An efficient interior point method for linear optimization using modified Newton method

Sajad Fathi Hafshejani, Daya Gaur, and Robert Benkoczi

Department of Math and Computer Science, University of Lethbridge, Lethbridge, Canada.

February 16, 2024

Outline

1. Introduction
   - LO problems
   - Interior point method
   - Solution
2. The new approach
   - Newton's method
   - Two-step method
   - Algorithm
   - Convergence analysis
3. Numerical results
   - Implementation
   - Results

Linear optimization problems

Linear Optimization (LO) problems are given by:

$$(P) \qquad \min\{c^T x : Ax = b, x \geq 0\},$$

Linear optimization problems

Linear Optimization (LO) problems are given by:

$$(P) \qquad \min\{c^T x : Ax = b, x \geq 0\},$$

and its dual is given by:

$$(D) \qquad \max\{b^T y : A^T y + s = c, s \geq 0\},$$

where $x, c, s \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $y \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$ with $m \leq n$.

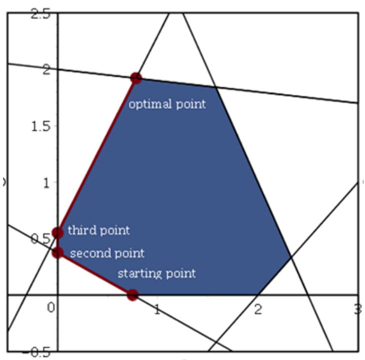
Linear Optimization (LO) problems are given by:

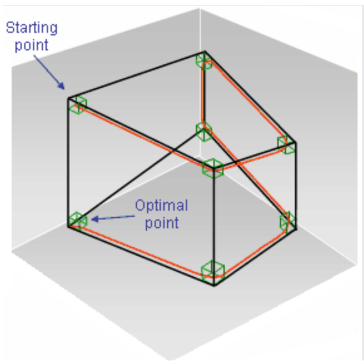
$$(P) \qquad\qquad \min\{c^T x : Ax = b, x \geq 0\},$$

and its dual is given by:

$$(D) \qquad\qquad \max\{b^T y : A^T y + s = c, s \geq 0\},$$

where $x, c, s \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $y \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$ with $m \leq n$.





IPMs

IPMs

- Feasible IPMs

IPMs

IPMs

- Feasible IPMs
- Infeasible IPMs

IPMs

IPMs

- Feasible IPMs
- Infeasible IPMs
- Best worst case complexity bound, i.e., $O(\sqrt{n}\log n \log \frac{n}{\epsilon})$.

IPMs

IPMs

- Feasible IPMs
- Infeasible IPMs
- Best worst case complexity bound, i.e., $O(\sqrt{n}\log n \log \frac{n}{\epsilon})$.

Assumptions:

- The matrix A is full row rank, i.e., rank $(A) = m \leq n$.

IPMs

IPMs

- Feasible IPMs
- Infeasible IPMs
- Best worst case complexity bound, i.e., $O(\sqrt{n}\log n \log \frac{n}{\epsilon})$.

Assumptions:

- The matrix A is full row rank, i.e., rank $(A) = m \leq n$.
- Both problems (P) and (D) satisfy the Interior Point Condition (IPC), i.e., there exists $x^0 > 0$ and (y^0, s^0) with $s^0 > 0$ such that:

$$Ax^0 = b, \qquad A^T y^0 + s^0 = c.$$

Interior point method

The KKT conditions for (P) and (D) are:

$$\begin{aligned} Ax &= b, & x &\geq 0, \\ A^T y + s &= c, & s &\geq 0, \\ xs &= 0, \end{aligned} \tag{1}$$

where the coordinate-wise product of vectors x and s is denoted as xs.

Interior point method

The KKT conditions for (P) and (D) are:

$$\begin{aligned} Ax &= b, & x &\geq 0, \\ A^T y + s &= c, & s &\geq 0, \\ xs &= 0, \end{aligned} \tag{1}$$

where the coordinate-wise product of vectors x and s is denoted as xs.

