J Comb Optim
DOI 10.1007/s10878-015-9842-5

Euclidean movement minimization

Nima Anari · MohammadAmin Fazli · Mohammad Ghodsi · MohammadAli Safari

Abstract We consider a class of optimization problems called movement minimization on euclidean plane. Given a set of m nodes on the plane, the aim is to achieve some specific property by minimum movement of the nodes. We consider two specific properties, namely the connectivity (CON) and realization of a given topology (TOPOL). By minimum movement, we mean either the sum of all movements (SUM) or the maximum movement (MAX). We obtain several approximation algorithms and some hardness results for these four problems. We obtain an $O(m)$-factor approximation for CONMAX and CONSUM and extend some known result on graphical grounds and obtain inapproximability results on the geometrical grounds. For the TOPOL problems (where the final decoration of the nodes must correspond to a given configuration), we find it much simpler and provide FPTAS for both MAX and SUM versions.

Keywords Movement minimization · NP-hardness · Approximation algorithms

N. Anari
Computer Science Division, University of California Berkeley, Berkeley, USA
e-mail: anari@berkeley.edu

M. Fazli (✉) · M. Ghodsi · M. Safari
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran
e-mail: fazli@ce.sharif.edu

M. Safari
e-mail: msafari@sharif.edu

M. Ghodsi
Institute of Research in Fundamental Sciences (IPM), Tehran, Iran
e-mail: ghodsi@sharif.edu

Published online: 24 February 2015

1 Introduction

Consider a number of moveable robots distributed over a plane in a far-flung manner. Each robot has an antenna with a limited maximum range, denoted by r_{max}. Robot s can communicate directly with robot t if and only if their distance is less than r_{max}. Robot s can also communicate indirectly with t if there is an ordered set of robots $s = r_1, r_2, \ldots, r_p = t$ so that each r_i can directly communicate with r_{i+1}. With this explanation, we can form a dynamic graph whose vertices are the moveable robots on the plane and edges are formed by connecting each robot to every other robot residing in the disk with radius r_{max} around it. These geometric graphs are called unit disk graphs (UDGs).

Definition 1 Given some points $p_1, \ldots, p_m$ in the euclidean plane, the UDG on these points is defined as a simple graph $G = (V, E)$, where $V = \{1, \ldots, m\}$ and $E = \{\{i, j\} \mid |p_i - p_j|_2 \leq 1\}$

Suppose that robots are initially located at points $p_1, p_2, \ldots, p_m$. It is clear that all robots can communicate directly or indirectly with each other if and only if their corresponding UDG is connected. Our first aim is to have the robots move in a way that they form a connected UDG after relocation (the points $p_1^*, p_2^*, \ldots, p_m^*$). We also want to efficiently optimize the travel distance of the robots before they reach their final locations. The term *efficiently* can be defined in many ways. In this paper, we consider two of such measures: namely Sum and Max. In Sum, the goal is to minimize the sum of the movements of all robots, or formally to minimize $SUM = \sum_{i=1}^{m} |p_i^* - p_i|_2$. This parameter roughly measures the total energy consumed by the robots.

In Max, the goal is to minimize the maximum movement of all robots, i.e. minimizing $MAX = \max_{i \in \{1,\ldots,m\}} |p_i^* - p_i|_2$. This parameter measures the amount of time needed to reach the final locations.

Using these two functions, we define two problems ConMax and ConSum which have applications in wireless networks (see Basagni et al. 2008b, a, 2009).

Definition 2 In ConMax (resp. ConSum) we want to move the robots so as to form a connected UDG and the optimization goal is Max (resp. Sum).

Each of these problems can be considered in both graphical or geometrical settings.

Definition 3 In a graphical setting, robots move on a graph. At first, robots are placed on some vertices of the graph and at each turn, each robot can move to one of the adjacent vertices (each edge is considered to have one unit of length). In geometrical settings, robots are points belonging to a geometrical space ($\mathbb{R}^2$ in this paper) and are free to move in any direction in the space.

In the final part of this paper, we introduce a new kind of movement problems which is more constrained, in some sense, than the previously proposed problems: TopolMax and TopolSum. In these problems, we want the moveable robots to form a given topology. Different topologies with useful properties such as cascading behavior (see e.g. Kleinberg 2007), transitivity (see e.g. Burda et al. 2004), connectivity (see e.g. Philips et al. 1989) and robustness (see e.g. Callaway et al. 2000) may be considered in this scenario in order to empower the communication between the robots.

