ISSN 1392-8619 print/ISSN 1822-3613 online
ŪKIO TECHNOLOGINIS IR EKONOMINIS VYSTYMAS
TECHNOLOGICAL AND ECONOMIC DEVELOPMENT OF ECONOMY
http://www.tede.vgtu.lt
2006, Vol XII, No 4, 341–346

A PRIORI FILTRATION OF POINTS FOR FINDING CONVEX HULL

Laura Vyšniauskaitė[1], Vydūnas Šaltenis[2]

Vilnius Pedagogical University, Studentų g. 39, LT-08106 Vilnius, Lithuania
E-mails: [1]valaura22@yahoo.com; [2]saltenis@ktl.mii.lt

Received 15 June 2006; accepted 20 November 2006

Abstract. Convex hull is the minimum area convex polygon containing the planar set. By now there are quite many convex hull algorithms (Graham Scan, Jarvis March, QuickHull, Incremental, Divide-and-Conquer, Marriage-before-Conquest, Monotone Chain, Brute Force). The main attention while choosing the algorithm is paid to the running time. In order to raise the efficiency of all the algorithms an idea of a priori filtration of points is given in this article. Besides, two new algorithms have been created and presented. The experiment research has shown a very good efficiency of these algorithms.

Keywords: convex hull, a priori filtration of points, efficiency, Graham Scan, Jarvis March, Quickhull.

1. Introduction

Convex hull is the minimum area convex polygon containing the planar set (Fig 1).

The definition of the convex hull which is given above is for planar points. The convex hull of a set of points S in n dimensions is the intersection of all convex sets containing S. For N points $p_1, p_2, ..., p_N$ the convex hull CH is given by the expression [1]:

$$CH \equiv \left\{ \sum_{j=1}^{N} \lambda_j p_j,\ \ \lambda_j \geq 0 \quad \text{for all } j \text{ and } \sum_{j=1}^{N} \lambda_j = 1 \right\}.$$

Finding the convex hull of a set of points is one of the main problems in computational geometry and computer graphics.

In robotics the convex hull is central to path planning and collision avoidance task. In pattern recognition and image processing the convex hull appears in clustering, and computing similarities between sets. In computational geometry, the convex hull is often a valuable tool in devising efficient algorithms for a number of seemingly unrelated problems [2].

Convex hull also serves as a first preprocessing step to many, if not most, geometric algorithms. For example, consider the problem of finding the diameter of a set of points, which is the pair of points a maximum distance apart. The diameter will always be the distance between two points on the convex hull. So the algorithm for computing diameter proceeds by first constructing the convex hull, then for each hull vertex finding which other hull vertex is farthest away from it [3].

Only a small part of implementation cases of convex hull has been mentioned. So it is no wonder that much work has been done creating algorithms of the convex hull.

We can find these algorithms in scientific press by now: Graham Scan, Jarvis March, QuickHull, Incremental, Divide-and-Conquer, Marriage-before-Conquest, Monotone Chain, Brute Force.

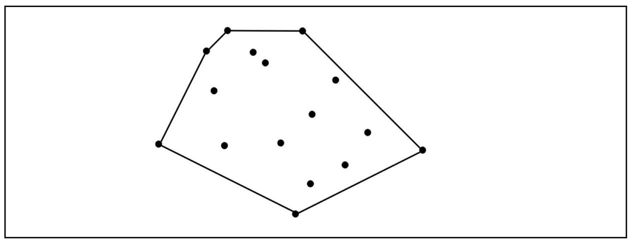
Fig 1. The convex hull for a random set of points

2. Graham Scan

The Graham scan algorithm is often cited as the first "computational geometry" algorithm. The algorithm works in three phases [4]:

1. This algorithm starts by finding the lowest point.

This point will be the pivot, and is guaranteed to be on the hull.

2. All the remaining points are sorted by their polar angle with respect to the low point and the *x*-axis (Fig 2).
3. The algorithm then goes counter-clockwise connecting points "around the circle", but if it has to make a right turn, then it backs up and skips that point (Fig 3).

This algorithm and its implementation has been covered in great detail by O'Rourke [5].

3. Jarvis March

This is perhaps the most simple-minded algorithm for the convex hull, and yet in some cases it can be very fast. The basic idea is as follows [6]:

1. This algorithm starts by finding the lowest point, which is guaranteed to be on the hull.
2. At each step, test each of the points, and find the one which makes the largest right-hand turn. That point has to be the next one on the hull.

