ENUMERATING TOPOLOGICAL (n_k)-CONFIGURATIONS

JÜRGEN BOKOWSKI AND VINCENT PILAUD

ABSTRACT. An (n_k)-configuration is a set of n points and n lines in the projective plane such that their 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, for given n and k, all topological (n_k)-configurations up to combinatorial isomorphism, without enumerating first all combinatorial (n_k)-configurations. We apply this algorithm to confirm efficiently a former result on topological (18_4)-configurations, from which we obtain a new geometric (18_4)-configuration. Preliminary results on (19_4)-configurations are also briefly reported.

1. INTRODUCTION

A *point – line configuration* is a set P of *points* and a set L of *lines* together with an *incidence relation*, where two points of P can be incident with at most one line of L and two lines of L can be incident with at most one point of P. Throughout the paper, we only consider *connected* configurations, where any two elements of $P \sqcup L$ are connected via a path of incident elements. An *isomorphism* (resp. a *duality*) between two configurations (P, L) and (P', L') is an incidence-preserving map from $P \sqcup L$ to $P' \sqcup L'$ which sends points to points and lines to lines (resp. which exchanges points and lines).

According to the underlying structure, we distinguish three different levels of configurations, in increasing generality:

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

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

Combinatorial configuration: Just an abstract incidence structure (P, L) as described above, with no additional geometric structure.

In this paper, we focus on *regular* configurations, *i.e.* whose incidence relation is regular. More precisely, 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. We have represented three famous 3-regular configurations in Figure 1 to illustrate the previous definitions.

Point – line configurations have a long history in discrete 2-dimensional geometry. We refer to Branko Grünbaum's recent monograph [Grü09] for a detailed treatment of the topic and for historical references. As underlined in this monograph, the current study of regular configurations focusses on the following two problems:

Vincent Pilaud was partially supported by grant MTM2011-22792 of the Spanish Ministerio de Ciencia e Innovación and by European Research Project ExploreMaps (ERC StG 208471).

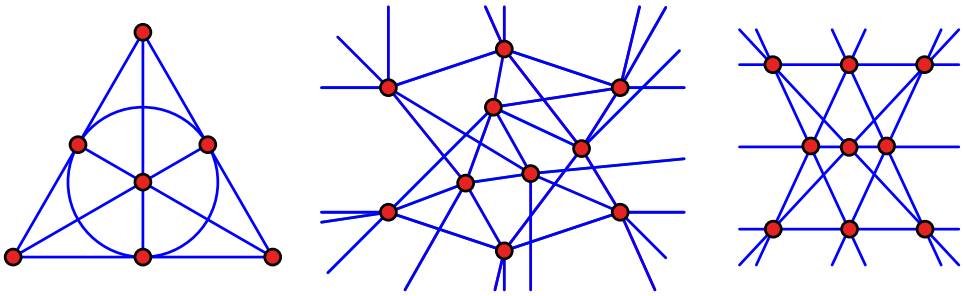


FIGURE 1. (Left) Fano's configuration is a combinatorial (7_3)-configuration but is not realizable topologically or geometrically. (Center) Kantor's topological (10_3)-configuration is not realizable geometrically. (Right) Pappus' configuration is a geometric (9_3)-configuration.

(i) For a given k, determine for which values of n do geometric, topological, and combinatorial (n_k)-configurations exist.
(ii) Enumerate and classify (n_k)-configurations for given k and n.

In particular, it is challenging to determine the minimal value n for which (n_k)-configurations exist and to enumerate these minimal configurations.

For $k \in \{3, 4\}$, the existence of (n_k)-configurations is almost completely understood. When $k = 3$, combinatorial (n_3)-configurations exist for every $n \geq 7$, but topological and geometric (n_3)-configurations exist only for every $n \geq 9$. When $k = 4$, combinatorial (n_4)-configurations exist iff $n \geq 13$, topological (n_4)-configurations exist iff $n \geq 17$ [BS05, BGS09] and geometric (n_4)-configurations exist iff $n \geq 18$ [Grü06, BS11], with the possible exceptions of 19, 22, 23, 26, 37 and 43. For $k \geq 5$, the situation is more involved, and the existence of combinatorial, topological and geometric (n_k)-configurations is not determined in general.

Concerning the enumeration, an important effort has been done on combinatorial (n_3)- and (n_4)-configurations. Table 1 provides the known values of the number $c_k(n)$ of combinatorial (n_k)-configurations up to isomorphism. The first row of this table ($k = 3$) appeared in [BBP00], except $c_3(19)$ which was announced later on in [PBM+04, p.275]. The second row ($k = 4$) appeared in [BB99, p.34], except $c_4(19)$ which was only computed recently in [OC12].

In this paper, we are interested in the numbers $t_k(n)$ and $g_k(n)$ of topological and geometric (n_k)-configurations up to isomorphism. To obtain these numbers, one method is to select the topologically or geometrically realizable configurations among the list of all combinatorial (n_k)-configurations. For example, the numbers

n	≤ 6	7	8	9	10	11	12	13	14	15	16	17	18	19
$c_3(n)$	0	1	1	3	10	31	229	2 036	21 399	245 342	3 004 881	38 904 499	530 452 205	7 640 941 062
$c_4(n)$	0	0	0	0	0	0	0	1	1	4	19	1 972	971 171	269 224 652

TABLE 1. The number $c_k(n)$ of combinatorial (n_k)-configurations up to isomorphism.

n	$c_3(n)$	$t_3(n)$	$g_3(n)$
≤ 6	0	0	0
7	1	0	0
8	1	0	0
9	3	3	3
10	10	10	9
11	31	31	31
12	229	229	229
13	2 036	?	?

n	$c_4(n)$	$t_4(n)$	$g_4(n)$
≤ 12	0	0	0
13	1	0	0
14	1	0	0
15	4	0	0
16	19	0	0
17	1 972	1	0
18	971 191	16	**2**
19	269 224 652	**4 028**	?

