Ann Oper Res
DOI 10.1007/s10479-014-1707-2

A characterization theorem and an algorithm for a convex hull problem

Bahman Kalantari

Abstract Given $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$ and $p \in \mathbb{R}^m$, testing if $p \in conv(S)$, the convex hull of S, is a fundamental problem in computational geometry and linear programming. First, we prove a Euclidean *distance duality*, distinct from classical separation theorems such as Farkas Lemma: p lies in $conv(S)$ if and only if for each $p' \in conv(S)$ there exists a *pivot*, $v_j \in S$ satisfying $d(p', v_j) \geq d(p, v_j)$. Equivalently, $p \notin conv(S)$ if and only if there exists a *witness*, $p' \in conv(S)$ whose Voronoi cell relative to p contains S. A witness separates p from $conv(S)$ and approximate $d(p, conv(S))$ to within a factor of two. Next, we describe the *Triangle Algorithm*: given $\epsilon \in (0, 1)$, an *iterate*, $p' \in conv(S)$, and $v \in S$, if $d(p, p') < \epsilon d(p, v)$, it stops. Otherwise, if there exists a pivot v_j, it replace v with v_j and p' with the projection of p onto the line $p'v_j$. Repeating this process, the algorithm terminates in $O(mn \min\{\epsilon^{-2}, c^{-1} \ln \epsilon^{-1}\})$ arithmetic operations, where c is the *visibility factor*, a constant satisfying $c \geq \epsilon^2$ and $\sin(\angle pp'v_j) \leq 1/\sqrt{1+c}$, over all iterates p'. In particular, the geometry of the input data may result in efficient complexities such as $O(mn\sqrt[t]{\epsilon^{-2}} \ln \epsilon^{-1})$, t a natural number, or even $O(mn \ln \epsilon^{-1})$. Additionally, (i) we prove a *strict distance duality* and a related minimax theorem, resulting in more effective pivots; (ii) describe $O(mn \ln \epsilon^{-1})$-time algorithms that may compute a witness or a good approximate solution; (iii) prove *generalized distance duality* and describe a corresponding generalized Triangle Algorithm; (iv) prove a *sensitivity theorem* to analyze the complexity of solving LP feasibility via the Triangle Algorithm. The Triangle Algorithm is practical and competitive with the simplex method, sparse greedy approximation and first-order methods. Finally, we discuss future work on applications and generalizations.

Keywords Convex hull · Linear programming · Duality · Approximation algorithms · Gilbert's algorithm · Frank–Wolfe algorithm · Minimax

B. Kalantari (✉)
Department of Computer Science, Rutgers University, Piscataway, NJ, USA
e-mail: kalantari@cs.rutgers.edu

Published online: 24 August 2014

1 Introduction

Given a set $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$ and a distinguished point $p \in \mathbb{R}^m$, we consider the problem of testing if p lies in $conv(S)$, the convex hull of S. Throughout the article we shall refer to this problem as the *convex hull decision problem*, or simply as problem (P). The convex hull decision problem is a basic problem in computational geometry and a very special case of the *convex hull problem*, a problem that according to Goodman and O'Rourke (2004), is a "catch-all phrase for computing various descriptions of a polytope that is either specified as the convex hull of a finite point set or the intersection of a finite number of halfspaces." The descriptions include those of vertices, facets, and adjacencies.

Problem (P) is not only a fundamental problem in computational geometry, but in linear programming (LP). This can be argued on different grounds. On the one hand problem (P) is a very special case of LP. On the other hand, it is well known that by the LP duality theory, the general LP problem may be cast as a single LP feasibility problem, see e.g. Chvátal (1983). The LP feasibility problem can then be converted into problem (P) via several different approaches. To argue the significance of (P) in linear programming, it can be justified that the two most famous polynomial-time LP algorithms, the ellipsoid algorithm of Khachiyan (1979) and the projective algorithm of Karmarkar (1984), are in fact explicitly or implicitly designed to solve a case of problem (P) where $p = 0$, see Jin and Kalantari (2006). Furthermore, using an approach suggested by Chvátal, in Jin and Kalantari (2006) it is shown that there is a direct connection between a general LP feasibility and this homogeneous case of problem (P), over integer or real input data. For integer inputs all known polynomial-time LP algorithms—when applied to solve problem (P)—would exhibit a theoretical complexity that is polynomial in m, n, and the size of encoding of the input data, often denoted by L, see e.g. Khachiyan (1979). The number L can generally be taken to be dependent on the logarithm of mn and of the logarithm of the absolute value of the largest entry in the input data. These are sometimes known as *weakly polynomial-time* algorithms. No *strongly polynomial-time* algorithm is known for LP, i.e. an algorithm that would in particular solve problem (P) in time complexity polynomial in m and n.

Problem (P) finds applications in computational geometry itself, in particular among the variations of the convex hull problem. One such a variation is the *irredundancy problem*, the problem of computing all the vertices of $conv(S)$, see Goodman and O'Rourke (2004). Clearly, any algorithm for LP can be used to solve the irredundancy problem by solving a sequence of $O(n)$ convex hull decision problems. Clarkson (1994), has given a more efficient algorithm than the straightforward approach of solving n linear programs. The complexity of his algorithm depends on the number of vertices of the convex hull. Furthermore, by using an approach originally due to Matoušek (1993) for solving closely related linear programming problems, Chan (1996) has given a faster algorithm for the irredundancy problem. What is generally considered to be the convex hull problem is a problem more complicated than (P) and the irredundancy problem, requiring the description of $conv(S)$ in terms of its vertices, facets and adjacencies, see Chazelle (1993) who offers an optimal algorithm.

Problem (P) can also be formulated as the minimization of a convex quadratic function over a simplex. This particular convex program has found applications in statistics, approximation theory, and machine learning, see e.g Clarkson (2008) and Zhang (2003) who consider the analysis of a greedy algorithm for the more general problem of minimizing certain convex functions over a simplex (equivalently, maximizing concave functions over a simplex). The oldest version of such greedy algorithms is Frank–Wolfe algorithm, Frank and Wolfe

(1956). Special cases of the problem include support vector machines (SVM), approximating functions as convex combinations of other functions, see e.g. Clarkson (2008). The problem of computing the closest point of the convex hull of a set of points to the origin, known as *polytope distance* is related to the case of problem (P) where p is the origin. In some applications the polytope distance refers to the distance between two convex hulls. Gilbert's algorithm Gilbert (1966) for the polytope distance problem is one of the earliest known algorithms for the problem. Gärtner and Jaggi (2009) show Gilbert's algorithm coincides with Frank–Wolfe algorithm when applied to the minimization of a convex quadratic function over a simplex. In this case the algorithm is known as *sparse greedy approximation*. For many other results regarding the applications of the minimization of a quadratic function over a simplex, see the bibliographies in Zhang (2003), Clarkson (2008) and Gärtner and Jaggi (2009). Clarkson (2008) gives a through analysis of Frank–Wolfe and its variations. Another class of algorithms that are applicable to the convex hull problem are the so-called *first-order* methods, see Nesterov (2013) and Lan et al. (2011).

Despite all previous approaches and achievements in solving problem (P), various formulations and algorithms, investigation of this fundamental problem will most likely continue in the future. The present article focuses on solving problem (P). It reveals new and interesting geometric properties of this fundamental convex hull problem, including characterization theorems, leading to new separating hyperplane theorems, called *distance duality* theorems. It then utilizes the distance dualities to describe the *Triangle Algorithm*. We analyze theoretical complexity of the Triangle Algorithm showing that it is very attractive for solving problem (P). Additionally, utilizing the Triangle Algorithm, together by proving a sensitivity theorem, we describe a new algorithm for the LP feasibility problem and analyze its complexity.

The article is a substantially extended version of Kalantari (2012a). It is organized as follows. In Sect. 2 we give a detailed overview of the Triangle Algorithm, its properties and applications. Specifically, in Sect. 2.1 we describe the geometry of the Triangle Algorithm and introduce two geometric problems that can be viewed as problems that are dual to (P). In Sect. 2.2 we describe three complexity bounds for the Triangle Algorithm in terms of problem parameters. In Sect. 2.3 we describe the complexity of the Triangle Algorithm for solving LP feasibility and optimization problems. In Sects. 2.4–2.6 we give a comparison of the complexity bound of the Triangle Algorithm to that of several alternate algorithms that can solve the convex hull decision problem, namely, the simplex method, sparse greedy approximation, and first-order methods. In Sect. 2.7 we describe connections between problem (P) and the general LP feasibility and optimization problems, as well as citation of some of the vast and relevant literature on LP. In Sect. 2.8, we give the outline of results in the remaining sections.

2 Overview of Triangle Algorithm, properties and applications

Given $u, v \in \mathbb{R}^m$, the Euclidean distance is $d(u, v) = \sqrt{\sum_{i=1}^m (u_i - v_i)^2} = \|u - v\|$. The Triangle Algorithm takes as input a subset $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$ and a point $p \in \mathbb{R}^m$, as well as a tolerance $\epsilon \in (0, 1)$.

Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, $p, \epsilon \in (0, 1)$)

- **Step 0.** (**Initialization**) Let $p' = v = \text{argmin}\{d(p, v_i) : v_i \in S\}$.
- **Step 1.** If $d(p, p') < \epsilon d(p, v)$, output p' as *ϵ-approximate solution*, stop. Otherwise, if there exists $v_j \in S$ where $d(p', v_j) \geq d(p, v_j)$, call v_j a *pivot* (or *p-pivot*); replace v with v_j. If no pivot exists, then output p' as a *witness* (or *p-witness*), stop.
- **Step 2.** Given $p' = \sum_{i=1}^{n} \alpha_i v_i$, $\sum_{i=1}^{n} \alpha_i = 1, \alpha_i \geq 0$, compute the new iterate $p'' = \sum_{i=1}^{n} \alpha_i' v_i$, $\sum_{i=1}^{n} \alpha_i' = 1, \alpha_i' \geq 0$ as the nearest point to p on the line segment $p'v_j$. Replace p' with p'', α_i with α_i', for all i, go to Step 1.

The justification in the name of the algorithm lies in the fact that in each iteration the algorithm searches for a triangle $\triangle pp'v_j$ where $v_j \in S$, $p' \in conv(S)$, such that $d(p', v_j) \geq d(p, v_j)$. Given that such triangle exists, it uses v_j as a pivot to "pull" the current iterate p' closer to p to get a new iterate $p'' \in conv(S)$. The Triangle Algorithm either computes an ϵ-approximate solution $p' \in conv(S)$ satisfying

$$d(p', p) < \epsilon R, \quad R = \max\{d(v_j, p) : v_j \in S\}, \tag{1}$$

or a witness $p' \in conv(S)$ satisfying

$$d(p', v_i) < d(p, v_i), \quad \forall i = 1, \ldots, n. \tag{2}$$

Remark 1 The Triangle Algorithm seeks to find an approximate solution $p' \in conv(S)$ whose relative error, $d(p', p)/R$, is within prescribed error ϵ, see (1), as opposed to computing a point within a prescribed absolute error. For this reason we make three important points.

(i) Stating the quality of approximation in terms of the relative error (1) is a more meaningful measure of approximation than approximation to within prescribed absolute error. Consider the case where the diameter of $conv(S)$ is very small, say less than ϵ and $p \notin conv(S)$, but $d(p, S) < \epsilon$.

(ii) Clearly approximate solutions with a prescribed absolute error can also be computed, simply by replacing ϵ with ϵ/R. This only affects the complexity by a constant factor.

(iii) All existing algorithms for the convex hull decision problem that claim to produce an approximate solution with absolute error ϵ would necessarily have a complexity bound in terms of some parameters dependent on the data. In particular, this is the case if we solve the convex hull problem decision problem via polynomial-time LP algorithms, or approximation schemes such as sparse greedy approximation, or first-order methods (see (2.4–2.6)). Analogous to polynomial-time LP algorithms, for rational inputs S and p, when ϵ is sufficiently small the existence of an ϵ-approximate solution implies p lies in $conv(S)$.

We refer to a point p' satisfying (2) as *p-witness* or simply *witness* because this condition holds if and only if $p \notin conv(S)$. In this case we prove the Voronoi cell of p' with respect to the two point set $\{p, p'\}$ contains $conv(S)$ (see Fig. 1). Equivalently, the hyperplane that orthogonally bisects the line segment pp' separates p from $conv(S)$. The set W_p of all such witnesses is the intersection of $conv(S)$ and the open balls, $B_i = \{x \in \mathbb{R}^m : \ d(x, v_i) < r_i\}$, $i = 1, \ldots, n$. W_p is a convex subset of $conv(S)$ (see Fig. 3). It has the same dimension as $conv(S)$.

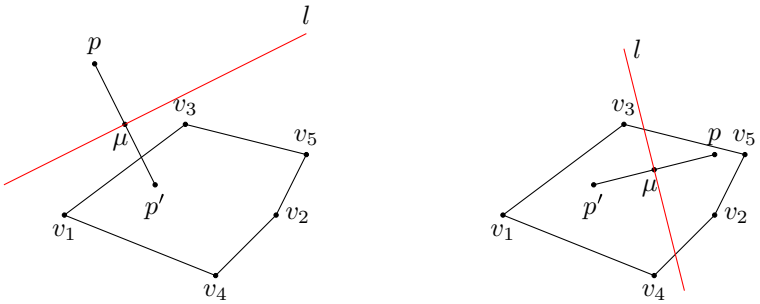

Fig. 1 Example of cases where orthogonal bisector of pp' does and does not separate p from $conv(S)$

The correctness of the Triangle Algorithm lies in a new duality (theorem of the alternative) we call *distance duality*:

Distance Duality
Precisely one of the two conditions is satisfied:

(i): For each $p' \in conv(S)$, there exists $v_j \in S$ such that $d(p', v_j) \geq d(p, v_j)$;
(ii): There exists $p' \in conv(S)$ such that $d(p', v_i) < d(p, v_i)$, for all $i = 1, \ldots, n$.

The first condition is valid if and only if $p \in conv(S)$, and the second condition if and only if $p \notin conv(S)$. From the description of the Triangle Algorithm we see that given a point $p' \in conv(S)$ that is not a witness, having $d(p, p')$ as the current *gap*, the Triangle Algorithm moves to a new point $p'' \in conv(S)$ where the new gap $d(p, p'')$ is reduced.

2.1 The geometry of the Triangle Algorithm

To arrive at the distance duality mentioned above, we first prove a characterization theorem that leads to this theorem of the alternative for problem (P). We remark here that the distance duality theorem is distinct from the classical Farkas lemma, or Gordan theorem. To arrive at this theorem we first prove:

$p \in conv(S)$ if and only if given any point $p' \in conv(S) \setminus \{p\}$, there exists v_j such that $d(p', v_j) > d(p, v_j)$.

Next, we show the strict inequality can be replaced with $d(p', v_j) \geq d(p, v_j)$. Thus the contrapositive theorem is:

$p \notin conv(S)$ if and only if there exists $p' \in conv(S)$ such that $d(p', v_i) < d(p, v_i)$, for all $i = 1, \ldots, n$.

These two results together imply the distance duality. A corollary of our characterization theorem reveals a geometric property of a set of balls. To describe this property, denote an open ball by B, its closure $\overline{B}$, and its boundary by ∂B, i.e.

$$\begin{aligned} B &= \{x \in \mathbb{R}^m : \quad d(x, v) < r\}, \quad \overline{B} = \{x \in \mathbb{R}^m : \quad d(x, v) \leq r\}, \\ \partial B &= \{x \in \mathbb{R}^m : \quad d(x, v) = r\}. \end{aligned} \tag{3}$$

Consider a set of open balls $B_i = \{x \in \mathbb{R}^m : \quad d(x, v_i) < r_i\}$, $i = 1, \ldots, n$, and let $S = \{v_1, \ldots, v_n\}$.

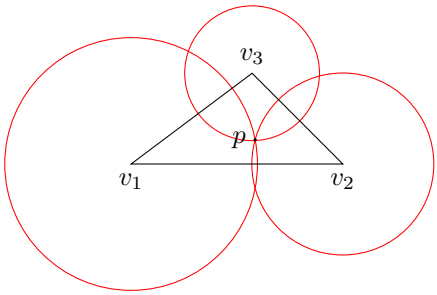

Fig. 2 A case with no p-witnesses: $p \in conv(S)$

Intersecting Balls Property:
If $p \in \cap_{i=1}^{n} \partial B_i$, then

$$p \in conv(S) \iff (\cap_{i=1}^{n} B_i) \cap conv(S) = \emptyset \iff (\cap_{i=1}^{n} \overline{B}_i) \cap conv(S) = \{p\}. \tag{4}$$

In words, suppose a set of open balls have a common boundary point p. Then p lies in the convex hull of their centers, if and only if the intersection of the open balls is empty, if and only if p is the only point in the intersection of the closure of the balls. A depiction of this property for a triangle is given in Fig. 2. This property suggests we can define a geometric *dual* for problem (P):

Problem (Q) (Intersecting Balls Problem):
Suppose there exists $p \in \cap_{i=1}^{n} \partial B_i$. Determine if $(\cap_{i=1}^{n} B_i) \cap conv(S)$ is nonempty.

In fact the intersecting balls problem can be stated in more generality:

Problem (Q′) (General Intersecting Balls Problem):
Suppose there exists $p \in \cap_{i=1}^{n} \partial B_i$. Determine if $(\cap_{i=1}^{n} B_i)$ is nonempty.

The Triangle Algorithm results in a fully polynomial-time approximation scheme for solving problems (Q) and (Q′) with the same time complexity as that of solving problem (P).

When a point p lies in $conv(S) \cap (\cap_{i=1}^{n} \partial B_i)$, the union of the balls, $\cup_{i=1}^{n} B_i$ is referred as the *forbidden zone* of the convex hull of the centers, see de Baisi et al. (2010) or (2011) for the definition and some of its properties. The notion of forbidden zone of a convex set is significant and intrinsic in the characterization of the so-called *mollified zone diagrams*, a variation of *zone diagram* of a finite set of points in the Euclidean plane, see de Biasi et al. (2011). The notion of zone diagram, introduced by Asano et al. (2007), is itself a very rich and interesting variation of the classical Voronoi diagram, see e.g. Aurenhammer (1991), Preparata and Shamos (1985). Forbidden zones help give a characterization of mollified zone diagrams, in particular a zone diagram, de Biasi et al. (2011). For some geometric properties of forbidden zones of polygons and polytopes, see Berkowitz et al. (2012).

2.2 Complexity bounds for the Triangle Algorithm

The Triangle Algorithm is geometric in nature, simple in description, and very easy to implement. Its complexity analysis also uses geometric ideas. We derive three different complexity bounds on the number of arithmetic operations of the Triangle Algorithm. The first bound on the number of arithmetic operations is

$$48mn\epsilon^{-2} = O(mn\epsilon^{-2}). \tag{5}$$

In the first analysis we will prove that when $p \in conv(S)$, the number of iterations K_ϵ, needed to get an approximate solution p' satisfying (1) is bounded above by $48\epsilon^{-2}$. In the worst-case each iteration of Step 1 requires $O(mn)$ arithmetic operations. However, it may also take only $O(m)$ operations. The number of arithmetic operations in each iteration of Step 2 is only $O(m+n)$ ($O(m)$ to find a pivot and the new iterate p'', plus $O(n)$ to update the coefficients α'_i of p''). Thus according to this complexity bound the Triangle Algorithm is a fully polynomial-time approximation scheme whose complexity for computing an ϵ-approximate solution is $O(mn\epsilon^{-2})$ arithmetic operation. In particular, for fixed ϵ the complexity of the algorithm is only $O(mn)$.

The worst-case iteration complexity estimate stated above is under the assumption of worst-case performance in the reduction of error in each iteration of the algorithm. Also the worst-case arithmetic complexity is under the assumption that in each iteration the algorithm has to go through the entire list of points in S to determine a pivot, or a witness. Thus our first worst-case arithmetic complexity bound for the Triangle Algorithm is under the assumption that worst-cases happen in each of the two steps. In practice we would expect a more efficient complexity. Note that by squaring the distances we have

$$d(p', v_j) \geq d(p, v_j) \iff \|p'\|^2 - \|p\|^2 \geq 2v_j^T(p' - p). \tag{6}$$

Thus Step 1 does not require taking square-roots. Neither does the computation of p''. These imply elementary operations are sufficient.