Because this process marches around the hull in counter-clockwise order, like a ribbon wrapping itself around the points, this algorithm is also called the "gift wrapping" algorithm.

4. Quickhull

This is an algorithm that deserves its name. The basic idea is as follows:

1. Choose the points with smallest (p_1) and largest (p_2) x coordinates.

2. Divide the rest of the points into those in the upper part of the hull and those in the lower part.

3. Recursively do the following for each part:

1) Find a point p_3 , that maximizes the triangular area (p_1, p_2, p_3). It will be on the hull.

2) For each other point, determine if it is:

- inside the triangle (disregard it);
- outside the left;
- outside the right

4. Call recursively with left and right points.

The algorithm has been suggested by Eddy [7] and Bykat [8].

5. The Running Time

The convex hull is often used as implementation of other algorithms. If there are only a few points, then it is not a big deal which algorithm to pick. But more usually different numbers of points are processed, so the main criterion of these algorithms is the running time of the convex hull.

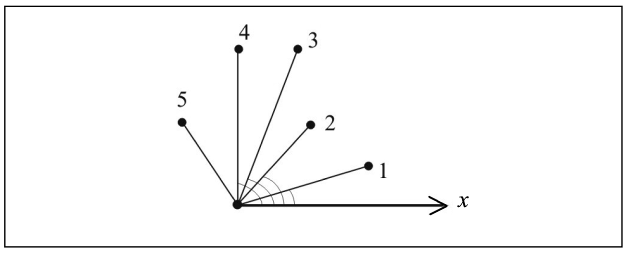

Fig 2. The sorted points

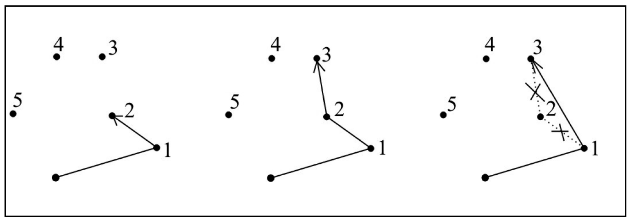

Fig 3. The main stage illustration of the Graham Scan

Table 1. The running time of the algorithms [9]

Algorithm	Complexity
Jarvis March	$O(nh)$
Quickhull	$O(nh)$
Graham Scan	$O(n \log n)$
Incremental	$O(n \log n)$
Divide-and-Conquer	$O(n \log n)$
Monotone Chain	$O(n \log n)$
Marriage-before-Conquest	$O(n \log h)$
Naive Brute force	$O(n^4)$
Better Brute force	$O(n^3)$

n is the number of points,
n is the number of points in *CH* ($h \leq n$).

The running time depends on the distribution of points, the number of vertexes and especially – the number of original points (Table 1).

So one way to increase the efficiency of all the algorithms – is to decrease the number of original points as much as possible. For reaching this purpose we can use a priori filtration of points, which is presented in this article.

6. A Priori Filtration of Points

The purpose of a priori filtration is not only to decrease the number of original points, but also to divide the array of points into the smaller arrays, and only then to start the search of the convex hull. There are suggested the following stages of filtration:

1. To find the extreme points: the leftmost point *(L)*, the rightmost point (*R*), the uppermost point (*U*) and the lowermost one (*D*). These points are sure to be vertexes of the convex hull.

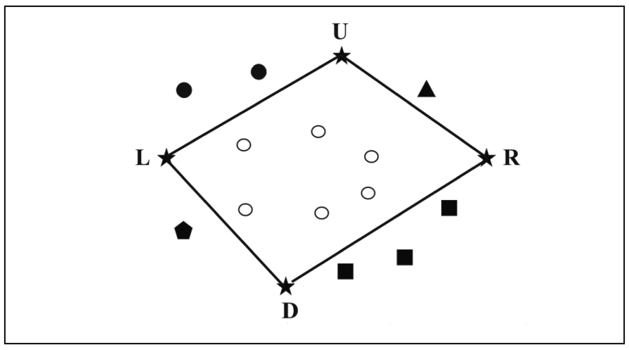

Fig 4. The square, after connecting extreme points

2. When extreme points are connected by segments we get a square (Fig 4).

3. All points, which are inside the square *ULDR*, are discarded and are not analysed anymore, be-cause they will not be the vertexes of the convex hull. The remaining points, taking into account the future operations, are divided into 1–4 arrays. The smaller number of points there is in the array, the easier and faster the further computation.

