Convex Hulls on Cellular Automata

Luidnel Maignan[1] and Fredric Gruau[1,2,3]

[1] Institut National de Recherche en Informatique et Automatique, Centre de Saclay, Parc Orsay Université, 4 rue J. Monod, 91893 Orsay Cedex, France
[2] Laboratoire de Recherche en Informatique
Bât 490, Université Paris-Sud 11, 91405 Orsay Cedex, France
[3] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

Abstract. In the cellular automata domain, the discrete convex hull computation rules proposed until now only deal with a connected set of seeds in infinite space, or with distant set of seeds in finite space. We present a cellular automata rule that constructs the discrete convex hull of arbitrary set of seeds in infinite spaces. The rule is expressed using intrinsic and general properties of the cellular spaces, considering them as metric spaces. In particular, this rule is a direct application of metric Gabriel graphs. This allows the rule and its components to be used on all common 2D and 3D grids used in cellular automata.

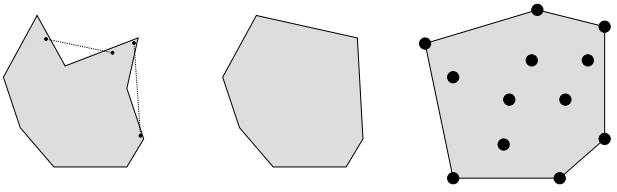
1 Introduction

In abstract convexity theory[10,13], a *convexity* $\mathcal{C} \subset 2^S$ over a set S is a collection of subsets of S, called *convex subsets* or *convex sets*, that is closed under intersection. The *convex hull* $H_{\mathcal{C}}(S_0)$ of an arbitrary subset $S_0 \subset S$, whose elements will be called *seeds*, is the minimal convex set containing S_0. For specific S and $\mathcal{C}$ depending on the domain of application, finding an algorithm that realizes the convex hull operator $H_{\mathcal{C}} \colon 2^S \rightarrow \mathcal{C}$ is a common problem.

The most known and studied convexity is defined over Euclidean spaces $\mathbb{R}^d$. A subset is *Euclidean-convex* if it contains all the segments joining two of its points. For example, Fig. 1(a) shows a non Euclidean-convex set, commonly called concave set, along with two segments that are not contained in the set. In Fig. 1(b), it is not possible to find any missing segment. Figure 1(c) gives an example of convex hull. Algorithms that construct Euclidean-convex hulls for different dimensionality include Jarvis's gift wrapping [8], Graham's scan [6], Kirkpatrick–Seidel algorithm [9], and Chan's algorithm [3].

Euclidean convexity is a particular case of *metric convexity*. This latter is defined over metric spaces, i.e. sets of points associated with a distance function (also called *metric*). A subset is metric-convex if it contains all the shortest paths joining two of its points. In domains like computer graphics or cellular automata, the metric spaces typically consist of a set of pixels or cells, the length of a path being related to the number of pixels or cells lying on it. Networks of computers, wireless devices or sensors consider the metric space corresponding to the graph

S. Bandini et al. (Eds.): ACRI 2010, LNCS 6350, pp. 69–78, 2010.



(a) Non-convex set (b) Convex set (c) Convex hull

Fig. 1. Euclidean convexity examples. Considered set for convexity in gray, seeds in black. Note that (b) is also the convex hull of (a).

of communication, each edge having an arbitrary length, and the length of a path being the sum of the lengths of its edges. More abstract metric spaces are considered in machine learning and multivariate analysis for example.

In this article, we tackle the convex hull problem in the cellular automata framework [7]. In this parallel computation framework, the set of processing elements themselves are the points of the space and can only communicate locally. The problem is then to have each processing element to compute the convex hull of a selected subset of them by selecting itself if it is in it. We start by describing the cellular automata framework, and the metric spaces and convexities related to it in Sect. 2. We review the existing results in this domain and compare them with ours in Sect 3. We describe our solution incrementally, by giving the intuition that leads to it in Sect 4.

