On the generation of topological (n_k)-configurations

Jürgen Bokowski * Vincent Pilaud ‡

Abstract

An (n_k)-configuration is a set of n points and n lines in the projective plane such that the point – line incidence graph is k-regular. The configuration is geometric, topological, or combinatorial depending on whether lines are considered to be straight lines, pseudolines or just combinatorial lines.

We provide an algorithm for generating all combinatorial (n_k)-configurations that admit a topological realization, for given n and k. This is done without enumerating first all combinatorial (n_k)-configurations.

Among other results, our algorithm enables us to confirm, in just one hour with a Java code of the second author, a satisfiability result of Lars Schewe in [11], obtained after several months of CPU-time.

1 Introduction

An (n_k)-*configuration* (P, L) is a set P of n points and a set L of n lines such that each point of P is contained in k lines of L and each line of L contains k points of P. Two lines of L are allowed to meet in at most one point of P and two points of P can lie in at most one common line of L. According to the underlying incidence structure, we distinguish three different levels of configurations, in increasing generality:

Geometric configurations: Points and lines are ordinary points and lines in the real projective plane $\mathbb{P}$.

Topological configurations: Points are ordinary points in $\mathbb{P}$, but lines are *pseudolines*, *i.e.* non-separating simple closed curves of $\mathbb{P}$.

Combinatorial configurations: Points and lines are just required to form an abstract incidence structure (P, L) as described above.

The study of point – line configurations has a long history in discrete 2-dimensional geometry. We refer to Branko Grünbaum's recent monograph [8] for a detailed treatment of the topic. The current challenge is to determine for which values of n do geometric, topological, and combinatorial (n_k)-configurations exist for a given k, and to enumerate and classify them.

For $k = 3$, the existence of (n_3)-configurations is well understood: combinatorial (n_3)-configurations exist for every $n \geq 7$, but topological and geometric (n_3)-configurations exist only for every $n \geq 9$. For example, Fano's combinatorial (7_3)-configuration cannot be realized as a topological configuration. As further examples, Pappus' and Desargues' theorems form famous (9_3)- and (10_3)-configurations respectively. This description is still almost complete for $k = 4$: combinatorial (n_4)-configurations exist iff $n \geq 13$, topological (n_4)-configurations exist iff $n \geq 17$ [3] and geometric (n_4)-configurations exist iff $n \geq 18$ [7, 4], with the possible exceptions of 19, 22, 23, 26, 37 and 43. For general k, the situation is more involved, and the existence of combinatorial, topological and geometric (n_k)-configurations is not determined in general.

In this paper, we describe an algorithm for generating, for given n and k, all combinatorial (n_k)-configurations that admit a topological realization, without enumerating first all combinatorial (n_k)-configurations. The algorithm sweeps the projective plane to construct a topological (n_k)-configuration (P, L), but only considers as relevant the events corresponding to the sweep of points of P. This strategy enables us to identify along the way some topological configurations which realize the same combinatorial configuration, and thus to maintain a reasonable computation space and time.

We developed two different implementations of this algorithm. The first one was written in Haskell by the first author to develop the strategy of the enumeration process. Once the general idea of the algorithm was settled, the second author wrote another implementation in Java, focusing on the optimization of computation space and time of the process.

We underline three motivations for this algorithm. First, the algorithm is interesting in its own right. Before describing some special methods for constructing topological configurations, Branko Grünbaum writes in [8, p. 165] that *"the examples of topological configurations presented so far have been ad hoc, obtained essentially through (lots of) trial and error"*. Our algorithm can reduce considerably the *trial and error method*. Second our algorithm enables us to check and confirm the previous results obtained in earlier papers, *e.g.* for $k = 4$ and $n \leq 18$ in [4, 11]. We can use a single method and reduce considerably the computation time (*e.g.* the computation of the (18_4)-configurations

*Tech. Univ. Darmstadt, juergen.bokowski@googlemail.com
‡Univ. Paris 7, vincent.pilaud@liafa.jussieu.fr, Research supported by Spanish MEC grant MTM2008-04699-C03-02.