TABLE 2. The numbers $t_k(n)$ of topological (n_k)-configurations and $g_k(n)$ of geometric (n_k)-configurations up to isomorphism.

$t_3(n)$ and $g_3(n)$ presented in Table 2 were derived from a careful study of the corresponding combinatorial configurations (see the historical remarks and references in [Grü09]). In [Sch07], Lars Schewe provided a general method to study the topological realizability of a combinatorial configuration using satisfiability solvers, and obtained the numbers $t_4(17) = 1$ and $t_4(18) = 16$. In [BS11], Jürgen Bokowski and Lars Schewe studied the geometric realizability of a combinatorial configuration. This question is clearly an instance of the existential theory of the reals (ETR): it boils down to determining whether a set of polynomial equalities and inequalities admits a solution in the reals (indeed, the inclusion of a point in a line can be tested by a polynomial equation). Using the construction sequences presented in [BS11], the complexity of this instance of ETR can be decreased significantly. With this method, Jürgen Bokowski and Lars Schewe showed that the only combinatorial (17_4)-configuration which is topologically realizable is not geometrically realizable and they exhibited a geometric (18_4)-configuration.

Table 2 summarizes the values of $t_3(n), g_3(n), t_4(n)$ and $g_4(n)$ known up-to-date (we have additionally included our results in bold letters; see below). This table indicates a clear difference of behavior between 3- and 4-regular configurations. On the one hand, when $k = 3$, most of the combinatorial (n_3)-configurations are topologically and geometrically realizable for small values of n. For $n \leq 12$, the only counter-examples are the Fano (7_3)-configuration, the Möbius-Kantor (8_3)-configuration, and Kantor's (10_3)-configuration — see Figure 1 (left & center). On the other hand, when $k = 4$, it is not reasonable to look for geometric (n_4)-configurations among all combinatorial (n_4)-configurations. To further extend our knowledge on geometric configurations, it thus seems crucial to limit our research to those combinatorial configurations which are already topologically realizable.

Motivated by this observation, we present an algorithm for generating, for given n and k, all topological (n_k)-configurations up to isomorphism, 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 isomorphic topological configurations, and thus to maintain a reasonable computation space and time.

We have 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 outline three applications of our algorithm. First, the algorithm is interesting in its own right. Before describing some special methods for constructing topological configurations, Branko Grünbaum writes in [Grü09, 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 all values of $t_4(n)$, for $n \leq 18$, obtained in earlier papers. We can use for that a single method and reduce considerably the computation time (*e.g.* the computation of the (18_4)-configurations needed several months of CPU-time in [Sch07], and only one hour with our JAVA implementation). Finally, this algorithm enables us to compute all $t_4(19) = 4028$ isomorphism classes of topological (19_4)-configurations.

As an application of our enumeration results, we studied in detail the possible geometric realizations of the topological (18_4)-configurations. Using a MAPLE implementation of the construction sequence method of Jürgen Bokowski and Lars Schewe [BS05], we obtain that there are precisely 2 geometric (18_4)-configurations: the first (18_4)-configuration constructed in [BS05], plus an additional one which appears for the first time in this paper. In contrast, deriving the list of geometric (19_4)-configurations from the list of topological (19_4)-configurations still requires some computational effort and is left to a subsequent paper.

The first section of this paper is devoted to the enumeration algorithm for isomorphism classes of topological configurations. The second section presents the application to the enumeration of geometric (18_4)-configurations.

Topological configurations are pseudoline arrangements, or rank 3 oriented matroids. We assume the reader to have some basic knowledge on these topics. We refer to [Bok06, BLS+99, Knu92] for introductions.

2. Topological configurations

In this section, we present our algorithm to generate all isomorphism classes of topological (n_k)-configurations, for given n and k. Let us insist again on the crucial fact that we do not need to enumerate first all combinatorial (n_k)-configurations. The main idea of the algorithm is to sweep the projective plane to construct a topological (n_k)-configuration (P, L), only focussing on the "relative positions of the points of P" and ignoring at first the "relative positions of the other crossings of the pseudolines of L" (precise definitions are given in Section 2.1). This strategy enables us to identify along the way some isomorphic topological configurations, and thus to maintain a reasonable computation space and time.

2.1. Three equivalence relations.

There are three distinct notions of equivalence on topological configurations.

The finest notion is the usual notion of topological 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 that of combinatorial equivalence: two (n_k)-configurations are *combinatorially equivalent* if they are isomorphic as combinatorial (n_k)-configurations.

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 one can be obtained from the other by a (possibly empty) sequence of admissible mutations followed by an homeomorphism of the underlying projective space.

FIGURE 2. An admissible mutation.

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

As an illustration, the two (18_4)-configurations depicted in Figure 3 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.