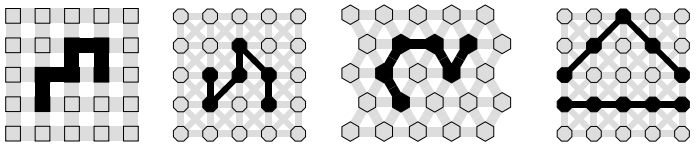
2 Framework and Definitions

2.1 Cellular Automata Framework

A cellular automaton is made of an infinite set S of *sites*, i.e. processing elements having a particular position in space. S is called the *cellular space* and forms a crystalline graph, whose neighborhood relation is denoted as N. Being processing elements, each site x has a particular state $v_t(x) \in V$ (as value) that changes with time, V being finite. The updating of the sites is synchronous and local. This means that all the sites update their states at the same time using the same updating rule. The latter determines the next state $v_{t+1}(x)$ of each site x based on the previous state of a finite number of its closest sites. In this framework, the convex hull problem is formulated this way: Given a set of seeds S_0, represented by a configuration src defined as $src(x) \equiv x \in S_0$, the goal is to find a rule R such that, from some instant t, $R_t(x) \equiv x \in H_{\mathcal{C}}(S_0)$. Examples of initial and associated final configuration for different grids are given in Fig. 4.

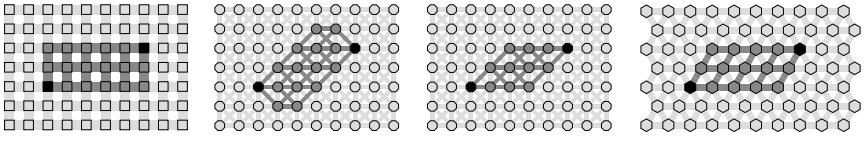
2.2 Cellular Spaces and Metric Spaces

Different cellular spaces may be used and, regarding convexities, different metrics can be associated to them. The most commonly used bi-dimensional cellular



(a) 4-square (b) 8-square (c) hexagonal (d) 8-square

Fig. 2. Grids used in this article: the polygons (squares, octagons and hexagons) correspond to the sites, the lines correspond to the edges, and example paths are in black.



(a) 4-square (b) 8-square (c) 8-square-$\sqrt{2}$ (d) hexagonal

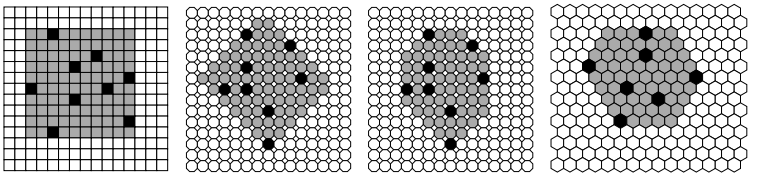
Fig. 3. Intervals for different grids

spaces are the square grids, with 4 or 8 neighbors per site, and the hexagonal grids, having 6 neighbors per site. Figure 2 shows these grids, exposing the nodes and the edges.

For the distance function, we consider the *hop count metric*, which associates to each edge the unitary length. This is a natural choice for the 4-square and hexagonal grids since all edges have the same length when we draw them. However, this is not the case for the 8-square grids, whose diagonal edges are drawn $\sqrt{2}$ times longer than the vertical and horizontal ones. Therefore, we also consider two metrics for the 8-square grids: the hop count metric, and the $\{1, \sqrt{2}\}$ *metric* that associates unitary and $\sqrt{2}$ lengths to non-diagonal and diagonal edges respectively. With the hop count metric, paths represented on Figs. 2(a), 2(b), and 2(c) have length 5. With the $\{1, \sqrt{2}\}$ metric, the path of Fig. 2(b) has length $3 + 2\sqrt{2}$. It is important to note that paths of Fig. 2(d) have the same length for hop count and different lengths for $\{1, \sqrt{2}\}$, since this has an effect on the notions of shortest path and convexity.

