Maximum Centre-Disjoint Mergeable Disks

Ali Gholami Rudi[*]

Department of Electrical and Computer Engineering
Babol Noshirvani University of Technology
Babol, Mazandaran, Iran
gholamirudi@nit.ac.ir

Abstract. Given a set of disks in the plane, the goal of the problem studied in this paper is to choose a subset of these disks such that none of its members contains the centre of any other. Each disk not in this subset must be merged with one of its nearby disks that is, increasing the latter's radius. This problem has applications in labelling rotating maps and in visualizing the distribution of entities in static maps. We prove that this problem is NP-hard. We also present an ILP formulation for this problem, and a polynomial-time algorithm for the special case in which the centres of all disks are on a line.

1. Introduction

A motivating example for the problem studied in this paper is the following about drawing text labels on a digital map that can be rotated: suppose there are a number of points on the map that represent map features. To each of these *feature points* a text label is assigned that describes the feature, like the name of a junction. When the map is rotated by the user, these labels must remain horizontal for the sake of readability, and therefore, they are rotated in the reverse direction around their feature

---

[*]Address for correspondence: Department of Electrical and Computer Engineering, Babol Noshirvani University of Technology, Babol, Mazandaran, Iran.

point (see the first two parts of Figure 1). Labels are difficult to read if they overlap, and therefore, only a non-overlapping subset of the labels are drawn on the map. If a label cannot be drawn because it overlaps with other labels, the text of its label must be appended to a nearby label that is drawn. The goal is to draw the maximum number of labels on the map such that none of them overlap when rotating the map. This is demonstrated in Figure 1. Part (a) shows four feature points and their labels. Part (b) shows the map when it is rotated 45 degrees counterclockwise; instead of rotating the map, the labels are equivalently rotated 45 degrees clockwise. Obviously the two labels on the left side of the map overlap, making the map of Part (a) infeasible. Part (c) shows what happens when these labels are merged. The remaining three labels never overlap when the map is rotated.

Figure 1. An example rotating map with 4 labels. (a) The initial configuration in which two of the labels overlap during rotation (the circles show the area covered by the labels during rotation). (b) After rotating the map 45 degrees counterclockwise. (c) Two of the labels are merged so that none of the label overlap during rotation.

Placing as many labels as possible on a map (known as map labelling) is a classical optimization problem in cartography and graph drawing [1]. For static maps, i.e. maps whose contents does not change, the problem of placing labels on a map can be stated as an instance of geometric independent set problem (also known as packing for fixed geometric objects): given a set of geometric objects, the goal is to find its largest non-intersecting subset. In the weighted version, each object also has a weight and the goal is to find a non-intersecting subset of the maximum possible weight.

A geometric intersection graph, with a vertex for each object and an edge between intersecting objects, converts this geometric problem to the classical maximum independent set for graphs, which is NP-hard and difficult to approximate even within a factor of $n^{1-\epsilon}$, where n is the number of vertices and ϵ is any non-zero positive constant [2]. Although the geometric version remains NP-hard even for unit disks [3], it is easier to approximate, and several polynomial-time approximation schemes (PTAS) have been presented for this problem [4, 5, 6, 7, 8].

Dynamic maps allow zooming, panning, or rotation, and labelling in such maps seems more challenging. Most work on labelling dynamic maps consider zooming and panning operations [9]. Gemsa et al. [10] were the first to formally study labelling rotating maps. With the goal of maximising the total duration in which labels are visible without intersecting other labels, they proved the problem to be NP-hard and presented a $1/4$-approximation algorithm and a PTAS, with the presence of restrictions

on the distribution of labels on the map. Heuristic algorithms and Integer Linear Programming (ILP) formulations have also been presented for this problem [11, 12]. Note that in these problems, invisible labels do not get merged with visible labels. Yokosuka and Imai [13] examined a variant of this problem, in which all of the labels are always present in the solution and the goal is to maximise their size.

A related problem is crushing disks [14], in which a set of prioritized disks are given as input, whose radii grow over time, as map labels do when zooming in. When two disks touch, the one with the lower priority disappears. The radii of the disks grow linearly, and when a disk disappears, the radius of the other disk does not change. The goal is to find the order in which disks disappear and the process finishes when only one disk remains.

In this paper, we investigate a problem similar to geometric independent set for a set of disks, except that i) the disks in the output must be centre-disjoint (none of them can contain the centre of another) but they may overlap, ii) each disk that does not appear in the output must be merged with a disk, containing its centre, that does. When a disk is merged with another, the radius of the latter is increased by the radius of the former. Also to preserve the locality of the merges, a disk A can be merged with another disk B, only if all disks closer to B than A (considering the distance between disk centres) are also merged with B, and after merging these closer disks, B must contain the centre of A. This problem is formally defined in Section 2. We prove this problem to be NP-hard via a reduction from Planar Monotone 3-SAT [15]. Note that even without this restriction on merge order, the problem remains NP-hard; we have presented a PTAS in an earlier paper [16] for the case in which this restriction on merge order is not assumed.

To observe how the introductory example at the beginning of this section reduces to this problem, consider the disks in Figure 1 (a). The disk centred at each feature point shows the region covered by its label during rotation. Only if a disk contains the centre of another, their corresponding labels intersect at some point during rotation. As another application of this problem, centre-disjoint disks can show the distribution of facilities in an area. For instance, Figure 2 shows the distribution of schools in Munich. It was obtained by placing a disk of radius 50 meters on each school (the coordinates of schools were obtained from OpenStreetMap data). Then, an ILP (Section 4.1) was used to obtain the maximum number of centre-disjoint disks in our problem

Note that the centre-disjointness property of disks, which we assume for the output of our problem, is also used in the definition of transmission graphs of a set of disks, in which a vertex is assigned to each disk and a directed edge from a disk to another shows that the former contains the centre of the latter [17]. These graphs have been studied, for instance, for computing their spanners (a subgraph to estimate the distance between pairs of vertices) [18], counting their number of triangles and computing their girth [17], and their recognition [19]. Transmission graphs show the static relation between input disks and do not directly help us in our problem, in which the goal is to find a set of radius-changing disk merges, which makes them centre-disjoint.

To obtain an efficient algorithm for the problem, we also examine a more restricted version, in which the centres of input disks are on a line (Section 4). For this version, we present a polynomial-time algorithm that incrementally obtains a set of centre-disjoint disks with the maximum size. Note that the assumption of collinear inputs have been applied to many other challenging problems, such as [20].