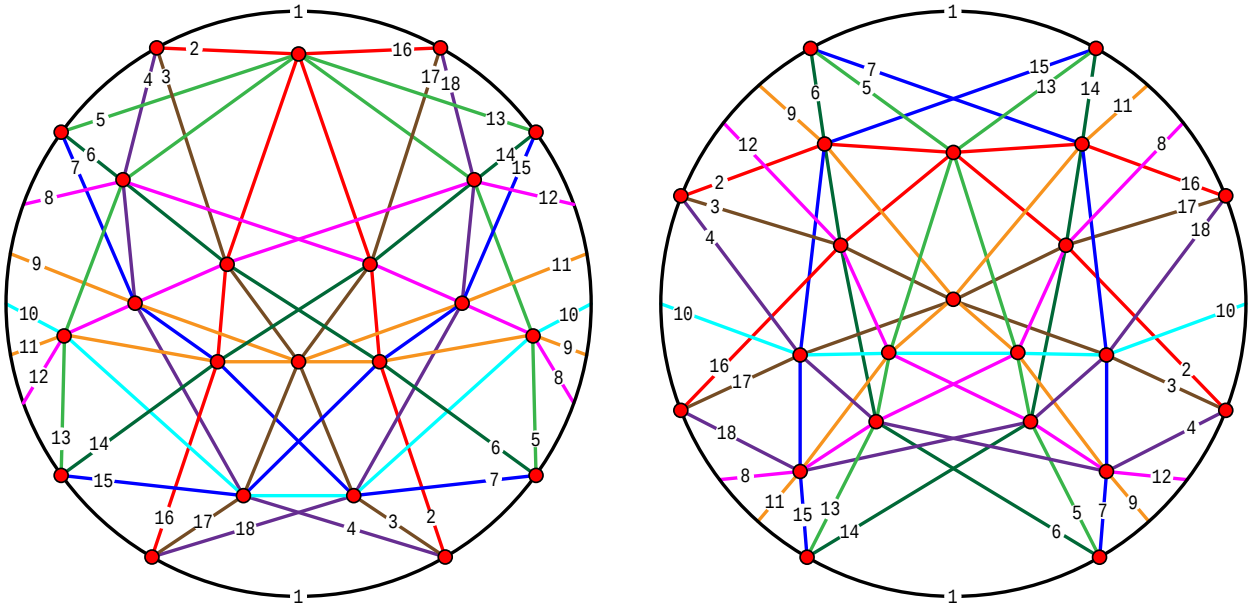


FIGURE 3. Two (18_4)-configurations which are combinatorially equivalent but neither mutation nor topologically equivalent.

2.2. **Representation of arrangements.** In this section, we state certain properties of configurations that we can assume without loss of generality. In particular, we choose a suitable representation of our pseudoline arrangements that we will use for the description of the algorithm in Section 2.3.

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 intersection 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 circular sequence of the segment lengths on a pseudoline of L forms 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 $n = 17$, we have $\Pi = [4,0,0,0]$, $[3,1,0,0]$, $[3,0,1,0]$, $[2,2,0,0]$, $[2,0,2,0]$, $[2,1,1,0]$, $[2,1,0,1]$, $[1,1,1,1]$.

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 4 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 *base point*. The k pseudolines of L passing through the base point 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 *base line*. 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 base line is a longest segment of the configuration.

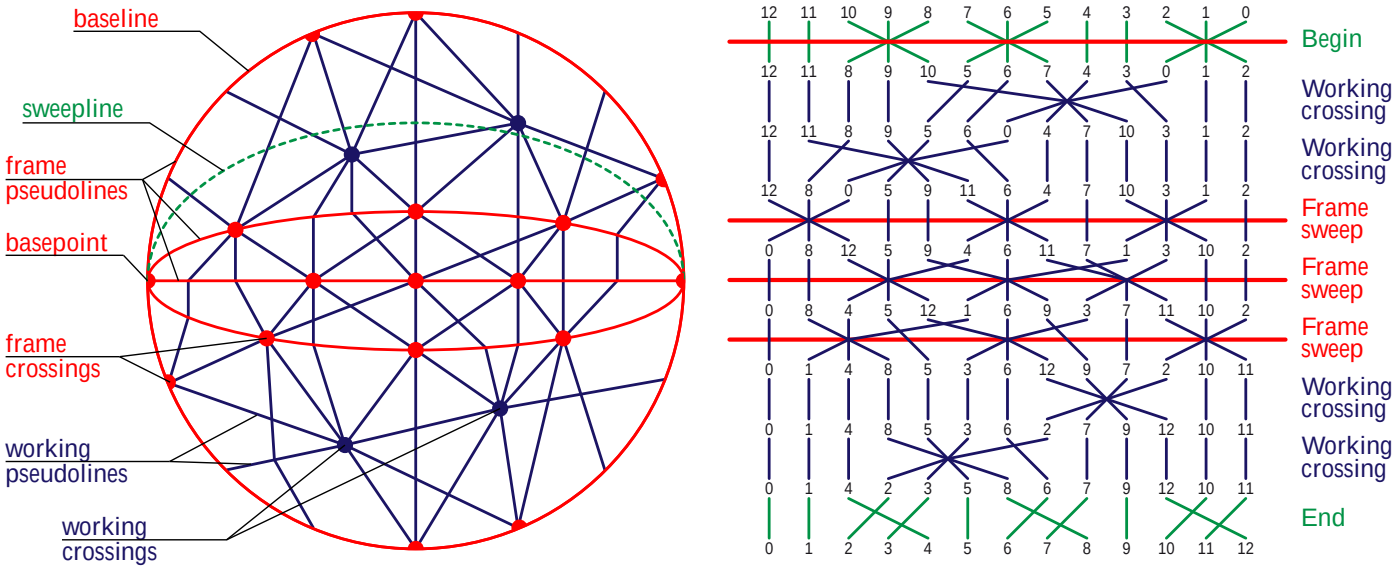


FIGURE 4. Suitable representation of a (17_4)-configuration, and the corresponding wiring diagram.

WIRING DIAGRAM AND ALLOWABLE SEQUENCE — Another interesting representation of our (n_k)-configuration is the *wiring diagram* [GP93] of its working pseudolines (see Figure 4 right). It is obtained by sending the base point 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 [GP93].

2.3. **Description of the algorithm.** 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 as defined in Section 2.1.

More precisely, we first enumerate at least one representative of each mutation equivalence class of topological (n_k)-configurations. 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.

Since our motivation is to study geometric (n_k)-configurations, we are only interested by point (2). We discuss a relatively efficient approach to test combinatorial equivalence in Section 2.4. In this section, we give details on the different steps in our algorithm.

Sweeping process — Our algorithm sweeps the projective plane to construct a topological (n_k)-configuration. The *sweep line* sweeps the configuration from the base line on the top of the disk to the base line on the bottom of the disk. Inside each frame region, it always passes through the base point and always completes the configuration into an arrangement of $n+1$ pseudolines. When it switches from one frame region to the next one, it coincides with the separating frame pseudoline. Along the way, it sweeps completely all 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 sweep line. After the sweep line 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 admissible mutations are irrelevant for us, we only focus on the moments when our sweep line sweeps a k-crossing. Thus, two different events can occur:

- when the sweep line sweeps a working k-crossing, and
- when the sweep line 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 sweep line 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 stack with all subconfigurations which have been constructed so far, remembering for each one:

(i) the order of the working pseudolines on the current sweep line,
(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 sweep line, and
(iv) the history of the sweeps performed to reach this subconfiguration.