When the array of original points is decreased and divided into several and smaller arrays, we can invoke any convex hull algorithm chosen.

7. The New Convex Hull Algorithms

As mentioned above two new algorithms for finding convex hull of points are presented in this article.

7.1. An Idea of the Algorithm No 1

1. The algorithm starts with a priori filtration of points.

2. When the part of points, which does not belong to the convex hull is discarded, all remaining points are divided into 4 different arrays:

1) *S1* – points, outside the edge *LU* (these points are marked by a black circle in Fig 4);
2) *S2* – points, outside the edge *LD* (these points are marked by a black pentagon in Fig 4);
3) *S3* – points, outside the edge *DR* (these points are marked by a black square in Fig 4);
4) *S4* – points, outside the edge *UR* (these points are marked by a black triangle in Fig 4);

3. Every array of points is connected separately. Let us start with *S1*.

First of all we are searching for the point of array *S1* that is farthest from the segment *LU*. When this point is found (let it be the point F_1), it is included in the array of hull points and the decrease of *S1* is started. For that reason we draw two segments LF_1 (the leftmost point with the chosen point) and F_1U (chosen point with the uppermost point). So we get triangular LF_1U. All points, which are inside this triangular, are discarded from the array, because they do not belong to the convex hull (Fig 5).

Then we are searching for the farthest point of array *S1* of remaining points. This point is included in the array of hull and connected by segments with extreme points *L*, *U*. All the points inside the new triangular are discarded from the array *S1* (Fig 6).

Repeat everything until array *S1* is empty or the last point remains. This point is also included in the array of hull (Fig 7).

Then all chosen vertexes of hull are sorted by decreasing the *x*-coordinate. All chosen points are checked to belong to the convex hull invoking an idea of the Graham Scan. We are checking the curve turn by connecting one point with another one. If connecting one point with another one the curve turns to the right, then this point is discarded from the array of the hull (Fig 8).

When this test is finished, we have one part of convex hull vertexes (Fig 9).

