Mathematical Programming 82 (1998) 339–355

A modified layered-step interior-point algorithm for linear programming [1]

Nimrod Megiddo [a,b,2], Shinji Mizuno [c,3], Takashi Tsuchiya [c,*]

[a] *IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA*

[b] *School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel*

[c] *The Institute of Statistical Mathematics, 4-6-7 Minami-Azabu, Minato-ku, Tokyo 106, Japan*

Received 25 June 1996; revised manuscript received 19 May 1997

Abstract

The layered-step interior-point algorithm was introduced by Vavasis and Ye. The algorithm accelerates the path following interior-point algorithm and its arithmetic complexity depends only on the coefficient matrix A. The main drawback of the algorithm is the use of an unknown big constant $\bar{\chi}_A$ in computing the search direction and to initiate the algorithm. We propose a modified layered-step interior-point algorithm which does not use the big constant in computing the search direction. The constant is required only for initialization when a well-centered feasible solution is not available, and it is not required if an upper bound on the norm of a primal–dual optimal solution is known in advance. The complexity of the simplified algorithm is the same as that of Vavasis and Ye. © 1998 The Mathematical Programming Society, Inc. Published by Elsevier Science B.V.

Keywords: Linear programming; Layered-step interior-point method; Path of centers; Crossover events

1. Introduction

Interior-point methods for solving linear programming problems were introduced by Karmarkar [1]. Although the arithmetic complexity of the interior-point methods is polynomial, it depends on the size L of input data (A, b, c), where A is the coefficient matrix of a linear programming instance, b the right-hand side vector, and c the

* Corresponding author. E-mail: tsuchiya@sun312.ism.ac.jp.

[1] Research supported in part by ONR contract N00014-94-C-0007 and the Grant-in-Aid for Scientific Research (C) 08680478 and the Grant-in-Aid for Encouragement of Young Scientists (A) 08780227 of the Ministry of Science, Education and Culture of Japan. This research was partially done while S. Mizuno and T. Tsuchiya were visiting IBM Almaden Research Center in the summer of 1995.

[2] E-mail: megiddo@almaden.ibm.com.

[3] E-mail: mizuno@ism.ac.jp.

S0025-5610/98/$19.00 © 1998 The Mathematical Programming Society, Inc.
Published by Elsevier Science B.V.
PII S0025-5610(97)00073-7

coefficient vector of the object function. Recently Vavasis and Ye [2] proposed a very elegant algorithm, whose arithmetic complexity does not depend on b or c. Henceforth, we refer to the Vavasis and Ye algorithm as the VY algorithm. Like Tardos' algorithm [3] does, the VY algorithm solves flow problems in strongly polynomial time. The VY algorithm is also called a "layered-step" interior-point algorithm, since it occasionally uses a layered least squares (LLS) direction to compute a new iterate.

The VY algorithm is at least as fast as the $O(\sqrt{n}L)$-iteration primal–dual path-following algorithm proposed by Kojima et al. [4] and Monteiro and Adler [5] or the predictor-corrector algorithm by Mizuno et al. [6]. Furthermore, if the path of centers has an almost straight part, the layered step may accelerate the algorithm. In particular, it attains an exact optimal solution when the iterate is close enough to a solution. So a layered-step interior-point algorithm is not only efficient in theory, but may also become a very good algorithm in practice.

The number of arithmetic operations performed by the VY algorithm is bounded in terms of a big constant $\bar{\chi}_A$ which is defined as the maximum of $\|A^{\mathrm{T}}(ADA^{\mathrm{T}})^{-1}AD\|$, where D is a diagonal matrix whose diagonal entries are positive. This number is used for (i) computing the search direction, and (ii) constructing a problem which is equivalent to the original problem with a trivial initial primal–dual interior feasible solution when no feasible initial point is available. A drawback of the VY algorithm is that a good estimate of $\bar{\chi}_A$ should be known in advance, which may be difficult to compute. It may, however, be estimated by 2^L if A is an integral matrix of input size L.

In this paper, we propose a modification of the layered-step interior-point algorithm. Our algorithm does not use any unknown number for computing the search direction. Instead, we need an estimate of the norm of an optimal solution, but it is only necessary for constructing an equivalent problem to initiate the algorithm. If we know $\bar{\chi}_A$, we obtain a bound on the norm of an optimal solution as shown in [2] and hence our algorithm is implementable. Thus, our algorithm is an extension of the VY algorithm. We will show that the worst-case complexity of our algorithm is the same as that of the VY algorithm, and does not depend on the estimate of the norm of the optimal solution. We believe this is a significant step towards implementation of the VY algorithm.

2. Layered least squares step

Let A be an $m \times n$ matrix, where $m \leqslant n$. In this section, we explain the LLS step according to [2]. This step computes an LLS direction $(\delta x^*, \delta y^*, \delta s^*)$ for given $x \in \mathbb{R}^n$, $s \in \mathbb{R}^n$, a positive $n \times n$ diagonal matrix Δ, and a partition $J = (J_1, J_2, \ldots, J_p)$ of the index set $\{1, 2, \ldots, n\}$. (In the algorithm by Vavasis and Ye [2], the vectors x and s are current iterates and the diagonal matrix Δ and the partition J are computed from them.)

Let $x_{J_1}, \ldots, x_{J_p}$ be subvectors of x indexed by $J_1, J_2, \ldots, J_p$. Similarly, we define subvectors of s, δx, and δs. We also denote diagonal submatrices of Δ by $\Delta_{J_1}, \ldots, \Delta_{J_p}$.

The dual layered least squares (DLLS) step is defined as follows: given a vector s, let $L_0^D = \{\delta s\colon A^T y + \delta s = 0,\ y \in \mathbb{R}^m\}$. Then for $k = 1, 2, \ldots, p$ define the subspaces

$$I_k^D = \{\text{minimizers } \delta s \text{ of } \|\Delta_{J_k}^{-1}(\delta s_{J_k} + s_{J_k})\| \text{ subject to } \delta s \in L_{k-1}^D\}$$

so that $L_0^D \supset L_1^D \supset \cdots \supset L_p^D$. Vavasis and Ye proved that L_p^D has a unique element δs^* such that $A^T \delta y^* + \delta s^* = 0$ for a δy^*. If $p = 1$ then we have $J_1 = \{1, 2, \ldots, n\}$ and

$$\delta s^* = -A^T(A\Delta^{-2}A^T)^{-1}A\Delta^{-2}s.$$

If Δ is appropriately determined, it can be shown that this direction coincides with the part of the primal–dual affine scaling direction that corresponds to the dual slack variables.

