}(\mu),\tau(\mu),\theta(\mu),\bar{s}(\mu),\kappa(\mu))$,

$$\left\{\begin{array}{cccccccc}
 & \bar{A}\bar{x}(\mu) & -\bar{b}\tau(\mu) & +r_p\theta(\mu) & & & = & 0 \\
-\bar{A}^T\bar{y}(\mu) & & +\bar{c}\tau(\mu) & +\bar{r}_d\theta(\mu) & -\bar{s}(\mu) & & = & 0 \\
\bar{b}^T\bar{y}(\mu) & -\bar{c}^T\bar{x}(\mu) & & +\bar{r}_g\theta(\mu) & & -\kappa(\mu) & = & 0 \\
-\bar{r}_p^T\bar{y}(\mu) & -\bar{r}_d^T\bar{x}(\mu) & -\bar{r}_g\tau(\mu) & & & & = & -\beta \\
 & & & & \bar{s}(\mu)-\sum_{i=1}^m \bar{s}_i(\mu) & & = & 0 \\
 & & \tau(\mu)\kappa(\mu) & & & & = & \mu \\
 & & & & \bar{s}(\mu) & & = & -\mu\nabla F(\bar{x}(\mu)).
\end{array}\right. \tag{2}$$

We use the chain rule to calculate $\nabla F(\bar{x})$ and then substitute the expression. After re-arranging the terms we get the following KKT optimality conditions

$$(\star)\left\{\begin{array}{cccccccc}
 & \bar{A}\bar{x}(\mu) & -\bar{b}\tau(\mu) & +r_p\theta(\mu) & & & = & 0 \\
-\bar{A}^T\bar{y}(\mu) & & +\bar{c}\tau(\mu) & +r_d\theta(\mu) & -\bar{s}(\mu) & & = & 0 \\
\bar{b}^T y(\mu) & -\bar{c}x(\mu) & & +r_g\theta(\mu) & & -\kappa(\mu) & = & 0 \\
-r_p^T\bar{y}(\mu) & -r_d^T\bar{x}(\mu) & -r_g\tau(\mu) & & & & = & -\beta \\
 & & \bar{s}(\mu)-\sum_{i=1}^m \bar{s}_i(\mu) & = & 0 & & & \\
 & & \tau(\mu)\kappa(\mu) & = & \mu & & & \\
u_i(\mu)[q(\mu)-p(\mu)f_i(x(\mu)/p(\mu))] & & & = & \mu[q(\mu)/p(\mu)-2f_i(x(\mu)/p(\mu)) & & & \\
 & & & & +\nabla^T f_i(x(\mu)/p(\mu))(x(\mu)/p(\mu))] & & & \\
v_i(\mu)[q(\mu)-p(\mu)f_i(x(\mu)/p(\mu))] & & & = & \mu & & & \\
s_i(\mu)[q(\mu)-p(\mu)f_i(x(\mu)/p(\mu))] & & & = & -\mu\nabla f_i(x(\mu)/p(\mu)) \text{ for } i=1,...,m. & & &
\end{array}\right.$$

3 An implementation of the method

In this section, we shall construct an algorithm to implement the method introduced in the previous section. In particular, we shall first use Newton's method to solve the approximative KKT system.

Then we shall follow the approach of Andersen and Ye [1] to construct a merit function, which is used to select the step length and measure the progress in all iterations.

3.1 The new search direction

As discussed in Section 2, we have KKT system in $(\star)$ for the central solution. Suppose that the current iterates is $(p, x, u, v, s, y, \theta, \tau, \kappa)$. Let $(\Delta p, \Delta x, \Delta u, \Delta v, \Delta s, \Delta y, \Delta\theta, \Delta\tau, \Delta\kappa)$ be the displacements, which should satisfy

$$
(\star\star)\begin{cases}
\left.\begin{array}{lllllll}
 & \bar{A}\Delta\bar{x} & -\bar{b}\Delta\tau & +\bar{r}_p\Delta\theta & & & = 0 \\
-\bar{A}^T\Delta\bar{y} & & +\bar{c}\Delta\tau & +\bar{r}_d\Delta\theta & -\Delta\bar{s} & & = 0 \\
\bar{b}^T\Delta\bar{y} & -\bar{c}^T\Delta\bar{x} & & +\bar{r}_g\Delta\theta & & -\Delta\kappa & = 0 \\
-\bar{r}_p^T\Delta\bar{y} & -\bar{r}_d^T\Delta\bar{x} & -\bar{r}_g\Delta\tau & & & & = 0 \\
 & & & & \Delta\bar{s} & -\sum_{i=1}^m \Delta\bar{s}_i & = 0
\end{array}\right\} & (\triangle_1) \\
(\tau+\Delta\tau)(\kappa+\Delta\kappa) = \mu & (\triangle_2) \\
(u_i+\Delta u_i)(p+\Delta p)\left[(q+\Delta q)-(p+\Delta p)f_i\left(\frac{x+\Delta x}{p+\Delta p}\right)\right] = \mu(q+\Delta q)-2\mu\,(p+\Delta p)\,f_i\left(\frac{x+\Delta x}{p+\Delta p}\right) & \\
\qquad +\mu\nabla f_i\left(\frac{x+\Delta x}{p+\Delta p}\right)^T(x+\Delta x) & (\triangle_3) \\
(v_i+\Delta v_i)\left[(q+\Delta q)-(p+\Delta p)\,f_i\left(\frac{x+\Delta x}{p+\Delta p}\right)\right] = \mu & (\triangle_4) \\
(s_i+\Delta s_i)\left[(q+\Delta q)-(p+\Delta p)\,f_i\left(\frac{x+\Delta x}{p+\Delta p}\right)\right] = -\mu\nabla f_i\left(\frac{x+\Delta x}{p+\Delta p}\right),\ i=1,...,m. & (\triangle_5)
\end{cases}
$$

We shall linearize the system as suggested by the Newton method for solving nonlinear equations. The nonlinear equation $(\triangle_2)$ can be straightforwardly linearized as

$$\kappa\Delta\tau+\tau\Delta\kappa=\mu-\kappa\tau.$$

Consider now the three remaining nonlinear equations. Let us start with $(\triangle_3)$, whose left hand side can be approximated as follows using the Taylor expansion:

$$
\begin{aligned}
& (u_i+\Delta u_i)\,(p+\Delta p)\,[(q+\Delta q)-(p+\Delta p)\,f_i\,((x+\Delta x)/(p+\Delta p))] \\
\cong\ & (u_ip+u_i\Delta p+p\Delta u_i)\left[(q+\Delta q)-(p+\Delta p)\,f_i\left(x/p+\Delta x/p-x\Delta p/p^2\right)\right].
\end{aligned}
$$

We further use the Taylor expansion and drop the higher order terms, and the above quantity becomes approximately

$$\begin{aligned}
& u_i pq + u_i q\Delta p + qp\Delta u_i + u_i p\Delta q - \left(u_i p^2 + u_i p\Delta p + p^2\Delta u_i + u_i p\Delta p\right) f_i\left(x/p\right) \\
& -u_i p^2 \nabla f_i\left(x/p\right)^T \left(\Delta x/p - x\Delta p/p^2\right) \\
\cong\ & u_i pq + \left[u_i q - 2u_i p f_i\left(x/p\right) + u_i \nabla f_i\left(x/p\right)^T x\right]\Delta p + \left[qp - p^2\left(x/p\right)\right]\Delta u_i \\
& + u_i p\Delta q - u_i p\nabla f_i\left(x/p\right)^T \Delta x.
\end{aligned}$$

For the right hand side of $(\triangle_3)$, similarly we have

$$\begin{aligned}
& \mu\left[q + \Delta q - 2(p+\Delta p) f_i((x+\Delta x)/(p+\Delta p)) + \nabla f_i((x+\Delta x)/(p+\Delta p))^T (x+\Delta x)\right] \\
\cong\ & \mu q + \mu\Delta q - 2\mu(p+\Delta p) f_i\left(x/p\right) - 2\mu p\nabla f_i\left(x/p\right)^T \left(\Delta x/p - x\Delta p/p^2\right) \\
& + \mu\nabla f_i\left(x/p\right)^T (x+\Delta x) + \mu\left(\Delta x/p - x\Delta p/p^2\right)^T \nabla^2 f_i\left(x/p\right)(x+\Delta x) \\
\cong\ & \mu q - 2\mu p f_i\left(x/p\right) + \mu\nabla f_i\left(x/p\right)^T x + \left[-\mu\nabla f_i\left(x/p\right)^T + (\mu/p) x^T \nabla^2 f_i\left(x/p\right)\right]\Delta x \\
& - \left[2\mu f_i\left(x/p\right) + 2\mu/p\nabla f_i\left(x/p\right)^T x + \mu/p^2 x^T \nabla^2 f_i\left(x/p\right) x\right]\Delta p.
\end{aligned}$$