Our second complexity bound takes into account the relative location of p with respect to S. To describe the second complexity bound, let $A = [v_1, \ldots, v_n]$, the matrix of points in $S = \{v_1, \ldots, v_n\}$. We write $conv(A)$ to mean $conv(S)$. Given $\epsilon \in (0, 1)$, we associate a number $c(p, A, \epsilon)$ to problem (P), called *visibility factor*. It is an indicator of the relative position of p with respect to the iterates in the Triangle Algorithm and points in S.

Definition 1 Given p, A, $p' \in conv(S)$, and a corresponding pivot v, we refer to $\angle pp'v$ as the *pivot angle*.

To describe the visibility factor, let $B_{\epsilon R}(p)$ be the open ball of radius ϵR at p, where $R = \max\{d(p, v_i), i = 1, \ldots, n\}$.

Definition 2 Given p, A, ϵ, for each iterate $p' \in conv(A) \setminus B_{\epsilon R}(p)$, let $v_{p'}$ denote the p-pivot at p' with the smallest pivot angle $\theta' = \angle pp'v_{p'}$. Let θ_p be the maximum of θ' over all the iterates in the Triangle Algorithm before the algorithm terminates.

Definition 3 Given input p, A, ϵ, *the visibility constant*, $\nu = \nu(p, A, \epsilon)$ of the Triangle Algorithm is $\sin\theta_p$. The *visibility factor* is the constant $c = c(p, A, \epsilon)$, satisfying

$$\nu = \frac{1}{\sqrt{1+c}}. \tag{7}$$

As will be shown later, the significance of ν lies in the fact that if we iterate the Triangle Algorithm k times getting no witnesses, the k-th iterate $p_k \in conv(S)$ satisfies

$$d(p_k, p) \leq \nu^k d(p_0, p). \tag{8}$$

This implies an alternate complexity bound on the number of arithmetic operations of the Triangle Algorithm (our second complexity bound):

$$O\left(mn\frac{1}{c}\ln\frac{1}{\epsilon}\right). \tag{9}$$

Thus combining (5) and (9) we can state the following complexity bound for the Triangle Algorithm as

$$O\left(mn\min\left\{\frac{1}{\epsilon^2}, \frac{1}{c}\ln\frac{1}{\epsilon}\right\}\right). \tag{10}$$

We will show that $c \geq \epsilon^2$. However, depending upon c the Triangle Algorithm could exhibit very efficient complexity bound, possibly even a polynomial-time complexity bound for solving the convex hull decision problem. Specifically, consider integer input data. If the size of encoding of the convex hull decision problem is L and $1/c$ is a polynomial in m, n and, L, then in polynomial-time complexity the Triangle Algorithm can compute $p' \in conv(S)$ such that $d(p', p) = d(Ax, p) \leq 2^{-L}$. The representation of such approximate solution p' can be rounded into an exact representation of p as a convex combination of v_i's.

As an example of a case where c can be a very good constant, we show that when the relative interior of $conv(A)$ contains the ball of radius ρ centered at p (see Rockafellar (1997) for definition and properties of relative interior), then $c \geq (\rho/R)^2$. This results in a third complexity bound. Thus when ρ/R is a constant that is independent of ϵ the Triangle Algorithm performs very well (See Fig. 12).

Remark 2 In case the Triangle Algorithm computes a witness $p' \in conv(S)$ (see (2)), we will show $p \notin conv(S)$ by explicitly describing a hyperplane that separates p from $conv(S)$. However, if $p \notin conv(S)$, for sufficiently small ϵ the Triangle Algorithm will necessarily compute a witness. Since we do not know the magnitude of such ϵ, one possible approach in this case is to first run the algorithm with a large value of ϵ, say, $\epsilon = 0.5$. If it finds a corresponding ϵ-approximate solution instead of a witness, we replace ϵ with $\epsilon/2$ and repeat this process. The overall complexity would be as if we would set $\epsilon = \Delta/R$, where $\Delta = d(p, conv(S))$, the distance between p and $conv(S)$. The complexity of computing a witness can be derived by substituting $\epsilon = \Delta/R$ in (10)

$$O\left(mn\min\left\{\frac{R^2}{\Delta^2}, \frac{1}{c}\ln\frac{R}{\Delta}\right\}\right). \tag{11}$$

When $p \notin conv(S)$, the Triangle Algorithm does not attempt to compute p_*, rather a separating hyperplane. However, by virtue of the fact that it finds a very special separating hyperplane, i.e. a hyperplane orthogonally bisecting the line pp', it in the process computes an approximation to $d(p, p_*) = \Delta$ to within a factor of two. More precisely, any witness p' satisfies the inequality

$$\frac{1}{2}d(p, p') \leq d(p, p_*) = \Delta \leq d(p, p'). \tag{12}$$

Remark 3 Not only this approximation is useful for problem (P), but for the case of computing the distance between two convex hulls, i.e. the polytope distance problem. It is well known

that the Minkowski difference of two convex hulls is a polytope whose shortest vector has norm equal to the distance between the two polytopes, see e.g. Clarkson (2008) and Gärtner and Jaggi (2009). In a forthcoming article, Kalantari [32], we will give a Triangle Algorithm that can compute the distance between two convex sets, in particular the case of two compact convex hulls.

2.3 The complexity of solving LP via the Triangle Algorithm

We consider the applicability of the Triangle Algorithm in order to solve the LP feasibility problem. This may be written as the problem of testing if the polyhedron

$$\Omega = \{x \in \mathbb{R}^n : \quad Ax = b, \quad x \geq 0\} \tag{13}$$

is nonempty, where $A = [a_1, \ldots, a_n]$ is an $m \times n$ real matrix. To apply the Triangle Algorithm, either we need to assume a known bound M on the one-norm of the vertices of Ω, i.e. a constraint $\sum_{i=1}^n x_i \leq M$, or it is known Ω has no *recession direction*, i.e. there does not exist a nontrivial $d \geq 0$ with $Ad = 0$. In case of a known bound M, by introducing a slack variable, x_{n+1}, in the new inequality constraint, as well as augmenting A by a zero column vector, the LP feasibility problem reduces to the convex hull decision problem of testing if b/M lies in $conv([A, 0]) = conv(\{a_1, \ldots, a_n, 0\})$. In theory, when no such a bound M is available but the input data are integers, one can argue an upper bound $2^{O(L)}$, where L is the size of encoding of A and b. Such bounds coming from LP-type analysis, initially stated in Khachiyan (1979) are well-known and discussed in many books. For an analysis, see Kalantari and Emamy-K (1997). In practice, one can start with a smaller bound M and gradually increase it while testing feasibility.

If it is known that Ω has no recession direction, then no such bound is necessary. In such case it is easy to prove $\Omega \neq \emptyset$ if and only if $0 \in conv([A, -b]) = conv(\{a_1, \ldots, a_n, -b\})$. In this case, given any $\epsilon \in (0, 1)$, the Triangle Algorithm gives an ϵ-approximate solution:

$$p' = \sum_{i=1}^n \alpha_i a_i - \alpha_{n+1} b \in conv\Big(\{a_1, \ldots, a_n, -b\}\Big), \tag{14}$$

where

$$d(p', 0) = \|p'\| < \epsilon R', \quad R' = \max\Big\{\|a_1\|, \ldots, \|a_n\|, \|b\|\Big\}. \tag{15}$$

Setting $x_0 = (\alpha_1, \ldots, \alpha_n)^T/\alpha_{n+1}$, $x_0 \geq 0$ and from (15) we have

$$d(Ax_0, b) < \frac{\epsilon R'}{\alpha_{n+1}}. \tag{16}$$

A practical and straightforward algorithm for solving LP feasibility via the Triangle Algorithm is to run the algorithm and in each iteration check if the bound in (16) is within a prescribe tolerance $\epsilon_0 \in (0, 1)$. However, to give a theoretical complexity bound on the number of iterations, we need a lower bound on α_{n+1}. We prove a sensitivity theorem (Theorem 20) that provides such a lower bound. We prove it suffices to have $\epsilon \leq \epsilon_0 \Delta_0/4R'$, where

$$\Delta_0 = \min\Big\{\|p\| : \quad p \in conv\Big(\{a_1, \ldots, a_n\}\Big)\Big\} = \min\Big\{\|Ax\| : \quad \sum_{i=1}^n x_i = 1, \quad x \geq 0\Big\}. \tag{17}$$

Indeed we offer a Two-Phase Triangle Algorithm that in Phase I computes a witness $p' \in conv(A)$ to estimate Δ_0. This follows from the inequality

$$\frac{1}{2}\|p'\| \leq \Delta_0 \leq \|p'\|. \tag{18}$$

It then proceeds to Phase II to compute an approximation to a point x_0 so that $d(Ax_0, b) < \epsilon_0 R'$.

Table 1 summarizes the complexity of solving three different LP feasibility problems and LP itself via the Triangle Algorithm. These will be proved in subsequent sections. The first block corresponds to the convex hull decision problem itself, where specifically A represents the $m \times n$ matrix of the points, and b represents p. Its columns are represented by a_i and $conv(A)$ represents the convex hull of its columns.

The second block correspond to solving $Ax = b, x \geq 0$ when $0 \notin conv(A)$. The third block corresponds to solving $Ax = b, x \geq 0$ when a bound is given on the sum of the variables. The forth block corresponds to solving an LP by converting it into a feasibility problem with a known bound. Specifically, the general LP problem, $\min\{c^T x : Ax \geq b, x \geq 0\}$ when combined with its dual, $\max\{b^T y : A^T y \leq c, y \geq 0\}$, can be formulated as an LP feasibility problem

$$Ax - x_a = b, \quad A^T y + y_a = c, \quad c^T x = b^T y, \quad x, x_a, y, y_a \geq 0. \tag{19}$$

This can be written as $\widehat{A}\widehat{x} = \widehat{b}, \widehat{x} \geq 0$ where $\widehat{A}$ is an $O(m+n)$ matrix. Thus, given a bound $\widehat{M}$ on the feasible solutions, the complexity to get a solution $\widehat{x} \geq 0$ such that $d(\widehat{A}\widehat{x}, \widehat{b}) < \epsilon \widehat{R}$ can be stated.

2.4 Triangle Algorithm versus simplex method to solve problem (P)

The convex hull decision problem is clearly a special LP feasibility, testing the feasibility of $Ax = b, x \geq 0, e^T x = 1$, where e is the n-vector of ones. Without loss of generality we may assume $b_i \geq 0$. We can formulate the convex hull decision problem as the following LP:

$$\min\left\{\sum_{i=1}^{m+1} z_i : Ax + z = b, \quad e^T x + z_{m+1} = 1, \quad x \geq 0, z_i \geq 0, i = 1, \ldots, n\right\}. \tag{20}$$

In the above formulation we have introduced an artificial variable z_i for each of the $m+1$ constraints. The feasible region of the LP is nonempty since we can set $z_i = b_i, i = 1, \ldots, m$ and $z_{m+1} = 1$.

By scaling the x component of each basic feasible solution of the LP that arises in the course of applying the simplex method, if $x \neq 0$, and $x' = x/e^T x \geq 0$, then $p' = Ax'$ lies in $conv(A)$. Thus the simplex method gives rise to a sequence of points in $conv(A)$ the last one of which either proves that $b \in conv(A)$, or if the objective value is nonzero, proves $b \notin conv(A)$. If $b \in conv(A)$ and we represent the sequence of distinct iterates of the simplex method by $p_1 = Ax^{(1)}, \ldots, p_{t-1} = Ax^{(t-1)}, p_t = Ax^{(t)} = b$, then the objective value is zero only at the last iteration and therefore no iterate of the simplex method can get closer to b than $\delta_s = \min\{d(b, p_i) : i = 1, \ldots, t-1\}$. However, the sequence of iterates in the Triangle Algorithm converge to b and thus after a certain number of iterations the gap will satisfy $\epsilon R \leq \delta_s$. The precise number of iteration thus can be determined by setting $\epsilon = \delta_s / R$.

The number of elementary operations at each iteration of the revised simplex method is $O(mn)$. According to Dantzig (1963) and Shamir's survey article Shamir (1987), the

Table 1 Complexity of solving via Triangle Algorithm

Application type	Quality of approximation	Time complexity of Triangle Algorithm	Description of constants
$Ax = b, x \geq 0, \sum x_i = 1$	Compute $x \geq 0$ such that $d(Ax, b) < \epsilon R$	$O\left(mn \min\{\frac{1}{\epsilon^2}, \frac{1}{c} \ln \frac{1}{\epsilon}\}\right)$	$R = \max\{d(b, a_i)\}$ $c = c(b, A, \epsilon) \geq \epsilon^2$
$Ax = b, x \geq 0\ 0 \notin conv(A)$	Compute $x \geq 0$ such that $d(Ax, b) < \epsilon R'$	$O\left(mn \min\{\frac{R'^2}{\epsilon^2 {\Delta_0}^2}, \frac{1}{c'} \ln \frac{R'}{\epsilon \Delta_0}\}\right)$	$R' = \max\{\|a_i\|, \|b\|\}\ \Delta_0 = d(0, conv(A))$ $c' = c(0, [A, -b], \frac{\epsilon \Delta_0}{4R'}) \geq \frac{\epsilon^2 {\Delta_0}^2}{4R'^2}$
$Ax = b, x \geq 0, \sum x_i \leq M$	Compute $x \geq 0$ such that $d(Ax, b) < \epsilon R'$	$O\left(mn \min\{\frac{M^2}{\epsilon^2}, \frac{1}{c''} \ln \frac{M}{\epsilon}\}\right)$	$R' = \max\{\|a_i\|, \|b\|\}\ c'' = c(\frac{1}{M} b, [A, 0], \epsilon) \geq \frac{\epsilon^2}{M^2}$
$\min c^T x$ s.t. $Ax = b, x \geq 0$, $\widehat{A}\widehat{x} = \widehat{b}, \widehat{x} \geq 0\ \sum \widehat{x}_i \leq \widehat{M}$	Compute $\widehat{x} \geq 0$ such that $d(\widehat{A}\widehat{x}, \widehat{b}) < \epsilon \widehat{R}$	$O\left((m + n)^2 \min\{\frac{\widehat{M}^2}{\epsilon^2}, \frac{1}{\widehat{c}} \ln \frac{\widehat{M}}{\epsilon}\}\right)$	$\widehat{R} = \max\{\|\widehat{a}_i\|, \|\widehat{b}\|\}$ $\widehat{c} = c(\frac{1}{\widehat{M}} \widehat{b}, [\widehat{A}, 0], \frac{\epsilon}{\widehat{M}}) \geq \frac{\epsilon^2}{\widehat{M}^2}$

(1) The convex hull decision problem (first block); (2) LP feasibility with no recession direction (second block); (3) LP feasibility with a known bound on feasible set (third block); (4) LP optimization converted into an LP feasibility. In all cases A is an $m \times n$ matrix

expected number of iterations of the simplex method to find a feasible solution to a linear program, say $Ax = b, x \geq 0$ where A is $m \times n$, is conjectured to be of the order αm, α is 2 or 3. Thus based on the above we would expect that the complexity of the revised simplex method to solve the convex hull decision problem to be $O(m^2 n)$. Ignoring the worst-case exponential complexity of the simplex method or possible cycling, and taking the average case complexity for granted when solving the convex hull decision problem, we may ask how does the simplex method compare with the Triangle Algorithm? In k iterations of the Triangle Algorithm, a time complexity of $O(mnk)$, we can computes an approximation p_k so that $d(p_k, p) \leq \nu^k d(p_0, p)$, ν the visibility constant (see (8)). Depending upon the value of ν we could obtain an extremely good approximation for k much less than m. In Li and Kalantari (2013) we have carried out some computational comparison between the Triangle Algorithm and the simplex method for solving the convex hull decision problem. The Triangle Algorithm has favorable performance. We hope to carry out more extensive comparisons.

2.5 Triangle Algorithm versus sparse greedy to solve problem (P)

Formally, the distance between p and $conv(S)$ is defined as

$$\Delta = d\Big(p, conv(S)\Big) = \min\Big\{d(p', p): \quad p' \in conv(S)\} = d(p_*, p)\Big\}. \tag{21}$$

We have, $p \notin conv(S)$, if and only if $\Delta > 0$. While solving problem (P) does not require the computation of Δ when it is positive, in some applications this distance is required. However, as stated in (12), any witness approximates Δ to within a factor of two. This fact indicates another useful property of the Triangle Algorithm.

One of the best known algorithms for determining the distance between two convex polyposes is Gilbert's algorithm, Gilbert (1966). The connections and equivalence of Gilbert's algorithm and Frank–Wolfe algorithm, a gradient descent algorithm, when applied to the minimization of a convex quadratic over a simplex is formally studied in Gärtner and Jaggi (2009). They make use of a notion called *coreset*, previously studied in Clarkson (2008), and define a notion of ϵ-approximation which is different from our notion given in (1). Furthermore, from the description of Gilbert's algorithm in Gärtner and Jaggi (2009) it does not follow that Gilbert's algorithm and the Triangle Algorithm are identical. However, there are similarities in theoretical performance of the two algorithms and we will discuss these next. Indeed we believe that the simplicity of the Triangle Algorithm and the distance duality theorems that inspires the algorithm, as well as the its theoretical performance makes it distinct from other algorithms for the convex hull decision problem. Furthermore, these features of the Triangle Algorithm may encourage and inspire new applications of the algorithm and further theoretical analysis, in particular amortized complexity of the Triangle Algorithm. In upcoming reports we shall present some such results.

The convex hull decision problem, and its optimization form in (21) can equivalently be formulated as the minimization of a convex quadratic function over a simplex:

$$\min\left\{f(x) = d\Big(\sum_{i=1}^{n} x_i v_i, p\Big)^2: \quad x \in \Sigma_n\right\}, \quad \Sigma_n = \left\{x \in \mathbb{R}^n: \quad \sum_{i=1}^{n} x_i = 1, x \geq 0\right\}. \tag{22}$$

Let x_* be an optimal solution of (22). Suppose $f(x_*) = 0$. Then an approximation algorithm would attempt to compute a point x' in the simplex so that $f(x')$ is small. Let us examine the performance of Triangle Algorithm. As before, let

$$R = \max\{d(p, v_i) : v_i \in S\}. \tag{23}$$

Given $\epsilon \in (0, 1)$, letting $p' = Ax', x' \in \Sigma_n$, then $d(p', p) < \epsilon R$ if and only if $f(x') < \epsilon^2 R^2$. Thus using Table 1, given $\epsilon \in (0, 1)$, the complexity of computing x' so that

$$f(x') < \epsilon R^2 \tag{24}$$

can be computed by substituting for ϵ, $\sqrt{\epsilon}$ in the first complexity block to give

$$O\left(mn \min\left\{\frac{1}{\epsilon}, \frac{1}{c}\ln\frac{1}{\epsilon}\right\}\right). \tag{25}$$

Furthermore, if $f(x_*) > 0$, and $p' = Ax'$ is a witness, then $0.5\Delta \le d(p'p) \le \Delta$. Since $f(x_*) = \Delta^2$, we get

$$\frac{1}{4} f(x_*) \le f(x') \le f(x_*). \tag{26}$$

Then by substituting for ϵ in the first block of Table 1 the quantity Δ/R, the complexity of computing the above approximation is

$$O\left(mn \min\left\{\frac{R^2}{f(x_*)}, \frac{1}{c}\ln\frac{R^2}{f(x_*)}\right\}\right). \tag{27}$$

We now contrast this complexity with the *greedy algorithm* for the optimization of $f(x)$ (as well as more general smooth convex function $f(x)$), as described in Clarkson (2008). It can equivalently be described as the following concave maximization:

Greedy Algorithm

- **Step I.** Given $x' \in \Sigma_n$, let j be the index satisfying $\frac{\partial f(x')}{\partial x_j} = \min\{\frac{\partial f(x')}{\partial x_i}, i = 1, \ldots, n\}$.
- **Step II.** Compute $x'' = \operatorname{argmin}\{f(x' + \alpha(e_j - x')) : \quad \alpha \in [0, 1]\}$, where e_j is the j-th vector of the standard basis. Replace x' with x'', go to Step I.