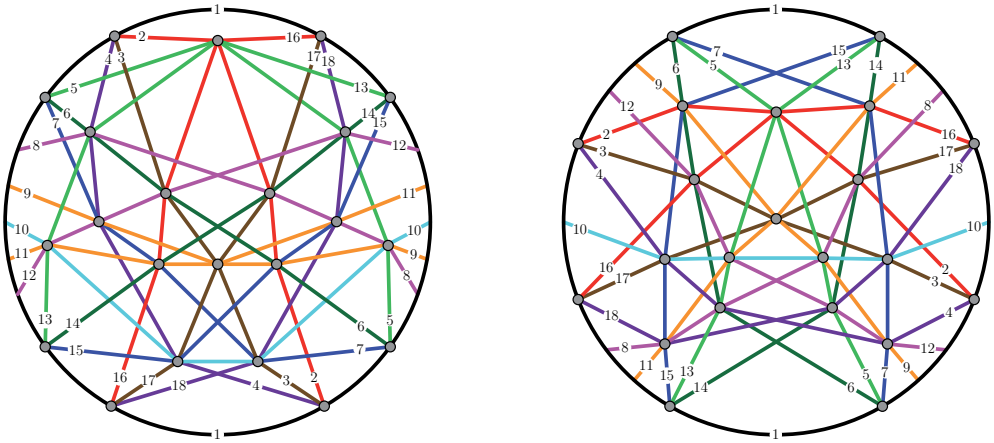

Figure 1: Two rather different (18_4)-configuration which are combinatorially equivalent.

needed several months of CPU-time in [11], and only one hour with our Java implementation). Finally, our algorithm opens new opportunities of research based on enumeration to answer several open questions on configurations. Among others: Is there a symmetrical topological (19_4)-configuration [8, p. 169]? What is the smallest topological (n_5)-configuration? Is there a geometric (19_4)-configuration?

Topological configurations are pseudoline arrangements, or rank 3 oriented matroids. We assume the reader to have some basic knowledge on these topics — see [2, 1, 9].

2 Preliminaries

What do we need? Our guideline and motivation in the study of configurations is the question of the existence of geometric (n_k)-configurations. In particular, it is challenging to determine, for a given k, which is the first n for which geometric (n_k)-configurations exist. For $k = 3$, Pappus' configuration is the first example (with two other combinatorially distinct (9_3)-configurations). For $k = 4$, it was known for a long time that no combinatorial (n_4)-configurations exist when $n \leq 12$. However, the smallest geometric configuration was unknown until the first author proved with Lars Schewe that no topological (n_4)-configurations exist when $n \leq 16$ [4], that the only combinatorial (17_4)-configuration which is topologically realizable is not geometrically realizable [5], and that there exists a geometric (18_4)-configuration [5].

The method presented in [5] makes it possible to decide whether a combinatorial configuration is geometrically realizable. The goal of our algorithm is to limit the research to combinatorial configurations which are already topologically realizable. In other words, for given n and k, we want to enumerate all topological (n_k)-configurations under combinatorial equivalence.

Three equivalence relations. There are three distinct notions of equivalence on topological configurations.

The finest notion is the usual notion of equivalence between pseudoline arrangements in the projective plane: two configurations are *topologically equivalent* if there is an homeomorphism of their underlying projective planes that sends one arrangement onto the other.

The coarsest notion is combinatorial equivalence: two (n_k)-configurations are *combinatorially equivalent* if they realize the same combinatorial (n_k)-configuration.

The intermediate notion is based on the graph of admissible mutations. Remember that a *mutation* in a pseudoline arrangement is a local transformation of the arrangement where only one pseudoline ℓ moves, sweeping a single vertex v of the remaining arrangement. It only changes the position of the crossings of ℓ with the pseudolines incident to v. If those crossings are all 2-crossings, the mutation does not perturb the k-crossings of the arrangement, and thus produces another topological (n_k)-configuration. We say that such a mutation is *admissible*. Two configurations are *mutation equivalent* if they belong to the same connected component of the graph of admissible mutations.

