Algorithms for Center and Tverberg Points*

Pankaj K. Agarwal
Dept. Computer Science
Duke University, Durham,
NC 27708-0129, USA
pankaj@cs.duke.edu

Micha Sharir
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel, and
Courant Inst. of Math. Sci.
New York University
michas@post.tau.ac.il

Emo Welzl
Inst. Theoretische Informatik
ETH Zürich
CH-8092 Zürich, Switzerland
emo@inf.ethz.ch

ABSTRACT

We present a near-quadratic algorithm for computing the center region of a set of n points in three dimensions. This is nearly tight in the worst case since the center region can have $\Omega(n^2)$ complexity. We then consider the problem of recognizing whether a given point q is a colored Tverberg point of a set of n colored points in the plane, and present the first polynomial-time algorithm for this problem.

Categories and Subject Descriptors: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*Geometrical problems and computations*

General Terms: Algorithms,Theory

Keywords: Center Points, Colored Tverberg's Theorem, j-Facets

1. INTRODUCTION

Given a set S of n points in $\mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, the *depth* of x with respect to S is the minimum number of points of S contained in a closed halfspace whose bounding hyperplane passes through x. The set of all points of depth k is called the *depth-k region* of S. The region, $c_k(S)$, of points of depths at least k can be written as $\bigcap_{h\in\mathcal{H}_{n-k+1}} h$, with $\mathcal{H}_j = \mathcal{H}_j(S)$ the set of all closed halfspaces (bounded by hyperplanes) that contain at least j points from S.[1]

Note $c_1(S) = \mathrm{conv}(S)$.

If $j > dn/(d+1)$, then any $d+1$ halfspaces in $\mathcal{H}_j$ have a point of S in common, and Helly's theorem (cf. [9, 14]) implies $\bigcap_{h\in\mathcal{H}_j} h \neq \emptyset$. That is, if $n-k+1 > dn/(d+1)$, or equivalently, if $k \leq \lceil n/(d+1)\rceil$, the region $c_k(S)$ is nonempty (as it was first observed by Rado [17]).

Points of depth at least $\lceil n/(d+1)\rceil$ are called *center points* of S, and the region $c_{\lceil n/(d+1)\rceil}(S)$—which we know is nonempty—is called the *center region* of S, denoted by $c(S)$.

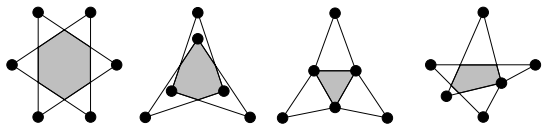

Figure 1. Sets of six points, with their center regions, the points of depth at least 2. Equivalently, these are the intersections of all halfplanes containing at least 5 points. (The meaning of the extra edges indicated will be explained later.)

Relatively little progress has been made on the algorithmic issues related to center points. Teng [20] showed that if d is part of the input, the problem of determining whether a given point is a center point of S is coNP-Complete. Jadhav and Mukhopadhyay [10] gave a linear-time algorithm for computing a center point in $\mathbb{R}^2$. Matoušek later developed an $O(n\log^4 n)$-time algorithm for computing the center region of a set of n points in $\mathbb{R}^2$ (see possible improvement according to [5, Remark pg. 427]). Naor and Sharir [16] gave an $O(n^2\,\mathrm{polylog}(n))$ algorithm for computing a center point in $\mathbb{R}^3$; very recently, Chan [5] supplied a randomized $O(n\log n + n^{d-1})$ algorithm for this problem in $\mathbb{R}^d$. For computing the center region in $\mathbb{R}^3$ there is a naïve cubic algorithm, that can be improved to $O(n^{5/2+\varepsilon})$ by running a k-level construction algorithm in the dual and computing the convex hull of the obtained vertices. Clarkson *et al.* [7] proposed a simple algorithm for computing an approximate center point of a set of points in arbitrary dimensions.

A generalization of a center point is the so-called Tverberg point. Let r be a positive integer. A partition of S into r disjoint subsets $S_1,\ldots,S_r$ is called a *Tverberg partition* if $\bigcap_{i=1}^r \mathrm{conv}(S_i) \neq \emptyset$, and a point lying in the intersection is called a *Tverberg point* (or an *r-divisible point*). Tverberg [21] proved that if $|S| > (d+1)(r-1)$, then a Tverberg partition always exists. Subsequent to his original proof, several simpler proofs have been proposed; see the book by Matoušek [14] and the survey paper by Kalai [11] for a history

*Work by P.A. and M.S. was supported by a grant from the U.S.-Israeli Binational Science Foundation. Work by P.A. was also supported by NSF under grants CCR-00-86013 EIA-98-70724, EIA-99-72879, EIA-01-31905, and CCR-02-04118. Work by M.S. was also supported by NSF Grants CCR-97-32101 and CCR-00-98246, by a grant from the Israel Science Fund (for a Center of Excellence in Geometric Computing), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University.

[1]For a proof, note that if $x \notin h \in \mathcal{H}_{n-k+1}$, then the depth is at most $k-1$ since the complement of h contains a closed halfspace with x on its boundary and containing at most $n-(n-k+1) = k-1$ points from S. On the other hand, if x lies on the boundary of a closed halfspace with at most $k-1$ points, then the complement contains a closed halfspace with at least $n-k+1$ points and thus $x \notin \bigcap_{h\in\mathcal{H}_{n-k+1}} h$.

SCG'04, June 9–11, 2004, Brooklyn, New York, USA.

of the problem. Note that any halfspace containing an r-divisible point contains at least r points of S, so by setting $r = \lceil n/(d+1) \rceil$, we get that every r-divisible point of S is also a center point of S. For $d = 2$ and n a multiple of 3, the converse is also true, i.e., every center point of S is also a $\lceil n/(d+1) \rceil$-divisible point of S, but it is not true for $d \geq 3$ [2]. Teng [20] showed that if d is part of the input, then the problem of determining whether a given point is an r-divisible point of S is NP-Complete. On the other hand if d is fixed, a polynomial-time algorithm (with running time $n^{O(d^2)}$) can be obtained by modifying Tverberg's existence proof. For $d = 2$ and n a multiple of 3, a Tverberg point can be computed in linear time using the algorithm for computing a center point [10], and we can determine in $O(n \log n)$ time whether a point is a Tverberg point.