2.3 Convexities and Convex Hulls

For an arbitrary metric space, a point z lying on a shortest path joining two points x and y is said to be *between* the two points. It is denoted as $z \in [x, y]$, where $[x, y]$ is called the *interval* between x and y. Formally, we have $[x, y] = \{ z \in S \mid d(x, z) + d(z, y) = d(x, y) \}$ where d is the metric and S the set of points. In Euclidean spaces, $[x, y]$ corresponds to the segment joining the points x and y, since it is the unique shortest path joining these points. In other metric spaces, including square and hexagonal grids, there may be many shortest paths between two points, leading to more complex intervals as shown in Fig. 3.



(a) 4-square (b) 8-square (c) 8-square-$\sqrt{2}$ (d) hexagonal

Fig. 4. Each site x is black if $x \in S_0$, or gray if $x \in H_{\mathcal{C}}(S_0)$

Using intervals, metric convexity can be defined as follows. A subset $C \subseteq M$ is metric-convex if and only if $[x, y] \subseteq C$ for any pair of points $(x, y) \in C^2$. If we define the operator $I(P) = \bigcup\{\, [x, y] \mid (x, y) \in P^2 \,\}$ that adds to a set of points the shortest path joining them, we can say that a set C is metric-convex if and only if $I(C) = C$.

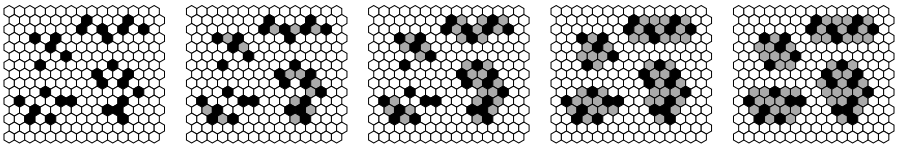
The convex hull $H_C(S_0)$ of a set $S_0 \subset S$ of seeds is the minimal convex set containing S_0. Using $I(\bullet)$, it corresponds to the limit of the sequence $I(S_0), I(I(S_0)), \ldots$, since $I(\bullet)$ adds points that have to be in the convex set, and the limit L verifies $I(L) = L$. Our algorithm also adds points iteratively.

3 State of the Art

The research about convex hulls on cellular automata mainly studies 8-square grids with $\{1, \sqrt{2}\}$ metric. In fact, the convexity is often not defined using sets, as we did, but using angles, and intersecting half planes. Although the definitions are distinct, the convex hulls are the same objects. For example, the 45-convexity is a version of the Euclidean convexity that only allows lines having angles of multiples of 45 to be used on the boundary of the convex hull. In this framework, the four metric spaces described previously correspond to the set of angles $\{0, 90\}$, $\{45, 135\}$, $\{0, 45, 90, 135\}$ and $\{0, 60, 120\}$.

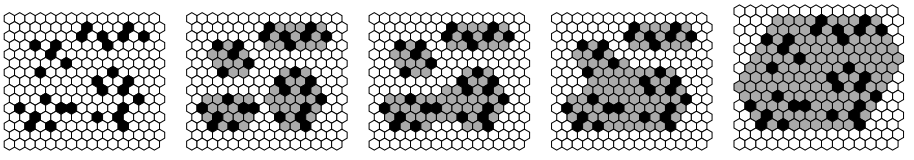
3.1 Rule for Connected Set of Seeds

In [1], some of the first proposed rules for the convex hull problem are described. The intuition is that any bordering site that does not have a local configuration corresponding to the shape of a convex set boundary has to select itself. After observing that all rejected local configuration have 1, 2 or 3 selected sites, the rule is simplified to a counter that checks whether there are at least 4 selected sites in the neighborhood. This rule can be generalized to the majority rule, since 4 can be interpreted as the half of 8, the number of neighbors. The convex hull behavior of the majority rule is described in [7]. In fact, the set of seeds does not need to be connected, but simply denser enough, since only their quantity matters. Also, as more and more sites are selected by the rule, many local convex hulls can merge to form bigger convex hulls that can also merge, etc. Using our



