Peeling Digital Potatoes

Loïc Crombez[1], Guilherme D. da Fonseca[1], and Yan Gérard[1]

1 **Université Clermont Auvergne and LIMOS, Clermont-Ferrand, France**

Abstract

The potato-peeling problem (also known as convex skull) is a fundamental computational geometry problem and the fastest algorithm to date runs in $O(n^8)$ time for a polygon with n vertices that may have holes. In this paper, we consider a digital version of the problem. A set $K \subset \mathbb{Z}^2$ is *digital convex* if $\text{conv}(K) \cap \mathbb{Z}^2 = K$, where $\text{conv}(K)$ denotes the convex hull of K. Given a set S of n lattice points, we present polynomial time algorithms for the problems of finding the largest digital convex subset K of S (*digital potato-peeling problem*) and the largest union of two digital convex subsets of S. The two algorithms take roughly $O(n^3)$ and $O(n^9)$ time, respectively. We also show that those algorithms provide an approximation to the continuous versions.

1 Introduction

The *potato-peeling problem* [16] (also known as *convex skull* [23]) consists of finding the convex polygon of maximum area that is contained inside a given polygon (possibly with holes) with n vertices. The fastest exact algorithm known takes $O(n^7)$ time without holes and $O(n^8)$ if there are holes [9]. The problem is arguably the simplest geometric problem for which the fastest exact algorithm known is a polynomial of high degree and this high complexity motivated the study of approximation algorithms [8, 17]. Multiple variations of the problem have been considered, including triangle-mesh [1] and orthogonal [14, 24] versions. In this paper, we consider a digital geometry version of the problem.

The *digital potato-peeling problem* is defined as follows and is illustrated in Figure 1(a,b).

▶ Problem 1 (Digital potato-peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points described by their coordinates, determine the *largest* set $K \subseteq S$ that is digital convex (i.e., $\text{conv}(K) \cap \mathbb{Z}^2 = K$), where largest refers either to the area of $\text{conv}(K)$, or $|K|$.

Heuristics for the digital potato-peeling problem have been presented in [7, 10], but no exact algorithm. We also consider the question of covering the largest area with two digital convex subsets. The problem is defined as follows and is illustrated in Figure 1(a,c).

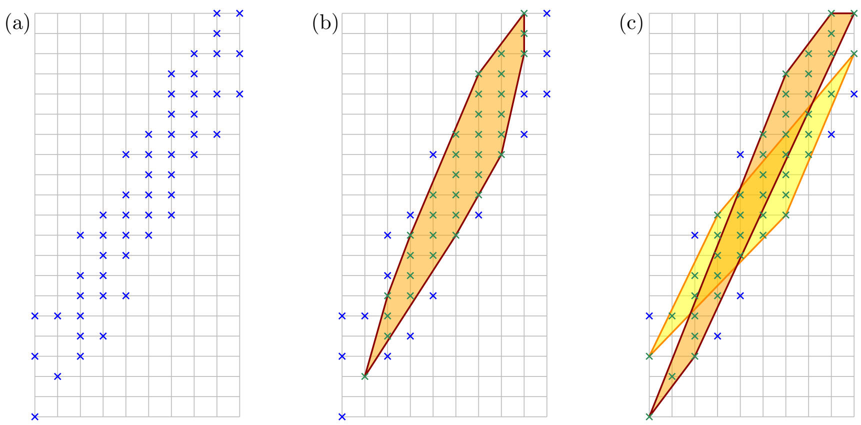

Figure 1 (a) Input lattice set S. (b) Largest digital convex subset of S (Problem 1). (c) Largest union of two digital convex subsets of S (Problem 2).

35th European Workshop on Computational Geometry, Utrecht, The Netherlands, March 18–20, 2019.
This is an extended abstract of a presentation given at EuroCG'19. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

▶ Problem 2 (Digital 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points described by their coordinates, determine the largest set $K = K_1 \cup K_2 \subseteq S$ such that K_1 and K_2 are both digital convex, where largest refers to the area of $\text{conv}(K_1) \cup \text{conv}(K_2)$.

