A strongly polynomial algorithm for linear optimization problems having 0-1 optimal solutions

Sergei Chubanov[1]

Institute of Information Systems at the University of Siegen, Germany

January 23, 2014

Abstract

We present a strongly polynomial algorithm for linear optimization problems of the form $\min\{cx|Ax = b, x \geq \mathbf{0}\}$ having 0-1 vectors among their optimal solutions. The algorithm runs in time $O(n^4 \max\{m, \log n\})$, where n is the number of variables an m is the number of equations. The algorithm also constructs necessary and sufficient optimality conditions for 0-1 solutions in the form of a linear system.

Keywords: Linear programming, strongly polynomial algorithm.

1 Introduction

The algorithm presented in this paper solves a linear optimization problem $\min\{cx|Ax = b, x \geq \mathbf{0}\}$ in strongly polynomial time, provided that this problem has a 0-1 optimal solution, i.e., a solution whose components are zeros and ones. (Of course, the optimal set may also

[1]*Chubanov, S.; Universität Siegen, Fakultät III, Hölderlinstr. 3, 57068 Siegen, Deutschland. e-mail: sergei.chubanov@uni-siegen.de*

contain solutions that are not 0-1 vectors.) The assignment problem and the minimum-cut problem are among well-known examples of combinatorial problems that are solvable in strongly polynomial time and can be represented in the above form. Their 0-1 optimal solutions are characteristic vectors of the respective combinatorial structures. In this paper we show that the property that there exists a 0-1 optimal solution is sufficient for the existence of a strongly polynomial algorithm for a linear optimization problem of the above form. That is, we do not need to impose any additional restrictions on A, b, and c. One should also emphasize that we do not use any traditional polynomial algorithm.

One of the distinguishing features of our algorithm is that it constructs necessary and sufficient optimality conditions for 0-1 solutions. These conditions have the form of a system $x_j = 0$, $j \in J_0$; $x_j = 1$, $j \in J_1$. A 0-1 feasible solution x is optimal if and only if it satisfies these equations. That it, the conditions say what variables have the same values for all 0-1 optimal solutions and what values these variables must take. Note that the strongly polynomial algorithms presented by the author in [2] and [3] are only able to find a solution of a system $Ax = b, x \geq \mathbf{0}$, or prove that no 0-1 solutions exist. These algorithms do not directly imply an analogous result for the optimization problem. However, they are used for finding an optimal solution at the second phase of the algorithm presented in this paper. The running time $O(n^4 \max\{m, \log n\})$ of our algorithm, where n is the number of variables and m is the number of equations, seems to be better than one we can obtain by traditional linear-programming algorithms if we apply them to perform the same task. Indeed, if we directly applied a traditional polynomial algorithm for linear programming to construct the necessary and sufficient optimality conditions for 0-1 solutions then we would need $O(n)$ calls to this algorithm. This would lead to the running time $O(n^4 size(A, b, c))$, where $size(A, b, c)$ is the size of the instance, because $O(n^3 size(A, b, c))$ is the best known theoretical estimate of the running time of a linear-programming algorithm. (See Renegar [7].) Since $size(A, b, c)$ is greater than nm, the running time $O(n^4 \max\{m, \log n\})$ of our algorithm seems to be more preferable in our case.

Actually, the algorithm presented in this paper can be converted to a polynomial algorithm for linear programming. Therefore, applying Tardos' method [8], we can make our algorithm strongly polynomial in the case when the size of A is polynomial in n.

2 Alternating projections and upper bounds on optimal values

We consider a linear optimization problem

$$\min\{cx|Ax = b, x \geq \mathbf{0}\},$$

where A is an m by n matrix of rank m, the vector c is a row vector, and b is a column vector. Denote

$$L = \{x|Ax = b\}$$

and

$$L(\xi) = L \cap \{x|cx \leq \xi\}.$$

If ξ is an upper bound on the optimal value, then we will say that ξ is achievable. Our algorithm maintains a point x^k in $L(\xi)$ for each facet B^k, $k = 1, \ldots, 2n$, of the unit cube $B = \{x \mid \mathbf{0} \leq x \leq \mathbf{1}\}$. (The facets are listed in an arbitrary order.) Each point x^k is initialized with the projection of $\mathbf{0}$ on $L(\xi)$, where ξ is an initial achievable value. This guarantees that the initial distance between x^k and any 0-1 feasible solution is bounded by $\sqrt{n}$ because the distance between $\mathbf{0}$ and any 0-1 solution is not greater than $\sqrt{n}$ and the projection of $\mathbf{0}$ can only bring $\mathbf{0}$ closer to a 0-1 feasible solution. For each k, the algorithm also maintains an upper bound u_k on the square of the distance between x^k and any point in $L(\xi) \cap B^k$. That is, u_k is an upper bound on the square of the distance between x^k and $L(\xi) \cap B^k$, if this set is not empty, and, at the same time, on the square of the distance from x^k to every closed subset of $L(\xi) \cap B^k$. These bounds are improved in the course of the algorithm.

