Experimental Study of The Convex Hull Decision Problem Via a New Geometric Algorithm

Meng Li* Bahman Kalantari*

Abstract

Given a set S of n points and a point p in m-dimensional Euclidean space, the *convex hull decision problem* is to test if $p \in conv(S)$, the convex hull of S. A simple geometric iterative method, the *Triangle algorithm*, is described for the problem in [5]. In this paper, we describe the experimental performance of the Triangle algorithm on some large problems. We also compare its performance to two well-known algorithms for solving the convex hull problem, the Simplex method and the Frank-Wolfe algorithm. Our results shows that the Triangle algorithm outperforms these algorithms, especially when n is much larger than m.

1 Introduction

Given a set $S = \{v_1, \ldots, v_n\} \subset \mathbb{R}^m$ and a point $p \in \mathbb{R}^m$, testing if $p \in conv(S)$, the convex hull of S, is called the *convex hull decision problem*. It is a fundamental problem in computational geometry and linear programming. One application is the *irredundancy problem*, the problem of computing all the vertices of $conv(S)$, see [3]. It can be formulated as a special case of linear programming (LP) feasibility problem: testing if $Ax = p, e^T x = 1, x \geq 0$ has a solution, where the column vectors of A are $v_1, \ldots, v_n$ and e is the vector of ones. One can convert an LP feasibility to a convex hull decision problem. Furthermore, it is well known that through LP duality theory the general LP problem may be cast as a single LP feasibility problem, see e.g. Chvátal [2]. Hence the significance of convex hull decision problem in LP.

To solve the convex hull decision problem, a simple and natural geometric method, the *Triangle algorithm*, has been proposed in [5]. It is described in Figure 1. In each iteration of the Triangle algorithm we have a current approximation $p' \in conv(S)$. Using this, we select a *pivot* $v_j \in S$, i.e. $d(p', v_j) \geq d(p, v_j)$, where $d(\cdot, \cdot)$ is the Euclidean distance. Then approaches p greedily along the direction from current iterate point to the pivot.

In the next section, we discuss the implementation of the Triangle algorithm and comparison of its performance with the Simplex method and the Frank-Wolfe algorithm, a gradient decent method when applied to a convex quadratic function over a simplex, see [4], [1]. According to our computational results the Triangle algorithm has better performance than the other two when the number of points n is much larger than the dimension m.

Input: $S = \{v_1, \ldots, v_n\}$, a point p and a tolerance ϵ
Output: α, the convex combination of $\{v_1, \ldots, v_n\}$ for p if $p \in conv(S)$ or **No**, if $p \notin conv(S)$
$R = \max_{j \in \{1,\ldots,n\}} d(p, v_j), k = 0;$
Arbitrarily choose initial point $p_0 \in conv(S)$ and corresponding α_0;
while $(d(p, p_k) > \epsilon R)$ **do**
  find v_j such that $d(p, v_j) \leq d(p_k, v_j)$;
  if *no such* v_j **then**
    Output **NO** and halt;
  else
    compute p_{k+1}, the projection of p on line segment $p_k v_j$, update α_{k+1};
    $k = k + 1$;
  end if
end while
Output α_k;

Figure 1: Triangle algorithms

2 The experiments

From now on, we let A to be the matrix whose column vectors are $\{v_1, \ldots, v_n\}$, e is the vector of ones, and p the query point. Frank-Wolfe algorithm for the convex hull decision problem is quite straightforward, it is based on formulating it as the following quadratic programming:

$$\min\{f(x) = (Ax - p)^T (Ax - p) : e^T x = 1, x \geq 0\}$$

In each iteration, Frank-Wolfe uses $O(mn)$ operations to compute the gradient of f and chooses a direction to progress, using partial derivatives. In contrast, Triangle algorithm does not necessarily go through all the column vectors. It moves to next iteration as long as it finds a "good" v_j by checking the angle $\angle p p_k v_j$. Therefore, Triangle algorithm usually spends much less operations than Frank-Wolfe in each iteration.

To apply the Simplex method on convex hull decision problem, consider the following LP, minimizing the sum of the slack variables, $e^T s + s_{m+1}$ subject to the constraints

$$Ax + s = p, \quad e^T x + s_{m+1} = 1, \quad (x, s, s_{m+1}) \geq 0.$$

By computing $Ax/(1 - s_{m+1})$ at each iteration of the Simplex method, we can associate a point $p_k \in conv(S)$ and

*Department of Computer Science, Rutgers, State University of New Jersey. Email:{ ml910, kalantari } @cs.rutgers.edu

view it as an iterative method for testing if $p \in conv(S)$, generating a finite sequence of points, assuming no cycling, whose last iteration is p, if $p \in conv(S)$. The Simplex method spends $O(m^2 + mn)$ arithmetic operations in each iteration, more expensive than the other two algorithms.

