Randomized Triangle Algorithms for Convex Hull Membership

Bahman Kalantari[*]

Abstract

The *triangle algorithm* introduced in [6], tests if a given $p \in \mathbb{R}^m$ lies in the convex hull of a set S of n points in $\mathbb{R}^m$. It computes $p' \in conv(S)$ such that either $d(p', p)$ is within prescribed tolerance, or p' certifies $p \notin conv(S)$. In order to improve its performance, we propose two randomized versions. Bounds on their expected complexity is identical with deterministic bounds. One is inspired by the *chaos game* and a property of resulting Sierpinski triangle.

1 Introduction

Given a finite set $S = \{v_1, \dots, v_n\} \subset \mathbb{R}^m$, and a distinguished point $p \in \mathbb{R}^m$, the *convex hull membership problem* (or *convex hull decision problem*) is to test if $p \in conv(S)$, the convex hull of S. Given a desired tolerance $\varepsilon \in (0, 1)$, we call a point $p_\varepsilon \in conv(S)$ an ε-approximate solution if $d(p_\varepsilon, p) \le \varepsilon R$, where $R = \max\{d(p, v) : \ v \in S\}$, and $d(\cdot, \cdot)$ denotes the Euclidean distance. The convex hull membership problem is the most basic of the convex hull problems. Nevertheless, the convex hull membership problem is a fundamental problem in computational geometry and linear programming and finds applications in statistics, approximation theory, and machine learning. Problems related to the convex hull membership include, computing the distance from a point to the convex hull of a finite point set, support vector machines (SVM) and more, see e.g. Clarkson [1], Gilbert [5], Gärtner and Jaggi [3].

A recent algorithm for the convex hull membership problem is the *triangle algorithm* [6]. It either computes an ε-approximate solution, or a point that proves $p \notin conv(S)$ while also approximating the distance from p to $conv(S)$ to within a factor of 2. For applications in solving linear programming, linear systems, variations and experimental studies, see [6] - [11]. Based on preliminary experiments, the triangle algorithm performs quite well when compared with existing algorithms, see [11], [4]. When p is close to the boundary of $conv(S)$ it may experience zig-zagging in achieving high accuracy approximations. In [6] we have described several strategies to remedy this, such as adding new auxiliary points.

In this article we propose two randomized versions of the triangle algorithm. We first review the triangle algorithm, its relevant properties as well as its worst-case time complexities. Next, we describe *Greedy-Randomized Triangle Algorithm.* Finally, we describe what we call the *Sierpinski-Randomized Triangle Algorithm* inspired by the chaos game, see Devaney [2], which results in the Sierpinski triangle.

We review some terminology and results from [6]. Given $p' \in conv(S)$, $v \in S$ is a *pivot* relative to p at p' (or p-pivot, or simply pivot) if $d(p', v) \ge d(p, v)$. Given $p' \in conv(S)$, $v \in S$ is a *strict pivot* relative to p at p' (or *strict* p-pivot, or simply *strict* pivot) if $\theta = \angle p'pv \ge \pi/2$. A point $p' \in conv(S)$ is a p-*witness* (or a *witness*) if $d(p', v) < d(p, v)$, for all $v \in S$. A witness has the property that the orthogonal bisecting hyperplane to the line segment pp' separates p from $conv(S)$.

Theorem 1. (Distance Duality [6]) *$p \in conv(S)$ if and only if for any $p' \in conv(S)$, there exists a pivot.*

Theorem 2. (Strict Distance Duality [6]) *Assume $p \notin S$. Then $p \in conv(S)$ if and only if for any $p' \in conv(S)$, there exists a strict pivot.*

Given $p' \in conv(S)$, we compute a pivot v, if it exists, then $p'' = nearest(p; p'v)$, the nearest of p on the line segment $p'v$. Next, replace p' with p'' and repeat. Each iteration takes $O(mn)$ operations.

Theorem 3. ([6]) *If $p \in conv(S)$, the number of arithmetic operations of the triangle algorithm to compute p_ε so that $d(p, p_\varepsilon) \le \varepsilon R$ is $O(\frac{mn}{\varepsilon^2})$.*

Theorem 4. ([6]) *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.*

[*]Department of Computer Science, Rutgers, State University of New Jersey. Email: kalantari@cs.rutgers.edu

Given $\varepsilon \in (0,1)$, suppose the triangle algorithm uses a strict pivot in each iteration. The number of arithmetic operations to compute $p_\varepsilon \in conv(S)$ so that $d(p_\varepsilon, p) < \varepsilon R$ is $O(mn(\frac{R}{\rho})^2 \ln \frac{1}{\varepsilon})$.