At each step, we remove the first subconfiguration from the stack, 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.

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 base line.

We now detail individually each step of the algorithm.

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

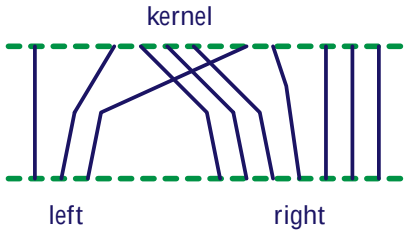


FIGURE 5. Sweeping a working k-crossing.

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 5).

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 base line; 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 sweep line 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 6).

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;

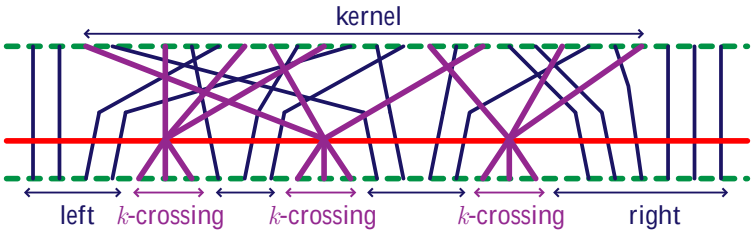


FIGURE 6. Sweeping a frame pseudoline (right).

(iii) no segment becomes longer than the leftmost segment of the base line; 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 sweep line 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 sweep line, without creating segments that are too long.

After this last selection, all the constructed configurations are finally guaranteed to be valid topological (n_k)-configurations. To make sure that we indeed obtain the representation presented in Section 2.2, 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 base line.

2.4. **Testing combinatorial equivalence.** In Section 2.1, we have seen three equivalence relations between topological (n_k)-configurations: combinatorial, mutation and topological equivalence. As explained in Section 2.3, our algorithm outputs at least one representative per mutation equivalence class of topological (n_k)-configurations. However, we can obtain more than one representative per class, and two topological (n_k)-configurations which are not mutation equivalent can still be combinatorially equivalent. We thus need to reduce the output of our algorithm.

Note that the topological equivalence between two (n_k)-configurations (P_1, L_1) and (P_2, L_2) can be tested in $\Theta(n^3)$ time. Indeed, since the topological configurations are embedded on the projective plane, the matchings between P_1 and P_2 and between L_1 and L_2 induced by an homeomorphism mapping (P_1, L_1) to (P_2, L_2) are determined by the images of any two distinguished pseudolines ℓ, ℓ' of L_1. Therefore, for each of the $\Theta(n^2)$ possible choices for the images of ℓ, ℓ', we can test in linear time whether this choice yields or not an homeomorphism between (P_1, L_1) and (P_2, L_2). Both combinatorial and mutation equivalences are however harder to decide computationally. We focus here on methods and heuristics to quickly test combinatorial equivalence.

In order to limit unnecessary computation, we make use of *combinatorial invariants* associated to configurations. If two configurations have distinct invariants, they cannot be combinatorially equivalent. Reciprocally, if they share the same invariant, it provides us with information on the possible combinatorial isomorphisms

between these two configurations. The invariants we have chosen are the *clique* and *coclique distributions*. We furthermore need a *multiscale invariant* technique, based on the notion of *derivation* of a combinatorial invariant. We introduce these notions and methods in the next paragraphs.

CLIQUES AND COCLIQUES — Let (P, L) be a combinatorial configuration. For $j \geq 3$, define a *j-clique* of (P, L) to be any set of j points of P which are pairwise related by lines of L. For any point p of P, let $\gamma_j(p)$ be the number of j-cliques containing p, and let $\gamma(p) := (\gamma_j(p))_{j\geq 3}$. The *clique distribution* of (P, L) is the multiset $\gamma(P) := \{\!\{\gamma(p) \mid p \in P\}\!\}$.

Similarly, a *j-coclique* of (P, L) is a set of j lines of L which are pairwise intersecting at points of P. For any line ℓ of L, let $\delta_j(\ell)$ be the number of j-cocliques containing ℓ, and let $\delta(\ell) := (\delta_j(\ell))_{j\geq 3}$. The *coclique distribution* of (P, L) is the multiset $\delta(L) := \{\!\{\delta(\ell) \mid \ell \in L\}\!\}$. In other words, the coclique distribution of (P, L) is the clique distribution of its dual configuration (L, P).

The pair $(\gamma(P), \delta(L))$ of clique and coclique distributions of the configuration (P, L) is a natural and powerful combinatorial invariant of (P, L).

DERIVATION OF COMBINATORIAL INVARIANTS — Let (P, L) be a (n_k)-configuration. Assume that $\gamma : P \to X$ and $\delta : L \to Y$ are two functions from the point set and the line set of (P, L) respectively to arbitrary sets X and Y, such that the multisets $\gamma(P) := \{\!\{\gamma(p) \mid p \in P\}\!\} \subset X$ and $\delta(L) := \{\!\{\delta(\ell) \mid \ell \in L\}\!\} \subset Y$ are combinatorial invariants of (P, L). The clique and coclique distributions are typical examples of such functions γ and δ. Observe that we again abuse notation: the functions γ and δ usually depend on the configuration (P, L), but we consider that this dependence is clear from the context. Note however that the target sets X and Y of γ and δ do not depend upon (P, L).

While reducing a set of configurations up to combinatorial equivalence, such a pair of combinatorial invariants $(\gamma(P), \delta(L))$ can be used in two different ways:

(i) either to separate classes of combinatorial isomorphism: two configurations with different invariants cannot be combinatorially equivalent;
(ii) or to guess combinatorial isomorphisms: an isomorphism between two configurations should respect the invariants γ and δ.