Step 1 of the Triangle Algorithm and Step I of the Greedy Algorithm (also known as sparse greedy approximation) have in common the fact that they select an index j so that v_j will be used as a p-pivot. Having computed such a p-pivot for p', Step 2 of the Triangle Algorithm and Step II of the Greedy Algorithm simply perform a line search. However, the motivation behind the selection of the index j is very different. The Greedy Algorithm coincides with Frank–Wolfe algorithm and Gilbert's algorithm. The Greedy Algorithm is algebraically motivated (using gradients), while the Triangle Algorithm is geometrically motivated. The Triangle Algorithm does not need to search over all the indices to find a p-pivot v_j. In its best case it finds such j in one iteration over the indices by performing only $O(m)$ arithmetic operations. It then needs to update the representation of the current iterate, taking $O(n)$ operations. Thus one iteration of the Triangle Algorithm could cost $O(m + n)$ operations ($O(m)$ to find a pivot, plus $O(n)$ to update the coefficients α'_i of p''). In the worst-case an iteration will require $O(mn)$ operations. The Greedy Algorithm in contrast requires $O(mn)$ arithmetic operations in every iteration. This can be seen when $f(x)$ is written as $d(Ax, p)^2$. Its gradient at a point would in particular require the computation of $A^T Ax$.

The Greedy Algorithm generates a sequence of vectors $x_{(k)} \in \mathbb{R}^n$ where $x_{(k+1)}$ has at most k nonzero coordinates. This is advantageous when n is very large. As will be easily verifiable this property also holds for the Triangle Algorithm when the initial iterate p_0 is

taken to be sparse. Another property of the Greedy Algorithm is that if x_* is the optimal solution of (22), then

$$f(x_{(k)}) - f(x_*) \leq \frac{C_f}{k} = O\left(\frac{1}{k}\right), \tag{28}$$

where C_f is a constant that depends on the Hessian of f, see Clarkson (2008) and Zhang (2003).

The Triangle Algorithm generates a sequence of points $p'_{(k)}$ in $conv(S)$ that get closer and closer to p. This sequence corresponds to a sequence $x'_{(k)}$ in Σ_n, where $f(x'_{(k)}) = d(p'_{(k)}, p)^2$. If $p \in conv(S)$, $f(x_*) = 0$. Given an $\epsilon \in (0, 1)$, according to the first complexity bound, the Triangle Algorithm in $K_\epsilon \leq 48\epsilon^{-2}$ iterations will generate a point $p_\epsilon \in conv(S)$ satisfying

$$d(p'_{(K_\epsilon)}, p) < \epsilon R, \quad R = \max\left\{d(p, v_i), i = 1, \ldots, n\right\}. \tag{29}$$

Given an index k, by reversing the role of k and ϵ and solving for ϵ in the equation $48/\epsilon^2 = k$, we get $\epsilon = \sqrt{48/k}$ so that we may write

$$d(p'_{(k)}, p) < \sqrt{\frac{48}{k}} R. \tag{30}$$

Equivalently,

$$f(x'_{(k)}) < \frac{48R^2}{k} = O\left(\frac{1}{k}\right). \tag{31}$$

In summary, when $p \in conv(S)$ the Triangle Algorithm according one complexity analysis works similar to the Greedy Algorithm, however it may perform better because it only needs to find a pivot v_j, as opposed to finding the minimum of partial derivatives in Step I of the Greedy Algorithm. Then when it uses the best pivot strategy, according to the second complexity, it could make much more effective steps than Greedy Algorithm. When p is not in $conv(S)$, the Triangle Algorithm can use any witness to give an approximation of closest point to within a factor of two, see (12). We may conclude that the Triangle Algorithm in theory is at least as effective as the Greedy Algorithm, and possibly faster whether approximating $p \in conv(S)$, or estimating the distance Δ to within a factor of two.

In the context support vector machines (SVM) (see Burges (1998) for applications), Har-Peled et al. (2007) use coreset to give an approximation algorithm, see also Zimak (2006). We mention these because we feel that the Triangle Algorithm, despite some similarities with existing algorithms or their analysis, is distinct from them. It is quite simple and geometrically inspired by the distance duality, a simple but new and rather surprising property.

In fact the complexity of the Triangle Algorithm could be much more favorable in contrast with these algorithms. In Li and Kalantari (2013) some computational comparison between the Triangle Algorithm and the Frank–Wolfe algorithm for solving the convex hull decision problem demonstrates that the Triangle Algorithm has favorable performance.

As mentioned earlier the Triangle Algorithm is not designed to approximate $f(x_*)$ to prescribed tolerance when $f(x_*) > 0$. However, in a forthcoming article, Kalantari (2014), we describe a generalization of the Triangle Algorithm that in particular approximates the minimum value $f(x_*)$ to prescribed tolerance, almost with similar complexity.

2.6 Triangle Algorithm versus first-order methods to solve problem (P)

Consider the problem

$$\min\left\{g(x) = d(Ax, b) = \sqrt{f(x)}, x \in \Sigma_n\right\},$$
$$\Sigma_n = \left\{x \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1, x_i \geq 0, i = 1, \ldots, n\right\}. \quad (32)$$

When $p \in conv(S)$, to compute x_ϵ such that $g(x_\epsilon) < \epsilon$ can be solved in $O(\sqrt{\ln n} \max \|a_i\|_2/\epsilon)$ iterations of the fast gradient scheme of Nesterov (2005), so-called $O(1/\epsilon)$-method, as applied to a smoothed version of $g(x)$, where each iteration takes $O(mn)$ operations, Nesterov (2013). A cone programming version of such method is described in Lan et al. (2011).

Let us compare this complexity with the Triangle Algorithm. In one analysis of the complexity, the Triangle Algorithm computes an approximate solution $x_\epsilon \in \Sigma_n$ where $g(x_\epsilon) < \epsilon R$ in $O(1/\epsilon^2)$ iterations. Each iteration takes mn operations in the worst case when any pivot is used, but an iteration could only take $O(m + n)$ arithmetic operations. Changing ϵ in Nesterov's method to ϵR, we conclude Nesterov's method computes x'_ϵ such that $g(x'_\epsilon) < \epsilon R$ in $O(\sqrt{\ln n}/\epsilon)$ iterations as opposed to $1/\epsilon^2$. However, as we shall show when $p \in conv(S)$ and its distance to a boundary point of S is at least $\epsilon^{1/t} R$, t a natural number, the number of iterations of the Triangle Algorithm is $O(\sqrt[t]{\epsilon^{-2}} \ln \delta_0/\epsilon R)$, $\delta_0 = d(p_0, p)$, initial gap. This already makes it comparative or better than the first-order methods for testing if $p \in conv(S)$.

2.7 Problem (P) and linear programming algorithms

Linear programming has found numerous practical and theoretical applications in applied mathematics, computer science, operation research and more. In particular, the simplex method of Dantzig is not only a significant algorithm for solving LP but also a theoretical tool to prove many results. Ever since the Klee and Minty (1972) example showed exponential worst-case time complexity of the simplex method, many LP algorithms have been invented. The trend will most likely continue.

Problem (P) is a very special case of the LP feasibility problem. However, in fact the general LP with integer inputs can be formulated as a homogeneous case of problem (P), i.e. $p = 0$. The corresponding problem (P), may be referred as *homogeneous feasibility problem* (HFP), see Kalantari (1990). A classical duality corresponding to HFP is Gordan's theorem, a special case of the separating hyperplane theorem, easily provable from Farkas lemma: either $0 \in conv(S)$, or there exists $y \in \mathbb{R}^m$ such that $y^T v_i > 0$ for all $i = 1, \ldots, n$, see e.g. Chvátal (1983). It can be justified that both Khachiyan and Karmarkar algorithms explicitly or implicitly are designed to solve HFP. This is because on the one hand Karmarkar's canonical formulation of LP can easily be converted into an HFP. On the other hand, Khachiyan's ellipsoid algorithm solves a system of strict inequalities ($Ax < b$) whose alternative system, by Gordan's theorem is an HFP ($A^T y = 0, b^T y + s = 0, \sum y_i + s = 1, y \geq 0, s \geq 0$)). For this and additional results on the connections between HFP and LP feasibility, see Jin and Kalantari (2006).

By exploring the close relationship between HFP, equivalent to the problem of computing a nontrivial nonnegative zero of a quadratic form, and the diagonal matrix scaling problem, Khachiyan and Kalantari (1992) have given a very simple path-following polynomial-time algorithm for LP as well as for quasi doubly stochastic diagonal scaling of a positive semidef-

inite matrix. In particular, the algorithm can test the existence of an ϵ-approximate solution of HFP in a number of arithmetic operations proportional to $n^{3.5}$ and $\ln \epsilon^{-1}$. As approximation schemes, all known polynomial-time algorithms for LP have a complexity that is polynomial in the dimension of the data and in $\ln \epsilon^{-1}$. As is well known, an exact solution for an LP with integral input can be computed by rounding any approximate solution having sufficient precision. Even if the complexity of a polynomial-time algorithms for LP would allow solving problem (P) to within ϵ accuracy in $O(m^2 n \ln \epsilon^{-1})$ arithmetic operations, the Triangle Algorithm still offers an attractive alternative when the dimensions of the problems are large, or when the visibility constant is good.

Other algorithms for LP include, Megiddo's algorithm Megiddo (1984) with a running time that for fixed m is linear in n, however, has exponential complexity in m. Since Mediggo's work a number of randomized LP algorithms have been devised, e.g. Dyer and Frieze (1989), Clarkson (1988), Seidel (1991), Sharir and Welzl (1992). Kalai (1992) gave a randomized LP simplex method with subexponential complexity bound. Matoušek et al. (1992) proved another randomized subexponential complexity algorithm for LP. See also Motawani and Raghavan (1995). Kelner and Spielman (2006) have given the first randomized polynomial-time simplex method that analogous to the other known polynomial-time algorithms for linear programming has a running time dependent polynomially on the bit-length of the input. A history of linear programming algorithms from computational, geometric, and complexity points of view that includes simplex, ellipsoid, and interior-point methods is given in Todd (2002).

2.8 Outline of results

The remaining sections of the article are as follows. In Sect. 3, we prove several characterization theorems leading into the *distance duality* theorem. We then describe several associated geometric properties and problems, as well as generalizations of the distance duality. In Sect. 4, while using purely geometric arguments, we give an analysis of the worst-case reduction of the gap in moving from one approximation in the convex hull of points to the next. In Sect. 5, we formally describe the steps of the Triangle Algorithm. We then use the results in Sects. 3 and 4 to derive a bound on the worst-case complexity of the Triangle Algorithm. In Sect. 6, we prove a *strict distance duality* and define corresponding strict pivots and strict witness. We also prove a minimax theorem related to the strict distance duality. In Sect. 7, we derive an alternate complexity bound on the Triangle Algorithm in terms of constants defined as *visibility constant* and *visibility factor*. We analyze this alternate complexity in terms of these constants and their relations to the location of p relative to S. These suggest that the Triangle Algorithm could result in a very efficient algorithm. In Sect. 8, we describe *Virtual Triangle Algorithm* and its approximated version. These serve are fast algorithms that if successful can efficiently lead to a proof of infeasibility, or to the computation of a good approximate solution. In Sect. 9, we describe *auxiliary pivots*, points that can be added to S so as to improve visibility constants, hence the computational efficiency of the Triangle Algorithm. In Sect. 10, we describe *Generalized Triangle Algorithm* based on a *generalized distance duality*. Each iteration of the Generalized Triangle Algorithm computes a more effective approximation, however at the cost of more computation. In Sect. 11, we consider solving the LP feasibility problem having no recession direction. We prove a *sensitivity theorem* that gives the necessary accuracy in solving a corresponding convex hull decision problem. Using the sensitivity theorem we derive complexity bound for computing an approximate feasible point, via the straightforward application of the Triangle Algorithm, or a *Two-Phase* Triangle Algorithm. In Sect. 12, we derive the complexity for solving the

general LP feasibility via the Triangle Algorithm making no assumption on the recession direction, however with a known bound on the one-norm of the vertices. In particular, this gives a complexity bound for solving a general LP optimization via the Triangle Algorithm. Finally, we conclude the article with some remarks on applications and extensions of the results.

3 Characterizations and applications

Throughout the section, let $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$, and p a distinguished point in $\mathbb{R}^m$.

Theorem 1 (Characterization of Feasibility) *$p \in conv(S)$ if and only if given any $p' \in conv(S) \setminus \{p\}$, there exists $v_j \in S$ such that $d(p', v_j) > d(p, v_j)$.*

Proof Suppose $p \in conv(S)$. Consider the Voronoi cell of p with respect to the two point set $\{p, p'\}$, i.e. $V(p) = \{x \in \mathbb{R}^m : \ d(x, p) < d(x, p')\}$ (see Fig. 1). We claim there exists $v_j \in V(p)$. If not, S is a subset of $\overline{V}(p') = \{x \in \mathbb{R}^m : \ d(x, p') \leq d(x, p)\}$, the closure of the Voronoi cell of p'. But since $\overline{V}(p')$ is convex it contains $conv(S)$. But since $V(p) \cap \overline{V}(p') = \emptyset$ this contradicts that $p \in conv(S)$.

Conversely, suppose that for any $p' \in conv(S) \setminus \{p\}$ there exists v_j such that $d(p', v_j) > d(p, v_j)$. If $p \notin conv(S)$, let $p' \in conv(S)$ be the closest point to p. Since the closest point is unique, for each $i = 1, \ldots, n$, in the triangle $\triangle pp'v_i$ the angle $\angle pp'v_i$ must be non-acute. Hence, $d(p', v_i) < d(p, v_i)$ for all i, a contradiction. Thus, $p \in conv(S)$. □

Remark 4 We may view Theorem 1 as a characterization theorem for feasibility or infeasibility of a point with respect to the convex hull of a finite set of points. The present proof is a simpler version of a proof given in the first version of the present article, Kalantari (2012a). It was brought to our attention that a proof by Kuhn (2011) was given for points in the Euclidean plane. Kuhn's proof makes use of several results, including Ville's Lemma. Kuhn's proof makes no connections to Voronoi diagram which is important in the development of the Triangle Algorithm and its analysis. Some generalizations of the theorem over normed spaces is given by Durier and Michelot (1986). In this article we offer other generalizations and use them in variations of the Triangle Algorithm. We refer to Theorem 1 as *distance duality* because it is based on comparisons of distances. It can be viewed as a stronger version of the classical Gordan's Theorem on separation of zero from the convex hull of finite set of points. Gordan's Theorem and its conic version, Farkas Lemma, are theorems of the alternative and closely related to duality theory in linear programming. In a forthcoming article, Kalantari [32], we prove a substantially general version of Theorem 1 that gives rise to an algorithm for approximation of the distance between two compact convex subsets of the Euclidean, and a separating hyperplane if they ate disjoint.

The following is a convenient restatement of Theorem 1, relaxing the strict inequality, $d(p', v_j) > d(p, v_j)$. It has an identical proof to that theorem.

Theorem 2 *$p \in conv(S)$ if and only if given any $p' \in conv(S) \setminus \{p\}$, there exists $v_j \in S$ such that $d(p', v_j) \geq d(p, v_j)$.* □

Definition 4 We call a point $p' \in conv(S)$ a *p-witness* (or simply a *witness*) if $d(p', v_i) < d(p, v_i)$, for all $i = 1, \ldots, n$. We denote the set of all p-witnesses by W_p.

The following is a characterization of infeasibility.

Theorem 3 (Characterization of Infeasibility) *$p \notin conv(S)$ if and only if there exists a p-witness p'.*

Proof Suppose $p \notin conv(S)$. Then by Theorem 2 there exists $p' \in conv(S)$ such that $d(p', v_i) < d(p, v_i)$, for all $i = 1, \ldots, n$. But then p' is a p-witness. Conversely, given that p' is a p-witness, Theorem 1 implies $p \notin conv(S)$. □

Thus we may conclude the following, a non-traditional duality for the convex hull decision problem.

Theorem 4 (Distance Duality) *Precisely one of the two conditions is satisfied:*

(i) For each $p' \in conv(S)$ there exists $v_j \in S$ such that $d(p', v_j) \geq d(p, v_j)$.
(ii) There exists a p-witness. □

The following is a straightforward but geometrically appealing characterization of the set of p-witnesses as the intersection of open balls and $conv(S)$.

Proposition 1 *Let $B_i = \{x \in \mathbb{R}^m : \ d(x, v_i) < d(p, v_i)\}$, $i = 1, \ldots, n$. Then $W_p = conv(S) \cap (\cap_{i=1}^n B_i)$. In particular, W_p is a convex subset of $conv(S)$.* □

Figure 2 gives a case when W_p is empty. Figure 3 gives several scenarios when W_p is nonempty. Next, we state a corollary of Theorem 1 and Theorem 2.

Corollary 1 (Intersecting Balls Property) *Consider the set of open balls $B_i = \{x \in \mathbb{R}^m : d(x, v_i) < r_i\}$. Assume they have a common boundary point p, i.e. $p \in \cap_{i=1}^n \partial B_i = \{x \in \mathbb{R}^m : d(x, v_i) = r_i\}$. Let $S = \{v_1, \ldots, v_n\}$. Then $p \in conv(S)$ if and only if $(\cap_{i=1}^n B_i) \cap conv(S) = \emptyset$* (see Fig. 2).

Proof Suppose $p \in conv(S)$. Pick any point $p' \in conv(S) \setminus \{p\}$. Then by Theorem 1 there exists v_j such that $d(p', v_j) > d(p, v_j)$. But this implies $p' \notin B_j$, hence $(\cap_{i=1}^n B_i) \cap conv(S) = \emptyset$.

Conversely, suppose that $(\cap_{i=1}^n B_i) \cap conv(S) = \emptyset$. Thus for each $p' \in conv(S)$ there exists v_j such that $d(p', v_j) \geq d(p, v_j)$. Then by Theorem 2 we have $p \in conv(S)$. □

Proposition 2 (The Orthogonal Bisector Property) *Suppose p' is a p-witness, i.e. $p' \in conv(S)$ satisfies $d(p', v_i) < d(p, v_i)$, for all $i = 1, \ldots, n$. Then, the orthogonal bisector hyperplane of the line segment pp' separates p from $conv(S)$. More specifically, let $c = p - p'$ and $\gamma = \frac{1}{2}(\|p\|^2 - \|p'\|^2)$. If $H = \{x \in \mathbb{R}^m : \ c^T x = \gamma\}$, then $p \in H_+ = \{x \in \mathbb{R}^m : c^T x > \gamma\}$ and $conv(S) \subset H_- = \{x \in \mathbb{R}^m : \ c^T x < \gamma\}$.*

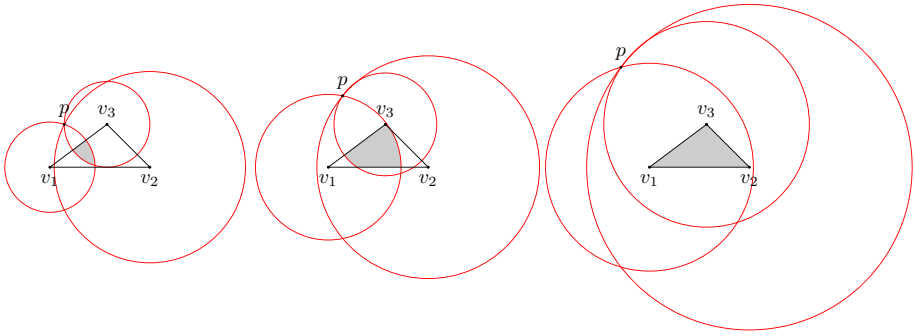

Fig. 3 Examples of nonempty p-witness set W_p, *gray areas*: $p \notin conv(S)$

Proof It is easy to verify that $p \in \overline{H}_+$. We claim $v_i \in H_-$, for each i. For each $i = 1, \ldots, n$ we have, $d^2(p', v_i) < d^2(p, v_i)$. Equivalently, $(p' - v_i)^T(p' - v_i) < (p - v_i)^T(p - v_i)$. Simplifying, gives

$$2(p - p')^T v_i < (\|p\|^2 - \|p'\|^2). \tag{33}$$

Hence $S \subset H_-$. Since H_- is convex, any convex combination of points in S is also in H_-. □

We now give a complete characterization of the p-witness set.

Theorem 5 (Characterization of p-Witness Set) *$p' \in W_p$ if and only if the orthogonal bisector hyperplane of the line segment pp' separates p from $conv(S)$.*

Proof By Proposition 2, $p' \in W_p$ implies the orthogonal bisector hyperplane of the line segment pp' separates p from $conv(S)$. Conversely, suppose for some $p' \in conv(S)$ the orthogonal bisector hyperplane of the line segment connecting pp' separates p from $conv(S)$. Then, in particular we have $d(p', v_i) < d(p, v_i)$, for all $i = 1, \ldots, n$. Hence, $p' \in W_p$. □

Corollary 2 *Suppose $p \notin conv(S)$. Let*

$$\Delta = d(p, conv(S)) = \min\{d(p, x) : \quad x \in conv(S)\}. \tag{34}$$

Then any $p' \in W_p$ gives an estimate of Δ to within a factor of two. More precisely,

$$\frac{1}{2}d(p, p') \leq \Delta \leq d(p, p'). \tag{35}$$

Proof The inequality $\Delta \leq d(p, p')$ is obvious. The first inequality follows from Theorem 5. □

Next we give another characterization of a witness in terms of forbidden zones.