Figure 2: An admissible mutation.

Obviously, topological equivalence implies mutation equivalence, which in turn implies combinatorial equivalence. The reciprocal implications are wrong.

Note that the topological equivalence between two (n_k)-configurations can be tested in $\Theta(n^3)$ time. Indeed, since the topological configurations are embedded

on the projective plane, the images of two pseudolines under an isomorphism of the projective plane determine the images of all the other pseudolines. Thus, the complexity to compute the topological equivalence classes among p topological (n_k)-configurations is in $\Theta(p^2n^3)$. Both combinatorial and mutation equivalences are however much harder to decide computationally.

In order to limit unnecessary computation, we can use topological (resp. mutation, resp. combinatorial) invariants associated to topological configurations. If two configurations have distinct invariants, they cannot be equivalent. Reciprocally, if they share the same invariant, it provides us information on the possible isomorphism between these two configurations. For example, the *face size vector* (the number of faces of each size) is a topological invariant, and the *distribution of the triangles* on the pseudolines is a combinatorial invariant (a triangle of a configuration (P, L) is a triple of points of P which are pairwise related by pseudolines of L).

As an illustration, the two (18_4)-configurations depicted in Figure 1 are combinatorially equivalent (the labels on the pseudolines provide a combinatorial isomorphism) but not topologically equivalent (the left one has 22 quadrangles and 2 pentagons, while the right one has 23 quadrangles). In fact, one can even check that they are not mutation equivalent.

What do we obtain? Our algorithm can enumerate all topological (n_k)-configurations up to either topological or combinatorial equivalence. In order to maintain a reasonable computation space and time, the main idea is to focus on the relative positions of the points of the configurations and to ignore at first the relative positions of the other crossings among the pseudolines. In other words, to work modulo mutation equivalence.

More precisely, we first enumerate at least one representative of each mutation equivalence class of topological (n_k)-configuration. From these representatives, we can derive:

1. all topological (n_k)-configurations up to topological equivalence: we explore each connected component of the mutation graph with our representatives as starting nodes.

2. all combinatorial (n_k)-configurations that are topologically realizable: we reduce the result modulo combinatorial equivalence.

3 Representation of arrangements

Simple configurations. A topological configuration (P, L) is *simple* if no three pseudolines of L meet at a common point except if it is a point of P. Since any topological (n_k)-configuration can be arbitrarily perturbed to become simple, we only consider simple topological (n_k)-configurations. Once we obtain all simple topological (n_k)-configurations, it is usual to obtain all (non-necessarily simple) topological (n_k)-configurations up to topological equivalence by exploring the mutation graph, and we do not report on this aspect.

In a simple (n_k)-configuration (P, L), there are two kinds of intersection points among pseudolines of L: the points of P, which we also call *k-crossings*, and the other points, which we call *2-crossings*. Each pseudoline of L contains k k-crossings and $n-1-k(k-1)$ 2-crossings. In total, a simple (n_k)-configuration has n k-crossings and $\binom{n}{2} - n(\binom{k}{2} - 1)$ 2-crossings.

Segment length distributions. A *segment* of a topological configuration (P, L) is the portion of a pseudoline of L located between two consecutive points of P. If (P, L) is simple, a segment contains no k-crossing except its endpoints, but may contain some 2-crossings. The *length* of a segment is the number of 2-crossings it contains.

The lengths of the segments of a pseudoline of L form a k-partition of $n-1-k(k-1)$. We call a *maximal representative* of a k-tuple the lexicographic maximum of its orbit under the action of the dihedral group (*i.e.* rotations and reflections of the k-tuple). We denote by Π the list of all distinct maximal representatives of the k-partitions of $n-1-k(k-1)$, ordered lexicograhically. For example, when $k=4$ and $n=17$, $\Pi = [4,0,0,0]$, $[3,1,0,0]$, $[3,0,1,0]$, $[2,2,0,0]$, $[2,0,2,0]$, $[2,1,1,0]$, $[2,1,0,1]$, $[1,1,1,1]$.