Bárány, Füredi and Lovász [4] suggested a *colored version of Tverberg's theorem*, which was then established by Živaljević and Vrećica [22] in arbitrary dimensions. The planar case allowed a quantitative improvement, provided by Bárány and Larman [3]. They showed that given a planar set S, which is the disjoint union of three sets R, B, G consisting respectively of n red points, n blue points, and n green points (in general position), there exists a partition of S into n pairwise-disjoint triples $S_1, \ldots, S_n$, where each triple consists of one point of each of R, B, G, so that $\bigcap_{i=1}^{n} \mathrm{conv}(S_i) \neq \emptyset$. In fact, their argument is constructive and yields an $O(n^6)$-time algorithm for computing such a partition. However no polynomial-time algorithm is known for determining whether a given point is a colored Tverberg point of S.

Our results. We present two main results in this paper. First, given a set S of n points in $\mathbb{R}^3$, we describe an $O(n^{2+\varepsilon})$ algorithm to compute the center region $c(S)$ (cf. Section 2). In fact, we show that for any k, the depth-k region of S has $O(n^2)$ complexity, and that it can be computed in $O(n^{2+\varepsilon})$ time. By performing a binary search, we can compute the region of maximum depth in $O(n^{2+\varepsilon})$ time.

Next, given a S of $3n$ points in $\mathbb{R}^2$, which is the disjoint union of three sets R, B, and G consisting of n points each, we present a polynomial-time algorithm to determine whether a point q is a colored Tverberg point of S (cf. Section 3). The running time of the algorithm is $O(n^{11})$.

2. CENTER REGION IN 3D

We first make sure that the problem of computing the the center region, or $\bigcap_{h \in \mathcal{H}_j} h$ in general, is a finite problem. Then we discuss the structure of the center region in the dual, before we proceed to the description of the algorithm.

2.1 j-Facets and the center region

We assume $|S| \geq d+1$ and that S is in general position, i.e., no $d+1$ points lie on a common hyperplane. A *j-facet* is an oriented simplex spanned by d points in S that has exactly j points from S on its positive side[2]. $\overline{\mathcal{H}}_j$ is the set of closed halfspaces that contain j points and have d points on their boundary. Clearly, these are exactly the halfspaces induced by $(j-d)$-facets. Figure 1 displays all 5-facets (orientations omitted) of the respective point sets.

LEMMA 2.1. *For any set S of $n \geq d+1$ points in general position in $\mathbb{R}^d$ and any positive integer $j \leq n$ we have:*

(1) $c_{n-j+1}(S) = \bigcap_{h \in \mathcal{H}_j} h = \bigcap_{h \in \overline{\mathcal{H}}_j} h$.

(2) $\bigcap_{h \in \mathcal{H}_j} h$ *is a convex polytope (not necessarily of full dimension) with at most* $2\binom{n}{d-1}$ *facets, each of which is contained in some $(j-d)$-facet of S.*

Proof: (1) Consider first the set $\overline{\mathcal{H}}_{\geq j} := \bigcup_{i \geq j} \overline{\mathcal{H}}_i$. Since $\overline{\mathcal{H}}_{\geq j} \subseteq \mathcal{H}_j$, the inclusion

$$\bigcap_{h \in \mathcal{H}_j} h \subseteq \bigcap_{h \in \overline{\mathcal{H}}_{\geq j}} h$$

readily follows. It can be shown that, for each $h \in \mathcal{H}_j$, there are d halfspaces in $\overline{\mathcal{H}}_{\geq j}$ whose intersection is contained in h. It then follows that all $h \in \mathcal{H}_j \setminus \overline{\mathcal{H}}_{\geq j}$ are redundant. Hence, $\bigcap_{h \in \mathcal{H}_j} h = \bigcap_{h \in \overline{\mathcal{H}}_{\geq j}} h$; hence $\bigcap_{h \in \mathcal{H}_j} h$ is a polyhedron whose facets are determined by (i.e., lie in the affine hull of) some of the halfspaces $h \in \overline{\mathcal{H}}_{\geq j}$. On the other hand, every halfspace in $\overline{\mathcal{H}}_i$, for $i > j$, contains an intersection of d halfspaces in $\mathcal{H}_j$ in its interior, so none of these halfspaces can determine a facet of the polyhedron and thus can be omitted in its definition. Assertion (1) of the lemma follows.

(2) We have already shown that $P := \bigcap_{h \in \mathcal{H}_j} h$ is an intersection of a finite number of halfspaces, and since it is contained in $\mathrm{conv}(S)$, it is bounded and thus a polytope. We further reduce the number of constraints in $\overline{\mathcal{H}}_j$ that are needed to determine P.

If P is empty, we are done since then it is the intersection of $d+1 \leq 2\binom{n}{d-1}$ halfspaces in $\overline{\mathcal{H}}_j$ (recall $n \geq d+1$). Otherwise consider some set K of $d-1$ points in S. Either no $(j-d)$-facet contains K, or all but two halfspaces in $\overline{\mathcal{H}}_j$ with K on their boundary are redundant. Since the number of $(d-1)$-element subsets of S is $\binom{n}{d-1}$, the asserted upper bound on the number of facets follows.

Now consider a halfspace $h \in \overline{\mathcal{H}}_j$, and the $(j-d)$-facet it contains. We rotate the bounding hyperplane of h about any $d-1$ of the points of the $(j-d)$-facet, while keeping the $(j-d)$-facet in its halfspace. It follows that no portion of the hyperplane bounding h other than its $(j-d)$-facet can be part of P. □

2.2 The structure of the center region

Let S be a set of n points in $\mathbb{R}^d$. A standard duality transform [9] maps a point p in $\mathbb{R}^d$ to a hyperplane p^* in $\mathbb{R}^d$ and vice versa so that the above/below relationships between the points and hyperplanes are preserved, i.e., if p lies below (resp., above, on) a hyperplane h, then the dual hyperplane p^* lies below (resp., above, on) the point h^*. Using this dual transform we map S to a set S^* of n hyperplanes in $\mathbb{R}^d$. The *level* of a point $x \in \mathbb{R}^d$ with respect to S^* is the number of hyperplanes in S^* lying below x. All points on the same face of $\mathcal{A}(S^*)$ have the same level. For a given integer $0 \leq k < n$, the *k-level* of $\mathcal{A}(S^*)$, denoted as $\Lambda_k(S^*)$, is the closure of all facets of $\mathcal{A}(S^*)$ whose level is k. $\Lambda_k(S^*)$ is the graph of a continuous, piecewise-linear $(d-1)$-variate function.

