Mathematical Programming 61 (1993) 233–249
North-Holland

A recursive algorithm for finding the minimum norm point in a polytope and a pair of closest points in two polytopes

Kazuyuki Sekitani and Yoshitsugu Yamamoto
Institute of Socio-Economic Planning, University of Tsukuba, Ibaraki, Japan

Received 27 July 1991
Revised manuscript received 22 June 1992

For a given pair of finite point sets P and Q in some Euclidean space we consider two problems: Problem 1 of finding the minimum Euclidean norm point in the convex hull of P and Problem 2 of finding a minimum Euclidean distance pair of points in the convex hulls of P and Q. We propose a finite recursive algorithm for these problems. The algorithm is not based on the simplicial decomposition of convex sets and does not require to solve systems of linear equations.

Key words: Minimum norm point, minimum distance pair of points, recursive algorithm, convex quadratic program.

1. Introduction

Let P and Q be a given pair of sets of finite points of $\mathbb{R}^n$. Let us denote the convex hull of P by $C(P)$. We consider the following two problems:

Problem 1. Find the point of $C(P)$ which has the minimum Euclidean norm.

Problem 2. Find a pair of points of $C(P)$ and $C(Q)$ which has the minimum Euclidean distance.

Introducing the convex combination coefficients λ_p Problem 1 reduces to the following strictly convex quadratic program:

$$\begin{aligned} \min \quad & \|x\|^2, \\ \text{s.t.} \quad & x = \sum_{p\in P} \lambda_p p, \\ & \sum_{p\in P} \lambda_p = 1, \\ & \lambda_p \geqslant 0 \quad \text{for all } p\in P. \end{aligned}$$

Correspondence to: Institute of Socio-Economic Planning, University of Tsukuba, Tsukuba, Ibaraki 305, Japan. Phone: 0298-53-5001/-5545, fax: 0298-53-5070/-55-3849, e-mail: yamamoto@shako.sk.tsukuba.ac.jp.

Then Problem 1 has a unique solution, which we denote by $\mathrm{Nr}(C(P))$. On the other hand, the solution of Problem 2 may not be unique. Take for example $P=\{(1,1),(1,-1)\}$ and $Q=\{(-1,1),(-1,-1)\}$, then $(x_1^P, x_2^P)\in C(P)$ and $(x_1^Q, x_2^Q)\in C(Q)$ are a solution if $x_2^P=x_2^Q$. Then we denote the set of pairs of points of $C(P)$ and $C(Q)$ which have the minimum Euclidean distance by $\mathrm{Nr}(C(P), C(Q))$.

For a point $\hat{x}$ and a real number $\hat{\alpha}$ we denote the hyperplane $\{x \mid \hat{x}^{\mathrm{t}}x=\hat{\alpha}\}$ by $H(\hat{x}, \hat{\alpha})$ and the half space $\{x \mid \hat{x}^{\mathrm{t}}x\geqslant\hat{\alpha}\}$by $H^+(\hat{x}, \hat{\alpha})$. The following basic optimality condition is proved by Wolfe [9, Theorem 2.1].

Theorem 1.1. *$\hat{x}\in C(P)$ is $\mathrm{Nr}(C(P))$ if and only if*

$$P\subseteq H^+(\hat{x}, \|\hat{x}\|^2),$$

or equivalently

$$\|\hat{x}\|^2\leqslant\hat{x}^{\mathrm{t}}p \quad \textit{for every } p\in P. \qquad \square$$

Concerning the relation between Problem 1 and 2, Canon and Cullum [1] showed that Problem 2 is reduced to Problem 1 with the set P replaced by $P-Q=\{p-q \mid p\in P, q\in Q\}$. Hence we see the following optimality condition for Problem 2.

Theorem 1.2. *$\hat{x}^P\in C(P)$ and $\hat{x}^Q\in C(Q)$ are a minimum Euclidean distance pair if and only if*

$$P-Q\subseteq H^+(\hat{x}^P-\hat{x}^Q, \|\hat{x}^P-\hat{x}^Q\|^2). \qquad \square$$

These two problems have been considered by Wolfe [9]. He provided several basic results and also proposed an algorithm based on the simplicial decomposition of the convex hull $C(P)$. His algorithm forms a simplex being the convex hull of several affinely independent points of P. Solving a system of linear equations, it finds the minimum norm point on the affine hull of the simplex. When the point lies in the relative interior of the simplex, another point of P is chosen to form a simplex of one dimension higher than the current simplex. Otherwise, it drops a point to form a lower dimensional simplex. Fujishige and Zhan [5, 6] proposed a dual algorithm for these problems which is based on the dual formulation as well as the simplicial decomposition (see for example Freund [4] for the duality of the problems). Fujishige–Zhan's algorithm rotates a supporting hyperplane of $C(P)$ which separates $C(P)$ from the origin so that the distance between the hyperplane and the origin increases monotonically. Their algorithm also generates simplices of various dimension and solves systems of linear equations to find the minimum norm point on the affine hull of each simplex generated. At the start the algorithm requires an initial hyperplane separating $C(P)$ from the origin. Unless it is known in advance, they add another dimension and consider the problem for $P\times\{1\}\subseteq\mathbb{R}^{n+1}$.

In Section 2 we propose a recursive algorithm for Problem 1 and show that it provides $\mathrm{Nr}(C(P))$ after a finite number of iterations. The algorithm has the advantage that it can start with an arbitrary point of $C(P)$ and requires neither the initial separating hyperplane nor the additional dimension. Due to its recursive structure the algorithm does not require to solve linear equations. In Section 3 we show some norm monotonicity property of iterates and how the accuracy of solutions to be obtained is affected by the computational error. In Section 4 an algorithm for Problem 2 is proposed based on the equivalence of $\mathrm{Nr}(C(P), C(Q))$ to $\mathrm{Nr}(C(P-Q))$. Some remarks are given in Section 5.

2. Algorithm $\mathcal{N}_1$ for finding $\mathrm{Nr}(C(P))$

In this section we consider Problem 1 of finding $\mathrm{Nr}(C(P))$ for a given finite point set P. The algorithm first chooses a point x_0 from the convex hull $C(P)$. A point of P with the minimum norm of all points of P is recommended. In the kth iteration with x_{k-1} as the current point it generates a proper subset P_k of P being the set of points minimizing the linear function $x_{k-1}^{\mathrm{t}}p$ over all $p \in P$. Namely, $P_k = \{p \mid p \in P \cap H(x_{k-1}, \alpha_k)\}$, where $\alpha_k = \min\{x_{k-1}^{\mathrm{t}}p \mid p \in P\}$. Then it calls itself with P_k as the input and sets y_k be the output of the recursive call. Note that $C(P) \subseteq H^+(x_{k-1}, \alpha_k)$ and $y_k \in C(P) \cap H(x_{k-1}, \alpha_k)$. The hyperplane $H(x_{k-1}, \alpha_k)$ is rotated on y_k by changing the normal vector from x_{k-1} toward y_k as long as it supports $C(P)$. This is done by finding the maximum value of λ such that

$$\{(1-\lambda)x_{k-1} + \lambda y_k\}^{\mathrm{t}} y_k \leqslant \{(1-\lambda)x_{k-1} + \lambda y_k\}^{\mathrm{t}} p \tag{2.1}$$

holds for every $p \in P$. Since $x_{k-1}^{\mathrm{t}}p = x_{k-1}^{\mathrm{t}}y_k$ and $\|y_k\|^2 \leqslant y_k^{\mathrm{t}}p$ for $p \in P_k$, (2.1) holds for ever $p \in P_k$ and for $\lambda \geqslant 0$. Therefore we have only to consider the points of $P \backslash P_k$ when finding the maximum λ.