4. The analogous operations are done for remaining arrays *S2*, *S3* and *S4*.

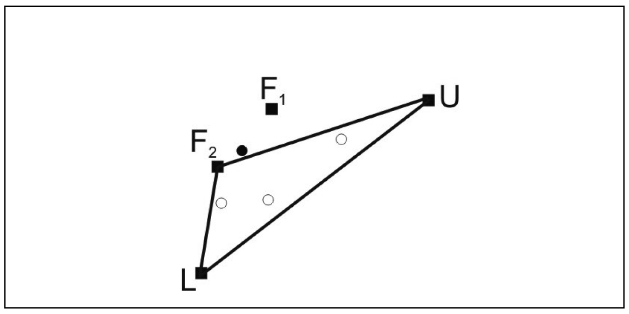

Fig 6. An intermediate stage illustration of the algorithm No 1

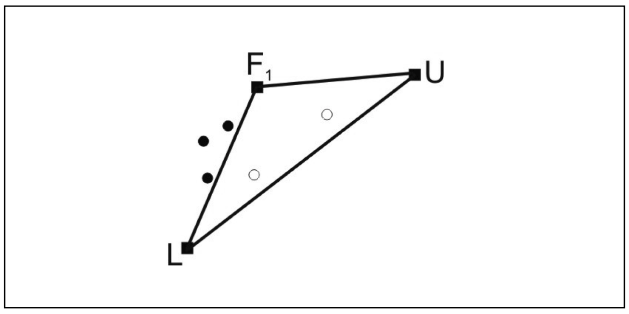

Fig 5. The original stage illustration of the algorithm No 1

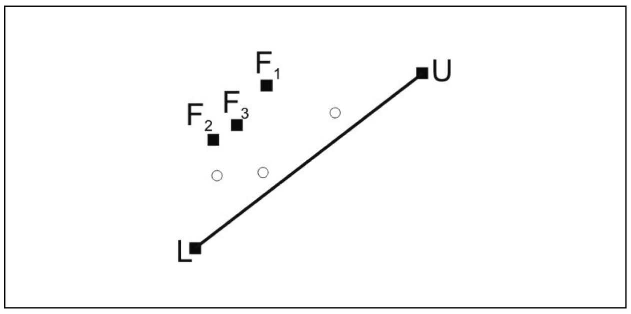

Fig 7. An intermediate stage illustration of the algorithm No 1

7.2. An Idea of the Algorithm No 2

1. The algorithm starts with a priori filtration of points.
2. When the part of points, which does not belong to the convex hull is discarded, all remaining points are divided into 2 different arrays:
   1) *S1* – points, outside the edges *RU* and *UL* (these points are marked by a black circle and a black triangle in Fig 2).
   2) *S2* – points, outside the edges *LD* and *DR* (these points are marked by a black pentagon and a black square in Fig 2).
3. Then all points of array *S1* are sorted by decreasing the *x*-coordinate and all points of array *S2* are sorted by increasing the *x*-coordinate.
4. Every array of points is connected separately. Let us start with *S1*.

The hull is constructed by invoking an idea of the Graham Scan. We are checking the curve turn by connecting one point with another one. If connecting one point with another one the curve turns to the right, then this point is discarded from the array of the hull (Fig 10).

When all points of array have been checked, we have the upper part of the convex hull vertexes (Fig 11).

5. The analogous operations are done for remaining array *S2*.

8. Experimental Investigation

8.1. The Efficiency of A Priori Filtration of Points

The efficiency of a priori filtration of points has been researched by the experiments. It has been counted how many points have been discarded during the filtration. The points have been generated in the area of size 100 000 000 × 100 000 000 points, repeating 50 times. The number of points has been changed from 10 to 100 000 points. The total average of discarded points is illustrated in Fig 12.

The efficiency of a priori filtration of points has been confirmed. After disposing only a few extra operations it is possible to decrease the amount of the original points by almost 50 %.

8.2. Efficiency Experiments of the Algorithms

The efficiency of new algorithms has been compared experimentally to the three most popular algorithms (Graham Scan, Jarvis March and Quickhull). The running time of algorithms depends not only on the number of points but on the number of vertexes too, so two different distributions of points are analysed:

- All points are generated inside a square;
- All points are arranged on a circle.

The main information about computer that has been used in research is: Pentium 4 CPU 2 GHz.