It often happens however that the pair of combinatorial invariants $(\gamma(P), \delta(L))$ is not precise enough neither to distinguish two configurations, nor to guess a combinatorial isomorphism between them. It occurs when many points (resp. many lines) of a configuration (P, L) get the same image under γ (resp. under δ). Two fundamentally different cases can lead to this situation. On the one hand, the configuration (P, L) can have a large automorphism group. In this case, points (resp. lines) in a common orbit under the automorphism group cannot be distinguished combinatorially, and thus no invariant can speed up the isomorphism test. On the other hand, it could also be that the combinatorial invariant $(\gamma(P), \delta(L))$ is not precise enough to distinguish the neighborhood properties of the points with the same image under γ (resp. the lines with the same image under δ). In the later case, we can construct a new pair of combinatorial invariants which refines $(\gamma(P), \delta(L))$, taking into account the neighborhoods of points and lines in the configuration. We call these invariants the *derivatives* of γ and δ and denote them γ' and δ'.

The *derivative* of the invariant $\gamma : P \to X$ is the function $\gamma' : L \to X^k$ which associates to a line ℓ of L the multiset $\gamma'(\ell) := \{\!\{\gamma(p) \mid p \in P, p \in \ell\}\!\}$. Intuitively,

the image $\gamma'(\ell)$ of a line ℓ contains all the combinatorial information carried by γ concerning the points of P contained in ℓ. Similarly, the *derivative* of the invariant $\delta : L \to Y$ is the function $\delta' : P \to Y^k$ which associates to a point p of P the multiset $\delta'(p) := \{\!\{\delta(\ell) \mid \ell \in L, p \in \ell\}\!\}$. The pair $(\delta'(P), \gamma'(L))$ is a pair of combinatorial invariants as defined previously, and it refines the previous pair $(\gamma(P), \delta(L))$.

If this new invariant is still not precise enough, we can consider higher order derivatives $\gamma^{(u)} := (\gamma^{(u-1)})'$ and $\delta^{(u)} := (\delta^{(u-1)})'$ of the initial invariants. We obtain this way a family of refinements of $(\gamma(P), \delta(L))$. Of course, these invariants ultimately carry the same combinatorial information. We use this family in the following multiscale technique.

MULTISCALE INVARIANTS — The main idea of our reduction process is to use derivative invariants in a multiscale process. Consider a set $\mathcal{C}$ of configurations that we want to reduce up to combinatorial equivalence. Assume that $\gamma : P \to X$ and $\delta : L \to Y$ are two functions defining a pair of combinatorial invariants $(\gamma(P), \delta(L))$ of a configuration (P, L). We separate the configurations of $\mathcal{C}$ into classes with distinct invariants, which we can consider independently. We now compute the derivative invariants $(\delta'(P), \gamma'(L))$ for each configuration (P, L). For a given class, we then have three possible situations:

(1) If the derivative invariants $(\delta'(P), \gamma'(L))$ are not the same for all configurations (P, L) of the class, we split the class into refined subclasses and reiterate the refinement (computing one more derivative).
(2) If the derivative invariants $(\delta'(P), \gamma'(L))$ are the same for all configurations (P, L) of the class but determine more information on the possible isomorphisms between configurations of the class than the original invariants $(\gamma(P), \delta(L))$, then we reiterate the refinement.
(3) Otherwise, the derivative invariants $(\delta'(P), \gamma'(L))$, as well as any further derivative, provide the same combinatorial information as the original invariants $(\gamma(P), \delta(L))$. Thus, we stop the refinement process and start a brute-force search for possible isomorphisms between the remaining configurations in the class. The efficiency of this brute-force search depends on the quality of the combinatorial information provided by the invariants $(\gamma(P), \delta(L))$.

This process can be seen as a multiscale process: typically, some invariants provide sufficiently information to deal with certain classes of $\mathcal{C}$, while other classes require far more precision (obtained by derivatives) to be reduced.

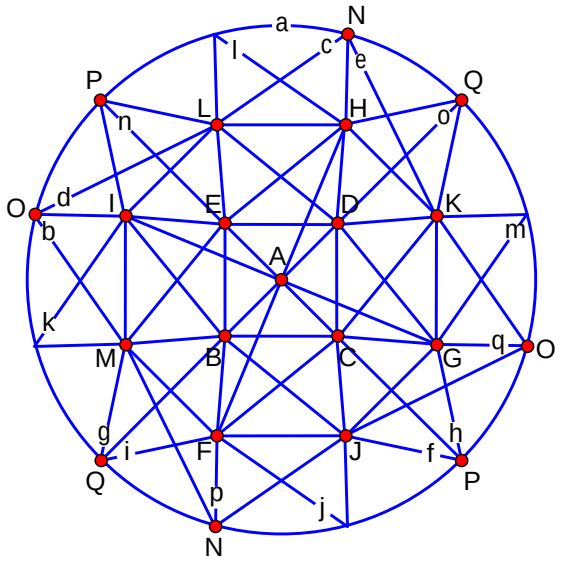
Using this multiscale technique, starting from the clique and coclique distributions of configurations, we managed to reduce the 69 991 topological (19_4)-configurations produced by our sweeping algorithm into 4 028 classes of combinatorial equivalence in about one hour[1].

2.5. **Results.** We present in this section the results of our algorithm. First, it enables us to check efficiently all former enumerations of topological (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 $n \leq 16$ [BS05], and that there is a single topological (17_4)-configuration

[1]Computation times on a 2.4 GHz Intel Core 2 Duo processor with 4Go of RAM.

up to combinatorial isomorphism. This configuration is represented in Figure 7, and labeled in such a way that:

- the quarter-turn rotation which generates the symmetry group of the picture is the permutation (A)(B,C,D,E)(F,G,H,I)(J,K,L,M)(N,O)(P,Q); and
- the permutation (A,a)(B,b)... (P,p)(Q,q) is a self-polarity of the topological configuration.



lines	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
	N	F	I	H	G	B	E	D	C	B	E	D	C	A	A	A	A
points	P	J	M	L	K	I	H	G	F	E	D	C	B	C	B	F	G
in lines	O	O	N	O	N	P	Q	P	Q	L	K	J	M	P	Q	N	O
	Q	M	L	K	J	J	M	L	K	F	I	H	G	E	D	H	I

FIGURE 7. The topological (17_4)-configuration [BGS09].