Algorithm $\mathcal{N}_1(P)$.

Step 0. Choose a point x_0 from $C(P)$ and $k := 1$.

Step 1. $\alpha_k := \min\{x_{k-1}^{\mathrm{t}}p \mid p \in P\}$.
If $\|x_{k-1}\|^2 \leqslant \alpha_k$, then $\hat{x} := x_{k-1}$ and stop.

Step 2. $P_k := \{p \mid p \in P \text{ and } x_{k-1}^{\mathrm{t}}p = \alpha_k\}$.
Call $\mathcal{N}_1(P_k)$ and $y_k := \mathrm{Nr}(C(P_k))$.

Step 3. $\beta_k := \min\{y_k^{\mathrm{t}}p \mid p \in P \backslash P_k\}$.
If $\|y_k\|^2 \leqslant \beta_k$, then $\hat{x} := y_k$ and stop.

Step 4. $\lambda_k := \max\{\lambda \mid \{(1-\lambda)x_{k-1} + \lambda y_k\}^{\mathrm{t}} y_k \leqslant \{(1-\lambda)x_{k-1} + \lambda y_k\}^{\mathrm{t}} p$ for every $p \in P \backslash P_k\}$.
$x_k := (1-\lambda_k)x_{k-1} + \lambda_k y_k$, $k := k+1$ and go to Step 1.

First we will show that the supporting hyperplane $H(x_{k-1}, \alpha_k)$ can be rotated on y_k.

Lemma 2.1. $0<\lambda_k<1$ *for* $k=1,2,\ldots$.

Proof. Clearly λ_k is determined by

$$\lambda_k=\min\left\{\frac{x_{k-1}^{\mathrm{t}}(p-y_k)}{(y_k-x_{k-1})^{\mathrm{t}}(y_k-p)}\,\middle|\, p\in P\backslash P_k \text{ and } (y_k-x_{k-1})^{\mathrm{t}}(y_k-p)>0\right\}. \tag{2.2}$$

Since $y_k\in C(P_k)$, we see that $x_{k-1}^{\mathrm{t}}(p-y_k)=x_{k-1}^{\mathrm{t}}p-x_{k-1}^{\mathrm{t}}y_k>0$ for every $p\in P\backslash P_k$. Therefore $\lambda_k>0$. Since the algorithm did not stop at Step 3, there is a point $p'\in P\backslash P_k$ such that $\|y_k\|^2>y_k^{\mathrm{t}}p'$. This means that $\lambda_k<1$. □

By the above lemma we see that each point x_{k-1} as well as y_k is contained in $C(P)$. Then by applying Theorem 1.1 we will see that the point $\hat{x}$ obtained either in Step 1 or in Step 3 is $\mathrm{Nr}(C(P))$.

Lemma 2.2. $\hat{x}=\mathrm{Nr}(C(P))$.

Proof. Note that the initial point x_0 is chosen from $C(P)$, $y_{k-1}=\mathrm{Nr}(C(P_{k-1}))\in C(P_{k-1})\subseteq C(P)$ and $0<\lambda_{k-1}<1$. Then we see $x_{k-1}\in C(P)$ by induction. If termination occurs in Step 1, then $\hat{x}$ obtained there is $\mathrm{Nr}(C(P))$ by Theorem 1.1. To prove that $\hat{x}$ obtained in Step 3 is $\mathrm{Nr}(C(P))$ we have only to point out that $P_k\subseteq H^+(y_k,\|y_k\|^2)$ because $y_k=\mathrm{Nr}(C(P_k))$ and $P\backslash P_k\subseteq H^+(y_k,\|y_k\|^2)$ because $\|y_k\|^2\leqslant\beta_k$. □

We will see that the set P_k generated in Step 2 is a proper subset of P.

Lemma 2.3. *If* $\|x_{k-1}\|^2>\alpha_k$, *then* P_k *is a proper subset of* P *and* $C(P_k)$ *is a proper face of* $C(P)$.

Proof. Since $\|x_{k-1}\|^2>\alpha_k=\min\{x_{k-1}^{\mathrm{t}}p\,|\,p\in P\}$, there exists a point of P which does not lie on the affine hull of P_k. Then $\dim C(P_k)<\dim C(P)$. This proves the lemma. □

Lemma 2.4. *There exists a point* $\bar{p}$ *of* $P_{k+1}\backslash P_k$ *such that* $\bar{p}\notin H^+(y_k,\|y_k\|^2)$.

Proof. Let us choose a point $\bar{p}$ from the point set attaining the minimum of (2.2) in the proof of Lemma 2.1. Equivalently the point $\bar{p}$ of $P\backslash P_k$ satisfies $(y_k-x_{k-1})^{\mathrm{t}}(y_k-\bar{p})>0$ and

$$\lambda_k=\frac{x_{k-1}^{\mathrm{t}}(\bar{p}-y_k)}{(y_k-x_{k-1})^{\mathrm{t}}(y_k-\bar{p})}. \tag{2.3}$$

We will see that $\bar{p}\in P_{k+1}$. In fact for a point $p\in P\backslash P_k$ we have

$$\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}\bar{p}=\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}y_k \leqslant\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}p$$

by the definition of λ_k. For a point $p\in P_k$ we also see from the definition of y_k,

$$\begin{aligned}\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}\bar{p}&=\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}y_k=(1-\lambda_k)x_{k-1}^{\mathrm{t}}y_k+\lambda_k\|y_k\|^2\\&=(1-\lambda_k)x_{k-1}^{\mathrm{t}}p+\lambda_k\|y_k\|^2\leqslant(1-\lambda_k)x_{k-1}^{\mathrm{t}}p+\lambda_k y_k^{\mathrm{t}}p\\&=\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}p.\end{aligned}$$

Since $\lambda_k<1$ as shown in Lemma 2.1, we see from (2.3) that $x_{k-1}^{\mathrm{t}}(\bar{p}-y_k)<(y_k-x_{k-1})^{\mathrm{t}}(y_k-\bar{p})$, which is equivalent to $\bar{p}\notin H^+(y_k, \|y_k\|^2)$. □

Lemma 2.5. *$y_k\in C(P_{k+1})$ for $k=1, 2, \ldots$.*

Proof. Since $y_k=\mathrm{Nr}(C(P_k))\in C(P_k)\subseteq C(P)$ and we have seen

$$\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}y_k=\min\{\{(1-\lambda_k)x_{k-1}+\lambda_k y_k\}^{\mathrm{t}}p\,|\,p\in P\} \tag{2.4}$$

in the proof of Lemma 2.4, the lemma follows from the definition of P_{k+1}. □

The following lemma is the key to finite convergence of the algorithm.

