COMPUTING COLOURFUL SIMPLICIAL DEPTH AND MEDIAN IN $\mathbb{R}^2$

GREG ALOUPIS, TAMON STEPHEN, AND OLGA ZASENKO

Abstract. The *colourful simplicial depth* (CSD) of a point $x \in \mathbb{R}^2$ relative to a configuration $P = (P^1, P^2, \ldots, P^k)$ of n points in k colour classes is the number of closed simplices (triangles) with vertices from three different colour classes that contain x in their convex hull. We consider the problems of efficiently computing the colourful simplicial depth of a given point x, and of finding a point in $\mathbb{R}^2$, called a *median*, that maximizes colourful simplicial depth.

Our algorithm for colourful simplicial depth runs in $O(n \log n)$ time, and in $O(n)$ time if the points are already sorted around x. This is optimal for sorted inputs. Our algorithm for computing the colourful median runs in $O(n^4)$ time. Both results extend known algorithms for the monochromatic versions of these problems, and match the corresponding time complexities.

1. Introduction

The *simplicial depth* of a point $x \in \mathbb{R}^2$ relative to a set P of n input points is the number of simplices (triangles) formed with the points from P that contain x in their convex hull. A *simplicial median* of the set P is any point in $\mathbb{R}^2$ which is contained in the most triangles formed by elements of P, i.e., has maximum simplicial depth with respect to P. Here we consider a set P that consists of k colour classes $P^1, \ldots, P^k$. The *colourful simplicial depth* (CSD) of x with respect to configuration P is the number of triangles with vertices from three different colour classes that contain x. A *colourful simplicial median* of a configuration $P = (P^1, P^2, \ldots, P^k)$ is any point in the plane with maximum colourful simplicial depth. Trivially this point must be in the convex hull of P.

Monochrome simplicial depth was introduced by Liu [Liu90]. Up to a constant, it can be interpreted as the probability that x is in the convex hull of a random simplex generated by P. The colourful version generalizes this to selecting points from k distributions, see [DHST06]. In this setting, medians are central points which are in some sense most representative of the distribution(s). Our objective is to find efficient algorithms for finding both the colourful simplicial depth of a given point x with respect to a configuration, and a colourful simplicial median of a configuration.

1.1. **Background.** Both monochrome and colourful simplicial depth are well defined in all dimensions, and are natural objects of study in discrete geometry. For more background on simplicial depth and competing measures of data depth, see [Alo06] and [FR05]. Monochrome depth has seen a flurry of activity in the past few years, most notably relating to the *First Selection Lemma*, which is a lower bound for the depth of the median, see e.g. [MW14].

The colourful setting for simplicial depth is suggested by Bárány's approach [Bár82] to proving a colourful version of Carathéodory's theorem. Deza et al. [DHST06] formalized the notion and considered bounds for the colourful depth of points in the intersection of the convex hulls of the colours. Among the recent work on colourful depth are proofs of the lower [Sar15] and upper [ABP⁺17] bounds conjectured by Deza et al., with the latter result showing beautiful connections to Minkowski sums of polytopes.

The colourful simplicial depth represents the number of basic solutions to a colourful linear programming problem, see [BO97, DHST08]. Applications of colourful linear programming include computing Nash equilibria in a bimatrix game [MS18a].

Monochrome simplicial depth can be computed trivially by enumerating simplices, in $O(n^{d+1})$ time for dimension d. Afshani, Sheehy and Stein [ASS16] produced the first non-trivial algorithm for $d > 4$, running in $O(n^d \log n)$ time. They also provided a range of approximation algorithms for all dimensions. Until recently, the best known time bounds for 3D and 4D were $O(n^2)$ and $O(n^4)$ respectively, by Cheng and Ouyang [CO01]. All exact computation bounds for $d > 3$ were matched or improved by Pilz, Welzl and Wettstein, whose algorithm runs in $O(n^{d-1})$ time [PWW17].

The two-dimensional case was considered by several authors in the 90's. Algorithms for monochrome depth running in $O(n \log n)$ time were given by Khuller and Mitchell [KM90], Gil, Steiger and Wigderson [GSW92] and Rousseeuw and Ruts [RR96]. In each case, the bottleneck was a radial sort around the query point, without which the time would be linear. A worst-case time lower bound of $\Omega(n \log n)$ was given in [ACG⁺02]. Elmasry and Elbassioni provided an output-sensitive algorithm that computes the simplicial depth of a point in $O(n + n \log(1 + t/n))$ time, if the depth of the point happens to be t [EE05]. For more about enumerating the simplices that contain a given point, see [EEM11].

As for the simplicial median, Khuller and Mitchell [KM90], and Gil, Steiger, and Wigderson [GSW92] studied an *in-sample* version of the problem, by computing a point *from* P with maximum simplicial depth. In fact they computed the depth of all input points in quadratic time, by obtaining the required sorted ordering for all n points within that time bound and then applying the standard depth algorithm n times. However, we consider a *simplicial median* to be *any* point $x \in \mathbb{R}^2$ maximizing the simplicial depth. Rousseeuw and Ruts [RR96] provided an algorithm to compute the simplicial median in $O(n^5 \log n)$ time. Aloupis et al. [ALST03] improved the time complexity to $O(n^4)$. We conjecture that this is optimal, and observe (in Lemma 3.1, Section 3.3) that there are $O(n^4)$ candidate points for the location of the colourful median as well.

