An interior point algorithm for continuous minimax: Implementation and Computation∗

B. Rustem†‡, S. Žaković‡, P. Parpas‡

August 10, 2007

Abstract

In this paper we propose an algorithm for the constrained continuous minimax problem. The algorithm, motivation and numerical experience are reported in this paper. Theoretical properties, and the convergence of the proposed method are discussed in a separate paper[31]. The algorithm uses quasi–Newton search direction, based on sub–gradient information, conditional on maximizers. The initial problem is transformed to an equivalent equality constrained problem, where the logarithmic barrier function is used to ensure feasibility. In the case of multiple maximizers, the algorithm adopts semi–infinite programming iterations towards epi–convergence. Satisfaction of the equality constraints is ensured by an adaptive quadratic penalty function. The algorithm is augmented by a discrete minimax procedure to compute the semi-infinite programming steps and ensure overall progress when required by the adaptive penalty procedure. Progress towards the solution is maintained using merit functions. Computational results are included to illustrate the efficient performance of the algorithm.

1 Formulation of the Problem

Consider the following problem:

$$\min_{x} \max_{y \in Y} \{ f(x, y) \mid g(x) = 0, \ x \geq 0 \},$$

(1)

where $Y$ is a compact subset of $\mathbb{R}^m$, $x \in \mathbb{R}^n$, $f(x, y)$ is continuous in $x$ and $y$, twice continuously differentiable in $x$, and $g : \mathbb{R}^n \rightarrow \mathbb{R}^k$ is continuous and twice differentiable in $x$. We also denote the feasible region with $X_f = \{ x \in \mathbb{R}^n \mid g(x) = 0, \ x \geq 0 \}$.

When the maximizer $y$ is defined on a discrete set, (1) becomes a discrete minimax problem and algorithms for solving such problems have been considered by a number of authors, including Womersley and Fletcher [34], Polak, Mayne and Higgins [22] and Rustem and Nguyen [29], Obasanjo and Rustem [19].

∗Financial support of the EPSRC grants GR/R51377/01 and EP/C513584/1 are gratefully acknowledged. The authors are also grateful to Professor Elijah Polak for valuable suggestions and advice on this paper. Any remaining errors are the responsibility of the authors.

†Corresponding author. E-mail: br@doc.ic.ac.uk, tel: +44-20-7594-8345.

‡Department of Computing, Imperial College, London SW7 2AZ.
The continuous problem (1) arises in numerous disciplines, including \( n \)-person games [28], finance [30], economics [38], policy optimization [3]. In general, they are used by the decision maker to assess the worst-case strategy of the opponent and compute the optimal response. The opponent can also be interpreted as nature choosing the worst-case value of the uncertainty, and the solution would be the strategy which ensures the optimal response to the worst-case. Neither the robust decision maker nor the opponent would benefit by deviating unilaterally from this strategy. The solution of (1) can be characterized as a saddle point when \( f \) is convex in \( x \) and \( f \) is concave in \( y \). A survey of algorithms for computing saddle points can be found in [7], [30].

We consider an algorithm in which \( f(x, y) \) is not necessarily convex in \( x \) and concave in \( y \). Previous attempts to solve this problem have mostly focused on unconstrained problems (Kiwiel [15], Dentyanov and Pevnyi [7], Rustem and Howe [30]), with notable exceptions such as Breton and El Achem [5] who use projected sub-gradient and bundle methods. An earlier attempt to solve a similar problem, with decoupled constraints (i.e. constraints that only contain either \( x \) or \( y \) ), was made by Sasai [32]. Among recent work is that of Polak et al. [24], where a smoothing technique (using exponential penalty functions) is applied to a finite (discrete) minimax problem. This work has been extended in [23] where the smoothing technique is applied to solving finite and semi-infinite min-max-min problems.

The approach in this paper uses an interior point methodology and an adaptive penalty merit function to encourage feasibility and descent. It differs from the above in the general framework as well as the choice of the barrier and merit functions. It uses semi-infinite programming steps that: (i) involve the solution of a discrete minimax problem; (ii) are initiated to complement the interior point algorithm descent procedure and to prevent the penalty parameter increasing indefinitely; (iii) have an additional role in the epi-convergent procedure to glean information about maximizers and construct an effective algorithm robust to the existence of single or multiple maximizers.

The algorithm is thus intended to combine the desirable aspects of semi-infinite programming with descent steps for the max function based on subgradients and a merit function. Semi-infinite programming performs well when the set of maximizers \( Y(x_k) \) at some \( x_k \), is stable in that the maximizers do not change at every iteration. Additionally, in this case, multiple maximizers are used to generate subgradients. Helpful descent steps can then be performed provided that sufficient number of maximizers are known, so that the merit function can be reduced further. If \( Y(x_k) \) changes radically at every iteration, as in the case when \( f(x, y) \) is concave in \( y \) and there usually is a different unique maximizer corresponding to every \( x \), a gradient-based descent direction can perform significantly better. As also mentioned in [42], there are clearly computational difficulties regarding a step size strategy to be overcome and these are discussed in Section 5. However, semi-infinite programming steps and a sub-gradient based approach can complement each other rather well in accumulating maximizer information and in expediting progress through descent directions respectively.

The semi-infinite programming formulation (1) is given by:

\[
\min_{x \in X, \tau} \{ \tau \mid f(x, y) - \tau \leq 0, \forall y \in Y \} \tag{2}
\]
which has an infinite number of constraints corresponding to the elements in \( Y \).

We compare our results with an algorithm based on semi–infinite programming ([4], [12], and [39]). Other recent contributions in this area are by Lawrence and Tits [17] and Stein and Still [33] where sequential quadratic programming and interior point techniques are considered for generalized semi–infinite problems.

One strategy for problems with an infinite number of constraints is through discretization. Under this framework, the original problem is replaced by a problem with a finite number of constraints. There are a number of discretization schemes, exchange methods, or methods based on local reduction [14]. Polak in [21] discusses discretization and finer consistent approximations of the original semi–infinite problem through epi–graphs and epi–convergence that we also introduce later in this paper. Convergence is established through a sequence of the solutions to the discretized problems, assuming that such sequences have an accumulation point. The present paper augments the discretization scheme by a procedure to generate a descent direction and a merit function to stabilize progress towards the solution. This allows a direct convergence result for the algorithm.