Summarizing, Equation $(\triangle_3)$ is now linearized into

$$\begin{aligned}
& \left[u_i q + 2\left(\mu - u_i p\right) f_i\left(x/p\right) + \left(u_i + 2\mu/p\right)\nabla f_i\left(x/p\right)^T x + (\mu/p^2) x^T \nabla^2 f_i\left(x/p\right) x\right]\Delta p \\
& + \left[qp - p^2 f_i\left(x/p\right)\right]\Delta u_i + \left(u_i p - \mu\right)\Delta q + \left[\left(\mu - u_i p\right)\nabla f_i\left(x/p\right)^T - (\mu/p) x^T \nabla^2 f_i\left(x/p\right)\right]\Delta x \\
=\ & \mu q - u_i pq - 2\mu p f_i\left(x/p\right) + \mu\nabla f_i\left(x/p\right)^T x + u_i p^2 f_i\left(x/p\right).
\end{aligned}$$

Consider now Equation $(\triangle_4)$. Again, by the Taylor expansion and dropping high order terms, this yields

$$\begin{aligned}
\mu\ \cong\ & v_i q + v_i\Delta q + q\Delta v_i \\
& -\left(v_i p + v_i\Delta p + p\Delta v_i\right)\left[f_i\left(x/p\right) + \nabla f_i\left(x/p\right)^T \left(\Delta x/p - (x\Delta p)/p^2\right)\right] \\
\cong\ & v_i q - v_i p f_i\left(x/p\right) + v_i\Delta q + \left[q - p f_i\left(x/p\right)\right]\Delta v_i \\
& + \left[v_i/p\nabla f_i\left(x/p\right)^T x - v_i f\left(x/p\right)\right]\Delta p - v_i\nabla f_i\left(x/p\right)^T \Delta x.
\end{aligned}$$

Therefore, we get the Newton equation

$$\begin{aligned}
& \left[q - p f_i\left(x/p\right)\right]\Delta v_i + \left[v_i/p\nabla f_i\left(x/p\right)^T x - v_i f_i\left(x/p\right)\right]\Delta p + v\Delta q - v\nabla f_i\left(x/p\right)^T \Delta x \\
=\ & \mu - v_i q + v_i p f_i\left(x/p\right).
\end{aligned}$$

In a similar fashion, Equation $(\triangle_5)$ can be approximated by

$$\left(s_i + \Delta s_i\right)\left[\left(q + \Delta q\right) - \left(p + \Delta p\right) f_i\left(\left(x + \Delta x\right)/\left(p + \Delta p\right)\right)\right] = -\mu\nabla f_i\left(\left(x + \Delta x\right)/p + \Delta p\right),$$

leading to

$$\begin{aligned} & s_i q + s_i \Delta q + q \Delta s_i - \left(p s_i + s_i \Delta p + p \Delta s_i \right) f_i \left(x/p \right) - p s_i \nabla f_i \left(x/p \right)^T \left(\Delta x/p - (x \Delta p)/p^2 \right) \\ = \; & -\mu \nabla f_i \left(x/p \right) - \mu \nabla^2 f_i \left(x/p \right)^T \left(\Delta x/p - x \Delta p/p^2 \right) \end{aligned}$$

and this further leads to

$$\begin{aligned} & \left[q - p f_i \left(x/p \right) \right] \Delta s_i + \left[-s_i f_i \left(x/p \right) + s_i/p \nabla f_i \left(x/p \right)^T x - \mu/p^2 \nabla^2 f_i \left(x/p \right)^T x \right] \Delta p + s_i \Delta q \\ & + \left[-s_i \nabla f_i \left(x/p \right)^T + \mu/p \nabla^2 f_i \left(x/p \right)^T \right] \Delta x = -\mu \nabla f_i \left(x/p \right) - s_i q + s_i p f_i \left(x/p \right). \end{aligned}$$

Now we shall put the equations in matrix format. Let

$$\begin{aligned} L_1^i &:= pq - p^2 f_i \left(x/p \right) \\ L_2^i &:= u_i q + 2 \left(\mu - u_i p \right) f_i \left(x/p \right) + \left(u_i + 2\mu/p \right) \nabla f_i \left(x/p \right)^T x + \mu/p^2 x^T \nabla^2 f_i \left(x/p \right) x \\ L_3^i &:= \left(\mu - u_i p \right) \nabla f_i \left(x/p \right)^T - (\mu/p) x^T \nabla^2 f_i \left(x/p \right) \\ L_4^i &:= (v_i/p) \nabla f_i \left(x/p \right)^T x - v_i f_i \left(x/p \right) \\ L_5^i &:= -s_i \nabla f_i \left(x/p \right)^T + (\mu/p) \nabla^2 f_i (x/p) \\ L_6^i &:= -s_i f_i \left(x/p \right) + s_i \nabla f_i \left(x/p \right)^T x/p - (\mu/p^2) \nabla^2 f_i \left(x/p \right) x \end{aligned}$$

and

$$\begin{aligned} M_1^i &:= \mu q - u_i p q - 2\mu p f_i \left(x/p \right) + \mu \nabla f_i \left(x/p \right)^T x + u_i p^2 f_i \left(x/p \right) \\ M_2^i &:= \mu - v_i q + v_i p f_i \left(x/p \right) \\ M_3^i &:= -\mu \nabla f_i \left(x/p \right) - s_i q + s_i p f_i \left(x/p \right). \end{aligned}$$

We have

$$D_1^i \Delta \bar{s} + D_2^i \Delta \bar{x} = \bar{M}^i, \text{ for } i = 1, ..., m,$$

where

$$\begin{aligned} D_1^i &:= \begin{pmatrix} p^2 L_1^i & 0 & 0 \\ 0 & L_1^i & 0 \\ 0 & 0 & p L_1^i \end{pmatrix} \\ D_2^i &:= \begin{pmatrix} L_2^i & u_i p - \mu & L_3^i \\ L_4^i & v_i p & v_i p \nabla f_i \left(x/p \right)^T \\ L_6^i & p^2 s_i & L_5^i \end{pmatrix} \\ \bar{M}^i &:= \begin{pmatrix} M_1^i \\ M_2^i \\ M_3^i \end{pmatrix}. \end{aligned}$$

Hence

$$\Delta\bar{s}_i \;=\; (D_1^i)^{-1}(\bar{M}^i - D_2^i\Delta\bar{x}) \text{ for } \; i=1,...,m.$$

Consequently,

$$\sum_{i=1}^{m}\Delta\bar{s}_i \;=\; \sum_{i=1}^{m}(D_1^i)^{-1}(\bar{M}^i - D_2^i\Delta\bar{x}).$$

Since $\sum_{i=1}^{m}\Delta\bar{s}_i = \Delta\bar{s}$, we have

$$\Delta\bar{s} \;=\; \sum_{i=1}^{m}(D_1^i)^{-1}\bar{M}^i - \sum_{i=1}^{m}(D_1^i)^{-1}D_2^i\Delta\bar{x}.$$

Let

$$\begin{aligned}\bar{M} &:= \sum_{i=1}^{m}(D_1^i)^{-1}\bar{M}^i\\ D_2 &:= \sum_{i=1}^{m}(D_1^i)^{-1}D_2^i.\end{aligned}$$

We have a new equation in matrix form

$$\Delta\bar{s} + D_2\Delta\bar{x} = \bar{M}. \tag{3}$$

Note that D_2 is an $(n+2)\times(n+2)$ matrix, which in general may not be invertible. In that case, we shall introduce a perturbation and replace it by $D_2+\varepsilon I$ with $\varepsilon>0$ a small perturbation parameter.