By construction, the dual of a point x whose depth is k ($k \leq n/2$) with respect to S is a hyperplane that lies between $\Lambda_k(S^*)$ and $\Lambda_{n-k}(S^*)$. More precisely if we let L_k (resp., U_k) denote the lower (resp., upper) convex hull of $\Lambda_k(S^*)$, then x^* separates U_k and L_{n-k}. Hence, the dual of the center region $c(S)$ is the region lying between $U_{\lceil \frac{n}{d+1} \rceil}$ and $L_{\lfloor \frac{dn}{d+1} \rfloor}$. In order to compute $c(S)$, it suffices to describe the algorithm for computing the convex hull of a level in $\mathcal{A}(S^*)$.

The following lemma follows from the observation that the intersection line of any $d-1$ hyperplanes of S^* intersects $\mathrm{conv}(\Lambda_k(S^*))$ in at most two points (the dual analogy of Lemma 2.1).

[2] To be precise, exactly j points from S on the positive side of its affine hull.

LEMMA 2.2. *Let S^* be a set of n hyperplanes in $\mathbb{R}^d$. For any $0 \le k < n$, the convex hull of $\Lambda_k(S^*)$ has $O(n^{d-1})$ vertices.*

LEMMA 2.3. *Let S be a set of n points in $\mathbb{R}^3$. The combinatorial complexity of $c(S)$ is $O(n^2)$, and this bound is tight in the worst case.*

Proof: For the upper bound, we first note that the complexity of each of $U_{\lceil \frac{n}{d+1} \rceil}$ and $L_{\lfloor \frac{dn}{d+1} \rfloor}$ is $O(n^2)$, which follows from Lemma 2.2 and the fact that the total complexity of a convex polyhedron in $\mathbb{R}^3$ is proportional to the number of its vertices. A vertex x of $c(S)$ is mapped to a plane that either supports a facet of one of these two hulls, or is a common inner tangent to the two hulls, and then it supports an edge of one of them. Since there are at most two such planes per edge, the upper bound follows.

For the lower bound, let us first assume that n is an integer multiple of 12. We are interested in $c(S) = c_{n/4}(S) = \bigcap_{h \in \overline{\mathcal{H}}_{3n/4+1}} h$, the intersection of halfspaces induced by $(3n/4 - 2)$-facets (see Lemma 2.1).

Take a triangle Δuvw in the xy-plane, pass a vertical line through each of its three vertices u, v, w, and place $n/3$ points on each line at heights $\sqrt{1}, \sqrt{2}, \ldots, \sqrt{n/3}$. This yields a set S of n points in $\mathbb{R}^3$. We fix $j = 3n/4 - 2$, and we consider the j-facets of S that have one point on each of the vertical lines. There are $\Theta(j^2) = \Theta(n^2)$ choices of triples (a, b, c) such that $a + b + c = j + 3$, and any such triple defines a j-facet whose vertices are $(u, \sqrt{a})$, $(v, \sqrt{b})$, $(w, \sqrt{c})$.

Passing to the dual space, a plane containing any such j-facet becomes the intersection point of the three dual planes (where $\mathbf{x}$ denotes the vector (x, y) in the xy-plane)

$$z = u \cdot \mathbf{x} + \sqrt{a}, \quad z = v \cdot \mathbf{x} + \sqrt{b}, \quad \text{and} \quad z = w \cdot \mathbf{x} + \sqrt{c}.$$

Clearly, any such dual point lies on the ellipsoid

$$(z - u \cdot \mathbf{x})^2 + (z - v \cdot \mathbf{x})^2 + (z - w \cdot \mathbf{x})^2 = a + b + c = j + 3.$$

Since this is a convex surface, standard properties of the duality transform imply that each of the planes containing the j-facets in the primal space is tangent to a dual convex surface. Since each of these tangency points necessarily lies on the boundary of the intersection polytope, it follows that each of these j-facets contributes a facet to $c(S)$. The lower bound follows. □

Remark. We are currently unable to prove a similar bound on the complexity of $c(S)$ in higher dimensions. The upper bound theorem implies that the complexity of $c(S)$ is $O(n^{(d-1)\lfloor d/2 \rfloor})$. However, we conjecture that the actual bound is much smaller, maybe even $O(n^{d-1})$.

2.3 Computing the convex hull of a level

Let H be a set of n planes in $\mathbb{R}^3$ in general position, and let $k < n$ be an integer. We describe an algorithm for computing the convex hull of $\Lambda = \Lambda_k(H)$. We denote the convex hull of a set X by $\mathrm{conv}(X)$.

LEMMA 2.4. $\mathrm{conv}(\Lambda) = \mathrm{conv}\Big(\bigcup_{h \in H} \mathrm{conv}(\Lambda \cap h)\Big)$.

Proof: Clearly, $\mathrm{conv}(\Lambda)$ contains the set on the right-hand side. To establish the converse containment, we first argue that each vertex of Λ belongs to the right-hand side. Indeed, let v be such a vertex, incident upon three planes $h, h', h'' \in H$. Then v is a vertex of $\Lambda \cap h$, say, and the claim follows. Next, we approximate $\mathrm{conv}(\Lambda)$ by intersecting it with a sufficiently large ball B that encloses all vertices of $\mathcal{A}(H)$. Consider a point w that lies on the intersection of ∂B with an unbounded edge e of the level, incident upon two planes $h, h' \in H$. Then w belongs to $\Lambda \cap h$, say. Letting the radius of B tend to infinity completes the proof. □

LEMMA 2.5. *For any $h \in H$, $\mathrm{conv}(\Lambda \cap h)$ has linear complexity.*

Proof: Any vertex of $\Lambda \cap h$, and thus also of $\mathrm{conv}(\Lambda \cap h)$, lies on two of the lines in $\{g \cap h \mid g \in H \setminus \{h\}\}$, and any such line $g \cap h$ can contain at most two vertices of $\mathrm{conv}(\Lambda \cap h)$. A similar reasoning is applied to the unbounded edges of $\Lambda \cap h$. □

Our algorithm processes the planes of H one at a time. For each plane h it computes a convex (generally unbounded) polygon K_h with $O(n)$ edges and with the property that

$$\mathrm{conv}(\Lambda \cap h) \subseteq K_h \subseteq \mathrm{conv}(\Lambda). \tag{1}$$

By Lemma 2.4, $\mathrm{conv}(\bigcup_{h \in H} K_h) = \mathrm{conv}(\Lambda)$, so we simply compute and output the convex hull of $\bigcup_{h \in H} K_h$.