Let \( x_i \) denote the \( i^{th} \) element of vector \( x \). The penalty formulation to ensure feasibility regarding the inequality constraints on the slack variable is realized using a barrier function such as \(-\log(x_i)\). The framework for solving the problem is closely related to the interior point literature (e.g. [1], [9]). The transformed minimax problem, for \( x > 0 \), is given by:

\[
\min_x \max_{y \in Y} \left\{ f(x, y) - \mu \sum_{i=1}^{n} \log(x_i) \mid g(x) = 0 \right\}. \tag{3}
\]

We define the augmented objective function as:

\[
P(x, y; c, \mu) = f(x, y) + \frac{c}{2} \left\| g(x) \right\|_2^2 - \mu \sum_{i=1}^{n} \log(x_i), \tag{4}
\]

and the merit function \( \Psi \) as the maximum of the augmented objective (4):

\[
\Psi(x; c, \mu) = \max_{y \in Y} P(x, y; c, \mu). \tag{5}
\]

The algorithm discussed below solves (1) and is based on a sequence of optimization problems characterized by penalty parameter \( c \geq 0 \) and barrier parameter \( \mu \geq 0 \). The max function \( \Phi(x) \) is defined as follows:

\[
\Phi(x) = \max_{y \in Y} f(x, y), \tag{6}
\]

and the following set of maximizers at \( x \):

\[
\hat{Y}(x) = \{ y \in Y \mid f(x, y) = \Phi(x) \}.
\]

Given the maximizers \( y \in \hat{Y}(x) \) and using the fact that the maximum over a set of scalars is equal to the maximum over their convex combination, (3) can be equivalently expressed as:

\[
\min_x \max_{\beta \in B} \left\{ \sum_{y \in \hat{Y}(x)} \beta^y f(x, y) - \mu \sum_{i=1}^{n} \log(x_i) \mid g(x) = 0 \right\},
\]

where \( B \) is the set of all non-negative \( \beta \) such that \( \sum_{y \in Y} \beta^y = 1 \).
where $\beta^y$ is the element of $\beta$ corresponding to maximizer $y$ and
\[
B = \{ \beta \mid \sum_y \beta^y = 1, \beta^y \geq 0 \}.
\] (7)

A similar reasoning can be applied to the merit function $\Psi(x; c, \mu)$:
\[
\Psi(x; c, \mu) = \sum_{y \in \hat{Y}(x)} \beta^y P(x, y; c, \mu).
\]

We summarize the three definitions related to the objective function:
\[
\begin{align*}
\Phi(x) &= \max_{y \in Y} f(x, y) \\
P(x, y; c, \mu) &= f(x, y) + \frac{c}{2} \|g(x)\|^2 - \mu \sum_{i=1}^n \log(x^i) \\
\Psi(x; c, \mu) &= \max_{y \in Y} P(x, y; c, \mu).
\end{align*}
\]

2 Sub–gradient of $\Phi(x)$ and Basic Iteration

Problem (6) poses several difficulties: Firstly, $\Phi(x)$ is in general continuous but may have kinks, so it might not be differentiable. At a kink the maximizer is not unique and the choice of sub–gradient to generate a search direction is not simple. Secondly, $\Phi(x)$ may not be computed accurately as it would require infinitely many iterations of an algorithm to maximize $f(x, y)$. Finally, in (5) global maxima are required in view of possible multiple solutions. The use of local, or insufficient number of global, maxima cannot ensure a monotonic decrease in $\Phi(x)$.

The sub–differential of the $\max$ function $\Phi(x)$ can be expressed as:
\[
\partial \Phi(x) = \text{conv} \{ \nabla_x f(x, y) \mid y \in \hat{Y}(x) \}.
\]

Similarly, the sub–differential of the merit function:
\[
\partial \Psi(x; c, \mu) = \text{conv} \{ \nabla_x P(x, y; c, \mu) \mid y \in \hat{Y}(x) \}.
\]

For non–unique maximizers $y \in \hat{Y}(x)$, by Caratheodory’s theorem [26]: for every vector $\nabla \Psi(x; c, \mu) \in \partial \Psi(x; c, \mu)$ there exist $n + 1$ points $y_j \in \hat{Y}(x), j = 1, \ldots, n + 1$, and multipliers $\beta_j, j = 1, \ldots, n + 1$, with $\sum_{j=1}^{n+1} \beta_j = 1, \beta_j \geq 0$, such that:
\[
\nabla \Psi(x; c, \mu) = \sum_{y \in \hat{Y}(x), \beta \in B} \beta^y \nabla_x f(x, y) + \frac{c}{2} \nabla g(x)^T g(x) - \mu X^{-1} e,
\]
where $X^{-1}$ is the diagonal matrix, defined as $X^{-1} = \text{diag}(\frac{1}{x_1}, \frac{1}{x_2}, \ldots, \frac{1}{x_n})$. Similarly, we have
\[
\nabla \Phi(x) = \max_{y \in Y(x), \beta \in B} \sum_{y \in \hat{Y}(x), \beta \in B} \beta^y \nabla_x f(x, y).
\]

The determination of $\beta^y$ is discussed in the next section. A problem which the set $\hat{Y}(x)$ poses is that it can be of infinite cardinality. For example, the problem
\[
\min_{x \in \mathbb{R}} \max_{-2 \leq y \leq 2} (x - 1)^2 y
\]
Table 1: Maximizer concepts

<table>
<thead>
<tr>
<th>Set</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y(x)$</td>
<td>$Y(x) = { y \in Y</td>
</tr>
<tr>
<td>$Y_k$</td>
<td>set of potential maximizers accumulated from $x_0$ to $x_k$</td>
</tr>
<tr>
<td>$Y(x_k)$</td>
<td>$Y(x_k) = { y \in Y_k</td>
</tr>
</tbody>
</table>

has an infinite number of solutions, as every pair $(x^*, y^*) = (1, y)$ for $y \in \hat{Y} = \{ y | -2 \leq y \leq 2 \}$ is a solution. To avoid such problems we use the discretization described below.

Instead of using the whole set of maximizers at $x$, $\hat{Y}(x)$, starting with $y_0 \in \arg \max_{y \in Y} f(x_0, y)$, we define the sets $Y_i \subset Y$ so that:

- $Y_0 = \{ y_0 \}$; $y_i \in \arg \max_{y \in Y} f(x_i, y)$; $Y_i = Y_{i-1} \cup \{ y_i \}, \ i = 1, 2, ...$

Now, it is possible to define a finite set of maximizers at the current point $x_k$: $Y(x_k) = \{ y \in Y_k | f(x_k, y) = \Phi(x_k) \}$. Hence, we note that $Y(x_k) \subset Y_k$. Table 1 summarizes the various maximizer concepts introduced in this paper. The cardinality of $Y(x_k)$ is denoted by $| Y(x_k) |$. For any compact set $Y \subset Y$ let $\Phi_Y(x) \equiv \max_{y \in Y} f(x, y)$. Then for any $x \in \mathbb{R}^n$ we have $\Phi_Y(x) \leq \Phi(x) = \Phi_Y(x)$. The original problem (1) is approximated with a finite number of maximizers:

$$ \min_{x \in X_f} \Phi_{Y_i}(x), \ i = 0, 1, 2, ... $$

To clarify the relationship between the original problem (1) and the approximation (9), we restate these using the following epigraphs:

$$ E = \{(x^0, x) | x \in X_f, \ x^0 \geq \Phi(x) \} $$

$$ E_i = \{(x^0, x) | x \in X_f, \ x^0 \geq \Phi_{Y_i}(x) \}. $$

Given the epigraphs $E$ and $E_i$ in (10), the problems (1) and (9) can be reformulated as follows:

$$ \min_{(x^0, x) \in E} x^0, $$

$$ \min_{(x^0, x) \in E_i} x^0. $$

It can be seen, from (11) and (12), that the two problems differ only in the constraint set. For the sequence of problems (12) to be approximate (11) well, $E_i$ must converge to $E$.