1.2. **Organization and Main Results.** In Section 2, we show how the classic monochromatic simplicial depth algorithm can be extended to colourful input, resulting in a time complexity of $O(k + n \log n)$, in the real RAM model. Again, sorting is the bottleneck, so if the input is already sorted about the query point, x, the time complexity drops to $O(k + n)$. Given that $k \leq n$, the complexities are respectively $O(n \log n)$ and $O(n)$. The algorithm is optimal up to a constant factor as in the monochromatic case. Formally, the main result presented at the end of Section 2 is:

Theorem. *Given a set $P \subset \mathbb{R}^2$ of n input points in k different colours, ($3 \leq k \leq n$), and a point x, with $P \cup \{x\}$ in general position, the colourful simplicial depth of x relative to P can be found in $O(n \log n)$ time. If the points in P are already sorted around x, the depth of x can be computed in $O(n)$ time.*

Note that we are using *general position* to mean that no three points are colinear.

In Section 3, we turn our attention to computing a colourful simplicial median. We extend the monochromatic algorithm of [ALST03] and preserve its time complexity of $O(n^4)$, independent of k. Formally, the main result presented at the end of Section 3 is:

Theorem. *Given a set of input points $P \subset \mathbb{R}^2$ in general position in k different colours, $k \geq 3$, the colourful simplicial median of P can be found using $O(n^4)$ time and $O(n^2)$ space, where $|P| = n$.*

Section 4 contains conclusions and discussion about future directions.

2. Computing Colourful Simplicial Depth

2.1. **Preliminaries.** We consider a point $x \in \mathbb{R}^2$ and family of sets $P^1, P^2, \ldots, P^k \subseteq \mathbb{R}^2$, $k \geq 3$, where each P^i consists of the points of some particular colour i. Refer to the j^{th} element of P^i as P^i_j. We generally use superscripts for colour classes, while subscripts indicate the position in the array. We denote the union of all colour sets by P: $P = \bigcup_{i=1}^{k} P^i$. The total number of points is n, where $|P^i| = n_i$, and $\sum_{i=1}^{k} n_i = n$. Clearly, if each set is non-empty, $k \leq n$; and it would be trivial to identify and eliminate empty colour sets in linear time. To avoid technicalities, we assume that points of $P \bigcup \{x\}$ are in general position, and only two edges formed by pairs of input points cross at any given position. Such degeneracy issues can be taken care of with some special handling without changing the asymptotic running time, see [EM88]. Note that we use the word *edge* to refer to the line segment formed by two input points.

Definition 2.1. *The line segment formed by two points of P is called an* edge*. The proper intersection of two edges is a* vertex*. An edge is* colourful *if its two endpoints, a, b, have different colours, i.e., belong to distinct sets P^a, P^b, where $a \neq b$.*

Definition 2.2. *A* colourful triangle *is the convex hull of three points from P that define three colourful edges. In other words the points have three distinct colours. At times when the context is clear we may refer to the generating points as triangles.*

Definition 2.3. *The* colourful simplicial depth $D(x, P)$ *of a point x relative to P is the number of colourful triangles, formed by elements of P, containing x. (See Figure 1)*

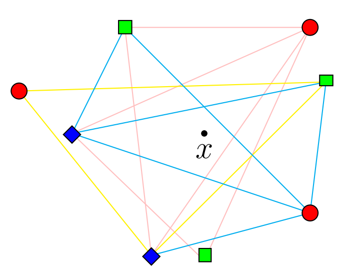

Figure 1. A configuration P of 8 points in $\mathbb{R}^2$ surrounding a point x with $D(x, P) = 6$.

Remark 2.4. *We are checking containment in* closed *triangles. With our general position assumption, this will not affect the value of $D(x, P)$. We find it more natural to consider closed triangles than open triangles in defining colourful medians. Arguments in this paper can be adapted to the open triangle context.*

2.2. **Monochrome Depth Algorithm and an Extension to Colourful Depth.** We begin by briefly recalling how to compute the monochromatic simplicial depth of a point x, as we will be relying on this for our result. See [ACG$^+$02] for details. The depth of x can be determined trivially by counting the triangles formed by P that do *not* contain x, so we will focus on that task. Let r^- be the ray from x passing through an arbitrary point v in P. Let H be the open halfplane to the left of this ray, and let r^+ be the ray from x pointed in the opposite direction of r^-. Any two points (b, c) in H, taken together with v, form a triangle that does not contain x. Thus we can use v as an *anchor*, to which we will charge a subset of all the triangles not containing x. The size of this subset is $\binom{h}{2}$, where h is the number of points in H (trivially, the size is zero if $h < 2$). The algorithm computes h by brute force, and then rotates the two rays counterclockwise, pausing every time a ray hits an input point, until r^- reaches v again. If a pause is triggered by r^+, h is incremented. Otherwise, r^- reaches a new anchor, so h is decremented and the updated value of $\binom{h}{2}$ is added to a global count (being charged to the new anchor). Thus every stop costs constant time. The bottleneck is the angular sort of all input points about x which allows the rays to perform their sweep. Finally, we note that there is no double-counting: for example, it is easy to see that when b or c becomes the anchor, v will not be in the updated halfplane, H. Thus the triangle v, b, c can be charged uniquely to its anchor, v.