Lemma 2.6. *$\|y_{k+1}\|<\|y_k\|$ for $k=1, 2, \ldots$.*

Proof. For the point $\bar{p}$ of Lemma 2.4, we have $y_k^{\mathrm{t}}(y_k-\bar{p})/\|y_k-\bar{p}\|^2>0$. Choose a λ such that

$$0<\lambda<\min\left\{\frac{2y_k^{\mathrm{t}}(y_k-\bar{p})}{\|y_k-\bar{p}\|^2}, 1\right\},$$

and consider the point $z=(1-\lambda)y_k+\lambda\bar{p}$. Then $z\in C(P_{k+1})$ by Lemma 2.4 and Lemma 2.5. Furthermore $\|z\|^2=\|y_k\|^2+\lambda\{2y_k^{\mathrm{t}}(\bar{p}-y_k)+\lambda\|\bar{p}-y_k\|^2\}<\|y_k\|^2$. Therefore we obtain $\|y_{k+1}\|=\|\mathrm{Nr}(C(P_{k+1}))\|\leqslant\|z\|<\|y_k\|$. □

Lemma 2.7. *When P consists of a single point, the algorithm $\mathcal{N}_1$ terminates within a finite number of iterations.*

Proof. Let p be the point of P. Then x_0 must be p and $\alpha_1=\|p\|^2$. Therefore the algorithm $\mathcal{N}_1$ stops in Step 1 with $\hat{x}=p$. □

Theorem 2.8. *When P consists of finitely many points, the algorithm $\mathcal{N}_1$ provides the minimum norm point $\mathrm{Nr}(C(P))$ within a finite number of iterations.*

Proof. We prove the theorem by induction over the number of points of P. By Lemma 2.7 we assume that $\mathcal{N}_1(P')$ is finite whenever P' has fewer points than P has. Then by Lemma 2.3 we see that each step of the algorithm $\mathcal{N}_1(P)$ is finite. Since $\|\mathrm{Nr}(C(P_{k+1}))\| = \|y_{k+1}\| < \|y_k\| = \|\mathrm{Nr}(C(P_k))\|$ from Lemma 2.6, no P_k is generated more than once. Thus we see that $\mathcal{N}_1(P)$ terminates within a finite number of iterations. □

When we are given a set P of m points on a plane, the convex hull $C(P)$ has at most m vertices and at most m facets. This observation yields the following theorem.

Theorem 2.9. *The algorithm* $\mathcal{N}_1$ *finds* $\mathrm{Nr}(C(P))$ *with* $\mathrm{O}(m^2)$ *time complexity when* $P \subseteq \mathbb{R}^2$.

Proof. Clearly the minimum norm point problem on a vertex, a single point, is solved in a constant time. We should remark that the minimum norm point problem on a 1-dimensional polytope is solved within linear time of the number of points p in the polytope by applying the algorithm $\mathcal{N}_1$ since $\mathcal{N}_1$ does not repeat Step 2 more than once for this problem. Hence each step of $\mathcal{N}_1(P)$ is of at most $\mathrm{O}(m)$ time complexity. Since the algorithm $\mathcal{N}_1(P)$ does not generate the same facet or the same vertex of $C(P)$, it provides $\mathrm{Nr}(C(P))$ with $\mathrm{O}(m^2)$ time complexity. □

3. Example

We illustrate the behavior of the algorithm for the example shown in Figure 3.1. The set P consists of four points $p^1, \ldots, p^4$ in $\mathbb{R}^2$. Suppose we have chosen p^3 as the initial point x_0 in Step 0. Since P is not contained in $H^+(x_0, \|x_0\|^2)$, the algorithm generates the subset $P_1 = \{p^1\}$ which attains $\min\{x_0^t p \mid p \in P\} = \alpha_1$. Because P_1 is a singleton, p^1 itself is $\mathrm{Nr}(C(P_1))$ and we let $y_1 = p^1$. Since P is not yet contained in $H^+(y_1, \|y_1\|^2)$, we go to Step 4. The algorithm rotates the hyperplane $H(x_0, \alpha_1)$ on y_1 by moving the normal vector of the hyperplane from x_0 toward y_1 while keeping it supporting $C(P)$. We let x_1 be the normal vector of the hyperplane which would no longer support $C(P)$ if rotated more and return to Step 1. Since P is not contained in $H^+(x_1, \|x_1\|^2)$, we go to Step 2. The algorithm generates the subset $P_2 = \{p^1, p^2\}$, which lies on $H(x_1, \alpha_2)$, where $\alpha_2 = \min\{x_1^t p \mid p \in P\}$. We solve the subproblem $\min\{\|x\|^2 \mid x \in C(P_2)\}$ by the recursive call and obtain $y_2 = \mathrm{Nr}(C(P_2))$. Since P is not yet contained in $H^+(y_2, \|y_2\|^2)$, we go to Step 4. In the same way as the first iteration the algorithm moves the normal vector from x_1 towards y_2 and rotates the hyperplane $H(x_1, \alpha_2)$ on y_2. Then we find x_2 and return to Step 1. The algorithm generates the subset $P_3 = \{p^2, p^3\}$ lying on $H(x_2, \alpha_3)$, where $\alpha_3 = \min\{x_2^t p \mid p \in P\}$. By the recursive call with the set P_3 as the input data we obtain $y_3 = \mathrm{Nr}(C(P_3))$. Finally we see that P is contained in $H^+(y_3, \|y_3\|^2)$, and we have $y_3 = \mathrm{Nr}(C(P))$.

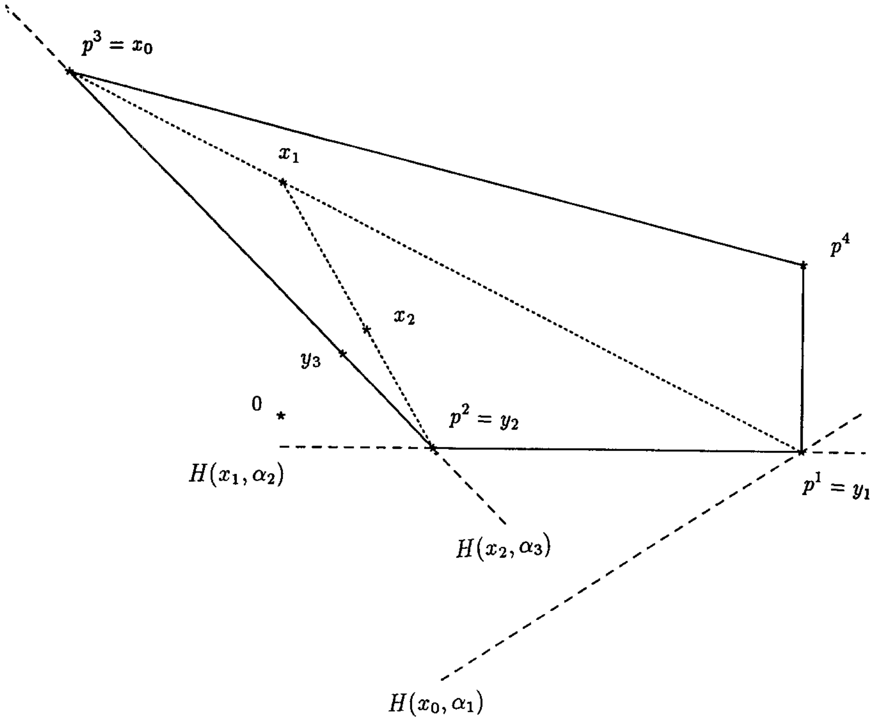

Fig. 3.1.