Although Λ is the graph of a continuous piecewise-linear totally defined function of x, y, the set $\Lambda \cap h$ within a single plane $h \in H$ is much less structured. It need not even be connected, and can in fact have quadratic complexity (in contrast with its convex hull, which has only linear complexity). Specifically, for each $g \in H \setminus \{h\}$, consider the halfplane $g^+ \cap h$ within h, where g^+ is the (closed) halfspace bounded from below by g. The level of a point $w \in h$ is the number of halfplanes $g^+ \cap h$ that contain w. These halfplanes can have a rather "erratic" structure, such as the one shown in Figure 2, which may cause $\Lambda \cap h$ to consist of up to $\Theta(n^2)$ connected components, as the figure illustrates.

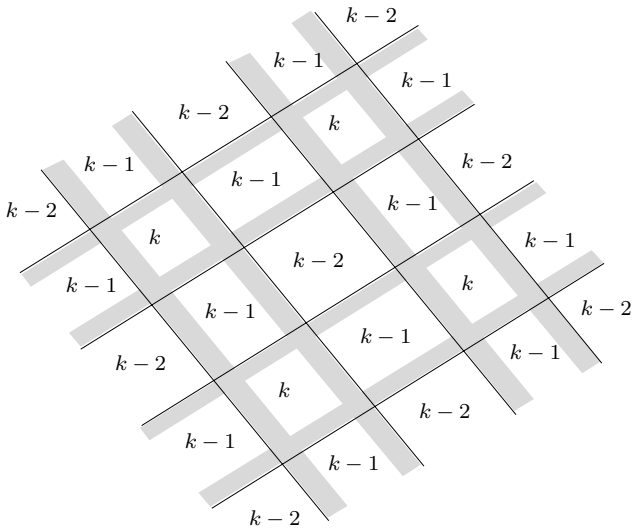

Figure 2. $\Lambda \cap h$ may have quadratic complexity.

Fix a coordinate frame within h whose axes project vertically to the x and y-axes of the 3-dimensional frame. (This is not an orthogonal frame, but can be made so with an appropriate affine transformation within h.) Let $g \in H \setminus \{h\}$. Write the equations of h and of g as $z = a_h x + b_h y + c_h$ and $z = a_g x + b_g y + c_g$. Observe that the halfplane $g^+ \cap h$ is a lower (resp., upper) halfplane within h in this coordinate frame if and only if $b_g > b_h$ (resp., $b_g < b_h$). The general-position assumption, and an appropriate choice of the

coordinate frame, allow us to assume that all the coefficients b_h, for $h \in H$, are distinct.

Sort the planes $h \in H$ in decreasing order of the y-coefficients b_h of their equations. Let the sorted order be $h_1, \ldots, h_n$. We process the planes in this order. Consider the processing of h_1. All halfplanes $g^+ \cap h_1$ are upper halfplanes, implying that $\Lambda \cap h_1$ is a level of the arrangement of the lines $g \cap h_1$, for $g \in H \setminus \{h_1\}$. We set $K_1 = K_{h_1}$ to be $\mathrm{conv}(\Lambda \cap h_1)$. Using the algorithm of Matoušek [12], K_1 can be computed in $O(n \log^4 n)$ time.

Suppose that $h_1, \ldots, h_{j-1}$ have already been processed, and that for each $i < j$ we have computed a (generally unbounded) convex polygonal region $K_i = K_{h_i}$ with the property (1). The processing of h_j then proceeds as follows.

We first compute the segments (or rays, or lines) $\gamma_i = K_i \cap h_j$. This can be done in $O(\log n)$ time for each $i < j$, by forming the line $h_i \cap h_j$ and by intersecting it, within h_i, with K_i. Put $\Gamma = \{\gamma_i \mid 1 \le i < j\}$, and refer to these segments as *red* segments. Let $R_j \subseteq h$ denote the convex hull of Γ, which can be computed in $O(n \log n)$ time. For each $i > j$ put $\ell_i = h_i \cap h_j$, refer to these lines as *green*, and put $G_j = \{\ell_i \mid j < i \le n\}$.

LEMMA 2.6. *The portion of $\Lambda \cap h_j$ that lies on green lines consists of pairwise disjoint x-monotone polygonal chains, each starting and ending either at infinity or on some red segment in Γ.*

Proof: Let w be a point in $\Lambda \cap h_j$ that lies on a green line, and trace $\Lambda \cap h_j$ from w to the right (i.e., in $(+x)$-direction). When encountering a new green line, the level switches to the new line and continues to the right; this follows from the fact that all the green halfplanes $h_i^+ \cap h_j$ are upper halfplanes in h_j. Continuing the tracing, we either reach $+\infty$ or an intersection point u with some red line $h_i \cap h_j$, with $i < j$. Then $u \in \Lambda \cap h_i$, so $u \in K_i$, and thus $u \in \gamma_i$. Tracing the level from w to the left, and repeating this analysis to each connected component of $\Lambda \cap h_j$ completes the proof of the lemma. □

LEMMA 2.7. *Let u, v be two y-covertical green points in $\Lambda \cap h_j$. Then the segment uv must cross at least one red segment of Γ.*

Proof: Suppose that u lies above v (in the y-direction). As we move from v in the positive y-direction, the level increases by 1 to $k+1$. Just before reaching u, the level is restored to its original value k. Thus there had to be a point w on uv so that after crossing w the level *decreases* by 1 and becomes k. Clearly, w has to lie on some red line h_i and in $\Lambda \cap h_j$. Hence, arguing as above, $w \in \gamma_i$. □

Consider the set $R_j \cup (\Lambda \cap h_j)$. The preceding lemmas imply that either this is a connected set, or it is the union of up to two connected components, one of which contains R_j and the other is an x-monotone unbounded green portion of $\Lambda \cap h_j$, passing either above or below R_j. Indeed, Lemma 2.7 and the convexity of R_j imply that if there are *two* green components enclosing R_j between them, then R_j is unbounded and the unbounded rays of its boundary are parallel to those of the green components. The latter is ruled out by the general-position assumption, so we can assume that there is at most one connected (unbounded) green component disjoint from R_j.