When $k = 4$ and $n = 18$, we reconstructed the 16 combinatorial equivalence classes of topological (18_4)-configurations obtained in [Sch07] with satisfiability solvers. See [BS11, Figure 6] for a description of these configurations. To obtain this result, our implementation needed about one hour[1], compared to several months of CPU-time required in [Sch07]. The two (18_4)-configurations presented in Figure 3, which are combinatorially equivalent but not mutation equivalent, occurred 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 defined in [BS12]. In the next section, we present two combinatorially distinct geometric (18_4)-configurations obtained from the list of topological (18_4)-configurations.

Finally, we want to report on preliminary results concerning the enumeration of topological (19_4)-configurations, which initially motivated our work. In about 15 days of computation time[1], we obtained the complete list of topological (19_4)-configurations:

[1]Computation times on a 2.4 GHz Intel Core 2 Duo processor with 4Go of RAM.

Result 1. *There are precisely* 4 028 *topological* (19_4)*-configurations up to combinatorial equivalence. Among them,* 222 *are self-dual.*

From this list, we can immediately extract examples of topological (19_4)-configurations with non-trivial symmetry groups, closing along the way an open question of Branko Grünbaum [Grü09, p. 169, Question 5]. The next step is naturally to study the possible geometric realizations of all these topological (19_4)-configurations. This work in progress still requires an important computational effort and will be reported in a subsequent paper.

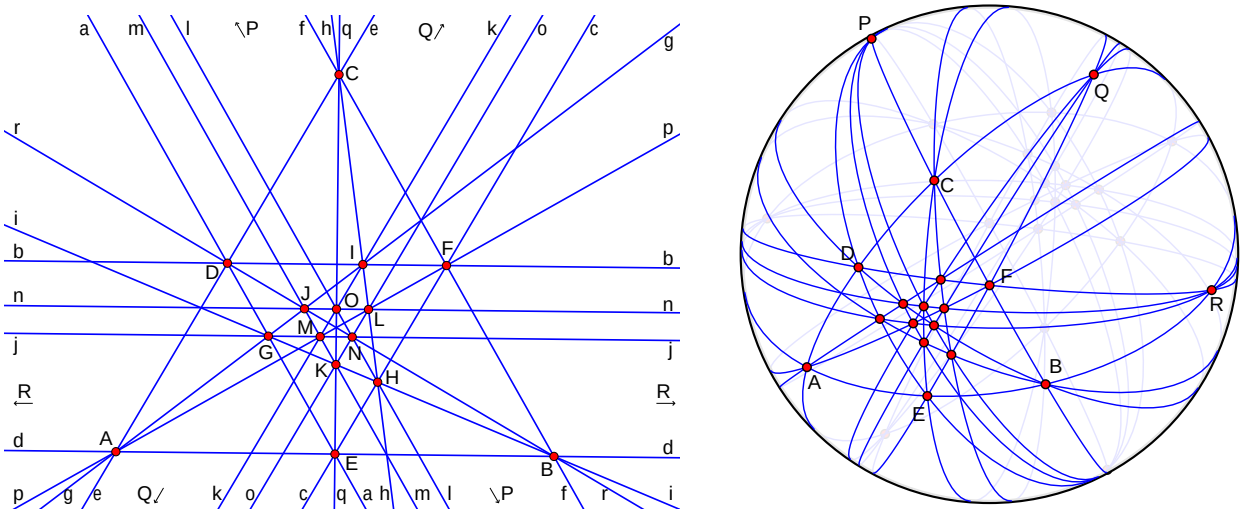
3. Application to geometric (18_4)-configurations

As an application of the enumeration of topological configurations, we derive all isomorphism classes of geometric (18_4)-configurations. To obtain it, we implemented in MAPLE the construction sequence method of Jürgen Bokowski and Lars Schewe [BS11]. Among the 16 topological (18_4)-configurations (first generated by Lars Schewe [Sch07] and now confirmed by our JAVA program), only 8 are compatible with Pappus' and Desargues' Theorem. Starting from these remaining configurations, we run our MAPLE code and obtain the following result:

Result 2. *There are precisely two geometric* (18_4)*-configurations up to combinatorial isomorphism.*

The first geometric (18_4)-configuration was obtained in [BS05, Section 4].

In Figure 8, we have labeled its points A, ..., R and lines a, ..., r in such a way that the permutation (A,a)(B,b) ... (Q,q)(R,r) is a self-duality of the configuration.



lines	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
	D	F	E	A	C	B	A	C	B	G	I	H	J	L	K	A	C	B
points	G	I	H	E	D	F	G	I	H	M	O	N	M	O	N	F	E	D
in lines	E	D	F	B	A	C	J	L	K	N	M	O	K	J	L	L	K	J
	P	R	Q	R	Q	P	I	H	G	R	Q	P	P	R	Q	M	O	N

FIGURE 8. Bokowski and Schewe's geometric (18_4)-configuration [BS05].

The automorphism group of the combinatorial configuration is generated by the permutations:

(A,B,C)(D,E,F)(G,H,I)(J,K,L)(M,N,O)(P,Q,R)
(A)(K)(B,C,L,J)(D,F,I,R)(E,Q,M,G)(H,O,N,P)
(A)(K)(B,L)(C,J)(D,I)(E,M)(F,R)(G,Q)(H,N)(O,P)

and is isomorphic to the symmetric group on 4 elements. Together with the self-duality (A,a)(B,b)... (Q,q)(R,r), the automorphism group of the Levi graph of the configuration is thus isomorphic to $\mathfrak{S}_4 \times \mathbb{Z}_2$. Observe that only the first permutation (A,B,C)... (P,Q,R) and the duality (A,a)(B,b)... (Q,q)(R,r) are geometrically visible, while the other generators of the automorphism group of the combinatorial configuration are not isometries of the geometric configuration of Figure 8. In Figure 9, we have performed a projective transformation of the configuration of Figure 8 (sending the four 3-valent points in Figure 8 to a square). The last generator (A)(K)(B,L)... (O,P) then becomes a central symmetry in the new geometric (18_4)-configuration.