Definition 4 In problems TopolMax and TopolSum, we are given m initial points $p_1, \ldots, p_m \in \mathbb{R}^2$ and a set of edges $E \subseteq \{\{i, j\} \mid i, j \in \{1, \ldots, m\}\}$. We are supposed to determine m points $p_1^*, \ldots, p_m^* \in \mathbb{R}^2$ in such a way that the UDG defined on $p_1^*, \ldots, p_m^*$ contain all of the edges in E. The objective function we are trying to minimize can be either MAX or SUM which results in two different problems we call TopolMax and TopolSum.

1.1 Other works

Demaine et al. (2009a) first introduced movement problems in graphical settings and extensively studied them. They defined 15 types of movement problems (borrowing from their terminology, from here on we use the words robot and *pebble* interchangeably). They consider five properties: connectivity, directed connectivity, path, independent set and matching and consider three objective functions: maximum movements, total movement and number of pebbles that move. This results in the following 15 problems: ConMax, ConSum, ConNum, DirConMax, DirConSum, DirConNum, PathMax, PathSum, PathNum, IndMax, IndSum, IndNum, IndMax, IndSum, IndNum.

Most of their salient results were proven in the context of graphs. They proposed an $\mathcal{O}\left(\sqrt{\frac{m}{OPT}}\right)$-factor approximation algorithm for ConMax and PathMax (m is the number of pebbles) and proved $\Omega\left(n^{1-\varepsilon}\right)$ inapproximability result for ConSum and DirConMax (n is the number of vertices in the ground graph) in graphical settings. They also gave an $\mathcal{O}(1)$-approximation for IndMax with an additive error of $\mathcal{O}(1)$ in geometrical settings.

Note that all the algorithms presented in Demaine et al. (2009a) are in fact polynomial in n, the number of the nodes in the base graph, which makes them inefficient when $n \gg m$ which is a realistic assumption. Dealing with this, given that the number of mobile agents is typically much smaller than the complexity of the environment, in Demaine et al. (2009b) the authors turned to fixed-parameter tractability. They characterized the boundary between tractable and intractable movement problems in a very general set up and showed that many movement problems of interest have fixed parameter tractable algorithms.

Later Berman et al. (2011) found a constant factor approximation for PathMax and ConMax problems in graphical settings. They also introduced a generalized version of PathMax and proposed and approximation algorithm for it.

1.2 Our results

Our results include algorithms for ConMax, ConSum, TopolMax, TopolSum and an inapproximability result for ConMax.

In Sect. 2.1 we prove $(2 - \frac{\sqrt{2}}{2})$-inapproximability for ConMax in geometric settings which extends the hardness result of Demaine et al. (2009a) about ConMax in graphical settings.

In Sects. 2.2 and 2.3, we give approximation algorithms for ConMax and ConSum on geometrical grounds. We present $O(m)$-factor approximation algorithm for both problems.

Finally, we consider TopolMax and TopolSum problems and prove that there exist FPTAS for them.

Our results are stated in two dimensions, most of them can be easily extended to higher dimensions. In particular all of our approximation algorithms work for higher dimensions too.

2 ConMax and ConSum

2.1 Hardness results

In this section, first, we prove that ConMax is 2-approximable on UDGs (graphical ground) only if $\mathcal{P} = \mathcal{NP}$. Then, with minor modifications, we prove the inapproximability result about ConMax. Our main idea is a proof of Demaine et al. (2009a) for hardness of ConMax problem in graphical settings, but our case is more involved and needs many modifications.

For this, we reduce the hamiltonian cycle problem on 3-regular planar graphs which is known to be $\mathcal{NP}$-hard (Garey et al. 1976). Let us call this problem 3PHP.

We first start with a useful way of embedding planar graphs:

Lemma 1 (Valiant 1981) *A planar graph G with maximum degree 4 can be embedded in the plane using $O(|V|)$ area in such a way that its vertices are at integer coordinates and its edges are drawn so that they are made up of line segments of the form $x = i$ or $y = j$, for integers i and j.*

There is also a polynomial time algorithm to compute such an embedding (Itai et al. 1982).

We are now ready to prove the NP-hardness of the ConMax problem on UDG grounds.

Theorem 1 *There is no polynomial algorithm for* ConMax *on UDG graphical grounds with approximation factor less than* 2 *unless $\mathcal{P} = \mathcal{NP}$*

Proof We prove this by reducing 3PHP. Assume that we have an instance of 3PHP problem; a 3-regular planar graph G in which we want to check for the existence of a hamiltonian path between two specified vertices s and t. See Fig. 1a for an example.

First, we use Lemma 1 to get an embedding of G with integer coordinated vertices and horizontal or vertical edges (Fig. 1b). Then we put a vertex on each integer point which resides on this embedding's edges and name the resulted graph H (Fig. 1c).