Our algorithm is based on the fact that if $L(\xi) \cap B = \emptyset$ then there is B^k such that the distance between B^k and every point in $L(\xi)$ is greater than $1/\sqrt{n}$. Let us suppose that there exists x^k whose distance to B^k is not less than $1/(\gamma\sqrt{n})$, where $\gamma > 1$ is a constant.

Then the projection of x^k onto B^k and after that back onto $L(\xi)$ reduces the square of the distance between x^k and $L(\xi) \cap B^k$ by at least $1/(\gamma^2 n)$, provided that $L(\xi) \cap B^k$ is not empty. That is, having an upper bound $u_k \leq n$ on the square of the distance, we can reduce it by at least $1/(\gamma^2 n)$. If u_k becomes negative, then $L(\xi) \cap B^k$ is empty. In this case all 0-1 optimal solutions must lie in the facet that is opposite to B^k. So the number of such steps with respect to the same facet B^k is bounded by $O(n^2)$. If all the distances between B^k and x^k are less than $1/(\gamma\sqrt{n})$, then $L(\xi) \cap B$ is not empty, which means that ξ is achievable. In this case, as we will see later, the ξ can be reduced so that the obtained new value ξ is achievable and has the property that the distance between the projection of x^k on $L(\xi)$ and B^k is not less than $1/(\gamma\sqrt{n})$. Then we can replace the points x^k by their projections on $L(\xi)$, where ξ is the new achievable value, and continue the above process.

Let $\pi_S(x)$ denote the projection of a point x on a closed convex set S. The composition of two projections π_{S_1} and π_{S_2} is denoted by $\pi_{S_2}\pi_{S_1}$. It is clear that consecutive projections onto closed convex sets containing a given set generates a Fejér-monotone sequence with respect to this set, which means that the distance from each point in this sequence to each point x of this set is not greater than the distance to x from the previous point in the sequence. Moreover,

$$\|\pi_S(x^0) - x\|^2 \leq \|x^0 - x\|^2 - \|\pi_S(x^0) - x^0\|^2$$

for each point x^0 and for all x in S. This well-known fact is implied by the following lemma that is often used to prove the convergence of the alternating projections, which is a very popular method for convex feasibility problems that dates back to von Neumann (see, for instance, Dattorro [4] for more details), and of the closely related relaxation method for linear inequalities (see Agmon [1] and Motzkin and Schoenberg [6]):

Lemma 2.1 *Let a hyperplane H separate a point x^0 from a set Q. Then*

$$\|\pi_H(x^0) - x\|^2 \leq \|x^0 - x\|^2 - \|\pi_H(x^0) - x^0\|^2$$

for all x in Q.

Proof. Let H' contain x of Q and be parallel to H. Then

$$\|x^0 - x\|^2 = \|\pi_{H'}(x^0) - x^0\|^2 + \|\pi_{H'}(x^0) - x\|^2.$$

On the other hand,

$$\|\pi_H(x^0) - x\|^2 = \|\pi_{H'}(x^0) - \pi_H(x^0)\|^2 + \|\pi_{H'}(x^0) - x\|^2.$$

It follows that

$$\|\pi_H(x^0) - x\|^2 - \|x^0 - x\|^2 = \|\pi_{H'}(x^0) - \pi_H(x^0)\|^2 - \|\pi_{H'}(x^0) - x^0\|^2 \leq -\|\pi_H(x^0) - x^0\|^2.$$

■

The orthogonal projection onto the affine subspace L is computed as

$$\pi_L(x) = x - A^T(AA^T)^{-1}Ax + A^T(AA^T)^{-1}b.$$

(The matrix AA^T is invertible because A has full rank.) The following matrix P is the matrix of the orthogonal projection onto the linear subspace $\{x|Ax = \mathbf{0}\}$:

$$P = I - A^T(AA^T)^{-1}A.$$

That is,

$$\pi_L(x) = Px + A^T(AA^T)^{-1}b.$$

Note that both P and $A^T(AA^T)^{-1}$ can be computed in $O(n^3)$ time by Gaussian elimination (see Edmonds [5] and Renegar [7]). If these matrices are known, the projection $\pi_L(x)$ is computable in $O(nm)$ time.