A suitable representation. We represent the projective plane as a disk where we identify antipodal boundary points. Given a simple topological (n_k)-configuration (P, L), we fix a representation of its underlying projective plane which satisfies the following properties (see Figure 3 left).

The leftmost point of the disk (which is identified with the rightmost point of the disk) is a point of P, which we call the *basepoint*. The k pseudolines of L passing through the basepoint are called the *frame pseudolines*, while the other $n-k$ pseudolines of L are called *working pseudolines*. The frame pseudolines decompose the projective plane into k connected regions which we call *frame regions*. A crossing is a *frame* crossing if it involves a frame pseudoline and a *working* crossing if it involves only working pseudolines.

The boundary of the disk is a frame pseudoline, which we call the *baseline*. We furthermore assume that the segment length distribution Λ on the top half-circle appears in Π (*i.e.* is its own maximal representative), and that no maximal representative of the segment length distribution of a pseudoline of L appears before Λ in Π. In particular, the leftmost segment of the baseline is a longest segment of the configuration.

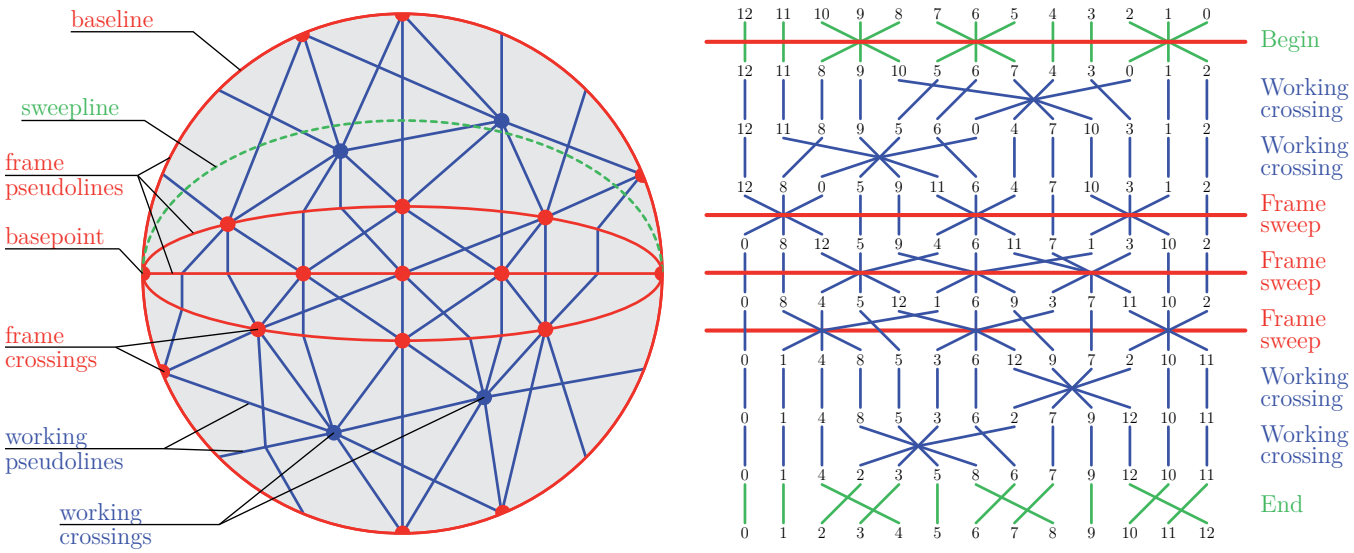

Figure 3: Suitable representation of a (17_4)-configuration, and the corresponding wiring diagram.