4. Norm monotonicity and error analysis

The monotonicity property of norm $\|y_k\|$ was crucial in proving the finite termination of the algorithm $\mathcal{N}_1$. We will see the same monotonicity for another iterates x_k. We denote the affine combination of x_{k-1} and y_k with coefficient λ by $z(\lambda)$, i.e., $z(\lambda) = (1-\lambda)x_{k-1} + \lambda y_k$.

Theorem 4.1. $\|x_k\| < \|x_{k-1}\|$ *for* $k = 1, 2, \ldots$.

Proof. Since $y_k \in C(P_k)$ and $\|x_{k-1}\|^2 > \alpha_k = x_{k-1}^t p$ for every $p \in P_k$, $x_{k-1}^t(x_{k-1} - y_k) > 0$ and $x_{k-1} \neq y_k$. We consider the point $z(\lambda)$ for $\lambda = x_{k-1}^t(x_{k-1} - y_k)/\|x_{k-1} - y_k\|^2$. Note that $\lambda > 0$ and

$$\begin{aligned}\|z(\lambda)\|^2 &= \|y_k - x_{k-1}\|^2 \left\{\lambda - \frac{x_{k-1}^t(x_{k-1} - y_k)}{\|x_{k-1} - y_k\|^2}\right\}^2 + \|x_{k-1}\|^2 - \frac{\{x_{k-1}^t(x_{k-1} - y_k)\}^2}{\|x_{k-1} - y_k\|^2} \\ &= \|x_{k-1}\|^2 - \lambda^2\|x_{k-1} - y_k\|^2 < \|x_{k-1}\|^2.\end{aligned}$$

The equality (2.4) implies from $x_{k-1} \in C(P)$ that

$$\{(1-\lambda_k)x_{k-1} + \lambda_k y_k\}^t y_k \leq \{(1-\lambda_k)x_{k-1} + \lambda_k y_k\}^t x_{k-1}.$$

Therefore we see from Lemma 2.1 that $0<\lambda_k\leqslant x_{k-1}^{\mathrm{t}}(x_{k-1}-y_k)/\|x_{k-1}-y_k\|^2=\lambda$ and $x_k=(1-\mu)x_{k-1}+\mu z(\lambda)$ for some $0<\mu\leqslant 1$. Hence $\|x_k\|=\|(1-\mu)x_{k-1}+\mu z(\lambda)\|<\|x_{k-1}\|$. □

Computational errors in evaluating α_k, β_k and norms $\|x_{k-1}\|$ and $\|y_k\|$ are unavoidable and some tolerance should be introduced to the termination criteria, e.g., $\|x_{k-1}\|^2\leqslant\alpha_k$ be relaxed to $\|x_{k-1}\|^2\leqslant\alpha_k+\delta_1$ for some user-specified tolerance $\delta_1>0$. We show in Corollary 4.4 and Theorem 4.5 how the solution to be obtained is affected by introducing the tolerance. The following theorem is proved by Wolfe [9, Theorem 2.2].

Theorem 4.2. *If* $0\neq x\in C(P)$ *and* $\alpha=\min\{x^{\mathrm{t}}p\,|\,p\in P\}$, *then*

$$\frac{\alpha}{\|x\|}\leqslant\|\mathrm{Nr}(C(P))\|\leqslant\|x\|. \qquad \square$$

Lemma 4.3. *If* $x\in C(P)$ *and* $\alpha=\min\{x^{\mathrm{t}}p\,|\,p\in P\}$, *then*

$$\|x-\mathrm{Nr}(C(P))\|^2\leqslant\|x\|^2-\alpha.$$

Proof. We see from Theorem 1.1 and the definition of α that

$$\begin{aligned}\|x-\mathrm{Nr}(C(P))\|^2&=\|x\|^2-2x^{\mathrm{t}}\,\mathrm{Nr}(C(P))+\|\mathrm{Nr}(C(P))\|^2\\&\leqslant\|x\|^2-2x^{\mathrm{t}}\,\mathrm{Nr}(C(P))+\mathrm{Nr}(C(P))^{\mathrm{t}}x\\&=\|x\|^2-x^{\mathrm{t}}\,\mathrm{Nr}(C(P))\leqslant\|x\|^2-\alpha. \qquad \square\end{aligned}$$

If we cast Problem 1 into the convex quadratic program as in Section 1 and apply Theorem 2.2 or Corollary 2.4 of Mangasarian and De Leone [7], we could obtain a similar error bound as above (see also Corollary 2.8 of Mangasarian and Shiau [8]). The next result immediately follows from Lemma 4.3. It shows that the minimum norm point $\mathrm{Nr}(C(P))$ lies within distance $\sqrt{\delta_1}$ (or $\sqrt{\delta_2}$) of x_{k-1} (or y_k) when tolerance δ_1 (or δ_2) is introduced to the termination criteria.

Corollary 4.4. *If* $\alpha_k\leqslant\|x_{k-1}\|^2\leqslant\alpha_k+\delta_1$ *in Step* 1, *then*

$$\|x_{k-1}-\mathrm{Nr}(C(P))\|^2\leqslant\delta_1.$$

If $\beta_k\leqslant\|y_k\|^2\leqslant\beta_k+\delta_2$ *in Step* 3, *then*

$$\|y_k-\mathrm{Nr}(C(P))\|^2\leqslant\delta_2. \qquad \square$$

The following theorem provides the absolute and relative errors of the norm of current point.

Theorem 4.5. *Assume* $x_{k-1} \neq 0$ *and* $y_k \neq 0$. *If* $\alpha_k \leqslant \|x_{k-1}\|^2 \leqslant \alpha_k + \delta_1$, *then*

$$0 \leqslant \|x_{k-1}\| - \|\mathrm{Nr}(C(P))\| \leqslant \min\left\{\sqrt{\delta_1}, \sqrt{\alpha_k + \delta_1}, \frac{\delta_1}{\|x_{k-1}\|}\right\}.$$

Moreover if $\alpha_k > 0$ *and* $\mathrm{Nr}(C(P)) \neq 0$, *then*

$$0 \leqslant \frac{\|x_{k-1}\| - \|\mathrm{Nr}(C(P))\|}{\|\mathrm{Nr}(C(P))\|} \leqslant \min\left\{\frac{\|x_{k-1}\|}{\alpha_k}\sqrt{\delta_1}, \frac{\delta_1}{\alpha_k}\right\}.$$