**Definition** The sequence of approximating problems (12) epi-converges to the problem (11) if the epigraphs $E_i$ converge to the epigraph $E$ in the following sense

$$ \lim E_i = \overline{\lim E_i} = E. $$

Clearly, approximate problems in optimization are necessary in order to facilitate computation. The practical importance of epi–convergence stems from the fact that an approximation that epi–converges to the original problem will also
(under some conditions) have the same minima. Epi-convergence and its use in
algorithm design is discussed in [21] and [27]. In the algorithm described in this
paper, epigraphs \( E_i \) are generated by:

\[
x_{i+1} \in \arg\min_{x \in X} \Phi(Y_i(x));
\]

\[
y_{i+1} \in \arg\max_{y \in Y} f(x_{i+1}, y),
\]

\[
Y_{i+1} = Y_i \cup \{y_{i+1}\};
\]

\[
E_{i+1} = \{(x^0, x) | x \in X, x^0 \geq \Phi(Y_{i+1})\}.
\]

The difficulties associated with \( \Phi(x) \) and possible large numbers of maximizers,
are addressed through a number of methods based on the characterization of
maximizers of \( f(x, y) \). Another related difficulty is the potentially costly eval-
uation of \( \Phi(x) \) during line search, required to ensure global convergence. A
class of algorithms, including Demyanov and Malozemov [8], Hager and Presler
[13] and Zhou and Tits [42] are based on approximating \( \Phi(x) \) by progressively
increasing the number of discretization points. The algorithm presented in this
paper invokes Carathéodory’s theorem [26], to characterize the search direction
with a small number of maximizers that would ensure converge
ce. Rustem and Howe [30] consider an algorithm that defines a direction of progress for \( \Phi(x) \)
based on the maximizers at \( x \) corresponding to the minimum norm sub– gradient
of \( \nabla_x f(x, \cdot) \) and a Hessian approximation. The algorithm extends the first or-
der order approach of Panin [20] and Kiwiel [15] to quasi–Newton descent directions,
conditional on the maximizer and also attempts to deal with the problem of mul-
tiple maximizers. The algorithm presented in [30] is in some respects similar
to the algorithm presented here. In [30] it assumed that sufficient maximizers
are available to compute a descent direction. In this paper we complement this
approach with semi–infinite steps when a descent direction cannot be easily
computed.

The Lagrangian associated with (3) is

\[
L(x, y, \lambda; \mu) = \max_{y \in Y} \{f(x, y) - \mu \sum_{i=1}^{n} \log(x^i) - \lambda^t g(x)\}
\]

Consider the diagonal matrix \( X = \text{diag}(x_1, x_2, ..., x_n) \) and the column vector
\( e = [1, 1, ..., 1]^t \in \mathbb{R}^n \). The first order optimality condition for the problem (3)
are given by:

\[
\nabla \Phi(x) - \mu X^{-1}e - \nabla g^t(x)\lambda = 0; \ g(x) = 0,
\]

where \( \nabla_x \Phi(x) \) is a sub–gradient of \( \Phi(x) \) and its evaluation is considered in the
next section. Optimality conditions for this class of problems are discussed in
[30]. Invoking the nonlinear transformation \( z = \mu X^{-1}e \) yields

\[
F = [\nabla \Phi(x) - z - \nabla g^t(x)\lambda, \ g(x), \ XZe - \mu e]^T = 0,
\]

where \( F = F(x, \lambda, z; \mu) \) and \( Z \) is the diagonal matrix \( Z = \text{diag}(z_1, z_2, ..., z_k) \).
These equations represent the perturbed optimality conditions for problem (1).
We note that \( F = F(x, \lambda, z; 0), x \geq 0, z \geq 0 \) represent the optimality conditions.
Returning to the original problem with equality and non-negativity constraints,
(1), we define the Lagrangian

\[
L_e(x, z, \lambda) = \Phi(x) - z^t x - \lambda^t g(x),
\]

\[
6
\]
where \( z \in \{ v \in \mathbb{R}^n | v \geq 0 \} \) is the multiplier vector corresponding to \( x \geq 0 \). The Lagrangian (14) is defined over a set of maximizers. Its subgradient is defined as the convex combination of the gradients corresponding to the set of maximizers. A consistent Hessian definition, mentioned below, is based on the convex combination of the Hessians corresponding to the maximizers\(^1\).

The primal-dual interior point algorithm is essentially the Newton or quasi-Newton method for solving approximately the perturbed conditions (13) for a fixed value of \( \mu \). The Newton system for (13) above is

\[
\begin{bmatrix}
H_k & -\nabla g_k^t & -I \\
\nabla g_k & 0 & 0 \\
Z_k & 0 & X_k
\end{bmatrix}
\begin{bmatrix}
\Delta x_k \\
\Delta \lambda_k \\
\Delta z_k
\end{bmatrix}
= -\begin{bmatrix}
\nabla \Phi(x_k) - z_k - \nabla g_k^t \lambda_k \\
g_k \\
X_k Z_k e - \mu e
\end{bmatrix},
\]

(15)

where \( H_k \) is a positive definite approximation of the Hessian of the Lagrangian (14). The Hessian \( H_k \) is approximated using the updating formula suggested by Powell [25]. Using matrix-vector notation, (15) can be expressed as \( \nabla F_k \Delta w_k = -F_k \) where \( w_k = (x_k, \lambda_k, z_k) \), \( \Delta w_k = (\Delta x_k, \Delta \lambda_k, \Delta z_k) \) and \( F_k = F(x_k, \lambda_k, z_k; \mu) \).

The solution of (15) is given by:

\[
\begin{align*}
\Delta x_k &= \Omega_k^{-1} \nabla g_k \Delta \lambda_k - \Omega_k^{-1} h_k \\
\Delta \lambda_k &= -[\nabla g_k \Omega_k^{-1} \nabla g_k^t]^{-1} (g_k - \nabla g_k \Omega_k^{-1} h_k) \\
\Delta z_k &= -z_k + \mu X_k^{-1} e - X_k^{-1} Z_k \Delta x_k,
\end{align*}
\]

(16)

where \( \Omega_k = H_k + X_k^{-1} Z_k \), and \( h_k = \nabla \Phi(x_k) - \mu X_k^{-1} e - \nabla g_k^t \lambda_k \). Introducing two new matrices \( M_k \) and \( \mathcal{P}_k \) given by: \( M_k = \nabla g_k \Omega_k^{-1} \nabla g_k^t \), \( \mathcal{P}_k = (I - \Omega_k^{-1} \nabla g_k^t M_k^{-1} \nabla g_k) \). Then the first two equations of the system (16) can be written as:

\[
\begin{align*}
\Delta x_k &= -\mathcal{P}_k \Omega_k^{-1} (\nabla \Phi(x_k) - \mu X_k^{-1} e) - \Omega_k^{-1} \nabla g_k^t M_k^{-1} g_k \\
\Delta \lambda_k &= -M_k^{-1} (g_k - \nabla g_k \Omega_k^{-1} h_k).
\end{align*}
\]

(17)

Starting from an initial point \( w_0 \), the algorithm generates a sequence \( \{w_k\} \):

\[
w_{k+1} = w_k + \alpha_k \Delta w_k, \quad k = 0, 1, 2, ...
\]

In order to maintain feasibility of \( w_{k+1} \), the algorithm needs to ensure \( x_{k+1}, z_{k+1} > 0 \). In [40], [41] a single step size is used for all variables. In [36], different step size strategies are suggested. The algorithm in this paper uses a merit function with an adaptive penalty parameter and \( \alpha_k = (\alpha_{x_k}, \alpha_{z_k}, \alpha_{z_k}) \) where \( \alpha_{x_k} \) and \( \alpha_{z_k} \) are different step sizes for the primal \( x \) variables and dual pair \( \lambda, z \) respectively.