Now we are ready to solve the linearized equations following $(\star\star)$. First, let us denote

$$\begin{aligned}Q_1 &:= -\left(\bar{A}D_2^{-1}D_1\bar{A}^T\right)^{-1}\bar{M}\\ Q_2 &:= -\left(AD_2^{-1}D_1\bar{A}^T\right)^{-1}\left(\bar{b}-\bar{A}D_2^{-1}D_1\bar{c}\right)\\ Q_3 &:= -\left(\bar{A}D_2^{-1}D_1\bar{A}^T\right)^{-1}\left(\bar{r}_p+\bar{A}D_2^{-1}D_1\bar{r}_d\right).\end{aligned}$$

Using $(\triangle_1)$ and (3), we have

$$\Delta\bar{y} \;=\; Q_1 + Q_2\Delta\tau + Q_3\Delta\theta,$$

and furthermore, by letting

$$\begin{aligned}R_1 &:= D_2^{-1}\left(\bar{M}+D_1\bar{A}^TQ_1\right)\\ R_2 &:= D_2^{-1}\left(D_1\bar{A}^TQ_2 - D_1\bar{c}\right)\\ R_3 &:= D_2^{-1}\left(D_1\bar{A}^TQ_3 - D_1\bar{r}_d\right)\end{aligned}$$

we obtain

$$\Delta\bar{x} = R_1 + R_2\Delta\tau + R_3\Delta\theta.$$

The equations can now be explicitly solved. Denoting

$$\begin{aligned} N_1 &:= \frac{-\bar{r}_p^T Q_1 - \bar{r}_d^T R_1}{\bar{r}_g + \bar{r}_p^T Q_2 + \bar{r}_d^T R_2} \\ N_2 &:= \frac{-\bar{r}_p^T Q_2 - \bar{r}_d^T R_3}{\bar{r}_g + \bar{r}_p^T Q_2 + \bar{r}_d^T R_2} \end{aligned}$$

we have

$$\Delta\theta = \frac{\mu - \kappa\tau - \tau\bar{b}^T Q_1 - \tau\bar{b}^T Q_2 N_1 + \tau\bar{c}R_1 + \tau\bar{c}^T R_2 N_1 - \kappa N_1}{\tau\bar{b}^T Q_2 N_2 + \tau\bar{b}^T Q_3 - \tau\bar{c}^T R_2 N_2 - \tau\bar{c}^T R_3 + \kappa N_3 + \tau\bar{r_g}} \tag{4}$$

and

$$\Delta\tau = N_1 + N_2\Delta\theta.$$

All the other variables can be further solved out as follows

$$\begin{aligned} \Delta\tau &= N_1 + N_2\Delta\theta, & (5)\\ \Delta\bar{x} &= R_1 + R_2\Delta\tau + R_3\Delta\theta, & (6)\\ \Delta\bar{y} &= Q_1 + Q_2\Delta\tau + Q_3\Delta\theta, & (7)\\ \Delta\kappa &= \bar{b}^T\Delta\bar{y} - \bar{c}^T\Delta\bar{x} + \bar{r}_g\Delta\theta & (8)\\ \Delta\bar{s} &= -\bar{A}^T\Delta\bar{y} + \bar{c}\Delta\tau + \bar{r}_d\Delta\theta, & (9) \end{aligned}$$

and

$$\Delta\bar{s}_i = (D_1^i)^{-1}(\bar{M}^i - D_2^i\Delta\bar{x}), \tag{10}$$

for $i = 1, ..., m$.

3.2 Select the step-length

After computing the search direction, the iterates may in principle be updated. Next, we would like to discuss how to choose a proper step-size, to be denoted as α hereafter. First, let

$$\Psi_1 := \begin{pmatrix} \tau\kappa \\ \frac{u_1(q-pf_1(x/p))}{q/p-2f_1(x/p)+\nabla^T f_1(x/p)(x/p)} \\ v_1\left(q-pf_1\left(x/p\right)\right) \\ -\left(q-pf_1\left(x/p\right)\right)s_1 \circ \left[\nabla f_1\left(x/p\right)\right]^{-1} \\ \vdots \\ \frac{u_m(q-pf_m(x/p))}{q/p-2f_m(x/p)+\nabla^T f_m(x/p)(x/p)} \\ v_m\left(q-pf_m\left(x/p\right)\right) \\ -\left(q-pf_m\left(x/p\right)\right)s_m \circ \left[\nabla f_m\left(x/p\right)\right]^{-1} \end{pmatrix},$$

and

$$\Psi_2 \quad := \quad \begin{pmatrix} \bar{A}\bar{x} - \bar{b}\tau + \bar{r}_p\theta \\ -\bar{A}\bar{y} + \bar{c}\tau + \bar{r}_d\theta - \sum_{i=1}^m \bar{s}_i \\ \bar{b}^T\bar{y} - \bar{c}^T\bar{x} + \bar{r}_g\theta - \kappa \\ -\bar{r}_p^T\bar{y} - \bar{r}_d^T\bar{x} - \bar{r}_g\tau + \bar{\beta} \end{pmatrix}$$

where "$\circ$" is the Hadamard product of the two vectors, and "$\left[\nabla f\left(x/p\right)\right]^{-1}$" is the component-wise inverse of $\nabla f\left(x/p\right)$. In particular,

$$\nabla f\left(x/p\right)^{-1} = \begin{cases} \frac{1}{\nabla f_i(x/p)}, & \text{if } \nabla f_i(x/p) \neq 0; \\ 0, & \text{if } \nabla f_i(x/p) = 0. \end{cases}$$

It is clear that Ψ_1 is used to measure the duality gap and Ψ_2 is used to measure the feasibility, as can be seen from Equation $(\star)$.

Following Andersen and Ye [1], we introduce the merit function defined as

$$\Phi(y, p, q, x, \tau, \theta, u_1, v_1, s_1, ..., u_m, v_m, s_m, \kappa) := \lambda\Psi_1^T e + \|\Psi_2\|,$$

where $\lambda \in (0, 1)$ is a given parameter.

In addition, the step-size is selected such that all iterates satisfy the following three conditions.

The first condition:

$$\frac{\left(\Psi_1^+\right)^T e}{\left(\Psi_1^0\right)^T e} \geq \beta_1 \frac{\|\Psi_2^+\|}{\|\Psi_2^0\|}, \tag{11}$$

where $\Psi_i^+ = \Psi_i\left(y^+, p^+, q^+, x^+, \tau^+, \theta^+, u_1^+, v_1^+, s_1^+, ..., u_m^+, v_m^+, s_m^+, \kappa^+\right)$ for $i = 1, 2$. This condition prevents the iterates from converging towards a complementarity solution faster than the feasibility measure.

The second condition:

$$\min \Psi_1^+ \geq \beta_2 \frac{\left(\Psi_1^+\right)^T e}{n+3}. \tag{12}$$

This condition prevents the iterates from converging towards the boundary of the positive orthant prematurely.

The third condition:

$$\Phi^+ \leq \Phi + \beta_3 \alpha \nabla \Phi^T \cdot (\Delta y; \Delta p; \Delta q; \Delta x; \Delta \tau; \Delta \theta; \Delta u_1; \Delta v_1; \Delta s_1; ...; \Delta u_m; \Delta v_m; \Delta s_m; \Delta \kappa). \tag{13}$$

The last Armijo-like condition requires that the merit function to be reduced in all iterations.

All the parameters λ, β_1, β_2 and β_3 need to be fine-tuned for each specific problem class, as we shall discuss in Section 6.

We now arrive at a general algorithmic scheme as follows.

Algorithm SEDEAP (Self-Dual Embedding And Path-following)

Step 0. Let

$$(y, p, q, x, \tau, \theta, u_1, v_1, s_1, ..., u_m, v_m, s_m, \kappa) := (y^0, p^0, q^0, x^0, \tau^0, \theta^0, u_1^0, v_1^0, s_1^0, ..., u_m^0, v_m^0, s_m^0, \kappa^0)$$

be the initial solution.

Step 1. If $\Phi(y, p, q, x, \tau, \theta, u_1, v_1, s_1, ..., u_m, v_m, s_m, \kappa) > \epsilon$, then go to **Step 2**. Otherwise stop.

(*Remark*: If the value of the merit function is close to zero, then the iterate is close to the optimal solution.)

Step 2. Let $\mu = \gamma \times \Psi_1(y, p, q, x, \tau, \theta, u_1, v_1, s_1, ..., u_m, v_m, s_m, \kappa)/(n+3)$.

Step 3. Solving the direction

$$(\Delta y, \Delta p, \Delta q, \Delta x, \Delta \tau, \Delta \theta, \Delta u_1, \Delta v_1, \Delta s_1, ..., \Delta u_m, \Delta v_m, \Delta s_m, \Delta \kappa)$$

as given by equations (5)–(10), $i = 1, ..., m$.