We also need to compute projections onto the set $L(\xi)$. A point x in L belongs to $L(\xi)$ if and only if

$$cPx \leq \xi - cA^T(AA^T)^{-1}b.$$

Let $H(\xi)$ denote the half-space consisting of points satisfying this inequality. Note that $L(\xi) = L \cap H(\xi)$. We have

$$\pi_{L(\xi)}(x) = \pi_{L\cap H(\xi)}(x) = \pi_{H(\xi)}\pi_L(x) = \pi_L\pi_{H(\xi)}(x).$$

The latter equation means that π_L and $\pi_{H(\xi)}$ commute.

Lemma 2.2 *If ξ is not achievable, then there exists B^k such that the distance between B^k and every point x^0 in $L(\xi)$ is greater than $\frac{1}{\sqrt{n}}$.*

Proof. By the separation theorem for convex sets, there exists a row vector h with $hx^0 < \min\{hx | x \in [0,1]^n\}$ for all x^0 in $L(\xi)$. Let x^* be a minimum of hx over $[0,1]^n$. We have $x^*_j = 1$ for all negative components h_j. For all positive h_j we must have $x^*_j = 0$. Consider $s \in \arg\max_j |h_j|$. If $h_s > 0$ then set $\sigma = 1$, else set $\sigma = 0$. Note that $h_j(x_j - x^*_j) \geq 0$ for all j and x in $[0,1]^n$. For all x in $[0,1]^n$ with $x_s = \sigma$ we have

$$\frac{hx - hx^*}{\|h\|} \geq \frac{h_s(\sigma - x^*_s)}{\|h\|} = \frac{|h_s|}{\|h\|} \geq \frac{1}{\sqrt{n}}.$$

That is, the distance between $\{x | hx = hx^*\}$ and the facet $[0,1]^n \cap \{x | x_s = \sigma\}$ is not less than $\frac{1}{\sqrt{n}}$. Since $\{x | hx = hx^*\}$ separates $L(\xi)$ from the unit cube, it follows that the distance between this facet and every point x^0 in $L(\xi)$ is greater than $\frac{1}{\sqrt{n}}$.

■

Note that if cP is a zero vector, then all feasible solutions of the problem are optimal. Otherwise, the set $L(\xi')$ is nonempty for every value ξ'. Assuming that cP is nonzero, for each k we consider the system

$$\begin{aligned} &\xi' \leq cx^k, \\ &\|\pi_{L(\xi')}(x^k) - \pi_{B^k}\pi_{L(\xi')}(x^k)\|^2 \leq \tfrac{1}{n}, \end{aligned}$$

with respect to a variable ξ'. Since x^k belongs to L, for every $\xi' \leq cx^k$ we have

$$\pi_{L(\xi')}(x^k) = \pi_{H(\xi')}(x^k) = x^k - \frac{cx^k - \xi'}{\|cP\|^2} Pc^T,$$

which means that the system can be written as

$$\begin{aligned} &\xi' \leq cx^k, \\ &\left\| x^k - \tfrac{cx^k - \xi'}{\|cP\|^2} Pc^T - \pi_{B^k}\left(x^k - \tfrac{cx^k - \xi'}{\|cP\|^2} Pc^T \right) \right\|^2 \leq \tfrac{1}{n}. \end{aligned}$$

That is, if cP is already computed, in linear time we can check if ξ' is feasible.

In the theorem below, we assume that each point x^k satisfies the condition

$$\|x^k - \pi_{B^k}(x^k)\|^2 \leq \frac{1}{n}. \tag{1}$$

Under this condition, the above systems are feasible because then $\xi' = cx^k$ are feasible solutions. In this case, let ξ_k^* denote the minimum value of ξ' that satisfies the respective system. Denote $\xi^* = \max \xi_k^*$. We have the following theorem:

Theorem 2.1 *Let each of the points $x^k, k = 1, \ldots, 2n$, belong to L and satisfy the inequality (1). The value ξ^* is achievable.*

Proof. It follows from (1) that the feasible sets of the above systems are the segments $[\xi_k^*, cx^k]$. If $cx^k < \xi^*$, then x^k belongs to $L(\xi^*)$ and $\pi_{L(\xi^*)}(x^k) = x^k$. Otherwise, $\xi' = \xi^*$ is feasible for the respective system because at the same time $\xi^* \geq \xi_k^*$. It follows that for every k we have

$$\|\pi_{L(\xi^*)}(x^k) - \pi_{B^k}\pi_{L(\xi^*)}(x^k)\|^2 \leq \frac{1}{n},$$

i.e., the square of the distance from $\pi_{L(\xi^*)}(x^k)$ to B^k is not greater than $1/n$ for every k. Now we see by Lemma 2.2 that the value ξ^* is achievable. ■