Definition 5 The *forbidden zone* $F(X, q)$ for a region $X \subseteq \mathbb{R}^m$ and a point $q \in X$ is the set of all points that are closer to some point $y \in X$ than y is to q, i.e.

$$F(X, q) = \left\{ z : d(z, y) < d(y, q) \text{ for some } y \in X \right\}. \tag{36}$$

The following is simple to prove.

Theorem 6 (de Biasi et al. 2011) *For a polytope P with vertices w_i, $i = 1, \ldots, n$, the forbidden zone $F(P, q)$ is the union of the open balls centered at each w_i with radius $d(q, v_i)$. That is:*

$$F(P, q) = \bigcup_{i=1}^{n} \left\{ z : d(z, w_i) < d(z, q) \right\}. \tag{37}$$

□

We may thus conclude:

Corollary 3 *Given $S = \{v_1, \ldots, v_n\}$ and p, $p' \in W_p$ if and only if $F(conv(S), p')$ does not contain p.*

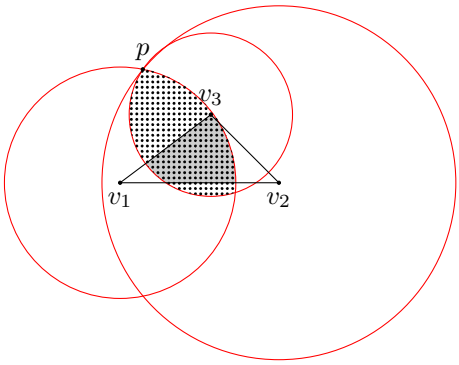

Fig. 4 The set $\overline{W}_p$ of general p-witness (*dotted area*) is a superset of W_p (*gray area*)

Visually speaking, this says when $p \notin conv(S)$ and p' is a witness, the union of the open balls centered at v_i, having radius $d(p', v_i)$, a region that contains $conv(S)$, excludes p.

Before we utilize the characterization theorems proved here we wish to give a variation of Theorem 1. The theorem shows that the notion of a p-witness need not be restricted to the convex hull of S. The proof is identical to the proof of Theorem 1 and is omitted.

Theorem 7 *$p \in conv(S)$ if and only if for any point $p' \neq p$, there exists $v_j \in S$ such that $d(p', v_j) > d(p, v_j)$.* □

We may thus give a more general distance duality as well as definition for a p-witness.

Definition 6 We call a point $p' \in \mathbb{R}^m$ a *general p-witness* if $d(p', v_i) < d(p, v_i)$, for all $i = 1, \ldots, n$. We denote the set of all general p-witnesses by $\overline{W}_p$.

Figure 4 depicts the set $\overline{W}_p$ which of course contains W_p as a subset. The following is a corollary of Theorem 2 and Theorem 7.

Corollary 4 (General Intersecting Balls Property) *Consider the set of open balls $B_i = \{x \in \mathbb{R}^m : \ d(x, v_i) < r_i\}$. Assume $p \in \cap_{i=1}^n \partial B_i$. Then $p \in conv(\{v_1, \ldots, v_n\})$ if and only if $(\cap_{i=1}^n B_i) = \emptyset$.*

Proof Suppose $p \in conv(S)$. Pick any point $p' \neq p$. Then by Theorem 7 there exists v_j such that $d(p', v_j) > d(p, v_j)$. But this implies $p' \notin B_i$, hence $(\cap_{i=1}^n B_i) = \emptyset$.

Conversely, suppose that $(\cap_{i=1}^n B_i) = \emptyset$. In particular, for each $p' \in conv(S)$ there exists v_j such that $d(p', v_j) \geq d(p, v_j)$. Then by Theorem 2 we have $p \in conv(S)$. □

Remark 5 The general open balls property suggests that in proving the infeasibility of p we have the freedom of choosing a p-witnesses outside of the convex hulls of S. However, algorithmically it may have no advantage over the Triangle Algorithm to be formally described later.

4 Reduction of gap and its worst-case analysis

In what follows we state a theorem that is fundamental in the analysis of the algorithm to be described in the next section. It relates to one iteration of the algorithm to test if p lies in $conv(S)$ and reveals its worst-case performance. Before formally analyzing the worst-case of the Triangle Algorithm we give a definition.

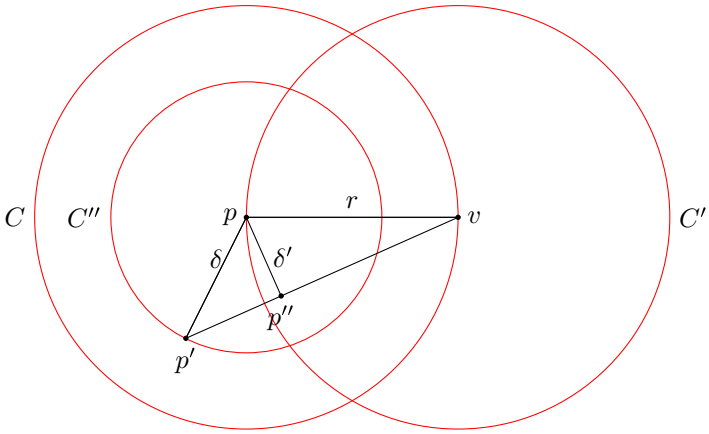

Fig. 5 Depiction of gaps $\delta = d(p', p)$, $\delta' = d(p'', p)$, when $\delta \leq r = d(p, v)$

Definition 7 Given $p' \in conv(S)$, we say $v_j \in S$ is a *pivot* relative to p (or p-pivot) if $d(p', v_j) \geq d(p, v_j)$.

In the theorem below the reader may consider p' as a given point in $conv(S)$ and v as a point in S to be used as a *p-pivot* in order to compute a new point p'' in $conv(S)$ where the new gap $d(p'', p)$ is to be a reduction of the current gap $d(p', p)$.

Theorem 8 *Let p, p', v be distinct points in $\mathbb{R}^m$. Suppose $d(p, v) \leq d(p', v)$. Let p'' be the point on the line segment $p'v$ that is closest to p. Assume $p'' \neq v$. Let $\delta = d(p', p)$, $\delta' = d(p'', p)$, and $r = d(p, v)$* (see Fig. 5)*. Then,*

$$\delta' \leq \begin{cases} \delta\sqrt{1 - \frac{\delta^2}{4r^2}}, & \text{if } \delta \leq r; \\ r, & \text{otherwise.} \end{cases} \tag{38}$$

Proof Given $\delta \leq r$, consider p' as a variable x' and the corresponding p'' as x''. We will consider the maximum value of $d(x'', p)$ subject to the desired constraints. We will prove

$$\delta^* = \max \left\{ d(x'', p) : \quad x \in \mathbb{R}^m, \quad d(x', p) = \delta, \quad d(x', v) \geq r \right\} = \delta\sqrt{1 - \frac{\delta^2}{4r^2}}. \tag{39}$$

This optimization problem can be stated in the two-dimensional Euclidean plane. Assume $p \neq p''$, and consider the two-dimensional plane that passes through the points p, p', v. Given that $\delta \leq r$, p' must lie inside or on the boundary of the circle of radius δ centered at p, but outside or on the boundary of the circle of radius r centered at v, see Fig. 5, circles C, C', and C''.

Now consider the circle of radius δ centered at p, C'' in Fig. 5. Consider the ratio δ'/r as p' ranges over all the points on the circumference of C'' while outside or on the boundary of C'. It is geometrically obvious and easy to argue that this ratio is maximized when p' is a point of intersection of the circles C' and C'', denoted by x^* in Fig. 6. We now compute the corresponding ratio.

Considering Fig. 6, and the isosceles triangle $\triangle vpx^*$, let h denote the length of the bisector line from v to the base, and let q denote the midpoint of p and $x*$. Consider the right triangles

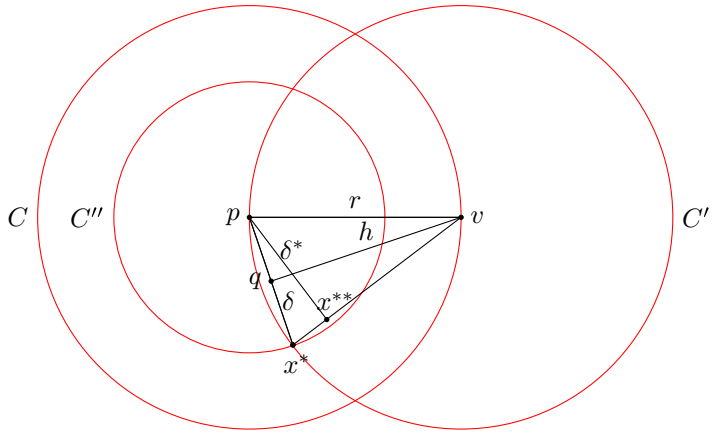

Fig. 6 The worst-case scenario for the gap $\delta' = \delta^*$, when $\delta \leq r$

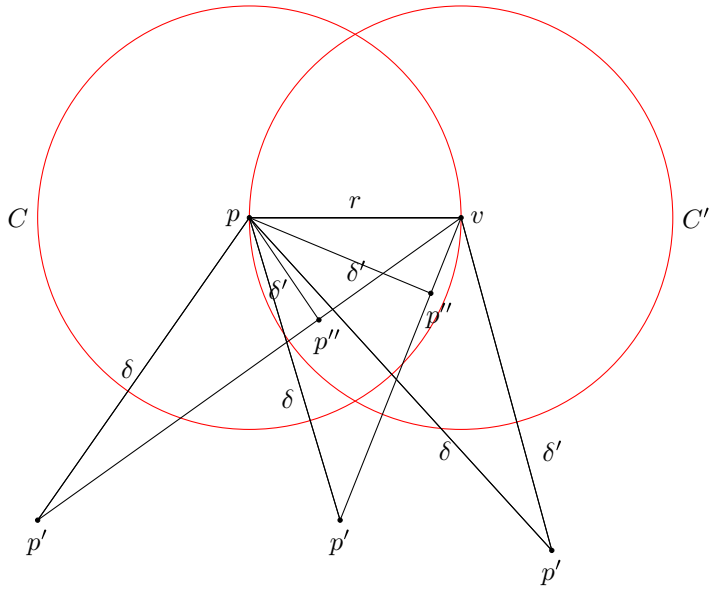

Fig. 7 Depiction of gaps $\delta = d(p', p)$, $\delta' = d(p'', p)$, when $\delta > r = d(p, v)$

$\triangle pvq$ and $\triangle px^*x^{**}$. The angle $\angle vpq$ is identical with $\angle px^*x^{**}$. Hence, the two triangles are similar and we may write

$$\frac{\delta^*}{\delta} = \frac{h}{r} = \frac{1}{r}\sqrt{r^2 - \frac{\delta^2}{4}} = \sqrt{1 - \frac{\delta^2}{4r^2}}. \tag{40}$$

This proves the first inequality in (38).

Next, suppose $\delta > r$. Figure 7 considers several possible scenarios for this case. If in the triangle $\triangle pvp'$ the angle $\angle pvp'$ is acute, the line segment $p'v$ must necessarily intersect C'. This implies p'' is the bisector of a chord in C', hence inside of C'. If the angle $\angle pvp'$ is at least $\pi/2$, then p'' will coincide with v. Hence, proving the second inequality in (38). □

5 The Triangle Algorithm and its analysis

In this section we describe a simple algorithm for solving problem (P). For convenience we shall refer to this algorithm as the *Triangle Algorithm*. The justification in the name lies in the fact that in each iteration the algorithm searches for a triangle $\triangle pp'v_j$ where $v_j \in S$, $p' \in conv(S) \setminus \{p\}$, such that $d(p', v_j) \geq d(p, v_j)$. Given that such triangle exists, it uses v_j as a p-pivot to "pull" the current iterate p' closer to p to get a new iterate $p'' \in conv(S)$. If no such a triangle exists, then by Theorem 3, p' is a p-witness certifying that p is not in $conv(S)$. The steps of the algorithm are described in the box. Note that given the coordinates of p' and α_i's that give its representation as a convex combination of the v_i's, it takes $O(m)$ operations to compute p'' and $O(n)$ operations to compute the α_i''s.

Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, $p, \epsilon \in (0, 1)$)

- **Step 0.** (**Initialization**) Let $p' = v = \text{argmin}\{d(p, v_i) : v_i \in S\}$.
- **Step 1.** If $d(p, p') < \epsilon d(p, v)$, output p' as ϵ-approximate solution, stop. Otherwise, if there exists a pivot v_j, replace v with v_j. If no pivot exists, then output p' as a witness, stop.
- **Step 2.** Compute the *step-size*

$$\alpha = \frac{(p - p')^T (v_j - p')}{d^2(v_j, p')}. \tag{41}$$

Given the current iterate $p' = \sum_{i=1}^{n} \alpha_i v_i$, set the new *iterate* as:

$$p'' = (1 - \alpha)p' + \alpha v_j = \sum_{i=1}^{n} \alpha_i' v_i, \quad \alpha_j' = (1-\alpha)\alpha_j + \alpha, \quad \alpha_i = (1-\alpha)\alpha_i, \quad \forall i \neq j. \tag{42}$$

Replace p' with p'', α_i with α_i', for all $i = 1, \ldots, n$. Go to Step 1.

By an easy calculation that shift p' to the origin, it follows that the point p'' in Step 2 is the closest point to p on the line $p'v_j$. Since p'' is a convex combination of p' and v_j it will remain in $conv(S)$. The algorithm replaces p' with p'' and repeats the above iterative step. Note that a p-pivot v_j may or may not be a vertex of $conv(S)$. In the following we state some basic properties of the algorithm to be used in the analysis of its complexity.

We are now ready to analyze the complexity of the algorithm. Set

$$R = \max \left\{ d(p, v_i), i = 1, \ldots, n \right\}. \tag{43}$$

Corollary 5 *Let p, p', p'', v be as in Theorem 8, and $r = d(p, v)$, $\delta = d(p, p')$, $\delta' = d(p, p'')$. If $p'' \neq v$ and $\delta \leq r$, then*

$$\delta' \leq \delta \sqrt{1 - \frac{\delta^2}{4R^2}} < \delta \exp\left(- \frac{\delta^2}{8R^2} \right). \tag{44}$$

Proof The first inequality follows from Theorem 8 and the definition of R. To prove the next inequality, we use that when $x \neq 0$, $1 + x < \exp(x)$, and set $x = -\delta^2/4R^2$. □

Lemma 1 *Assume p is in $conv(S)$. Pick $p_0 \in conv(S)$, let $\delta_0 = d(p_0, p)$. Assume $\delta_0 \leq R_0 = \min\{d(p, v_i), i = 1, \ldots, n\}$.*

Let $k \equiv k(\delta_0)$ be the maximum number of iterations of the Triangle Algorithm in order to compute $p_k \in conv(S)$ so that if $\delta_j = d(p_j, p)$ for $j = 1, \ldots, k$, we have

$$\delta_k < \frac{\delta_0}{2} \leq \delta_j, \quad j = 1, \ldots, k-1. \tag{45}$$

Then, k satisfies

$$k = k(\delta_0) \leq \lceil N_0 \rceil, \quad N_0 \equiv N(\delta_0) = (32 \ln 2)\frac{R^2}{\delta_0^2} < 23\frac{R^2}{\delta_0^2} \tag{46}$$

Proof For each $j = 1, \ldots, k-1$, Corollary 5 applies. The repeated application of (44) in Corollary 5, the assumption (45) that for each such j, $\delta_j \geq \delta_0/2$, and the monotonicity of the exponential function implies

$$\delta_j < \delta_{j-1} \exp\left(-\frac{\delta_{j-1}^2}{8R^2}\right) \leq \delta_{j-1} \exp\left(-\frac{\delta_0^2}{32R^2}\right) < \delta_{j-2} \exp\left(-\frac{2\delta_0^2}{32R^2}\right) \leq \cdots \tag{47}$$

It follows that

$$\delta_{k-1} < \delta_0 \exp\left(-\frac{(k-1)\delta_0^2}{32R^2}\right). \tag{48}$$

Thus from (44) and (48) we have

$$\delta_k < \delta_{k-1} \exp\left(-\frac{\delta_{k-1}^2}{8R^2}\right) \leq \delta_0 \exp\left(-\frac{k\delta_0^2}{32R^2}\right). \tag{49}$$

To have $\delta_k < \delta_0/2$, it suffices to satisfy

$$\exp\left(-\frac{k\delta_0^2}{32R^2}\right) \leq \frac{1}{2}. \tag{50}$$

Solving for k in the above inequality implies the claimed bound in (46). □

Theorem 9 *The Triangle Algorithm correctly solves problem (P) as follows:*

(i) Suppose $p \in conv(S)$. Given $\epsilon > 0$, $p_0 \in conv(S)$, with $\delta_0 = d(p, p_0) \leq R_0 = \min\{d(p, v_i) : i = 1, \ldots, n\}$. The number of iterations K_ϵ to compute a point p_ϵ in $conv(S)$ so that $d(p, p_\epsilon) < \epsilon d(p, v_i)$, for some $v_i \in S$ satisfies

$$K_\epsilon \leq \frac{48}{\epsilon^2} = O(\epsilon^{-2}). \tag{51}$$

(ii) Suppose $p \notin conv(S)$. If Δ denotes the distance from p to $conv(S)$, i.e.

$$\Delta = \min\left\{d(x, p) : \quad x \in conv(S)\right\}, \tag{52}$$

the number of iterations K_Δ to compute a p-witness, a point p_Δ in $conv(S)$ so that $d(p_\Delta, v_i) < d(p, v_i)$ for all $v_i \in S$, satisfies

$$K_\Delta \leq \frac{48R^2}{\Delta^2} = O\left(\frac{R^2}{\Delta^2}\right). \tag{53}$$

Proof From Lemma 1 and definition of $k(\delta_0)$ (see (46)), in order to half the initial gap from δ_0 to $\delta_0/2$, in the worst-case the Triangle Algorithm requires $k(\delta_0)$ iterations. Then, in order to reduce the gap from $\delta_0/2$ to $\delta_0/4$ it requires at most $k(\delta_0/2)$ iterations, and so on. From (46), for each nonnegative integer r the worst-case number of iterations to reduce a gap from $\delta_0/2^r$ to $\delta_0/2^{r+1}$ is given by

$$k\left(\frac{\delta_0}{2^r}\right) \leq \left\lceil N\left(\frac{\delta_0}{2^r}\right)\right\rceil = \lceil 2^{2r} N_0 \rceil \leq 2^{2r} \lceil N_0 \rceil. \tag{54}$$

Therefore, if t is the smallest index such that $\delta_0/2^t < \epsilon R$, i.e.

$$2^{t-1} \leq \frac{\delta_0}{R\epsilon} < 2^t, \tag{55}$$

then the total number of iterations of the algorithm, K_ϵ, to test if condition (i) is valid satisfies:

$$\begin{aligned} K_\epsilon &\leq \lceil N_0 \rceil (1 + 2^2 + 2^4 + \cdots + 2^{2(t-1)}) \leq \lceil N_0 \rceil \frac{2^{2t} - 1}{3} \leq \lceil N_0 \rceil 2 \times 2^{2(t-1)} \\ &\leq (N_0 + 1) \frac{2\delta_0^2}{R^2 \epsilon^2} \end{aligned} \tag{56}$$

From (46) we get

$$K_\epsilon \leq \left(23 \frac{R^2}{\delta_0^2} + 1\right) \frac{2\delta_0^2}{R^2 \epsilon^2} = \left(23 + \frac{\delta_0^2}{R^2}\right) \frac{2}{\epsilon^2}. \tag{57}$$

Since $p \in conv(S)$ and from the definition of R (see (43)) we have $\delta_0 = d(p, p_0) \leq R$, hence we get the claimed bound on K_ϵ in (51).

Suppose $p \notin conv(S)$. If suffices to choose $\epsilon = \Delta / R$. Then from (51) and the definition of Δ we get the bound on K_Δ in (53). □

6 Strict distance duality and a minimax duality

Here we first prove a stricter version of the distance duality, giving a stronger distance duality when $p \in conv(S)$. First we give a definition.