Step 4. Find the maximum step length

$$\begin{aligned}\alpha := \text{argmax}\ \{\alpha \geq 0 : \quad & (p;q;x) + \alpha\,(\Delta p;\Delta q;\Delta x) \in \mathcal{K} \\ & (u_i;v_i;s_i) + \alpha\,(\Delta u_i;\Delta v_i;\Delta s_i) \in \mathcal{K}_i^*,\ \text{for } i = 1,...,m\}.\end{aligned}$$

Step 5. Let

$$\begin{aligned}& \left(y^+,p^+,q^+,x^+,\tau^+,\theta^+,u_1^+,v_1^+,s_1^+,...,u_m^+,v_m^+,s_m^+,\kappa^+\right) \\ := \ & (y,p,q,x,\tau,\theta,u_1,v_1,s_1,...,u_m,v_m,s_m,\kappa) \\ & + \alpha \times (\Delta y,\Delta p,\Delta q,\Delta x,\Delta\tau,\Delta\theta,\Delta u_1,\Delta v_1,\Delta s_1,...,\Delta u_m,\Delta v_m,\Delta s_m,\Delta\kappa)\,.\end{aligned}$$

Step 6. If

$$\frac{\left(\Psi_1^+\right)^T e}{\left(\Psi_1^0\right)^T e} < \beta_1 \frac{\|\Psi_2^+\|}{\|\Psi_2^0\|}$$

or

$$\min \Psi_1^+ < \beta_2 \frac{\left(\Psi_1^+\right)^T e}{n+3}$$

or

$$\Phi^+ > \Phi + \beta_3 \alpha \nabla\Phi^T \cdot (\Delta y,\Delta p,\Delta q,\Delta x,\Delta\tau,\Delta\theta,\Delta u_1,\Delta v_1,\Delta s_1,...,\Delta u_m,\Delta v_m,\Delta s_m,\Delta\kappa)$$

then $\alpha := 0.8 \times \alpha$, go to **Step 5**; otherwise go to **Step 7**.

Step 7. Let

$$\begin{aligned}& (y,p,q,x,\tau,\theta,u_1,v_1,s_1,...,u_m,v_m,s_m,\kappa) \\ := \ & \left(y^+,p^+,q^+,x^+,\tau^+,\theta^+,u_1^+,v_1^+,s_1^+,...,u_m^+,v_m^+,s_m^+,\kappa^+\right).\end{aligned}$$

Step 8. Update the value of $\Phi\,(y,p,q,x,\tau,\theta,u_1,v_1,s_1,...,u_m,v_m,s_m,\kappa)$. Go to **Step 1**.

4 Preliminary tests

In this section, we report the computational results of our algorithm, as introduced in Section 3, to solve some very small but typical problems, and also test it on randomly generated test problems. The algorithm is coded in Matlab and the tests are conducted on Ultra 5-333 computer with 333MHz CPU and 128MB RAM running Solaris 2.8. We have not used the parallel capability of the computer in our test; that is, the program is run on one CPU only. In this and the next section, as a convention, for each problem type and parameter set (e.g. the dimension) we run 20 random problem instances,

and the statistics of the tests will be reported. Let us first introduce the notations that we shall use in the tables to follow:

n:	the number of variables;
n_1:	the number of additional variables;
m_1:	the number of inequality constraints;
m_2:	the number of equality constraints;
iter.:	the mean number of iterations;
SD_{iter}:	the standard deviation on the number of iterations;
$iter_N$:	the mean number of normal iterations;
SD_N:	the standard deviation on the number of normal iterations;
$iter_W$:	the mean number of iterations by using warm-start strategy;
SD_W:	the standard deviation on the number of iterations by using warm-start strategy;
error:	the (absolute) error to the optimal objective value;
SD_{error}:	the standard deviation on the error.

The following three problems 4.1 – 4.3 are taken from [4]. The results are meant to illustrate the fact that the selection of the initial solution matters.

Problem 4.1

$$\begin{array}{ll} \min & x_1^2 - 2x_1 + x_2 \\ \text{s.t.} & 3 - x_1 - x_2 \leq 0 \\ & x_1^2 - x_2 + 1 \leq 0 \\ & x_2 - 4 \leq 0 \\ & -x_1 \leq 0 \\ & -x_2 \leq 0. \end{array}$$

The known global solution is $x^* = (1, 2)$.

initial point	*iter.*	*remark*
(8,8)	44	optimal
(5,5)	31	optimal
(1,1)	3	optimal
(0.5,1)	21	optimal

Table 1: Computational results for Problem 4.1

Problem 4.2

$$\begin{aligned} \min \quad & (x_1-2)^2+(x_2-1)^2 \\ \text{s.t.} \quad & x_1+x_2-2\le 0 \\ & x_1^2-x_2\le 0. \end{aligned}$$

The known global solution is $x^*=(1,1)$.

initial point	*iter.*	*remark*
(10,10)	45	optimal
(5,5)	34	optimal
(2,2)	19	optimal

Table 2: Computational results for Problem 4.2

Problem 4.3

$$\begin{aligned} \min \quad & 9-8x_1-6x_2-4x_3+2x_1^2+2x_2^2+x_3^2+2x_1x_2+2x_1x_3 \\ \text{s.t.} \quad & x_1+x_2+2x_3-3\le 0 \\ & -x_1\le 0 \\ & -x_2\le 0 \\ & -x_3\le 0. \end{aligned}$$

The known global solution is $x^*=(\frac{4}{3},\frac{7}{9},\frac{4}{9})$.

initial point	*iter.*	*remark*
(3,3,3)	46	optimal
(-3,3,3)	16	optimal
(1,1,1)	36	optimal
(-1,1,1)	14	optimal

Table 3: Computational results for Problem 4.3

We now wish to test our algorithm to solve problems with more inequality constraints. For this purpose, we generate convex quadratic programs in the following format.

Problem 4.4

$$\begin{aligned} \min \quad & \mathbf{1}^T x \\ \text{s.t.} \quad & Ax=b \\ & x^TQ_ix+\bar{p}_i^Tx+r_i\le 0, \text{ for } i=1,...,m_1, \end{aligned}$$

where $Q_i \succeq 0$, $A \in \Re^{m_2 \times n}$, $b \in \Re^{m_2}$, $\bar{p}_i \in \Re^n$ and $r_i \in \Re$ for all $i = 1, ..., m_1$.

We fix $m_1 = 10$ and $m_1 = 20$ respectively for the experiments. The numerical results are shown in Tables 4 and 5.

n	m_2	*iter.*	SD_{iter}	*error*	SD_{error}
50	13	28.14	5.63	1.34×10^{-5}	5.63×10^{-6}
100	25	22.61	1.28	2.37×10^{-5}	9.21×10^{-6}
150	38	26.70	1.17	9.27×10^{-5}	6.46×10^{-5}
200	50	28.44	3.29	2.44×10^{-5}	9.13×10^{-5}
250	63	28.32	3.06	3.37×10^{-5}	6.12×10^{-6}
300	75	28.69	4.38	7.69×10^{-5}	3.31×10^{-6}
350	88	30.86	2.31	5.67×10^{-5}	8.51×10^{-6}
400	100	26.54	1.92	2.39×10^{-5}	1.13×10^{-6}

Table 4: Numerical results of Problem 4.4 for $m_1 = 10$

n	m_2	*iter.*	SD_{iter}	*error*	SD_{error}
100	25	33.42	3.48	3.72×10^{-5}	9.13×10^{-6}
150	38	29.76	1.81	4.32×10^{-5}	1.56×10^{-6}
200	50	34.32	3.26	8.11×10^{-5}	6.25×10^{-6}
250	63	27.51	3.93	1.34×10^{-5}	5.78×10^{-6}
300	75	30.10	4.23	2.27×10^{-5}	4.61×10^{-6}
350	88	27.85	1.12	7.35×10^{-5}	7.88×10^{-6}
400	100	28.21	5.17	5.26×10^{-5}	1.68×10^{-6}

Table 5: Numerical results of Problem 4.4 for $m_1 = 20$

According to the experimental results, we see that the convex quadratically constrained optimization problems can be solved by our algorithm rather efficiently and stably, meaning that the number of total iterations required is rather small and insensitive to the number of decision variables and the number of inequality constraints.

5 Structured problems

In practice, most optimization problems are not random, but structured, and the structure may lead to quite different type of behaviors as compared with randomly generated problem instances. In this

section we shall consider two special classes of optimization problems: (1) geometric programming; and (2) L_p-programming. We shall test the performance of our method on those special problems.