The primal layered least squares (PLLS) step is similar, except that we work with the null space of A, Δ instead of Δ^{-1}, and the reverse order of the partitions. That is, for a given vector x, we define $L_0^P = \{\delta x\colon A\delta x = 0\}$. Then, for $k = p, p-1, \ldots, 1$, define the subspaces

$$L_{k-1}^P = \{\text{minimizers } \delta x \text{ of } \|\Delta_{J_k}(\delta x_{J_k} + x_{J_k})\| \text{ subject to } \delta x \in L_k^P\}$$

so that $L_p^P \supset L_{p-1}^P \supset \cdots \supset L_0^P$. The direction δx^* is the unique element in L_0^P. If $p = 1$ then we have $J_1 = \{1, 2, \ldots, n\}$ and

$$\delta x^* = -(x - \Delta^{-2}A^T(A\Delta^{-2}A^T)^{-1}Ax).$$

If Δ is appropriately determined, it can be shown that this direction coincides with the part of the primal–dual affine scaling direction that corresponds to the primal variables.

It was shown in [2] that the DLLS step and PLLS step are computed in $O(nm^2)$ arithmetic operations.

3. Algorithm

In this section, we summarize VY algorithm and explain our algorithm. For an $m \times n$ matrix A and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, we define a primal–dual pair of linear programming problems

$$\text{minimize} \quad c^T x \quad \text{subject to} \quad Ax = b, x \geqslant 0 \tag{1}$$

and

$$\text{maximize} \quad b^T y \quad \text{subject to} \quad A^T y + s = c, \quad s \geqslant 0. \tag{2}$$

We assume that the primal–dual pair has a feasible interior point (x, y, s) (i.e., $x > 0$ and $s > 0$) and $\text{Rank}(A) = m$.

For each $\mu > 0$, we denote by $(x(\mu), y(\mu), s(\mu))$ the solution of the system

$$Ax = b, \quad A^T y + s = c, \quad Xs = \mu e, \quad x \geqslant 0, \quad s \geqslant 0,$$

where e is the vector of 1's and $X = \text{diag}(x)$ is the diagonal matrix such that $Xe = x$. The solution is called a center and the set $P = \{(x(\mu), y(\mu), s(\mu))\colon \mu > 0\}$ is called a

path of centers. Let $\beta \in [0,1)$. For each center $(x(\mu), y(\mu), s(\mu))$, we define a neighborhood

$$N(\beta,\mu) = \{(x,y,s)\colon\ Ax = b,\ A^{\mathrm{T}}y + s = c,\ \|Xs - \mu e\| \leqslant \beta\mu,\ x \geqslant 0,\ s \geqslant 0\}.$$

Then the set

$$N(\beta) = \bigcup_{\mu>0} N(\beta,\mu)$$

is a neighborhood of the path of centers P.

Let $1 > \beta_2 > \beta_1 > 0$. For a given point $(x,y,s) \in N(\beta_1,\mu)$ and $\bar{g} \geqslant 1$, we construct a partition $(J_1, J_2, \ldots, J_p)$ of $\{1,2,\ldots,n\}$ as follows. Let

$$\delta_i = \sqrt{\mu s_i / x_i} \quad \text{for } i = 1,2,\ldots,n.$$

Note that if $(x,y,s) \in P$ then $\delta_i = s_i$. We define the diagonal matrix $\Delta = \operatorname{diag}(\delta) = \sqrt{\mu} X^{-1/2} S^{1/2}$. Let π be a permutation that sorts the δ_is in nondecreasing order:

$$\delta_{\pi(1)} \leqslant \delta_{\pi(2)} \leqslant \cdots \leqslant \delta_{\pi(n)}.$$

Let $J_1 = \{\pi(1), \ldots, \pi(i)\}$ be the set of successive indices of π such that the "ratio-gap" satisfies $\delta_{\pi(j+1)}/\delta_{\pi(j)} \leqslant \bar{g}$ for $j = 1,\ldots,i-1$, but $\delta_{\pi(i+1)}/\delta_{\pi(i)} > \bar{g}$. Then put $\pi(i+1), \pi(i+2), \ldots$ in J_2, until another ratio-gap greater than $\bar{g}$ is encountered, and so on. Let J_p be the last set which contains $\pi(n)$. Each component J_i of the partition is referred to as a layer.

3.1. Vavasis and Ye's algorithm

We now summarize the results by Vavasis and Ye [2]. The complexity of their algorithm is expressed in terms of the following constant:

$$\bar{\chi}_A \equiv \max\{\|A^{\mathrm{T}}(ADA^{\mathrm{T}})^{-1}AD\|\colon\ D \text{ is a diagonal matrix with positive diagonal entries}\}.$$

Existence of this constant was first established by Dikin [7] and rediscovered by Vanderbei and Lagarias [8], Todd [9] and Stewart [10]. It is hard, in general, to compute this number, but if A is an integral matrix with input size L, then we can estimate $\bar{\chi}_A$ by $2^{O(L)}$. Below we summarize the VY algorithm.

Algorithm. Let $1 > \beta_2 > \beta_1 > 0$, $\bar{g} = g^* \equiv 64(1+\beta_1)n^2(\bar{\chi}_A + 1)$, and $k = 0$. Let $(x^0, y^0, s^0) \in N(\beta_1, \mu^0)$ be an initial interior-point.

Step 1: Let $(x,y,s) = (x^k, y^k, s^k)$, $\mu = \mu^k$, and $\delta = \sqrt{\mu} X^{-1/2} S^{1/2} e$. Determine the partition $(J_1,\ldots,J_p)$ for $\bar{g}$, and compute the LLS direction $(\delta x^*, \delta y^*, \delta s^*)$ for the partition $(J_1,\ldots,J_p)$. Apply a test for deciding whether or not to use the LLS direction. If so, then a step size α is computed (see [2]) and we set

$$(x', y', s') = (x,y,s) + \alpha(\delta x^*, \delta y^*, \delta s^*), \qquad \mu' = (1-\alpha)\mu.$$

$(x', y', s') \in N(\beta_2, \mu')$ is ensured theoretically. Otherwise, an ordinary predictor step (as in [6]) is taken.

Step 2: Compute a new iterate $(x^{k+1}, y^{k+1}, s^{k+1}) \in N(\beta_1, \mu')$ from the point $(x', y', s') \in N(\beta_2, \mu')$ such that $\mu^{k+1} = \mu'$.

Step 3: Increase k by 1 and go to Step 1.

The values of β_1 and β_2 used in [2] are 0.2 and 0.65. The initialization will be discussed later. The complexity analysis of the VY algorithm is based on the idea of crossover events.

Definition 1. Let $(x^1, y^1, s^1) \in N(\beta_1, \mu^1)$ and let $(x^2, y^2, s^2) \in N(\beta_1, \mu^2)$ be two points in the neighborhood of the central trajectory such that $\mu^1 > \mu^2$. If there exist dual slack variables s_i and s_j such that

$$s_i^1 \leqslant 3\bar{g}^n s_j^1 \tag{3}$$

and

$$s_i \geqslant 5\bar{g}^n s_j \text{ for all } (x, y, s) \in N(\beta_1, \mu) \text{ such that } 0 < \mu \leqslant \mu^2, \tag{4}$$

then we say a crossover event occurs between (x^1, y^1, s^1, μ^1) and (x^2, y^2, s^2, μ^2). We call $[\mu^1, \mu^2]$ the interval of the crossover event. [4]

Suppose that we have several crossover events. If the intervals of these events do not intersect, then we say that these crossover events are disjoint. Vavasis and Ye proved that there are at most $\frac{1}{2}n(n-1)$ disjoint crossover events.