If $\beta_k \leqslant \|y_k\|^2 \leqslant \beta_k + \delta_2$, *then*

$$0 \leqslant \|y_k\| - \|\mathrm{Nr}(C(P))\| \leqslant \min\left\{\sqrt{\delta_2}, \sqrt{\beta_k + \delta_2}, \frac{\delta_2}{\|y_k\|}\right\}.$$

Moreover if $\beta_k > 0$ *and* $\mathrm{Nr}(C(P)) \neq 0$, *then*

$$0 \leqslant \frac{\|y_k\| - \|\mathrm{Nr}(C(P))\|}{\|\mathrm{Nr}(C(P))\|} \leqslant \min\left\{\frac{\|y_k\|}{\beta_k}\sqrt{\delta_2}, \frac{\delta_2}{\beta_k}\right\}.$$

Proof. From Corollary 4.4 it holds that $\|x_{k-1}\| - \|\mathrm{Nr}(C(P))\| \leqslant \|x_{k-1} - \mathrm{Nr}(C(P))\| \leqslant \sqrt{\delta_1}$. We see from the assumption $\|x_{k-1}\|^2 \leqslant \alpha_k + \delta_1$ that

$$\|x_{k-1}\| - \|\mathrm{Nr}(C(P))\| \leqslant \sqrt{\alpha_k + \delta_1} - \|\mathrm{Nr}(C(P))\| \leqslant \sqrt{\alpha_k + \delta_1}.$$

Theorem 3.2 implies that

$$0 \leqslant \|x_{k-1}\| - \|\mathrm{Nr}(C(P))\| \leqslant \|x_{k-1}\| - \frac{\alpha_k}{\|x_{k-1}\|} = \frac{\|x_{k-1}\|^2 - \alpha_k}{\|x_{k-1}\|} \leqslant \frac{\delta_1}{\|x_{k-1}\|}.$$

Hence we obtain

$$0 \leqslant \|x_{k-1}\| - \|\mathrm{Nr}(C(P))\| \leqslant \min\left\{\sqrt{\delta_1}, \sqrt{\alpha_k + \delta_1}, \frac{\delta_1}{\|x_{k-1}\|}\right\}.$$

When $\alpha_k > 0$, we see that $\sqrt{\delta_1} < \sqrt{\alpha_k + \delta_1}$ and

$$\begin{aligned} 0 \leqslant \frac{\|x_{k-1}\| - \|\mathrm{Nr}(C(P))\|}{\|\mathrm{Nr}(C(P))\|} &\leqslant \min\left\{\sqrt{\delta_1}, \frac{\delta_1}{\|x_{k-1}\|}\right\} / \|\mathrm{Nr}(C(P))\| \\ &\leqslant \frac{\|x_{k-1}\|}{\alpha_k} \min\left\{\sqrt{\delta_1}, \frac{\delta_1}{\|x_{k-1}\|}\right\} \\ &= \min\left\{\frac{\|x_{k-1}\|}{\alpha_k}\sqrt{\delta_1}, \frac{\delta_1}{\alpha_k}\right\}. \end{aligned}$$

The inequalities with respect to y_k can be seen in the same way as x_{k-1}. □

The step size λ_k in Step 4 could be another measure to evaluate the error of current point. Given a real number λ, we consider the set $P(x_{k-1}, \lambda)$ of the affine combinations of x_{k-1} and points $p \in P$ with coefficient λ:

$$P(x_{k-1}, \lambda) = \{p' \mid p' = (1-\lambda)x_{k-1} + \lambda p, p \in P\}.$$

Lemma 4.6. *If* $0 \leqslant \lambda \leqslant 1$ *and* $z(\lambda)^{\mathrm{t}} y_k \leqslant z(\lambda)^{\mathrm{t}} p$ *for every* $p \in P$, *then*

$$z(\lambda) = \mathrm{Nr}(C(P(x_{k-1}, \lambda))).$$

Proof. Let p be an arbitrary point of P. Then

$$z(\lambda)^{\mathrm{t}}\{(1-\lambda)x_{k-1} + \lambda p\} - z(\lambda)^{\mathrm{t}}\{(1-\lambda)x_{k-1} + \lambda y_k\} = \lambda z(\lambda)^{\mathrm{t}}(p - y_k) \geqslant 0,$$

by the assumption. This means that $C(P(x_{k-1}, \lambda)) \subseteq H^+(z(\lambda), \|z(\lambda)\|^2)$. Since $y_k \in C(P)$ and $0 \leqslant \lambda \leqslant 1$, $z(\lambda) = (1-\lambda)x_{k-1} + \lambda y_k \in C(P(x_{k-1}, \lambda))$. Hence it follows from Theorem 1.1 that $z(\lambda) = \mathrm{Nr}(C(P(x_{k-1}, \lambda)))$. □

Lemma 4.7. *Assume* $0 \leqslant \lambda \leqslant 1$ *and* $z(\lambda)^{\mathrm{t}} y_k \leqslant z(\lambda)^{\mathrm{t}} p$ *for every* $p \in P$. *Then*

$$z(\lambda)^{\mathrm{t}} x \geqslant \|z(\lambda)\|^2 + (1-\lambda) z(\lambda)^{\mathrm{t}}(x - x_{k-1})$$

for every point $x \in C(P)$.

Proof. Let x be an arbitrary point of $C(P)$ and let $x(\lambda) = (1-\lambda)x_{k-1} + \lambda x$. Then $x(\lambda) \in C(P(x_{k-1}, \lambda)) \subseteq H^+(z(\lambda), \|z(\lambda)\|^2)$ from Lemma 4.6. Hence

$$\begin{aligned} z(\lambda)^{\mathrm{t}} x &= z(\lambda)^{\mathrm{t}}\{x - x(\lambda) + x(\lambda)\} = z(\lambda)^{\mathrm{t}}\{x - x(\lambda)\} + z(\lambda)^{\mathrm{t}} x(\lambda) \\ &\geqslant (1-\lambda) z(\lambda)^{\mathrm{t}}(x - x_{k-1}) + \|z(\lambda)\|^2. \qquad \square \end{aligned}$$

Theorem 4.8. $\|x_k - \mathrm{Nr}(C(P))\|^2 \leqslant 2(1-\lambda_k)(x_{k-1}^{\mathrm{t}} x_k - \alpha_{k+1})$.