The algorithm generates a descent direction based on a sub–gradient of \( \Phi(x) \) and an approximate Hessian, in the presence of possible multiple maximizers of (5) or (6). It uses a switching scheme between a continuous minimax based interior point algorithm incorporating a minimum-norm sub–gradient and a discrete minimax formulation appropriately incorporating epigraphs to determine potential multiple maximizers. This is discussed further in Section 4 (see (25)). Each approach uses a different step size strategy to ensure sufficient progress.

The overall iterative process is in two stages: first, (3) is solved for \( \mu \) fixed. This is the inner iteration. Once (3) is solved, \( \mu \) is reduced, convergence criteria is checked (outer iteration) and, if necessary, another inner iteration is performed.

\(^1\)The algorithm utilizes a positive definite approximation to this Hessian.
3 Computing Sub–gradients of \( \Phi(x) \)

\( \Psi(x; c, \mu) \) is in general continuous but may have kinks, so it might not be differentiable. At a kink the maximizer is not unique and the choice of sub–gradient to generate a search direction is not simple. The quasi–Newton direction in this paper is based on the combination of the gradients corresponding to the multiple maximizers that ensures descent. The quadratic approximation to \( P(x, y; c, \mu) \) in (4) at \( x_k \) is given by:

\[
P_k(\Delta x_k, y; c, \mu) = P(x_k, y; c, \mu) + \Delta x^t \nabla_x P(x_k, y; c, \mu) + \frac{1}{2} \| \Delta x_k \|_{H_k}^2. \tag{18}
\]

We consider two approaches for determining \( \beta \). The first is motivated by the selection of the worst–case descent direction based on a combination of possible maximizers \( y \in Y(x_k) \). In the presence of multiple maximizers the new direction \( \Delta w_k(\beta_k) \) is computed such that:

\[
\beta_k = \arg \max_{\beta \in B} \{ \Delta x_k(\beta)^t (\sum_{y \in Y(x_k)} \beta^y \nabla_x f(x_k, y) + c_k \nabla g_k^t g_k - \mu X_k^{-1} e) + \frac{1}{2} \| \Delta x_k(\beta) \|_{H_k}^2 \}, \tag{19}
\]

which is a concave maximization problem in \( \beta \). The direction \( \Delta x_k(\beta_k) \), which is conditional on the given \( \beta_k \), is given by

\[
\Delta x_k(\beta_k) = -P_k \Omega_k^{-1}(\sum_{y \in Y(x_k)} \beta^y \nabla_x f(x_k, y) - \mu X_k^{-1} e) - \Omega_k^{-1} \nabla g_k M_k^{-1} g_k, \tag{20}
\]

where \( F^\beta_k \) is defined as

\[
F^\beta_k = \left[ \sum_{y \in Y(x_k)} \beta^y \nabla_x f(x_k, y) - z_k - \nabla g_k \lambda_k, g_k, X_k Z_k e - \mu e \right]^T. \tag{21}
\]

The solution to problem (19) is unique when the vectors: \( \nabla_x f(x_k, y), \quad y \in Y(x_k) \) are linearly independent. Otherwise, a minimum norm \( \beta_k \) is determined.

The second approach to determining \( \beta \) is given by:

\[
\beta_k^y = 1 \text{ for some } y = y_{k+1} \in Y(x_k) \quad \beta_k^y = 0, \forall y \neq y_{k+1}, y \in Y(x_k).
\]

Thus, \( \Delta x_k(1) \) corresponds to one element of \( \beta_k \) being unity and the rest null. The choice of the corresponding maximizer is:

\[
\hat{y}_{k+1} \in \arg \max_{y \in Y(x_k)} \{ \Delta x_k(1)^t (\nabla f(x_k, y) + c \nabla g_k^t g_k - \mu X_k^{-1} e) \} + \frac{1}{2} \| \Delta x_k(1) \|_{H_k}^2 \}. \tag{22}
\]

The motivation for this choice is the selection of the worst-case descent direction among the maximizers. Hence, for such \( \hat{y}_{k+1} \) we have \( F^\beta_k = 1 \):

\[
F^\beta_{k+1} = [\nabla_x f(x_k, \hat{y}_{k+1}) - z_k - \nabla g_k \lambda_k, g_k, X_k Z_k e - \mu e]^T.
\]
Consider, therefore, two possible directions $\Delta w(1)$ and $\Delta w(\beta_k)$, depending on the sub-gradient $\nabla \Phi(x)$ used. The direction $\Delta w(1)$ is easier to compute, as it does not entail the solution of the quadratic programming problem (19). We define two characterizations of $\nabla P(x_k,\ldots,c_k,\mu)$. One corresponds to the maximizer $y_{k+1}$ responsible for the worst-case descent direction and the other corresponds to multiple maximizers, $y \in Y(x_k)$. These are given by

$$\nabla P(x_k,y_{k+1};c_k,\mu) = \nabla_x f(x_k,y_{k+1}) + \frac{c_k}{2} \nabla g(x_k) y(x_k) - \mu X_k^{-1} e$$

$$\sum_{y \in Y(x_k)} \beta_k^{\mu} \nabla P(x_k,y;c_k,\mu) = \sum_{y \in Y(x_k)} \beta_k^{\mu} \nabla_x f(x_k,y) + \frac{c_k}{2} \nabla g(x_k) y(x_k) - \mu X_k^{-1} e.$$ 

These two characterizations of $\nabla P(x_k,\ldots,c_k,\mu)$ lead to two possible subgradient choices for $\nabla \Phi(x_k)$ and $\nabla \Psi(x_k;c_k,\mu)$. These are summarized in Table 2 together with the corresponding $\Delta w_k$. In the rest of this paper we ignore the argument $\beta$ and use $\Delta w_k$, except when distinguishing between $\Delta w_k(1)$ and $\Delta w_k(\beta_k)$. The new maximizer $y_{k+1}$ is chosen as a maximizer of the following augmented quadratic approximation to $\Psi(x,c,\mu)$:

$$y_{k+1} \in \arg \max_{y \in Y} \{ P_k(\Delta x_k(1),y;c,\mu) - C[\Phi(x_k) - f(x_k,y)]^2 \}.$$ 

The problem above can be interpreted as follows: among all the maximizers we wish to choose the one that is also a maximizer $Y(x_k)$. We formulate this problem as a penalty function to enforce the choice of $y$ to a maximiser at $x_k$. If $C \geq 0$ is large enough, then we can ensure that $\Phi - f = 0$.

### 4 Primal-dual Step Size Selection

To determine the new iterate $x_{k+1}$ we adopt Armijo’s rule [2] to ensure sufficient decrease in the merit function. The maximum allowable step size is determined by the boundary of the feasible region. The pseudo code for choosing the sub-gradient of $\Phi(x)$, the penalty parameter $c$ and the merit function is presented in Algorithm 3 below.

Step size determination for the dual variables $z$ uses the information provided by the new primal iterate $x_{k+1}$, in order to find the new iterate $z_{k+1}$. This is a modification of the strategy suggested by Yamashita [35] and Yamashita and Yabe [36] and Akrotirianakis and Rustem [1].