The following theorem summarizes the main issues of the complexity analysis of the VY algorithm.

Theorem 1 (Based on Theorem 2 in [2] and [11]). *Let* $(x, y, s) \in N(\beta_1, \mu)$, *and let* $(x^+, y^+, s^+) \in N(\beta_1, \mu^+)$ *be the point obtained by performing one iteration of the VY algorithm. Let* $n(A) = \max\{n_1(A), n_3(A)\}$, *where* $n_1(A)$ *and* $n_3(A)$ *are as defined in* [2]. ($n(A) = O(n^{1.5}(\log \bar{\chi}_A + \log n))$.) *Then,*

(*i*) (x^+, y^+, s^+) *is an optimal solution, or*

(*ii*) *a crossover event occurs between* (x, y, s, μ) *and* (x', y', s', μ'), *where* (x', y', s', μ') *is any point such that* $(x', y', s') \in N(\beta_1, \mu')$ *and*

$$\mu' \leqslant \left(1 - \frac{c_0}{\sqrt{n}}\right)^{n(A)} \mu^+,$$

and c_0 *is a constant depending only on* β_1.

[4] This definition of the crossover event is slightly different from the original one by Vavasis and Ye [2]. Instead of (3) they used the condition that $\delta_i^1 \leqslant \bar{g}^n \delta_j^2$ (this implies (3) when $\beta = 0.2$), and instead of (4) they required that $s_i \geqslant 5\bar{g}^n s_j$ holds for all (x, y, s) on the central trajectory such that $0 < \mu \leqslant \mu^2$. But our modification does not make any substantial change in the arguments.

Proof. When an ordinary predictor step is taken from a point such that $(x, y, s) \in N(\beta_1, \mu)$, then $\mu^+ \leqslant (1 - c_0/\sqrt{n})\mu$ for some constant c_0. The execution and the analysis of the main loop of the VY algorithm is done by cases (cf. Section 5 of [2]). In their original proof they have three cases, case I–III. In a recent paper [11] they proved that case II never takes place, so we only count cases I and III. In case I the algorithm takes the ordinary predictor step, whereas in case III it takes the LLS step. From Vavasis and Ye's proof of their Theorem 2, the count is as follows. In case I, one crossover event occurs in $n_1(A)$ ordinary path-following interior-point steps. In case III, either an optimal solution is reached, or the crossover event occurs in one LLS step plus $n_3(A)$ ordinary steps. This observation and the definition of the crossover event imply the theorem. □

From the result above, the number of steps required by the algorithm [2] is bounded by $n(A)\times$(the number of disjoint crossover events). The value of $n(A)$ is about order of $n^{1.5}(\log \bar{\chi}_A + \log n)$. Since the number of the crossover events is $O(n^2)$, the algorithm solves the problem in $O(n^{3.5}(\log \bar{\chi}_A + \log n))$ steps, where each step requires $O(m^2 n)$ arithmetic operations.

As a by-product of their analysis, Vavasis and Ye proved that the number of disjoint crossover events is more tightly bounded by $\frac{1}{4}n^2$. This bound cannot be further improved, since Mizuno et al. [12] present a linear programming instance with $\frac{1}{8}n^2$ disjoint crossover events.

3.2. A modification

A drawback of the VY algorithm is that it relies on a number $\bar{\chi}_A$ which is difficult to compute or even estimate in a practical way. To overcome this difficulty, we do the following. We do not know the value of $\bar{\chi}_A$, but observe that it is used only for determining the correct partition of the set of indices. The first question is how many partitions can be generated when we vary the value of $\bar{g}$ from 1 to infinity. As we argue below, there can be at most n such partitions, which can easily be computed without knowing $\bar{\chi}_A$. One of these partitions is the right partition with which we can compute the LLS step of Vavasis and Ye. Thus, we can compute all the candidates for the correct LLS direction associated with each of the n patterns, and then take the step which decreases μ the most. Apparently, this idea increases the complexity of the algorithm by a factor of n since we compute n directions instead of one direction, but as we will show in Section 4, all of these directions can be computed in $O(nm^2)$ arithmetic operations, which is the same complexity of computing one LLS step.

Now, observe how the partition varies as we increase $\bar{g}$ from 1 to infinity. It is readily seen that the number of layers is decreasing and the change of layers occurs in such a way that two neighboring layers are merged into one larger layer, and for sufficiently large $\bar{g}$, the partition consists of one component, the entire index set $\{1, 2, \ldots, n\}$ itself. Thus, we have at most n partitions, one of which gives the LLS

search direction of Vavasis and Ye. To describe this observation mathematically, let q be the number of layers when $\bar{g} = 1$. We define $g_1, \ldots, g_q$ recursively as

$$g_q = 1,$$
$$g_{i-1} = \min\{\delta_{\pi(j+1)}/\delta_{\pi(j)} : \delta_{\pi(j+1)}/\delta_{\pi(j)} > g_i:\ j = 1, 2, \ldots, n-1\}$$
$$\text{for } i = q, q-1, \ldots, 2$$

until g_1 attains $\max\{\delta_{\pi(j+1)}/\delta_{\pi(j)}\}$. Then, g_i is the least value for which the number of the layers of the partition is i, and it remains unchanged as long as $\bar{g} \in [g_i, g_{i-1})$.

Now we are ready to state our algorithm.

Algorithm. Let $1 > \beta_2 > \beta_1 > 0$ and $k = 0$. Let $(x^0, y^0, s^0) \in N(\beta_1, \mu^0)$ be an initial interior point.

Step 1: Let $(x, y, s) = (x^k, y^k, s^k)$, $\mu = \mu^k$, and $\delta = \sqrt{\mu} X^{-1/2} S^{1/2} e$. Compute g_i $(i = 1, \ldots, q)$ as above. For each $i = 1, \ldots, q$, compute the partition $(J_1^i, J_2^i, \ldots, J_i^i)$ of $\{1, 2, \ldots, n\}$ for $\bar{g} = g_i$, an LLS direction $(\delta x^i, \delta y^i, \delta s^i)$ for the partition, and a step size α^i according to VY algorithm (regarding that $\bar{g}$ gives the correct partition). Let $\mu^i = (1 - \alpha^i)\mu$. Choose j which minimizes μ^i. Let $(x', y', s') = (x, y, s) + \alpha^j(\delta x^j, \delta y^j, \delta s^j)$ and $\mu' = \mu^j$.