The algorithm above can be extended to compute the colourful simplicial depth of x, in $O(n \log n + kn)$ time. The objective is to count all colourful triangles that do not contain x. Again, a line through x is rotated, stopping at every input point to increment or decrement the number of points in a halfplane, and whenever a new anchor point is reached, a sum is computed (now in $O(k)$ time) and added to a global sum.

Specifically, we now maintain the number of points, h_i, of colour i in H, for every colour class P^i. All together, the h_i terms are initialized in linear time. Then of course $h = \sum h_i$. As before, we start off by calculating the number of colourful triangles, anchored at v, that do not contain x. Without loss of generality, suppose that v belongs to P^1. Clearly we must ignore all points in H within this colour class. Consider how to compute all desired triangles that involve a point q of colour P^2. This is equal to the count of all points in H that do not belong to P^1 or P^2. Let this be denoted by $h_{\overline{1,2}}$. Trivially, $h_{\overline{1,2}} = h - h_1 - h_2$ can be computed in constant time. Thus the number of triangles in H (anchored at v) that avoid x, and contain a non-anchor point of colour P^2 is $h_2 \cdot h_{\overline{1,2}}$. Similarly for any colour class P_j, such that $j \neq 1$, the number of triangles anchored at v that avoid x is $h_j \cdot h_{\overline{1,j}}$. The sum of all these terms, over all $j \neq 1$, equals the number of colourful triangles anchored at v that avoid x, except that we have double-counted each triangle. This sum for v has $k-1$ terms (even if we factor in a term for colour classes missing in H) and is therefore computed in $O(k)$ time.

After the sum for v has been computed, we rotate the rays counterclockwise. We then increment h_i (and h) for any point that enters H as we rotate, until r^- hits a new anchor. After decrementing the anchor's colour count in H, we are ready to compute a new sum. All the ingredients are ready, namely h and all the h_i, so the sum for the new anchor can be computed in $O(k)$ time. Thus for all n anchors, the total time is $O(kn)$, after the initial sorting step.

Note that a similar algorithm with the above time complexity appeared in a preliminary version of this work [ZS16], and has been implemented [Zas16]. The present algorithm is conceptually simpler, but either way in the following section we show how to avoid spending linear time (in terms of k) to compute the sum for each anchor. Instead, we spend only constant time per anchor.

2.3. **Colourful Depth in $O(n \log n)$ time.** The problem with the preceding approach is that we explicitly used all k terms h_i $(1 \leq i \leq k)$, for every anchor's sum. To fix this, we will still maintain all h_i, by updating one such term per step, but we will also maintain a more useful count that can be updated in constant time per step, and that will allow the sum at each anchor to be computed in constant time as well.

Consider any anchor, v, using the general framework described in the preceding section. Suppose that v belongs to P^1. Counting the triangles anchored at v that avoid x is equivalent to knowing the number of bichromatic pairs of points in H, that do not involve points in P^1. This number can be obtained in constant time if we know the total number of bichromatic pairs in H, and the number of bichromatic pairs that involve precisely one point from P^1, so that the latter can be subtracted from the former.

Denote the total number of bichromatic pairs in H by w. This can be obtained from h and the h_i's in linear time in k: for every colour, i, compute $h_i \cdot (h - h_i)$, then compute the sum of these products over all i. Finally divide by two to take care of double-counting. We perform this initialization when we select our first anchor, v, that belongs to some arbitrary colour class, say P^1. The number of triangles that we need to count for v is $w - h_1 \cdot (h - h_1)$.

Whenever we rotate, updating w is simple. For instance, suppose that a point y, belonging to P^c, is about to enter H. How many new bichromatic pairs will there be, after including y in H? Precisely $h - h_c$. Thus we add this amount to w, and then we increment h and h_c. When a point exits H we subtract from w and decrement accordingly. Of course, in this case we are dealing with a new anchor, so after updating w we compute the sum $w - h_c \cdot (h - h_c)$ for the new anchor.

In conclusion, there are $2n$ steps after initialization; at each step we increment or decrement h and one h_i term, and we update w in constant time. Also for each of the n anchors we compute the number of triangles that get charged to the anchor, in constant time, and add this number to a global sum. After sorting, the total time complexity of the algorithm is $\Theta(n)$.

Theorem 2.5. *Given a set $P \subset \mathbb{R}^2$ of n input points in k different colours, ($3 \leq k \leq n$), and a point x, with $P \cup \{x\}$ in general position, the colourful simplicial depth of x relative to P can be found in $O(n \log n)$ time. If the points in P are already sorted around x, the depth of x can be computed in $O(n)$ time.*