If there exist 0-1 optimal solutions, then

$$LB = -\|cP\|_1 + cA^T(AA^T)^{-1}b$$

is a lower bound on the optimal value and

$$UB = \|cP\|_1 + cA^T(AA^T)^{-1}b$$

is an upper bound on the optimal value, where $\|\cdot\|_1$ denotes the 1-norm being simply the sum of absolute values of components. These bounds follow from the fact that for every feasible solution x^* we have $x^* = \pi_L(x^*)$, which implies $cx^* = c\pi_L(x^*)$. Considering an optimal solution x^* with $\mathbf{0} \leq x^* \leq \mathbf{1}$, we obtain the bounds.

Lemma 2.3 *Consider x^k in L, $k = 1, \ldots, 2n$, such that*

$$\|x^k - \pi_{B^k}(x^k)\|^2 < \frac{1}{4n}.$$

If $cx^k \leq UB$ for all k, then in $O(n^2 \log n)$ time we can find an achievable value $\xi^\#$ such that there exists k with

$$\|\pi_{L(\xi^\#)}(x^k) - \pi_{B^k}\pi_{L(\xi^\#)}(x^k)\|^2 \geq \frac{1}{4n}.$$

Proof. Note that

$$\frac{1}{\sqrt{n}} = \|\pi_{L(\xi_k^*)}(x^k) - \pi_{B^k}\pi_{L(\xi_k^*)}(x^k)\| \leq \|\pi_{L(\xi_k^*)}(x^k) - \pi_{B^k}(x^k)\| \leq$$

$$\leq \|x^k - \pi_{B^k}(x^k)\| + \|x^k - \pi_{L(\xi_k^*)}(x^k)\| < \frac{1}{2\sqrt{n}} + \|x^k - \pi_{L(\xi_k^*)}(x^k)\|.$$

Then

$$\|x^k - \pi_{L(\xi_k^*)}(x^k)\| > \frac{1}{2\sqrt{n}}.$$

This implies

$$cx^k - \xi_k^* = \|cP\| \cdot \|x^k - \pi_{L(\xi_k^*)}(x^k)\| > \frac{\|cP\|_1}{2n}.$$

Then each segment $[\xi_k^*, cx^k]$ contains at least one integer multiple of $\frac{\|cP\|_1}{2n}$. It follows that a rounding of ξ_k^* up to the nearest integer multiple of $\frac{\|cP\|_1}{2n}$ yields a value $\xi_k^{\#}$ in the respective segment. Consider $\xi^{\#} = \max \xi_k^{\#}$. Since $\xi^{\#} \geq \xi^*$, it follows that $\xi^{\#}$ is achievable because ξ^* is achievable by Theorem 2.1. For ξ^* there exists k with

$$\|\pi_{L(\xi^*)}(x^k) - \pi_{B^k}\pi_{L(\xi^*)}(x^k)\|^2 = \frac{1}{n}.$$

It follows that, for this k,

$$\frac{1}{\sqrt{n}} = \|\pi_{L(\xi^*)}(x^k) - \pi_{B^k}\pi_{L(\xi^*)}(x^k)\| \leq \|\pi_{L(\xi^*)}(x^k) - \pi_{B^k}\pi_{L(\xi^{\#})}(x^k)\|$$

$$\leq \|\pi_{L(\xi^{\#})}(x^k) - \pi_{B^k}\pi_{L(\xi^{\#})}(x^k)\| + \|\pi_{L(\xi^{\#})}(x^k) - \pi_{L(\xi^*)}(x^k)\|.$$

Then we have the required inequality for $\xi^{\#}$ because the inequality $\xi^{\#} - \xi^* \leq \frac{\|cP\|_1}{2n}$ implies that the distance between $\pi_{L(\xi^{\#})}(x^k)$ and $\pi_{L(\xi^*)}(x^k)$ (it is equal to $\frac{\xi^{\#}-\xi^*}{\|cP\|}$) is bounded by $\frac{1}{2\sqrt{n}}$ due to $\|cP\|_1 \leq \sqrt{n}\|cP\|$.