While the barrier parameter $\mu$ is fixed, we determine a step $\alpha_{z_k}^i$ along the direction $\Delta z_k$, for each dual variable $z_k^i$, $i = 1,2,\ldots,n$, such that the box constraints are satisfied

$$\alpha_{z_k}^i = \max \{ \alpha > 0 : LB_k \leq (x_k^i + \alpha x_k \Delta x_k^i)(z_k^i + \alpha \Delta z_k^i) \leq UB_k \}. \quad (23)$$

<table>
<thead>
<tr>
<th>$\nabla_x \Phi(x_k)$</th>
<th>$\Delta w_k$</th>
<th>$\nabla_x \Psi(x_k;c_k,\mu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nabla_x f(x_k,y_{k+1})$</td>
<td>$\Delta w_k(1) = -(\nabla c_k</td>
<td>_{c_k=1}^{y=1} - 1 F_k^{(1)})$</td>
</tr>
<tr>
<td>$\sum_y \beta_k^{\mu} \nabla_x f(x_k,y)$</td>
<td>$\Delta w_k(\beta_k) = -(\nabla c_k</td>
<td>_{c_k=1}^{y=1} - 1 F_k^{(1)})$</td>
</tr>
</tbody>
</table>

Table 2: Choices for $\nabla_x \Phi(x_k)$, $\nabla_x \Psi(x_k;c_k,\mu)$ with $y \in Y(x_k)$

9
The lower bounds $LB_i^k$ and upper bounds $UB_i^k$, $i = 1, 2, \ldots, n$ are defined as

$$LB_i^k = \min \left\{ \frac{1}{2} m \mu, (x_i^k + \alpha x_k \Delta x_i^k) z_k^i \right\}, \quad UB_i^k = \max \left\{ 2 M \mu, (x_i^k + \alpha x_k \Delta x_i^k) z_k^i \right\},$$

where the parameters $m$ and $M$ are chosen such that

$$0 < m \leq \min \left\{ 1, \left( 1 - \gamma \right) \left( 1 - \frac{\gamma}{(M_0 \mu)} \right) \chi_l \right\}, \quad M \geq \max \left\{ 1, \frac{\chi_l}{\mu} \right\} > 0, \quad (24)$$

where $\chi_l, \hat{\chi}_l$ are set in Algorithm 3, $\gamma \in (0, 1)$ and $M_0$ is a positive large number. These two parameters are always fixed to constants which satisfy (24) while $\mu$ is fixed. The values of $m$ and $M$ change when the barrier parameter $\mu$ is decreased.

The common dual step length $\alpha z_k$ is the minimum of all individual step lengths $\alpha_{z_k}$ with the restriction of being always not more than one. The step size for the dual variables $y$ can be either $\alpha y_k = 1$ or $\alpha y_k = \alpha z_k$.

Merit function (5) penalizes infeasibility and measures progress toward the optimal solution. There are, however, two issues regarding (5). Firstly, penalty parameter $c$ might grow unacceptably, potentially leading to numerical instability. Secondly, the algorithm may not have the required number of maximizers to generate a descent direction.

The two problems are detected through a single observation based on the descent property of the direction $\Delta x(\beta)$. If $\Delta x(\beta)$ cannot ensure descent it is necessary to either increase $c$ or generate further maximizers. The latter is achieved by performing individual steps of semi–infinite programming. This requires the solution of the following discrete minimax problem, at the $k-th$ iteration, defined as:

$$\min_{x \in X_f} \max_{y \in Y(x_k)} \{ f(x, y) \}. \quad (25)$$

5 Choice of $c$ and $\mu$

The set $\{(x(\mu), \lambda(\mu), z(\mu)) : \mu > 0\}$, which contains solutions of the perturbed optimality conditions (13) for different values of the barrier parameter, is called the central path. Since the perturbed optimality conditions approximate the optimality conditions of the original problem as $\mu \rightarrow 0$, the points in the central path converge to a solution of the initial problem. The central path can thus be thought of as the route, followed by primal-dual interior point algorithms, towards the solution of a nonlinear optimization problem. The distance of the current point from the central path is measured by the Euclidean norm of the perturbed optimality conditions.

The barrier parameter $\mu$ plays an important role in interior point methods. By decreasing the parameter to zero, the algorithm converges to the optimal solution. Again, there are a number of different strategies for reducing the barrier parameter $[9], [11]$. We adopt the strategy originally developed for nonlinear programming in $[1, 11, 16]$. This barrier selection rule has performed very effectively in $[1]$, and is presented in Algorithm 1 below.

Algorithm 1: Barrier Parameter Update Rule

**STEP 0:** Initialize $x = x_k, \lambda = \lambda_k, z = z_k, c = c_k, \mu = \mu^l, r_1 = 0.85
It has been shown in [9, 11] that the direction $\Delta w$ defined by (16) is a descent direction for $\|F(x, \lambda; z; c, \mu)\|_2^2$ as long as the matrix $\nabla F_k$ is nonsingular.

The penalty parameter $c$ plays an important role in the descent of the merit function $\Psi(x; c, \mu)$. At each iteration, its value is determined such that a descent property is ensured. This is simple when all the maximizers are known at the current point, since Caratheodory’s theorem can be used to select an appropriate subset of maximizers to ensure descent. If sufficient number of maximizers are not known, an epi-convergent procedure is initiated with individual steps of semi–infinite programming adding new maximizers.

The sub–gradient of $\Psi(x_k; c, \mu)$ at the $k$-th iteration is

$$\nabla \Psi(x_k; c_k, \mu) = \nabla \Phi(x_k) + c_k \nabla g_k^TG_k - \mu X_k^{-1} e.$$  

The direction $\Delta x_k$ is a descent direction for $\Psi$, at the current point $x_k$, if

$$\langle \nabla \Phi(x_k) + c_k \nabla g_k^TG_k - \mu X_k^{-1} e, \Delta x_k \rangle + \frac{1}{2}\|\Delta x_k\|_{l_k}^2 \leq 0. (26)$$

By considering the second equation of the Newton system (15), the directional derivative $\Delta x_k \nabla \Psi(x_k; c_k, \mu)$ can be written as

$$\Delta x_k \nabla \Psi(x_k; c_k, \mu) = \Delta x_k \nabla \Phi(x_k) - c_k \|g_k\|^2 - \mu \Delta x_k X_k^{-1} e, (27)$$

where $c_k$ is the value of the penalty parameter at the beginning of the $k$-th iteration. Since the barrier parameter $\mu$ is fixed throughout the inner iterations, we can deduce from (27) that the sign of $\Delta x_k \nabla \Psi(x_k; c_k, \mu)$ depends on the value of the penalty parameter. If $c_k$ is not large enough then the descent property (26) may not be satisfied. Thus, a new value $c_{k+1} > c_k$ must be determined to guarantee the satisfaction of the descent property.