(a) t=01 (b) t=02 (c) t=03 (d) t=05 (e) t=08

Fig. 5. With the set of seeds (a), we obtain a set of convex hulls



(a) t=01 (b) t=08 (c) t=10 (d) t=14 (e) t=26

Fig. 6. Shifting only one seed to its neighbors connects recursively all the hulls

notations, the majority rule can be expressed as follows, and gives the results shown in Figs. 5 and 6 for hexagonal grids.

$$majo_{t+1}(x) = \begin{cases} \top \text{ if } src(x) \\ \top \text{ if } card\{\, y \in N(x) \mid majo_t(y) \,\} \geq \frac{card(N(x))}{2} \\ \bot \text{ otherwise.} \end{cases}$$

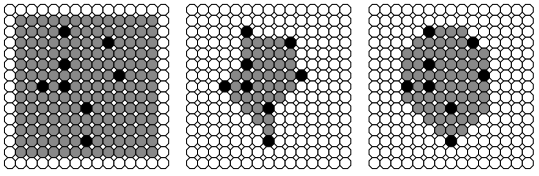
3.2 Rule for Already Wrapped Seeds

In [14,4], a rule is proposed for non connected set of seeds. However, it requires the seeds to be finitely wrapped in a connected and compact pattern. One can also consider that this rule only works for finite spaces, considering the space itself as the finite wrapper. The proposed rule consists of two globally successive stages. The first one erodes the wrapper until a minimal isometric set is obtained. Thus between any pair of points of the set, at least one shortest path is in the set. The second stage is the application of a rule to transform this connected set of sites into its convex hull, as done in the previous subsection. The stages are shown in Fig. 7.

3.3 Comparison with Our Rule

In comparison with this approach, we do not require any bound on the space, which can therefore be infinite. As a result, convergence of our cellular automaton takes an infinite time, as there can be sites infinitely far from the seeds. Still, each site converges locally, and the convex hull itself will be built in finite time, if the distances between nearest seeds are upper bounded.

Our rule does not work directly for $\{1, \sqrt{2}\}$ metrics, because we manipulate distances that have to be integers in order to be encoded using finite states.



(a) input (b) erosion (c) convex hull

Fig. 7. Stages from wrapped seeds to their convex hull: The initial wrapping (a) is shrunk to (b) and is then grown to convex hull (c)

However, this is not a problem, since we can still produce the $\{1, \sqrt{2}\}$ convex hull having both diagonal and vertical-horizontal border, by intersecting two convex hulls: the 4-square one, having only vertical-horizontal borders, and the 8-square one with hop count, having only diagonals.

The formulation of our rule does not use boundaries at all, which has an important consequence for generalization: the rule applies directly to many bidimensional grids, including the ones listed in Fig. 2. It also works for their tridimensional counterparts. Lastly, it can be easily applied to bigger neighborhoods, resulting in faster convergence when needed.

4 Rules for Arbitrary Set of Seeds Using Hop Count

4.1 Rule for Local Convexity

Computing the convex hull locally means that each site has to select itself if it belongs to the convex hull of the selected sites present in its neighborhood. The computation of the local convex hull is an easy task due to the simplicity and finiteness of the space in the neighborhood of each site. Indeed, it is enough for a site to check if it lies on a shortest path joining two of its selected neighbors. This gives the following local convexity rule:

$$conv_t(x) = \begin{cases} \top \text{ if } src(x) \\ \top \text{ if } \exists \{y_0, y_1\} \subset \{\, y \in N(x) \mid conv_{t-1}(y) \,\}; x \in [y_0, y_1] \\ \bot \text{ otherwise.} \end{cases}$$

Let us mention that testing $x \in [y_0, y_1]$ with hop count metric is equivalent to testing $d(y_0, y_1) = 2$ since $x \in [y_0, y_1] \Leftrightarrow d(y_0, x) + d(x, y_1) = d(y_0, y_1)$ and $y \in N(x) \Leftrightarrow d(x, y) = 1$. For the $\{1, \sqrt{2}\}$ metric, the test has to be done explicitly.