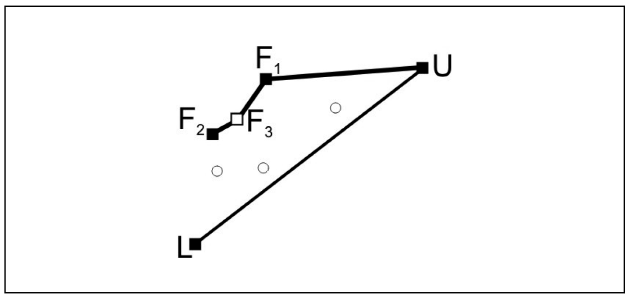

Fig 8. An intermediate stage illustration of the algorithm No 1

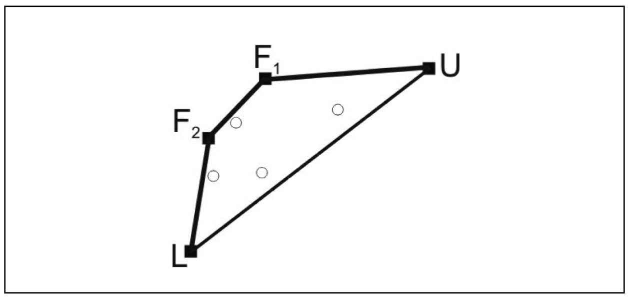

Fig 9. The last stage illustration of the algorithm No 1

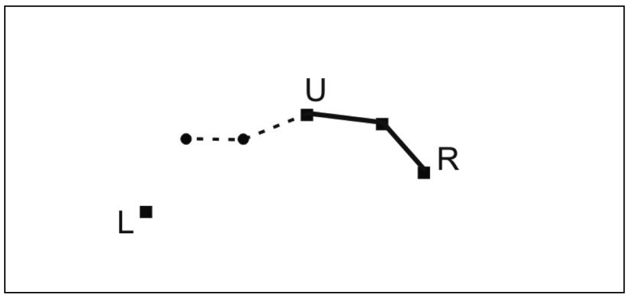

Fig 10. The original stage illustration of the algorithm No 2

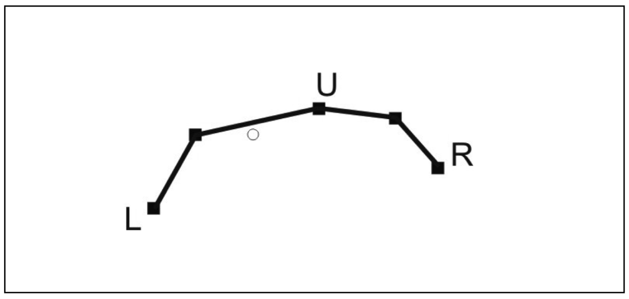

Fig 11. The last stage illustration of the algorithm No 2

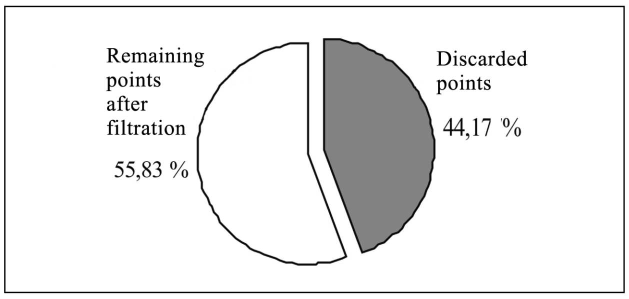

Fig 12. The total average of discarded points

8.2.1. All Points Are Inside The Square

The original random points with uniform distribution have been generated in the area of size 15 000 × 10 000 points. The results of running time dependence on number of points are illustrated in Figs 13–15.

Newly suggested algorithm No 2 has outrun the Graham Scan, and algorithm No 1 has been the fastest of all analysed algorithms.

The running time of the three most efficient algorithms has been checked generating original points in the 100 000 000 × 100 000 000 area of points. The number of points has been changed from 5000 to 100 000 points. The results were the same (Fig 16). Newly suggested algorithm No 1 has been the first in efficiency test.

All the three most efficient algorithms are input-sensitive, i.e. the running time depends on distribution of points. The experiments have been carried out with unfavourable conditions, when all points belong to the convex hull.

8.2.2. All Points of a Set are on a Circle

The original points have been generated on the circle within 50 000 000 points radius. The results of running time dependence on number of points are illustrated in Fig 17.

This time Quickhull and both suggested algorithms have proved to be the most efficient of all analysed algorithms. Algorithm No 1 has outrun the two most popular algorithms (Jarvis March and Graham Scan). Algorithm No 2 has outrun even Quickhull and has been the first in this efficiency test.

