sciendo

Műszaki Tudományos Közlemények vol. 11. (2019) 47–50.
https://doi.org/10.33894/mtk-2019.11.08
Hungarian: https://doi.org/10.33895/mtk-2019.11.08
https://eda.eme.ro/handle/10598/31245

EME MTSZ MTK

IMPLEMENTATION OF AN INTERIOR-POINT ALGORITHM FOR LINEAR COMPLEMENTARITY PROBLEM WORKING IN A WIDE NEIGHBORHOOD

Zsolt DARVAY [1], Attila-Szabolcs ORBÁN[2]

Babeș-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

[1] *darvay@cs.ubbcluj.ro*

[2] *orban.attila@yahoo.com*

Abstract

In this article, we study the interior-point algorithm for solving linear complementarity problems, published by Xiaouje Ma, Hongwei Liu, Jianke Zhang and Weijie Cong from the implementation point of view. The algorithm was implemented in C++ programming language, thus supporting the effectiveness of the method. Despite the fact that the theoretical results refer only to the monotone linear complementarity problem, practical testing showed that the algorithm also works well in more general cases.

Keywords*: interior-point algorithm, linear comlementarity problem, wide neighborhood, path-following algorithm, object-oriented technique.*

1. Introduction

Various interior-point algorithms have been introduced [1,2,3] to solve linear optimalization problems, and which have proven to be effective. Some of the methods were generalized to linear complementarity problems (LCPs) [4], which are often used to solve engineering problems. There are two types of path-following interior-point algorithms, namely the short and long step versions. As it turns out the theoretical efficiency of short step algorithms is generally better, but the long step ones perform better in practice. Ai [5] introduced the first large update algorithm for linear optimalization (APF) which has the same theoretical complexity as the best short step methods.

Ai and Zhang [6] generalized Ai's method to LCP. Ma, Liu, Zhang and Cong [7] extended the above algorithm (APF+) by introducing two different steps. They called the first one fast step and the second one safe step. Both work in the wide neighborhood specified by Ai, but in the case of fast step, the analyzed neighborhood is modified by changing the parameters. In the following sections we investigate this algorithm and discuss the possibilities for its implementation.

2. The description of the problem

The LCP can be formulated in the following way:

$$\begin{cases} s = Mx + q, \\ x \geq 0, s \geq 0, x^T s = 0, \end{cases}$$

where $q \in \mathbb{R}^n$ and $M \in \mathbb{R}^{n \times n}$. Furthermore, we assume that matrix $M+M^T$ is positive semidefinite, i.e. $x^T Mx \geq 0$ for any $x \in \mathbb{R}^n$. The matrix M and the vector q are given, and x, s are the variables of the problem.

Primal-dual interior-point methods have proven to be very effective for solving LCPs.

We assume that $\mathcal{F}_{++}$ the set of feasible solutions of the LCP is nonempty:

$$\mathcal{F}_{++} := \{(x,s) | s = Mx + q, \quad x > 0, \quad s > 0\}.$$

The following set defines the central path, which is a basic concept for primal-dual interior-point methods:

$$C := \{(x,s) \in \mathcal{F}_{++} | xs = \mu e, \mu > 0\},$$

where xs denotes the componentwise product of $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^n$.

Other notations used in the paper are: x_i-denotes the i-th element of the vector $x \in \mathbb{R}^n$; e is

the n-dimensional all-one vector; for any number $a \in \mathbb{R}^n$ $a^+ := \max\{a, 0\}$ and $a^- := \min\{a, 0\}$; $\|x\|$ is the l_2 norm, $\|x\|_1$ is the l_1 norm. We extend the notations a^+ and a^- for vectors as well.

3. The modified algorithm

Ai defined the wide neighborhood for the APF algorithm as follows:

$$\mathcal{N}(\tau,\beta) := \{(x,s) \in \mathcal{F}_{++} \mid \|(\tau\mu e - xs)^+\|_1 \le \beta\tau\mu \}$$

where $0 < \beta < 1$ and $\mu = (x^T s) / n$. For the APF+ algorithm we need to introduce the following parameters:

$$\theta \in \left(0, \frac{1}{2}\right], \kappa \in (0,\theta), \beta \in [\beta_0, \beta_{max}],$$
$$0 < \beta_0 < \beta_{max} \le \frac{1}{3}.$$

Assume that the pair of current iterates (x, s) belongs to the wide neighborhood $\mathcal{N}(\tau, \beta)$. We use a fast step to solve the following system of equations:

$$\begin{cases} \Delta s^a = M\Delta x^a, \\ s\Delta x^a + x\Delta s^a = -xs. \end{cases} \quad (1)$$

After determining Δx^a and Δs^a we try to calculate the step size α, such that the new vectors $x(\alpha) = x + \alpha\Delta x^a$ and $s(\alpha) = s + \alpha\Delta s^a$ reside in the wider neighborhood $\mathcal{N}(\tau, \theta\beta + (1 - \theta)\beta_{max})$. This is achieved by slightly increasing the value of β, which is one of the defining characteristics of the APF+ algorithm. The normalized duality gap is calculated using the formula $\mu(\alpha)=(x(\alpha)^T s(\alpha)) / n$.