To compare the three algorithms, we implemented them in Matlab. In the experiment, we randomly and uniformly generated n points to form S, and a query point p, all in an m-dimensional unit ball, giving a very dense matrix. We set $\epsilon = 10^{-4}$. Each test case was repeated 100 times. The result is shown in Figure 2 and Figure 3.

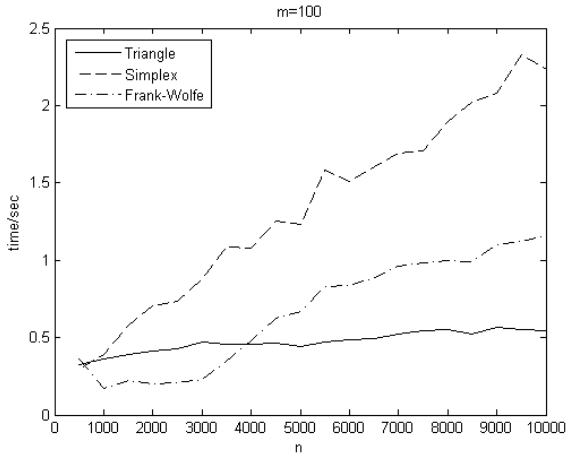

Figure 2: Running time comparison as n grows

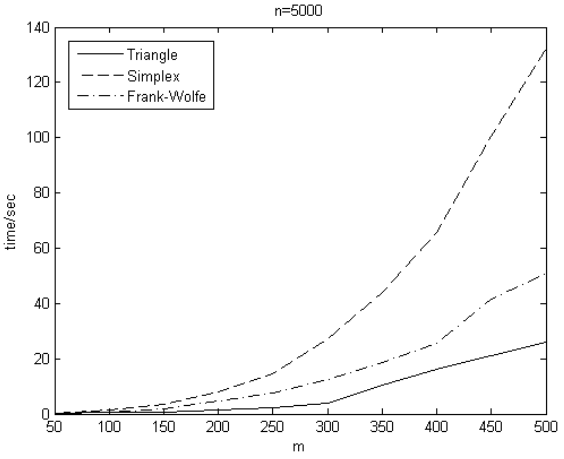

Figure 3: Running time comparison as m grows

As we can see in Figure 2, when the number of points grows, the running time of the Simplex and Frank-Wolfe methods increase while the Triangle algorithm performs very well with only a slight increase in the running time. This can be explained by the fact that the Triangle algorithm does not need to visit all the n points and thus spends less time than the other two in each iteration. Moreover, the Triangle algorithm has a better chance for finding a "good" pivot v_j to approach p efficiently when n increases as shown in Table 1. Therefore the overall iterations will not increase dramatically and the performance becomes better. On the other hand, fewer points in the space would make the situation worse. This can be shown in Figure 3. The increase in m makes the n points distribute in a much more sparse fashion. This would cause the difficulty for computing a good pivot point v_j and thus increases the number of iterations, as shown in Table 2. However, the Triangle algorithm is still the best one among the three algorithms with less running time and less iterations. Triangle algorithm exhibits very good performance when n is larger than m. But while intuitively surprisingly, when $m \geq n$, Triangle algorithm would perform much slower. Consider $m = n = 100$, it turns out that the average of running time and iterations of Triangle algorithm is 13.06 sec and 7573.6 while the Simplex method only needs 0.5122 sec running time and 148.23 iterations. But if one adds many additional points, say 900 auxiliary points in the convex hull, it will improve the performance of the Triangle algorithm as we can see in Figure 2.

n	# of points visited per iteration	iterations
500	185	459.6
1000	228.26	479.6
3000	240.37	540.4
5000	242.22	541.6
10000	254.84	535.4

Table 1: The performance of Triangle algorithm,m=100

m	Triangle	Simplex	Frank-Wolfe
100	539	234.2	251.4
200	531.6	633.8	786
300	560.4	1103.6	1408.2
400	738.8	1649.2	2257.2
500	989.2	2149.2	3576.8

Table 2: The number of iterations when n=5000

In conclusion, the Triangle algorithm does well when the number of points n is much larger than m. Furthermore, it is more efficient than the Simplex method and the Frank-Wolfe algorithm, both in running time and in the number of iterations. In general the Triangle algorithm is a good choice of an algorithm when the size of data set is large.

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] V. Chvátal, *Linear Programming*, W.H. Freeman and Company, New York, 1983.

[3] J. E. Goodman, J. O'Rourke (Editors), *Handbook of Discrete and Computational Geometry*, 2nd Edition (Discrete Mathematics and Its Applications) 2004, Chapman & Hall Boca Raton.

[4] M. Frank and P. Wolfe, An algorithm for quadratic programming, *Naval Res. Logist. Quart.*, 3 (1956), 95 - 110.

[5] B. Kalantari, A characterization theorem and an algorithm for a convex hull problem, arxiv.org/pdf/1204.1873v3.pdf, 2013.