A related continuous problem consists of completely covering a polygon by a small number of convex polygons inside of it. O'Rourke showed that covering a polygon with the minimum number of convex polygons is decidable [18, 19], but the problem has been shown to be NP-Hard with or without holes [13, 20]. Shermer [22] presents a linear time algorithm for the case of two convex polygons and Belleville [5] provides a linear time algorithm for three. We are not aware of any previous results on finding a fixed (non-unit) number of convex polygons inside a given polygon and maximizing the area covered.

Our results

We present polynomial time algorithms to solve each of these two problems. In Section 2, we show how to solve the digital potato-peeling problem in $O(n^3 \log r)$ time, where r is the diameter of the input S. Our algorithm builds the convex polygon $\text{conv}(K)$ through its triangulation, using a triangle range counting data structure [11] together with Pick's theorem [21] to test the validity of each triangle. The $O(\log r)$ factor comes from the gcd computation to apply Pick's theorem. Our algorithm makes use of the following two properties: (i) it is possible to triangulate K using only triangles that share a common bottom-most vertex v and (ii) if the polygons lying on both sides of one such triangle (including the triangle itself) are convex, then the whole polygon is convex.

These two properties are no longer valid for Problem 2, in which the solution $\text{conv}(K_1) \cup \text{conv}(K_2)$ is the union of two convex polygons. Also, since convex shapes are not pseudo-disks (the boundaries may cross an arbitrarily large number of times), separating the input with a constant number of lines is not an option. Instead of property (i), our approach uses the fact that the union of two (intersecting) convex polygons can be triangulated with triangles that share a common vertex ρ (that may not be a vertex of either convex polygon). Since ρ may not have integer coordinates, we can no longer use Pick's theorem, and resort to the formulas from Beck and Robins [4] or the algorithm from Barvinok [3] to count the lattice points inside each triangle in $O(\text{polylog}\, r)$ time.

Furthermore, to circumvent the fact that the solution no longer obeys property (ii), we use a directed acyclic graph (DAG) that encapsulates the orientation of the edges of both convex polygons. For those reasons, the running time of our algorithm for Problem 2 increases to $O(n^9 + n^6 \text{polylog}\, r)$. The corresponding algorithm is described in Section 3.

2 Digital Potato Peeling

In this section, we present an algorithm to solve the digital potato-peeling problem in $O(n^3 \log r)$ time, where n is the number of input points and r is the diameter of the set.

A digital convex set K can be described by its convex hull $\text{conv}(K)$ whose vertices are lattice points. Instead of explicitly building K, our algorithm constructs $\mathit{conv}(K)$. Note that it is always possible to triangulate a convex polygon with k vertices using $k-2$ triangles that share a bottom-most vertex ρ (*fan triangulation*). We first consider the rooted variation of the digital potato-peeling problem, where the point ρ is given as part of the input.

▶ Problem 3 (Rooted digital potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates and a point $\rho \in S$, determine the *largest* set $K \subseteq S$ that is digital convex and has ρ as the right-most point at the bottom-most row of K.

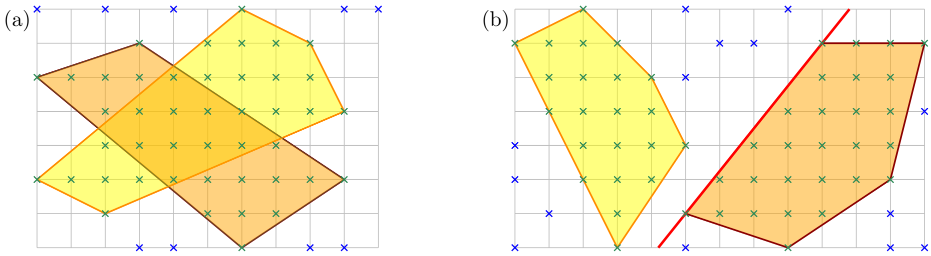

Figure 2 (a) The two optimal sets intersect. (b) The two optimal sets are disjoint and there is a supporting separating line.