Wiring diagram and allowable sequence. Another interesting representation of our (n_k)-configuration is the *wiring diagram* [6] of its working pseudolines (see Figure 3 right). It is obtained by sending the basepoint to infinity in the horizontal direction. The frame pseudolines are k horizontal lines, and the $n-k$ working pseudolines are vertical wires. The orders of the working pseudolines on a horizontal line sweeping the wiring diagram from top to bottom form the so-called *allowable sequence* of the working arrangement, as defined in [6].

4 Description of the algorithm

Main idea. Let us recall here the main idea of the algorithm: to enumerate (n_k)-configurations, we focus on the relative positions of the k-crossings and ignore at first the relative positions of the 2-crossings. More precisely, we first generate at least one (but as few as possible) representative of each mutation equivalent class of (n_k)-configurations.

Sweeping process. The algorithm sweeps the projective plane to construct a topological (n_k)-configuration. The *sweepline* sweeps the configuration from the baseline on the top of the disk to the baseline on the bottom of the disk. It always passes through the basepoint and always completes the configuration into an arrangement of $n+1$ pseudolines. In other words, it sweeps the k frame regions from top to bottom, reaching the separating frame pseudoline when passing from one frame region to the next one, and discovers along the way the working pseudolines. Except those located on the frame pseudolines, we assume that the crossings of the configuration are reached one after the other by the sweepline. After the sweepline swept a crossing, we remember the order of its intersections with the working pseudolines. In other words, the sweeping process provides us with the allowable sequence of the working pseudolines of our configuration.

Since any admissible mutation is irrelevant for us, we only focus on the steps of the sweeping process where our sweepline sweeps a k-crossing. Thus, two different events can occur: when the sweepline sweeps a working k-crossing, and when the sweepline sweeps a frame pseudoline. In the later case, we sweep simultaneously $k-1$ frame k-crossings (each involving the frame pseudoline and $k-1$ working pseudolines), and $n-1-k(k-1)$ frame 2-crossings (each involving the frame pseudoline and a working pseudoline). Between two such events, the sweepline may sweep working 2-crossings which are only taken into account when we reach a new event. Let us repeat again that the precise positions of these working 2-crossings is irrelevant in our enumeration.

To obtain all possible solutions, we maintain a priority queue with all subconfigurations which have been constructed so far, remembering for each one (i) the order of the working pseudolines on the current sweepline, (ii) the number of frame and working k-crossings and 2-crossings which have already been swept on each working pseudoline, (iii) the length of the segment currently swept by the sweepline, and (iv) the history of the sweeps which have been performed to reach this subconfiguration. At each step, we remove the first subconfiguration from the priority queue, and insert all admissible subconfigurations which can arise after sweeping a new working k-crossing or a new frame pseudoline. We finally accept a configuration once we have swept k frame pseudolines and $n-k(k-1)-1$ working k-crossings.

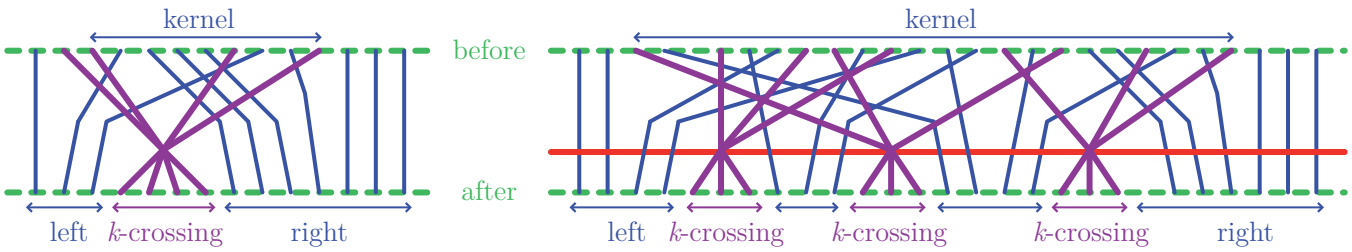

Figure 4: Sweeping a working k-crossing (left) and a frame pseudoline (right).