Because it follows directly from the convex hull definition, this rule is more precise than the *majo* rule. It is possible to check that whenever one selects half of the neighbors of a site x, then two of them are joined by a shortest path containing the site x. The reverse is obviously not true, which means the

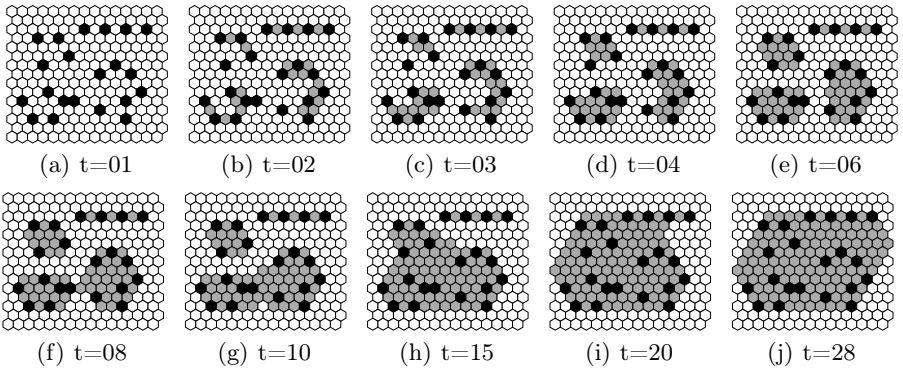


Fig. 8. Evolution of the *conv* rule. The *majo* rule is stationary on (a).

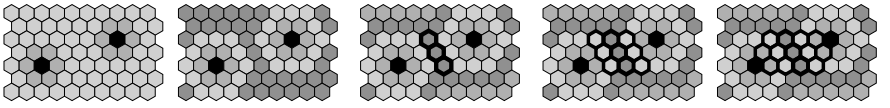
conv rule is able to construct the convex hull in more cases than the *majo* rule. One can also note that applying *majo* on an 8-square grid can only give a $\{1, \sqrt{2}\}$-convex hull. The formulation of *conv* allows choosing the metric to use, and tackles many grid topologies at once, such as the ones considered in this article and their tridimensional counterparts. However, it exhibits roughly the same behavior. Figure 8 shows the evolution of the *conv* rule with an initial configuration on which *majo* is stationary.

4.2 Global Convexity with Only Two Seeds

Since we have seen solutions to transform a connected set of seeds into its convex hull, a natural idea to obtain the convex hull of an arbitrary set of seeds is to connect them in a minimal way that remains in the desired convex hull. We start by studying the simpler case of only two seeds. In this setting, the goal is to select sites of the interval, as shown in Fig. 3.

To do so, we compute the distance of each site to the nearest seed, and look at the resulting pattern (Fig. 9(a)). We can notice distinct sites, namely the middle sites which are exactly the middle of the shortest path joining the two seeds. It turns out that these middles can always be detected by looking at the distance values in a bounded neighborhood. Therefore, they will be the first sites identified as being in the convex hull (Fig. 9(b)). All the other sites of the interval can then be selected by back-propagating from the middles to the seeds, by traveling towards neighbors that are closer to the seeds, again by using the distance values (Fig. 9(c)). This achieves the desired construction, without using any global phase transition but only local interaction. Let us now describe each rule in more details.