Proof. Since $x_k = z(\lambda_k)$, $x_k^{\mathrm{t}} y_k \leqslant x_k^{\mathrm{t}} p$ for every $p \in P$ by the definition of λ_k in Step 4 and the choice of y_k. Take $\mathrm{Nr}(C(P))$ as x of Lemma 4.7 and we see by the definitions of $\mathrm{Nr}(C(P))$ and α_{k+1} that

$$\begin{aligned} \|x_k - \mathrm{Nr}(C(P))\|^2 &= \|x_k\|^2 - 2x_k^{\mathrm{t}}\,\mathrm{Nr}(C(P)) + \|\mathrm{Nr}(C(P))\|^2 \\ &\leqslant \|x_k\|^2 - 2\{\|x_k\|^2 + (1-\lambda_k)x_k^{\mathrm{t}}(\mathrm{Nr}(C(P)) - x_{k-1})\} \\ &\quad + \|\mathrm{Nr}(C(P))\|^2 \\ &\leqslant 2\|x_k\|^2 - 2\{\|x_k\|^2 + (1-\lambda_k)x_k^{\mathrm{t}}(\mathrm{Nr}(C(P)) - x_{k-1})\} \\ &= 2(1-\lambda_k)x_k^{\mathrm{t}}(x_{k-1} - \mathrm{Nr}(C(P))) \\ &\leqslant 2(1-\lambda_k)(x_k^{\mathrm{t}} x_{k-1} - \alpha_{k+1}). \qquad \square \end{aligned}$$

5. Algorithm $\mathcal{N}_2$ for finding Nr($C(P)$, $C(Q)$)

We consider Problem 2 of finding a pair of Nr($C(P)$, $C(Q)$). As pointed out in [1] it is equivalent to Problem 1 for $P-Q$. Note that

$$\min\{x^{\mathrm{t}}r \mid r\in P-Q\}=\min\{x^{\mathrm{t}}p \mid p\in P\}-\max\{x^{\mathrm{t}}q \mid q\in Q\},$$

and each step below will be seen to be equivalent to each step of the algorithm $\mathcal{N}_1$.

Algorithm $\mathcal{N}_2(P, Q)$.

Step 0. Choose a point x_0^P from $C(P)$ and a point x_0^Q from $C(Q)$.
$x_0 := x_0^P - x_0^Q$, $k:=1$.

Step 1. $\alpha_k^P := \min\{x_{k-1}^{\mathrm{t}}p \mid p\in P\}$ and $\alpha_k^Q := \max\{x_{k-1}^{\mathrm{t}}q \mid q\in Q\}$.
$\alpha_k := \alpha_k^P - \alpha_k^Q$.
If $\|x_{k-1}\|^2 \leqslant \alpha_k$, then $\hat{x}^P := x_{k-1}^P$, $\hat{x}^Q := x_{k-1}^Q$ and stop.

Step 2. $P_k := \{p \mid p\in P$ and $x_{k-1}^{\mathrm{t}}p=\alpha_k^P\}$ and $Q_k := \{q \mid q\in Q$ and $x_{k-1}^{\mathrm{t}}q=\alpha_k^Q\}$.
Call $\mathcal{N}_2(P_k, Q_k)$ and let (y_k^P, y_k^Q) be a pair of Nr($C(P_k)$, $C(Q_k)$).
$y_k := y_k^P - y_k^Q$.

Step 3. $\beta_k^P := \min\{y_k^{\mathrm{t}}p \mid p\in P\}$ and $\beta_k^Q := \max\{y_k^{\mathrm{t}}q \mid q\in Q\}$.
$\beta_k := \beta_k^P - \beta_k^Q$.
If $\|y_k\|^2 \leqslant \beta_k$, then $\hat{x}^P := y_k^P$, $\hat{x}^Q := y_k^Q$ and stop.

Step 4. $\gamma_k^P(\lambda) := \max\{\{(1-\lambda)x_{k-1}+\lambda y_k\}^{\mathrm{t}}(y_k^P - p) \mid p\in P\}$ and
$\gamma_k^Q(\lambda) := \min\{\{(1-\lambda)x_{k-1}+\lambda y_k\}^{\mathrm{t}}(y_k^Q - q) \mid q\in Q\}$.
$\lambda_k := \max\{\lambda \mid \gamma_k^P(\lambda) \leqslant \gamma_k^Q(\lambda)\}$.
$x_k^P := (1-\lambda_k)x_{k-1}^P + \lambda_k y_k^P$ and $x_k^Q := (1-\lambda_k)x_{k-1}^Q + \lambda_k y_k^Q$.
$x_k := x_k^P - x_k^Q$, $k:=k+1$ and go to Step 1.

Let r be a point of $(P-Q)\backslash(P_k-Q_k)$. Then $r=p-q$ for some $p\in P$ and $q\in Q$ such that either $p\notin P_k$ or $q\notin Q_k$. Therefore we cannot simplify Step 3 to $\beta_k^P := \min\{y_k^{\mathrm{t}}p \mid p\in P\backslash P_k\}$ and $\beta_k^Q := \max\{y_k^{\mathrm{t}}q \mid q\in Q\backslash Q_k\}$. The value λ_k determined in Step 4 is easily seen to be equal to $\max\{\lambda \mid \{(1-\lambda)x_{k-1}+\lambda y_k\}^{\mathrm{t}}y_k \leqslant \{(1-\lambda)x_{k-1}+\lambda y_k\}^{\mathrm{t}}(p-q)$ for every $p\in P$ and every $q\in Q\}$.

In exactly the same way as in Section 2 we obtain the following lemmas and finite convergence of $\mathcal{N}_2$. We omit the proof.

Lemma 5.1. $0<\lambda_k<1$ *for* $k=1, 2, \ldots$. □

Lemma 5.2. $(\hat{x}^P, \hat{x}^Q)\in$ Nr($C(P)$, $C(Q)$). □

Lemma 5.3. *If* $\|x_{k-1}\|^2 > \alpha_k$, *then either* P_k *is a proper subset of* P *and* $C(P_k)$ *is a proper face of* $C(P)$ *or* Q_k *is a proper subset of* Q *and* $C(Q_k)$ *is a proper face of* $C(Q)$. □

Lemma 5.4. *There exists a pair of points* $(\bar{p}, \bar{q})$ *such that*
(i) $\bar{p} \notin P_k$ *or* $\bar{q} \notin Q_k$,
(ii) $\bar{p} \in P_{k+1}$ *and* $\bar{q} \in Q_{k+1}$,
(iii) $y_k^{\mathrm{t}}(\bar{p}-\bar{q}) < \|y_k\|^2$. □

Lemma 5.5. $y_k \in C(P_{k+1} - Q_{k+1})$ *for* $k = 1, 2, \ldots$. □

Lemma 5.6. $\|y_{k+1}\| < \|y_k\|$ *for* $k = 1, 2, \ldots$. □

Lemma 5.7. *When either P or Q consists of a single point, Algorithm* $\mathcal{N}_2$ *reduces to Algorithm* $\mathcal{N}_1$ *and hence terminates within a finite number of iterations.* □

Theorem 5.8. *When both P and Q consist of finitely many points, Algorithm* $\mathcal{N}_2$ *provides a pair of points of* $\mathrm{Nr}(C(P), C(Q))$. □

6. Computational experiments

To make clear the behavior of $\mathcal{N}_1$ and to compare its efficiency with that of Wolfe's algorithm we have made programs of two methods in Sun Pascal running on a Sun IPX workstation. For the implementation of $\mathcal{N}_1$ we have made devices as follows.