IPM's idea

$$\begin{aligned} Ax &= b, & x &\geq 0, \\ A^T y + s &= c, & s &\geq 0, \\ xs &= \mu \mathbf{e}, \end{aligned} \tag{2}$$

Interior point method

The KKT conditions for (P) and (D) are:

$$\begin{aligned} Ax &= b, & x &\geq 0, \\ A^T y + s &= c, & s &\geq 0, \\ xs &= 0, \end{aligned} \tag{1}$$

where the coordinate-wise product of vectors x and s is denoted as xs.

IPM's idea

$$\begin{aligned} Ax &= b, & x &\geq 0, \\ A^T y + s &= c, & s &\geq 0, \\ xs &= \mu \mathbf{e}, \end{aligned} \tag{2}$$

Based on the IPC condition and the full row rank property of matrix A, the system (2) has a unique solution $(x(\mu), y(\mu), s(\mu))$. The terms $x(\mu)$ and $(y(\mu), s(\mu))$ are called the μ-centers of (P) and (D), respectively,

Solution

To find the solution, we can consider the following problem:

$$\xi = \begin{bmatrix} x \\ y \\ s \end{bmatrix} \text{ and } F(\xi) = \begin{bmatrix} Ax - b \\ A^T y + s - c \\ \mu \mathbf{e} - xs \end{bmatrix} = 0 \tag{3}$$

where the operator F is defined on the Banach space B_1 with values in a Banach space B_2. We can find the root of equation (3) denoted by ξ^* where $F(\xi^*) = 0$.

Newton's method

Applying a linear approximation using the Taylor series expansion around ξ, we have

$$F(\xi) + F'(\xi)\Delta\xi \simeq 0$$

where

$$\text{Jacobian } F' = \begin{bmatrix} \frac{\partial F_1}{\partial \xi_1} & \frac{\partial F_1}{\partial \xi_2} & \frac{\partial F_1}{\partial \xi_3} \\ \frac{\partial F_2}{\partial \xi_1} & \frac{\partial F_2}{\partial \xi_2} & \frac{\partial F_2}{\partial \xi_3} \\ \frac{\partial F_3}{\partial \xi_1} & \frac{\partial F_3}{\partial \xi_2} & \frac{\partial F_3}{\partial \xi_3} \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \text{ and } \Delta\xi = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix},$$

where X, S are diagonal matrices constructed from x and s.

Applying a linear approximation using the Taylor series expansion around ξ, we have

$$F(\xi) + F'(\xi)\Delta\xi \simeq 0$$

where

$$\text{Jacobian } F' = \begin{bmatrix} \frac{\partial F_1}{\partial \xi_1} & \frac{\partial F_1}{\partial \xi_2} & \frac{\partial F_1}{\partial \xi_3} \\ \frac{\partial F_2}{\partial \xi_1} & \frac{\partial F_2}{\partial \xi_2} & \frac{\partial F_2}{\partial \xi_3} \\ \frac{\partial F_3}{\partial \xi_1} & \frac{\partial F_3}{\partial \xi_2} & \frac{\partial F_3}{\partial \xi_3} \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \text{ and } \Delta\xi = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix},$$

where X, S are diagonal matrices constructed from x and s.
We update the current estimate ξ_n of the root using the following rule for some appropriate step size α:

$$\xi_{n+1} = \xi_n - \alpha[F'(\xi_n)]^{-1}F(\xi_n)$$

Original idea

To find the root of function $f(x) = 0$, two-step method is given:

Original idea

To find the root of function $f(x) = 0$, two-step method is given:

First step:

$$\begin{aligned}\tilde{x}_0 &= x_0\\ x_1 &= x_0 - \frac{f(x_0)}{f'(\frac{1}{2}[x_0 + \tilde{x}_0])} = x_0 - \frac{f(x_0)}{f'(x_0)},\end{aligned}$$