We have $\xi^{\#} \geq LB$ because $\xi^{\#}$ is achievable. Therefore, $\xi^{\#} = \max_k \max\{\xi_k^{\#}, LB\}$. On the other hand, for all k,

$$\xi_k^{\#} \leq cx^k + \frac{\|cP\|_1}{2n} \leq UB + \frac{\|cP\|_1}{2n}.$$

Then each value $\max\{\xi_k^{\#}, LB\}$ can be found in time $O(n \log n)$ by a binary search over the integer multiples of $\frac{\|cP\|_1}{2n}$ lying between LB and $UB + \frac{\|cP\|_1}{2n}$, because we need $O(n)$ time to check if a value ξ' is feasible for the respective system. We obtain the running time $O(n^2 \log n)$ for computing $\xi^{\#}$. ■

3 Strongly polynomial algorithm

The algorithm described below finds a face B_{OPT} of the unit cube that gives the following necessary and sufficient optimality condition for 0-1 solutions:

> If there exist 0-1 optimal solutions, then a 0-1 feasible solution x is optimal if and only if $x \in B_{OPT}$.

Algorithm 3.1

```
**Input:** $\min\{cx | Ax = b, x \geq \mathbf{0}\}$;
**Output:** $B_{OPT}$.
$\xi := UB$;
$x^k := \pi_{L(\xi)}(\mathbf{0}), k = 1, \ldots, 2n$;
$u_k := n$;
$J_0 := \emptyset$ and $J_1 := \emptyset$;
**for** $t = 1, \ldots, n$
    Compute $P$, $cP$, $A^T(AA^T)^{-1}b$, and $cA^T(AA^T)^{-1}b$;
    **if** $cP = \mathbf{0}$ **then** break the for-loop;
    **while** $u_k \geq 0$ for all $k$
        **if** $\|x^k - \pi_{B^k}(x^k)\|^2 \geq \frac{1}{4n}$ for some $k$ **then**
            $u_k := u_k - \|x^k - \pi_{B^k}(x^k)\|^2 - \|\pi_{B^k}(x^k) - \pi_{L(\xi)}\pi_{B^k}(x^k)\|^2$;
            $x^k := \pi_{L(\xi)}\pi_{B^k}(x^k)$;
        **else**
            Compute $\xi^{\#}$ as in Lemma 2.3;
            $\xi := \xi^{\#}$;
            $x^k := \pi_{L(\xi)}(x^k)$ for all $k$;
        **end**
    **end**
    Find $r$ with $u_r < 0$;
    Let $B^l$ be the facet that is opposite to $B^r$. ($B^l$ contains all 0-1 optimal solutions.)
```

```
        Project the points $x^k$ onto $B^l$;
        Let $x_j$ be the variable, of the original problem, corresponding to $B^r$ and $B^l$;
        if $x_j$ must be 0 in 0-1 optimal solutions then $J_0 := J_0 \cup \{j\}$; else $J_1 := J_1 \cup \{j\}$;
        Remove the respective component of each $x^k$;
        Remove the respective column $a$ of $A$;
        if $x_j$ must be 1 in 0-1 optimal solutions then
            $b := b - a$;
            $\xi := \xi - c_j$, where $c_j$ is the respective component of the original cost vector;
        end
        Remove $x^l$ and $x^r$ from the list of points;
        $t := t + 1$;
        Let $B$ be the unit cube in $\mathbb{R}^{n-t}$;
        Let $B^k$ be the facets of $B$ corresponding to the remaining points $x^k$;
        $x^k := \pi_{L(\xi)}(x^k)$ for all $k$;
    end
    $B_{OPT} := \{x \in \mathbb{R}^n | \mathbf{0} \le x \le \mathbf{1}\} \cap \{x \in \mathbb{R}^n | x_j = 0, j \in J_0\} \cap \{x \in \mathbb{R}^n | x_j = 1, j \in J_1\}$.
    Return $B_{OPT}$.
```

Remark. Of course, it is extremely useful to have good initial values of u_k. We can improve them by simply using a certain number of alternating projections with respect to $L(\xi) \cap B$. In this case, we start with $\pi_{L(\xi)}(\mathbf{0})$ and an initial bound on the square of the distance from $\pi_{L(\xi)}(\mathbf{0})$ to $L(\xi) \cap B$ and reduce this bound by the square of the distance from the current point to the previous point after each projection. Then we initialize u_k with the obtained bound and x^k with the current point obtained by the alternating projections.