Step 2: Compute a new iterate $(x^{k+1}, y^{k+1}, s^{k+1}) \in N(\beta_1, \mu^{k+1})$ from the point $(x', y', s') \in N(\beta_2, \mu')$ such that $\mu^{k+1} = \mu'$.

Step 3: Increase k by 1 and go to Step 1.

Thus, several LLS steps (directions) for various values of $\bar{g}$ are computed by the algorithm. Henceforth, the LLS step (direction) which uses the value g^* (which is used by the VY algorithm) is referred to as the "correct LLS step (direction)".

The complexity analysis summarized in Section 3.1 is based on the decrease of μ. Hence we can get at least the same complexity result as in Section 3.1 if we use a direction which assures the same or more reduction of μ instead of the correct LLS direction. Since Step 1 of our algorithm uses such a direction, the number of steps in our algorithm is not more than that of the VY algorithm. Furthermore, as we will show in Section 4, we can compute all the candidate search directions in $O(nm^2)$ arithmetic operations which is the same as computing a correct LLS step. Since the number of arithmetic operations to test each candidate is negligible compared with that of computing the search directions, we obtain the following result.

Theorem 2. *The worst-case complexity of our algorithm is at least as good as that of the VY algorithm* [2].

4. Computing all the candidates of the correct LLS step in $O(m^2n)$ arithmetic operations

In this section, we show that the number of arithmetic operations for computing all the (at most n) candidates for the correct LLS direction is asymptotically the same

as the number of arithmetic operations for computing the correct LLS step itself. Here arithmetic operations mean addition, subtraction, multiplication, division, comparison and taking square root.

4.1. Computing LLS step for given value of $\bar{g}$ by using a block lower triangular form (BLT form)

Let $(J_1, \ldots, J_q)$ be a partition of indices which determines the layers at the value $\bar{g}$. We also denote by A_{J_i} the submatrix of A associated with J_i. Let n_{J_i} be the number of variables in the ith layer s_{J_i}, and let r_{J_i} be the number such that

$$r_{J_i} = \mathrm{Rank}\begin{pmatrix} A_{J_1}^{\mathrm{T}} \\ \vdots \\ A_{J_i}^{\mathrm{T}} \end{pmatrix} - \mathrm{Rank}\begin{pmatrix} A_{J_1}^{\mathrm{T}} \\ \vdots \\ A_{J_{i-1}}^{\mathrm{T}} \end{pmatrix}$$

for all $i = 1, \ldots, q$. We call r_{J_i} the degree of freedom associated with the layer s_{J_i}. Note that r_{J_i} can be 0 if the dimension of the range space does not change by adding $A_{J_i}^{\mathrm{T}}$.

Now we introduce a representation of the range space of A^{T} suitable for computing the LLS step. We have the following theorem.

Theorem 3. *Let $(J_1, \ldots, J_q)$ be a partition, and let $J_1^*, J_2^*, \ldots, J_q^*$ be the index sets $\{1, \ldots, r_{J_1}\}, \{r_{J_1}+1, \ldots, r_{J_1}+r_{J_2}\}, \ldots, \{\sum_{i=1}^{q-1} r_{J_i}+1, \ldots, \sum_{i=1}^{q} r_{J_i}\}$. There exists a $q \times q$ block lower triangular matrix T partitioned row-wise with respect to $J_1, \ldots, J_q$ and column-wise with respect to $J_1^*, \ldots, J_q^*$, satisfying the following properties.*

1. Rank $T_{J_i J_i^*} = r_{J_i}$, i.e., *the columns of $T_{J_i J_i^*}$ are independent.*
2. *For all $k = 1, \ldots, q$, we have*

$$\mathrm{Im}\begin{pmatrix} A_{J_1}^{\mathrm{T}} \\ A_{J_2}^{\mathrm{T}} \\ \vdots \\ A_{J_k}^{\mathrm{T}} \end{pmatrix} = \mathrm{Im}\begin{pmatrix} T_{J_1 J_1^*} & 0 & \cdots & 0 \\ T_{J_2 J_1^*} & T_{J_2 J_2^*} & 0 & 0 \\ \vdots & \vdots & \vdots & 0 \\ T_{J_k J_1^*} & T_{J_k J_2^*} & \cdots & T_{J_k J_k^*} \end{pmatrix}.$$

Conventionally, we regard $T_{J_j J_i^*}$ as a block even if $r_{J_i} = 0$ (in such a case the block is an empty set).

We omit the proof of this theorem here, because it is an elementary argument of linear algebra. This representation can be computed within $O(nm^2)$ arithmetic operations, e.g., by using the techniques similar to LU decomposition.

Definition 2. We call the matrix T in Theorem 3 as a BLT form of A^{T} with respect to the partition $(J_1, \ldots, J_q)$.

The BLT form is a modification of a representation of $\mathrm{Im}(A^{\mathrm{T}})$ introduced in [13] for analyzing boundary behavior of the affine scaling algorithm for degenerate LP problems.

We write the ith block column of T as $T_{J_i^*}$. Then we have

$$T = \begin{pmatrix} T_{J_1^*} & \cdots & T_{J_q^*} \end{pmatrix}.$$

In terms of T, the tangent space of L_k^{D} is simply written as $\sum_{j=k+1}^{q} \mathrm{Im}(T_{J_k^*})$. Then, we have the following proposition.

Proposition 1. *Let $\delta s^{k-1} \in L_{k-1}^{\mathrm{D}}$ (δs^{k-1} may not be unique, but $\delta s_{J_l}^{k-1}$ $(l = 1, \ldots, k-1)$ is unique). Then L_{k-1}^{D} is written as follows.*

$$L_{k-1}^{\mathrm{D}} = \left\{ \delta s:\ \delta s = \delta s^{k-1} + \sum_{i=k}^{q} \mathrm{Im}(T_{J_i^*}) \right\}.$$

We define $\tilde{T} \equiv \Delta^{-1} T$, and write the ith block column of $\tilde{T}$ as $\tilde{T}_{J_i^*}$. Based on Proposition 1, the DLLS step can be computed as follows. Suppose that L_{k-1}^{D} is already determined and $\delta s^{k-1} \in L_{k-1}^{\mathrm{D}}$ is available. Then L_k^{D} is represented as

$$L_k^{\mathrm{D}} = \left\{ \text{minimizers } \delta s \text{ of } \|\Delta_{J_k}^{-1}(\delta s_{J^k} + s_{J^k})\| \text{ subject to } \delta s = \delta s^{k-1} + \sum_{i=k}^{q} T_{J_i^*} u_{J_i^*} \right\}.$$

Since $T_{J_k J_i^*} = 0$ for all $i = k+1, \ldots, q$, we see that the minimization problem involved in L_k^{D} is essentially an optimization problem with respect to $u_{J_k^*}$. Therefore, L_k^{D} is determined by computing a solution of the least squares problem

$$\underset{u_{J_k^*}}{\text{minimize}} \ \|\Delta_{J_k}^{-1}(\delta s_{J^k}^{k-1} + T_{J_k J_k^*} u_{J_k^*} + s_{J^k})\|.$$