Original idea

To find the root of function $f(x) = 0$, two-step method is given:

First step:

$$\begin{aligned} \tilde{x}_0 &= x_0 \\ x_1 &= x_0 - \frac{f(x_0)}{f'(\frac{1}{2}[x_0 + \tilde{x}_0])} = x_0 - \frac{f(x_0)}{f'(x_0)}, \end{aligned}$$

and for $k \geq 1$, we have:

Second step:

$$\begin{aligned} \tilde{x}_k &= x_k - \frac{f(x_k)}{f'(\frac{1}{2}[x_{k-1} + \tilde{x}_{k-1}])} \\ x_{k+1} &= x_k - \frac{f(x_k)}{f'(\frac{1}{2}[x_k + \tilde{x}_k])}. \end{aligned}$$

Two-step method for IPM

Let F be any function. The first step involves updating an auxiliary point $\tilde{\xi}_0 = \xi_0$.
The update rules used in the n^{th} iteration can be concisely summarized as:

$$
\begin{aligned}
\tilde{\xi}_{n+1} &= \xi_n - \alpha[F'(\hat{\xi}_n)]^{-1}F(\xi_n) \\
\hat{\xi}_{n+1} &= \frac{1}{2}(\tilde{\xi}_{n+1} + \xi_n) \\
\xi_{n+1} &= \xi_n - \alpha[F'(\hat{\xi}_{n+1})]^{-1}F(\xi_n)
\end{aligned}
$$

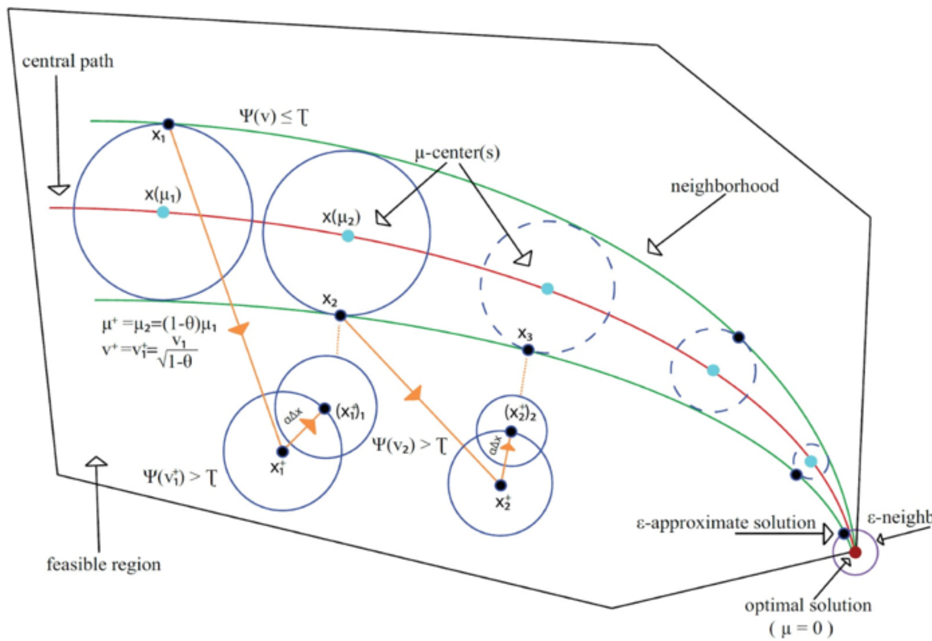
Interior point algorithm

Algorithm 1 Generic Primal–dual IPM for LO.

Input

a proximity function $\Psi(v)$
a threshold parameter $\tau > 0$
an accuracy parameter $\varepsilon > 0$
a barrier update parameter $\theta, 0 < \theta < 1$

begin
  $x := \mathbf{e}; s := \mathbf{e}; \mu := 1; v := \mathbf{e};$
  while $n\mu > \varepsilon$ do
  begin
    $\mu := (1-\theta)\mu;$
    while $\Psi(v) > \tau$ do
    begin
      $x := x + \alpha \Delta x$
      $s := s + \alpha \Delta s$
      $y = y + \alpha \Delta y$
      $v := \sqrt{\frac{xs}{\mu}}$
    end
  end
end



Algorithm

- Start with a feasible point (x, y, s).