Distance Field. The distance computation described earlier is what we call a *distance field*. For hop count metrics, it associates to each site an integer and



(a) Distance (b) Middles (c) Back propagation

Fig. 9. The rules *dist*, *cent* and *back* in action

can be expressed by the following rule. The latter converges to $dist_\infty(x) = min\{\, d(x,y) \mid src(y) \,\}$:

$$dist_t(x) = \begin{cases} 0 \text{ if } src(x) \\ 1 \text{ if } \neg src(x) \wedge t = 0 \\ \min\{\, 1 + dist_{t-1}(y) \mid y \in N(x) \,\} \text{ otherwise.} \end{cases}$$

While this rule has an infinite number of possible state, we only need its gradient, i.e. the differences between the distance of neighboring sites. In [11], we have shown how to represent the gradients of some kind of integer fields with a finite number of states, thanks to the modulo operator. Applied on the *dist* rule, it allows representing it modulo 3 and gives the way to compute the gradient from these modulo values. For brevity and readability, we use directly *dist* in the rest of the paper and redirect the interested reader to [11]. Finally, we do not have any mean to represent the distance field for $\{1, \sqrt{2}\}$ metric with finite number of states, which is the reason why the final rule can not be directly applied to this metric.

From Middles Back to Seeds. As mentioned earlier, the middle sites are detected using the distance values present in their neighborhood. The global idea is to detect distance patterns that only happen when there are two nearest seeds for the sites, such that the site is between them. Because we treat this detection in our general framework instead of a particular grid, we delay the discussion about this detection to the next subsection, and directly use $cent(x)$ to denote that a site x is a middle.

For the back-propagation, each site having a selected site in its neighborhood has to determine if it is closer or not than the selected neighbor. If it is so, it can select itself, since it is between the selected neighbor and the seeds. This is expressed as:

$$back_t(x) = \begin{cases} \top \text{ if } cent_t(x) \\ \top \text{ if } t > 0 \wedge \exists y \in N(x), back_{t-1}(y) \wedge dist_{t-1}(x) < dist_{t-1}(y) \\ \bot \text{ otherwise} \end{cases}$$

4.3 Global Convexity and Metric Gabriel Graphs

When considering the general case with many seeds, some questions naturally arise: do the rules presented for only two seeds do all the work pairwise? Do they

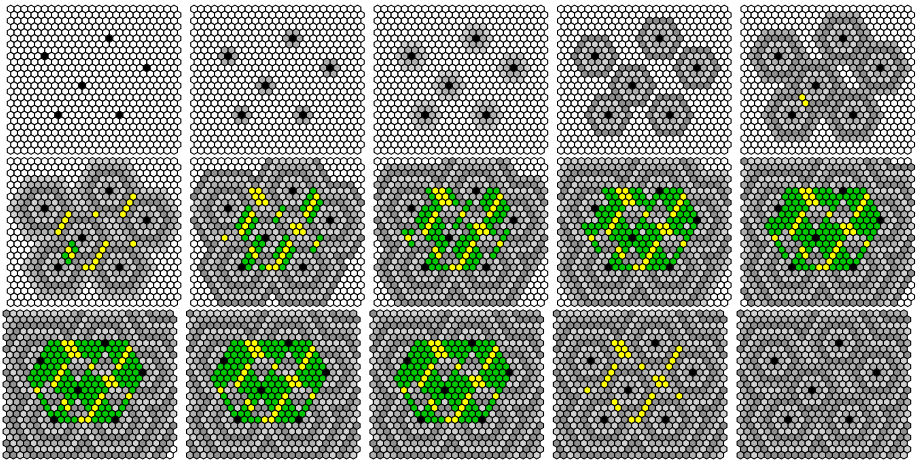


Fig. 10. Snapshots of the computation of the three layers simultaneously. The two last snapshot show the final configuration with, firstly *back* hidden and then *back* and *cent* hidden. Red: generators, gray: *dist*, yellow: *cent*, green: *back*.

produce a connected set? What structure is constructed? The answer is that we produce a connected set, connecting the seeds pairwise to draw a structure related to Delaunay graphs. A complete description is beyond the scope of this article but can be found in [12]. We only give here the material that allows understanding the global structure of the constructed graph.