Figure 2. The distribution of schools in Munich; disks corresponding to neighbouring schools were merged using the ILP of Section 4.1 to obtain larger, centre-disjoint disks.

This paper is organised as follows. In Section 2 we introduce the notation used in this paper and formally state the problem. Then, in Section 3 we show that the problem studied in this paper is NP-hard. In Section 4, we present algorithms for solving this problem: we present an ILP formulation for solving the general case of the problem, and a polynomial-time dynamic programming algorithm for the case in which all disk centres are on a line. Finally, in Section 5 we conclude this paper.

2. Notation and preliminary results

Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of n disks. The radius of d_i is denoted as r_i, and its centre is denoted as p_i.

Definition 2.1. A function ϕ from D to itself is an *assignment*, if $\phi(\phi(d_i))$ is $\phi(d_i)$ for every d_i in D. According to an assignment ϕ, the disks in D can be either *selected* or *merged*: if $\phi(d_i)$ is d_i, the disk d_i is selected, and otherwise, it is merged. The cardinality of an assignment, denoted as $|\phi|$, is the number of selected disks in ϕ.

The relation defined by assignments (Definition 2.1) describes disk merges in our problem. For any disk d_i, if we have $\phi(d_i) = d_j$ and $i \neq j$, it implies that d_i is merged with d_j. On the other hand, the relation $\phi(d_i) = d_i$ implies that d_i is a selected disk and is not merged with any other disk. Since a disk can be merged with selected disks only, for any disk d_i, we have $\phi(\phi(d_i)) = \phi(d_i)$.

Definition 2.2. The aggregate radius of a selected disk d_i with respect to an assignment ϕ, denoted with some misuse of notation as $r_i(\phi)$, is the sum of its radius and that of every disk merged with it, or equivalently,

$$r_i(\phi) = \sum_{j: \, \phi(d_j)=d_i} r_j.$$

Let δ_i be the sequence of disks in $D \setminus \{d_i\}$, ordered increasingly by the distance of their centres from the centre of d_i, and let $\delta_i(j)$ denote its j-th disk. The j-th aggregate radius of d_i, denoted as $r_i(j)$, is defined as its aggregate radius if $\{\delta_i(1), \delta_i(2), \ldots, \delta_i(j)\}$ are merged with d_i.

We now define proper assignments (Definition 2.3). In the rest of this paper, the distance between two disks is defined as the Euclidean distance between their centres.

Definition 2.3. An assignment ϕ is *proper* if it meets the following conditions.

1. The disk $\delta_i(j)$ can be merged with d_i, only if $\delta_i(k)$, for every k where $1 \leq k < j$, are also merged with d_i. In other words, all disks closer to d_i than $\delta_i(j)$ are also merged with d_i.

2. The disk $\delta_i(j)$ can be merged with d_i, only if the distance between the centre of d_i and $\delta_i(j)$ is less than $r_i(j-1)$. In other words, after merging $\delta_i(k)$ for $1 \leq k < j$, d_i must contain the centre of $\delta_i(j)$.

3. Selected disks must be centre-disjoint with respect to their aggregate radii; i.e. none of them can contain the centre of any other selected disk. More precisely, for indices i and j such that $i \neq j$, $\phi(d_i) = d_i$, and $\phi(d_j) = d_j$, we have $|p_i p_j| \geq \max(r_i(\phi), r_j(\phi))$.

Note the first two items in Definition 2.3 ensure the locality of the merges, which is especially important in the labelling application mentioned in the Introduction.

Definition 2.4. Given a set of disks, in the Maximum Centre-Disjoint Mergeable Disks Problem (MCMD), the goal is to find a proper assignment of the maximum possible cardinality.

Figure 3. An example set of disks with two proper assignments: Either all disks are merged with d_1, which gives a proper assignment of cardinality 1, or just d_3 is merged with d_2, which gives a proper assignment of cardinality 4. The latter is a solution to MCMD, because it has the maximum cardinality.

Figure 3 shows a configuration of five disks with more than one proper assignment. Disk d_3 can be merged with d_1, after which, d_1 would contain the centre of d_4 and d_5, both of which then have to be merged with d_1. These merges result in d_1 containing the centre of d_2, which would also be merged. Therefore, in this assignment ϕ_1, we have $\phi_1(d_i) = d_1$, for $1 \leq i \leq 5$, and its cardinality is one. Alternatively, in assignment ϕ_2 we can merge d_3 with d_2, as the latter contains the centre of the former. The remaining disks are centre-disjoint. Therefore, we have $\phi_2(d_1) = d_1$, $\phi_2(d_2) = d_2$, $\phi_2(d_3) = d_2$, $\phi_2(d_4) = d_4$, $\phi_2(d_5) = d_5$, and its cardinality is four. Assignment ϕ_2 maximises the number of selected disks, and is a solution to MCMD for the configuration of disks in Figure 3.

Not every set of disks has a proper assignment. Figure 4 shows an example. Disk d_3 can be merged with either d_1 or d_2. If d_3 is merged with d_1, d_5 cannot be merged with d_2, because of the second condition of proper assignments: d_5 can be merged with d_2, only if all closer disks to d_2 are merged with it (but d_3 which is closer to d_2 than d_5 is not). Therefore, d_5 can be neither merged, nor selected (because its centre is contained in d_2). Similarly, if d_3 is merged with d_2, d_4 can neither be merged nor selected. Thus, there exists no proper assignment for these set of disks. In Section 3.2 we introduce a variant of MCMD by relaxing the second condition of Definition 2.3, in which every instance has a solution.

Figure 4. An example set of disks with no proper assignment. Either d_3 and d_4 are merged with d_1, after which d_5 cannot be merged with d_2 (but must be), or d_3 and d_5 are merged with d_2, after which d_4 cannot be merged with d_1 (but must be).

3. Hardness of maximum centre-disjoint mergeable disks

Instead of proving that the decision version of MCMD (Definition 3.1) is NP-complete, we show that even deciding whether a set of disks has a proper assignment (Definition 3.2) is NP-complete (clearly the latter implies the former). To do so, we perform a reduction from the NP-complete PLANAR MONOTONE 3-SAT (Definition 3.3) [21] to PROPER MCMD (Definition 3.2).

3.1. Hardness of MCMD

Definition 3.1. In the k-MCMD problem, we are given a set of disks and we have to decide if there exists a proper assignment of cardinality at least k or not.

Definition 3.2. In the PROPER MCMD problem, we are given a set of disks and we have to decide if there exists a proper assignment.