Definition 8 Given $p' \in conv(S)$, we say $v_j \in S$ is a *strict pivot* relative to p (or *strict* p-pivot, or simply *strict* pivot) if $\angle p' p v_j \geq \pi/2$ (see Fig. 8).

Theorem 10 (Strict Distance Duality) *Assume $p \notin S$. Then $p \in conv(S)$ if and only if for each $p' \in conv(S)$ there exists strict p-pivot, v_j. In particular,*

$$(p' - p)^T (v_j - p) \leq 0. \tag{58}$$

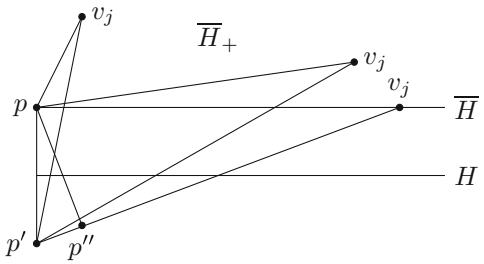

Fig. 8 Several examples of strict p-pivot

Proof Suppose $p \in conv(S)$. Consider the orthogonal bisecting hyperplane to the line $p'p$, H, and the hyperplane parallel to it passing through p, $\overline{H}$, see Fig. 8. Let $\overline{H}_+$ be the halfspace determined by this hyperplane that excludes p'. We claim it must contain a point $v_j \in S$ (see Fig. 8). Otherwise, p must be an extreme point of $conv(S \cup \{p\})$, but since $p \notin S$, this implies $p \notin conv(S)$. This implies $\angle p'pv_j$ is at least $\pi/2$. Hence, (58) is satisfied.

Conversely, suppose that for each $p' \in conv(S)$, there exists $v_j \in conv(S)$ such that $\angle p'pv_j$ is at least $\pi/2$. If $p \notin conv(S)$, consider a witness p'. Then for each $v_j \in S$, $\angle p'pv_j < \pi/2$, a contradiction. □

Theorem 11 *Given $p' \in conv(S)$, suppose v_j is a strict p-pivot. Then if $\delta = d(p, p')$, $r = d(p, v_j)$, and $\delta' = d(p, p'')$, p'' the nearest to p on the line segment $p'v_j$, we have*

$$\delta' = \frac{\delta r}{\sqrt{r^2 + \delta^2}} \leq \delta\sqrt{1 - \frac{\delta^2}{2r^2}} \leq \delta \exp\left(-\frac{\delta^2}{4r^2}\right). \tag{59}$$

Proof It is easy to see that for a given strict pivot v_j, the worst-case of error occurs when $\angle p'pv_j = \pi/2$, (see Fig. 8). Now using the similarity of the triangles $\triangle p'pv_j$ and $\triangle pp''p'$ (see Fig. 8), we get the equality in the theorem. To prove the first inequality, we use the fact that for a positive number $x < 1$,

$$\frac{1}{1+x} \leq 1 - \frac{x}{2}, \tag{60}$$

and let $x = \delta^2/r^2$. The second inequality follows from the inequality $1 - x \leq \exp(x)$. □

Thus when $p \in conv(S)$ using a strict pivot we get a better constant in the worst-case complexity of the Triangle Algorithm than using any pivot.

Definition 9 We say $p' \in conv(S)$ is a *strict witness* relative to p (or simply *strict* witness) if there is no strict pivot at p'. Equivalently, p' is a strict witness if the orthogonal hyperplane to the line $p'p$ at p separates p from $conv(S)$. We denote the set of all strict witnesses by $\widehat{W}_p$.

Clearly $\widehat{W}_p$ contains W_p (see Definition 4). However, interestingly while W_p is not described by a set of linear inequalities, $\widehat{W}_p$ can be characterized by a set of strict linear inequalities. The following is straightforward.

Proposition 3 *We have*

$$\widehat{W}_p = \left\{x \in conv(S) : (x - p)^T(v_i - p) > 0, i = 1, \ldots, n\right\}. \tag{61}$$

□

Clearly $p \in conv(S)$ if and only if $\widehat{W}_p$ is empty. Figure 9 shows the difference between W_p and $\widehat{W}_p$. Its witness set was considered in an earlier example. The figure suggest that when $p \notin conv(S)$, using a strict pivot would detect the infeasibility of p sooner than using any pivot.

Let us consider the strict distance duality for the case when $p = 0$, and let $A = [a_1, \ldots, a_n]$ be the matrix of the points in S. This can always be assumed since we can shift each of the points by p. In this case $p \in conv(S)$ if and only if $Ax = 0, e^Tx = 1, x \geq 0$ is feasible. The corresponding strict duality then implies given $x \in \Sigma_n = \{x : e^Tx = 1, x \geq 0\}$, there exists a_j such that $a_j^TAx \leq 0$. This together with the fact that minimum of a linear function

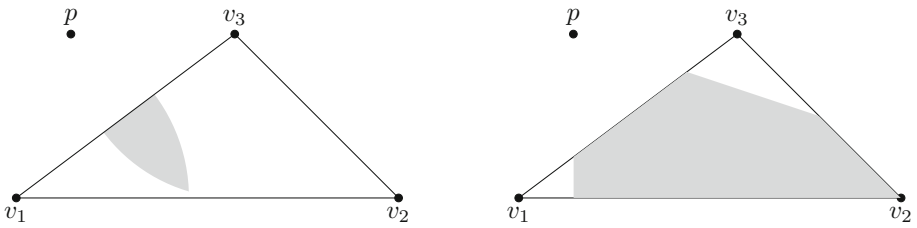

Fig. 9 Witness set W_p (*left*) and strict Witness set $\widehat{W}_p$ (*right*)

over the simplex Σ_n is attained at a vertex, implies the following: given $x \in \Sigma_n$, there exists a_j such that

$$\min_{y \in \Sigma_n} y^T A^T A x = a_j^T A x \leq 0. \tag{62}$$

Combining this with von Neumann's minimax theorem we can thus state the following

Theorem 12 *The set* $\{x : Ax = 0, x \in \Sigma_n\}$ *is feasible if and only if*

$$w_* = \max_{x \in \Sigma_n} \min_{y \in \Sigma_n} y^T A^T A x = \min_{y \in \Sigma_n} \max_{x \in \Sigma_n} y^T A^T A x \leq 0.$$

The quality w_* is the value of the matrix $Q = A^T A$. If $w_* < 0$ then it can be shown that there exists a number $\lambda_* \in (0, 1)$ such that in each iteration of the Triangle Algorithm we have $\delta_{k+1} \leq \lambda_* \delta_k$ (see (97)).

7 Alternate complexity bound for Triangle Algorithm

Throughout the section we assume p, S and R are as defined previously, and $\epsilon \in (0, 1)$.

Definition 10 Given $\epsilon \in (0, 1)$, let

$$\theta^* = \max\{\theta' = \angle v p' p : \quad v \in S, \quad p' \in conv(S), \quad d(p', v) \geq d(p, v), \quad d(p, p') \geq \epsilon R\}. \tag{63}$$

For a given $p' \in conv(S)$ satisfying $d(p, p') \geq \epsilon R$, set

$$\theta_{p'} = \min\{\theta' = \angle v p' p : \quad v \in S, \quad d(p', v) \geq d(p, v)\}. \tag{64}$$

Let

$$\theta_* = \max\{\theta_{p'} : \quad p' \in conv(S), \quad d(p', v) \geq d(p, v), \quad d(p, p') \geq \epsilon R\}. \tag{65}$$

Let

$$\widehat{\theta}_* = \sup\{\theta_{p'} : \quad p' \in conv(S), \quad d(p', v) \geq d(p, v)\}. \tag{66}$$

To describe θ^* and θ_* geometrically and the corresponding visibility constants, let $B_{\epsilon R}(p)$ be the open ball of radius ϵR at p. Then θ^* is the maximum of all pivot angles as p' ranges in $conv(S) \setminus B_{\epsilon R}(p)$. For each iterate $p' \in conv(S) \setminus B_{\epsilon R}(p)$, let $v_{p'}$ denote a pivot where the pivot angle $\angle p p' v_{p'}$ is the least among all such pivots. We refer to $v_{p'}$ as the *best pivot* at p'. Also, θ_* is the maximum of these angles as p' ranges in $conv(S) \setminus B_{\epsilon R}(p)$. Then $\widehat{\theta}_*$ is the supremum of these angles as p' ranges in $conv(S)$. The supremum is not necessarily attained, e.g. the case where $conv(S)$ is a square and p lies on an edge. In general because

θ^*, θ_* depend on ϵ, they lie in $[0, \pi/2)$ but $\widehat{\theta}_*$ lies in $[0, \pi/2]$. For instance, in the example of square mentioned above with p on an edge $\widehat{\theta}_* = \pi/2$. The farther away these angles are from $\pi/2$, the more effective steps the Triangle Algorithm will take in each iteration.

Proposition 4 *We have*

(i)

$$\theta_* \leq \theta^*. \tag{67}$$

(ii)

$$\sin\theta_* \leq \sin\theta^* \leq \frac{1}{\sqrt{1+\epsilon^2}}. \tag{68}$$

Proof (i) is immediate from the definitions of θ_* and θ^*. Since both angles are acute the first inequality in (ii) follows. To prove the second inequality in (ii), consider the triangle $\triangle pp'v$ (see Fig. 11). Since $\angle p'pv$ is non-acute it is easy to argue

$$\sin\theta' \leq \frac{d(p, v)}{\sqrt{d^2(p, v) + d^2(p, p')}}. \tag{69}$$

Dividing the numerator and denominator of the right-hand-side of the above by $d(p, v)$, and using that $d(p, p')/d(p, v) \geq d(p, p')/R \geq \epsilon$, we get the second inequality in (ii). □

Definition 11 Given p, $S = \{v_1, \ldots, v_n\}$, $A = [v_1, \ldots, v_n]$, and $\epsilon \in (0, 1)$, the *arbitrary-pivot visibility constant*, ν^*, and the corresponding *arbitrary-pivot visibility factor*, $c^* = c^*[p, A, \epsilon]$, are determined from the equation

$$\nu^* = \sin\theta^* = \frac{1}{\sqrt{1+c^*}}. \tag{70}$$

The *best-pivot visibility constant*, ν_*, and the corresponding *best-pivot visibility factor*, $c_* = c_*[p, A, \epsilon]$ are determined from the equation

$$\nu_* = \sin\theta_* = \frac{1}{\sqrt{1+c_*}}. \tag{71}$$

The *absolute visibility constant*, $\widehat{\nu}_*$, and the corresponding *absolute visibility factor*, $\widehat{c}_* = \widehat{c}_*[p, A]$ are determined from the equation

$$\widehat{\nu}_* = \sin\widehat{\theta}_* = \frac{1}{\sqrt{1+\widehat{c}_*}}. \tag{72}$$

Let θ_p be the maximum of pivot-angles over a sequence of iterates of the Triangle Algorithm as applied to the given input data, S, p, ϵ, before it terminates. The *(observed) visibility constant*, ν, and the corresponding *(observed) visibility factor*, $c = c[p, A, \epsilon]$ are determined from he equation

$$\nu = \sin\theta_p = \frac{1}{\sqrt{1+c}}. \tag{73}$$

Using the inner product formula $\cos\theta = u^T v/\|u\|\|v\|$, where θ is the angle between $u, v \in \mathbb{R}^m$, and setting $F(x) = (x-p)^T(x-v)/\|x-p\|\|x-v\|$, we may give alternative definition the above visibility parameters. For instance, ν_* and $\widehat{\nu}_*$ can be defined as

$$\nu_* = \max_{\{x \in conv(S),\ d(x,p) \geq \epsilon R\}} \ \min_{\{v \in S,\ d(x,v) \geq d(p,v)\}} \sqrt{1 - F(x)^2}, \tag{74}$$

$$\widehat{\nu}_* = \sup_{\{x \in conv(S)\}} \ \min_{\{v \in S,\ d(x,v)s \geq d(p,v)\}} \sqrt{1 - F(x)^2}. \tag{75}$$

Note that generally ν_* is dependent on ϵ while $\widehat{\nu}_*$ is independent of ϵ.

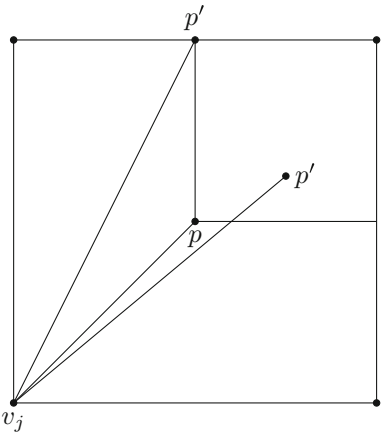

Fig. 10 When $conv(S)$ is a square, p the center, $\theta_* = \pi/8$, $\widehat{\nu}_* \approx 38$

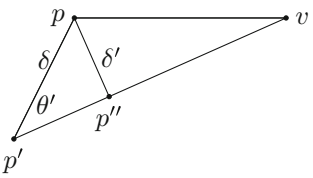

Fig. 11 Relationship between $\delta = d(p', p)$, $\delta' = d(p'', p)$, and θ'

Example 1 Consider the case where S is the vertices of a square and p is the center, Fig. 10. In this case $\theta_* = \pi/8$ and the absolute visibility constant satisfies $\widehat{\nu}_* = \sin \pi/8 \approx .3826$. By moving p inside the square, $c(p, A, \epsilon)$ varies. However, not drastically if it stays reasonably away from the boundary. The visibility constants become worst when p is at the midpoint of one of the edges. The absolute visibility constant in this case is at its worst, equal to one.

We now prove an alternative complexity bound for the Triangle Algorithm.

Theorem 13 *Suppose $p \in conv(S)$. Let $\delta_0 = d(p, p_0)$, $p_0 \in conv(S)$. The number of arithmetic operations of the Triangle Algorithm to get $p_\epsilon \in conv(S)$ such that $d(p_\epsilon, p) < \epsilon R$, $R = \max\{d(p, v_j) : v_j \in S\}$, having (observed) visibility constant ν and (observed) visibility factor c, is*

$$O\left(mn\frac{1}{c}\ln\frac{\delta_0}{\epsilon R}\right). \tag{76}$$

In particular, when the algorithm uses the arbitrary-pivot strategy its complexity is

$$O\left(mn\frac{1}{c^*}\ln\frac{\delta_0}{\epsilon R}\right), \tag{77}$$

and when it uses the best pivot-strategy its complexity is

$$O\left(mn\frac{1}{c_*}\ln\frac{\delta_0}{\epsilon R}\right). \tag{78}$$

Proof With $\delta = d(p', p)$ and $\delta' = d(p'', p)$, where p' and p'' are two consecutive iterates and v the corresponding pivot, and $\theta' = \angle pp'v$, we have (see Fig. 11)

$$\frac{\delta'}{\delta} = \sin\theta' \leq \nu. \tag{79}$$

Thus if the iterates in the Triangle Algorithm are p_i, and $\delta_i = d(p_i, p)$, then after k iterations we will have

$$\delta_k \leq \nu^k \delta_0. \tag{80}$$

To have the right-hand-side to be bounded above by ϵR, k must satisfy

$$k \ln \nu = k \ln \frac{1}{\sqrt{1+c}} \leq \ln \frac{\epsilon R}{\delta_0}. \tag{81}$$

Equivalently,

$$\frac{k}{2} \ln(1+c) \geq \ln \frac{\delta_0}{\epsilon R}. \tag{82}$$

For $u \in (0, 1)$ we may write

$$\ln(1+u) \geq \frac{u}{2}. \tag{83}$$

Thus it suffices to choose k satisfying

$$\frac{k}{4} c \geq \ln \frac{\delta_0}{\epsilon R}. \tag{84}$$

Each iteration takes at most $O(mn)$ arithmetic operations. Under arbitrary pivot strategy we have, $\nu \leq \nu^*$ and under the best pivot strategy we have $\nu \leq \nu_*$. These imply, $c^* \leq c$ and $c_* \leq c$, respectively. Hence the claimed complexity bounds. □

Remark 6 Let us assume $\epsilon = 2^{-L}$ for some natural number L. We examine the number of iterations of the Triangle Algorithm to compute p_k so that $\delta_k \leq 2^{-L}\delta_0$. Since $\delta_k \leq \nu^k \delta_0$, if the visibility constant satisfies $\nu = 0.5$, then clearly the number of iterations is L. Table 2 shows the number of such iterations when $\nu = .9, .99, .999, .9999$, and $.99999$. As we see the numbers are quite reasonable and independent of m, n.

Theorem 14 *Assume p lies in the relative interior of $conv(S)$. Let ρ be the supremum of radii of the balls centered at p in this relative interior. Let $\delta_0 = d(p_0, p)$, $p_0 \in conv(S)$. Let $R = max\{d(p, v_i), i = 1, \ldots, n\}$. Given $\epsilon \in (0, 1)$, suppose the Triangle Algorithm uses a strict pivot in each iteration. The number of arithmetic operations to compute $p' \in conv(S)$ such that $d(p', p) < \epsilon R$ satisfies*

$$O\left(mn \frac{R^2}{\rho^2} \ln \frac{\delta_0}{\epsilon R}\right), \quad \delta_0 = d(p_0, p). \tag{85}$$

In particular, if $\rho \geq \epsilon^{1/t} R$, t a natural number then the complexity is

$$O\left(mn \frac{1}{\sqrt[t]{\epsilon^2}} \ln \frac{\delta_0}{\epsilon R}\right). \tag{86}$$

Table 2 Number of iterations of Triangle Algorithm for different visibility constants in order to obtain $\delta_k \leq 2^{-L}\delta_0$, independent of m and n

Visibility constant ν	Bound on no. of iterations k
.9	7 × L
.99	70 × L
.999	700 × L
.9999	7000 × L
.99999	70000 × L

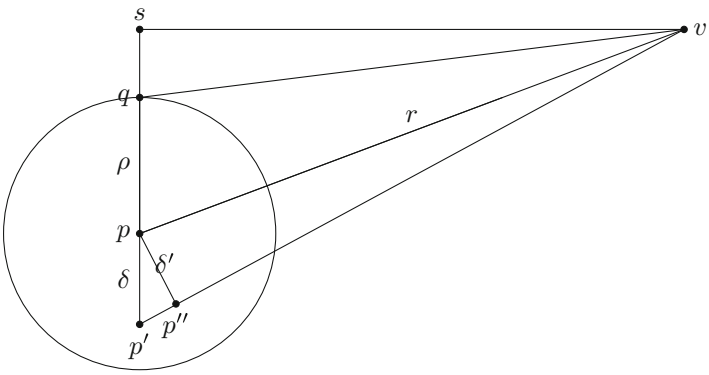

Fig. 12 Existence of a strict pivot with visibility constant bounded by $1/\sqrt{1+\rho^2/R^2}$

Proof Given the current iterate p' where $d(p, p') \geq \epsilon R$, let q be the point on the extension of the line $p'p$, where $d(p, q) = \rho$, see Fig. 12. Then we have $q \in conv(S)$. The orthogonal hyperplane to the line segment pq must contain a point v in S on the side that excludes p, see Fig. 12. For such strict pivot v, $\angle pqv$ is non-acute.

Let p'' be the next iterate. Then we have (see Fig. 12):

$$\sin \angle pp'v = \frac{\delta'}{\delta}. \tag{87}$$

Let $r = d(p, v)$. We claim

$$\sin \angle pp'v \leq 1/\sqrt{1+\frac{\rho^2}{r^2}}. \tag{88}$$

To prove this, note that we have

$$\sin \angle pp'v \leq \sin \angle qpv. \tag{89}$$

However, considering that $\angle pqv$ is non-acute, we can introduce s so that $\angle psv = \pi/2$. Thus we may write

$$\begin{aligned} \sin \angle qpv = \sin \angle spv &= \frac{\sqrt{r^2 - (d(q,s)+\rho)^2}}{r} \leq \frac{\sqrt{r^2-\rho^2}}{r} \\ &= \sqrt{1-\frac{\rho^2}{r^2}} \leq 1/\sqrt{1+\frac{\rho^2}{r^2}} \leq 1/\sqrt{1+\frac{\rho^2}{R^2}}. \end{aligned} \tag{90}$$

Thus if δ_k represents the gap in k iterations and δ_0 the initial gap, then we have

$$\delta_k \leq \left(1/\sqrt{1+\frac{\rho^2}{R^2}}\right)^k \delta_0. \tag{91}$$

To have $\delta_k < \epsilon R$ it suffices to find k so that

$$-\frac{k}{2} \ln\left(1+\frac{\rho^2}{R^2}\right) < \ln \frac{\epsilon R}{\delta_0}. \tag{92}$$

Equivalently,

$$\frac{k}{2} \ln\left(1+\frac{\rho^2}{R^2}\right) > \ln \frac{\delta_0}{\epsilon R}. \tag{93}$$

As shown in previous theorem, for $u \in (0, 1)$ we have, $\ln(1+u) \geq \frac{u}{2}$. Thus we choose k satisfying

$$k > \frac{4R^2}{\rho^2} \ln \frac{\delta_0}{\epsilon R}. \tag{94}$$

□

Remark 7 According to the above theorem the best-pivot visibility constant, ν_*, will be good when p lies in a reasonable size ball in the relative interior of $conv(S)$. For instance, in the example of a square in Fig. 10, aside from the case when p is center of the square, for points in a reasonably large circle we would expect the complexity of the Triangle Algorithm to be very good. Only points near a boundary line can slow down the algorithm. However, even for these points when iteration begins to slow down we can introduce *auxiliary pivots* to improve the visibility constant. This will be considered in a subsequent section.