The descent of the merit function depends not only on the value of $c$ but also on the maximizers $y \in Y(x)$. In general, the presence of multiple maximizers requires for the sub–gradient of the max function to be computed as in (19), in order to ensure the descent. In [30] this is simplified to a direction dependent only on one $y_{k+1}$ as long as it ensures descent. In this paper, when only one maximizer is known descent is assured with an appropriate choice of $c$. In the case of multiple potential maximizers, the direction based on one maximizer only is used as long as it ensures descent without an increase in $c$. This avoids unnecessarily computation of $\beta$ in (19). When a single maximizer cannot ensure descent, problem (19) is solved and a new sub–gradient, that depends on all known maximizers, is computed. If this new direction is still not descent, the value of $c$ is increased. Thus, the direction $\Delta w_k$ is given by

$$\Delta w_k = \left\{ \begin{array}{ll} -(\nabla F_k^{\beta_1})^{-1} F_k^{\beta_1} if n_{max} = 1 or \Delta x_k(1)^{\nabla \Psi(x_k, y_{k+1}; c_k, \mu) \leq 0} \\
-(\nabla F_k^{\beta_1})^{-1} F_k^{\beta_1}, otherwise. \end{array} \right.$$

11
We note that $\Delta x_k(1)$ corresponds to the direction for $\beta = 1$. When this direction does not lead to descent and there are other known potential maximizers, the algorithm proceeds along the direction $-(\nabla F^\beta_k)^{-1} F^\beta_k$.

The choice for $c$ and $\nabla \Phi(x)$ is shown in Algorithm 2 below. The merit function is based on (5) and ensures progress towards a minimum. We note that the algorithm may exhibit the Maratos effect [18] with this merit function. Checks on the growth of $c$ are incorporated to address this problem.

Let $\epsilon_g > 0$ be a finite precision to which the equality constraints are satisfied. Thus, for merit function (5), we have a worst case feasibility precision

$$\|g(x)\|^2 > \epsilon_g.$$  

(28)

Assume that at, some inner iteration $k$, $0 < \|g(x_k)\|^2 \leq \epsilon_g$ and the descent condition (26) is not satisfied. In this case, (25) is initiated as an epi–convergent procedure to generate further maximizers. This is the semi-infinite programming step discussed in Step 2 of Algorithm 2 below. If a further maximizer is thereby identified, Algorithm 2 returns to minimizing merit function (5). As mentioned earlier, this is a variation of the “watchdog” technique [6]. In the context of interior point methods it was also used by Gay et al. in [11].

In Step 2 of Algorithm 2, when $\Delta x_k$ is not a descent direction for the merit function (5) and $0 < \|g_k\|^2 < \epsilon_g$, then $x_{k+1}$ and $v$ are given by the solution of (25). The value $v$ is used in Step 2(a) of Algorithm 2. At $x_{k+1}$, the new maximizer is computed as

$$\hat{y}_{k+1} = \arg \max_{y \in Y} f(x_{k+1}, y).$$  

(29)

The algorithm terminates if

$$f(x_{k+1}, \hat{y}_{k+1}) \leq v,$$  

(30)

otherwise the new maximizer is added to the set of maximizers

$$Y_{k+1} = Y_k \cup \hat{y}_{k+1}$$  

(31)

and a new iteration is performed. Where $v$ is the optimal value of the problem given by (25).

The descent condition (26) for merit function (5) may not always be satisfied for the following reasons: Firstly, the direction $-(\nabla F^\beta_k)^{-1} F^\beta_k$ is computed using only one maximizer, but there are other known potential maximizers. This is addressed by computing the direction $-(\nabla F^\beta_k)^{-1} F^\beta_k$ and restarting the iteration without altering the penalty parameter $c$. Secondly, the direction $-(\nabla F^\beta_k)^{-1} F^\beta_k$ and the corresponding $\Delta x_k(\beta)$ are justified in view of the multiple maximizers employed but the penalty parameter $c$ is not sufficiently large to ensure descent. An increase in $c$ is required. Thirdly, insufficient number of maximizers are included in the computation of the direction $-(\nabla F^\beta_k)^{-1} F^\beta_k$.

A semi-infinite programming step is employed in Step 2(a) below, by solving a discrete minimax problem. The corresponding maximizer is added the potential maximizer set $Y_{k+1}$ and the new iteration is started without increasing $c$. Even when a new maximizer has just been added to $Y_{k+1}$, there is the possibility that the number of known maximizers are not sufficient to generate a descent direction. To address this possibility, the algorithm searches for a step...
size that ensures satisfaction of the Armijo inequality in Step 2c below until a relative tolerance is reached. Step sizes less than this value are disallowed and the algorithm explores the possibility of further maxima using semi-infinite programming steps. The algorithm may alternate between semi-infinite programming and descent steps. However, as the set of potential maximizers $Y_k$ grows, the epigraph of the max function $\Psi(x_k; c_*, \mu)$ is characterized with increasing accuracy and the maximizers generate descent steps. At that stage, Theorem 1 in [31], ensures monotonic decrease of $\Psi(x_{k+1}; c_*, \mu)$.

Semi-infinite programming ensures that

$$\min_{x} \max_{y \in Y(x_k)} \{ P(x, y; c, \mu) \mid g(x) = 0 \} \leq \max_{y \in Y} \{ P(x_k, y; c, \mu) \mid g(x) = 0 \} = \Psi(x_k; c_*, \mu).$$

Given $x_{k+1}, v, \hat{y}_{k+1}$ computed using (29), if (30) is satisfied, then $x_{k+1}$ is the solution of the inner iteration (Algorithm 2), otherwise, using (31), the algorithm proceeds further with the multiple maximizers in $Y_{k+1}$. In such cases, convergence is due to compactness with semi-infinite programming aided further by the decrease in $\Psi(x_k; c_*, \mu)$ ensured by subsequent to accumulation of sufficient number of maximizers. In cases where sufficient number of maximizers are attained $\forall k \geq k_*$, for some $k_*$, the monotonic decrease of the sequence \{\Psi(x_k; c_*, \mu)\} is established in Theorem 1 in the second part of this paper.

A detailed description of the algorithm for solving the inner iteration problem (the barrier parameter $\mu$ is fixed) is given below.

Algorithm 2: Inner Iteration

**STEP 0:** Initialization Set $\beta = 1$, $\Delta x_k(1)$ is used when there is a single maximizer or descent is assured even in the presence of multiple maximizers.

**STEP 1(a):** If $\|F(x_k, y_k, \lambda_k, z_k; \mu)\|_2 \leq \eta \mu$ exit to Algorithm 3.

*Compute the descent direction if $\beta = 1$ then*

$$y_{k+1} = \arg\max_{y \in Y(x_k)} \{ \Delta x_k(1) \nabla_x P(x_k, y; c_k, \mu) + \frac{1}{2} \| \Delta x_k(1) \|_{H_k} \}$$

$$\Delta x_k = \Delta x_k(1) \text{ and } \nabla \Phi(x_k) = \nabla f(x_k, y_{k+1}) \text{ go to STEP 1(c)}$$

**STEP 1(b):** If a single maximizer is not sufficient for progress, compute $\Delta x_k(\beta)$

$$\beta_k = \arg\max_{\beta \in B} \{ \Delta x_k(\beta)^T \sum_{y \in Y(x_k)} \beta^y \nabla_x P(x_k, y; c_k, \mu) + \frac{1}{2} \| \Delta x_k(\beta) \|_{H_k} \}$$

This implies the values $\nabla \Phi(x_k) = \beta_k^T \nabla f(x_k, y)$ and $\Delta x_k = \Delta x_k(\beta_k)$ which are actually computed in **STEP 1(c)**

**STEP 1(c):** Interior point step

$$\Delta w_k(\beta_k) = - (\nabla F_k)^{-1} F_k^{\beta_k} \quad \alpha_{x k}^{\max} = \min_{1 \leq i \leq n} \{ \frac{x_i}{\Delta x_k^i} : \Delta x_k^i < 0 \}$$

$$\hat{\alpha}_{x_k} = \min \{ \gamma \alpha_{x k}^{\max}, 1 \}.$$
STEP 2(a)(i): Test for descent of the merit function \( \mathcal{M}_{\text{num}} = \Delta x_k^i \nabla \Phi(x_k) - c_k \) if \( g_k \| \frac{\Delta x_k^i}{2} - \mu \Delta x_k^i X_k^{-1} e + \| \Delta x_k \|_H_k \| > 0 \) then If descent condition is not satisfied, and \( c \) cannot be increased due to small \( g_k \|_2^2 \), generate new maximizer and \( x_{k+1} \) using semi-infinite programming step.