Without loss of generality, we assume that all points in S lie either on the same row or on a row above ρ and all points on the same row of ρ are to the left of ρ. We refer to ρ as the *root*. Let $p_1, \ldots, p_n$ denote the points of S sorted clockwise around ρ, starting from left.

Let $\triangle_{i,j}$ denote the (closed) triangle whose vertices are ρ, p_i, p_j with $i < j$. We say that a triangle $\triangle_{i,j}$ is *valid* if $\triangle_{i,j} \cap \mathbb{Z}^2 = \triangle_{i,j} \cap S$. To algorithmically verify that $\triangle_{i,j}$ is valid, we compare $|\triangle_{i,j} \cap S|$ and $|\triangle_{i,j} \cap \mathbb{Z}^2|$ using Pick's theorem and a triangle range counting query [11]. The total time to test the validity of a triangle (after preprocessing) is $O(\log r)$.

The algorithm incrementally builds the fan triangulation of $\text{conv}(K)$ by appending valid triangles from left to right using dynamic programming.

For all $p_i, p_j \in S$ with $i < j$ and such that $\triangle_{i,j}$ is valid, the algorithm determines the largest convex polygon that has $\triangle_{i,j}$ as the right-most triangle. We refer to this convex polygon as $C_{i,j}$. The key property to efficiently compute $C_{i,j}$ is

$$C_{i,j} = \triangle_{i,j} \cup \max_h C_{h,i}, \text{ where } h < i \text{ is such that } \triangle_{i,j} \cup \triangle_{h,i} \text{ is convex.}$$

For a given i, by sorting all $C_{h,i}$ with $h < i$ according to their size and sorting all $\triangle_{i,j}$ according to the position of p_j around p_i, all the $C_{i,j}$ can be computed in $O(n \log n)$ time using the aforementioned property.Considering all n values of i and the initial sorting, the total time to solve Problem 3 is $O(n^2 \log r)$. In order to solve Problem 1, we test all n possible values of $\rho \in S$, proving the following theorem.

▶ **Theorem 1.** *There exists an algorithm to solve Problem 1 (digital potato peeling) in $O(n^3 \log r)$ time, where n is the number of input points and r is the diameter of the input.*

3 Digital 2-Potato Peeling

In this section, we show how to find two digital convex sets K_1, K_2, maximizing the area of $\text{conv}(K_1) \cup \text{conv}(K_2)$. Either the two convex hulls intersect or they do not (Figure 2). We treat those two cases separately and the solution to Problem 2 is the largest among both. Hence, we consider the two following variations of the 2-potato-peeling problem.

▶ Problem 4 (Disjoint 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates, determine the *largest* two digital convex sets $K_1 \cup K_2 \subseteq S$ such that $\text{conv}(K_1) \cap \text{conv}(K_2) = \emptyset$.

▶ Problem 5 (Intersecting 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points given by their coordinates, determine the *largest* union of two digital convex sets $K_1 \cup K_2 \subseteq S$ such that $\text{conv}(K_1) \cap \text{conv}(K_2) \neq \emptyset$. In this case, largest means the maximum area of $\text{conv}(K_1) \cup \text{conv}(K_2)$.

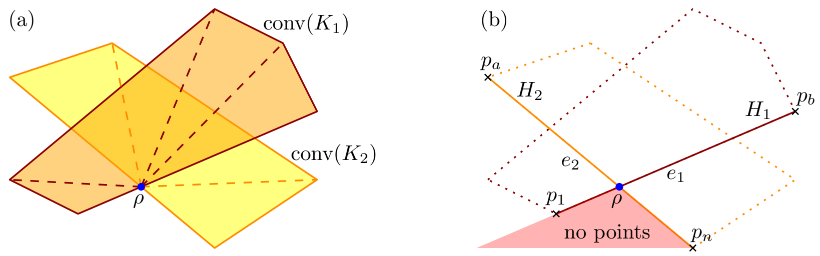

Figure 3 (a) A fan triangulation of two intersecting convex polygons from a point ρ. (b) Definitions used to solve Problem 6.