Definition 3.3. MONOTONE 3-SAT is a variant of 3-SAT, in which all variables of each clause are either positive or negative. An instance of MONOTONE 3-SAT is called PLANAR, if it can be modeled as a planar bipartite graph with parts V corresponding to variables and C corresponding to clauses; each vertex in C is incident to at most three variables, which correspond to the variables that appear in the clause.

Figure 5.   A monotone rectilinear representation of a PLANAR MONOTONE 3-SAT instance with three clauses and four variables. Horizontal segments on the x-axis denote the variables, and horizontal segments above and below the x-axis denote positive and negative clauses, respectively: $c_1 = v_1 \vee v_2 \vee v_3$, $c_2 = v_1 \vee v_3 \vee v_4$, $c_3 = \neg v_1 \vee \neg v_2 \vee \neg v_4$.

Deciding if an instance of PLANAR MONOTONE 3-SAT is satisfiable is NP-complete [15]. It can be proved that every instance of PLANAR MONOTONE 3-SAT has a *monotone rectilinear represen-tation* (Definition 3.4; an example is shown in Figure 5), and also, if for every instance of PLANAR MONOTONE 3-SAT its monotone rectilinear representation is also given, the problem remains NP-Complete [15].

Definition 3.4. A *monotone rectilinear representation* of an instance of PLANAR MONOTONE 3-SAT is a drawing of the instance with the following properties: i) Variable are drawn as disjoint horizontal segments on the x-axis, ii) positive clauses are drawn as horizontal segments above the x-axis, iii) negative clauses are drawn as horizontal segments below the x-axis, iv) an edge is drawn as a vertical segment between a clause segment and the segments corresponding to its variable, and v) the drawing is crossing-free.

Figure 5 shows a monotone rectilinear representation of an instance of PLANAR MONOTONE 3-SAT with three clauses. Lemma 3.5 shows how to map a PLANAR MONOTONE 3-SAT to a two-dimensional integer grid (its proof is presented at the end of this section).

Lemma 3.5. For an instance of PLANAR MONOTONE 3-SAT with v variables and c clauses, there exists a monotone rectilinear representation on a two-dimensional integer grid with $c + 1$ rows and $3c + v$ columns, such that horizontal segments, which represent variables and clauses, appear on horizontal grid lines, and vertical segments appear on vertical grid lines.

To prove that PROPER MCMD is NP-complete, we perform a reduction from PLANAR MONO-TONE 3-SAT to PROPER MCMD. To do so, we create an instance of PROPER MCMD from the monotone rectilinear representation of any instance of PLANAR MONOTONE 3-SAT in Theorem 3.6. In our construction, we use two types of disks:

- Disks, which by our construction, are always selected (their centres can never be inside any other disk). We call them *s-disks* for brevity.

- Disks of very small radius, which are contained in at least one s-disk, and thus, are surely merged in our construction. We call these disks *m-disks*. We assume that the radius of m-disks is so small compared to the radius of s-disks that after merging any number of m-disks with an s-disk, the centre of no new disk would enter the s-disk in our configuration. In the instance of PROPER MCMD that we construct, each s-disk contains at least one m-disk.

We create a configuration of disks using *gadgets*, each of which consists of some m-disks and s-disks. The m-disks of a gadget are either internal (internal m-disks) or can be shared with other gadgets (shared m-disks). Parts (a) and (b) of Figure 6 show two gadgets (from each gadget, only an s-disk and an m-disk is shown). In Figure 6 (c) these two gadgets are joined at m-disk m. In a proper assignment, m is merged either with an s-disk of A or with an s-disk of B. With respect to gadget A, if m is merged with A in a proper assignment, we say that it is *merged in*, and otherwise, *merged out* with respect to A.

Figure 6.   Two gadgets (Parts (a) and (b)), joined at one of their m-disks (Part (c)).

We use the following gadgets in our construction. The gadgets and the distance between shared m-disks of each of them are shown in Figure 7; s-disks (denoted as s_i) are large disks and m-disks (denoted as m_i) are small disks. More details about these gadgets are provided now:

- Input: This gadget has only one shared m-disk, which can be either merged in or merged out.

Figure 7. Gadgets used in the proof of Theorem 3.6; s_i and m_i for different values of index i denote s-disks and m-disks, respectively. Shared m-disks are indicated with overlines (like m_1 in Input). The sizes of the gadgets are specified such that the gadgets fit together in the proof of Theorem 3.6.

- Copy: We use two gadgets for copy in our construction: one with two m-disks and one with four (both of them are demonstrated in Figure 7), which we reference as 2-Copy and 4-Copy, respectively. The logic behind both of them is similar and is explained thus. If m_1 is merged in, m_2 (also m_5 and m_6 if present) is merged out, and if m_1 is merged out, m_2 (also m_5 and m_6) is merged in. To see why, note that m_3 can be merged either with s_1 or with s_2. If m_3 is merged with s_1, both m_1 and m_4 must also be merged with s_1, because m_3 is farther than both to s_1. Since m_4 is merged with s_1, m_2 (also m_5 and m_6) cannot be merged with s_2 and therefore they must be merged out. Similarly, if m_3 is merged with s_2, m_2 (also m_5 and m_6) must be merged with s_2 as well, and m_1 must be merged out.

- Disjunction: One or more of its shared m-disks are merged in. Clearly, m_4 must be merged with s_1, s_2, or s_3. If it is merged with s_i ($i \in \{1, 2, 3\}$), m_i must also be merged with s_i, and m_j ($j \neq i$) may or may not be merged in.

- Not: Either both m_1 and m_2 are merged in or both of them are merged out. This is because m_4 can be merged either with s_1 or s_2. If it is merged with s_1, m-disks m_1, m_2, and m_3 must also be merged with s_1, because m_4 is farther than all of them. Otherwise, if m_4 is merged with s_2, m-disk m_3 must also be merged with s_2 and therefore, none of m_1 and m_2 can be merged with s_1, because m_3 (which is closer than both) is not merged with s_1. Thus, m_1 and m_2 must merge out.

In our construction of the proof of Theorem 3.6, some of the shared m-disks of 4-Copy gadget are unused and are not shared with any other gadget; for such instances of 4-Copy, their unused shared m-disks must be removed.

Theorem 3.6. PROPER MCMD is NP-complete.

Proof:
It is trivial to show that PROPER MCMD is in NP. To show that it is NP-hard, we reduce PLANAR MONOTONE 3-SAT to PROPER MCMD. Let I be an instance of PLANAR MONOTONE 3-SAT, with variables V and clauses C. Based on Lemma 3.5, there exists a monotone rectilinear representation of I on a $(|C| + 1) \times (3 \cdot |C| + |V|)$ integer grid. Let R denote this representation.