STEP 2(a)(ii): Semi-infinite programming step

\[
x_{k+1} = \arg \min_{x \in X_k} \max_{y \in Y(x_k)} \{ f(x, y) \}, \quad v = \min_{x \in X_k} \max_{y \in Y(x_k)} \{ f(x, y) \}
\]

\[\hat{y}_{k+1} = \arg \max_{y \in Y} \{ f(x_{k+1}, y) \}\]

If \( f(x_{k+1}, \hat{y}_{k+1}) \leq v \) STOP: the additional maximizer(s) \( \hat{y}_{k+1} \) do not improve the current function value so \( x_{k+1} \) is the minimax solution.

go to STEP 3

STEP 2(b):

if \( \mathcal{M}_{\text{num}} \leq 0 \) then
descent assured and \( c_k \) remains unchanged otherwise no decrease with \( \Delta x_k = \Delta x_k(1) \) and \( n_{\text{max}} > 1 \), a new direction \( \Delta x_k(\beta) \) needs to be computed if \( (n_{\text{max}} \neq 1) \) and \( (\beta = 1) \) then goto STEP 1(b) otherwise increase penalty parameter \( c_k \):

\[
c_{k+1} = \max \{ \frac{\Delta x_k^i \nabla \Phi(x_k) - \mu \Delta x_k^i X_k^{-1} e + \| \Delta x_k \|_H_k}{\| g_k \|_2^2}, \ c_k + \delta \}.
\]

STEP 2(c): Compute \( w_{k+1} \)

\[
\alpha_{x_k} = \theta^i \hat{a}_k, \quad \text{where } i = \min \{0,1,2,...\} \text{ such that:}
\]

\[
\Psi(x_k + \alpha_{x_k} \Delta x_k; c_{k+1}, \mu) - \Psi(x_k; c_{k+1}, \mu) \leq \rho \alpha_{x_k} \nabla x \Psi(x_k; c_{k+1}, \mu)^T \Delta x_k
\]

If \( \| x_k + \alpha_{x_k} \Delta x_k \| \| x_k \| \leq \varepsilon_{\text{tol}} \) then go to STEP 2(a)(i)

\[
LB^i_k = \min \{ \frac{1}{2} m \mu, (x_k^i + \alpha_{x_k} \Delta x_k^i) z_k^i \}, \quad UB^i_k = \max \{ 2 M \mu, (x_k^i + \alpha_{x_k} \Delta x_k^i) z_k^i \}
\]

\[
\alpha^i_k = \max \{ \alpha > 0 : LB^i_k \leq (x_k^i + \alpha_{x_k} \Delta x_k^i) (z_k^i + \alpha \Delta z_k^i) \leq UB^i_k \}
\]

\[
\alpha_{z_k} = \min \{ 1, \min_{1 \leq i \leq n} \{ \alpha^i_k \} \}, \quad \alpha_{\lambda k} = \alpha_{z_k} \quad \alpha_k = (\alpha_{x_k}, \alpha_{z_k}, \alpha_{z_k})^T
\]

\[
w_{k+1} = w_k + \alpha_k \Delta w_k, \quad \hat{y}_{k+1} = \arg \max_{y \in Y} \{ f(x_{k+1}, y) \}
\]

If \( \hat{y}_{k+1} \in Y_k \), \( k = k + 1 \), go to STEP 2(a)(ii)

STEP 3: Update the set of potential maximizers \( Y_{k+1} = Y_k \cup \hat{y}_{k+1}, k = k + 1 \), go to STEP 1(a).

When updating the penalty parameter \( c \) the following is computed:

\[
\mathcal{M}_{\text{num}} = \Delta x_k^i \nabla \Phi(x_k) - c_k \| g_k \|_2^2 - \mu \Delta x_k^i X_k^{-1} e + \| \Delta x_k \|^2_\mathcal{H}_k.
\]

The algorithm moves from one inner iteration to another (with \( \mu \) fixed) by minimizing the merit function \( \Psi(x; c, \mu) \) which is achieved by appropriately selecting the penalty parameter \( c \) at each inner iteration. The monotonic decrease of (5) and the rules for determining the primal and dual step sizes ensure that the inner iterates converge to the solution of (3) for fixed value of \( \mu \). By reducing \( \mu \) and \( \{ \mu \} \rightarrow 0 \) the optimum of the initial problem (1) is obtained.

In the implementation of the algorithm, the following criterion is used for inner iteration: \( \| F(x_k, \lambda_k, z_k; \mu) \|_2 \leq \eta \mu \). As long as the criteria is not satisfied, \( \mu \) stays fixed and new \( w_k \) is computed.
The interior point algorithm consists of inner iterations (Algorithm 2) and outer iterations (Algorithm 3 below). As can be verified, the main work is done in the inner loop, where problem (3) is solved, with $\mu$ being fixed.

**Algorithm 3: Outer Iteration**

**STEP 0:** Initialization:
Choose $x^0, \lambda^0, z^0, (\eta, \mu^0, \sigma, \delta, \varepsilon_0, \varepsilon_{tol}) > 0, M$ and $m$ given by (24) for $(\theta, \gamma) \in (0,1), \rho \in (0, \frac{1}{2})$;
Compute $y_0 = \arg \max_{y \in Y} \{f(x^0, y)\}$; set $l = 0, k = 0, Y_0 = \{y_0\}$.

**STEP 1:** Termination check for overall optimality:
if $\|F(x^l, \lambda^l, z^l; \mu)\|_2/(1 + \|x^l, \lambda^l, z^l\|_2) \leq \varepsilon_0$ stop
set $(x_k, \lambda_k, z_k) = (x^l, \lambda^l, z^l), \chi_l = \min_i \{x_i^l z_i^l\}, \hat{\chi}_l = \max_i \{x_i^l z_i^l\}$

**STEP 3:** Execute inner iteration - Algorithm 2

**STEP 4:** Update $\mu$ - Algorithm 1

**STEP 5:** Set $(x^{l+1}, \lambda^{l+1}, z^{l+1}) = (x_k, \lambda_k, z_k); l = l + 1, \text{Go to STEP 1}.$

The overall structure of the interior point algorithm is given in Figure 1.