3.1 Disjoint Convex Polygons

It is well known that any two disjoint convex shapes can be separated by a straight line. Moreover two convex polygons can be separated by a supporting line of an edge of one of the convex polygons that contains no vertex of the other convex polygon (Figure 2(b)).

For each ordered pair of distinct points $p_1, p_2 \in S$, we define two subsets S_1, S_2. The set S_1 contains the points on the line p_1, p_2 or to the left of it (according to the direction $p_2 - p_1$). The set S_2 contains the remaining points of S.

For each pair of sets S_1, S_2, we independently solve Problem 1 for S_1 and S_2. Since there are $O(n^2)$ pairs and each pair takes $O(n^3 \log r)$ time, we solve Problem 4 in $O(n^5 \log r)$ time.

3.2 Intersecting Convex Polygons

The more interesting case is when the two convex polygons intersect (Problem 5). Note that it is possible to triangulate the union of two convex polygons that share a common boundary point ρ using a fan triangulation around ρ (Figure 3). Hence we consider the following rooted version of the problem.

▶ Problem 6 (Rooted 2-potato peeling). Given a set $S \subset \mathbb{Z}^2$ of n lattice points represented by their coordinates and two edges $e_1, e_2 \in S^2$ that cross at a point ρ, determine the *largest* union of two digital convex sets $K_1, K_2 \subseteq S$ such that e_1 is an edge of conv(K_1) and e_2 is an edge of conv(K_2).

Let ρ be the intersection point of e_1, e_2. To solve Problem 6 we encode the problem into a DAG (V, E) whose longest path corresponds to the solution. To avoid confusion, we use the terms *node* and *arc* for the DAG and keep the terms *vertex* and *edge* for the polygons.

Let $\mathcal{T}$ be the set of valid triangles with two vertices from S and ρ as the remaining vertex. The nodes $V = \mathcal{T}^2 \cup \{v_0\}$ are ordered pairs of valid triangles and a starting node v_0. The number of nodes is $|V| = O(n^4)$.

Each node $(\triangle_1, \triangle_2) \in V$ is such that $\triangle_1$ (resp. $\triangle_2$) is used to build the fan triangulation of conv(K_1) (resp. conv(K_2)). The arcs are defined in a way such that, at each step as we go through a path of the DAG, we add one triangle either to conv(K_1) or to conv(K_2). The arcs enforce the convexity of both conv(K_1) and conv(K_2). Furthermore, we enforce that we always append a triangle to the triangulation that is the least advanced of the two (in clockwise order), unless we have already reached the last triangle of conv(K_1). This last condition allows us to define the arc lengths in a way that it corresponds to the area of the union of the two convex polygons. Figure 4 illustrates the result of following a path on the DAG. As there is $O(n^4)$ pairs of starting edges and each DAG has $O(n^5)$ arcs, the total running time is roughly $O(n^9)$.