3. Computing Colourful Simplicial Medians

3.1. **Overview of Monochromatic Simplicial Median Computation.** A brute-force algorithm to find a monochromatic simplicial median takes $O(n^5 \log n)$ time by computing the depth of every intersection point formed by edges between input points. In other words, the depth is computed at all crossings in the complete geometric graph defined on the input. The crossing number lemma [ACNS82, Lei83] gives that there are $\Theta(n^4)$ such points independent of the positions of the vertices. The justification for only considering intersection points is simple, but for completeness we prove this statement in Lemma 3.1. A speedup can be obtained by iteratively dealing with each edge, processing all the incident intersection points in order. In general, as we walk on an edge (a, b), the depth changes only when there is an intersection with some other edge. Suppose that we are at the position t on (a, b) where edge (c, d) crosses, and $\{a, b, c, d\}$ are distinct points. Given our general position assumption, (c, d) is the base of $n-2$ triangles that contain t. It also defines two open halfplanes. As we step away from t along (a, b) into one of the two halfplanes, we exit a certain number of the triangles just mentioned; one for each input point in the other halfplane. Therefore by precomputing the number of points in the halfplanes defined by each of the edges, the depth along (a, b) can be updated in constant time. As we keep walking, if we encounter another crossing edge (e, f), the situation is symmetric: we enter the set of triangles with base (e, f) and apex in a halfplane defined by (e, f). The preprocessing step of counting points in halfplanes takes $O(n)$ time per edge, so $O(n^3)$ time overall.

To walk on (a, b), we do not begin at one of its endpoints; we begin at a vertex if one exists, so that the conditions mentioned in the preceding paragraph will hold. In fact we begin at an intersection closest to an endpoint (call such an intersection *special*). Therefore in preprocessing we compute the depth of the special intersection(s) on each edge, which can be found by brute force in quadratic time per edge. The total time of this preprocessing step is thus $O(n^4)$.

As mentioned, to walk on one edge, we need all the intersection points in sorted order. There is the option of sorting all these points in $O(n^2 \log n)$ time and quadratic space. Thus, iterating on the set of edges, the time complexity is $O(n^4 \log n)$, still using quadratic space. However, as mentioned in [ALST03], by using a topological sweep [EG89] on the graph, the time complexity can be reduced by a log factor to $O(n^4)$. As a reminder, the effect of a topological sweep is to scan through the graph with a curve, so that the intersections on each edge are swept in correct order. When the sweep passes through a input point or a special intersection, we simply look up its depth. Otherwise we update depth as described above in constant time.

3.2. **Extension to Colourful Simplicial Median Computation.** In the colourful setting, the underlying structure is the same, except that monochromatic edges can be ignored. In other words, to find a median it suffices to consider the depth only at the input points and at vertices where colourful edges intersect. The reason for this is identical to the analogous monochromatic case. As a result, the graph in question is now a complete geometric k-partite graph, G. Note that there are many examples of such graphs that contain $\Omega(n^4)$ candidates for the location of a median, see for example [GHL+16] for an analysis of the balanced case.

Unlike before, now we only walk on colourful edges. While walking on (a, b), if a colourful edge (c, d) is crossed we need to know the number of points in the open halfplanes that it defines, involving only colours other than c or d. Analogous to the monochromatic case, this is because any such point forms a colourful triangle with base (c, d), that we either enter or exit at this particular intersection. For example, in Figure 2, when walking on (y_1, b_2), as soon as we step onto c_3 coming from b_2, we enter two triangles with base (g_1, r_2): one with apex b_1 and the other with apex y_1. So as preprocessing for each colourful edge we store the corresponding number of points in each of its two halfplanes. Brute-force suffices, taking $O(n)$ time per edge, and $O(n^3)$ time overall, as in the monochromatic case.

The definition of a *special* intersection holds as before. See Figure 2. Also as before, we can process each colourful edge iteratively, after pre-sorting all its intersections in G. In other words we can update the depth in constant time per intersection, beginning at a special intersection if one exists. The depth of each input point is computed as well. Finally, with a topological sort on G the time complexity becomes $O(n^4)$.

Some of the steps for monochromatic or colourful median computation can be optimized, however no known improvement exists for the asymptotic time complexity. For completeness, in the following section we outline certain details and include pseudocode.

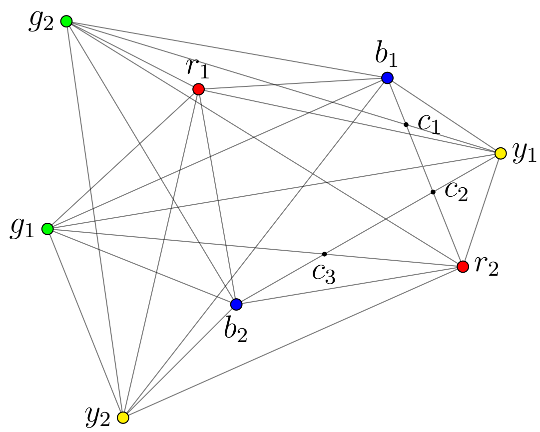

FIGURE 2. Examples of special points: c_1 and c_2 are special on edge (b_1, r_2). c_2 is also special on (y_1, b_2), along with c_3. The middle intersection on (y_1, b_2) is not special with respect to this edge, but it is with respect to (r_2, g_2). Some edges like (r_1, g_2) have no (special) intersection.

3.3. **Details and notes on implementation.** We define P, P^i, n_i, and $D(x, P)$ as in section 2.1. Our objective is to find a point x in the plane, maximizing $D(x, P)$. Trivially, x will be inside the convex hull of P, denoted $conv(P)$. The depth of x is defined as $\hat{\mu}(P)$. For an example, see Figure 3. Let S be the set of edges formed by all possible pairs of points (a, b), where $a \in P^i$, $b \in P^j$, $i < j$. These are the *colourful edges* mentioned in the preceding section. The intersection points of colourful edges will be referred to as *vertices*.