Remark 1. When $\Delta = d(p, conv(S)) > 0$, setting $\varepsilon R \geq \Delta$ in Theorem 3 gives the complexity to compute a witness.

2 Greedy Randomization

In this section we describe a randomized algorithm we call *Greedy-Randomized Triangle Algorithm.* Given an iterate p', it computes a pivot v, if it exists. Then it randomly selects the new iterate as the midpoint of p' and v, or $nearest(p; p'v)$. It records the closest known point to p as p_*, *the current incumbent candidate*, and updates it when necessary.

Theorem 5. ([10]) *Bound on expected complexity of Greedy-Randomized Algorithm to compute an approximate solution coincide those of Theorems 3, 4.* $\square$

3 Randomization and Chaos

For three points placed as vertices of an equilateral triangle, the *chaos game* it is known to results in one of the most famous fractals of all, the *Sierpinski triangle.* Devaney [2] on chaos game write, "*its multitude of variations provides a wonderful opportunity to combine elementary ideas from geometry, linear algebra, probability, and topology with some quite contemporary mathematics.*"

Definition 1. (General Chaos Game) Given $S = \{v_1, \dots, v_n\} \subset \mathbb{R}^m$, let $\Sigma(S)$ be defined according to the following process: Start with a seed $p' \in conv(S)$, and with probability $1/n$ randomly select $v \in S$, then record $(p'+v)/2$ as a new point and place it in $\Sigma(S)$. Replace p' with $(p'+v)/2$ and repeat.

The *Sierpinski-Randomized Triangle Algorithm* is inspired by the chaos game, but it keeps track of the current incumbent candidate, p_*, the closest known point to p. Given an iterate p', it randomly (equal probability) selects $v \in S \cup \{p_*\}$. If $v \neq p_*$ and not a pivot, the next iterate is $(p'+v)/2$. If $v \neq p_*$, and a pivot, it randomly either replaces p' with $(p'+v)/2$, or with $nearest(p, p'v)$. When $v = p_*$, it searches for a pivot v' at p_*. When such a pivot exists, the next iterate is the $nearest(p, p_* v')$. Except for this case, other cases take $O(m+n)$ operations.

Lemma 1. ([10]) *The expected number of arithmetic operations in each iteration of the Sierpinski-Randomized Triangle Algorithm is $O(m+n)$.*

Theorem 6. ([10]) *Bound on expected complexity of Sierpinski-Randomized Algorithm to compute an approximate solution coincide those of Theorems 3, 4.* $\square$

Concluding Remarks. The proposed algorithms may well improve the iteration complexity of the triangle algorithm, and dependence on ε. For example, in the Sierpinski-Randomized Triangle Algorithm at each iteration, as the current incumbent candidate p_* gets close to p, the chances are good that when an iterate p' randomly selects $v = p_*$ that p_* is actually a pivot at p'. Hence with probability $1/2$ the next iteration will get closer to p. To check if p_* is a pivot at p' takes $O(m+n)$ time as opposed to $O(mn)$ time. Computational experimentations will be carried out and reported in future work.

Acknowledgements. I thank Mike Saks for a discussion regarding randomization.

References

[1] K. L. Clarkson, Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. In SODA '08: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, 922 - 931.

[2] R. L. Devaney, Chaos Rules!, *Math Horizons*, (2004), 11-14.

[3] B, Gärtner and M. Jaggi, Coresets for polytope distance, Symposium on Computational Geometry (2009), 33 - 42.

[4] T. Gibson and B. Kalantari, Experiments with the triangle algorithm for linear systems, 2-page Extended Abstract, 23nd Annual Fall Workshop on Computational Geometry, City College of New York, 2013.

[5] E. G. Gilbert, An iterative procedure for computing the minimum of a quadratic form on a convex set, *SIAM Journal on Control*, 4 (1966), 61 - 80.

[6] B. Kalantari, A characterization theorem and an algorithm for a convex hull problem, to appear in *Annals of Operations Research*, available online August, 2014.

[7] B. Kalantari, Finding a lost treasure in convex hull of points from known distances. In the Proceedings of the 24th Canadian Conference on Computational Geometry (2012), 271 - 276.

[8] B. Kalantari, Solving linear system of equations via a convex hull algorithm, arxiv.org/pdf/1210.7858v1.pdf, 2012.

[9] B. Kalantari and M. Saks, 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, 2013.

[10] B. Kalantari, Randomized triangle algorithms for convex hull membership, http://arxiv.org/pdf/1410.3564v1.pdf, 2014.

[11] M. Li and B. Kalantari, 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, 2013.