(1) We did not call $\mathcal{N}_1$ recursively to suppress the overhead time of recursive calls when P_k has less than three points. In fact when P_k is a singleton, $\{p\}$, then $p = \mathrm{Nr}(C(P_k))$. For the case where P_k has two points, e.g., p^1 and p^2, let us define

$$\lambda^* = \frac{(p^1)^{\mathrm{t}}(p^1 - p^2)}{\|p^1 - p^2\|^2}.$$

Then $\mathrm{Nr}(C(P_k))$ is determined in the following way:
(1a) $\mathrm{Nr}(C(P_k)) = p^2$ if $\lambda^* \geqslant 1$,
(1b) $\mathrm{Nr}(C(P_k)) = p^1$ if $\lambda^* \leqslant 0$, and
(1c) $\mathrm{Nr}(C(P_k)) = (1-\lambda^*)p^1 + \lambda^* p^2$ if $0 < \lambda^* < 1$.

(2) Let us split P_{k+1} into two parts: $\hat{P}_{k+1} = P_{k+1} \setminus P_k$ and $\bar{P}_{k+1} = P_{k+1} \cap P_k$. As pointed out in the proof of Lemma 2.4 $\hat{P}_{k+1}$ is found as the set of points attaining λ_k in Step 4. To make $\bar{P}_{k+1}$ we have to evaluate $x_k^{\mathrm{t}} p$ only for points of P_k and to collect those points satisfying $x_k^{\mathrm{t}} p = x_k^{\mathrm{t}} y_k$. From the preliminary experiment we observed that the set P_k is always fairly small regardless of the size of P and the dimension. Therefore this device will reduce the computation.

We consider the following two types of problems. Here $X=\{x \mid e \leqslant x \leqslant a\}$ is an n-cube defined by $e=(1, \ldots, 1)^{t}$ and $a=(50, \ldots, 50)^{t}$.

Type 1: We consider the 2-dimensional problem. The set P consists of m points distributed uniformly over X. We have varied the number of points, m, as 500, 1000, 2000, 3000, 5000, 10 000, 20 000, 25 000 and 30 000 and have solved 10 problems for each m.

Type 2: We consider $n=20$. The set P consists of m points distributed uniformly over the set of integer points in X. We have varied the number of points, m, as 200, 400, 600, 800 and 1000 and have solved 10 problems for each m.

Figure 6.1 is the logarithmic plot of computational time t, exclusive of input and output, of the proposed algorithm and Wolfe's algorithm for the problem of Type 1 as a function of m. The regression line, shown as the dashed line, for the proposed algorithm $\mathcal{N}_1$ is $\log t=-1.221+0.853 \log m$ and that for Wolfe's algorithm is $\log t=-1.441+0.972 \log m$. Namely, the computational time t of $\mathcal{N}_1$ is approximately $0.060 m^{0.853}$ millisecond, which grows more slowly than that of Wolfe's algorithm and even more slowly than the worst case analysis in Theorem 2.9. Figure 6.2 shows the plot of $\log t$ versus $\log m$ for the problem of Type 2. No firm conclusion should be drawn from such limited experiments, however, it is worth mentioning that both the algorithms exhibit empirical computational complexity lower than the linear order of the number of given points as long as the dimension is fixed. To compare the efficiency of the algorithms further experiments should be needed. This task is beyond the aim of this paper, so we leave it for future research.

7. Concluding remarks

When we are given a polytope $X=\{x \mid Ax \geqslant b, x \geqslant 0\}$ instead of the finite set P, we could apply Algorithm $\mathcal{N}_1$ to find the minimum Euclidean norm point in X. Determining α_k and β_k is reduced to a linear program on X. The step size λ_k in Step 4 is given by

$$\max\{\lambda \mid \{(1-\lambda)x_{k-1}+\lambda y_k\}^{t} y_k \leqslant \min\{\{(1-\lambda)x_{k-1}+\lambda y_k\}^{t} x \mid x \in X\}\}.$$

By the duality theorem of linear program it is equivalent to

$$\max\{\lambda \mid \{(1-\lambda)x_{k-1}+\lambda y_k\}^{t} y_k \leqslant \max\{b^{t} z \mid A^{t} z \leqslant (1-\lambda)x_{k-1}+\lambda y_k,\ z \geqslant 0\}\},$$

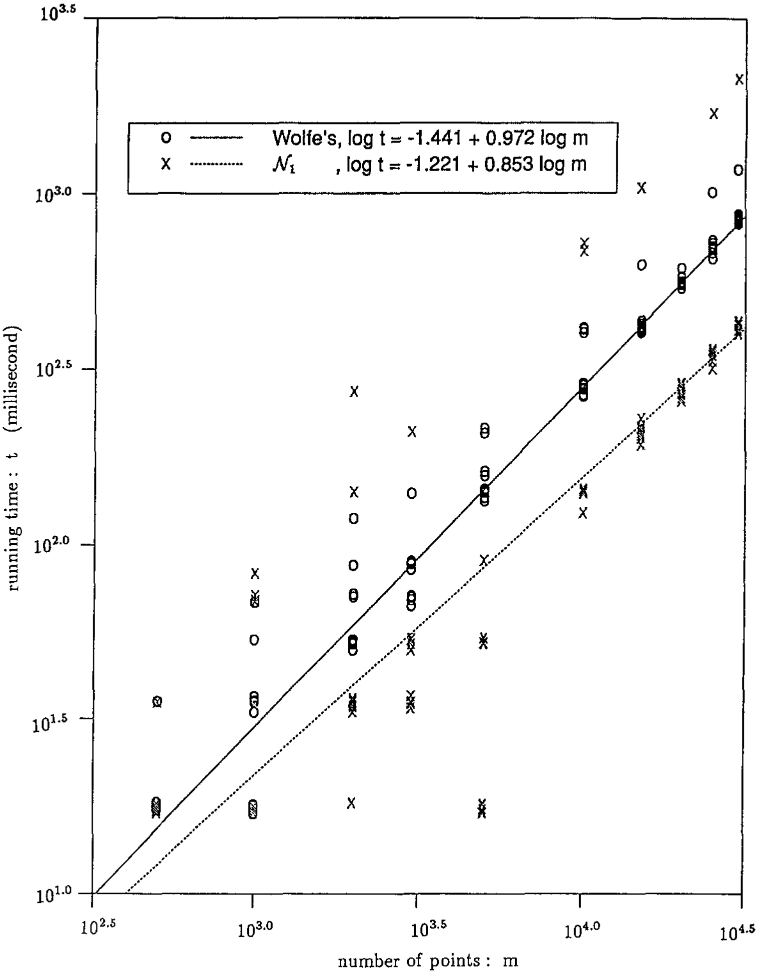

Fig. 6.1. Plot of log t vs. log m for Type 1.

and hence λ_k is obtained by solving the linear program

$$\begin{aligned} \max \quad & \lambda, \\ \text{s.t.} \quad & \{(1-\lambda)x_{k-1}+\lambda y_k\}^t y_k \leqslant b^t z, \\ & A^t z \leqslant (1-\lambda)x_{k-1}+\lambda y_k, \\ & z \geqslant 0. \end{aligned}$$

The set P_k in Step 2 must be replaced by the face of optimal solutions of

$$\begin{aligned} \min \quad & x_{k-1}^t x, \\ \text{s.t.} \quad & Ax \geqslant b, \ x \geqslant 0. \end{aligned}$$