The following lemma for the colourful setting follows the same reasoning as the monochromatic setting (e.g., see [ALST03]).

Lemma 3.1. *To find a point with maximum colourful simplicial depth it suffices to consider the intersection points of the colourful edges in S (including the input points themselves).*

Proof. The colourful edges of S partition the union of all colourful triangles into polygonal cells. Unlike the monochromatic case, here some cells may not be convex (see cell r_3, b_1, r_2, i in Figure 3), and some points of $conv(P)$ may fall outside any cell (see Figure 2). Within the interior of any cell, the colourful simplicial depth remains constant. Consider any cell, C, such as the one bounded by vertices d, k, i, h, f in Figure 3. Let p be a point in the interior of C, and let q be a point in the interior of an edge of C, say (f, h). Consider the endpoint h

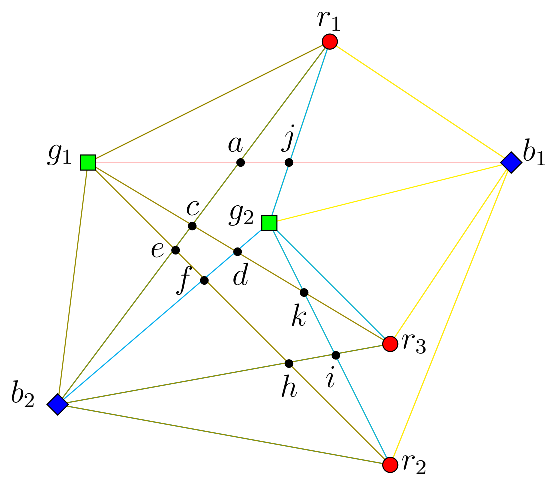

FIGURE 3. A configuration P of 7 points in $\mathbb{R}^2$, whose simplicial median has depth 6 and occurs at points $b_1, g_1, b_2, g_2, d, f, k$.

of this edge. Then the following inequality holds: $D(p, P) \leq D(q, P) \leq D(h, P)$, since any colourful simplices containing p also contain q, and any containing q also contain h. $\square$

Preprocessing: Counting points in halfplanes. Recall that, as we walk on an edge (a, b) and encounter edge (c, d) crossing at vertex t, we update depth in constant time. As mentioned, this is possible because of a preprocessing step where we count the number of points with colours other than c, d in each halfplane defined by (c, d). This can be done by brute force in linear time per edge (thus cubic time for all edges) but we can do better, not unlike the monochromatic case. This does not affect the overall asymptotic time complexity of the algorithm. Details are as follows.

Let $col(y)$ denote the colour of a point y. Given two points, y, z, such that $col(y) < col(z)$, the directed edge $s = \overrightarrow{yz}$ defines two open half-spaces: s^+ and s^-, where s^+ lies to the right of s, and s^- to the left. Denote the number of points in s^+ that have colours different from the endpoints of s by $r(s)$, and those in s^- by $l(s)$. Let $r^i(s)$ and $l^i(s)$ be the number of points of a colour i in s^+ and s^- respectively. Let $\bar{r}^i(s)$ and $\bar{l}^i(s)$ be the number of points of all k colours except for the colour i in s^+ and s^- respectively. For example, given an edge $s = \overrightarrow{yz}$, the aforementioned quantities are:

$$\bar{r}^{col(y)}(s) = \sum_{\substack{i=1,\\ i\neq col(y)}}^{k} r^i(s)\ , \qquad \bar{l}^{col(y)}(s) = \sum_{\substack{i=1,\\ i\neq col(y)}}^{k} l^i(s)\ . \tag{1}$$

Then

$$r(s) = \bar{r}^{col(y)}(s) - r^{col(z)}(s)\ , \qquad l(s) = \bar{l}^{col(y)}(s) - l^{col(z)}(s)\ . \tag{2}$$

As described in Section 2.2, for a given point p of P it is easy to compute the number of points in each of the $2(n-1)$ halfplanes determined by p and other points of P. This is done by sorting all points radially about p, then rotating a line and incrementing as necessary in constant time per point. In the colourful setting, we can choose to do this while only counting points of colour $col(p)$, or while ignoring only points of colour $col(p)$. Thus for $s = \overrightarrow{yz}$ we can obtain and store the required quantities (i.e., the righthand sides in Equation 2) to compute $r(s)$ and $l(s)$.

We can avoid explicitly sorting radially about each point by using the algorithm in [LC85] that finds all required sorted orderings in quadratic time.

Topological Sweep. To compute the CSD of all vertices, we carry out a topological sweep for a complete graph as presented by Rafalin and Souvaine [RS08]. This removes some of the overhead of the general topological sweep framework [EG89], notably in avoiding the use of *phantom vertices*. These are intersections of an extension of an edge with either another edge or with an extension of another edge.