Consider the first iteration of the for-loop. The initial values of u_k are all equal to n. Each value u_k is an upper bound on the square of the distance between x^k and the intersection of B^k with the set M of 0-1 optimal solutions, provided that this intersection is not empty. This property of u_k is preserved in the course of the while-loop. This follows from the fact that ξ is achievable by Lemma 2.3 and the formula for u_k just reflects the reduction of the square of the distance from B^k to M after performing the two consecutive projections onto B^k and $L(\xi)$. The while-loop terminates when there is $u_r < 0$. In this case $B^r \cap M$ is empty. This means that M must be contained in the opposite facet B^l. Then the algorithm reduces the number of variables by assigning the respective value (0 or 1) to the variable x_j that corresponds to B^r and B^l. If it is proved that x_j is equal to 1 in all 0-1 optimal solutions, then the respective column of A should be subtracted from b. If the original problem has 0-1 optimal solutions, we obtain a problem that has 0-1 optimal solutions. Since the points x^r and x^l are not needed anymore, the algorithm simply eliminates them from the list of points x^k. Since it has been proved that B^l contains M, the projection of each point x^k onto B^l and the elimination of the respective component of each x^k preserves the property that each value u_k is an upper bound on the square of the distance from x^k to $B^k \cap M$, provided that $B^k \cap M \neq \emptyset$. Here, M denotes the set of 0-1 optimal solutions of the current problem and B^k denote the respective facets of the unit cube B in $\mathbb{R}^{n-1}$. Note that the value ξ is modified if the eliminated variable must be equal to 1 in all 0-1 optimal solutions. So ξ is achievable for the new problem.

Theorem 3.1 *The algorithm runs in time $O(n^4 \max\{m, \log n\})$. If the original problem $\min\{cx|Ax = b, x \geq \mathbf{0}\}$ has 0-1 optimal solutions, then $L \cap B_{OPT} \cap \{0,1\}^n$ is the set of 0-1 optimal solutions and $L \cap B_{OPT}$ is a subset of the optimal set.*

Proof. Let us first prove that the for-loop works correctly. All the values ξ in the course of the algorithm are achievable with respect to the *current* problem by Lemma 2.3. Let $M^{(t)}$ denote the set of 0-1 optimal solutions of the current problem at iteration t of the algorithm. (The set $M^{(t)}$ is obtained from $M^{(t-1)}$ by removing the respective component of each vector in $M^{(t-1)}$.) Since $M^{(t)}$ is contained in $L(\xi)$, it follows from Lemma 2.1 that each projection at iteration t does not increase the distance from x^k to each point of $B^k \cap M^{(t)}$. The elimination of a variable at each iteration of the for-loop does not affect the distances from the points x^k to the points of $M^{(t)}$ because the elimination is performed after the projection of x^k onto the facet B^l that contains $M^{(t)}$. The formula for computing u_k follows from Lemma 2.1. That is, u_k is an upper bound on the square of the distance from x^k to $B^k \cap M^{(t)}$ in the course of the algorithm, provided that this intersection is not empty. If some value u_r becomes negative, then the facet B^r does not intersect with $M^{(t)}$. This means that $M^{(t)}$ is a subset of the opposite facet B^l.

We now estimate the running time. Every two iterations of the while-loop reduce the square of the distance from x^k to $B^k \cap M^{(t)}$ by at least $\frac{1}{4(n-t)}$. (To show this, we use Lemma 2.3 in the case when we calculate $\xi^{\#}$. The lemma guarantees that after computing $\xi^{\#}$ the while-loop reduces u_k for some k.) This implies the respective reduction of u_k by at least $\frac{1}{4(n-t)}$. Therefore, since u_k are initialized with n and we have $2(n-t)$ points x^k, the overall number of iterations of the while-loop in the course of the algorithm is bounded by $O(n^3)$. Then the overall number of arithmetic operations performed at the iterations that do not need the computation of $\xi^{\#}$ is bounded by $O(n^4 m)$ because a projection of a point on $L(\xi)$ can be found in time $O(nm)$. The values of $\xi^{\#}$ considered by the algorithm at iteration t form a decreasing sequence of integer multiples of $\frac{\|cP\|_1}{2(n-t)}$ between the bounds LB and UB on the optimal value of the current problem. It follows that the number of computations of $\xi^{\#}$ at iteration t is bounded by $O(n)$. So the overall number of arithmetic operations involved in the computation of $\xi^{\#}$ at iteration t is bounded by $O(n^3 \log n)$. (Here we should note that all the projections of the points x^k on $L(\xi^{\#})$ can be found in time $O(n^2)$ because the points x^k belong to L and we only need to project them onto $H(\xi^{\#})$.) The algorithm eliminates a variable at most n times, each time performing $O(n^3)$ arithmetic operations to compute a new projection matrix P, the vector cP, and the other vectors and values needed. It follows that the algorithm runs in time $O(n^4 \max\{m, \log n\})$.

Consider the current linear optimization problem at the last iteration of the algorithm. All feasible solutions of this problem have the same objective values and therefore are optimal for the current problem. Since the for-loop works correctly, it follows that a 0-1 feasible solution x is optimal if and only if it belongs to B_{OPT}.