We create an instance of PROPER MCMD from R as follows. The transformation is demonstrated in Figure 8, which corresponds to the monotone rectilinear representation of Figure 5.

Figure 8. A PROPER MCMD instance obtained from the PLANAR MONOTONE 3-SAT instance of Figure 5.

1. We replace the segment corresponding to a variable in R with an Input gadget and a chain of 2-Copy gadgets. For each intersection of this segment with a vertical segment, a 4-Copy gadget is used.

2. Let s be a horizontal segment corresponding to a clause in R. Three variables appear in the clause, for each of which there is a vertical segment that connects s to a variable segment. For the first and last intersections, 4-Copy gadgets are used. For the 2nd intersection, we use a Disjunction gadget. These gadgets are connected using two chains of 2-Copy gadgets.

3. For each vertical segment that connects a variable segment to a clause segment above the x-axis, we use a chain of 2-Copy gadgets to connect the 4-Copy gadget of the variable segment to the 4-Copy or Disjunction gadget (if it is the 2nd intersection) of the clause segment. For segments that appear below the x-axis, we do likewise, except that we place a Not gadget before the chain of 2-Copy gadgets.

Note that some of the gadgets of Figure 7 need to be rotated or mirrored, for instance, in the vertical chains that connect clauses and variables. Also note that based on the sizes shown in Figure 7, shared m-disks always appear on grid lines in our construction. Since the distance between the shared m-disks of any of the gadgets is at least 0.25, at most four gadgets can appear on a grid segment of unit length. Given that the total area of the grid, and therefore the total length of grid segments, is bounded by $O(|C|^2)$, the number of gadgets used in the resulting instance of PROPER MCMD is at most $O(|C|^2)$. Thus, the size of the resulting PROPER MCMD instance is polynomial in terms of the size of the input PLANAR MONOTONE 3-SAT instance.

Suppose there is a proper assignment for our PROPER MCMD instance. We obtain an assignment A of the variables of our PLANAR MONOTONE 3-SAT instance as follows. We assign one to a variable if the m-disk of its corresponding Input gadget is merged out, and assign zero otherwise. Consider any clause c in our PLANAR MONOTONE 3-SAT instance. Let g be the Disjunction gadget corresponding to c.

If c is a positive clause, a chain of 2-Copy gadgets connects the Input gadget of each of the variables that appear in c to g. Therefore, if variable v appears in the clause and if the shared m-disk of the Input gadget corresponding to v is merged out, the m-disk of the last 2-Copy gadget of its chain is merged in inside g. Since, one or more of the shared m-disks of g are merged in, at least one of the literals in g is satisfied. Similarly, if c is a negative clause, because there is a Not gadget in the chain that connects each variable v of c to its Disjunction gadget, if the shared m-disk of the Input gadget corresponding to v is merged out, the m-disk of the last 2-Copy gadget of its chain is also merged out inside g. Since, one or more of the shared m-disks of g are merged in, at least one of the variables in g is not satisfied.

Therefore, the PLANAR MONOTONE 3-SAT instance is satisfied with assignment A.

For the reverse direction, suppose there exists an assignment A of the variables, for which all clauses of I are satisfied. We can obtain a proper assignment in our PROPER MCMD instance as follows. For each variable v in V, if v is one, the shared m-disk of the Input gadget corresponding to v is merged out, and otherwise, it is merged in. Let c be a positive clause in which variable v with value one appears (since c is satisfied in A, variable v must exist), and let g be the disjunction gadget

corresponding to clause c. Since v is merged out, the m-disk of the last 2-Copy gadget that connects the gadget corresponding to v to g is merged in with respect to g. This implies that one of the shared m-disks of the Disjunction gadget of each positive clause is merged in. We can similarly show that at least one of the shared m-disks of the Disjunction gadgets corresponding to negative caluses are also merged in. This yields a proper assignment for the PROPER MCMD instance.   □

In Corollary 3.7 we show that even if all disks have the same radius, the problem remains NP-hard.

Corollary 3.7. PROPER MCMD remains NP-complete, even if all disks are required to be of the same radius.

Proof:
We fix the radius of m-disks to $r = 0.01$. We use the same construction as Theorem 3.6, with the difference that we replace each s-disk with a number of smaller disks of radius r with the same centre, so that the sum of the radii of these smaller disks equals the radius of the s-disk. Since the disks added for each s-disk are not centre-disjoint, and their centre cannot be contained in some other disk, exactly one of them must be selected and after merging others, it reaches the size of the original s-disk. The rest of the proof of Theorem 3.6 applies without significant changes.   □

In the proof of Corollary 3.7, we can adjust the position of the disks that replace each s-disk so that their centres do not coincide: they can be placed evenly on a very short line segment (for instance of length 0.0001). However, that they cannot be centre-disjoint, as they are to be merged.

Now we present the proof of Lemma 3.5.

Proof:
Let R be a monotone rectilinear representation of a PLANAR MONOTONE 3-SAT instance (such a representation certainly exists [15]). By extending horizontal segments of R we get at most $c + 1$ lines: one for the variables (the x-axis) and at most c for clauses. Let $\ell_1, \ell_2, \ldots, \ell_m$ be the lines that appear above the x-axis ordered by their y-coordinates. We move them (together with the segments appearing on them) so that, ℓ_i is moved to $y = i$; vertical segments that connect them to a segment on the x-axis may need to be shortend or lengthened during the movement. Given that the x-coordinate of the end points of horizontal segments, and also the vertical order of the segments, do not change, no new intersection is introduced by this transformation. The same is done for the lines that appear below the x-axis.

Repeating the same process for vertical segments, we get at most $3c$ vertical lines. We can similarly move these lines and the segments on them horizontally so that they appear in order and consecutively on vertical integer grid lines. Variables that do not appear in any clause, can be placed in at most v additional vertical grid lines. This results in a $(c + 1) \times (3c + v)$ grid.   □

3.2.   Relaxing merge order

Due to the first condition of proper assignments (Definition 2.3), in a proper assignment ϕ of a set of disks D, a disk d_i can be merged with another disk d_j, only if all closer disks to d_i than d_j are also merged with d_i. This condition, in addition to the second condition of Definition 2.3 (disk d_i

may be merged with d_j only if d_j contains the centre of d_i after disks closer to d_j than d_i are merged with d_j), ensures the locality of the merges. By requiring this ordering for merges, however, we get instances for which there is no solution, such as the one demonstrated in Figure 4. For such instances, a solution can be obtained by relaxing this condition. In this section, we relax the first condition of Definition 2.3.