Note that even though we could precompute the depth of all input points and special vertices as explained in sections 3.1 and 3.2, it is just as easy, and perhaps heuristically more efficient, to compute the depth of these positions from scratch (in $O(n \log n)$ time each) only when necessary, while performing the topological sweep. In fact by the nature of the sweep, we will end up computing the depth of at most one special vertex per colourful edge from scratch. The depth of each other vertex is computed in constant time during the sweep. Distinguishing vertices that are dealt with from scratch is done with the use of a flag for each edge (specifically, $ver()$, which is described later).

Combinatorially, the topological line that sweeps through the graph is a *cut*. It intersects every edge of the graph at most once, and separates the plane into two regions: one in which the depth of all points and vertices is known, and one to be discovered. The discovery of each new input point or vertex is called an *elementary step*. At each elementary step we compute the CSD of the discovered position. The topological sweep prioritizes processing vertices that have neighbouring vertices or input points with known depth. The prioritization of vertices is based on the topology of neighbouring processed positions with respect to the implied cut, but this detail is not important here. The reader is referred to [RS08] for details. In the pseudocode of Algorithm 2, we propagate the cut using $ver(s)$, which stores the last processed vertex on each edge s, along with its CSD. We also store the crossing edge that defines $ver(s)$ on s, and denote it by $cross(ver(s))$. Before starting the topological sweep, for each $s \in S$ we assign $ver(s) = \emptyset$. After completing an elementary step where we discover a vertex v that lies at the intersection of two edges, s_i and s_j, we assign $ver(s_i) \leftarrow v$, $ver(s_j) \leftarrow v$, $cross(ver(s_i)) \leftarrow s_j$, $cross(ver(s_j)) \leftarrow s_i$. We do this so that we can compute the CSD of a newly discovered vertex using the CSD of an adjacent one. When an elementary step discovers a input point, we compute its depth in $O(n \log n)$ time from scratch; see lines $6-7$ in Algorithm 2.

When we discover a new vertex v, we need to know if its depth can be computed in constant time via an update. We do not perform such an update if $ver(s_i) = \emptyset$ and $ver(s_k) = \emptyset$, as this implies that v is the first vertex to be discovered for both of its incident edges, which means that the only positions with known CSD adjacent to v are input points. In this case we compute the depth of v from scratch; see lines $10-11$ of Algorithm 2. Recall that when this happens, the other special vertices on the edges incident to v will end up being processed in constant time. As an example, in Figure 2, if c_2 happens to be discovered before c_1 and c_3, then we will not run the CSD algorithm for those two vertices. Instead they will be approached via a sequence of constant-time updates.

As mentioned in sections 3.1 and 3.2, while walking on a given edge s_i, if we know the depth at some vertex p on the intersection with edge s_j and then move to an adjacent vertex

v on s_k for some new k, it is easy to update depth in constant time. See Subroutine 1 as a reminder. The constant time update is possible if $ver(s_i)$ or $ver(s_k)$ are non-empty. If both are non-empty, we are free to update using either. This is handled in lines $12 - 19$ of Algorithm 2.

We maintain and update the deepest point found, using variables `median` and `max`.

Subroutine 1 `Computing` $D(v)$ `from` $D(p)$

```
Input: D(p), p, v, s_k, s_j. Output: D(v).
if v is to the left of s_j then
    D(v) ← D(p) − r(s_k);
else
    D(v) ← D(p) − l(s_k);
end if
if p is to the left of s_k then
    D(v) ← D(v) + r(s_j);
else
    D(v) ← D(v) + l(s_j);
end if
return D(v);
```

3.4. **Summary of Time and Space Complexity.** As preprocessing, we compute counts $r(s)$ and $l(s)$ for every s, i.e., for every pair of input points. This is done using quadratic time and space. Algorithm 2 can then look up these values whenever necessary.

The topological sweep algorithm in [RS08] runs in $O(K+nM)$ time, where n is the number of input points, M is the maximum number of edges cut by any topological line, and K is the number of *graph segments*. A graph segment is the subset of an edge between adjacent intersections with other edges. Given that $M = O(n^2)$ and $K = O(n^4)$, the time complexity of the topological sweep executed on our input is $O(n^4)$, not factoring in how to process each elementary step.

As already described, each of the n input points will be processed in $O(n \log n)$ time. The same may be true for at most one vertex per colourful edge, generating a total cost of $O(n^3 \log n)$. All other vertices cost $O(1)$. Thus the algorithm takes $O(n^4)$ time, where dominating complexity lies in the topological sweep.

We do not store all vertices, but only the list of all $ver()$, i.e., one vertex per edge. Thus the space used for this algorithm is $O(n^2)$, both for the sweep and for the list.