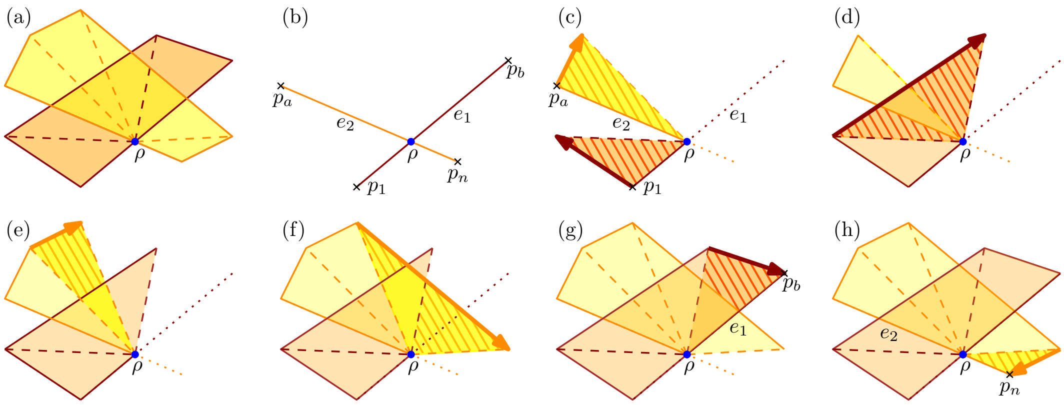

Figure 4 Steps of the algorithm from Section 3.2. Figure (a) represents the solution, while Figures (b) to (h) represent the triangulation obtained at each node of a path. The newly covered area that is assigned as the length of the corresponding arc is marked. In (b), we have the initial pair of edges e_1, e_2 which corresponds to the starting vertex. A first pair of triangles with vertices p_1 and p_a is obtained in (c). From (c) to (d) and from (f) to (g), the triangle $\triangle_1$ (less advanced than triangle $\triangle_2$) advances. From (d) to (e) and from (e) to (f), triangle $\triangle_2$ advances. In (g), the triangle $\triangle_1$ has reached the final node p_b. $\triangle_2$ advances until it reaches the end.

4 Conclusion and Open Problems

The (continuous) potato peeling problem is a very peculiar problem in computational geometry. The fastest algorithms known have running times that are polynomials of substantially high degree. Also, we are not aware of any algorithms (or difficulty results) for the natural extensions to higher dimensions (even 3d) or to a fixed number of convex bodies.

In this paper, we focused on a digital version of the problem. Many problems in the intersection of digital, convex, and computational geometry remain open. Our study falls in the following framework of problems, all of which receive as input a set of n lattice points $S \subset \mathbb{Z}^d$ and are based on a fixed parameter $k \geq 1$.

1. Is S the union of at most k digital convex sets?
2. What is the smallest superset of S that is the union of at most k digital convex sets?
3. What is the largest subset S that is the union of at most k digital convex sets?

In [12], the authors considered the first problem for $k = 1$, presenting polynomial time solutions (which may still leave room for major improvements for $d > 3$). We are not aware of any previous solutions for $k > 1$. In contrast, the continuous version of the problem is well studied. The case of $k = 1$ can be solved easily by a convex hull computation or by linear programming. Polynomial algorithms are known for $d = 2$ and $k \leq 3$ [5, 22], as well as for $d = 3$ and $k \leq 2$ [6]. The problem is already NP-complete for $d = k = 3$ [6]. Hence, the continuous version remains open only for $d = 2$ and fixed $k > 3$.

It is easy to obtain polynomial time algorithms for the second problem when $k = 1$, since the solution consists of all points in the convex hull of S. The continuous version for $d = k = 2$ can be solved in $O(n^4 \log n)$ time [2]. Also, the orthogonal version of the problem is well studied (see for example [15]). We know of no results for the digital version.

The third problem for $d > 2$ or $k > 2$ remains open. The DAG approach that we used for $d = 2$ is unlikely to generalize to higher dimensions, since there is no longer a single order by which to transverse the boundary of a convex polytope. Surprisingly, even the continuous version seems to be unresolved for $d > 2$ or $k \geq 2$.

References

1 Boris Aronov, Marc Van Kreveld, Maarten Löffler, and Rodrigo I. Silveira. Peeling meshed potatoes. *Algorithmica*, 60(2):349–367, 2011.

2 Sang Won Bae, Hwan-Gue Cho, William Evans, Noushin Saeedi, and Chan-Su Shin. Covering points with convex sets of minimum size. *Theoretical Computer Science*, 718:14–23, 2018.

3 Alexander I. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Mathematics of Operations Research*, 19(4):769–779, 1994.