We may summarize the performance of the Triangle Algorithm with the following complexity theorem.

Theorem 15 *Let $\delta_0 = d(p, p_0)$, $p_0 \in conv(S)$. Suppose $p \in conv(S)$. The number of arithmetic operations of the Triangle Algorithm to get $p_\epsilon \in conv(S)$ such that $d(p_\epsilon, p) < \epsilon R$, $R = \max\{d(p, v_j) : v_j \in S\}$, where the visibility factor is $c_1 = c(p, A, \epsilon)$, satisfies*

$$O\left(mn \min\left\{\frac{1}{\epsilon^2}, \frac{1}{c_1} \ln \frac{\delta_0}{\epsilon R}\right\}\right), \quad c_1 \geq \epsilon^2. \tag{95}$$

Suppose $p \notin conv(S)$. The number of arithmetic operations of the Triangle Algorithm to get a witness $p' \in conv(S)$, where the visibility factor is $c_2 = c(p, A, \Delta/R)$ satisfies

$$O\left(mn \min\left\{\frac{R^2}{\Delta^2}, \frac{1}{c_2} \ln \frac{\delta_0}{\Delta}\right\}\right), \quad c_2 \geq \frac{\Delta^2}{R^2}. \tag{96}$$

We close this section by using the strict distance duality to associate *best-strict pivot visibility constant* to a given input data p and $S = \{a_1, \ldots, a_n\}$. Consider the Triangle Algorithm where given an iterate $p' \in conv(S)$, it selects a pivot v so that the angle $\angle p'pv$ is the largest. Without loss of generality we assume $p = 0$ and let $A = [a_1, \ldots, a_n]$. By the strict duality theorem, given $x \in \Sigma_n = \{x : e^T x = 1, x \geq 0\}$, there exists a_j such that $a_j^T Ax \leq 0$.

Definition 12 Given $p = 0$ and $S = \{a_1, \ldots, a_n\}$, $A = [v_1, \ldots, a_n]$, the *best-strict pivot visibility constant* with respect to p and S is the number

$$\lambda_* = \sqrt{1 - \phi_*^2}, \tag{97}$$

where

$$\phi_* = \max_{x \in \Sigma_n} \min_{a_j \in S} \frac{a_j^T Ax}{\|a_j\| \|Ax\|}. \tag{98}$$

From the definition of λ_* and ν_* (see (70)), it is easy to conclude that $\nu_* \leq \lambda_*$.

Proposition 5 *Suppose $p = 0 \in conv(S)$. If $\lambda_* < 1$, then in k iteration of the Triangle Algorithm using strict pivot, the gap $\delta_k = d(p, p_k)$ will satisfy*

$$\delta_k \leq \lambda_*^k \delta_0. \tag{99}$$

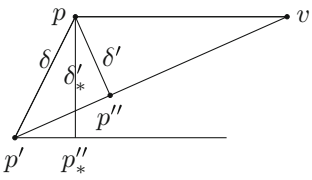

Fig. 13 $\delta' < \delta'_*$, since $\angle p'pv$ is non-acute

Proof It suffices to show that if $\delta = \|p'\|$ and $\delta' = \|p''\|$ are two consecutive gaps in the Triangle Algorithm, computed based on a strict pivot v at the current iterate p', then $\delta' \leq \lambda_* \delta$. By the definition of ϕ_* and the inner product formula for cosine, we must have $\cos(\angle p'pv) \leq \phi_*$. Thus $\sin(\pi - \angle p'pv) = \delta'_*/\delta$, for some δ'_* satisfying $\delta' \leq \delta'_*$, see Fig. 13. But from definition $\lambda_* = \sin(\pi - \angle p'pv)$. □

8 Virtual Triangle Algorithm and its approximated version

Here we will consider a version of the Triangle Algorithm that works with *virtual* iterates. We call the iterates as such because in each iteration we only know the coordinates of the iterates but not their representation as points in $conv(S)$. In fact they may not even lie in $conv(S)$. We thus call the algorithm *Virtual Triangle Algorithm*. Describing such an algorithm has two advantages: (i) it may quickly identify the case when $p \notin conv(S)$; (ii) it gives rise to a version of the Triangle Algorithm, *Approximated Virtual Triangle Algorithm*, that has very efficient complexity, does provide an approximate solution in $conv(S)$ when $p \in conv(S)$, however it may not necessarily give an approximation to within a desired accuracy. It may also quickly provide a witness when $p \notin conv(S)$. If it does not solve the problem it may simply be considered as a preprocessing, see (16) for its complexity. Consider the box below and Fig. 14.

Virtual Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, $p, \epsilon \in (0, 1)$)

- **Step 0.** (**Initialization**) Let $p' = v = \text{argmin}\{d(p, v_i) : v_i \in S\}$.
- **Step 1.** If $d(p, p') < \epsilon d(p, v)$, output p' as ϵ-approximate solution, stop. Otherwise, if there exists a pivot v_j, replace v with v_j. If no pivot exists, output p' as a *general p-witness*, stop.
- **Step 2.** Let

$$\overline{v}_j = p + \frac{d(p, p')}{d(p, v_j)}(v_j - p). \tag{100}$$

Replace p' with $\overline{p}''$ below, and go to Step 1.

$$\overline{p}'' = \frac{1}{2}p' + \frac{1}{2}\overline{v}_j. \tag{101}$$

Theorem 16 *If $p \in conv(S)$, the Virtual Triangle Algorithm in $O(mn \ln \epsilon^{-1})$ arithmetic operations generates a point $p' \in conv(S)$ such that $d(p, p') < \epsilon R$. However, p' has no certificate (i.e. the representation of p' as a convex combination v_i's is not available).*

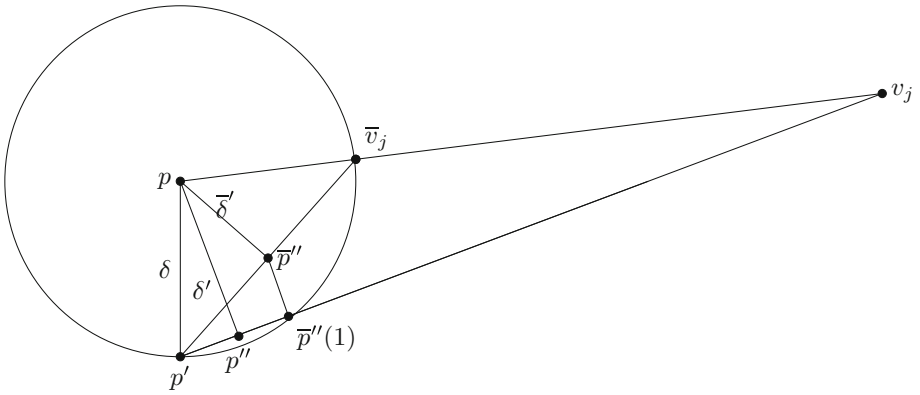

Fig. 14 One iteration of virtual Triangle Algorithm, $\overline{p}''$, and approximated version $\overline{p}''(1)$

Proof If $p \in conv(S)$, so is $\overline{v}_j$, we Fig. 14. In the worst-case the triangle $\triangle pp'\overline{v}_j$ is an equilateral. From this it follows that it in each iteration we have

$$\delta' \leq \frac{\sqrt{3}}{2}\delta. \tag{102}$$

Hence, analogous to the analysis of previous theorem the proof of complexity is immediate. □

Remark 8 If $p \notin conv(S)$ the Virtual Triangle Algorithm may still correctly determine this by finding a witness with respect to $\overline{v}_j$. For example, consider Fig. 14 and assume S consists of p', v_j and $\overline{v}_j$. Then $p \notin conv(S)$ and neither p' nor p'' is a witness. However, $\overline{p}''$ is a witness.

The Virtual Triangle Algorithm does not provide an approximation when $p \in conv(S)$, since it does not have a representation for $\overline{v}_j$ as a point in S. Thus even if the iterate $\overline{p}''$ lies in $conv(S)$ it has no representation as a point in $conv(S)$. To remedy this we offer a variation where, starting with a p'', we estimate $\overline{p}''$, using the Triangle Algorithm itself. Consider the following algorithm.

Approximated Virtual Triangle Algorithm ($S = \{v_1, \ldots, v_n\}$, $p, \epsilon \in (0, 1), t \in \mathbb{Z}_+$)

- **Step 0.** (**Initialization**) Let $p' = v = \text{argmin}\{d(p, v_i) : v_i \in S\}$.
- **Step 1.** If $d(p, p') < \epsilon d(p, v)$, output p' as ϵ-approximate solution, stop. Otherwise, if there exists a pivot v_j, replace v with v_j. If no such pivot exists, then output p' as a witness, stop.
- **Step 2.** Let $p'' = \sum_{i=1}^{n} \alpha' v_i$ be as in Step 2 of Triangle Algorithm (see (41)). Then, using $\overline{p}''(1)$, the nearest point to p'' on the line $p'v_j$, perform at most t iterations of the Triangle Algorithm to test if $\overline{p}''$ lies in $conv(S)$. If any of the iterates, $\overline{p}''(r)$, $r \leq t$, is a witness that $\overline{p}''(r) \notin conv(S)$, then $p \notin conv(S)$, stop.
- **Step 3.** Replace p' with $\overline{p}''(r)$ and go to Step 1.

The advantage of iterating towards $\overline{p}''$ as opposed to iterating directly toward p lies in that $\overline{p}''$, or an approximation to it, offers a new vantage point in order to get a good reduction

toward p itself in subsequent iterations. The following result assures that in each iteration the angle $pp'\overline{p}''(r)$ improves.

Proposition 6 *For any $r \geq 2$, $\angle pp'\overline{p}''(r) < \angle pp'p''$. Thus if $\overline{p}''(r)$ is used as a p-pivot, then $d(p, \overline{p}''(r)) < d(p, p'')$.*

Proof We first note that $\overline{p}''(1)$ is nearest point to p'' on the line $p'v_j$. However, for $r > 1$, $p''(r)$ may not lie in the same Euclidean plane as the one shown in the Fig. 14. We have

$$\angle pp'v_j = \angle pp'\overline{v}_j + \angle \overline{v}_j p'v_j. \tag{103}$$

On the other hand note that we must have

$$\angle pp'p''(r) \leq \angle pp'\overline{v}_j + \angle \overline{v}_j p'\overline{p}''(r). \tag{104}$$

However, since $d(\overline{p}''(r), \overline{p}'') < d(p'', \overline{p}'')$, it follows that

$$\angle \overline{v}_j p'\overline{p}''(r) < \angle \overline{v}_j p'v_j. \tag{105}$$

□

The following is straightforward and shows that the Approximated Virtual Triangle Algorithm may perform well if the visibility constant ν_* is a reasonable constant.

Theorem 17 *Suppose $p \in conv(S)$, and $\epsilon = 2^{-L}$, where L is a natural number. Suppose the Approximated Virtual Triangle Algorithm uses the best-pivot strategy. Then, in $O(mnLt)$ arithmetic operations it computes a point $p' \in conv(S)$ so that*

$$d(p, p') \leq \nu_*^{tL}\delta_0, \quad \delta_0 = d(p, p_0). \tag{106}$$

9 Triangle Algorithm with auxiliary pivots

According to Theorem 8 the worst-case error bound of Triangle Algorithm for a given iterate $p' \in conv(S)$, and pivot $v_j \in S = \{v_1, \ldots, v_n\}$ depends on $r = d(p, v_j)$ and $d(p, p')$. Specifically, if p'' is the next iterate then $\delta' = d(p, p'')$ is bounded above by $\delta\sqrt{1 - \delta^2/4r^2}$. According to the alternate analysis, the ratio δ'/δ is also related to the visibility constants. These affect the complexity of the Triangle Algorithm. In this section we discuss how to introduce new points into S so as to allow improving the visibility constant. In the Approximated Virtual Triangle Algorithm we already have made implicit use of such pivot, namely by introducing the point $\overline{v}_j$ (see Fig. 14).

Suppose we have a finite subset S' in $conv(S)$, distinct from S. We refer to S' as *auxiliary pivots*. Suppose that in an iteration of the Triangle Algorithm the search for a pivot is carried out over $S \cup S'$. Then it may be possible to improve δ' by using as pivot a point in S'. Not all such points may end up being used as pivots.

Overall this increases $n = |S|$ to $n' = |S| + |S'|$, but it could drastically improve visibility constants. Here we discuss three strategies. Many other strategies are possible.

- **Strategy I.** Given $p, p' \in conv(S)$ and a p-pivot v, if v is a strict pivot, or if $d(p, p') \leq d(p, v)$ (see Fig. 11), we can switch the role of p' and v. In other words, p' as an auxiliary point is a v-pivot, implying that the next iteration of the Triangle Algorithm, starting from v, is at least as effective. Thus if $\delta' = d(p, p'')$ does not sufficiently reduce $\delta = d(p, p')$, we add p' as auxiliary pivot and choose v to be the next iterate. We may wish to delete

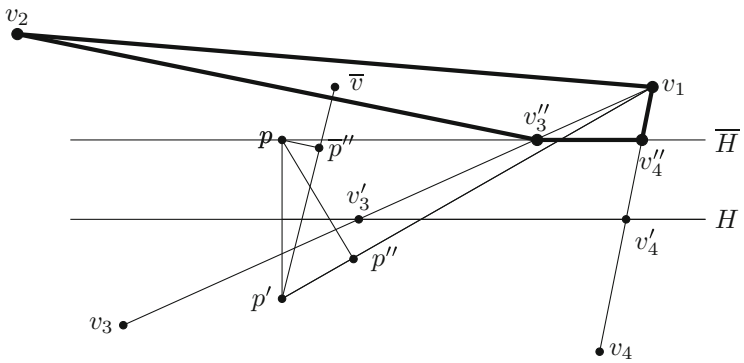

Fig. 15 Several examples of auxiliary pivots

p', or keep it as an auxiliary point for subsequent iterations. As an example, consider the case where p is a midpoint of an edge on a square. This strategy quickly detects p to lie in the convex hull.

- **Strategy II.** Given p, $p' \in conv(S)$ and p-pivot v in an iteration of the Triangle Algorithm, let H be the orthogonal bisecting hyperplane to the line pp' (see Fig. 15, consider p, p', $v = v_1$). In the next iteration replace S with $S \cup S'$, where S' is the set of all intersections of the line segments vv_j, $v_j \in S$, with the orthogonal hyperplane that bisects pp' (see Fig. 15 $S' = \{v_3', v_4'\}$).
- **Strategy III.** Given p, p', v, and H as above, let $\overline{H}$ be the hyperplane parallel to H, passing through p. Let $\overline{H}_+$ be the halfspace that excludes p'. Let S' be the set of intersections, if any, of the line segments vv_j with $\overline{H}$ (see Fig. 15, $\{v_3'', v_4''\}$). Consider the set $conv((S \cap \overline{H}_+) \cup S')$ (see Fig. 15, $conv\{v_1, v_2, v_3'', v_4''\}$, enclosed by the thick lines). Applying the Triangle Algorithm find an approximation, $\overline{v}$, to the closest point to p' in this set. This is a witness that $p' \notin conv((S \cap \overline{H}_+) \cup S')$. Then we find the closest point to p on the line $p'\overline{v}$, $\overline{p}''$. To find $\overline{v}$ we use the Triangle Algorithm itself. We would expect that $d(p', \overline{v}) < d(p', v)$ and that the computation of $\overline{p}''$ can be done efficiently. Instead of using v as pivot at p' to get p'', we try to use a better pivot $\overline{v}$ to get $\overline{p}''$.
- **Strategy IV.** We label a $v_i \in S$ as cycling pivot if it gets to be selected as a pivot with high frequency. If in the course of iterations of the Triangle Algorithm we witness the occurrence of a cycle of pivots with small reduction in the gap, we introduce their corresponding midpoints, or a subset of them, as auxiliary pivots. This will improve their reduction factor in the gap. Rather than introducing too many auxiliary pivots into S, we can first add the centroid of the cycling vertices.

10 Generalized Triangle Algorithm

Let S and p be as before. In this section we describe generalized Triangle Algorithm where an iterate $p' \in conv(S)$ may be replaced with a convex subset P' of $conv(S)$. First, we prove a theorem that gives a general notion of a pivot, a generalization of Theorem 2.

Theorem 18 (Generalized Distance Duality) $p \in conv(S)$, *if and only if for each closed convex subset* P' *of* $conv(S)$ *that does not contain* p *there exists* $v_j \in S$ *such that*

$$d(P', v_j) \geq d(p, v_j). \tag{107}$$

Proof Suppose $p \in conv(S)$. Since $p \notin P'$ and P' is closed, there exists $p' \in P'$ such that $p' = \text{argmin}\{d(p, x) : x \in P'\}$, and $p' \neq p$. Consider the Voronoi cell of p with respect to the two point set $\{p, p'\}$, i.e. $V(p) = \{x \in \mathbb{R}^m : \ d(x, p) < d(x, p')\}$. We claim that there exists $v_j \in V(p) \cap S$. If not, S is a subset of $\overline{V}(p') = \{x \in \mathbb{R}^m : \ d(x, p) \leq d(x, p')\}$. By convexity, $\overline{V}(p')$ contains $conv(S)$. But since $V(p) \cap \overline{V}(p') = \emptyset$, this contradicts that $p \in conv(S)$. Now by convexity of P', (107) is satisfied.

Conversely, suppose that for any convex subset $P' \subset conv(S)$ that does not contain p there exists $v_j \in S$ such that $d(P', v_j) \geq d(p, v_j)$. If $p \notin conv(S)$, let $p' \in conv(S)$ be the closest point to p, $P' = \{p'\}$. Then $d(P', v_i) < d(p, v_i)$ for all i, a contradiction. Thus, $p \in conv(S)$. □

The theorem asserts that the notion of pivot extends to more general convex subsets of $conv(S)$, suggesting a generalized Triangle Algorithm. In Fig. 16 we describe $\triangle_k$-Algorithm. When $k = 2$ the algorithm coincides with the Triangle Algorithm. In each iteration of the algorithm there is a polytope P' that is a convex hull of $t \leq k$ points in $conv(S)$, not containing p, and $p' = \text{argmin}\{d(p, x) : x \in P'\}$ is known. Initially, $P' = \{p'\}$, a single point. If $d(p, p')$ is sufficiently small, it stops. Otherwise, it computes a generalized pivot, $v_j \in S$ such that $d(P', v_j) \geq d(p, v_j)$. To compute such a v_j it suffices to pick a point in $V(p) \cap S$, where $V(p)$ is the Voronoi region of p with respect to p'. It then replaces P' with $conv(P' \cup \{v_j\})$ and updates p' with $p'' = \text{argmin}\{d(p, x) : x \in P'\}$, and repeats the process until $t = k$. It then replaces P' with $\{p'\}$ and repeats the cycle. Thus for $k = 2$ the computation of p'' is as in the Triangle Algorithm itself, finding the nearest point to p over a line segment. For $k = 3$ the computation of p'' in one cycle amounts to finding the nearest point to p, first over a line segment, then over a triangle. For $k \geq 4$ the minimum distance to p is computed, first over a line segment, then over a triangle, then over a quadrangle, and so on.

In practice we wish to keep k reasonably small so as to keep the computation of the nearest points sufficiently simple. As an example Fig. 17 shows one iteration with $k = 3$. In this example, starting with p'_1, two iterations of the Triangle Algorithm result in p'_2, while a single iteration of $\triangle_3$ results in p''_2 which is closer to p.