Theorem 3.2. *Given a set of input points $P \subset \mathbb{R}^2$ in general position in k different colours, $k \geq 3$, the colourful simplicial median of P can be found using $O(n^4)$ time and $O(n^2)$ space, where $|P| = n$.*

4. Conclusions and Questions

Our first main result, Theorem 2.5, is an algorithm computing the colourful simplicial depth of a point x relative to a configuration $P = \left(P^1, P^2, \ldots, P^k\right)$ of n points in $\mathbb{R}^2$ in k colour classes, with time complexity $O(n \log n)$, or $O(n)$ if the input is already sorted around x. Theorem 2.5 is optimal up to a constant factor given sorted input, or under the assumption that sorting is required. Alternatively, we expect that it is possible to drop this

Algorithm 2 Computing $\hat{\mu}(P)$

Input: $P, P^1, \ldots, P^k$. Output: $v, \hat{\mu}(P)$.

```
Compute all r(s), l(s), as described in Preprocessing (Section 3.3);
max ← 0;
for all s ∈ S, let ver(s) ← ∅;
while unprocessed vertices exist do                ▷ Start of the topological sweep.
    v ← next vertex provided by topological sweep;
    if v ∈ P then                                              ▷ Wlog let it be P_i
        D(v) = CSD of v with respect to P;
    else
        Identify s_i and s_k as edges intersecting at v.
        if ver(s_i) = ∅ & ver(s_k) = ∅ then            ▷ v has no adjacent vertices
            D(v) = CSD of v with respect to P;
        else if ver(s_i) ≠ ∅ then ▷ will update v using adjacent vertex on s_i
            p ← ver(s_i);
            s_j ← cross(ver(s_i));
            D(v) ← Subroutine 1 (D(p), p, v, s_j, s_k);
        else                          ▷ will update v using adjacent vertex on s_k
            p ← ver(s_k);
            s_j ← cross(ver(s_k));
            D(v) ← Subroutine 1 (D(p), p, v, s_j, s_i);
        end if
        ver(s_i) ← v, ver(s_k) ← v, cross(ver(s_i)) ← s_k, cross(ver(s_k)) ← s_i;
    end if
    if D(v) > max then
        max ← D(v);
        median ← v;
    end if
end while                                            ▷ End of the topological sweep.
return (median, max).
```

assumption by extending the $\Omega(n \log n)$ lower bound for computing monochromatic simplicial depth [ACG+02] to the colourful case.

The second main result, Theorem 3.2, is an algorithm computing the colourful simplicial median of a configuration $P = \left(P^1, P^2, \ldots, P^k\right)$ of n points in $\mathbb{R}^2$ in $O(n^4)$ time. This running time is optimal assuming we must generate all $\Theta(n^4)$ intersection points of the colourful edges formed by pairs of points from P. One would need to come up with a way of decreasing the pool of candidates for a colourful simplicial median, to improve this running time. We conjecture that this is not possible. The space used by our algorithm is $O(n^2)$.

Algorithm 2 returns a point that has maximum colourful simplicial depth along with its CSD. It is simple to modify the algorithm to return a list of all such points. We conjecture that the number of simplicial medians from the set of input points and the induced vertices is $O(n^2)$ and thus that maintaining such a list will not increase the required storage. A weaker version of this is also possible, where it is possible compute the output using $O(n^2)$ storage, but the output itself may be larger than $O(n^2)$.

For $(d+1)$ colours in $\mathbb{R}^d$, it is not clear how efficiently one can exhibit a single colourful simplex containing a given point [BO97, MS18a, MS18b].

It would be interesting to extend recent results on high-dimensional monochromatic depth [PWW17, ASS16] to the colourful setting. Another direction for future research is to obtain output-sensitive algorithms for colourful depth, for instance extending the work in [EE05].

Acknowledgments

This research was partially supported by an NSERC Discovery Grant to T. Stephen and by an SFU Graduate Fellowship to O. Zasenko. We thank A. Deza for comments on the presentation, and the anonymous referees for helpful comments.

A preliminary version of this work can be found in the proceedings of COCOA 2016 [ZS16].

References

[ABP+17] Karim A Adiprasito, Philip Brinkmann, Arnau Padrol, Pavel Paták, Zuzana Patáková, and Raman Sanyal, *Colorful Simplicial Depth, Minkowski Sums, and Generalized Gale Transforms*, International Mathematics Research Notices **2019** (2017), no. 6, 1894–1919.

[ACG+02] Greg Aloupis, Carmen Cortés, Francisco Gómez, Michael Soss, and Godfried Toussaint, *Lower bounds for computing statistical depth*, Comput. Stat. Data Anal. **40** (2002), no. 2, 223–229.

[ACNS82] M. Ajtai, V. Chvátal, M. Newborn, and E. Szemerédi, *Crossing-free subgraphs*, North-Holland Mathematics Studies **60** (1982), 9–12.

[Alo06] Greg Aloupis, *Geometric measures of data depth*, Data depth: robust multivariate analysis, computational geometry and applications, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 72, Amer. Math. Soc., Providence, RI, 2006, pp. 147–158.

[ALST03] Greg Aloupis, Stefan Langerman, Michael Soss, and Godfried Toussaint, *Algorithms for bivariate medians and a Fermat-Torricelli problem for lines*, Comput. Geom. **26** (2003), no. 1, 69–79.

[ASS16] Peyman Afshani, Donald R. Sheehy, and Yannik Stein, *Approximating the simplicial depth in high dimensions*, The European Workshop on Computational Geometry, 2016.

[Bár82] Imre Bárány, *A generalization of Carathéodory's theorem*, Discrete Math. **40** (1982), no. 2-3, 141–152.

[BO97] Imre Bárány and Shmuel Onn, *Colourful linear programming and its relatives*, Math. Oper. Res. **22** (1997), no. 3, 550–567.