Any subconfiguration considered during the algorithm is a potential (n_k)-configuration. Throughout the process, we make sure that any pair of working pseudolines cross at most once, that the number of frame pseudolines (resp. of working k-crossings) already swept never exceeds k (resp. $n-1-k(k-1)$), and that the total number of working 2-crossings never exceeds $(n-2k)(n-1-k(k-1))/2$. Furthermore, on each pseudoline, the number of frame and working k-crossings (resp. 2-crossings) already swept never exceeds k (resp. $n-1-k(k-1)$), the number of working 2- and k-crossings already swept never exceeds $n-1-k(k-1)$, and the segment currently swept is not longer than the leftmost segment of the baseline.

Initialization. We initialize our algorithm sweeping the baseline. We only have to choose the distribution of the lengths of the segments on the baseline. The possibilities are given by the list Π of maximal representatives of k-partitions of $n-1-k(k-1)$.

Sweep a working k-crossing. If we decide to sweep a working k-crossing, we have to choose the k working pseudolines which intersect at this k-crossing, and the direction of the other working pseudolines.

Since we are allowed to perform any admissible mutation, we can assume that all the pseudolines located to the left of the leftmost pseudoline of the working k-crossing, and all those located to the right of the rightmost pseudoline of the working k-crossing do not move.

We say that the pseudolines located between the leftmost and the rightmost pseudolines of the working k-crossing form the *kernel* of the working k-crossing. We have to choose the positions of the pseudolines of the kernel after the flip: each pseudoline of the kernel either belongs to the working k-crossing, or goes to its left, or goes to its right (see Figure 4 left).

A choice of directions for the kernel is admissible provided that (i) each pseudoline involved in the k-crossing can still accept a working k-crossing; (ii) each pseudoline of the kernel can still accept as many working 2-crossings as implied by the choice of directions for the kernel; (iii) no segment becomes longer than the leftmost segment of the baseline; and (iv) any two pseudolines which are forced to cross by the choice of directions for the kernel did not cross earlier (*i.e.* they still form an inversion on the sweepline before we sweep the working k-crossing).

Sweep a frame pseudoline. If we decide to sweep a frame pseudoline, we have to choose the $(k-1)^2$ working pseudolines involved in one of the $k-1$ frame k-crossings, and the direction of the other working pseudolines.

As before, we can assume that a pseudoline does not move if it is located to the left of the leftmost pseudoline involved in one of the $k-1$ frame k-crossings, or to the right of the rightmost pseudoline involved in one of the $k-1$ frame k-crossings. Otherwise, we can perform admissible mutations to ensure this situation.

The other pseudolines form again the *kernel* of the frame sweep, and we have to choose their positions after the flip. Each pseudoline of the kernel either belongs to one of the $k-1$ frame k-crossings, or can choose among k possible directions: before the first frame k-crossing, or between two consecutive frame k-crossings, or after the last frame k-crossing (see Figure 4 right).

As before, a choice of directions for the kernel is admissible if (i) each pseudoline involved (resp. not involved) in one of the $k-1$ frame k-crossings can still accept a frame k-crossing (resp. a frame 2-crossing); (ii) each pseudoline of the kernel can still accept as many working 2-crossings as implied by the choice of directions for the kernel; (iii) no segment becomes longer than the leftmost segment of the baseline; and (iv) any two pseudolines which are forced to cross by the choice of directions for the kernel did not cross earlier (*i.e.* they still form an inversion on the sweepline before we sweep the frame pseudoline).

Sweep the last frame region. Our sweeping process finishes once we have swept $n-1-k(k-1)$ working k-crossings and k frame pseudolines. Each resulting subconfiguration should still be completed into a topological (n_k)-configuration with some necessary remaining 2-crossings. More precisely, we need to add on each working pseudoline as many working 2-crossings as its number of inversions in the permutation given by

the working pseudolines on the final sweepline, without creating segments that are too long.

All the constructed configurations are guaranteed to be valid topological (n_k)-configurations. To make sure that we indeed obtain the representation presented in Section 3, we remove each configuration (P, L) in which the maximal representative of the segment length distribution of a pseudoline of L appears in the list Π before the segment length distribution of its baseline.