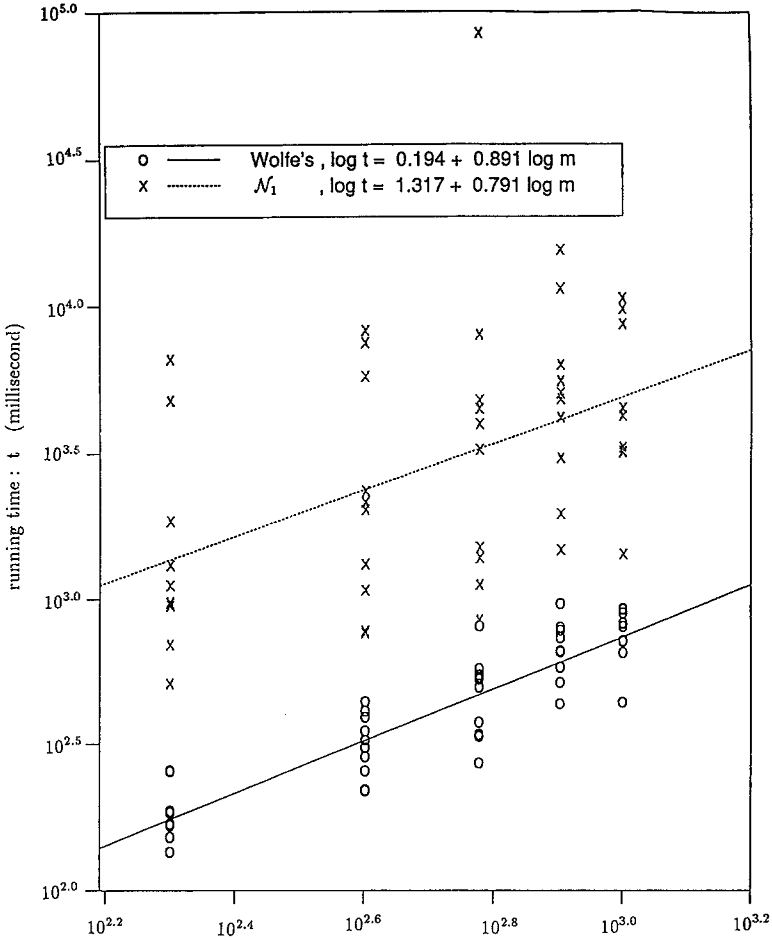

Fig. 6.2. Plot of log t vs. log m for Type 2.

Though the face is identified theoretically by $\{x \mid Ax \geqslant b, x \geqslant 0, x_{k-1}^{t}x = \alpha_k\}$, we should note that it is very vulnerable to the numerical error of α_k and it has one more constraint than X has.

The proposed Algorithm $\mathcal{N}_1$ has something in common with the classical algorithm by Frank and Wolfe [3]. Given the current iterate z_{k-1}, the latter algorithm, when applied to the minimum norm point problem, finds a minimum point of $z_{k-1}^{t}x$ over the convex hull of P. Let w_k be a minimum point. Then it finds a point with the minimum norm on the line segment between z_{k-1} and w_k and set the point to be the next iterate. The crucial differences between Franke–Wolfe's algorithm and ours are:

- the way of choosing w_k when there are multiple minimum points of $z_{k-1}^{t}x$, and
- the way of choosing the new iterate z_k.

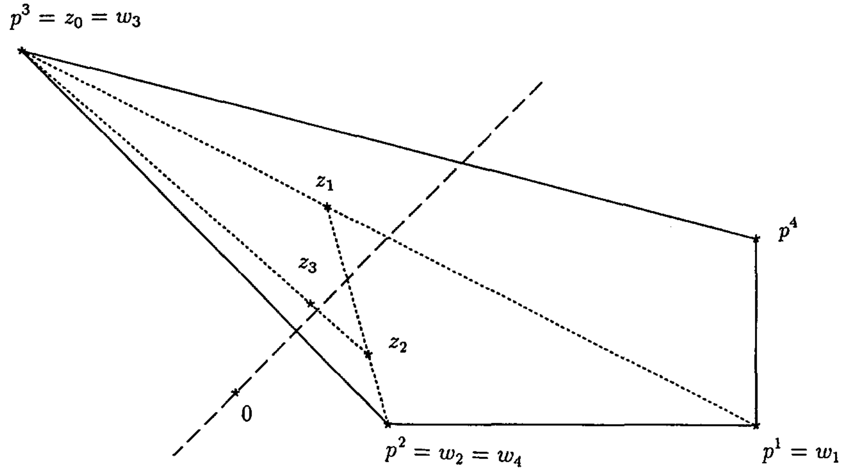

Fig. 7.1.

The iterates of Frank–Wolfe's algorithm are shown in Figure 7.1 when applied to the same example in Section 3. The broken line connects the origin with the minimum norm point $\mathrm{Nr}(C(P))$ which was denoted by y_3 in Figure 3.1. One can easily see that the iterates are zigzagging alternately on either side of the broken line toward y_3. It was shown by Canon and Cullum [2, Theorem 2] that for every constant a and for every $\varepsilon > 0$,

$$\|z_k\|^2 - \|\mathrm{Nr}(C(P))\|^2 \geqslant a/k^{1+\varepsilon}$$

holds for infinitely many k.

Acknowledgment

The authors would like to thank anonymous referees for their valuable comments, especially for bringing their attention to the papers [2], [7] and [8]. This research was partly carried out when the second author was supported by the Alexander von Humboldt-Foundation, Germany. It is also supported by the Grant-in-Aid for Scientific Research of the Ministry of Education, Science and Culture, Japan.

References

[1] M.D. Canon and C.D. Cullum, "The determination of optimum separating hyperplanes I. A finite step procedure," RC 2023, IBM Watson Research Center (Yorktown Heights, NY, 1968).

[2] M.D. Canon and C.D. Cullum, "A tight upper bound on the rate of convergence of the Frank–Wolfe algorithm," *SIAM Journal on Control* 6 (1968) 509–516.

[3] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Research Logistics Quarterly* 3 (1956) 95–110.

[4] R.M. Freund, "Dual gauge programs, with applications to quadratic programming and the minimum-norm problem," *Mathematical Programming* 38 (1987) 47–67.

[5] S. Fujishige and P. Zhan, "A dual algorithm for finding the minimum-norm point in a polytope," *Journal of the Operations Research Society of Japan* 33 (1990) 188–195.

[6] S. Fujishige and P. Zhan, "A dual algorithm for finding a nearest pair of points in two polytopes," DP No. 470, Institute of Socio-Economic Planning, University of Tsukuba (Tsukuba, 1991).

[7] O.L. Mangasarian and R. De Leone, "Error bounds for strongly convex program and (super)linearly convergent iterative schemes for the least 2-norm solution of linear programs," *Applied Mathematics and Optimization* 17 (1988) 1–14.

[8] O.L. Mangasarian and T.-H. Shiau, "Error bounds for monotone linear complementarity problems," *Mathematical Programming* 36 (1986) 81–89.

[9] P. Wolfe, "Finding the nearest point in a polytope," *Mathematical Programming* 11 (1976) 128–149.