The realization space of this configuration consists of two points, both expressed with coordinates in $\mathbb{Q}\left[1+\sqrt{5}\right]$.

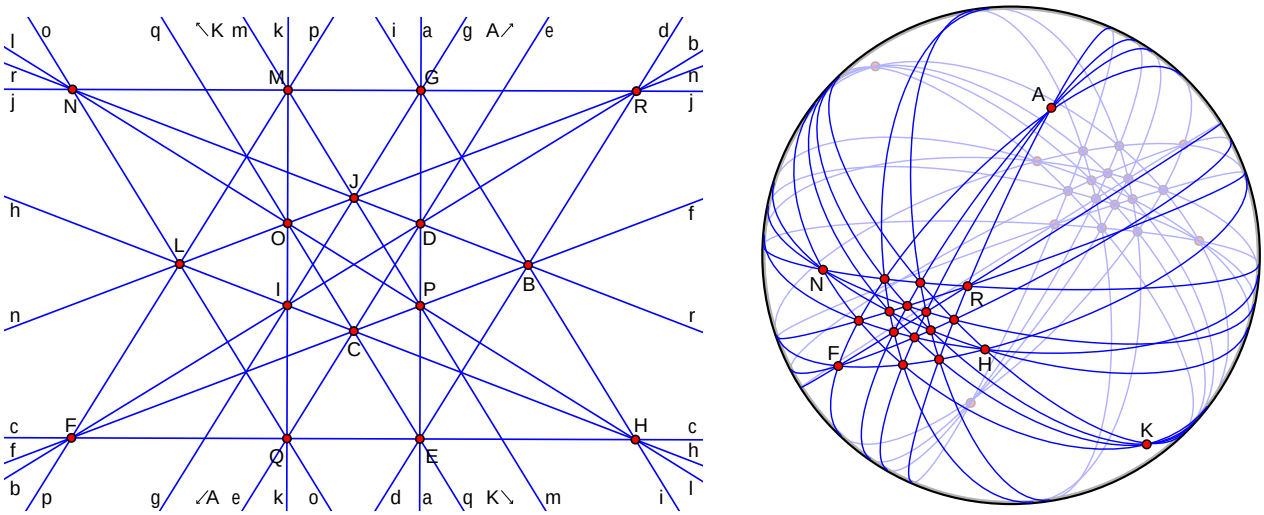


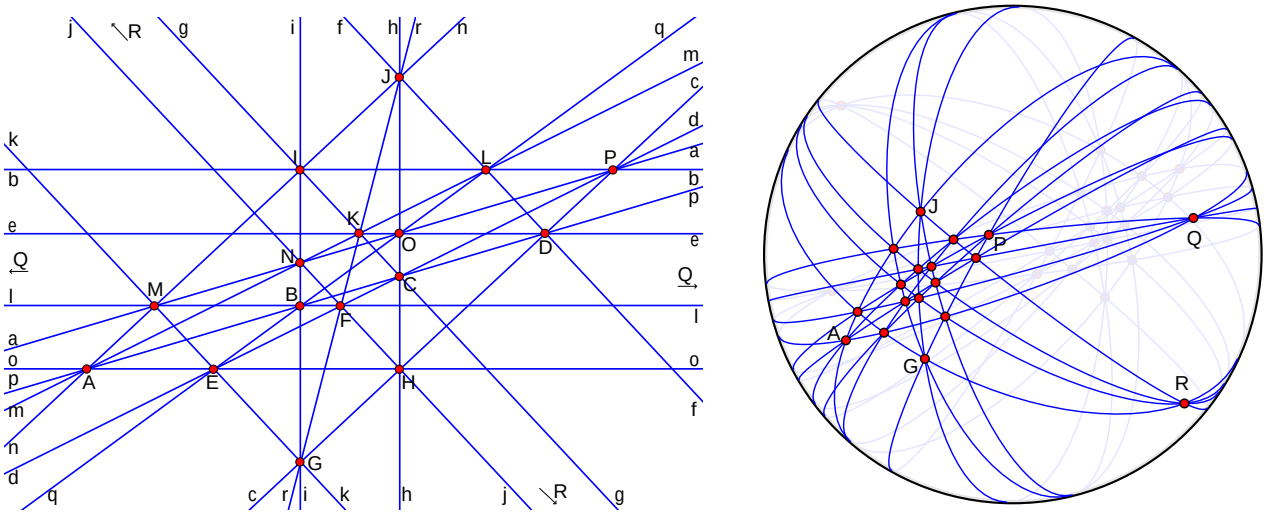
FIGURE 9. Another geometric realization of Bokowski and Schewe's geometric (18_4)-configuration [BS05] of Figure 8.

The second geometric (18_4)-configuration is a result of our MAPLE code and appears for the first time in this paper.

In Figure 10, we have labeled its points A, ..., R and lines a, ..., r in such a way that the permutation (A,a)(B,b)... (Q,q)(R,r) is a self-polarity of the configuration.

The automorphism group of the combinatorial configuration is generated by the permutation (Q)(R)(A,P)(B,O)(C,N)(D,M)(E,L)(F,K)(G,J)(H,I). Together with the self-polarity (A,a)(B,b)... (Q,q)(R,r), the automorphism group of the Levi graph of the configuration is thus isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_2$. This group is completely realized in the geometric representation of Figure 10.

The realization space of this configuration consists of two points, both expressed with coordinates in $\mathbb{Q}\left[\sqrt[3]{108+12\sqrt{93}}\right]$



lines	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
	M	I	D	C	D	D	C	C	N	N	M	M	N	M	H	D	B	F
points	N	L	H	F	O	L	K	H	I	F	E	B	K	I	E	C	E	G
in lines	O	P	G	E	K	J	I	J	G	H	G	F	L	J	A	B	L	J
	P	Q	P	P	Q	R	R	O	B	R	R	Q	A	A	Q	A	O	K

FIGURE 10. The new geometric (18_4)-configuration.