In [7] the authors intorduce a threshold value named κ and decide with the help of the inequality $\mu(\alpha) \le \kappa\mu$ if the safe step is needed. If the inequality holds, the new value of β will be $\theta\beta + (1 - \theta)\beta_{max}$. Otherwise, we take a safe step, such that the resulting points will be in the original neighborhood $N(\tau,\beta)$. In case of a safe step, we solve the following system of equations:

$$\begin{cases} \Delta s^p = M\Delta x^p, \\ s\Delta x^p + x\Delta s^p = \lambda(\tau\mu e - xs)^- + (\tau\mu e - xs)^+, \end{cases} \quad (2)$$

where $\lambda = \dfrac{\|(\tau\mu e - xs)^+\|_1}{\|(\tau\mu e - xs)^-\|_1}$,

$\lambda e^T(\tau\mu e - xs)^- + e^T(\tau\mu e - xs)^+ = 0$. However, the specified pair $(\Delta x^p, \Delta s^p)$ does not give the direction of the safe step, so we must also include $(\Delta x^a, \Delta s^a)$. Unlike the article [7] we don't work with a linear combination of directions $(\Delta x^a, \Delta s^a)$ and $(\Delta x^p, \Delta s^p)$, but we introduce a $0 < \gamma < 1$ constant, which will be used for weighting the safe step. So the step of size α results in:

$x(\alpha) = x + \alpha(\gamma\Delta x^p + \Delta x^a)$ and

$s(\alpha) = s + \alpha(\gamma\Delta s^p + \Delta s^a)$.

4. The algorithm

In the following we present the algorithm published in [7], modified by us using the parameter γ.

Input parameters: required precision $\varepsilon > 0$, $0 < \beta_0 < \beta_{max} \le 1/3$, $\tau < 1/2$, $0 < \theta \le 1/2$, $0 < \kappa < \theta$, $\gamma \in (0,1)$ and the initial point $(x^0, s^0) \in \mathcal{N}(\tau, \beta_0)$.

$(x, s) = (x^0, s^0)$; $\beta = \beta_0$;

while $x^T s > \varepsilon$ **do begin**

Calculate $(\Delta x^a, \Delta s^a)$ based on (1).

fast step:

α = alphaFast$(x, s, \Delta x^a, \Delta s^a, \beta, \beta_{max})$

$x(\alpha) = x + \alpha\Delta x^a$; $s(\alpha) = s + \alpha\Delta s^a$;

$\mu(\alpha) = (x(\alpha)^T s(\alpha)) / n$;

if $\mu(\alpha) \le \kappa\mu$ **then**

$\beta = \theta\beta + (1 - \theta)\beta_{max}$;

else begin

safe step:

Calculate $(\Delta x^p, \Delta s^p)$ based on (2).

α = alphaSafe(x, s, Δx^a, Δs^a, Δx^p, Δs^p, β);

$x(\alpha) = x + \alpha(\gamma\Delta x^p + \Delta x^a)$;

$s(\alpha) = s + \alpha(\gamma\Delta s^p + \Delta s^a)$;

$\mu(\alpha) = (x(\alpha)^T s(\alpha)) / n$;

end if

$(x, s) = (x(\alpha), s(\alpha))$; $\mu = \mu(\alpha)$;

end.

We mention that functions *alphaFast* and *alphaSafe* calculate the step length, such that the pair of vectors $(x(\alpha), s(\alpha))$ will be in neighborhoods $\mathcal{N}(\tau, \theta\beta + (1 - \theta)\beta_{max})$ and $\mathcal{N}(\tau, \beta)$.

5. Implementation

We implemented the algorithm in the C++ programming language, using the Visual Studio development environment and the code introduced in [8].

During the implementation of both steps we faced the question of how the maximum step-size can be determined so that the obtained iterates remain in the wide neighborhood.

The best step-size α can be computed by solving the optimalization problem $\min \mu(\alpha)$, $(x(\alpha), s(\alpha)) \in \mathcal{N}(\tau,\beta)$, $0 < \alpha \leq 1$ in each iteration [7].

Instead of this we calculated the step-size as follows: first we determined the maximum step length, which satisfies the feasibility conditions. Then we checked whether the new iterates were in the neighborhood. If the condition was not met, we halved the step and checked again whether the obtained points were in the neighborhood. We repeated this procedure until the corresponding step-size was obtained.

6. Numerical tests

The algorithm was tested on two monotone LCPs.

The first is the following (lcp01, see [9]):

$$M = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 0 & 1 \\ 1 & 0 & 1 & 2 \\ -1 & -1 & -2 & 0 \end{pmatrix}, q = \begin{bmatrix} -8 \\ -6 \\ -4 \\ 3 \end{bmatrix}.$$

The second LCP's matrix can be found in [10]. The problem, which was investigated by us is the following (lcp02):

$$M = (m_{ij})_{\substack{i=1..n \\ j=1..n}}, m_{ii} = 4i - 3, m_{ij} = 4i - 2, i \neq j$$

n = 10, q = [1, 5, 3, 6, 1, -7, 1, 8, 9, 1]^T.