We define

$$K_j = \mathrm{conv}(R_j \cup (\Lambda \cap h_j)),$$

and denote its boundary by ζ_j. To construct ζ_j, we adapt the technique of Matoušek [12] for computing the convex hull of a level in the plane, and slightly relax it to simplify its analysis in our new context. The overall algorithm proceeds as follows. Let

$$L_j = \{h_i \cap h_j \mid 1 \le i \ne j \le n\},$$

and consider the following range space

$$\mathbb{X}_j = (L_j, \{\{\ell \in L_j \mid \ell \cap \tau \ne \emptyset\} \mid \tau \text{ is a triangle}\}),$$

which, as is well known, has *finite VC-dimension* [6]. We choose a sufficiently large constant r, and compute, in $O(n)$ time, a $(1/r)$-net $N_j \subseteq L_j$ for $\mathbb{X}_j$ of size $O(r \log r)$, and a triangulation $\mathcal{A}^\nabla(N_j)$ of the arrangement $\mathcal{A}(N_j)$ [6]. We compute the edges of $\mathcal{A}^\nabla(N_j)$ that intersect ζ_j. More precisely, for each edge e of $\mathcal{A}^\nabla(N_j)$, we compute the one or two edges of ζ_j that cross e, or determine that e does not cross $\partial\zeta_j$. The details of this main subroutine of the algorithm are presented in Section 2.4. We thus obtain a collection E_j of some edges of ζ_j. The edges of E_j lie in the zone of ζ_j in $\mathcal{A}(N_j)$, which implies that $|E_j|$ is proportional to the number of vertices in the zone. Since ζ_j is convex, the complexity of its zone in $\mathcal{A}(N_j)$ is $O(|N_j|\alpha(|N_j|)) = O(r\alpha(r)\log r)$. Since the segments of E_j are in convex position, they can be sorted along ζ_j in $O(r\alpha(r)\log^2 r)$ time. Let η, η' be two consecutive edges in E_j. By construction, there exists a triangle τ of $\mathcal{A}^\nabla(N_j)$ such that each of η, η' has an endpoint inside τ, and the portion of ζ_j between η and η' is fully contained in τ and is delimited by these two endpoints. (The case where the endpoints coincide is trivial, since there is no need to fill in ζ_j between η and η'.) See Figure 3.

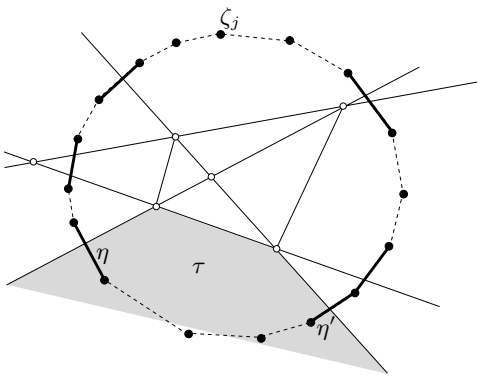

Figure 3. ζ_j (dashed polygon), E_j (whose edges are drawn with thick lines), and the triangles of $\mathcal{A}^\nabla(N_j)$. The portion of ζ_j between η and η' is fully contained in the single cell τ (shaded region).

We thus need to compute $\zeta_j \cap \tau$ for $O(r\alpha(r)\log r)$ triangles τ of $\mathcal{A}^\nabla(N_j)$. Since N_j is a $(1/r)$-net, each subproblem involves only at most n/r lines of L_j. We solve each subproblem recursively, using the same approach. This leads to a recurrence of the form

$$T(n) \le Ar\alpha(r)\log r \cdot T\left(\frac{n}{r}\right) + C(Q(n)+n),$$

where A is a constant independent of r, C a constant that does depend on r, and $Q(n)$ is the time needed to compute the zero, one, or two edges of ζ_j that cross a given line. Lemma 2.9 in the next subsection shows that $Q(n) = O(n\mathrm{polylog}(n))$. The above recurrence thus solves to $T(n) = O(n^{1+\varepsilon})$, for any $\varepsilon > 0$. We repeat this step for each of the planes h_j, and then compute the convex hull of the union of the resulting sets K_j, for $j = 1, \ldots, n$, to obtain the main result of this part of the paper:

THEOREM 2.8. *The center region of a set of n points in $\mathbb{R}^3$ can be computed in $O(n^{2+\varepsilon})$ time, for any $\varepsilon > 0$.*

2.4 Computing edges of K_j

We now describe the main procedure needed for the preceding algorithm: given a line ℓ, an integer $j \geq 1$, and the convex polygon R_j, return the edges of ζ_j that intersect ℓ. We recall that only the case $j > 1$ is relevant since, as noted, K_1 can be directly computed in $O(n \log^4 n)$ time [12]. Since we are given R_j, we assume that we have a point $o \in K_j$ at our disposal.

Let Q_j be the union of R_j, the monotone green portions of $\Lambda \cap h_j$ that terminate within R_j, and the unique monotone green portion of $\Lambda \cap h_j$ that lies above or below R_j, if it exists. In case the latter green component exists, we connect it to R_j by some vertical segment e that lies on the vertical line ℓ_o that passes through the point o in R_j. By computing the intersection points of ℓ_o with the planes in H, we can compute e in linear time. By construction, $K_j = \mathrm{conv}(Q_j)$.

We describe the overall algorithm in three stages. The first stage detects whether a query line g intersects K_j. If the answer is yes, it also returns an interval (possibly a single point) lying in $g \cap K_j$. Note that $\ell \cap K_j \neq \emptyset$ if and only if $\ell \cap Q_j \neq \emptyset$. The second stage determines whether a query point q lies in K_j, and computes the lines tangent to K_j from q if $q \notin K_j$. This stage computes the tangent lines using the previous procedure and the parametric-searching technique [15]. The third stage plugs the tangent-computation procedure into the parametric searching technique, to compute the edges of K_j that intersect a query line. We now describe each stage in detail.