[1] $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

Algorithm

- Start with a feasible point (x, y, s).
- Set $\mu \leftarrow \mu(1-\theta)$.

[1] $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

Algorithm

- Start with a feasible point (x, y, s).
- Set $\mu \leftarrow \mu(1-\theta)$.
- Compute the proximity function by:

$$\Psi(x, s, \mu) = \|\mu\mathbf{e} - xs\|_1.$$[1]

Check the condition $\Psi \geq \tau$. If $\Psi < \tau$ go back to step 2.

[1] $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

Algorithm

- Start with a feasible point (x, y, s).
- Set $\mu \leftarrow \mu(1-\theta)$.
- Compute the proximity function by:

$$\Psi(x, s, \mu) = \|\mu\mathbf{e} - xs\|_1.$$ [1]

  Check the condition $\Psi \geq \tau$. If $\Psi < \tau$ go back to step 2.
- Set $(\tilde{x}, \tilde{y}, \tilde{s}) = (x, y, s)$.

[1] $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

Algorithm

- Start with a feasible point (x, y, s).
- Set $\mu \leftarrow \mu(1-\theta)$.
- Compute the proximity function by:

$$\Psi(x, s, \mu) = \|\mu\mathbf{e} - xs\|_1.^{1}$$

  Check the condition $\Psi \geq \tau$. If $\Psi < \tau$ go back to step 2.
- Set $(\tilde{x}, \tilde{y}, \tilde{s}) = (x, y, s)$.
- Compute the average point $(\hat{x}_0, \hat{y}_0, \hat{s}_0) = (\frac{x+\tilde{x}}{2}, \frac{y+\tilde{y}}{2}, \frac{s+\tilde{s}}{2})$

---

[1] $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

Algorithm

- Start with a feasible point (x, y, s).
- Set $\mu \leftarrow \mu(1-\theta)$.
- Compute the proximity function by:

$$\Psi(x, s, \mu) = \|\mu\mathbf{e} - xs\|_1.^1$$

  Check the condition $\Psi \geq \tau$. If $\Psi < \tau$ go back to step 2.
- Set $(\tilde{x}, \tilde{y}, \tilde{s}) = (x, y, s)$.
- Compute the average point $(\hat{x}_0, \hat{y}_0, \hat{s}_0) = (\frac{x+\tilde{x}}{2}, \frac{y+\tilde{y}}{2}, \frac{s+\tilde{s}}{2})$
- Find the search direction using the following system:

$$\begin{aligned} A\Delta x &= 0 \\ A^T\Delta y + \Delta s &= 0 \\ \hat{S}\Delta x + \hat{X}\Delta s &= \mu\mathbf{e} - XS\mathbf{e} \end{aligned} \tag{4}$$

Note the $\hat{X}, \hat{S}$ are diagonal matrices constructed from $\hat{x}, \hat{s}$.

[1] $\|x\|_1 = \sum_{i=1}^{n} |x_i|$

Algorithm

- Find the maximum value for step size β to make the new point feasible.

Algorithm

- Find the maximum value for step size β to make the new point feasible.
- Update the current point by using the following role:

$$(x_+, y_+, s_+) \leftarrow (x + \beta\Delta x, y + \beta\Delta y, s + \beta\Delta s) \tag{5}$$

Iterative Update:

- Compute $\Psi(x, s, \mu)$ and check Step 3 in the Initialization phase and compute $\Psi(x, s, \mu)$.