6.1. Analyzing the change of θ

At each iteration of the algorithm, we change the neighborhood of the central path, which is characterized by the parameter β. Its value may vary between a predetermined lower and upper bound. The modification is made using parameter θ, which is positive and does not exceed 1/2 (see Table 1. and 2.).

We can observe that using lower values of θ generally results in a lower number of iterations.

6.2. Analyzing the change of γ

In the safe step, we introduced the parameter γ, which is responsible for weighting $(\Delta x^p, \Delta s^p)$. Thus, we can modify the new iterates $x(\alpha) = x + \alpha(\gamma\Delta x^p + \Delta x^a)$ and $s(\alpha) = s + \alpha(\gamma\Delta s^p + \Delta s^a)$. In the following, we study the effect of changing the parameter γ on the results determined by the algorithm.

It can be stated that if γ is very small then the effect of the safe step will be reduced so that more iterations are performed (see Table 3. and 4.).

Table 1. *Results for lcp01*

Number of the test	θ ∈ (0, 1/2]	Iterations
1	0.04	19
2	0.07	20
3	0.2	20
4	0.5	20

Table 2. *Results for lcp02*

Number of the test	θ ∈ (0, 1/2]	Iterations
1	0.04	30
2	0.07	31
3	0.2	32
4	0.5	33

Table 3. *Results for lcp01*

Number of the test	γ ∈ (0, 1)	Iterations
1	0.9	20
2	0.6	20
3	0.3	21
4	0.01	21

Table 4. *Results for lcp02*

Number of the test	γ ∈ (0, 1)	Iterations
1	0.9	32
2	0.6	32
3	0.3	33
4	0.01	34

7. Conclusions

In this article we investigated the algorithm of Ma, Liu, Zhang and Cong, which uses two different steps to achieve the theoretical efficiency of short-step algorithms.

We modified the algorithm from the implementation point of view, by weighting the safe step using a parameter γ.

The efficiency of the algorithm was proved using a code written in the C++ programming language. In the case of various monotone LCPs, we analyzed the change in the number of iterations depending on the θ and γ parameters.

Despite the fact that the algorithm applies to monotone LCPs, we also achieved good results for some test problems, which does not fulfill this

requirement. We solved the test problems published in **[11]** with matrices of size 10×10 and 20×20 in, at most, 24 iterations. We mention that the first numerical results for the problems presented in **[11]** were published in **[12]**.

Accordingly, in the future it is worth analyzing the theoretical efficiency of the algorithm for this more general class of matrices.

Acknowledgement

The authors acknowledge the research support of the Transylvanian Museum Society (Erdélyi Múzeum-Egyesület).

References

[1] Roos C., Terlaky T., Vial. J.-Ph.: *Theory and Algorithms for Linear Optimization*. Springer, NY, USA, 2005.

[2] Wright. S. J.: *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, USA, 1997.

[3] Ye. Y.: *Interior Point Algorithms, Theory and Analysis.* John Wiley & Sons, Chichester, UK, 1997/3. (1997)

[4] Kojima M., Megiddo N., Noma T., Yoshise A.: *A Unifed Approach to Interior Point Algorithms for Linear Complementarity Problems*. Lecture Notes in Computer Science 538, Springer Verlag, Berlin, Germany, 1991.

[5] Ai. W.: *Neighborhood-following algorithm for linear programming*. Sci. China serie A, 47. (2004) 812 – 820.

[6] Ai W., Zhang S.: *An O(√nL) iteration primal-dual path-following method, based on wide neighborhoods and large updates, for monotone LCP.* SIAM Journal on Optimization 16/2. (2005) 400–417. https://doi.org/10.1137/040604492

[7] Ma X., Liu H., Zhang J., Cong W.: *On superlinear and O(√nL) convergence of a path-following algorithm for monotone linear complementarity problems in a wide neighborhood.* Numerical Functional Analysis and Optimization, 38/5. (2017) 627–640. https://doi.org/10.1080/01630563.2017.1297824

[8] Darvay Zs., Takó I.: *Computational comparison of primal-dual algorithms based on a new software.* unpublished manuscript. (2012)

[9] Hock W., Shittkowski K.: *Test Examples for Nonlinear Programming Codes.* Lecture Notes in Economics and Mathematical Systems 187. Springer, Berlin (1981) https://doi.org/10.1007/978-3-642-48320-2

[10] Harker P. T., Pang J. S.: *A damped Newton method for linear complementarity problem.* In: Simulation an Optimization of Large Systems, Lectures on Applied Mathematics, AMS, Providence, RI, 26. (1990) 265–284.

[11] Morapitiye S.: *Sufficient Matrices*. (accessed on 9 February 2019). http://math.bme.hu/~sunil/su-matrices/

[12] Darvay Zs., Illés T., Povh J., Rigó P. R.: *Predictor-corrector interior-point algorithm for sufficient linear complementarity problems based on a new search direction*, manuscript. (2019)