Intersection detection between Q_j and a line. Let ℓ be a given line. To detect whether ℓ intersects Q_j, we proceed as in [12]. We intersect ℓ with each of the lines $h_i \cap h_j$ of L_j. We sort the intersection points along ℓ and scan ℓ in some direction, maintaining a count of the level we are in, and updating the count by ± 1 after crossing each intersection point. If we reach a point in $\Lambda \cap h_j$, we stop and report it. Otherwise, we test for intersections between ℓ and R_j, and report such an intersection point if found. Otherwise, it is still possible that ℓ intersects Q_j as it may pass between R_j and the unbounded x-monotone green chain that avoids R_j. To test whether this is the case, we compute, in a preprocessing step, the y-maximal and y-minimal intersections of the vertical line ℓ_o through o with Q_j, and then complete the query for ℓ by testing whether $\ell \cap \ell_o$ lies between these two intersections. If so, we report $\ell \cap \ell_o$. Otherwise, we determine that ℓ does not intersect Q_j. If $\ell \cap K_j \neq \emptyset$, we return the interval on ℓ bounded by the first and the last intersection points of $\ell \cap Q_j$. The total time spent by the procedure is $O(n \log n)$.

Computing a tangent to K_j from a point. Let q be a point in h_j. We wish to determine whether $q \in K_j$, and if the answer is no, we also want to compute the two tangents from q to K_j. Let ϱ denote the line passing through q and o and oriented from q to o. If $q \notin K_j$, then the two tangent rays from q to K_j lie on the opposite sides of ϱ, and we compute each of them separately. Without loss of generality, we assume that ϱ is the x-axis, q is the origin, and we wish to compute the tangent ray τ^* from q to K_j that lies in the halfplane $x \geq 0$, which has positive slope, say, σ^*. Using the parametric-searching technique [15], we simulate the above intersection-detection procedure generically at τ. During the simulation, we maintain an interval $I = [a, b] \subseteq [0, \infty]$. If I becomes empty, we conclude that $q \in K_j$, and if I becomes a singleton $[a, a]$, then $\sigma^* = a$. At each step of the simulation, we have a real value $\sigma \in [a, b]$ and we wish to determine whether the ray τ of slope σ passing through q intersects K_j. If the answer is yes, then $\sigma^* \geq \sigma$ and we set $I = [\sigma, b]$; otherwise $\sigma^* \leq \sigma$, and we set $I = [a, \sigma]$. Use the preceding algorithm, we can detect this intersection in $O(n \log n)$ time. If the procedure also returns an interval that contains q, we conclude that $q \in K_j$ and stop. The standard parametric-searching argument implies that the simulation always terminates with an empty interval or with a singleton.

The intersection of τ with K_j is a single point, an edge of K_j, or empty. The third situation arises when τ is parallel to an unbounded edge of K_j. We can easily compute this intersection, by noting that the endpoints of $\tau \cap K_j$ must belong to $R_j \cup (\Lambda \cap h_j)$, and we know how to compute such an intersection, in the same manner as in the intersection-detection procedure described above.

If $q \notin K_j$ then we repeat the same procedure to find the other tangent of K_j from q. The total time spent is $O(n \log^3 n)$.

Computing the edges of K_j crossed by a line. Armed with the tangent computation procedure, we next derive the main procedure by applying the parametric searching once more: Given a query line ℓ, compute the one or two edges of K_j that are crossed by ℓ, or determine that $\ell \cap K_j = \emptyset$. The latter task can be accomplished using the basic intersection testing procedure, so we may assume that $\ell \cap K_j \neq \emptyset$, and that we have computed a point $q_0 \in \ell \cap K_j$. Using parametric searching once again, we slide a point q from q_0 along each of the halflines of ℓ delimited by q_0, and test whether q lies in K_j, using the tangent computation procedure described above. This guides our search: If q lies in K_j, we proceed by moving further away from q_0, and otherwise we proceed by moving towards q_0. When we home in on the actual point q of intersection between ℓ and ∂K_j, the tangent computation procedure yields the desired edge of K_j that ℓ crosses at q. We omit the details from this abstract, and conclude with the following result, which is the promised missing ingredient for the proof of Theorem 2.8.

LEMMA 2.9. *Given a line ℓ, we can find in $O(n \,\mathrm{polylog}(n))$ time the edges of K_j that intersect ℓ.*

3. RECOGNIZING COLORED TVERBERG POINTS IN THE PLANE

Let S be a 3-colored set, which is the disjoint union of a set R of n red points, a set B of n blue points, and a set G of n green points. We assume that the points of S are in general position.

Let q be a given point that we want to test for being a colored Tverberg point of S. Let C be the unit circle centered at q. We may assume, without loss of generality, that all the points of S lie on C; otherwise we project these points on C, centrally from q, and note that q is a colored Tverberg point of the original set if and only if it is a colored Tverberg point of the projected set. If q is generic, all projected points are distinct. Otherwise, since S is in general position, at most two pairs of points of S may project to coinciding points on C. This will require easy and straightforward modifications of the following procedure, which we omit, and assume that all projected points are distinct. Similarly, we will also assume that no two projected points are diametrically opposite on C.

Let C_0 be a fixed semicircle of C, whose endpoints are disjoint from S. For each point u in the (open) complementary semicircle C_1, let $\bar{u}$ denote the antipodal point of u in C_0. Put $R^+ = R \cap C_0$ and $R^- = \{\bar{u} \mid u \in R \cap C_1\}$, and define similarly the sets B^+, B^-, G^+ and G^-. Sort the points of $R^+ \cup R^- \cup B^+ \cup B^- \cup G^+ \cup G^-$ in counterclockwise order along C_0, and denote the resulting (linear) sequence by E. (By our assumptions, all elements of E are distinct.) Note that a *rainbow* triangle uvw, with $u \in R$, $v \in B$, $w \in G$, contains q if and only if E contains one of the

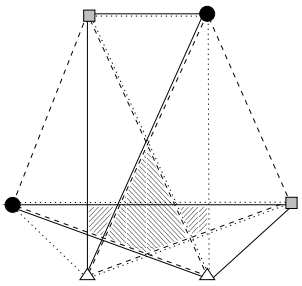

Figure 4. A non-convex region of colored Tverberg points in the plane; example due to [8].

ordered triples

$$(u, \bar{v}, w), (w, \bar{v}, u), (v, \bar{w}, u),$$
$$(u, \bar{w}, v), (w, \bar{u}, v), (v, \bar{u}, w),$$
$$(\bar{u}, v, \bar{w}), (\bar{w}, v, \bar{u}), (\bar{v}, w, \bar{u}),$$
$$(\bar{u}, w, \bar{v}), (\bar{w}, u, \bar{v}), (\bar{v}, u, \bar{w}),$$

as a (not necessarily contiguous) subsequence; the specific triple is determined by the locations of u, v, w along C. See Figure 5. Our goal is thus to determine whether E can be decomposed into n pairwise disjoint triples of these 12 kinds.