Definition 3.8. In an assignment ϕ for a set of disks D, let δ_i^{ϕ} denote the sequence of disks assigned to selected disk d_i, ordered by their distance to d_i. Also, let $\delta_i^{\phi}(j)$ denote the j-th disk in this sequence.

Definition 3.9. An assignment ϕ is *uproper* (short for unordered proper) if it meets the following conditions.

1. For each pair of possible indices i and j, in which $\phi(d_j) = d_i$, choose k such that $\delta_i^{\phi}(k) = d_j$. The distance between d_i and d_j must be at most $r_i + \sum_{x=1}^{k-1} r_{\delta_i^{\phi}(x)}$. In other words, after merging all closer disks in δ_i^{ϕ}, d_i must contain the centre of d_j.

2. Selected disks must be centre-disjoint with respect to their aggregate radii; i.e. none of them can contain the centre of any other selected disk.

Definition 3.10. Given a set of disks, the goal in the Relaxed Maximum Centre-Disjoint Mergeable Disks Problem (RMCMD) is to find a uproper assignment of the maximum possible cardinality.

Theorem 3.11 shows that any set of disks has a uproper assignment, and therefore, RMCMD always has a solution.

Theorem 3.11. There exists at least one uproper assignment for any set of disks D.

Proof:
Let S be the largest subset of D for which there exists a uproper assignment ϕ_S. If $S = D$, we are done; so we assume otherwise. Let d_k be any disk in $D \setminus S$. The centre of d_k is contained in at least one disk d_i of S (considering its aggregate radius); otherwise, d_k may be added to ϕ as a selected disk, which contradicts the choice of S.

We obtain a uproper assignment $\phi_{S'}$ for $S' = S \cup \{d_k\}$ as follows. Initially we set $\phi_{S'}(d_x) = \phi_S(d_x)$ for every $d_x \in S$. We also set $\phi_{S'}(d_k) = d_i$, as d_i contains the centre of d_k. Merging d_k increases the aggregate radius of d_i. If $\phi_{S'}$ is not uproper, d_i must contain the centre of another disk d_j such that $\phi_{S'}(d_j) = d_j$ (it must be a selected disk in ϕ_S). We modify $\phi_{S'}$ so that $\phi_{S'}(d_x) = d_i$ for every $d_x \in \delta_j^{\phi} \cup \{d_j\}$ (here, with some abuse of notation, assume δ_j^{ϕ} to be a set). These changes satisfy the second condition of Definition 3.10: after merging d_j with d_i, d_i contains d_j completely (because the centre of d_j was inside d_i before the merge and after it, the aggregate radius of d_i is increased by r_j), and therefore it must contain the centre of $\delta_j^{\phi}(1)$. We then merge $\delta_j^{\phi}(1)$ with d_i, then we can merge $\delta_j^{\phi}(2)$ with d_i, and so on.

If, after these changes, $\phi_{S'}$ is not uproper, then d_i contains the centre of another selected disk d_y. For every such disk, we similarly merge with d_i, d_y and every disk merged with it, until d_i does not

contain the centre of any other disk, and then, $\phi_{S'}$ becomes proper. This contradicts the choice of S and implies that we have a uproper assignment for D. □

To show that RMCMD is NP-hard, in Theorem 3.12 we reduce the PARTITION problem to RM-CMD. In PARTITION, we are given a set of positive integers and have to decide if there is a subset, whose sum is half of the sum of all numbers in the input list. PARTITION is known to be NP-complete [22].

Theorem 3.12. RMCMD is NP-hard.

Proof:
We reduce PARTITION to RMCMD. Let $A = \{a_1, a_2, \ldots, a_n\}$ be an instance of PARTITION and s be the sum of the members of A. Also let e be a real number such that $0 < e < 1$; we use e to create distances smaller than a segment of unit length. We create an instance of RMCMD as follows (see Figure 9).

1. Add disk d_1 of radius $2s$ and add d_2 with the same radius at distance $3s$ on the right of d_1.

2. Add d_3 at distance $5s/2 + e$ above d_1 with radius s. Similarly, add d_4 at distance $5s/2 + e$ above d_2 with the same radius.

3. Add one disk for each member of A in the midpoint of the centres of d_1 and d_2, such that the radius of the one corresponding to a_i is a_i.

Figure 9. The construction in the proof of Theorem 3.12. Only if we have a uproper assignment of cardinality four, input numbers can be divided into two partitions of equal sum.

Let ϕ be the solution of this RMCMD instance. We show that there is a valid solution to the PARTITION instance if and only if the cardinality of ϕ is four.

Suppose X is a subset of A with sum $s/2$. We obtain an assignment from X as follows: every disk corresponding to a member of X is assigned to d_1 and others are assigned to d_2. Since the sum of the members of X is $s/2$, the aggregate radii of both disks are exactly $5s/2$. Therefore, the centre of d_3 and d_4 are outside these disks. This yields a uproper assignment of cardinality 4.

For the reverse direction, suppose the cardinality of ϕ is four (note that it cannot be greater). If so, all of d_1, d_2, d_3, and d_4 are selected, and therefore, the aggregate radii of d_1 and d_2 are lower than $5s/2 + e$. Given that the sum of the radii of the disks corresponding to members of A is s (which is an integer) and $0 < e < 1$, the sum of the set of disks assigned to d_1 and d_2 (and therefore the subsets of A corresponding to them) are equal. $\qquad\square$

We cannot extend Theorem 3.12 to the case in which all disks have the same radius. The problem is that the radii of the disks may be arbitrarily large. This is necessary in Theorem 3.12, because the partition problem is weakly NP-complete, and using fixed-radius disks (the technique we used in Corollary 3.7 for MCMD) to construct larger disks may imply a number of disks that is not polynomial in the input size.

4. Algorithms for MCMD

In this section we present algorithms for solving MCMD. We present an ILP formulation for general MCMD instances in Section 4.1, and in Section 4.2 we present a dynamic programming algorithm for MCMD instances in which disk centres are collinear.

4.1. ILP formulation of MCMD

Theorem 4.1. Any instance of MCMD with n disks can be formulated as an integer linear programme with $O(n^2)$ binary variables.