Theorem 19 *Suppose $p \in conv(S)$. For each $2 \leq k \leq n - 1$, in each iteration $\triangle_{k+1}$-Algorithm produces an approximation to p at least as close as $\triangle_k$-Algorithm. In particular, $\triangle_k$-Algorithm is at least as strong as Triangle Algorithm.* □

$\triangle_k$-Algorithm ($S = \{v_1, \ldots, v_n\}$, p, $\epsilon \in (0, 1)$, $k \in \{2, \ldots, n\}$)

- **Step 0.** (**Initialization**) Let $p' = v = \text{argmin}\{d(p, v_i) : v_i \in S\}$. Let $P' = \{p'\}$. Set $t = 1$.
- **Step 1.** If $t = k$, replace P' with $\{p'\}$. If $d(p, p') < \epsilon d(p, v)$, output p' as ϵ-approximate solution, stop. It there exists $v_j \in S$ satisfying $d(P', v_j) \geq d(p, v_j)$, replace v with v_j. Otherwise, output p' as a witness, stop.
- **Step 2.** Replace P' with $conv(\{P' \cup \{v_j\})$. Replace t with $t + 1$. Compute

$$p'' = \sum_{i=1}^{n} \alpha'_i v_i = \text{argmin}\{d(p, x) : x \in P'\}. \tag{108}$$

  Replace p' with p'', α_i with α'_i, for all $i = 1, \ldots, n$. Go to Step 1.

Fig. 16 $\triangle_k$-Algorithm

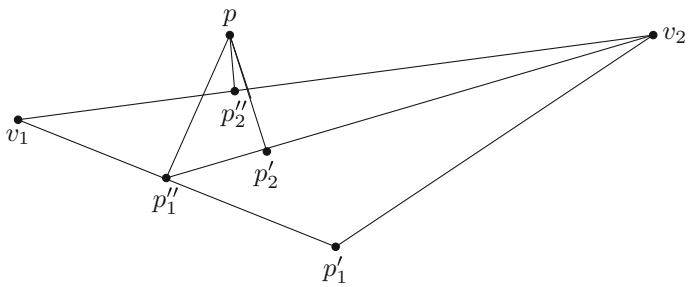

Fig. 17 While two iterations of the Triangle Algorithm would give p'_2, Δ_2-Algorithm would result in p''_2

Remark 9 The computation of the closest point to P' when it is the convex hull of $t+1$ vectors in $\mathbb{R}^m$ is a convex programming over a simplex in dimension t. However, using the constraint $\alpha_1 + \cdots + \alpha_{t+1} = 1$ we can reduce the optimization over a convex quadratic function in t variables over the constraint set $\{\alpha : \alpha_1 + \cdots + \alpha_t \leq 1, \alpha_i \geq 0, i = 1, \ldots, t\}$. In particular, for $t = 2$ it is the minimization of a convex quadratic function over the interval $[0, 1]$. For $t = 3$, the feasible region is the triangle $\{(\alpha_1, \alpha_2) : \alpha_1 + \alpha_2 \leq 1, \alpha_1 \geq 0, \alpha_2 \geq 2\}$. To minimize a convex quadratic function over a triangle we can first minimize it over the three sides, selecting the best value; then compare this value to the unconstrained minimum, assuming that it lies feasible to the triangle, and select the best. The complexity progressively gets harder as t increases. When $k > 2$, an alternative to computing p'' exactly is to find a p-witness in P'. Such witness gives an approximation of $d(p, P')$ to within a factor of two.

For the example when p is at the center of one of the edges of a square (see Fig. 10) the reader can verify that $\triangle_2$-Algorithm solves the problem in one iteration.

11 Solving LP feasibility via the Triangle Algorithm

Consider the LP feasibility problem, testing if Ω is nonempty, where

$$\Omega = \{x \in \mathbb{R}^n : \quad Ax = b, \quad x \geq 0\}, \tag{109}$$

with $A = [a_1, a_2, \ldots, a_n]$, $b, a_i \in \mathbb{R}^m$. It is well known that through linear programming duality, the general LP problem can be reduced to a single LP feasibility problem. In this section we show how the Triangle Algorithm for problem (P) can be modified to either prove that Ω is empty, or to compute an approximate feasible point when the set of recession directions of Ω is empty, where

$$Res(\Omega) = \{d : \quad Ad = 0, \quad d \geq 0, \quad d \neq 0\}. \tag{110}$$

Definition 13 Given ϵ, we shall say $x_0 \in \mathbb{R}^n$ is a ϵ-approximate solution (or feasible point) of Ω if the point Ax_0 lies in Cone$(\{a_1, \ldots, a_n\})$, i.e.

$$Ax_0 \in \{Ax : \quad x \geq 0\}, \tag{111}$$

satisfying

$$d(Ax_0, b) < \epsilon R', \tag{112}$$

where

$$R' = \max\{\|a_1\|, \ldots, \|a_n\|, \|b\|\}. \tag{113}$$

The following is easy to show.

Proposition 7 *Suppose* $0 \notin conv(\{a_1, \ldots, a_n\}) = \{Ax : \sum_{i=1}^{n} x_i = 1, x \geq 0\}$. *Then* $\Omega \neq \emptyset$ *if and only if* $0 \in conv(\{a_1, \ldots, a_n, -b\})$. □

Remark 10 It is easy to see that $Res(\Omega) = \emptyset$ if and only if $0 \notin conv(\{a_1, \ldots, a_n\})$. In particular, if $Res(\Omega) = \emptyset$, Ω is a bounded set, possibly empty. Thus, in this case LP feasibility reduces to a single convex hull decision problem.

In particular, the above implies we can offer a straightforward variation of the Triangle Algorithm to solve the LP feasibility problem:

LP-Feasibility Triangle Algorithm ($A = [a_1, \ldots, a_n]$, b, $\epsilon \in (0, 1)$)

- **Step 0.** (**Initialization**) Let $S = \{a_1, \ldots, a_n, a_{n+1} = -b\})$, $p = 0$. Let $p' = v = \text{argmin}\{\|a_i\| : a_i \in S\}$.
- **Step 1.** Given $p' = \sum_{i=1}^{n+1} \alpha_i a_i \in conv(S)$, if $\|p'\|/\alpha_{n+1} < \epsilon\|v\|$, stop. Otherwise, if there exists a 0-pivot, replace v with a_j. If no 0-pivot exists, then output p' as a 0-witness, stop.
- **Step 2.** Compute the new iterate p'' as the nearest point to p on the line segment $p'a_j$. Replace p' with this point and go to Step 1.

The following theorem establishes the needed accuracy to which an approximate solution in $conv(\{a_1, \ldots, a_n, -b\})$ should be computed.

Theorem 20 (Sensitivity Theorem) *Suppose* $0 \notin conv(\{a_1, \ldots, a_n\})$. *Let*

$$\Delta_0 = \min\left\{\|p\| : \quad p \in conv(\{a_1, \ldots, a_n\})\right\} = \min\left\{\|Ax\| : \quad \sum_{i=1}^{n} x_i = 1, x \geq 0\right\}, \tag{114}$$

Let Δ_0' *be any number such that* $0 < \Delta_0' \leq \Delta_0$. *Let* $b_0 = \|b\|$ *and* R' *as in* (113). *Suppose* $\epsilon > 0$ *satisfies*

$$\epsilon \leq \frac{\Delta_0'}{2R'}. \tag{115}$$

Suppose we have computed the following approximation to 0:

$$p' = \sum_{i=1}^{n} \alpha_i a_i - \alpha_{n+1} b \in conv\Big(\{a_1, \ldots, a_n, -b\}\Big) \tag{116}$$

satisfying

$$\|p'\| < \epsilon R' \tag{117}$$

Let

$$x_0 = \left(\frac{\alpha_1}{\alpha_{n+1}}, \ldots, \frac{\alpha_n}{\alpha_{n+1}}\right)^T. \tag{118}$$

Then, $x_0 \geq 0$*, and if*

$$\epsilon' = 2\left(1 + \frac{b_0}{\Delta_0'}\right)\epsilon, \tag{119}$$

we have

$$d(Ax_0, b) < \epsilon' R', \tag{120}$$

i.e. x_0 *is an* ϵ'*-approximate feasible point of* Ω*.*

Proof Letting $q' = p'/\alpha_{n+1}$, from (116) and (118) we have

$$q' = \frac{p'}{\alpha_{n+1}} = Ax_0 - b. \tag{121}$$

From (117) we have,

$$\|q'\| = d(Ax_0, b) < \frac{\epsilon R'}{\alpha_{n+1}}. \tag{122}$$

We wish to compute a lower bound on α_{n+1}. Let

$$q = \frac{\alpha_1}{1 - \alpha_{n+1}} a_1 + \cdots + \frac{\alpha_n}{1 - \alpha_{n+1}} a_n. \tag{123}$$

Note that $q \in conv(\{a_1, \ldots, a_n\})$. Then, by definition of Δ_0 we have,

$$\|q\| \geq \Delta_0 \geq \Delta_0'. \tag{124}$$

Let

$$q'' = \frac{p'}{1 - \alpha_{n+1}} = q - \frac{\alpha_{n+1}}{1 - \alpha_{n+1}} b. \tag{125}$$

From (117) we also have

$$\|q''\| < \frac{\epsilon R'}{1 - \alpha_{n+1}}. \tag{126}$$

Applying the triangle inequality, $\|u\| - \|v\| \leq d(u, v)$, to (125) and then using the bound in (126) we have

$$\|q\| - \frac{\alpha_{n+1}}{1 - \alpha_{n+1}}\|b\| \leq \|q''\| < \frac{\epsilon R'}{1 - \alpha_{n+1}}. \tag{127}$$

From (127) and (124), and that $\|b\| = b_0$ we get

$$\Delta_0' < \frac{\alpha_{n+1}}{1 - \alpha_{n+1}} b_0 + \frac{\epsilon R'}{1 - \alpha_{n+1}}. \tag{128}$$

Equivalently,

$$\Delta_0' - \epsilon R' < \alpha_{n+1}(\Delta_0' + b_0). \tag{129}$$

From (129) and the assumption in (115) we get

$$\alpha_{n+1} > \frac{\Delta_0' - \epsilon R'}{\Delta_0' + b_0} \geq \frac{\Delta_0'}{2(\Delta_0' + b_0)}. \tag{130}$$

Substituting the lower bound in (130) for α_{n+1} into (122) implies the claimed error bound in (120). □

Theorem 21 *Suppose Ω is nonempty and $Res(\Omega)$ is empty. Given $\epsilon_0 \in (0, 1)$, in order to compute $x_0 \geq 0$ such that $d(Ax_0, b) < \epsilon_0 R'$ it suffices to compute a point $p' \in conv(\{a_1, \ldots, a_n, -b\}$ so that*

$$\|p'\| < \epsilon R', \quad \epsilon \leq \frac{\Delta_0'}{2} \min\left\{\frac{1}{R'}, \quad \frac{\epsilon_0}{(\Delta_0' + b_0)}\right\}, \tag{131}$$

where Δ_0' is any number satisfying $0 < \Delta_0' \leq \Delta_0$. The number of arithmetic operations of the Triangle Algorithm is

$$O\left(mn \min\left\{\frac{R'^2}{\epsilon_0^2 {\Delta_0'}^2}, \frac{1}{c'} \ln \frac{R'}{\epsilon_0 \Delta_0'}\right\}\right), \quad c' = c\left(0, [A, -b], \frac{\epsilon_0 \Delta_0'}{4R'}\right) \geq \frac{\epsilon_0^2 {\Delta_0'}^2}{4R'^2}. \tag{132}$$

□

Proof The upper bound on ϵ in (131) follows from the sensitivity theorem. Since $\Delta_0 \leq R'$, from this upper bound it suffices to pick $\epsilon \leq \Delta_0' \epsilon_0 / 4R'$. Then the claimed complexity for computing an ϵ_0-approximate solution follows from Theorem 15. □

Remark 11 In theory, to estimate the number of needed iterations requires an estimate Δ_0', a lower bound to Δ_0. However, in practice given a prescribed accuracy ϵ_0 we merely need to run the Triangle Algorithm until either we have computed an ϵ_0-approximate solution of Ω, or a p-witness proving that it is empty. Despite this alternative, there is a way to get an estimate of Δ_0 as described in the next remark.

Remark 12 Since Δ_0 is unknown, in practice we can use an estimate. One possible approach is first to try to test if 0 lies in $conv(\{a_1, \ldots, a_n\})$. Assuming that Ω has no recession direction, 0 is not in this convex hull. Thus by applying the Triangle Algorithm we will get a p-witness p' such that $d(p', a_i) < \|p'\|$ for all $i = 1, \ldots, n$. Such a point p' by Corollary 2 will necessary satisfy:

$$\frac{1}{2}\|p'\| \leq \Delta_0 \leq \|p'\|. \tag{133}$$

Thus by solving this auxiliary convex hull decision problem we get a p-witness, p', whose norm can be used as Δ_0'. Next we use p' as the starting iterate as it already lies in the $conv(\{a_1, \ldots, a_n, -b\})$.

Following the above remark we offer a two-phase Triangle Algorithm for solving the feasibility problem in LP with the assumption that $0 \notin conv(\{a_1, \ldots, a_n\})$:

Two-Phase LP-Feasibility Triangle Algorithm ($A = [a_1, \ldots, a_n]$, b, $\epsilon_0 \in (0, 1)$)

- **Phase I.**
  - **Step 1.** Call **Triangle Algorithm**($S = \{a_1, \ldots, a_n\}$, $p = 0$, $\epsilon = 0.5$).
  - **Step 2.** If the output p' is not a witness, replace ϵ with $\epsilon/2$, go to Step 1.
- **Phase II.** Let $\Delta_0' = 0.5\|p'\|$, p' the 0-witness output in Phase I. Set

$$\epsilon_1 = \frac{\Delta_0'}{2} \min\left\{\frac{1}{R'}, \quad \frac{\epsilon_0}{(\Delta_0' + b_0)}\right\}. \tag{134}$$

  Call **Triangle Algorithm** ($S = \{a_1, \ldots, a_n, -b\}$, $p = 0, \epsilon = \epsilon_1$).

From the sensitivity theorem and the complexity theorem, Theorem 15, the complexity of the Two-Phase LP-Feasibility Triangle Algorithm can be stated as

Theorem 22 *Suppose Ω is nonempty and $Res(\Omega)$ is empty. Given $\epsilon_0 \in (0, 1)$, in order to compute an ϵ_0-approximate solution of Ω (i.e. $x_0 \geq 0$ such that $d(Ax_0, b) < \epsilon_0 R'$), it suffices to set $\Delta_0' = 0.5\|p'\|$, where p' is the p-witness computed in Phase I. Then in Phase II it suffices to compute a point $p' \in conv(\{a_1, \ldots, a_n, -b\}$ so that*

$$\|p'\| < \epsilon R', \quad \epsilon \leq \epsilon_1 = \frac{\Delta_0'}{2} \min\left\{\frac{1}{R'}, \quad \frac{\epsilon_0}{(\Delta_0' + b_0)}\right\}. \tag{135}$$

The number of arithmetic operation of Phase I is

$$O\left(mn \min\left\{\frac{R'^2}{\Delta_0^2}, \frac{1}{c_0} \ln \frac{\delta_0}{\Delta_0}\right\}\right), \tag{136}$$

where $c_0 = c(0, A, \Delta_0/R') \geq \Delta_0^2/R'^2$ is visibility factor of 0 with respect to A. The number of arithmetic operation of Phase II is

$$O\left(mn \min\left\{\frac{R'^2}{\Delta_0^2\epsilon_0^2}, \frac{1}{c_1} \ln \frac{\delta_0}{\Delta_0\epsilon_0}\right\}\right), \tag{137}$$

where $c_1 = c(0, [A, -b], \frac{\epsilon_0\Delta_0}{4R'}) \geq (\frac{\epsilon_0\Delta_0}{4R'})^2$ is visibility factor of 0 with respect to $[A, -b]$.

12 Solving general LP feasibility via Triangle Algorithm

Here again we consider the problem of computing ϵ-approximate solution to $\Omega = \{x \in \mathbb{R}^n : Ax = b, \quad x \geq 0\}$, is one exists (see (13)). Whether or not $0 \in conv(\{a_1, \ldots, a_n\})$, it is well known that to test the feasibility of Ω one can safely add the constraint $\sum_{i=1}^n x_i \leq M$, where M is a large enough constant. For integer inputs, such M can be computed, dependent on the size of encoding of A, b, see e.g. Schrijver (1986). In fact it can be shown that M can be taken to be $O(2^{O(L)})$, where L dependents on m, n and logarithm of the absolute value of the largest entry of A or b, see e.g. Kalantari and Emamy-K (1997).

Having such a bound M, by adding a slack variable, x_{n+1} to this constraint, testing the feasibility of Ω is equivalent to testing the feasibility of

$$\Omega_M = \left\{(x, x_{n+1}) \in \mathbb{R}^{n+1} : \quad Ax = b, \quad \sum_{i=1}^{n+1} x_i = M, \quad x \geq 0, \quad x_{n+1} \geq 0\right\}. \tag{138}$$

Dividing the equations in Ω_M by M and setting $y_i = x_i/M, i = 1, \ldots, n+1$, we conclude that testing the feasibility of Ω_M is equivalent to testing the feasibility of

$$\overline{\Omega}_M = \left\{(y, y_{n+1}) \in \mathbb{R}^{n+1} : \quad Ay = \frac{b}{M}, \quad \sum_{i=1}^{n+1} y_i = 1, \quad y \geq 0, \quad y_{n+1} \geq 0\right\}. \tag{139}$$

We call the *augmented* (P) the problem of testing if $p \in conv(\overline{S})$, where

$$p = \frac{b}{M}, \quad \overline{S} = \left\{a_1, \ldots, a_n, a_{n+1}\right\}, \quad a_{n+1} = 0. \tag{140}$$

We may solve the augmented (P) in two different ways, directly by solving a single convex hull decision problem, or by solving a sequence of such problems. We will describe the two approaches and analyze their complexities. First, we prove an auxiliary lemma, an intuitively simple geometric result.

Lemma 2 *Given $u, w \in \mathbb{R}^m$, we have:*

$$\sup\left\{d\left(u, \frac{w}{\mu}\right) : \quad \mu \in [1, \infty)\right\} = \max\left\{d(u, w), \|u\|\right\}. \tag{141}$$

Proof Consider the maximum of the function $f(t) = d^2(u, tw)$ over the interval $t \in [0, 1]$. Since $f(t)$ is convex, its maximum is attained at an endpoint of the interval. Letting $t = 1/\mu$, the proof is complete. □

Theorem 23 *Suppose that a positive number M satisfies the property that Ω is feasible if and only if Ω_M is feasible. Then, solving the augmented (P) to within accuracy of ϵ/M is equivalent to testing the feasibility of Ω to within accuracy of ϵ. More specifically, by applying the Triangle Algorithm to solve the augmented (P)* (see (140))*, in $O(mn \min\{M^2\epsilon^{-2}, M(c''\epsilon)^{-1}\})$ arithmetic operations, where $c'' = c(bM^{-1}, [A, 0], \epsilon) \geq M^2\epsilon^{-2}$, either we compute a p-witness $p' \in conv(\overline{S})$ proving that $b/M \notin \overline{S}$ (hence $\Omega = \emptyset$), or a point $p' \in conv(\overline{S})$ such that for some $i = 1, \ldots, n+1$ we have,*

$$d(b, Mp') < \epsilon \max\left\{d(b, a_i), \|a_i\|\right\} < 2R'\epsilon, \tag{142}$$

where R' is as in (113)*. Equivalently, $p' = Ay_0$, for some $y_0 \geq 0$, and if $x_0 = My_0$, then*

$$d(b, Ax_0) < \epsilon \max\left\{d(b, a_i), \|a_i\|\right\} < 2R'\epsilon. \tag{143}$$

Proof If solving the augmented (P) to accuracy ϵ/M leads to a p-witness $p' \in conv(\overline{S})$ proving that $b/M \notin \overline{S}$, then Ω is empty. Otherwise, by Theorem 9 the algorithm leads to a point $p' \in conv(\overline{S})$ such that for some $i = 1, \ldots, n+1$ we have

$$d\left(\frac{b}{M}, p'\right) < \frac{\epsilon}{M} d\left(\frac{b}{M}, a_i\right). \tag{144}$$

Multiplying the above by M, applying Lemma 2, and since $M \times d(b/M, p') = d(b, Mp')$, we get the proof of the first claimed inequalities in (142) and (143). The bound $2R'\epsilon$ follows from the triangle inequality, $d(b, a_i) \leq \|b\| + \|a_i\|$. The claimed complexity bound follows from Theorem 15. □

Instead of solving the augmented (P) with an a priori estimate for M, we may solve it as a sequence of augmented (P)'s with increasing estimates of M that successively doubles in value. To describe this approach, first consider the following.