Iterative Update:

- Compute $\Psi(x, s, \mu)$ and check Step 3 in the Initialization phase and compute $\Psi(x, s, \mu)$.
- Compute the search direction by solving the following system:

$$\begin{aligned} A\Delta\tilde{x} &= 0 \\ A^T\Delta\tilde{y} + \Delta\tilde{s} &= 0 \\ \hat{S}\Delta\tilde{x} + \hat{X}\Delta\tilde{s} &= \mu\mathbf{e} - XS\mathbf{e} \end{aligned} \tag{6}$$

Note that, in practice, we use the information of the previous iteration to calculate the search direction $(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{s})$.

Iterative Update:

- Compute $\Psi(x, s, \mu)$ and check Step 3 in the Initialization phase and compute $\Psi(x, s, \mu)$.
- Compute the search direction by solving the following system:

$$\begin{aligned} A\Delta\tilde{x} &= 0 \\ A^T\Delta\tilde{y} + \Delta\tilde{s} &= 0 \\ \hat{S}\Delta\tilde{x} + \hat{X}\Delta\tilde{s} &= \mu\mathbf{e} - XS\mathbf{e} \end{aligned} \tag{6}$$

  Note that, in practice, we use the information of the previous iteration to calculate the search direction $(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{s})$.
- Find the maximum value for α such that the new auxiliary point will remain feasible.

Iterative Update:

- Compute $\Psi(x, s, \mu)$ and check Step 3 in the Initialization phase and compute $\Psi(x, s, \mu)$.
- Compute the search direction by solving the following system:

$$\begin{aligned} A\Delta\tilde{x} &= 0 \\ A^T\Delta\tilde{y} + \Delta\tilde{s} &= 0 \\ \hat{S}\Delta\tilde{x} + \hat{X}\Delta\tilde{s} &= \mu\mathbf{e} - XS\mathbf{e} \end{aligned} \tag{6}$$

  Note that, in practice, we use the information of the previous iteration to calculate the search direction $(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{s})$.
- Find the maximum value for α such that the new auxiliary point will remain feasible.
- Update the auxiliary point by:

$$(\tilde{x}_+, \tilde{y}_+, \tilde{s}_+) \leftarrow (x + \alpha\Delta\tilde{x}, y + \alpha\Delta\tilde{y}, s + \alpha\Delta\tilde{s}) \tag{7}$$

Algorithm

- Compute the average.

$$(\hat{x}_+, \hat{y}_+, \hat{s}_+) \leftarrow \frac{1}{2}((\tilde{x}_+, \tilde{y}_+, \tilde{s}_+) + (x, y, s)) \tag{8}$$

Algorithm

- Compute the average.

$$(\hat{x}_+, \hat{y}_+, \hat{s}_+) \leftarrow \frac{1}{2}((\tilde{x}_+, \tilde{y}_+, \tilde{s}_+) + (x, y, s)) \tag{8}$$

- Solve the following system of equations to obtain the search direction.

$$\begin{aligned} A\Delta x &= 0 \\ A^T\Delta y + \Delta s &= 0 \\ \hat{S}_+\Delta x + \hat{X}_+\Delta s &= \mu\mathbf{e} - XS\mathbf{e} \end{aligned} \tag{9}$$

Algorithm

- Compute the average.

$$(\hat{x}_+, \hat{y}_+, \hat{s}_+) \leftarrow \frac{1}{2}((\tilde{x}_+, \tilde{y}_+, \tilde{s}_+) + (x, y, s)) \tag{8}$$

- Solve the following system of equations to obtain the search direction.

$$\begin{aligned} A\Delta x &= 0 \\ A^T \Delta y + \Delta s &= 0 \\ \hat{S}_+ \Delta x + \hat{X}_+ \Delta s &= \mu \mathbf{e} - XS\mathbf{e} \end{aligned} \tag{9}$$

- Find the maximum value for step size β.