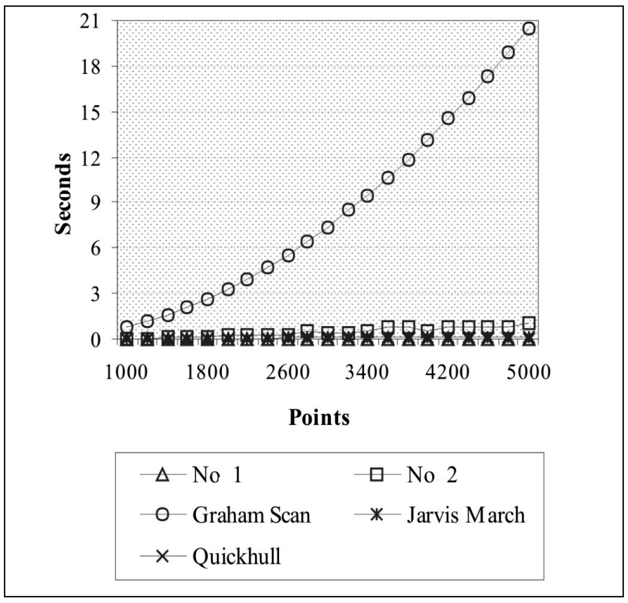

Fig 13. The results of running time dependence on number of points

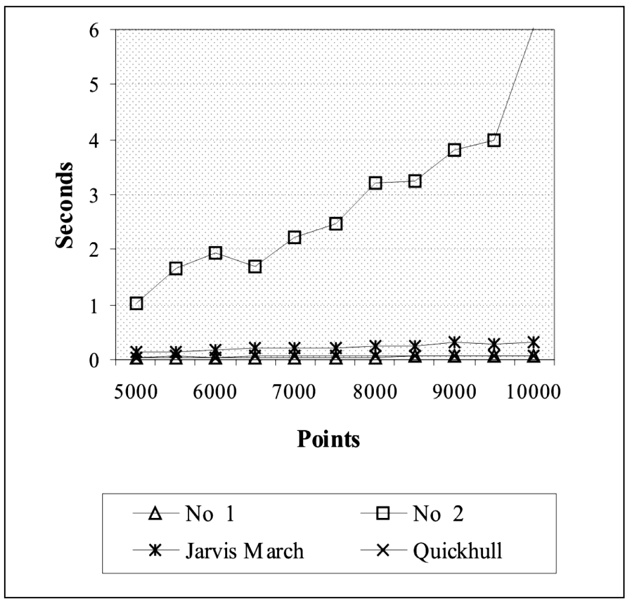

Fig 14. The results of running time dependence on number of points

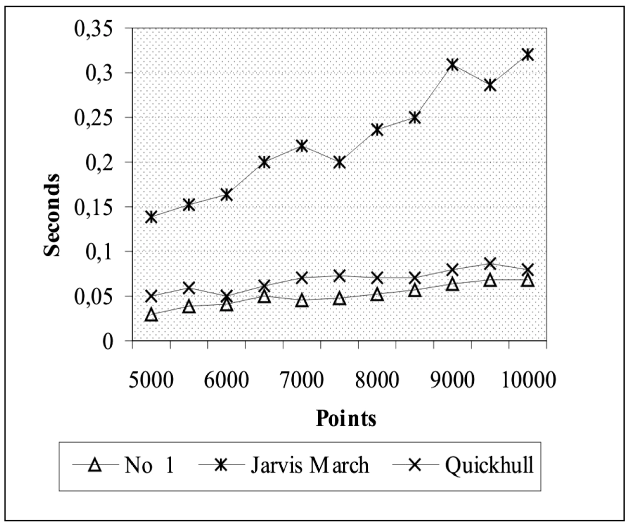

Fig 15. The results of running time dependence on the number of points for the three most efficient algorithms

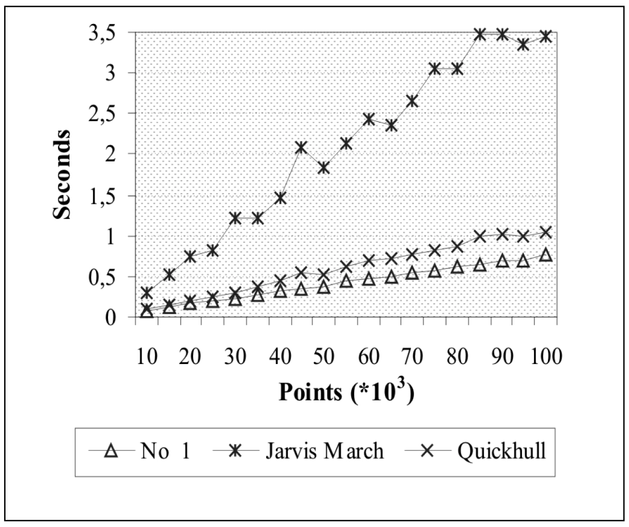

Fig 16. The results of running time dependence on number of points for the three most efficient algorithms

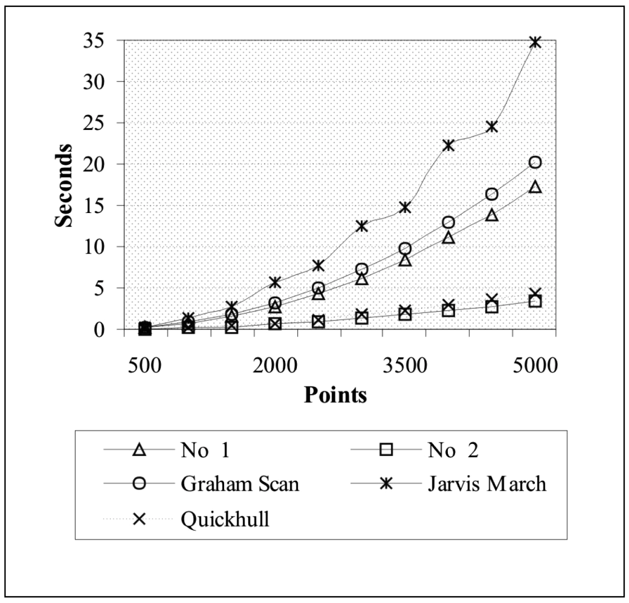

Fig 17. The results of running time dependence on number of points

9. Conclusion

A priori filtration has been suggested in this article. The idea of filtration is not only to decrease the number of original points, but also to divide ones into the smaller arrays, and in this way to ease and to speed up further computation. The results of the filtration research have shown that after several extra operations it is possible by almost 50 % to decrease the number of original points and, consequently, to increase the speed of algorithms.