Given $\mu > 0$, let $p_\mu = b/\mu$. Select any $p' \in conv(\overline{S})$ as the iterate and let

$$\mu_0 = \sup\left\{\mu : \quad d(p_\mu, a_i) > d(p', a_i), \quad \forall i = 1, \ldots, n+1\right\}. \tag{145}$$

Clearly, $0 < \mu_0 < \infty$. According to Theorem 3, and the definition of μ_0, for any $\mu \in (0, \mu_0)$, p' is a p-witness proving that $p_\mu \notin conv(\overline{S})$. From Lemma 2, we know $\mu_0 \geq 1$. Given $\epsilon > 0$, Set $\mu = \mu_0$ and consider the following iterative step:

Iterative Step. Let $\epsilon_\mu = \epsilon/\mu$. Use the Triangle Algorithm to solve the augmented (P) with $p_\mu = b/\mu$ to either compute $p'_\mu \in conv(\overline{S})$ such that

$$d(p_\mu, p'_\mu) < \epsilon_\mu \max\left\{d(p_\mu, a_i), \|a_i\|\right\}, \quad \text{for some } i, \tag{146}$$

or a p-witness $p'_\mu \in conv(\overline{S})$ proving that $p_\mu \notin S$. In the latter case replace μ with 2μ and repeat.

The following theorem analyzes the complexity of this approach. However, we only consider it according to the first complexity bound. An alternate complexity bound using visibility factor can also be given.

Theorem 24 *Repeating the iterative step, either we compute an ϵ-approximate feasible point of Ω, or a p-witness to the infeasibility of the augmented (P) with $\mu \geq M$. In the latter case Ω is empty. More specifically, if the algorithm requires r iterations of the iterative step, its arithmetic complexity is $O(R'^2 mn 2^{2r} \epsilon^{-2})$. Furthermore, $r \leq \lceil \log_2 M \rceil$.*

Proof Since $\mu_0 \geq 1$, the number of augmented (P)'s to be solved is bounded by $t = \lceil \log_2 M \rceil$. With the initial value of $\mu = \mu_0$ we solve the augmented (P) to either compute an approximate solution in Ω to within accuracy of ϵ, or a p-witness to the infeasibility of Ω_μ. By Theorem 9 this takes $O(\epsilon^{-2})$ arithmetic operations. If Ω_μ is infeasible, we double μ and test the feasibility of new Ω_μ. Then by (46) in Lemma 1, the number of iterations needed to halve the current error that is known to exceed ϵ is bounded by

$$(32 \ln 2)\frac{\overline{R}^2}{\epsilon^2}, \quad \overline{R} = \max\left\{\max\{d(b, a_i), \|a_i\|\} : \quad i = 1, \ldots, n+1\right\} \leq 2R', \tag{147}$$

where the bound $2R'$ is from the fact that $d(b, a_i) \leq \|b\| + \|a_i\|$, Repeating this r times we get the total complexity bounded by

$$\begin{aligned} \overline{K}_\epsilon &\leq (32 \ln 2)\frac{\overline{R}^2}{\epsilon^2}(1 + 2^2 + 2^4 + \cdots + 2^{2r}) \leq (32 \ln 2)\frac{\overline{R}^2}{\epsilon^2} 2^{2r+1} \qquad (148) \\ &= (64 \ln 2)\frac{\overline{R}'^2}{\epsilon^2} 2^{2r+1} = O\left(R'^2 \frac{2^{2r}}{\epsilon^2}\right). \end{aligned}$$

□

13 Concluding remarks and future work

In this article we have described several novel characterization theorems together with a very simple algorithm for the convex hull decision problem (problem (P)), the *Triangle Algorithm*. The Triangle Algorithm can also be considered as an algorithm that tests if a given set of open balls in $\mathbb{R}^m$ that are known to have a common point on their boundary, have a nonempty intersection. The Triangle Algorithm is straightforward to implement and it is quite flexible in the sense that it allows variations to improve practical performance. Its main step is the comparison of distances in identifying a good p-pivot or a p-witness. While the Triangle Algorithm works with distances, it requires only the four elementary operations and comparisons, no square-root operation is required. In the worst-case, each iteration requires $O(mn)$ arithmetic operations. However, when $p \in conv(S)$, the algorithm may be able to find a good pivot by searching a constant number of v_i's. In this case the complexity of each iteration is $O(m+n)$ operations ($O(m)$ to find a pivot and $O(n)$ to update the coefficients in the representation of the pivot as a convex combination of v_i's).

In this article we have also made some theoretical comparisons with such algorithms as the simplex method, Frank–Wolfe method, and first-order gradient algorithms. In Li and Kalantari (2013) we have made computation comparisons with some of these algorithms, indicating that the Triangle Algorithm is quite competitive with these algorithms when solving the convex hull decision problem. We emphasize that while the Triangle Algorithm computes an iterate so that the relative error is within prescribed tolerance, it can produce an approximation with arbitrarily small absolute error, possibly at the cost of a theoretical complexity with a larger constant. Such is the case with all approximation schemes, as well as polynomial-time algorithms. Analogous to polynomial-time algorithms for linear programming, when the input data is integer, a sufficient approximation can be rounded into an exact representation of p as a convex combination of v_i's. When the rank of the matrix of points in S is m, it is possible to represent each iterate as a convex combination of at most $m+1$ of the v_i's. It maybe convenient to maintain such representation within each iteration. While the Triangle Algorithm is designed to solve problem (P), we have analyzed its theoretical applicability in solving LP feasibility, as well as LP optimization.

We anticipate that the simplicity of the algorithm, its theoretical properties, and the distance dualities will lead to new analysis and applications. For instance, computational testing, average case analysis, amortized complexity, analysis of visibility constant for special problems, generalization of the characterization theorems and the Triangle Algorithm, as well as its specializations. In addition to the convex hull decision problem itself, some applications are straightforward. For instance, the Triangle Algorithm when applied to each $p = v_i$ with $S_i = S - \{v_i\}$ gives an approximation algorithm for solving the irredundancy problem. Some other applications are not so straightforward. In fact since the release of the first version of this article, Kalantari (2012a), we have considered several novel applications or extensions of the Triangle Algorithm.

(i) In Kalantari (2012c) we describe the application of the Triangle Algorithm in order to solve a linear system. As such it offers alternatives to iterative methods for solving a linear system, such as Jacobi, Gauss-Sidel, SOR, MOR, and others. Some preliminary testing is given in Gibson and Kalantari (2013). We will make further report on the computational performance of the Triangle Algorithm with such algorithms.

(ii) In Kalantari (2012b) we give a version of the Triangle Algorithm for the case of problem (P) where we wish to locate the coordinates of a point p having prescribed distances to the sites $S = \{v_1, \ldots, v_n\}$. We call this the *ambiguous convex hull problem* since

not only the coordinates of the point are unknown, so is the existence of such point. Despite this ambiguity, a variation of the Triangle Algorithm can solve the problem in $O(mn\epsilon^{-2}\ln\epsilon^{-1})$ arithmetic operations.

(iii): In Kalantari [32] we will give a generalization of the Triangle Algorithm that either separate two compact convex subsets, or computes an approximate point in their intersection. It can also approximate the distance between them when they are distinct, or compute supporting hyperplanes with the largest margin.

In addition to what is stated above, other applications and generalizations of the characterization theorems, as well as the Triangle Algorithm itself are possible and will be considered in our future work. These include, specialization of the Triangle Algorithm to LP with special structures, combinatorial and graph optimization problems. For instance, in another forthcoming article we will analyze the Triangle Algorithm for the cardinality matching in a bipartite graph, showing that it exhibits polynomial-time complexity. As such, the algorithm is very different than all the existing polynomial-time algorithms for this classic problem. As generalizations, it is possible to state problem (P) and the corresponding characterizations when the set S consists of one or a finite number of compact subsets of $\mathbb{R}^m$, e.g. polytopes, balls, or more general convex bodies. Moreover, problem (P) can also be defined over other cones. For instance, a canonical problem in semidefinite programming described in Kalantari (2003) is an analogue of problem (P) over the cone of positive semidefinite matrices. Among other interesting research topics, the computation of visibility constant such as ν^* (see (70)) and ν_* (see (71)), and λ_* (see (97)) for some special cases of the convex hull problem could result in proving very efficient or polynomial-time complexity.

An anonymous reviewer brought to our attention Dantzig (1992) who gives an analysis of the so-called *von Neumann* algorithm. The von Neumann algorithm is in fact identical with Frank–Wolfe algorithm (when applied to the minimization of a convex quadratic function over a simplex). Connections between Frank–Wolfe, Gilbert's, and sparse greedy algorithms are already discussed in the article, as are the differences between the Triangle Algorithm and Frank–Wolfe algorithm. Dantzig (1992) derives an $O(\epsilon^{-2})$ complexity bound on the number of iterations of the von Neumann algorithm with a different analysis and constant than those given in this article. Under the assumption that a ball centered at the given point p is contained in the convex hull, Dantzig's analysis (see also Epelman and Freund 1997) gives a complexity bound that is logarithmic in $1/\epsilon$. Our independent analysis in the article has shown that this logarithmic complexity is possible using any strict pivot whose existence is proved via our novel strict distance duality, given any iterate that is not already a witness. The Triangle Algorithm is based on a geometric notion while von Neumann (Frank–Wolfe) algorithm is algebraically motivated. These algorithms may coincide on some iterations, however the Triangle Algorithm is a different algorithm where its geometric foundation and the distance dualities offer many theoretical and algorithmic applications some of which have already been explored in this article. These dualities give insights on the nature of the geometry of the convex hull problem. For instance, the Triangle Algorithm has given rise to the characterization of witness set any of whose members give an approximation of distance of the point to the convex hull to within a factor of two. Our proofs of complexities are very simple, basically using high school geometry. Our preliminary computational analysis show the differences and advantages over Frank–Wolfe algorithm, see Li and Kalantari (2013) where we also make a computational comparison to the simplex method. The geometric foundation of the Triangle Algorithm also gives rise to the derivation of new complexity bounds as well as generalization of the algorithm to much more complicated cases of the convex hull problem. For instance, in Kalantari and Saks (2013) we show that with a preprocessing that consists

of computing the pairwise distances between the given points, each iteration of the Triangle Algorithm can be implemented in $O(m+n)$ time, a considerable improvement over $O(mn)$ complexity. In Kalantari [32], based on a generalization of the Triangle Algorithm, we give an algorithmic version of the classical separating hyperplane theorem for compact convex sets. We will show additional advantages, applications and generalizations of the Triangle Algorithm in our future work. In Kalantari (2012c) we use the sensitivity theorem proved here to give new iterative algorithms for solving a linear system. Some preliminary computational results on solving linear systems are already done in Gibson and Kalantari (2013) showing competitive performance with existing iterative methods. The Triangle Algorithm is a young algorithm. It needs time to be appreciated, the chance to be tested out widely and compared to other algorithms, on more extensive sets of data, and by other researchers and practitioners to verify for themselves. Finally, in the age of Big Data we feel the Triangle Algorithm will find many new practical applications.

Acknowledgments I thank Iraj Kalantari for a discussion on the use of Voronoi diagram argument to prove Theorem 1, resulting in a simpler geometric proof than the one given in an earlier version of this article. I thank Tong Zhang for a discussion regarding the Greedy Algorithm for general convex minimization over a simplex. I also thank Günter Rote for bringing to my attention the work of Kuhn (2011) and Durier and Michelot (1986) (Remark 4). I thank a referee for bringing to my attention Dantzig (1992) technical report on analysis of von Neumann algorithm. Also, when the article was being refereed Mike Grigoriadis brought to my attention Dantzig's analysis in Epelman and Freund (1997). I thank a second anonymous referee for his comments.

References

Asano, T., Matoušek, J., & Tokuyama, T. (2007). Zone diagrams: Existence, uniqueness, and algorithmic challenge. *SIAM Journal on Computing*, *37*, 1182–1198.

Aurenhammer, F. (1991). Voronoi diagrams—a survey of fundamental geometric data structure. *ACM Computing Surveys*, *23*, 345–405.

Berkowitz, R., Kalantari, B., Memendendez, D., & Kalantari, I. (2012). On properties of forbidden zone of polyhedrons and polytopes. In *Proceedings of the ninth annual international symposium on voronoi diagrams in science and engineering*, pp. 56–65.

Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*, 121–167.

Chan, T. M. (1996). Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete & Computational Geometry*, *16*, 369–387.

Chazelle, B. (1993). An optimal convex hull algorithm in any fixed dimension. *Discrete & Computational Geometry*, *10*, 377–409.

Chvátal, V. (1983). *Linear programming*. New York: W.H. Freeman and Company.

Clarkson, K. L. (1988). Las Vegas algorithms for linear and integer programming when the dimension is small. In: *Proceedings of 29th annual IEEE symposium on foundations of computer science*, pp. 452–456.

Clarkson, K. L. (1994). More output-sensitive geometric algorithm. In *Proceedings of the 35-th annual IEEE symposium on foundations of computer science*, pp. 695–702.

Clarkson, K. L. (2008). Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. In *SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on discrete algorithms* (pp. 922–931). Philadelphia: Society for Industrial and Applied Mathematics.

Dantzig, G. B. (1963). *Linear programming and extensions*. Princeton, NJ: Princeton University Press.

Dantzig, G. B. (1992). *An ϵ-precise feasible solution to a linear program with a convexity constraint in $1/\epsilon^2$ iterations independent of problem size*. Technical Report, Stanford University.

de Biasi, S. C., Kalantari, B., & Kalantari, I. (2010). Maximal zone diagrams and their computation. In *Proceedings of the seventh annual international symposium on voronoi diagrams in science and engineering*, pp. 171–180.

de Biasi, S. C., Kalantari, B., & Kalantari, I. (2011). Mollified zone diagrams and their computation. In *Transactions on computational science XIV: Lecture notes in computer science* (Vol. 6970, pp. 31–59).

Durier, R., & Michelot, C. (1986). Sets of efficient points in a normed space. *Journal of Mathematical Analysis and Applications*, *117*, 506–528.
Dyer, M. E., & Frieze, A. M. (1989). A randomized algorithm for fixed-dimensional linear programming. *Mathematical Programming*, *44*, 203–212.
Epelman, M., & Freund, R. M. (1997). *Condition number complexity of an elementary algorithm for resolving a conic linear system*, Technical Report. http://dspace.mit.edu/bitstream/handle/1721.1/5254/or-319-97-37692575?sequence=1.
Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, *3*, 95–110.
Gärtner, B., & Jaggi, M. (2009). Coresets for polytope distance. In *Symposium on computational geometry*, pp. 33–42.
Gibson, T., & Kalantari, B. (2013). Experiments with the triangle algorithm for linear systems, 2-page Extended Abstract, *23nd annual fall workshop on computational geometry*, City College of New York.
Gilbert, E. G. (1966). An iterative procedure for computing the minimum of a quadratic form on a convex set. *SIAM Journal on Control*, *4*, 61–80.
Goodman, J. E., & O'Rourke, J. (Eds.). (2004). *Handbook of discrete and computational geometry, 2nd Edn (Discrete Mathematics and Its Applications)*. Boca Raton: Chapman & Hall.
Har-Peled, S., Roth, D., & Zimak, D. (2007). *Maximum margin coresets for active and noise tolerant learning*. IJCAI.
Jin, Y., & Kalantari, B. (2006). A procedure of Chvátal for testing feasibility in linear programming and matrix scaling. *Linear Algebra and its Applications*, *416*, 795–798.
Kalai, G. (1992). A subexponential randomized simplex algorithm. In *Proceedings of the 24-th annual ACM symposium on theory of computing*, pp. 475–482.
Kalantari, B. (2014). *An algorithmic separating hyperplane theorem and its applications*.
Kalantari, B. (1990). Canonical problems for quadratic programming and projective methods for their solution. *Contemporary Mathematics*, *114*, 243–263.
Kalantari, B. (2003). Semidefinite programming and matrix scaling over the semidefinite cone. *Linear Algebra and its Applications*, *375*, 221–243.
Kalantari, B. (2012a). A characterization theorem and an algorithm for a convex hull problem. arXiv:1204.1873v2.
Kalantari, B. (2012b). Finding a lost treasure in convex hull of points from known distances. In *The Proceedings of the 24th Canadian conference on computational geometry*, pp. 271–276.
Kalantari, B. (2012c). Solving linear system of equations via a convex hull algorithm. arXiv:1210.7858v1.
Kalantari, B., & Emamy-K, M. R. (1997). On linear programming and matrix scaling over the algebraic numbers. *Linear Algebra and its Applications*, *262*, 283–306.
Kalantari, B., & Saks, M. (2013). On the triangle algorithm for the convex hull membership, 2-page Extended Abstract, *23nd annual fall workshop on computational geometry*, City College of New York.
Karmarkar, N. (1984). A new polynomial time algorithm for linear programming. *Combinatorica*, *4*, 373–395.
Kelner, J. A., & Spielman, D. A. (2006). A randomized polynomial-time simplex algorithm for linear programming, In *Proceedings of the 38th annual ACM symposium on theory of computing*, pp 51–60.
Khachiyan, L. G. (1979). A polynomial algorithm in linear programming. *Doklady Akademia Nauk SSSR*, *244*, 1093–1096 (translated in Soviet Mathematics Doklady 20, 191–194, 1979).
Khachiyan, L., & Kalantari, B. (1992). Diagonal matrix scaling and linear programming. *SIAM Journal on Optimization*, *4*, 668–672.
Klee, V., & Minty, G. J. (1972). How good is the simplex algorithm? In O. Shisha (Ed.), *Inequalities III* (pp. 159–175). London: Academic Press.
Kuhn, H. W. (2011). On a pair of dual nonlinear programs. In J. Abadie (Ed.), *Nonlinear programming* (pp. 37–54). New York: Wiley, 1967, see also, Locational problems and mathematical programming, in Mathematical Optimiation in Economics C.I.M.E. Summer Schools, Vol. 38, pp. 57–76.
Lan, G., Lu, Z., & Monteiro, R. D. C. (2011). Primal-dual first-order methods with $O(1/\epsilon)$ iteration-complexity for cone programming. *Mathematical Programming Series A*, *126*, 1–29.
Li, M., & Kalantari, B. (2013). Experimental study of the convex hull decision problem via a new geometric algorithm, 2-page Extended Abstract, *23nd annual fall workshop on computational geometry*, City College of New York.
Matoušek, J. (1993). Linear optimization queries. *Journal of Algorithms*, *14*, 432–448.
Matoušek, J., Sharir, M., & Welzl, E. (1992). A subexponential bound for linear programming. In *Proceedings of the 8th annual ACM symposium on computational geometry*, pp. 1–8.
Megiddo, N. (1984). Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, *31*, 114–127.
Motawani, R., & Raghavan, R. (1995). *Randomized algorithms*. Cambridge: Cambridge University Press.

Nesterov, Y. E. (2013). Private communications.
Nesterov, Y. E. (2005). Smooth minimization of nonsmooth functions. *Mathematical Programming*, *103*, 127–152.
Preparata, F. P., & Shamos, M. I. (1985). *Computational geometry: An introduction*. New York: Springer.
Rockafellar, R. T. (1997). *Convex analysis*. Princeton, New Jersey: Princeton University Press. [1970].
Schrijver, A. (1986). *Theory of linear and integer programming*. New York, Chichester, Brisbane, Toronto and Singapore: Wiley.
Seidel, R. (1991). Small-dimensional linear programming and convex hulls made easy. *Discrete Computational Geometry*, *6*, 423–434.
Shamir, R. (1987). The efficiency of the simplex method: A survey. *Management Science*, *33*, 301–334.
Sharir, M., & Welzl, E. (1992). A combinatorial bound for linear programming and related problems. In *Proceedings of the 9th symposium on theoretical aspects of computer science*, pp. 569–579.
Todd, M. J. (2002). The many facets of linear programming. *Mathematical Programming*, *91*, 417–436.
Zhang, T. (2003). Sequential greedy approximation for certain convex optimization problems. *IEEE Transactions on Information Theory*, *49*, 682–691.
Zimak, D. A. (2006). Algorithms and Analysis for Multi-Category Classification, Ph.D. thesis, Department of Computer Science, University of Illinois at Urbana-Champaign.