Parallelization. To close this description, we observe that our algorithm is easily parallelizable on different computers since it is a dynamic research in a tree. We did not use parallelization to obtain the current results, but it will certainly be an important advantage of the algorithm for exploring the question of finding the first integer n for which topological (n_5)-configurations exist.

5 Results

Check former results. As a first application, our algorithm enables us to check easily all former enumerations of topologically realizable combinatorial (n_k)-configurations. The Java implementation developed by the second author finds all (n_k)-configurations in less than a minute[1] when $k = 3$ and $n \leq 11$, or when $k = 4$ and $n \leq 17$. In particular, we checked that there is no topological (n_4)-configuration when $n \leq 16$ [4], and that there is a single combinatorial (17_4)-configuration which can be realized by (several) topological (17_4)-configurations, but which cannot be realized geometrically [5]. When $k = 4$ and $k = 18$, we reconstructed the 16 combinatorial equivalence classes of topological (18_4)-configurations depicted in [5, Figure 6]. To obtain this result, our implementation needed about one hour[1], compared to months of CPU-time used in [11]. The two (18_4)-configurations presented in Figure 1, which are combinatorially equivalent but not mutation equivalent, occured while we were reducing the list of (18_4)-configurations up to combinatorial equivalence, using as a first reduction a certain invariant of mutation equivalence.

Obtain new results. Our algorithm can furthermore be used to derive new enumerative results. To answer the question of the existence of geometric (19_4)-configurations, our first step is to compute the complete list of combinatorial (19_4)-configurations that admit a topological realization. We found 4028 combinatorially distinct topologically realizable (19_4)-configurations (222 of which are self-dual). This task has been accomplished by our algorithm within 16 days of CPU-time[1]. We underline again that we did not use the extended list of all 269224652 combinatorial (19_4)-configurations computed in [10] to obtain this result. A detailed investigation and analysis of this result will be published in a subsequent paper.

[1]Computation times on a double core processor on 2.4 GHz.

Acknowledgements

The first author thanks Leah Berman from the University of Alaska Fairbanks for discussions about the subject. He also thanks three colleagues from the Universidad Nacional Autónoma de México, namely Rodolfo San Augustin Chi, Ricardo Strausz Santiago, and Octavio Paez Osuna, for many stimulating discussions about various different earlier versions of the presented algorithm during his one year sabbatical stay (2008/2009) in México City.

References

[1] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. M. Ziegler. *Oriented matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1999.

[2] J. Bokowski. *Computational oriented matroids*. Cambridge University Press, Cambridge, 2006.

[3] J. Bokowski, B. Grünbaum, and L. Schewe. Topological configurations (n_4) exist for all $n \geq 17$. *European J. Combin.*, 30(8):1778–1785, 2009.

[4] J. Bokowski and L. Schewe. There are no realizable 15_4- and 16_4-configurations. *Rev. Roumaine Math. Pures Appl.*, 50(5-6):483–493, 2005.

[5] J. Bokowski and L. Schewe. On the finite set of missing geometric configurations (n_4). To appear in *Computational Geometry: Theory and Applications*, 2011.

[6] J. E. Goodman and R. Pollack. Allowable sequences and order types in discrete and computational geometry. In *New trends in discrete and computational geometry*, volume 10 of *Algorithms Combin.*, pages 103–134. Springer, Berlin, 1993.

[7] B. Grünbaum. Connected (n_4) configurations exist for almost all n—second update. *Geombinatorics*, 16(2):254–261, 2006.

[8] B. Grünbaum. *Configurations of points and lines*, volume 103 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2009.

[9] D. E. Knuth. *Axioms and hulls*, volume 606 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1992.

[10] O. Páez Osuna and R. San Agustín Chi. The combinatorial (19_4) configurations. Preprint, 2011.

[11] L. Schewe. *Satisfiability Problems in Discrete Geometry*. PhD thesis, Technische Universität Darmstadt, 2007.