Since $\tilde{T}_{J_k J_k^*}$ is column independent, the optimal solution is written as follows:

$$u_{J_k^*} = -(\tilde{T}_{J_k J_k^*}^{\mathrm{T}} \tilde{T}_{J_k J_k^*})^{-1} \tilde{T}_{J_k J_k^*}^{\mathrm{T}} \Delta_{J_k}^{-1}(\delta s_{J_k}^{k-1} + s_{J_k}).$$

Then $\delta s^k = \delta s^{k-1} + T_{J_k^*} u_{J_k^*}$ is an element of L_k^{D}. We can repeat this procedure increasing k one by one to the end. The procedure to compute the DLLS step is formally described as follows:

[Computation of the DLLS step δs^* at s]
begin
  $s^0 := s$; $\delta s^0 := 0$;
  for $i := 1$ **to** q **do**
    begin
      $u_{J_i^*} := -(\tilde{T}_{J_i J_i^*}^{\mathrm{T}} \tilde{T}_{J_i J_i^*})^{-1} \tilde{T}_{J_i J_i^*}^{\mathrm{T}} \Delta_{J_i}^{-1} s_{J_i}^{i-1}$;
      (We use the Cholesky factorization $\tilde{L}_{J_i^*} \tilde{L}_{J_i^*}^{\mathrm{T}}$ of $\tilde{T}_{J_i J_i^*}^{\mathrm{T}} \tilde{T}_{J_i J_i^*}$ to obtain $u_{J_i^*}$.)
      $\delta s^i := \delta s^{i-1} + T_{J_i^*} u_{J_i^*}$;
      ($\delta s_{J_j}^i (j = 1, \ldots, i-1)$ remains unchanged and $\delta s_{J_i}^i = \delta s_{J_i}^*$ holds.)

$s^i := s^0 + \delta s^i$

end

$\delta s^* := s^q - s^0$ $(= (\delta s^1_{J_1}, \ldots, \delta s^q_{J_q}))$

end

The PLLS step is computed as follows. We write the ith block row of T by T_{J_i}, then we have

$$T = \begin{pmatrix} T_{J_1} \\ \vdots \\ T_{J_q} \end{pmatrix}.$$

T^{T} is a block upper triangular matrix. We have the following proposition.

Proposition 2. *Let* $\delta x^{k+1} \in L^{\mathrm{P}}_{k+1}$. ($\delta x^{k+1}$ *may not be unique, but* $\delta x^{k+1}_{J_l}$ $(l = k+1, \ldots, q)$ *is unique.*) *Then* L^{P}_{k+1} *is written as follows.*

$$L^{\mathrm{P}}_{k+1} = \{\delta x\colon\ T^{\mathrm{T}}\delta x = 0,\ \delta x_{J_i} = \delta x^{k+1}_{J_i} \text{ for } i = k+1, \ldots, q\}$$
$$= \left\{ \delta x\colon\ \sum_{j=1}^{k} T^{\mathrm{T}}_{J_j}\delta x_{J_j} = -\sum_{j=k+1}^{q} T^{\mathrm{T}}_{J_j}\delta x^{k+1}_{J_j} \right\}.$$

Given $\delta x^{k+1} \in L^{\mathrm{P}}_{k+1}$, L^{P}_k is determined by computing displacement $\delta x^k_{J_k}$ associated with J_k. Due to Proposition 2 and the block upper triangular structure of T^{T}, we see that $\delta x^k_{J_k}$ is an optimal solution of the following optimization problem.

$$\text{minimize } \|\Delta_{J_k}(x_{J_k} + \delta x_{J_k})\|$$
$$\text{subject to } T^{\mathrm{T}}_{J_k J^*_k}\delta x_{J_k} = -\sum_{j=k+1}^{q} T^{\mathrm{T}}_{J_j J^*_k}\delta x^{k+1}_{J_j}.$$

Note that $\delta x^{k+1}_{J_j}$ is already determined if we determine $L^{\mathrm{P}}_q, L^{\mathrm{P}}_{q-1}, \ldots, L^{\mathrm{P}}_{k+1}$ backward. Based on these observations, the PLLS step can be computed according to the following procedure.

[Computation of the PLLS step δx^* at x]
begin
$v := 0$;
for $i := q$ **down to 1 do**
begin
$\delta x^*_{J_i} := -x_{J_i} + [\Delta^{-1}_{J_i}\tilde{T}_{J_i J^*_i}(\tilde{T}^{\mathrm{T}}_{J_i J^*_i}\tilde{T}_{J_i J^*_i})^{-1}(T^{\mathrm{T}}_{J_i J^*_i}x_{J_i} - v_{J^*_i})]$;
(We use the Cholesky factorization $\tilde{L}_{J^*_i}\tilde{L}^{\mathrm{T}}_{J^*_i}$ of $\tilde{T}^{\mathrm{T}}_{J_i J^*_i}\tilde{T}_{J_i J^*_i}$ to obtain $\delta x^*_{J_i}$.
Note that $v_{J^*_i} = \sum_{j=i+1}^{q} T^{\mathrm{T}}_{J_j J^*_i}\delta x^*_{J_j}$.)
$v := v + T^{\mathrm{T}}_{J_j}\delta x^*_{J_i}$
end
$\delta x^* := (\delta x^*_{J_1}, \ldots, \delta x^*_{J_q})$
end

The major work for computing the DLLS step and the PLLS step is the Cholesky factorization of $\tilde{T}^{\mathrm{T}}_{J_iJ_i^*}\tilde{T}_{J_iJ_i^*}$. Once this factorization is obtained for all those matrices, the steps are computed in $O(mn)$ arithmetic operations.

4.2. Computing the LLS steps efficiently when two layers are merged

In Section 4.1 we observed that given a partition J we can compute the LLS steps in $O(mn)$ arithmetic operations if (i) a BLT form T with respect to the partition J is available and (ii) the Cholesky factorization of $T^{\mathrm{T}}_{J_iJ_i^*}T_{J_iJ_i^*}$ are available for all the layers J_i.

Now, suppose that we increase $\bar{g}$ and that two layers J_i and J_{i+1} are joined. Then we have a new partition $J' = (J'_1,\ldots,J'_j,\ldots,J'_{q-1}) = (J_1,\ldots,J_i\cup J_{i+1},\ldots,J_q)$ and the number of layers is $q-1$.

In this section, we assume that (i) a BLT form T with respect to J and (ii) the Cholesky factorization of $\tilde{T}^{\mathrm{T}}_{J_iJ_i}\tilde{T}_{J_iJ_i}$ for all $i=1,\ldots,q$ are available, and show how we can compute efficiently (i) a BLT form T' with respect to the new partition J' and (ii) the Cholesky factorization of $\tilde{T}'^{\mathrm{T}}_{J'_iJ'^*_i}\tilde{T}'_{J'_iJ'^*_i}$ for all $J'=1,\ldots,q-1$.