---

**6 Computational Results**

In this section we report experience with numerical examples. The algorithm defined in the previous sections have been implemented using Fortran 90 on an IBM compatible Personal Computer, using the Intel Pentium IV 3GHz processor. The following ten examples are based on those in [30]. First five are convex–concave functions (i.e. $f(x, y)$ convex in $x$ and concave in $y$) and then
Table 3: Additional constraints for problems in [30]

<table>
<thead>
<tr>
<th>Problem no.</th>
<th>Constraint(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( x_1^2 + x_2^2 \leq 0.3 )</td>
</tr>
<tr>
<td>2</td>
<td>( x_1^2 + x_2^2 \leq 2.5 )</td>
</tr>
<tr>
<td>3</td>
<td>(-x_1^2 + x_2^2 \leq 1.75)</td>
</tr>
<tr>
<td>4</td>
<td>( x_1^2 \leq 0.15, \ x_2 \leq 0.1 )</td>
</tr>
<tr>
<td>5</td>
<td>( x_1^2 + x_2^2 \leq 0.03, \ x_2^2 + x_3^2 \leq 0.06, \ x_3^2 + x_1^2 \leq 0.05 )</td>
</tr>
<tr>
<td>6</td>
<td>( x_1 \geq 0.2, \ x_3 + x_4 \geq 0.8 )</td>
</tr>
<tr>
<td>7</td>
<td>( x_1 \geq 0.1, \ x_3 + x_4 \leq 0.4 )</td>
</tr>
<tr>
<td>8</td>
<td>( x_2 \geq 0.1, \ x_3 + x_4 \leq 1.45 )</td>
</tr>
<tr>
<td>9</td>
<td>( x_1 + x_2 \geq 0.1, \ x_3 + x_4 \leq 1.7 )</td>
</tr>
<tr>
<td>10</td>
<td>( x_1^2 + x_3^2 \leq 1.5, \ x_3 + x_4 \leq 0.1 )</td>
</tr>
</tbody>
</table>

The further five examples are convex–convex functions, where multiple maximizers are present, and we examine the behavior of the algorithm in presence of such points. Problems 1-6 correspond to problems 1-6 in [30]. While problems 7-10 correspond to problems 13-16. Problems in [30] are unconstrained in the \( x \) variables. Therefore, we added some constraints to these problems. These are given in Table 3.

The various parameters used in the overall algorithm are as follows. In Algorithm 1, the barrier reduction rule, we set \( \eta = 0.1 \) and \( \sigma = 6 \). In Algorithm 2, in Step 1(c), we set \( \gamma = 0.995 \), in Step 2(a) \( \epsilon_g = 10^{-12} \) and in Step 2(b) \( \delta = 10 \). In Step 2(c) \( \theta = 0.5, \ \rho = 10^{-4}, \ \epsilon_{tol} = 10^{-12}, \ m = 1, \) and \( M = 10 \). The accuracy of the stopping criterion (Algorithm 3) is \( \epsilon_0 = 10^{-8} \). The dimensionality of the problems in the \( x \) variables is indicated in the Dim-\( x \) column of tables 4 and 5; similarly for the \( y \) variables.

The results are obtained using the minimax algorithm presented in this paper and the semi–infinite algorithm (SIP) presented in [39].

6.1 \( f(x, y) \) locally convex in \( x \), concave in \( y \)

The examples in this section illustrate the algorithm when \( f(x, y) \) has a unique maximizer for \( x \) fixed. In Table 4, results in terms of CPU time and number of iterations are presented. For the SIP case, we present the number of Nonlinear Programs solved (in brackets) together with the total number of iterations. The minimax algorithm performs better in terms of CPU time and needs less iterations to converge, compared to the SIP algorithm discussed in [39].

6.2 \( f(x, y) \) locally convex in \( x \), locally convex in \( y \)

The five examples below involve \( f(x, y) \) locally convex in \( x \) and \( y \), so the maximizers of \( f(x, y) \) lie at the boundary the feasible region \( Y \). In general, such problems have multiple maximizers. We present five examples and each example is run twice – with the minimax algorithm and the semi-infinite programming algorithm in [39]. The CPU time and number of iterations for each of the problems 6–10 are presented in Table 5.
Table 4: Numerical results (1–5)

<table>
<thead>
<tr>
<th>Problem no.</th>
<th>Minimax</th>
<th>SIP</th>
<th>Dim-x</th>
<th>Dim-y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CPU</td>
<td>0.094</td>
<td>1.472</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>iterations</td>
<td>11</td>
<td>350(8)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 CPU</td>
<td>0.141</td>
<td>0.329</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>iterations</td>
<td>16</td>
<td>91(3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 CPU</td>
<td>1.250</td>
<td>3.063</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>iterations</td>
<td>71</td>
<td>307(12)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 CPU</td>
<td>0.109</td>
<td>0.203</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>iterations</td>
<td>11</td>
<td>46(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 CPU</td>
<td>0.266</td>
<td>0.469</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>iterations</td>
<td>18</td>
<td>78(4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Numerical results (6–10)

<table>
<thead>
<tr>
<th>Problem no.</th>
<th>Minimax</th>
<th>SIP</th>
<th>Dim-x</th>
<th>Dim-y</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 CPU</td>
<td>0.031</td>
<td>0.094</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>iterations</td>
<td>13</td>
<td>41(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 CPU</td>
<td>0.047</td>
<td>0.125</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>iterations</td>
<td>16</td>
<td>38(2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 CPU</td>
<td>0.188</td>
<td>0.215</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>iterations</td>
<td>29</td>
<td>75(3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 CPU</td>
<td>0.094</td>
<td>0.220</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>iterations</td>
<td>31</td>
<td>84(3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 CPU</td>
<td>0.172</td>
<td>0.204</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>iterations</td>
<td>53</td>
<td>58(4)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The results shown in table 5 are consistent with those from table 4 and indicate that the minimax algorithm converges faster. It is also worth observing that both algorithms perform better for problems with multiple maximizers. The reason for improved performance is that although at each point \(x\) there might be a number of different maximizers, the set of maximizers is discrete as it usually has a finite number of elements. In the case of convex – concave functions, this is not the case as the set

\[
\{ y \mid \Phi(x) = f(x, y), \forall x \in X_f \}
\]

has in general an infinite number of elements, as for each \(x\) there exists a different \(y\) so that \(\Phi(x) = f(x, y)\). Therefore, at each iteration global optimization is needed, whereas in the convex-convex case, a switch to the discrete minimax problem is justified (see Step 2(a)(ii), Algorithm 2) once the appropriate number of maximizers, as established in Caratheodory’s theorem [26], become available.
7 Conclusions

In this paper, we extend the algorithm for unconstrained continuous minimax problems in Rustem and Howe [30] to include constraints on the minimizing variable \( x \). An interior point approach is used to ensure feasibility of the variables. A semi-infinite programming based epi-convergent procedure accumulates maximizers. This is realized with a discrete minimax formulation that identifies the potential maximizers and also tests for termination. A quasi–Newton search direction, conditional on approximate maximizers is used and progress is maintained through a merit function.

To illustrate the performance of the algorithm a number of numerical examples are considered and compared with a purely semi-infinite programming approach. The results reported underline a superior performance of the algorithm in this paper. The main reason for this is (i) that the discrete minimax procedure is able to add new maximizers, if required, at every iteration of the interior point algorithm whereas semi-infinite programming adds the maximizers as new constraints at the end of a full minimization. Costly line searches are avoided by terminating the global maximization procedure after the first maximizer satisfying the Armijo step size rule is identified.

References


[40] Zhang, Y., Tapia, R. A. and Potra, F., ”On the superlinear convergence of interior point algorithms for a general class of problems”, Technical Report TR90–9, Department of Mathematical Sciences, Rice University, USA, 1990.