■

Now it only remains to introduce a rounding that will guarantee that the sizes of the numbers are polynomially bounded in the case of a rational input. We consider only the first iteration of the for-loop, where the number of variables is equal to n. At the other iterations,

n should be replaced by the current number of variables. In the while-loop we replace the projection $\pi_{L(\xi)}\pi_{B^k}(x^k)$ by the "rounded" projection $\pi_{L(\xi)}(\mu\lfloor\pi_{B^k}(x^k)/\mu\rceil)$, where $\mu = \frac{1}{64n^2}$. Here, $\lfloor\cdot\rceil$ denotes the rounding of each component. Since $\pi_{B^k}(x^k)$ belongs to the unit cube, the rounding takes $O(n\log n)$ time. The modification concerns only the computation of u_k and x^k in the while-loop:

$$\begin{aligned}
&u_k := u_k - \|x^k - \pi_{B^k}(x^k)\|^2 - \|\pi_{B^k}(x^k) - \pi_{L(\xi)}\pi_{B^k}(x^k)\|^2 + \frac{1}{8n};\\
&x^k := \pi_{L(\xi)}(\mu\lfloor\pi_{B^k}(x^k)/\mu\rceil).
\end{aligned}$$

We have

$$\|\pi_{B^k}(x^k) - \mu\lfloor\pi_{B^k}(x^k)/\mu\rceil\|^2 \le \mu^2 n.$$

Then

$$\|\pi_{L(\xi)}\pi_{B^k}(x^k) - \pi_{L(\xi)}(\mu\lfloor\pi_{B^k}(x^k)/\mu\rceil)\|^2 \le \left\|P\left(\pi_{B^k}(x^k) - \mu\lfloor\pi_{B^k}(x^k)/\mu\rceil\right)\right\|^2 \le \mu^2 n.$$

Then, taking into account that the previous u_k is not greater than n, we conclude that the new u_k provides an upper bound on the square of the distance between x^k and the set $B^k \cap M$ if this set is not empty. Here, M denotes the set of 0-1 optimal solutions. The value μ is sufficiently small to guarantee that u_k decreases by at least $\frac{1}{8n}$. We obtain a modified algorithm that preserves all important properties of the original one, in particular that u_k is an upper bound on the square of the distance between x^k and $B^k \cap M^{(t)}$, where $M^{(t)}$ is the set of 0-1 optimal solution of the current problem at iteration t. The theoretical estimate of the running time remains the same. The sizes of the numbers in the course of the modified algorithm are bounded by a polynomial in the size of the instance. Thus we have proved the following theorem:

Theorem 3.2 *The modified algorithm is a strongly polynomial algorithm that finds B_{OPT} in time $O(n^4\max\{m, \log n\})$.*

If there are 0-1 optimal solutions, then $L \cap B_{OPT}$ is a subset of optimal solutions. Using the algorithm presented in [3], in strongly polynomial time we can either find a solution in $L \cap B_{OPT}$ (which is not necessarily 0-1) or prove that this set contains no 0-1 solutions. This implies the following theorem:

Theorem 3.3 *There exists a strongly polynomial algorithm that runs in time* $O(n^4 \max\{m, \log n\})$ *for a linear optimization problem* $\min\{cx | Ax = b, x \geq \mathbf{0}\}$ *having 0-1 optimal solutions.*

Proof. The necessary and sufficient optimality conditions must be satisfied for each 0-1 optimal solution. Consider the system $Ax = b, x \in B_{OPT}, x \geq \mathbf{0}$. Note that B_{OPT} can be written as a system of linear equations. If there is a 0-1 solution, then this system can be solved in time $O(n^4)$ by means of the polynomial algorithm given in [3]. (The obtained solution is not necessarily 0-1.) If no solution is found by that algorithm, then no 0-1 optimal solutions exist. ■

Thus, in strongly polynomial time we either find a solution whose optimality is guaranteed if there exist 0-1 optimal solutions or prove that no 0-1 optimal solutions exist.

If it is known that there is an optimal solution $x^* = Fz$ where F is a nonnegative $n \times q$ matrix and $z \in S^q$ where S is some finite subset of nonnegative numbers, then we can find an optimal solution of $\min\{cx | Ax = b, x \geq \mathbf{0}\}$ in time $O((|S|q)^4 \max\{m, \log |S|q\})$. To see this, we formulate the problem as $\min\{cFz | AFz = b, z \in S^q\}$ and then replace each variable z_j by the sum $\sum_{s \in S} s \cdot z_{sj}$ of $|S|$ new variables z_{sj} multiplied by the respective coefficients. In place of $z \in S^q$ we write $z_{sj} \geq 0$ for all new variables. The obtained problem has a 0-1 optimal solution. An optimal solution of this problem can be translated back into an optimal solution of the original problem.