By definition of the BLT form, we have the following proposition.

Proposition 3. *The matrix T is also a BLT form with respect to the new partition J' by regarding the block rows associated with J_i and J_{i+1} as one block row associated with $J_i\cup J_{i+1}$ and by regarding the block columns associated with J_i^* and J_{i+1}^* as one block column associated with $J_i^*\cup J_{i+1}^*$.*

Now we estimate the complexity of computing the Cholesky factorizations. It is easy to see that $\tilde{T}_{J'_jJ'^*_j}=\tilde{T}_{J_jJ_j^*}$ for $j=1,\ldots,i-1$ and

$$\tilde{T}_{J'_iJ'^*_i}=\tilde{T}_{J_i\cup J_{i+1},J_i^*\cup J_{i+1}^*}=\begin{pmatrix}\tilde{T}_{J_iJ_i^*} & 0\\ \tilde{T}_{J_{i+1}J_{i+1}^*} & \tilde{T}_{J_{i+1}J_{i+1}^*}\end{pmatrix},$$

and $\tilde{T}_{J'_jJ'^*_j}=\tilde{T}_{J_{j+1}J_{j+1}^*}$ for $j=i+1,\ldots,q-1$. Since the Cholesky factorization of $\tilde{T}^{\mathrm{T}}_{J_iJ_i^*}\tilde{T}_{J_iJ_i^*}$ is already available for all $i=1,\ldots,q$, it is only necessary to compute the Cholesky factorization of $\tilde{T}^{\mathrm{T}}_{J_i\cup J_{i+1},J_i^*\cup J_{i+1}^*}\tilde{T}_{J_i\cup J_{i+1},J_i^*\cup J_{i+1}^*}$ for computing the new LLS steps associated with the new partition. We can efficiently compute the Cholesky factorization by using the previous factorizations and the rank-one update of the Cholesky factorization [14,15].

Lemma 1. *Let W be the matrix of the form*

$$W\equiv\begin{pmatrix}W_{11} & 0\\ W_{21} & W_{22}\end{pmatrix},$$

where W_{11} a $q_1 \times p_1$ matrix with independent columns, W_{21} a $q_2 \times p_1$ matrix, and W_{22} is a $q_2 \times p_2$ matrix with independent columns. If the Cholesky decompositions of $W_{11}^{\mathrm{T}} W_{11}$ and $W_{22}^{\mathrm{T}} W_{22}$ are already available, we can compute the Cholesky decomposition of $W^{\mathrm{T}} W$ in

$$O(p_1^2 q_2 + p_1 p_2 q_2 + \tfrac{1}{2} p_1^2 p_2 + p_1 p_2^2)$$

arithmetic operations.

Proof. Let LL^{T} be the Cholesky factorization of W, where

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix},$$

L_{11} is a $p_1 \times p_1$ lower triangular matrix, L_{21} a $p_2 \times p_1$ matrix, and L_{22} a $p_2 \times p_2$ lower triangular matrix. Writing the relation $W^{\mathrm{T}} W = LL^{\mathrm{T}}$ by components, we have

$$\begin{aligned} L_{11} L_{11}^{\mathrm{T}} &= W_{11}^{\mathrm{T}} W_{11} + W_{21}^{\mathrm{T}} W_{21} \\ L_{21} L_{11}^{\mathrm{T}} &= W_{22}^{\mathrm{T}} W_{21} \\ L_{21} L_{21}^{\mathrm{T}} + L_{22} L_{22}^{\mathrm{T}} &= W_{22}^{\mathrm{T}} W_{22}. \end{aligned} \tag{5}$$

Since the right-hand side of (5) is the sum of $W_{11}^{\mathrm{T}} W_{11}$ and q_2 rank-one matrices and the Cholesky factorization of $W_{11}^{\mathrm{T}} W_{11}$ is already available, we can compute L_{11} in $O(q_2 p_1^2)$ arithmetic operations by repeating q_2 times the rank-one update of the Cholesky factorization [14,15].

Next, we compute L_{21}. We compute $W_{21}^{\mathrm{T}} W_{22}$, and then solve the systems of linear equations p_2 times repeatedly to obtain each column of L_{21}^{T}, where L_{11} is the coefficient matrix and each column of $W_{21}^{\mathrm{T}} W_{22}$ is the right-hand side. This requires $O(p_1 p_2 q_2)$ arithmetic operations for computing $W_{21}^{\mathrm{T}} W_{22}$ and $O(p_2 p_1^2/2)$ arithmetic operations for solving the systems of linear equations.

Finally, we obtain L_{22} according to the formula $L_{22} L_{22}^{\mathrm{T}} = W_{22}^{\mathrm{T}} W_{22} - L_{21} L_{21}^{\mathrm{T}}$ by p_1 times rank-one updates of the Cholesky factorization. This requires $O(p_1 p_2^2)$ arithmetic operations. In total, we can compute the Cholesky factorization of $W^{\mathrm{T}} W$ in

$$O(q_2 p_1^2 + p_1 p_2 q_2 + \tfrac{1}{2} p_2 p_1^2 + p_1 p_2^2)$$

arithmetic operations. This completes the proof. □

Applying Lemma 1 with $W = \tilde{T}_{J_j \cup J_{j+1}, J_j^* \cup J_{j+1}^*}$, we see that the Cholesky factorization of $\tilde{T}^{\mathrm{T}}_{J_j \cup J_{j+1}, J_j^* \cup J_{j+1}^*} \tilde{T}_{J_j \cup J_{j+1}, J_j^* \cup J_{j+1}^*}$ can be computed in

$$O(n_{J_{j+1}} r_{J_j}^2 + n_{J_{j+1}} r_{J_j} r_{J_{j+1}} + \tfrac{1}{2} r_{J_{j+1}} r_{J_j}^2 + r_{J_j} r_{J_{j+1}}^2)$$

arithmetic operations. Thus we have the following theorem.

Theorem 4. *Let $J \equiv (J_1, \ldots, J_i, J_{i+1}, \ldots, J_q)$ be a partition, and T be a BLT form with respect to J. Let $J' \equiv (J_1, \ldots, J_{i-1}, J_i \cup J_{i+1}, J_{i+2}, \ldots, J_q)$. If the Cholesky factorization*

of $\tilde{T}^{\mathrm{T}}_{J_jJ^*_j}\tilde{T}_{J_jJ^*_j}$ $(j=1,\ldots,q)$ *is already computed, the Cholesky factorization of* $\tilde{T}^{\mathrm{T}}_{J'_jJ'^*_j}\tilde{T}_{J'_jJ'^*_j}$ $(j=1,\ldots,q-1)$ *for* J' *can be computed with*

$$O(n_{J_{i+1}}r^2_{J_i}+n_{J_{i+1}}r_{J_i}r_{J_{i+1}}+\tfrac{1}{2}r_{J_{i+1}}r^2_{J_i}+r_{J_i}r^2_{J_{i+1}})$$

arithmetic operations.