Two new convex hull algorithms (algorithm No 1, algorithm No 2) have been created and suggested in this article. The efficiency of new algorithms has been compared to the three most popular algorithms (Graham Scan, Jarvis March and Quickhull). Algorithms have been compared by the main criterion – the running time. The running time depends not only on the number of points but sometimes on distribution of points as well. Because of this reason two different distributions of points have been investigated: 1) the points are generated inside the square; 2) All points are arranged on a circle, i. e. all original points are vertexes of convex hull. The most efficient algorithm in the first case has been algorithm No 1, in the second one – algorithm No 2.

References

1. Weisstein, E. W. Convex Hull. http://mathworld.wolfram.com/ConvexHull.html
2. Bhagavathi, D.; Gurla, H.; Olariu, S.; Schwing, J. L. and Zhang, J. Square Meshes Are Not Optimal For Convex Hull Computation. In: Proceedings of the 1993 International Conference on Parallel Processing, III – Algorithms & Applications. Boca Raton, FL: CRC Press, 1993, p. 307–310.
3. Skiena, S. S. The Stony Brook Algorithm Repository. 2001. http://www.cs.sunysb.edu/~algorith/files/convex-hull.shtml
4. Graham, R. An efficient algorithm for determining the convex hull of a finite point Set. *Info. Proc. Letters*, 1, 1972, p. 132–133.
5. O'Rourke, J. Computational Geometry in C (2nd Edition), Chap. 3: Convex Hulls in 2D. 1998. 376 p.
6. Jarvis R. A. On the identification of the convex hull of of a finite set of points in the plane. *Info. Proc. Letters*, 2, 1973, p. 18–21.
7. Eddy, W. A new convex hull algorithm for planar sets. *ACM Trans. Math. Software*, 3(4), 1977, p. 398–403.
8. Bykat, A. Convex hull of a finite set of points in two dimensions. *Info. Proc. Letters,* 7, 1978, p. 296–298.
9 Convex hull. http://www.algorithmist.com/index.php/Convex_Hull

APRIORINIS TAŠKŲ FILTRAVIMAS, IEŠKANT IŠKILOJO APVALKALO

L. Vyšniauskaitė, V. Šaltenis

Santrauka

Plokštumos taškų iškilasis apvalkalas yra mažiausias galimas iškilasis daugiakampis, kai visi aibės taškai yra jo viduje arba ant briaunų bei viršūnių. Jau šiuo metu literatūroje aptinkama nemažai skaičius iškilojo apvalkalo radimo algoritmų (*Graham Scan, Jarvis March, QuickHull, Incremental, Divide-and-Conquer, Marriage-before-Conquest, Monotone Chain, Brute Force*). Renkantis algoritmą, daugiausia dėmesio skiriama algoritmo atlikimo spartai.

Straipsnyje pasiūlyta, kokių veiksmų imtis, siekiant padidinti algoritmų spartą. Šiomis idėjomis remiantis, sukurti du nauji algoritmai, pasižymintys labai dideliu efektyvumu.

Reikšminiai žodžiai: iškilasis apvalkalas, apriorinis taškų filtravimas, efektyvumas, *Graham Scan, Jarvis March, QuickHull.*

Laura VYŠNIAUSKAITĖ. Bachelor's degree in Mathematics and Informatics (2003), Master's degree in Informatics (2006) from Vilnius Pedagogical University. Research interests: algorithm analysis and optimization.

Vydūnas ŠALTENIS graduated from the Kaunas Technological Institute, Lithuania. He received a Ph.D. degree from the Moscow Energy Institute of the USSR Academy of Sciences in 1966 and the Degree of Habil. Doctor from the Institute of Mathematics and Informatics, Vilnius in 1998. He is a principal researcher of the Systems analysis department at the Institute of Mathematics and Informatics, Lithuania. His present research interests include both theory and applications of the structure of optimisation problems, data and image analysis.