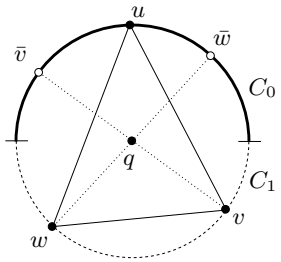

Figure 5. A triangle uvw that contains q, and the corresponding triple $(\bar{v}, u, \bar{w})$.

We first describe a less efficient algorithm for solving this problem, which is conceptually simpler, and then optimize it to improve its running time. Write E as $(e_1, e_2, \ldots, e_{3n})$, and denote by E_j its prefix $(e_1, \ldots, e_j)$, for $j = 0, 1, \ldots, 3n$. The algorithm uses dynamic programming and processes the elements of E in increasing order. At the beginning of the processing of e_i, it maintains a set X_{i-1} of *configurations*, each of which is an 18-tuple of integers, which represent possible partitions of $E_{i-1} = (e_1, \ldots, e_{i-1})$ into triples and prefixes of triples of the above 12 kinds. More specifically, the first six components of a configuration ξ, which we denote by $N_\xi(R^+)$, $N_\xi(R^-)$, $N_\xi(B^+)$, $N_\xi(B^-)$, $N_\xi(G^+)$, $N_\xi(G^-)$, record the number of singleton prefixes of triples that lie in E_{i-1}, where $N_\xi(R^+)$ is the number of such singleton prefixes in R^+, and similarly for the other five quantities. (Counting an element of E_{i-1} as a singleton prefix means that we expect it, in the configuration under consideration, to form a valid triple with two other elements that lie further ahead of E_{i-1}.) The next 12 components are each indexed by a pair of a positive color set and a negative color set, where the two colors are distinct, and record potential doubleton prefixes of triples in E_{i-1} (where the third element of such a triple is expected to come from the remainder of E). For example, $N_\xi(R^+G^-)$ is the number of doubleton prefixes (u, v) in the configuration, where $u \in R^+$, $v \in G^-$, and u precedes v in E_{i-1}. (To complete this pair into a valid triple, an element of B^+ will have to be chosen from the remainder of E.) The other 11 components are defined in complete analogy.

Initially, X_0 consists of only the all-zero tuple $\mathbf{0}$. When processing e_i, we produce the set X_i of configurations for E_i from X_{i-1} as follows. Suppose that $e_i \in R^+$, and let ξ be a configuration in X_{i-1}. We generate from ξ five configurations in X_i, according to the following choices of the role of e_i:

(i) e_i is the first component of a new triple: We generate a new configuration by increasing $N_\xi(R^+)$ by 1.

(ii) e_i is the second component of a triple whose first component is in B^-: We generate a new configuration by increasing $N_\xi(B^-R^+)$ by 1, and by decreasing $N_\xi(B^-)$ by 1.

(iii) e_i is the second component of a triple whose first component is in G^-: We generate a new configuration by increasing $N_\xi(G^-R^+)$ by 1, and by decreasing $N_\xi(G^-)$ by 1.

(iv) e_i is the third component of a triple whose first component is in B^+ and whose second component is in G^-: We generate a new configuration by decreasing $N_\xi(B^+G^-)$ by 1.

(v) e_i is the third component of a triple whose first component is in G^+ and whose second component is in B^-: We generate a new configuration by decreasing $N_\xi(G^+B^-)$ by 1.

The handling of the cases where e_i belongs to each of the other five signed color classes is handled in complete symmetry. We discard any generated configuration that has any negative component. The set X_i stores configurations without repetition. Whenever a new configuration ξ is inserted into X_i (for the first time), we store with it a pointer to the configuration $\xi' \in X_{i-1}$ from which ξ has been generated. More precisely, we simply store with ξ the type of incremental change that has produced it from ξ', using which ξ' can easily be reconstructed. (In general, there may be several configurations ξ' that can induce ξ, and we store a pointer to only the first one that has generated ξ.)

The number of configurations in any X_i is $O(n^{18})$, so the processing of each e_i takes $O(n^{18})$ time, using an appropriate hash-table structure to store X_i. The total running time is thus $O(n^{19})$. After processing e_{3n}, we test whether X_{3n} contains the all-zero tuple $\mathbf{0}$. If it does not, q is not a Tverberg point. If $\mathbf{0}$ is in X_{3n} then, using the additional data stored with each configuration, we trace back a sequence of configurations $\mathbf{0} = \xi_0, \xi_1, \xi_2, \ldots, \xi_{3n-1}, \xi_{3n} = \mathbf{0}$, so that each ξ_j belongs to X_j and can be generated from ξ_{j-1} when processing e_j. The corresponding decomposition of E into rainbow triples can then be easily performed by processing this sequence of configurations: We maintain a collection of prefixes of triples, and update it as dictated by the changes that transform each configuration to the next one in the sequence. For example, suppose that $e_j \in R^+$ and the change at e_j is of type (ii). We look for any element e_i in E_{j-1} that belongs to B^- and forms a singleton prefix in our maintained collection, remove it from the collection and replace it by the doubleton (e_i, e_j). Similar updates are done in all other cases. The correctness of this step follows from the invariant, easily established using induction on j, that if a configuration ξ belongs to X_j then E_j admits a decomposition into pairwise disjoint prefixes of triples (and complete triples), so that the number of prefixes of each type is equal to the corresponding component of ξ.

In summary, we have shown that determining whether a given point q is a colored Tverberg point of S can be done in $O(n^{19})$ time.