5.1 Geometric programming

The original (primal) geometric programming problem is given as follows:

Primal Geometric Program:

$$\begin{aligned} \min \quad & g_0(t) \\ \text{s.t.} \quad & g_1(t) \leq 1, \cdots, g_p(t) \leq 1, \\ & t_1 > 0, \cdots, t_m > 0 \end{aligned}$$

where

$$g_k(t) = \sum_{i \in J[k]} c_i t_1^{a_{i1}} t_2^{a_{i2}} \cdots t_m^{a_{im}}$$

for $k = 0, 1, ..., p$, and

$$J[k] = \{m_k, m_k + 1, m_k + 2, ..., n_k\}$$

for $k = 0, 1, ..., p$ and

$$m_0 = 1,\ m_1 = n_0 + 1,\ m_2 = n_1 + 1, ...,\ m_p = n_{p-1} + 1,\ n_p = n.$$

The components a_{ij} can be any real numbers, but the coefficients c_i's are assumed to be positive, i.e., the function $g_k(t)$ are posynomials. The posynomial to be minimized, namely $g_0(t)$, is termed the primal function, and the variables $t_1,\ t_2, ...,\ t_m$ are called primal variables.

The dual program corresponding to the primal problem is the following:

Dual Geometric Program:

$$\max\ v(\delta) = \left[\prod_{i=1}^{n} \left(\frac{c_i}{\delta_i}\right)^{\delta_i}\right] \prod_{k=1}^{p} \lambda_k(\delta)^{\lambda_k(\delta)},$$

where

$$\lambda_k(\delta) = \sum_{i \in J[k]} \delta_i,$$

for $k = 1, 2, ..., p$. The factors c_i are assumed to be positive and the vector $\delta = (\delta_1, ..., \delta_n)$ is subject to the following linear constraints: for $i = 1, ..., n$, and $j = 1, 2, ..., m$

$$\begin{gathered} \delta_i \geq 0, \\ \textstyle\sum_{i \in J[0]} \delta_i = 1, \end{gathered}$$

and

$$\sum_{i=1}^{n} a_{ij} \delta_i = 0.$$

5.1.1 The conic formulation

In general, geometric program is not a convex optimization problem in its original form, but it can be transformed into a convex problem by the change of variables: $x_i = \log t_i$, and so $t_i = e^{x_i},\ i = 1, ..., n$. After that, the geometric program can be expressed in terms of the new variable x in the following equivalent formulation:

$$\begin{array}{ll} \min & \sum_{k=1}^{N_0} e^{a_{0k}^T x - b_{0k}} \\ \text{s.t.} & \sum_{k=1}^{N_i} e^{a_{ik}^T x - b_{ik}} \leq 1,\ i = 1, ..., m. \end{array} \tag{14}$$

The system (14) is readily seen to be equivalent to

$$\begin{array}{ll} \min & e^{x_0} \\ \text{s.t.} & \sum_{k=1}^{N_0} e^{a_{0k}^T x - b_{0k}} \leq e^{x_0} \\ & \sum_{k=1}^{N_i} e^{a_{ik}^T x - b_{ik}} \leq 1,\ i = 1, ..., m, \end{array}$$

where we introduced a new variable x_0 to express the posynomial objective. Noticing that minimizing e^{x_0} amount to minimizing x_0, we can rewrite this last problem in a general form as follows:

$$\begin{array}{lll} (GP) & \min & c^T x \\ & \text{s.t.} & \sum_{k=1}^{N_i} e^{a_{ik}^T x - b_{ik}} \leq 1,\ i = 0, 1, ..., m. \end{array}$$

By adding N variables into the problem, with $N = \sum_{j=0}^{m} N_j$, namely, by introducing

$$x_{n+N_{i-1}+k} := -a_{ik}^T x + b_{ik}$$

for $i = 0, 1, ..., m;\ k = 1, ..., N_i$, and as a convention letting $N_{-1} := 0$, (GP) is equivalently transformed into the following problem

$$\begin{array}{lll} (NGP) & \min & c^T x \\ & \text{s.t.} & Ax = b \\ & & \sum_{k=1}^{N_i} e^{-x_{n+N_{i-1}+k}} - 1 \leq 0,\ i = 0, 1, ..., m. \end{array}$$

Clearly, (NGP) is a convex programming problem, and so Algorithm SEDEAP is expected to perform well.

5.1.2 Computational results of geometric optimization

In this subsection we shall first use an example to illustrate the transformation, and then additional geometric programming problems are generated and solved with the numerical results tabulated.

Consider the following geometric program.

Problem 5.1

$$\begin{array}{ll} \min & t_4 \\ s.t. & 0.1(t_2 t_3^{-1} + t_1) + 0.0005 t_1 t_3 \leq 1 \\ & (0.2423 t_1^{0.5172} t_2^{-0.9957} + 44.8261 t_1^{-0.4828} t_2^{0.0043}) t_4^{-0.5129} \leq 1. \end{array}$$

By putting $t_i = e^{x_i}$ for $i = 1, ..., 4$, we have

$$\begin{array}{ll} \min & e^{x_4} \\ \text{s.t.} & e^{x_2 - x_3 + \ln 0.01} + e^{x_1 + \ln 0.01} + e^{x_1 + x_3 + \ln 0.0005} \leq 1 \\ & e^{0.5172 x_1 - 0.9957 x_2 - 0.5129 x_4 + \ln 0.2423} + e^{-0.4828 x_1 + 0.0043 x_2 - 0.5129 x_4 + \ln 44.8261} \leq 1. \end{array}$$

The above problem is equivalent to the following:

$$\begin{array}{lll} \min & x_4 & \\ \text{s.t.} & x_2 - x_3 + x_5 & = -\ln 0.01 \\ & x_1 + x_6 & = -\ln 0.01 \\ & x_1 + x_3 + x_7 & = -\ln 0.0005 \\ & 0.5172 x_1 - 0.9957 x_2 - 0.5129 x_4 + x_8 & = -\ln 0.2423 \\ & -0.4828 x_1 + 0.0043 x_2 - 0.5129 x_4 + x_9 & = -\ln 44.8261 \\ & e^{-x_5} + e^{-x_6} + e^{-x_7} \leq 1 & \\ & e^{-x_8} + e^{-x_9} \leq 1. & \end{array}$$

We now generate some additional problems in this format to test our new algorithm. In particular, consider:

Problem 5.2

$$\begin{array}{ll} \min & \mathbf{1}^T x \\ \text{s.t.} & Ax = b \\ & \sum_{i=j}^{n+j-m_1} e^{-x_i} - (n - m_1) e \leq 0, \ j = 1, ..., m_1. \end{array}$$

The numerical result for this problem are shown in Tables 6, 7 and 8.

Moreover, we consider the following problem:

Problem 5.3

$$\begin{array}{ll} \min & \mathbf{1}^T x \\ \text{s.t.} & Ax = b \\ & e^{-x_i} + e^{-x_{i+1}} - 2e \leq 0, \ j = 1, ..., n - 1. \end{array}$$

This geometric program has $n - 1$ inequality-constraints. The computational results for this problem are shown in Table 9.

n	*iter.*	SD_{iter}
50	43.31	1.23
100	27.72	1.09
150	31.56	1.23
200	31.27	1.45
250	32.42	2.33
300	30.39	2.13

Table 6: Computational results of the Problem 5.2 for $m_1 = 10$

n	*iter.*	SD_{iter}
50	47.21	2.34
100	36.34	1.21
150	41.27	1.46
200	35.86	1.56
250	42.23	2.03
300	38.42	1.72
350	37.56	1.22
400	37.78	1.33

Table 7: Computational results of Problem 5.2 for $m_1 = 20$

5.2 The L_p-norm optimization

5.2.1 The L_p-norm optimization problem

In this subsection, we consider the problem of minimizing a sum of L_p-norms where p is a fixed real number in the interval $[1, \infty]$. Let $c_i \in \Re^d$ be column vectors and $A_i \in \Re^{n \times d}$ be $n \times d$ matrices, $i = 1, ..., m$. All the matrices A_i's have full column ranks. We want to find a point $u \in \Re^n$ such that the following sum of L_p-norm, $p \geq 1$, is minimized:

$$\begin{array}{ll} \min & \sum_{i=1}^m \|c_i - A_i^T u\|_p \\ \text{s.t.} & u \in \Re^n \end{array}$$

in which $\|\cdot\|_p$ is the L_p-norm defined as

$$\|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{\frac{1}{p}}.$$

This problem has a long history, dating back to the 17th century, when Fermat (see e.g [12]) studied the problem of finding a fourth point on the plane such that the total distances between this point

n	m_1	*iter.*	SD_{iter}
20	2	21.13	2.22
40	4	25.45	2.34
60	6	27.29	1.98
80	8	29.55	2.36
100	10	43.66	2.17
120	12	33.72	2.87
140	14	32.83	1.49

Table 8: Computational results of Problem 5.2

n	*iter.*	SD_{iter}
20	30.22	1.23
40	32.59	2.34
60	38.31	2.12
80	45.12	3.01
100	39.19	2.41
120	37.45	1.33
140	33.62	2.75

Table 9: Computational results of Problem 5.3

and the three existing points is minimized. The Fermat problem was later generalized to an arbitrary number of points with weights, and the problem was known as the Weber problem. The Weber problem was in turn further generalized to allow for any L_p-norms. (See [7] for a survey).