4.3. Computing all the candidates of the correct LLS step and its complexity

Now we consider applying the procedure of the previous sections repeatedly, as we increase $\bar{g}$, when a merge of two layers occurs. Let $(\hat{J}_1,\ldots,\hat{J}_q)$ be the initial partition determined by $\bar{g}=1$ in our algorithm. We can compute all the candidates for the LLS step according to the following procedure.

[Computation of the candidates of the correct LLS step]
begin
  $J\equiv(J_1,\ldots,J_q):=(\hat{J}_1,\ldots,\hat{J}_q)$;
  Compute a BLT form T with respect to the partition J;
  Compute the Cholesky factorization $\tilde{L}^{\mathrm{T}}_{J_j}\tilde{L}_{J^*_j}$ of $\tilde{T}^{\mathrm{T}}_{J_jJ^*_j}\tilde{T}_{J_jJ^*_j}$ for $j:=1,\ldots,q$;
  Compute DLLS step for J;
  Compute PLLS step for J;
  for $k:=q$ **down to 2 do**
  begin
    Determine the two layers J_i and J_{i+1} to be merged;
    Merge the two layers, i.e., $J':=(J_1,\ldots,J_i\cup J_{i+1},\ldots,J_k)$;
    Compute the Cholesky factorization $\tilde{L}^{\mathrm{T}}_{J^*_i\cup J^*_{i+1}}\tilde{L}_{J^*_i\cup J^*_{i+1}}$ of
    $\tilde{T}^{\mathrm{T}}_{J_i\cup J_{i+1},J^*_i\cup J^*_{i+1}}\tilde{T}_{J_i\cup J_{i+1},J^*_i\cup J^*_{i+1}}$;
    Compute DLLS step for J';
    Compute PLLS step for J';
    $J:=J'$
  end
end

A merge of two layers occurs at most n times while $\bar{g}$ is increased from 1 to infinity. The total number of arithmetic operations to compute the q directions is estimated as follows.

Theorem 5. *The q candidates for the LLS step can be computed in* $O(m^2n)$ *arithmetic operations.*

Proof. The major part in the algorithm is:
1. Initial computation of a BLT form with respect to the layer $(\hat{J}_1,\ldots,\hat{J}_q)$ and the Cholesky factorization of $\tilde{T}^{\mathrm{T}}_{J_iJ^*_i}\tilde{T}_{J_iJ^*_i}$ $(i=1,\ldots,q)$;

2. Updates of the Cholesky factorization in the main "for" loop;
3. Computation of the DLLS step and the PLLS step.

It is not difficult to see that the number of arithmetic operations for 1 is $O(m^2n)$.

The number of arithmetic operations for 3 is $O(m^2n)$ because, given the Cholesky factorizations, it takes $O(mn)$ arithmetic operations to compute one set of LLS directions, and at most m directions are computed throughout the process, even though the merge process occurs $q \geqslant m$ times.

Finally, the number of arithmetic operations in executing 2 is bounded as follows. For this purpose, we consider a binary tree where each node represents a layer generated in the merge process. A layer α with children α_1 and α_2 is obtained by merging α_1 and α_2. Let $n(\alpha)$ be the number of the variables in α, and let $r(\alpha)$ be the degree of freedom of α, which is defined by $\mathrm{Rank}(\tilde{T}_{\alpha\alpha})$. The root of this tree represents the layer consisting of all the variables. The leaves of the tree represent the initial layers $\hat{J}_1, \ldots, \hat{J}_q$. Now, let $C(\alpha)$ be the total number of arithmetic operations for computing the Cholesky factorization of $\tilde{T}_{\alpha\alpha}^{\mathrm{T}}\tilde{T}_{\alpha\alpha}$. From the previous arguments, we see that, if α has children α_1 and α_2, then

$$C(\alpha) \leqslant C(\alpha_1) + C(\alpha_2) + M_1(n_2r_1^2 + r_1r_2n_2 + \tfrac{1}{2}r_2r_1^2 + r_1r_2^2),$$

where $n_2 = n(\alpha_2), r_1 = r(\alpha_1)$ and $r_2 = r(\alpha_2)$ and M_1 is a positive constant. On the other hand, if α is a leaf, we have

$$C(\alpha) \leqslant M_2 n(\alpha) r(\alpha)^2, \tag{6}$$

where M_2 is a positive constant, as we form $\tilde{T}_{\alpha\alpha}^{\mathrm{T}}\tilde{T}_{\alpha\alpha}$ and then factorize it directly.

We will show that $C(\alpha) \leqslant \max\{M_1, M_2\} n(\alpha) r(\alpha)^2$ for every α. Then, $C(\{1, \ldots, n\})$, which is the total number of arithmetic operations for updating $\tilde{T}_{\alpha\alpha}^{\mathrm{T}}\tilde{T}_{\alpha\alpha}$ for all α, is bounded by $\max\{M_1, M_2\} nm^2$, and we are done.

The proof is by induction. Observe first that the claim holds for every leaf because of Eq. (6). Now pick α with children α_1 and α_2, and show that the claim always holds for a layer α if it holds for α_1 and α_2. Then we have $C(\{1, \ldots, n\}) \leqslant \max\{M_1, M_2\} nm^2$ as asserted.

By definitions and the assumptions and Theorem 4 and $r_1 \leqslant n_1$, we have

$$\begin{aligned} C(\alpha) &= C(\alpha_1) + C(\alpha_2) + M_1(n_2r_1^2 + r_1r_2n_2 + \tfrac{1}{2}r_2r_1^2 + r_1r_2^2) \\ &\leqslant \max\{M_1, M_2\}(n_1r_1^2 + n_2r_2^2 + n_2r_1^2 + r_1r_2n_2 + \tfrac{1}{2}r_2r_1^2 + r_1r_2^2), \\ &\leqslant \max\{M_1, M_2\}(n_1 + n_2)(r_1 + r_2)^2. \end{aligned}$$

This completes the proof. □

5. Initiating the algorithm

We describe how to initialize the algorithm. Here we need a bound on the norm of an optimal solution. Following the approach of Vavasis and Ye, we consider the following problem:

$$\begin{aligned} &\text{minimize} && c^{\mathrm{T}}x + Ne^{\mathrm{T}}x_2 \\ &\text{subject to} && \begin{pmatrix} 0 & A & -A \\ I & I & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x \\ x_2 \end{pmatrix} = \begin{pmatrix} b \\ 2Ne \end{pmatrix} \\ & && x_1, x, x_2 \geqslant 0 \end{aligned} \tag{7}$$

and its dual

$$\begin{aligned} &\text{maximize} && b^{\mathrm{T}}y + 2Ne^{\mathrm{T}}y_1 \\ &\text{subject to} && \begin{pmatrix} 0 & I & I & 0 & 0 \\ A^{\mathrm{T}} & I & 0 & I & 0 \\ -A^{\mathrm{T}} & 0 & 0 & 0 & I \end{pmatrix} \begin{pmatrix} y \\ y_1 \\ s_1 \\ s \\ s_2 \end{pmatrix} = \begin{pmatrix} 0 \\ c \\ Ne \end{pmatrix} \\ & && s_1, s, s_2 \geqslant 0. \end{aligned} \tag{8}$$

Vavasis and Ye show that if we set $N > (1 + \bar{\chi}_A)\|c\|$ and $N > \bar{\chi}_A\|d\|$, where d is a solution of $Ad = b$, then this extended problem is equivalent to the original problem with a trivial well-centered initial solution.