We next proceed to optimize the algorithm. In the revised version we apply the same general approach, but maintain configurations with fewer components. First we note that there is no need to maintain the two separate quantities $N_\xi(R^+B^-)$, $N_\xi(B^+R^-)$, and it suffices just to maintain their sum. Indeed, both quantities are accessed only when a further element of E that belongs to G^+ "decides" to become the last element of a triple, in which case it has to be matched with a doubleton that is counted in one of these two quantities, but it does not matter which of the two kinds of doubletons is being used. A configuration thus needs to record only 12 counts, six singleton counts, as above, and the six doubleton counts

$$\begin{aligned} N_\xi(R^+B^-) &+ N_\xi(B^+R^-), \\ N_\xi(R^-B^+) &+ N_\xi(B^-R^+), \\ N_\xi(R^+G^-) &+ N_\xi(G^+R^-), \\ N_\xi(R^-G^+) &+ N_\xi(G^-R^+), \\ N_\xi(B^+G^-) &+ N_\xi(G^+B^-), \\ N_\xi(B^-G^+) &+ N_\xi(G^-B^+)\,. \end{aligned}$$

This already yields an algorithm that runs in $O(n^{13})$ time (there are $O(n^{12})$ different configurations, and there are n iteration steps).

We can further reduce the number of components in a configuration to 10, as follows. Suppose that $\xi \in X_j$. Denote by $M_\xi(R)$ the sum of all components of ξ that record prefixes of tuples that involve an element of R (i.e., of $R^+ \cup R^-$), and define similarly $M_\xi(B), M_\xi(G)$. Let $K_j(R), K_j(B), K_j(G)$ denote the number of elements of E_j that are red, blue, and green, respectively. Let t denote the number of complete triples (contained in E_j) that have been generated by the incremental construction recorded in ξ (or, more precisely, in the unique sequence of configurations in $X_1, X_2, \ldots, X_j$ that terminates at ξ and whose reverse is obtained by following the stored back pointers, starting from ξ). Then we have

$$\begin{aligned} K_j(R) &= t + M_\xi(R) \\ K_j(B) &= t + M_\xi(B) \\ K_j(G) &= t + M_\xi(G)\,. \end{aligned}$$

That is,

$$\begin{aligned} M_\xi(B) - M_\xi(R) &= K_j(B) - K_j(R) \\ M_\xi(G) - M_\xi(R) &= K_j(G) - K_j(R)\,. \end{aligned}$$

This gives us two independent linear relations among the 12 components of a configuration, showing that it suffices to store and maintain only 10 of them. The number of tuples generated by the algorithm is thus $O(n^{10})$, and the total running time is $O(n^{11})$.

THEOREM 3.1. *Let S be a set of $3n$ points in the plane, n of which are red, n blue, and n green. For a given point q, we can determine whether q is a colored Tverberg point of S in time $O(n^{11})$.*

4. OPEN PROBLEMS

The paper raises a number of open problems for further research. We mention only a few:

(i) Obtain a tight bound for the maximum possible complexity of the center region $c(S)$ of a set S of n points in $\mathbb{R}^d$.

(ii) Can the center region in three dimensions be constructed in $O(n^2\mathrm{polylog}(n))$ time?

(iii) Can colored Tverberg points in the plane be recognized in a more efficient manner? What about colored Tverberg points in higher dimensions?

Acknowledgments. The authors thank Jirka Matoušek and Boris Aronov for useful discussions concerning this problem. The authors also thank the GWOP working group[8] for providing the construction in Figure 4.

References

[1] P. K. Agarwal, J. Matoušek, and O. Schwarzkopf, Computing many faces in arrangements of lines and segments, *SIAM J. Comput.* 27 (1998), 491–505.

[2] D. Avis, The m-core property contains the m-divisible points in space, *Pattern Recog. Letts.* 14 (1993), 703–705.

[3] I. Bárány and D. Larman, A colored version of Tverberg's theorem, *J. London Math. Soc., Ser. II*, 45 (1992), 314–320.

[4] I. Bárány, Z. Füredi and L. Lovász, On the number of halving planes, *Combinatorica* 10 (1990), 175–183.

[5] T. Chan, An optimal randomized algorithm for maximum Tukey depth, *Proc. 15th Ann. ACM-SIAM Symp. on Discrete Algorithms*, 2004, 423–429.

[6] B. Chazelle, *The Discrepancy Method*, Cambridge University Press, 2000.

[7] K. Clarkson, D. Eppstein, G. L. Miller, C. Sturtivant, and S.-H. Teng, Approximating center points with iterative Radon points, *Intl. J. Comput. Geom. Appls.* 6 (1996), 357–377.

[8] P. Csorba, K. Fischer, M. John, Cs.D. Tóth, Y. Okamoto, J. Solymosi, M. Stojanković, U. Wagner, A nonconvex Colored-Tverberg region, Working Group at 1st GWOP'03, 2003.

[9] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer Verlag, Heidelberg, 1987.

[10] S. Jadhav and A. Mukhopadhyay, Computing a centerpoint of a finite planar set of points in linear time, *Discrete Comput. Geom.* 12 (1994), 291–312.

[11] G. Kalai, Combinatorics with a geometric flavor: Some examples, in *Visions in Mathematics Toward 2000 (Geometric and Functional Analysis, Special Volume)*, 742–792.

[12] J. Matoušek, Computing the center of a planar point set, in *Discrete and Computational Geometry* (J. Goodman, J. Pollack, and W. Steiger, eds.), American Mathematical Society, 1991, 221–230.

[13] J. Matoušek, Efficient partition trees, *Discrete Comput. Geom.*, 8 (1992), 315–334.

[14] J. Matoušek, *Lectures in Discrete Geometry*, Springer Verlag, Heidelberg, 2002.

[15] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM*, 30 (1983), 852–865.

[16] N. Naor and M. Sharir, Computing a point in the center of a point set in three dimensions, *Proc. 2nd Canadian Conf. Comput. Geom.*, 1990, 10–13.

[17] R. Rado, A theorem on general measure, *J. London Math. Soc.* 21 (1947), 291–300.

[18] M. Sharir and P.K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, 1995.

[19] M. Sharir, S. Smorodinsky and G. Tardos, An improved bound for k-sets in three dimensions, *Discrete Comput. Geom.* 26 (2001), 195–204.

[20] S.-H. Teng, *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, PA, 1992.

[21] H. Tverberg, A generalization of Radon's theorem, *J. London Math. Soc.* 41 (1966), 123–128.

[22] R.T. Živaljević and S.T. Vrećica, The colored Tverberg's problem and complexes of injective functions, *J. Combin. Theory Ser. A* 6 (1992), 309–318.