[CO01] Andrew Y. Cheng and Ming Ouyang, *On algorithms for simplicial depth*, Proceedings of the 13th Canadian Conference on Computational Geometry, University of Waterloo, Ontario, Canada, August 13-15, 2001, 2001, pp. 53–56.

[DHST06] Antoine Deza, Sui Huang, Tamon Stephen, and Tamás Terlaky, *Colourful simplicial depth*, Discrete Comput. Geom. **35** (2006), no. 4, 597–615.

[DHST08] Antoine Deza, Sui Huang, Tamon Stephen, and Tamás Terlaky, *The colourful feasibility problem*, Discrete Appl. Math. **156** (2008), no. 11, 2166–2177.

[EE05] Amr Elmasry and Khaled M. Elbassioni, *Output-sensitive algorithms for enumerating and counting simplices containing a given point in th plane*, 17th Canadian Conference on Computational Geometry (CCCG'05) (Windsor, Canada), no. 17, University of Windsor, 2005, pp. 248–251.

[EEM11] Khaled Elbassioni, Amr Elmasry, and Kazuhisa Makino, *Finding simplices containing the origin in two and three dimensions*, International Journal of Computational Geometry & Applications **21** (2011), no. 5, 495–506.

[EG89] Herbert Edelsbrunner and Leonidas J. Guibas, *Topologically sweeping an arrangement*, J. Comput. System Sci. **38** (1989), no. 1, 165–194, 18th Annual ACM Symposium on Theory of Computing (Berkeley, CA, 1986).

[EM88] Herbert Edelsbrunner and Ernst Peter Mücke, *Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms*, Proceedings of the Fourth Annual Symposium on Computational Geometry (Urbana, IL, 1988), ACM, New York, 1988, pp. 118–133.

[FR05] Komei Fukuda and Vera Rosta, *Data depth and maximum feasible subsystems*, Graph theory and combinatorial optimization, GERAD 25th Anniv. Ser., vol. 8, Springer, New York, 2005, pp. 37–67.

[GHL+16] Ellen Gethner, Leslie Hogben, Bernard Lidický, Florian Pfender, Amanda Ruiz, and Michael Young, *On crossing numbers of complete tripartite and balanced complete multipartite graphs*, Journal of Graph Theory **84** (2016), no. 4, 552–565.

[GSW92] Joseph Gil, William Steiger, and Avi Wigderson, *Geometric medians*, Discrete Math. **108** (1992), no. 1-3, 37–51.

[KM90] Samir Khuller and Joseph S. B. Mitchell, *On a triangle counting problem*, Inform. Process. Lett. **33** (1990), no. 6, 319–321.

[LC85] D. T. Lee and Y. T. Ching, *The power of geometric duality revisited*, Inform. Process. Lett. **21** (1985), no. 3, 117–122.

[Lei83] F. T. Leighton, *Complexity issues in VLSI*, Foundations of Computing Series, MIT Press, 1983.

[Liu90] Regina Y. Liu, *On a notion of data depth based on random simplices*, Ann. Statist. **18** (1990), no. 1, 405–414.

[MS18a] Frédéric Meunier and Pauline Sarrabezolles, *Colorful linear programming, Nash equilibrium, and pivots*, Discrete Applied Mathematics **240** (2018), 78 – 91.

[MS18b] Wolfgang Mulzer and Yannik Stein, *Computational aspects of the colorful Carathéodory theorem*, Discret. Comput. Geom. **60** (2018), no. 3, 720–755.

[MW14] Jiří Matoušek and Uli Wagner, *On Gromov's method of selecting heavily covered points*, Discrete Comput. Geom. **52** (2014), no. 1, 1–33.

[PWW17] Alexander Pilz, Emo Welzl, and Manuel Wettstein, *From Crossing-Free Graphs on Wheel Sets to Embracing Simplices and Polytopes with Few Vertices*, 33rd International Symposium on Computational Geometry (SoCG 2017), vol. 77, 2017, pp. 54:1–54:16.

[RR96] Peter J Rousseeuw and Ida Ruts, *Bivariate location depth*, Applied Statistics: Journal of the Royal Statistical Society Series C **45** (1996), no. 4, 516–526.

[RS08] Eynat Rafalin and Diane L. Souvaine, *Topological sweep of the complete graph*, Discrete Appl. Math. **156** (2008), no. 17, 3276–3290.

[Sar15] Pauline Sarrabezolles, *The colourful simplicial depth conjecture*, J. Combin. Theory Ser. A **130** (2015), 119–128.

[Zas16] Olga Zasenko, *Colourful simplicial depth in the plane*, 2016, Java code, available at `http://github.com/olgazasenko/ColourfulSimplicialDepthInThePlane`, accessed February 2nd, 2017.

[ZS16] Olga Zasenko and Tamon Stephen, *Algorithms for colourful simplicial depth and medians in the plane*, Combinatorial Optimization and Applications - 10th International Conference, COCOA 2016, Hong Kong, China, December 16-18, 2016, Proceedings, Lecture Notes in Comput. Sci., vol. 10043, Springer, 2016, pp. 378–392.

Email address: `greg.aloupis@nyu.edu`

New York University, Tandon School of Engineering

Email address: `tamon@sfu.ca`

Email address: `ozasenko@sfu.ca`

Department of Mathematics, Simon Fraser University, British Columbia, Canada