Proof:
For $1 \le i, j \le n$, we introduce a binary variable $x_{i,j}$. If $i \ne j$, $x_{i,j}$ indicates whether disk d_i is merged with disk d_j. If $i = j$, it shows if disk d_i is a selected disk. The following constraint for $1 \le i \le n$ makes sure that each disk is either selected or merged with another disk.

$$\sum_{j=1}^{n} x_{i,j} = 1 \tag{1}$$

An assignment ϕ can be obtained from the values of variables $x_{i,j}$ by letting $\phi(d_i) = d_j$ if and only if $x_{i,j} = 1$.

Based on the conditions enumerated in Definition 2.3, we add additional constraints to make sure that the obtained assignment ϕ is proper. Let δ_i be as defined in Definition 2.2.

1. Based on the first condition of Definition 2.2, a disk can be merged with d_i, only if its closer disks are merged as well. In other words, $x_{\delta_i(j+1)i}$ can be one, only if $x_{\delta_i(j)i}$ is also one for $1 \le j \le n-2$, as expressed in the following constraint for $1 \le i \le n$ and $1 \le j \le n-2$.

$$x_{\delta_i(j)i} \ge x_{\delta i(j+1)i} \tag{2}$$

2. Based on the second condition of Definition 2.2, d_j can be merged with d_i, only if the distance of d_i and d_j is less than $r_i(j-1)$ (Definition 2.2) for $1 \leq j \leq n-1$; that is, after merging with d_i all disks closer than d_j to d_i, the resulting disk must contain the centre of d_j. We disallow merges that fail this condition: for $1 \leq i \leq n$ and $1 \leq j \leq n-k$, where $r_i(j-1) < |p_ip_j|$, we add the following constraint (for simplicity, let $r_i(0) = r_i$).

$$x_{ji} = 0 \tag{3}$$

3. Based on the third condition of Definition 2.2, selected disks must be centre-disjoint. We add the following constraint for $1 \leq i, j \leq n$.

$$\sum_{k=1}^{n} r_k \cdot x_{ki} \leq |p_ip_j| + \infty \cdot (1 - x_{jj}) \tag{4}$$

Obviously, the left side computes the aggregate radius of d_i; note that if d_i is not selected, the left side equals zero and the inequality is trivially satisfied. There are two cases based on whether d_j is selected or not. If d_j is a selected disk ($x_{jj} = 1$) the right side of the inequality simplifies to $|p_ip_j|$, making sure that d_i does not contain d_j. If, on the other hand, d_j is merged with another disk (maybe even with d_i), the right side of the inequality simplifies to $+\infty$ and the constraint is satisfied.

Finally, as the goal of MCMD (Definition 2.4) is to maximize the number of selected disks, the objective of the programme is simply to maximise $\sum_{i=1}^{n} x_{ii}$. ☐

The number of variables used in the integer programme of Theorem 4.1 can be reduced based on the following observation. The value of some variables of an MCMD instance is always zero by constraints of type 3. These variables can be removed. The implementation of the ILP of Theorem 4.1 with this optimization is publicly available[1]; it has been used to obtain Figure 2.

4.2. Collinear disk centres

In this section we present a polynomial-time algorithm for solving MCMD for a set of disk with collinear centres. Note that even if disk centres are collinear, there may exist no proper assignments, as demonstrated in Figure 4. In the rest of this section, let $\lambda = \langle d_1, d_2, \ldots, d_n \rangle$ be a sequence of input disks D, ordered by the x-coordinate of their centres. We assume that the centres of the members of D are collinear and are on the y-axis. We need Definitions 4.2 and 4.3 to present the algorithm in Theorem 4.4.

Definition 4.2. Let ϕ be an assignment of $\{d_1, d_2, \ldots, d_n\}$ and let ϕ' be an assignment of $\{d_1, d_2, \ldots, d_x\}$, such that $x \leq n$. ϕ is *an extension* of ϕ', if for every disk d_i in $\{d_1, d_2, \ldots, d_x\}$, we have $\phi(d_i) = \phi'(d_i)$. In other words, every selected disk in ϕ' is also a selected disk in ϕ, and every merged disk in ϕ' is also merged with the same disk in ϕ. Equivalently, when ϕ *is limited to* $\{d_1, d_2, \ldots, d_x\}$, ϕ' is obtained.

---

[1] https://github.com/nit-ce/mcmd.git

Definition 4.3. $M(x, y, z)$ denotes the maximum cardinality of a proper assignment of $X = \{d_1, d_2, \ldots, d_x\}$, such that the following conditions are met (we have $y \leq x \leq z \leq n$).

1. d_y is its right-most selected disk.

2. $d_{y+1}, d_{y+2}, \ldots, d_x$ are all merged with d_y.

3. d_z is the right-most disk in D, where $z \geq x$, whose centre is contained in d_y considering its aggregate radius.

Note that by the third condition of Definition 4.3, the centres of $d_{x+1}, d_{x+2}, \ldots, d_z$ are inside d_y with respect to D, but they are not merged with it, because they are outside X and not present in the assignment which is limited to set X. Also, note that actually the second condition of Definition 4.3 is implied by its first condition: since d_y is the right-most selected disk, all of the disks that appear on the right of d_y in X are surely merged. On the other hand, none of these disks can be merged with a selected disk d_w on the left of d_y, because, in that case d_w would contain the centre of d_y and the assignment cannot be proper.

Theorem 4.4. A proper assignment of the maximum cardinality for a set of n disks D, in which the centres of all disks are collinear, can be computed in polynomial time.

Proof:
Let M be defined as in Definition 4.3. Obviously, $\max_{i=1}^{n} M(n, i, n)$ is the cardinality of the solution to this MCMD instance.

The function M accepts $O(n^3)$ different input values. We can compute and store the values returned by M in a three dimensional table, which we reference also as M. Algorithm 1 uses dynamic programming to fill M, and computes its entries based on the values of previously computed entries. Before explaining the details of the algorithm, we give a high-level overview as follows. The order of the disks referenced here, in the algorithm, and its succeeding discussion is demonstrated in Figure 10.

The main idea behind the algorithm is that in every proper assignment of $\{d_1, d_2, \ldots, d_x\}$, there is at least one selected disk; take the right-most selected disk d_i. By the first condition of Definition 2.3, for some j where $0 \leq j \leq n - 1$, the first j entries of δ_i are merged with d_i. Thus, the algorithm considers different values of j for each disk d_i, to compute the maximum proper assignment of $\{d_1, d_2, \ldots, d_x\}$ for any x in which d_i is the right-most selected disk and j disks are merged with d_i; it updates the entries of M accordingly. The main challenge is to decide what to do with the disks that appear on the left of d_i and use the previously filled entries of M for them. To do so, the algorithm enumerates possible choices for the right-most selected disk d_t that appear on the left of d_i, and the number of disks k merged with it.