Instead of the assumption that the value of $\bar{\chi}_A$ is available, we assume that upper bounds C_1, C_2 on the norm of optimal solutions for (1) and (2) are available. Due to complementarity, if we let $N > \max\{C_1, C_2\}$, then the variables s_2 and x_1 are guaranteed to be positive at an optimal solution, which implies that their counterparts x_2 and s_1 are always zero on the optimal set. Thus, the (x, y, s)-part of an optimal solution of (7) and (8) is an optimal solution of the original problem whenever it has an optimal solution. Let $\tilde{A}$ be the coefficient matrix of (7). It was shown by Vavasis and Ye that $\bar{\chi}_{\tilde{A}} \leqslant 3\sqrt{2}(1 + \bar{\chi}_A)$. Thus, applying our algorithm to this extended problem, we are able to solve the problems (7) and (8) in $O(n^{3.5}(\log \bar{\chi}_A + \log n))$ steps, where each step requires $O((m+n)^2 n)$ arithmetic operations.

If we do not have any upper bound on $\bar{\chi}_A$, then we may resort to the following technique, which is a modification of the technique suggested by Renegar [2]. Let $g(k)$ be a function which is easily computed and very rapidly increasing and run the algorithm by letting $N = g(k)\max\{\|c\|, \|d\|\}$. Soon after N becomes greater than $\max\{(1 + \bar{\chi}_A)\|c\|, \bar{\chi}_A\|d\|\}$, we can solve the problem.

This modification costs a factor of $g^{-1}(\max\{(1 + \bar{\chi}_A), \bar{\chi}_A\})$ in complexity. In the original form of VY algorithm, we cannot take $g(k)$ as a function more quickly increasing than $O(2^{2^k})$, because we have to estimate $\bar{\chi}_A$ with certain accuracy. If $g(k)$ grows too quickly, it affects the complexity argument in their algorithm. But in our modification, we do not have this kind of concern. We can take any $g(k)$ which is quickly increasing, as long as the complexity of computing $g(k)$ is not greater than the work for executing another part of the algorithm. If we take g to be a function

defined recursively by $g(k+1) = 2^{g(k)}$ $(g(1) = 1)$ and run the algorithm by letting $N = g(k)$, for example, then our algorithm has a complexity of terminating within $O(\log^* \bar{\chi}_A \cdot n^{3.5}(\log \bar{\chi}_A + \log n))$ steps where each step requires $O(n(m+n)^2)$ arithmetic operations. This is a slight improvement on the complexity of VY algorithm (with Renegar's modification) when $\bar{\chi}_A$ is unknown.

Our algorithm may have a similar advantage over theirs when we resolve the problem by estimating an upper bound for $\bar{\chi}_A$. While they need an estimate which is "accurate" to some extent, any upper bound is enough in our approach.

6. Concluding remarks

We have proposed a modification of the VY algorithm, eliminating the need for $\bar{\chi}_A$ in computing the search direction. There are several topics for further research.

The first issue to be dealt with is sparsity. To solve large-scale problems, we have to develop an implementation which preserves sparsity of the original A. Since VY algorithm deals with layers, we have less freedom if we wish to exploit sparsity. But yet we have some freedom to exchange rows of A^T within the same layer, without changing the result of our paper. It is an interesting question to find the best way to preserve sparsity when making the BLT form. Developing techniques to preserve sparsity in the Cholesky factorization is also important.

We could remove $\bar{\chi}_A$ in computing search direction, but still we need it if neither an upper bound for a norm of a primal–dual optimal solution nor a well-centered primal–dual feasible solution is available. So, eliminating $\bar{\chi}_A$ from the initialization phase of the algorithm remains a challenging interesting question. As was mentioned in Section 5, any cheap way of computing an upper bound for $\bar{\chi}_A$ suffices for our purpose.

References

[1] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4 (1984) 373–395.
[2] S. Vavasis, Y. Ye, A primal-dual accelerated interior point method whose running time depends only on A, Mathematical Programming 74 (1996) 79–120.
[3] É. Tardos, A strongly polynomial algorithm to solve combinatorial linear programs, Operations Research 34 (1986) 250–256.
[4] M. Kojima, S. Mizuno, A. Yoshise, A polynomial-time algorithm for a class of linear complementary problems, Mathematical Programming 44 (1989) 1–26.
[5] R.D.C. Monteiro, I. Adler, Interior path following primal-dual algorithms. Part I: Linear programming, Mathematical Programming 44 (1989) 27–41.
[6] S. Mizuno, M.J. Todd, Y. Ye, On adaptive-step primal-dual interior-point algorithms for linear programming, Mathematics of Operations Research 18 (1993) 964–981.
[7] I.I. Dikin, V.I. Zorkalcev, Iterative Solution of Mathematical Programming Problems: Algorithms for the Method of Interior Points (in Russian), Nauka, Novosibirsk, USSR, 1980.
[8] R.J. Vanderbei, J.C. Lagarias, I.I. Dikin's convergence result for the affine-scaling algorithm, Contemporary Mathematics 114 (1990) 109–119.

[9] M.J. Todd, A Dantzig-Wolfe-like variant of Karmarkar's interior-point linear programming algorithm, Operations Research 38 (1990) 1006–1018.
[10] G.W. Stewart, On scaled projections and pseudoinverses, Linear Algebra and its Applications 112 (1989) 189–193.
[11] S. Vavasis, Y. Ye, A simplification to a primal–dual interior point method whose running time depends only on the constaint matrix, Technical Note, Department of Management Science, University of Iowa, Iowa City (1997).
[12] S. Mizuno, N. Megiddo, T. Tsuchiya, A linear programming instance with many crossover events, Journal of Complexity 12 (1996) 474–479.
[13] T. Tsuchiya, Global convergence property of the affine scaling method for primal degenerate linear programming problems, Mathematics of Operations Research 17 (1992) 527–557.
[14] J.M. Bennett, Triangular factors of modified matrices, Numerische Mathematik 7 (1965) 217–221.
[15] P.E. Gill, G.H. Golub, W. Murray, M.A. Saunders, Methods for modifying matrix factorization, Mathematics of Computation 28 (1974) 505–535.