For instance, the linear optimization problem over the fractional perfect matching polytope can be represented in this form. In this case A is the incidence matrix of a graph, F is the identity matrix and $S = \{0, \frac{1}{2}, 1\}$.

Another example is a situation when we only know that there exist optimal solutions whose components belong to a set $S = \{s_1, \ldots, s_n\}$ of nonnegative numbers. Then we consider $F = (s_1 I | \ldots | s_n I)$. It follows that the respective linear optimization problem can be solved in strongly polynomial time in this case.

The following theorem states that we can use a strongly polynomial algorithm for finding a feasible solution of an optimization problem to obtain a strongly polynomial algorithm for this optimization problem, provided that it can be formulated as a linear optimization problem having a 0-1 optimal solution:

Theorem 3.4 *Let $\mathcal{A}$ be a set of matrices such that $A' \subset A \Rightarrow A' \in \mathcal{A}$ for all A in $\mathcal{A}$. (Here, we treat matrices as sets of columns.) Consider the decision problem consisting of instances $Ax = b, x_j \in \{0,1\}$ for all j, where $A \in \mathcal{A}$. Let every feasible instance* $\min\{cx | Ax = b, x \geq \mathbf{0}\}$ *of the respective linear optimization problem have a 0-1 optimal solution. There exists a strongly polynomial algorithm for the decision problem if and only if there exists a strongly polynomial algorithm for finding 0-1 optimal solutions of the optimization problem.*

Proof. Let us prove the only-if part. We consider the first iteration of the for-loop, i.e., the one with $t = 0$. Let $B^k = \{x \in B | x_k = 1\}$, $k = 1, \dots, n$, and $B^k = \{x \in B | x_{k-n} = 0\}$, $k = n+1, \dots, 2n$. Note that if ξ is not achievable then there exists a *nonnegative* row vector h with $hx^0 < \min\{hx | x \in [0,1]^n\}$ for all x^0 in $L(\xi)$ because $L(\xi)$ and $\mathbb{R}^n_+$ can then be separated by a hyperplane, which implies that $L(\xi)$ can be separated from the unit cube by the same hyperplane. This implies a stronger variant of Lemma 2.2. Since h is nonnegative, by the proof of this lemma we obtain that there is B^k with $k = 1, \dots, n$, such that the distance between B^k and every point in $L(\xi)$ is greater than $\frac{1}{\sqrt{n}}$. Now we check only x^k with $k = 1, \dots, n$, in the while-loop. The remaining iterations of the for-loop of this variant of the algorithm work in the same way. It follows that $J_1 = \emptyset$. (That is, b remains the same in the course of this variant of the algorithm.) Thus, a 0-1 vector x is an optimal solution if and only if $Ax = b, x_j = 0, j \in J_0, x \in \{0,1\}^n$. Removing the respective columns, we obtain an equivalent system $A'x' = b, x' \in \{0,1\}^{n-|J_0|}$, with $A' \in \mathcal{A}$. The algorithm for the decision problem can be used to find a solution for this system in strongly polynomial time. Filling the variables x_j, $j \notin J_0$, with the respective components of the obtained solution x' and the variables x_j, $j \in J_0$, with zeros, we obtain a 0-1 optimal solution. ■

For example, the well-known linear programming formulation of the problem of finding minimum-weight perfect matchings in bipartite graphs satisfies the conditions of the above theorem. If applied to this problem, our algorithm actually constructs a subgraph having the property that all perfect matchings in this subgraph are optimal. To find a perfect matching in this subgraph, we can apply any algorithm for finding perfect matchings in bipartite graphs.

References

[1] Agmon, Sh. 1954. The relaxation method for linear inequalities. *Canad. J. Math.* **6**, 382-392.

[2] Chubanov, S. 2012. A strongly polynomial algorithm for linear systems having a binary solution. *Mathematical Programming* **134**: 533-570.

[3] Chubanov, S. 2013. A polynomial projection algorithm for linear programming. *Optimization Online.*

[4] Dattorro, J. 2005. Convex Optimization & Euclidean Distance Geometry, Meboo publishing, USA.

[5] Edmonds, J. 1967. Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards* **71 B**, 241-245.

[6] Motzkin, Th., and Schoenberg, I. J. 1954. The relaxation method for linear inequalities. *Canad. J. Math.* **6**, 393-404.

[7] Renegar, J. 1988. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming* **40**, 59-93.

[8] Tardos, E. 1986. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research* **34**, 250-256.