When computing a distance field, one implicitly associates to each site its nearest seed. It is strongly related to Voronoi diagram [2], the set of sites having the same nearest seed being called the Voronoi region of the seed and the sites having many nearest seeds being the boundaries between the Voronoi regions. In our case, we detect boundary sites that are on a shortest path between the corresponding seeds, to make sure to select only sites that are in the convex hull.

By doing so, we only connect seeds of neighboring Voronoi regions, such that there is a shortest path going trough the boundary between the two regions. Replacing the words "shortest path" by "segment", we obtain one of the properties of Gabriel graphs [5], a connected sub-graph of the Delaunay graph defined for Euclidean spaces. In [12], we generalize the definition of Gabriel graphs to arbitrary metric spaces and obtain *metric Gabriel graphs*, which identify exactly what we need to detect in order to have a connected set of sites. We also explain in detail the rule *cent* (as metric Gabriel ball *centers*).

By using metric Gabriel graphs, we have, roughly speaking, that $back \circ cent \circ dist$ constructs a connected set of sites that is a subset of the convex hull. In order to complete the convex hull, we simply have to consider $conv \circ back \circ cent \circ dist$. The final cellular automaton thus described has 7 states: (3 distance states) * (2 "in convex hull" states) + 1 special "seed" state. Because of *cent* rule, it uses a neighborhood of radius 2. The evolution of the rule without *conv* is shown in Fig. 10.

Acknowledgments

We would like to thank Prof. Adamatzky who pointed us first to the convex hull problem, which was the starting point of this work, and secondly to Gabriel graphs, when we showed him our particular graph and its properties.

References

1. Adamatzky, A.: Computing in nonlinear media and automata collectives. IOP Publishing Ltd., Bristol (2001)
2. Aurenhammer, F.: Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Comput. Surv. 23(3), 345–405 (1991)
3. Chan, T.M.: Optimal output-sensitive convex hull algorithms in two and three dimensions. Discrete & Computational Geometry 16, 361–368 (1996)
4. Clarridge, A.G., Salomaa, K.: An improved cellular automata based algorithm for the 45-convex hull problem. Journal of Cellular Automata (2008)
5. Gabriel, R.K., Sokal, R.R.: A new statistical approach to geographic variation analysis. Systematic Zoology 18(3), 259–278 (1969)
6. Graham, R.L.: An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters 1(4), 132 – 133 (1972), http://dx.doi.org/10.1016/0020-0190(72)90045-2
7. Ilachinski, A.: Cellular Automata: A Discrete Universe. World Scientific Publishing Co., Inc., River Edge (2001)
8. Jarvis, R.A.: On the identification of the convex hull of a finite set of points in the plane. Information Processing Letters 2(1), 18 (1973), http://dx.doi.org/10.1016/0020-0190(73)90020-3
9. Kirkpatrick, D.G., Seidel, R.: The ultimate planar convex hull algorithm? SIAM Journal on Computing 15(1), 287–299 (1986), http://link.aip.org/link/?SMJ/15/287/1
10. Levi, F.: On helly's theorem and the axioms of convexity. J. of Indian Math. Soc., 65–76 (1951)
11. Maignan, L., Gruau, F.: Integer gradient for cellular automata: Principle and examples. In: IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, pp. 321–325 (2008)
12. Maignan, L., Gruau, F.: Gabriel graphs in arbitrary metric space and their cellular automaton for many grids. ACM Trans. Auton. Adapt. Syst. (2010) (in press)
13. Singer, I.: Abstract convex analysis. John Wiley & Sons Inc., Chichester (1997)
14. Torbey, S., Akl, S.G.: An exact and optimal local solution to the two-dimensional convex hull of arbitrary points problem. Journal of Cellular Automata (2008)