4 Matthias Beck and Sinai Robins. Explicit and efficient formulas for the lattice point count in rational polygons using Dedekind-Rademacher sums. *Discrete & Computational Geometry*, 27(4):443–459, Jan 2002.

5 Patrice Belleville. On restricted boundary covers and convex three-covers. In *5th Canadian Conference on Computational Geometry (CCCG)*, pages 467–472, 1993.

6 Patrice Belleville. Convex covers in higher dimensions. In *7th Canadian Conference on Computional Geometry (CCCG)*, pages 145–150, 1995.

7 Gunilla Borgefors and Robin Strand. An approximation of the maximal inscribed convex set of a digital object. In *13th International Conference on Image Analysis and Processing (ICIAP)*, pages 438–445, 2005.

8 Sergio Cabello, Josef Cibulka, Jan Kyncl, Maria Saumell, and Pavel Valtr. Peeling potatoes near-optimally in near-linear time. *SIAM Journal on Computing*, 46(5):1574–1602, 2017.

9 Jyun S. Chang and Chee K. Yap. A polynomial solution for the potato-peeling problem. *Discrete & Computational Geometry*, 1(2):155–182, 1986.

10 Jean-Marc Chassery and David Coeurjolly. Optimal shape and inclusion: open problems. In *Mathematical Morphology: 40 Years On, International Symposium on Mathematical Morphology*, Computational Imaging and Vision. Springer Verlag, 2005.

11 Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8(1-6):407–429, 1992.

12 Loïc Crombez, Guilherme D. da Fonseca, and Yan Gérard. Efficient algorithms to test digital convexity. In *21st International Conference on Discrete Geometry for Computer Imagery (DGCI)*, 2019. URL: `http://fc.isima.fr/~fonseca/digitalconvexity.pdf`.

13 Joseph Culberson and Robert A. Reckhow. Covering polygons is hard. *Journal of Algorithms*, pages 17:2–44, 1994.

14 Mousumi Dutt, Arindam Biswas, Partha Bhowmick, and Bhargab B. Bhattacharya. On finding an orthogonal convex skull of a digital object. *International Journal of Imaging Systems and Technology*, 21(1):14–27, 2011.

15 Cem Evrendilek, Burkay Genç, and Brahim Hnich. Covering points with minimum/maximum area orthogonally convex polygons. *Computational Geometry*, 54:32–44, 2016.

16 Jacob E. Goodman. On the largest convex polygon contained in a non-convex n-gon, or how to peel a potato. *Geometriae Dedicata*, 11(1):99–106, 1981.

17 Olaf Hall-Holt, Matthew J. Katz, Piyush Kumar, Joseph S.B. Mitchell, and Arik Sityon. Finding large sticks and potatoes in polygons. In *17th annual ACM-SIAM symposium on Discrete algorithm (SODA)*, pages 474–483, 2006.

18 Joseph O'Rourke. The complexity of computing minimum convex covers for polygons. In *20th Allerton Conference on Communication, Control, and Computing*, pages 75–84, 1982.

19 Joseph O'Rourke. The decidability of covering by convex polygons. Technical Report JHU-EECS 82-4, Johns Hopking University, 1982.

20 Joseph O'Rourke. Some NP-hard polygon decomposition problems. *IEEE Transactions on Information Theory, IT-30*, pages 181–190, 1983.

21 Georg Pick. Geometrisches zur zahlenlehre. *Sitzungsberichte des Deutschen Naturwissenschaftlich-Medicinischen Vereines für Böhmen "Lotos" in Prag.*, v.47-48 1899-1900, 1899.

22 Thomas C. Shermer. On recognizing unions of two convex polygons and related problems. *Pattern Recognition Letters , 14(9)*, pages 737–745, 1993.

23 Tony C. Woo. The convex skull problem. Technical report, Department of Industrial and Operations Engineering, University of Michigan, 1986.

24 Derick Wood and Chee K. Yap. The orthogonal convex skull problem. *Discrete & Computational Geometry*, 3(4):349–365, 1988.