Algorithm

- Compute the average.

$$(\hat{x}_+, \hat{y}_+, \hat{s}_+) \leftarrow \frac{1}{2}((\tilde{x}_+, \tilde{y}_+, \tilde{s}_+) + (x, y, s)) \tag{8}$$

- Solve the following system of equations to obtain the search direction.

$$\begin{aligned} A\Delta x &= 0 \\ A^T\Delta y + \Delta s &= 0 \\ \hat{S}_+\Delta x + \hat{X}_+\Delta s &= \mu\mathbf{e} - XS\mathbf{e} \end{aligned} \tag{9}$$

- Find the maximum value for step size β.
- Update the current point by using the following role:

$$(x_+, y_+, s_+) \leftarrow (x + \beta\Delta x, y + \beta\Delta y, s + \beta\Delta s) \tag{10}$$

Algorithm

- Compute the average.

$$(\hat{x}_+, \hat{y}_+, \hat{s}_+) \leftarrow \frac{1}{2}((\tilde{x}_+, \tilde{y}_+, \tilde{s}_+) + (x, y, s)) \tag{8}$$

- Solve the following system of equations to obtain the search direction.

$$\begin{aligned} A\Delta x &= 0 \\ A^T\Delta y + \Delta s &= 0 \\ \hat{S}_+\Delta x + \hat{X}_+\Delta s &= \mu\mathbf{e} - XS\mathbf{e} \end{aligned} \tag{9}$$

- Find the maximum value for step size β.
- Update the current point by using the following role:

$$(x_+, y_+, s_+) \leftarrow (x + \beta\Delta x, y + \beta\Delta y, s + \beta\Delta s) \tag{10}$$

- Check Step 1. If $\Psi(x, s, \mu) < \tau$ stop the inner loop.

Algorithm

Algorithm 2: A two-step feasible IPM algorithm for LPs.

Input: $x_0,\ y_0,\ s_0,\ \mu > 0,\ \tau > 0,\ \varepsilon > 0,\ \theta \in (0,1)$, and $\Psi(x, s, \mu)$

$(x, y, s) \leftarrow (x_0, y_0, s_0)$

$k, m \leftarrow 0$

while *stopping criteria is not met* **do**

  $\mu \leftarrow \mu(1-\theta)$

  while $\Psi(x, y, s, \mu) \geq \tau$ **do**

    if $m{=}{=}0$ **then**

      $(\tilde{x}, \tilde{y}, \tilde{s}) \leftarrow (x, y, s)$

      Update $(\hat{x}_+, \hat{y}_+, \hat{s}_+) \leftarrow \frac{1}{2}((\tilde{x}, \tilde{y}, \tilde{s}) + (x, y, s))$

      Find search direction $(\Delta\tilde{x}, \Delta\tilde{y}, \Delta\tilde{s})$ using (6)

      Find the value for β and update $(x, y, s) \leftarrow (x + \beta\Delta x, y + \beta\Delta y, s + \beta\Delta s)$

    if $m \geq 1$ **then**

      Find the search direction using (6)

      Find the maximum step size α and update the auxiliary point using (7)

      Update the average point by using (8)

      Find search direction $(\Delta x, \Delta y, \Delta s)$ by solving (9)

      Find the maximum value for step size β and update the current point by using (10)

    $m \leftarrow m + 1$

  $k \leftarrow k + 1$

Convergence analysis

Convergence analysis

Lemma:

The total number of outer iterations to obtain $n\mu \leq \epsilon$ are

$$O\left(\frac{1}{\theta}\log\frac{n}{\epsilon}\right)$$

.

Convergence analysis

Lemma:

The total number of outer iterations to obtain $n\mu \leq \epsilon$ are

$$O\left(\frac{1}{\theta}\log\frac{n}{\epsilon}\right)$$

.

Theorem:

Suppose the outer loop updates the barrier parameter by factor $\theta \in (0,1)$ and $k \to \infty$. Then one has:

$$\|\xi_k - \xi^*\| \leq \epsilon.$$

Implementation

Implementation

Algorithms:

We conducted a comparison with the classical algorithm in [1].

Implementation

Algorithms:

We conducted a comparison with the classical algorithm in [1].