5.2.2 Computational results of L_p-norm optimization problem

Problem 5.4

$$\begin{aligned} \min \quad & c^T \boldsymbol{x} \\ \text{s.t.} \quad & A\boldsymbol{x} = b \\ & \|x^i\|_{p_i} - z_i \leq 0 \\ & \begin{bmatrix} x^i \\ z_i \end{bmatrix}^T \begin{bmatrix} x^i \\ z_i \end{bmatrix} - \frac{2n}{m_1} \leq 0 \end{aligned}$$

for $i = 1, ..., \frac{m_1}{2}$, where

$$\boldsymbol{x} = \begin{bmatrix} z_1 \\ x^1 \\ z_2 \\ x^2 \\ \vdots \\ z_{m_1/2} \\ x^{m_1/2} \end{bmatrix},$$

with $p_i \geq 1$, $A \in \Re^{m_2 \times n}$, $x^i \in \Re^{\frac{2n}{m_1}-1}$, and $z_i \in \Re$, $i = 1, 2, ..., \frac{m_1}{2}$. In the tests, we let $p_i = i$, which varies with the i value.

The numerical results are shown in Table 10.

n	m_2	*iter.*	SD_{iter}
60	15	12.23	2.71
100	25	11.45	1.22
160	40	8.78	3.15
200	50	7.29	1.09
260	65	6.57	2.03
300	75	11.59	3.18

Table 10: Numerical results of Problem 5.4 for $m_1 = 40$

6 Algorithmic parameters

Now we are in the position to talk about the value of algorithmic parameters. All problems are solved by using the value of parameters as shown in Table 11.

First of all, in our implementation, the target shifting rate is set to be $\gamma = 0.8$ throughout. In our experience, the algorithm is stable in this parameter value. It is also important to fine-tune the three step-length determining parameters β_1, β_2 and β_3 in conditions (11) – (13). Based on our experience, the second condition (12) is somehow more difficult to satisfy, and so the value of β_2 is set to be less than or equal to the other two parameters. The values for β_1 and β_3 are set to be constant, for convenience. We set the value of ϵ to be 1×10^{-5} for solving all test problems.

Parameter λ can be quite a sensitive parameter to fine-tune according to our experiments. We need

	ϵ	β_1	β_2	β_3	λ	γ
Problem 4.1	1×10^{-5}	1×10^{-6}	1×10^{-8}	1×10^{-6}	$0.8 \sim 0.9$	0.8
Problem 4.2	1×10^{-5}	1×10^{-6}	1×10^{-8}	1×10^{-6}	$0.8 \sim 0.9$	0.8
Problem 4.3	1×10^{-5}	1×10^{-6}	1×10^{-8}	1×10^{-6}	$0.8 \sim 0.9$	0.8
Problem 4.4	1×10^{-5}	1×10^{-8}	1×10^{-8}	1×10^{-8}	0.8	0.8
Problem 5.2	1×10^{-5}	1×10^{-10}	1×10^{-10}	1×10^{-10}	$0.8 \sim 0.9$	0.8
Problem 5.3	1×10^{-5}	1×10^{-10}	1×10^{-10}	1×10^{-10}	0.7	0.8
Problem 5.4	1×10^{-5}	1×10^{-8}	1×10^{-8}	1×10^{-8}	0.8	0.8

Table 11: Values of algorithmic parameters.

to adjust it to a suitable level for each problem type. For each problem type, we use the same algorithmic parameters to do the experiments for different n. We conclude that a good selection of the values for the parameters may not be dependent on the size of problem of similar structures, but it could be dependent on the structure of the problem. Thus, we may need to adjust the values of the algorithmic parameters for each new problem type.

In Table 11 we see that β_i's are set to be constant for the problems of the same type, but the value of λ may vary slightly to achieve better result. In particular, for Problem 5.3, the value of λ is chosen to be a little lower, due to the structure of this problem. Clearly, Problem 5.3 has $n-1$ inequality-constraints, where n is the number of the decision variables. Therefore, the difficulty for solving this type of problem is higher than that of the other problems discussed in Section 5. We found that if the number of constraints is dependent on the number of decision variables, then we need to decrease the value of λ (emphasizing more on the feasibility). However, the value of λ can also not be set too low. In particular, if $\lambda < 0.3$, then our algorithm does not seem to converge to the optimal solution.

7 Miscellaneous applications

As such, our method is not restricted to solve regular convex programs only. Notwithstanding the risk of non-convergence, it can be applied in principle to solve any nonlinear programming problems posed in the form of function inequalities, even without the knowledge whether a feasible solution exists or not, *a priori*. In this section we shall test Algorithm SEDEAP to solve two classes of such problems: non-convex problems, and infeasible problems.

7.1 Non-convex problems

The following test problems are taken from Kim and Myung [10]. We note that Algorithm SEDEAP is flexible enough to start from any given initial solution. This feature is important for non-convex problem, as we see that the convergence depends on the initial solutions. In the tables, the column '*initial point*' indicates the value of the initial point in terms of (p, q, x^T), where for problems with nonlinear objective, the last component in x is reserved for the artificial variable representing the objective function (see Section 2).

Problem 7.1

$$\begin{array}{ll} \min & 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \\ \text{s.t.} & -x_1 - x_2^2 \leq 0 \\ & -x_1^2 - x_2 \leq 0 \\ & -0.5 \leq x_1 \leq 0.5 \\ & x_2 \leq 1. \end{array}$$

The known global solution is $x^* = (0.5, 0.25)$ and $f(x^*) = 0.25$.

initial point	*iter.*	*remark*
(1,1,1,1,1)	20	stop at a local but not global optimum
(1,1,0.6,0.6,0.6)	18	optimal
(1,1,0.6,0.4,0.3)	14	optimal
(1,0.5,0.5,0.5,0.5)	18	stop at a local but not global optimum
(1,0.5,0.6,0.35,0.35)	15	optimal
(1,1,0.6,0.35,0.35)	15	optimal

Table 12: Computational results for Problem 7.1

Problem 7.2

$$\begin{array}{ll} \min & -x_1 - x_2 \\ \text{s.t.} & -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0 \\ & -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0 \\ & 0 \leq x_1 \leq 3 \\ & 0 \leq x_2 \leq 4. \end{array}$$

The known global solution is $x^* = (2.3295, 3.1783)$ and $f(x^*) = -5.5079$.

initial point	*iter.*	*remark*
(1,1,2.5,3.5)	17	optimal
(1,0.5,2.5,3.5)	20	optimal
(1,0.5,4,4)	—	does not converge
(1,0.5,3,3)	15	stop at a local but not global optimum

Table 13: Computational results for Problem 7.2

Problem 7.3

$$\begin{aligned} \min \quad & (x_1 - 10)^3 + (x_2 - 20)^3 \\ \text{s.t.} \quad & -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0 \\ & (x_1 - 6)^2 + (x_2 - 2)^2 - 82.81 \leq 0 \\ & 13 \leq x_1 \leq 100 \\ & 0 \leq x_2 \leq 100. \end{aligned}$$

The known global solution is $x^* = (14.095, 0.84296)$ and $f(x^*) = -6961.81381$.

initial point	*iter.*	*remark*
(1,7,14.1,0.85,-6960)	12	stop at a local but not global optimum
(1,3,15,1,-6000)	7	stop at a local but not global optimum

Table 14: Computational results for Problem 7.3

Problem 7.4

$$\begin{aligned} \min \quad & 0.01x_1^2 + x_2^2 \\ \text{s.t.} \quad & -x_1 x_2 + 25 \leq 0 \\ & -x_1^2 - x_2^2 + 25 \leq 0 \\ & 2 \leq x_1 \leq 50 \\ & 0 \leq x_2 \leq 50. \end{aligned}$$

The known global solution is $x^* = (\sqrt{250}, \sqrt{2.5}) = (15.8113, 1.5811)$ and $f(x^*) = 5.0$.