To conclude, we want to emphasize that the discovery of the first (18_4)-configuration of Figure 8 inspired Branko Grünbaum to find a new family of $(6m_4)$-configurations, for any $m \geq 3$ (see [Grü09, Chapter 3, p. 171] and Figure 11). This raises the following appealing open question:

Problem 3. *Generalize our second geometric* (18_4)*-configuration of Figure 10 to obtain another new infinite family of geometric* (n_4)*-configurations.*

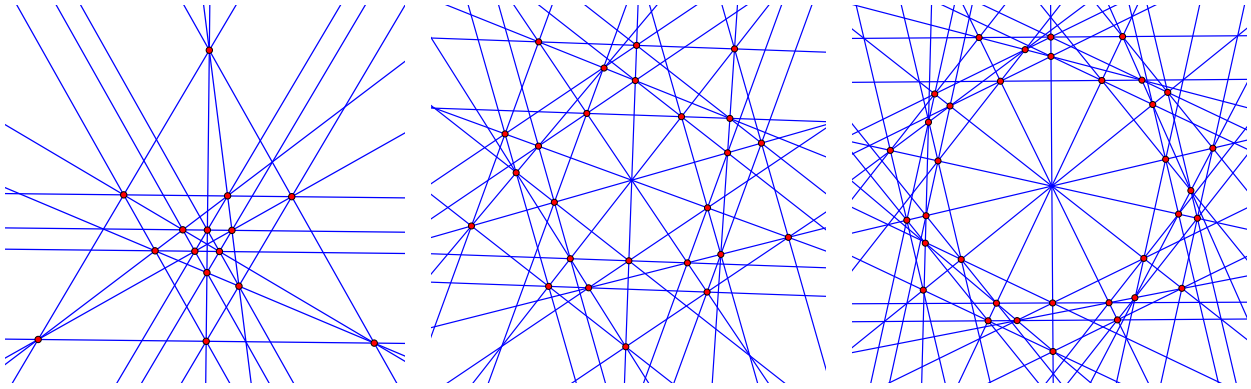


FIGURE 11. The $(6m)$-family inspired by the geometric (18_4)-configuration of Figure 8.

For example, we have been able to derive from the second geometric (18_4)-configuration of Figure 10 a family of $((18+17m)_4)$-configurations. Unfortunately, the set $18+17\mathbb{N}$ does not intersect the set $\{19, 22, 23, 26, 37, 43\}$ of values n for which no (n_4)-configuration is known.

Acknowledgements

The first author thanks three colleagues from the Universidad Nacional Autónoma de México, namely Ricardo Strausz Santiago, Rodolfo San Augustin Chi, 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. We also thank Leah Berman from the University of Alaska Fairbanks for valuable discussions and comments about the subject. We are grateful to Branko Grünbaum, Tomaž Pisanski, and Gunnar Brinkmann for encouragements and helpful communications. As frequent users, we are indebted to the development team of the geometric software CINDERELLA, in particular Jürgen Richter-Gebert and Ulrich Kortenkamp. Finally, we thank two anonymous referees for their comments and suggestions on the presentation.

References

[BB99] Anton Betten and Dieter Betten. Tactical decompositions and some configurations v_4. *J. Geom.*, 66(1-2):27–41, 1999.

[BBP00] Anton Betten, Gunnar Brinkmann, and Tomaž Pisanski. Counting symmetric configurations v_3. *Discrete Appl. Math.*, 99(1-3):331–338, 2000. Proceedings of the *5th Twente Workshop on Graphs and Combinatorial Optimization* (Enschede, 1997).

[BGS09] Jürgen Bokowski, Branko Grünbaum, and Lars Schewe. Topological configurations (n_4) exist for all $n \geq 17$. *European J. Combin.*, 30(8):1778–1785, 2009.

[BLS+99] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented matroids*, volume 46 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, second edition, 1999.

[Bok06] Jürgen Bokowski. *Computational oriented matroids*. Cambridge University Press, Cambridge, 2006.

[BS05] Jürgen Bokowski and Lars Schewe. There are no realizable 15_4- and 16_4-configurations. *Rev. Roumaine Math. Pures Appl.*, 50(5-6):483–493, 2005.

[BS11] Jürgen Bokowski and Lars Schewe. On the finite set of missing geometric configurations (n_4). To appear in *Computational Geometry: Theory and Applications*, 2011.

[BS12] Jürgen Bokowski and Ricardo Strausz Santiago. A manifold associated to a topological (n_k)-configuration. Preprint, 2012.

[GP93] Jacob E. Goodman and Richard 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.

[Grü06] Branko Grünbaum. Connected (n_4) configurations exist for almost all n—second update. *Geombinatorics*, 16(2):254–261, 2006.

[Grü09] Branko Grünbaum. *Configurations of points and lines*, volume 103 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2009.

[Knu92] Donald E. Knuth. *Axioms and hulls*, volume 606 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1992.

[OC12] Octavio Páez Osuna and Rodolfo San Agustín Chi. The combinatorial (19_4) configurations. *Ars Math. Contemp.*, 5(2):231–237, 2012.

[PBM+04] Tomaž Pisanski, Marko Boben, Dragan Marušič, Alen Orbanić, and Ante Graovac. The 10-cages and derived configurations. *Discrete Math.*, 275(1-3):265–276, 2004.

[Sch07] Lars Schewe. *Satisfiability Problems in Discrete Geometry*. PhD thesis, Technische Universität Darmstadt, 2007.

Technische Universität Darmstadt
E-mail address: juergen.bokowski@googlemail.com

CNRS & LIX, École Polytechnique, Palaiseau
E-mail address: vincent.pilaud@lix.polytechnique.fr
URL: http://www.lix.polytechnique.fr/~pilaud/