Device's detail:

We programmed the two algorithms in Python 3.10.
We conducted the tests on the Alliance Canada cluster CEDAR (https://alliancecanada.ca).

Implementation

Algorithms:

We conducted a comparison with the classical algorithm in [1].

Device's detail:

We programmed the two algorithms in Python 3.10.
We conducted the tests on the Alliance Canada cluster CEDAR (https://alliancecanada.ca).

Stopping condition:

For both algorithms, we stopped if the number of iterations exceeded 700 or if the relative gap was less than 10^{-6}. The relative gap is the absolute difference between $c^T x$ and $b^T y$ divided by $1 + |c^T x| + |b^T y|$.

Implementation

Test problems:

We have selected 46 test problems of varying sizes from the Netlib collection.

Implementation

Test problems:

We have selected 46 test problems of varying sizes from the Netlib collection.

Barrier parameter:

As for the barrier parameter, we have set the initial value to $\mu_0 = 1$ for both algorithms. In each iteration of the outer loop of the algorithm, we reduce the value of μ by $\mu = (1 - \theta)\mu$.

Implementation

Test problems:

We have selected 46 test problems of varying sizes from the Netlib collection.

Barrier parameter:

As for the barrier parameter, we have set the initial value to $\mu_0 = 1$ for both algorithms. In each iteration of the outer loop of the algorithm, we reduce the value of μ by $\mu = (1 - \theta)\mu$.

Proximity function:

For our algorithms, we rely on the proximity function specified as follows: $\Psi(x, s, \mu) = \|\mu\mathbf{e} - xs\|_1$, where $\|x\|_1 = \sum_{i=1}^{n} |x_i|$.

Implementation

Threshold parameter:

We are currently working on a large-update method, τ value should be set to $O(n)$.

Implementation

Threshold parameter:

We are currently working on a large-update method, τ value should be set to $O(n)$.

Barrier update parameter:

In all experiments, we use $\theta = 0.6$ to update μ.

Implementation

Threshold parameter:

We are currently working on a large-update method, τ value should be set to $O(n)$.

Barrier update parameter:

In all experiments, we use $\theta = 0.6$ to update μ.

Step size:

We use the following equations:

$$\alpha_x^{\max} = \frac{1}{\max_{i=1,2,\cdots,n}\{1, -\frac{x_i}{\Delta x_i}\}}, \qquad \alpha_s^{\max} = \frac{1}{\max_{i=1,2,\cdots,n}\{1, -\frac{s_i}{\Delta s_i}\}}.$$

To ensure we don't hit the boundary, we reduce the maximum allowable step sizes by a fixed factor of $0 < \alpha_0 < 1$. Therefore, our final step sizes are given by $\alpha_x = \alpha_0.\alpha_x^{\max}$ and $\alpha_s = \alpha_0.\alpha_s^{\max}$.

Results

Methods	Aver. Iter.	Aver. CPU
Classical Algorithm	95.29	132.46
Algorithm 2	**65.77**	**105.43**

Table 1: The average number of iterations and CPU time

Results

Methods	Aver. Iter.	Aver. CPU
Classical Algorithm	95.29	132.46
Algorithm 2	**65.77**	**105.43**

Table 1: The average number of iterations and CPU time

- The new proposed approach can significantly reduce the number of iterations and CPU times by %30.97 and %20.46, respectively.

References

- [1] Roos, C., Terlaky, T., Vial, J.P.: Theory and algorithms for linear optimization: an interior point approach. Wiley Chichester (1997).
- [2] McDougall, T.J., Wotherspoon, S.J.: A simple modification of Newton's method to achieve convergence of order $1+\sqrt{2}$. Applied Mathematics Letters 29, 20–25 (2014).
- [3] Argyros, I.K., Deep, G., Regmi, S.: Extended Newton-like midpoint method for solving equations in Banach space. Foundations 3(1), 82–98 (2023).

Thank You For Your Attention!

Any Questions?