Problem 7.5

$$\begin{aligned} \min \quad & (x_1 - 2)^2 + (x_2 - 1)^2 \\ \text{s.t.} \quad & -x_1^2 - x_2 \leq 0 \\ & x_1 + x_2 - 2 \leq 0 \end{aligned}$$

The known global solution is $x^* = (1, 1)$ and $f(x^*) = 1$.

initial point	*iter.*	*remark*
(1,6,15,1.5,6)	14	optimal
(1,1,8,8,8)	24	stop at a local but not global optimum
(1,1,1,1,1)	—	does not converge

Table 15: Computational results for Problem 7.4

initial point	*iter.*	*remark*
(1,0.5,1,1,1)	20	optimal
(1,0.5,1.5,1.5,1.5)	24	optimal
(1,0.5,2,2,2)	19	optimal

Table 16: Computational results for Problem 7.5

We see that most problems (Problems 7.1, 7.2, 7.4 and 7.5) can still be solved, provided that we start from a good initial solution, presumably close to the global optimum. As is common with non-convex problems, the success may not be guaranteed: our algorithm failed to solve Problem 7.3 with two attempts, starting from points that are quite close to the optimum.

7.2 Infeasible problems

By virtue of self-dual embedding, the embedded model itself always has a finite and attainable optimal solution, regardless the status of the original problem. Theoretically speaking, we know that if $\kappa > 0$ and $\tau = 0$, then the original problem is infeasible, and if $\kappa = 0$ and $\tau > 0$, then the original problem is solvable. To test what exactly happens in practice for infeasible problems we experiment with the following problems to show the working of this advantage.

Problem 7.1

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 \\ \text{s.t.} \quad & x_1 + x_2 = 2 \\ & x_2 + x_3 = 2 \\ & x_1^2 + x_2^2 + x_3^2 \leq 1 \end{aligned}$$

Obviously, Problem 7.1 is infeasible, and by applying our algorithm, indeed we get:

$$\kappa = 0.0020,\ \tau = 6.7138 \times 10^{-5} \ \text{ and } \theta = 3.7350 \times 10^{-5},$$

from which we can conclude the infeasibility. Let us consider another test problem by adding one more equality-constraint to Problem 4.3, namely

Problem 7.2

$$\begin{array}{ll} \min & \mathbf{1}^T x \\ \text{s.t.} & Ax = b \\ & \mathbf{1}^T x = -2n \\ & x^T x - n \leq 0 \\ & \sum_{i=1}^m e^{x_i} - ne \leq 0. \end{array}$$

The results of our experiments for different n values are shown in Table 17. In the table, one sees that the τ value reduces significantly faster than the κ value — a clear indication of the infeasibility as predicted by the theory.

n	m_2	*iter.*	κ	τ
20	5	40	0.0544	2.38×10^{-5}
40	10	38	0.5338	8.37×10^{-5}
60	15	34	0.4461	7.24×10^{-5}
80	20	39	0.0020	1.14×10^{-5}
100	25	44	0.3405	8.18×10^{-5}

Table 17: Numerical results for Problem 7.2

8 The warm-start strategy

In this section we shall explore the possibility that comes naturally with the self-dual embedding method for solving (P), i.e., the strategy of using good initial solutions if possible. In particular, suppose that we need to solve a series of problems. The new problem that we encounter, however, resembles in some way a previously solved problem. Then it is quite natural to consider using the previous optimal solution as a starting point of the embedded model for the new problem. The strategy is called *warm start* in this context.

To be specific, let us consider various examples of the warm-start strategies.

8.1 Change the constraint matrix

As a first example, suppose that we solve (P) by our method as described in Section 2 and get the optimal solution x^*. Next we need to solve another problem of the following form

$$\begin{array}{lll} (NPE) & \min & \tilde{c}^T x \\ & \text{s.t.} & \tilde{A}x = \tilde{b} \\ & & f_i(x) \le 0,\ i = 1, ..., m \end{array}$$

where $f_i(x)$ are the same as in (P), $i = 1, ..., m$, and $\|A - \tilde{A}\|$, $\|b - \tilde{b}\|$ and $\|c - \tilde{c}\|$ are small.

We then use the information of the previous solution and let

$$\tilde{x}_0 = \begin{bmatrix} 1 \\ q \\ x^* \end{bmatrix}$$

with $q = \max\{f_i(x^*) \mid i = 1, 2, ..., m\}$, and define

$$\tilde{r_p} = (b - \tilde{A}\tilde{x}_0)/\mu,\ \tilde{r_d} = (\tilde{s}_0 - c + \tilde{A}^T\tilde{y}_0)/\mu,\ \tilde{r_g} = 1 + (c^T\tilde{x}_0 - b^T\tilde{y}_0)/\mu,\ \text{and } \tilde{\beta} = 1 + (\tilde{x}_0^T\tilde{s}_0)/\mu > 1,$$

leading to the following embedded model

$$\begin{array}{lccccccc} \min & & & & \tilde{\beta}\theta & & & \\ \text{s.t.} & & \tilde{A}x & -b\tau & +\tilde{r}_p\theta & & = & 0 \\ & -\tilde{A}^T y & & +c\tau & +\tilde{r}_d\theta & -s & = & 0 \\ & b^T y & -c^T x & & +\tilde{r}_g\theta & -\kappa & = & 0 \\ & -\tilde{r}_p^T y & -\tilde{r}_d^T x & -\tilde{r}_g\tau & & & = & -\tilde{\beta} \\ & & x \in \mathcal{K}, & \tau \ge 0, & & s \in \mathcal{K}^*, & \kappa \ge 0 & \end{array} \tag{15}$$

where the decision variables are $(y, x, \tau, \theta, s, \kappa)$.

In our experiments, we take the following problem (PP) to be the original problem

$$\begin{array}{lll} (PP) & \min & \mathbf{1}^T x \\ & \text{s.t.} & Ax = b \\ & & f_{1k}(x) = x^T Q_k x + \bar{p}_k^T x + r_k \le 0 \\ & & f_{2k}(x) = -\sum_{i=1}^n d_{ik} \log x_i \le 0 \\ & & f_{3k}(x) = \sum_{i=1}^n l_{ik} e^{x_i} - ne \le 0,\ k = 1, 2, ..., m, \end{array}$$

and solve it to optimality. Now we want to solve a new problem as follows

$$\begin{array}{lll} \min & \mathbf{1}^T x & \\ \text{s.t.} & \tilde{A}x = b & \\ & f_{1k}(x) = x^T Q_k x + \bar{p}_k^T x + r_k \le 0 & \\ & f_{2k}(x) = -\sum_{i=1}^n d_{ik} \log x_i \le 0 & \\ & f_{3k}(x) = \sum_{i=1}^n l_{ik} e^{x_i} - ne \le 0,\ k = 1, 2, ..., m, & \end{array} \tag{16}$$

where $\sum_{i=1}^{n} d_{ik} = \sum_{i=1}^{n} l_{ik} = n$, $Q_k \succeq 0$, for all k, and assume that $\|A - \tilde{A}\|$ is small. Now we use the warm-start strategy to solve it. We set $\mu = 10^{-3}$ and $m = 7$ (i.e. $m_1 = 21$). The results are shown in Table 18, where '$iter_N$' column indicates the iteration numbers if the problem is solved as a completely new one without resorting to the warm-start strategy, and '$iter_W$' indicates the iteration numbers if the warm-start strategy is deployed.

n	m_2	$iter_N$	$iter_W$	*saving (%)*	$\|A - \tilde{A}\|$
50	13	38.12	18.97	50.24%	1.9799
100	25	40.85	19.20	53.00%	2.0111
150	38	31.74	14.32	54.88%	2.0972
200	50	33.13	16.21	51.07%	2.1085
250	63	37.56	17.41	53.65%	2.0741
300	75	35.13	15.68	55.37%	2.0834
350	88	37.78	19.68	47.91%	2.0741
400	100	28.46	15.21	46.56%	2.0781

Table 18: Numerical results of warm-start strategy for Problem (16)

According to the numerical results, we can see that the number of iterations is significantly reduced if we use the warm-start strategy.