After presenting Algorithm 1, we shall explain the steps of this algorithm in more detail.

Steps 1 and 2 of the algorithm initialize M and δ_i. In Step 3, we consider different cases in which d_i, for $1 \leq i \leq n$ in order, is selected and update the value of different entries of M. For every possible value of j from 0 to $n - 1$, suppose j disks are merged with d_i. These disks are the first j disks of δ_i by the first condition of Definition 2.3.

Algorithm 1: Find a solution to MCMD for a set of collinear disks

---

1. Compute the sequences δ_i for $1 \leq i \leq n$ (Definition 2.2).

2. Initialize every entry of M to 0.

3. For each i in $1 \leq i \leq n$, for each j in $0 \leq j \leq n-1$ repeat:

   (a) If the first j disks of δ_i can not be merged with d_i according to the second condition of Definition 2.3, continue with the next iteration of this loop for the next value of i and j.

   (b) Compute A, a, b, and B: a and b are the left-most and right-most disks in λ that are merged with d_i, respectively. Also, A and B are the left-most and right-most disks of D whose centres are contained in d_i, considering its aggregate radius (note that we have $A \leq a \leq b \leq B$).

   (c) If $a = 1$ and $M(b, i, B) == 0$, assign 1 to $M(b, i, B)$.

   (d) If $a > 1$, for t in $1 \leq t \leq A - 1$, for k in $0 \leq k \leq n-2$ repeat:

      i. If the first k disks of δ_t cannot be merged with d_t (according to the second condition of Definition 2.3), continue to the next iteration of this loop.

      ii. Compute indices f and g: d_f is the right-most disk that is merged with d_t, and d_g is the right-most disk of D whose centre is contained in d_t, considering its aggregate radius.

      iii. If $f \geq a$, $f \neq a-1$, or $g \geq i$ (disks merged with d_i are being merged with d_t or d_t contains the centre of d_i, both of which are not allowed in proper assignments), continue to the next iteration of this loop.

      iv. Replace the value of $M(b, i, B)$ with the maximum of its value and $M(a-1, t, g) + 1$.

4. Compute and return $\max_{i=1}^{n} M(n, i, n)$.

Figure 10.   Demonstrating the symbols used in Theorem 4.4. d_i is the right-most selected disk, $\{d_a, \ldots d_b\} \setminus \{d_i\}$ are merged with d_i, and d_i contains the centre of $\{d_A, \ldots, d_B\}$. d_t is the right-most selected disk on the left of d_i.

Let S denote the set of such disks. If this is not possible (the centre of one of these disks is not contained in d_i, after merging its previous disks), we skip this value of j, because it fails the second condition of Definition 2.3 (Step 3a). Note that if there exists no proper assignment in which j disks are merged with d_i, there cannot exists a proper assignment in which more than j disks are merged with d_i either, and we can safely skip the remaining values of j and continue the loop of Step 3 by incrementing the value of i.

Let a, b, A, and B be defined as Step 3b. If $a = 1$, selecting d_i and merging with it every disk in $\{d_1, d_2, \ldots, d_b\} \setminus \{d_i\}$ is a proper assignment of the first b disks of λ with cardinality one. Therefore, we update the value of $M(b, i, B)$ to be at least one in Step 3c.

If $a > 1$, let ϕ be any assignment of $\{d_1, \ldots, d_b\}$, in which i) d_i is selected, ii) the members of S are merged with d_i, and iii) the members of $\{d_A, \ldots, d_{a-1}\} \cup \{d_{b+1}, \ldots, d_B\}$ are contained in d_i after merging the members of S with d_i. By the definition of M, the value of $M(b, i, B)$ cannot be smaller than the cardinality of ϕ. When ϕ is limited to $L = \{d_1, \ldots, d_{a-1}\}$, it specifies a proper assignment of L. We denote this assignment with ϕ_L. We compute the value of $M(b, i, B)$ by considering all possible assignments for ϕ_L and extending them to obtain ϕ by selecting d_i.

Let d_t be the right-most selected disk of ϕ_L. The following conditions hold.

1. We have $t < A$, because $\{d_A, \ldots, d_{a-1}\}$ are contained in d_i in ϕ, and d_t cannot be a selected disk if $t \geq A$. Therefore, disks $\{d_{t+1}, \ldots, d_{a-1}\}$ are merged with d_t in ϕ_L.

2. Suppose k disks are merged with d_t in ϕ_L. Let d_f be the right-most disk of D contained in d_t after merging disks in ϕ_L. We have $f < i$; otherwise, d_f would contain the centre of d_i, and d_i cannot be selected in ϕ. Also let d_g be the right-most vertex of D contained in d_f. We have $g < i$; otherwise, d_f would contain the centre of d_i and ϕ cannot be an extension of ϕ_L.

By trying possible values of t and k that meet these conditions (Step 3d), we find the maximum cardinality of ϕ_L, which has been computed in the previous steps of this algorithm as $M(a - 1, t, g)$. Since ϕ is an extension of ϕ_L by adding exactly one selected disk d_i, the maximum cardinality of ϕ therefore is at least $1 + M(a - 1, t, g)$. Thus, we have

$$M(b, i, B) \geq 1 + \max_{\substack{t: 1 \leq t \leq A-1 \\ k: 0 \leq k \leq n-2 \\ \text{if 3.d.i and 3.d.iii hold}}} M(a - 1, t, g)$$

Step 3(d)iv updates $M(b, i, B)$ to be at least this value. □

Theorem 4.5. The time complexity of computing M for an instance of MCMD with a set of n disks, as described in Theorem 4.4, is $O(n^5)$.

Proof:
We analyse Algorithm 1. Constructing δ_i (Step 1) can be done in $O(n^2 \log n)$ and initializing M (Step 2) can be done in $O(n^3)$. For each pair of values for i and j, Steps 3a-3c can be performed in $O(n)$. In Step 3d, $O(n^2)$ possible cases for t and k are considered, and for each of these cases, the Steps 3(d)i, 3(d)ii, 3(d)iii, and 3(d)iv can be performed in $O(n)$. Since the loop of Step 3 is repeated $O(n^2)$ times, the time complexity of the whole algorithm is $O(n^5)$. □

Note that Algorithm 1 cannot be used for RMCMD (Defintion 3.10) as it relies on the first condition of Definition 2.3: d_j may be merged with d_i, only if disks closer to d_i than d_j are also merged with it. This does not hold for RMCMD.