8.2 Change the functional constraints

Now consider the situation where the inequality constraints in (PP) are changed slightly by adding some convex functions:

$$
\begin{aligned}
(NPF) \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& f_{1i}(x) + \epsilon h_{1i}(x) \le 0 \\
& f_{2i}(x) + \epsilon h_{2i}(x) \le 0 \\
& f_{3i}(x) + \epsilon h_{3i}(x) \le 0, \ i = 1, ..., m
\end{aligned}
$$

where $f_{1i}(x)$, $f_{2i}(x)$ and $f_{3i}(x)$ are the same as in (PP), and $h_i(x)$ are convex function, $i = 1, ..., m$. In our experiments, we let h_i's be quadratic function i.e. $h_i(x) = x^T B_i x + C_i x + D_i$. We set $\epsilon = 5\%$, $m = 7$, $\mu = 10^{-3}$. Table 19 presents the statistical results.

n	$iter_N$	$iter_W$	*saving (%)*	SD_N	SD_W
50	27.89	14.23	48.98%	1.33	2.33
100	31.66	14.91	52.91%	2.45	1.22
150	34.88	18.44	47.13%	3.12	2.11
200	28.97	16.89	41.70%	1.88	2.85
250	25.86	13.76	46.79%	3.44	1.99
300	32.77	16.88	48.49%	2.11	3.81
350	29.15	13.66	53.14%	3.11	2.56
400	30.44	14.77	51.48%	4.33	4.25

Table 19: Statistic results of problem (NPF)

Moreover we may add more functional inequality constraints into the problem, namely

$$\begin{aligned} (NPAF) \quad \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & f_i(x) \leq 0,\ i = 1, ..., m \\ & h_j(x) \leq 0,\ j = 1, ..., r, \end{aligned}$$

where $f_i(x)$ are the same as in (PP) and $h_i(x)$ are new convex function, $i = 1, ..., m$ and $j = 1, ..., r$. We set $m = 7$ (i.e. $m_1 = 21$) and $r = 7$. The numerical results are shown in Table 20.

n	$iter_N$	$iter_W$	*saving (%)*	SD_N	SD_W
50	37.12	24.60	33.73%	2.35	2.59
100	32.24	21.30	33.93%	3.12	2.83
150	30.83	20.20	34.48%	4.55	2.10
200	33.62	22.00	34.56%	1.23	4.74
250	33.71	21.60	35.92%	3.11	4.53
300	33.51	21.70	35.24%	2.77	4.19
350	34.44	23.00	33.22%	4.06	3.46
400	37.62	25.00	33.55%	1.74	4.08

Table 20: Statistic results of problem (NPAF)

8.3 Increase the number of variables

Finally, we wish to test the effect of the warm-start strategy when the number of variables has increased after the previous problem having been solved.

Suppose that the dimension of x is n. We want to add extra n_1 variables into the problem, where $n_1 \leq \frac{n}{4}$. The numerical results are shown in Table 21.

n	n_1	$iter_N$	$iter_W$	*saving (%)*	SD_N	SD_W
75	25	26.12	12.70	51.38%	2.34	1.25
100	25	24.12	11.90	50.66%	1.25	1.10
150	38	23.87	12.10	49.31%	3.01	2.02
200	50	28.15	13.90	50.62%	2.11	3.31
250	63	27.11	14.25	47.44%	1.79	3.40
300	75	30.76	16.32	46.94%	3.74	2.54

Table 21: Statistic results of increase the number of variable

9 Conclusions

In this paper we study the new self-dual embedding method for convex programming proposed by Zhang [22]. According to [22], we can reformulate a general convex optimization problem in conic form by using 2 extra variables. Then we can apply the self-dual embedding technique to solve the resulting conic model. An obvious advantage of the new approach is that it does not require an initial feasible solution of the convex program to start with; instead, any preferred initial solutions can be incorporated, which is a remarkable property of the self-dual embedding method. We then specialize an interior-point algorithm and discuss in detail how this algorithm can be constructed and implemented. We tested the algorithm by considering numerous test problems, including geometric programming and L_p-norm programming problems. We found that the number of iterations is insensitive to the size of the problem. Besides, the best values for the algorithmic parameters are not very much dependent on the size of problem, but can indeed be dependent on the structure of problem. We also experimented the effect of using *warm-start* to take full advantage of the self-dual embedding method. It turns out that in case the problem data does not change too much, one can save considerable amount of effort by using the information about the previously solved problem. Based on the results of this study, we conclude that the new self-dual embedding method is a numerically stable and effective approach for solving general inequality constrained convex programming problems.

References

[1] E.D. Andersen and Y. Ye (1998). A Computional Study of the Homogeneous Algorithm for Large-scale Convex Optmization, *Computational Optimization and Applications*, **10**, 243–269.

[2] E.D. Andersen and Y. Ye (1999). On a Homogeneous Algorithm for Monotone Complementary Problem, *Mathematical Programming*, **84**, 375–399.

[3] C. Beightler and D. Phillips (1976). *Applied Geometric Programming*, John Wiley and Sons, New York.

[4] D. Butnariu and E. Resmerita (2002). Averaged Subgradient Methods for Constrained Convex Optimization and Nash Equilibria Computation, *Opitmization*, **51**, 863–888.

[5] R.S. Dembo (1978). Dual to Primal Conversion of Geometric Programming, *Journal of Optimization Theory and Applications*, **26**.

[6] M. Dow (1999). A Fortran Code for Geometric Programming, Supercomputer Facility Australian National University Canberra Australia, http://anusf.anu.edu.au/ mld900/math/, August 25, 1999.

[7] Z. Drezner, ed., (1995). *Facility Location: A Survey of Applications and Methods*, Springer Series in Operations Research, New York.

[8] F. Glineur (2000), Topics in Convex Optimization: Interior-point Methods, Conic Duality and Approximations. Ph.D. thesis, Faculté Polytechnique de Mons, Belgium.

[9] F. Glineur (2001). Proving Strong Duality for Geometric Optimization using a Conic Formulation, *Annals of Operations Research*, **105**, 155–184.

[10] J.-H. Kim, and H. Myung (1997). Evolutionary Programming Techniques for Constrained Optimization Problems, *IEEE Transactions on Evolutionary Computation*, **1**, 129–140.

[11] M. Kojima, S. Mizuno, and A. Yoshise (1989). A Primal-Dual Interior Point Algorithm for Linear Programming. In N. Megiddo (ed.), *Progress in Mathematical Programming: Interior-Point Algorithm and Related Methods*, 29–47. Springer Verlag, Berlin.

[12] H.W. Kuhn (1973). A note on Fermat's problem, *Mathematical Programming*, **4**, 98–107.

[13] M.L. Lenard and M. Minkoff (1984). Randomly Generated Test Problems for Positive Definite Quadratic Programmin, *ACM Transactions on Mathematical Software*, **10**, 86–96.

[14] Z.-Q. Luo, J.F. Sturm, and S. Zhang (1997). Duality Results for Conic Covex Programming. Technical Report 9719/A, Econometric Institute, Eramus University Rotterdam, The Netherlands.

[15] Z.-Q. Luo, J.F. Sturm, and S. Zhang (2000). Conic Convex Programming and Self-dual Embedding, *Optimization Methods and Software*, **14**, 169–218.

[16] S. Mizuno, M.J. Todd and Y. Ye (1993). On Adaptive-Step Primal-Dual Interior-Point Algorithm for Linear Programming, *Mathematics of Operations Research*, **18**, 964–975.

[17] Yu. Nesterov and A. Nemirovski (1994). *Interior Piont Ploynomial Methods in Convex Programming*, Studies in Applied Mathematics, **13**, SIAM, Philadelphia.

[18] M.J. Rijckaert and X.M. Martens (1978). Comparison of Generalized Geometric Programming, *Journal of Optimization Theory and Apllications*, **26**, 243–245.

[19] J.F. Sturm (2000). *Duality (Chapter 2)*. In H. Frenk, T. Terlaky, K. Roos and S. Zhang (eds.), *High Performance Otimization*, Kluwer Academic Publishers, Dordrecht, The Netherlands.

[20] X. Xu, P.F. Hung, and Y. Ye (1996). A Simplified Homogeneous Self-dual Llinear Programming algorithm and its implementation, *Annals of Operations Research*, **62**, 151–171.

[21] Y. Ye, M.J. Todd and S. Mizuno (1994). An $O(\sqrt{n}L)$-Iteration Homogeneous and Self-Dual Linear Programming Algorithm, *Mathematics of Operations Research*, **19**, 53–67.

[22] S. Zhang (2001). A New Self-Dual Embedding Method for Convex Programming. SEEM Report 2001-09, The Chinese University of Hong Kong, 2001. To appear in *Joural of Global Optimization*.