5.   Concluding remarks

We introduced a variant of geometric independent set for a set of disks, such that the disks that do not appear in the output must be merged with a nearby disk that does (the problem was stated formally in Section 2). We proved that this problem is NP-hard (Theorem 3.6). Also by relaxing one of the conditions of the problem, we introduced a less restricted variant, which was proved NP-hard as well (Theorem 3.12). We presented an ILP for the general case, and a polynomial-time algorithm for the case in which disk centres are collinear.

The centre-disjointness property of the disks in the definition of MCMD and RMCMD implies that we are implicitly assuming Euclidean distance between the centre of the disks; disk d_i containing the centre of d_j implies that the Euclidean distance between p_i and p_j is at most r_i. If the problem is defined using other distance measures, we would have different shapes; for instance squares for L_∞ measure. We can even define the problem using any (maybe irregular) shape, or in higher dimensions using, for instance, spheres. Interestingly, the hardness results of Section 3.6 can be adapted to centre-disjoint squares, after adjusting the gadgets (using squares instead of disks and updating the placement of m-disks slightly). Other shapes (or distance measures) may be studied for hardness results or better algorithms.

Several interesting problems call for further investigation, such as: i) general approximation algorithms, ii) studying the case in which the number of disks that can be merged with a selected disk is bounded by some constant, iii) solving RMCMD for disks with collinear centres, and iv) solving MCMD when disks are pierced by a line (the line may not pass through disk centres).

References

[1] Formann M, Wagner F. A Packing Problem with Applications to Lettering of Maps. In: Symposium on Computational Geometry (SoCG). ACM, North Conway, NH, USA, 1991 pp. 281–288. doi:10.1145/109648.109680.

[2] Håstad J. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 1999. **182**(1):105–142. doi:10.1007/BF02392825.

[3] Fowler RJ, Paterson M, Tanimoto SL. Optimal Packing and Covering in the Plane are NP-Complete. *Information Processing Letters*, 1981. **12**(3):133–137. doi:10.1016/0020-0190(81)90111-3.

[4] Hochbaum DS, Maass W. Approximation Schemes for Covering and Packing Problems in Image Processing and VLSI. *J. ACM*, 1985. **32**(1):130–136. doi:10.1145/2455.214106.

[5] Agarwal PK, van Kreveld MJ, Suri S. Label Placement by Maximum Independent Set in Rectangles. *Comput. Geom.*, 1998. **11**(3–4):209–218. doi:10.1016/S0925-7721(98)00028-5.

[6] Erlebach T, Jansen K, Seidel E. Polynomial-Time Approximation Schemes for Geometric Intersection Graphs. *SIAM J. Comput.*, 2005. **34**(6):1302–1323. doi:10.1137/S0097539702402676.

[7] Chan TM. Polynomial-Time Approximation Schemes for Packing and Piercing Fat Objects. *J. Algorithms*, 2003. **46**(2):178–189. doi:10.1016/S0196-6774(02)00294-8.

[8] Chan TM, Har-Peled S. Approximation Algorithms for Maximum Independent Set of Pseudo-Disks. *Discrete & Comput. Geom.*, 2012. **48**(2):373–392. doi:10.1007/s00454-012-9417-5.

[9] Been K, Daiches E, Yap CK. Dynamic Map Labeling. *IEEE Trans. Vis. Comput. Graph.*, 2006. **12**(5):773–780. doi:10.1109/TVCG.2006.136.

[10] Gemsa A, Nöllenburg M, Rutter I. Consistent Labeling of Rotating Maps. *Comput. Geom.*, 2011. **7**(1):308–331. doi:10.20382/jocg.v7i1a15.

[11] Gemsa A, Nöllenburg M, Rutter I. Evaluation of Labeling Strategies for Rotating Maps. *ACM J. Experim. Algor.*, 2016. **21**(1):1.4:1–1.4:21. doi:10.1145/2851493.

[12] Cano RG, de Souza CC, de Rezende PJ. Fast Optimal Labelings for Rotating Maps. In: Workshop on Algorithms and Computation (WALCOM). Springer, Hsinchu, Taiwan, 2017 pp. 161–173. doi:10.1007/978-3-319-53925-6_13.

[13] Yokosuka Y, Imai K. Polynomial Time Algorithms for Label Size Maximization on Rotating Maps. *J. Inform. Process.*, 2017. **25**:572–579. doi:10.2197/ipsjjip.25.572.

[14] Funke S, Krumpe F, Storandt S. Crushing Disks Efficiently. In: IWOCA. Springer, Helsinki, Finland, 2016 pp. 43–54. doi:10.1007/978-3-319-44543-4_4.

[15] de Berg M, Khosravi A. Optimal Binary Space Partitions in the Plane. In: COCOON. Springer, Nha Trang, Vietnam, 2010 pp. 216–225. doi:10.1007/978-3-642-14031-0_25.

[16] Rudi AG. Maximizing the Number of Visible Labels on a Rotating Map. *Sci. Ann. Comput. Sci.*, 2021. **31**(2):275–287. doi:10.7561/SACS.2021.2.293.

[17] Kaplan H, Klost K, Mulzer W, Roditty L, Seiferth P, Sharir M. Triangles and Girth in Disk Graphs and Transmission Graphs. In: European Symposium on Algorithms (ESA). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Munic/Garching, Germany, 2019 pp. 64:1–64:14. doi:10.4230/LIPICS.ESA.2019.64.

[18] Kaplan H, Mulzer W, Roditty L, Seiferth P. Spanners for Directed Transmission Graphs. *SIAM J. Comput.*, 2018. **47**(4):1585–1609. doi:10.1137/16M1059692.

[19] Klost K, Mulzer W. Recognizing Generalized Transmission Graphs of Line Segments and Circular Sectors. In: LATIN. Springer, Buenos Aires, 2018 pp. 683–696. doi:10.1007/978-3-319-77404-6_50.

[20] Biniaz A, Bose P, Carmi P, Maheshwari A, Munro JI, Smid MHM. Faster Algorithms for some Optimization Problems on Collinear Points. In: Symposium on Computational Geometry (SoCG). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Budapest, Hungary, 2018 pp. 8:1–8:14. doi:10.4230/LIPICS.SOCG.2018.8.

[21] Lichtenstein D. Planar Formulae and Their Uses. *SIAM J. Comput.*, 1982. **11**(2):329–343. doi:10.1137/0211025.

[22] Karp RM. Reducibility Among Combinatorial Problems. In: Complexity of Computer Computations. Plenum Press, New York, USA, 1972 pp. 85–103. doi:10.1007/978-1-4684-2001-2_9.