We build an embedding H' of H by scaling up all H's vertices' coordinates by 6.0 to make each edge six times longer. The length of every edge $e = (u, v)$ in H' is now a multiple of 6.0. We build another graph G' from H' by taking two following steps:

– We put new vertices on every integer-coordinated point between u and v. So, the edge $e = (u, v)$ in H' is replaced by a path $P^e = u = v_0, v_1, \ldots, v_{6k-1}, v_{6k} = v$

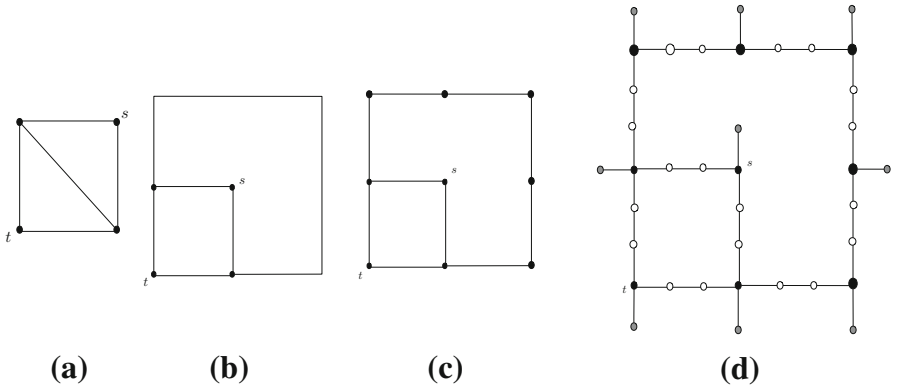

Fig. 1 Graph G and its transformation process

(notice that the distance between v_i and v_{i+1} is exactly one). We color vertices $u = v_0, v_3, v_6, \ldots, v_{3i}, \ldots, v_{6k-3}, v_{6k} = v$ as black and the remaining vertices as white. See the resulting graph G in Fig. 1d (in this figure we have scaled up everything by 3 and not 6 for clarity and better understanding).

– Since the degree of each vertex in the resulting UDG is at most 3, we can attach a new leaf to each black vertex via a unit length vertical or horizontal edge. We color these new leaves as gray.

Finally, we place one pebble on s and t and each gray vertex of G'. We also place two pebbles on each black vertex of G' except s and t. We show that G has a hamiltonian path between s and t if and only if the answer of ConMax on G' is 1. If there is a hamiltonian path between s and t in G, we can move the pebbles on each gray vertex to its neighboring vertex in $V(G')$ and move pebbles on each black vertex to its neighboring vertices along the path corresponding to G's hamiltonian path that induce a connected subgraph in G'. For the reverse side, we show that if G does not have a hamiltonian path between s and t then the value of ConMax is at least 2.0.

We show that when G is not hamiltonian, then establishing connectivity in G' requires a pebble in a gray vertex to move to a white vertex which requires a movement of 2.0.

Consider the optimal connectivity establishment in G'. This induces a connected subgraph of G which is not a hamiltonian path and, therefore, has a maximum degree at least 3. Let u be a vertex with degree 3. It has only 2 pebbles. So, one of its neighboring white vertices, say v, can not be covered by the pebbles on it. If we remove the edge between u and v, we would have a subtree T' in which we need at least 2 moves to connect its pebbles to u's pebbles. This completes the proof. It is clear that nothing would be changed in this proof if we replace general UDGs with their specific type grids because we used only vertical/horizontal edges and integer coordinated vertices. □

We can also use the above proof for the $(2 - \frac{\sqrt{2}}{2})$-inapproximability of ConMax on geometrical grounds.

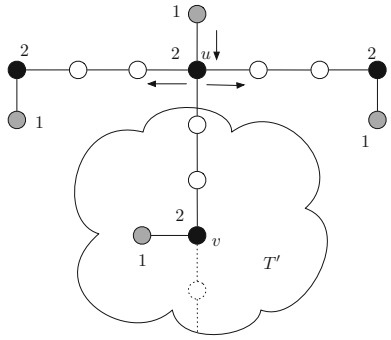

Fig. 2 The 3-degree vertex in minimum maximum degree spanning tree viewed in G'

Theorem 2 *There is no polynomial algorithm for* ConMax *in geometrical settings with an approximation factor of less than* $2 - \frac{\sqrt{2}}{2}$, *unless* $\mathcal{P} = \mathcal{NP}$

Proof The proof structure is almost identical to the proof of Theorem 1. In Fig. 2, the distance between vertex u and vertex v is 3. In the proof of Theorem 2, we had to move pebbles only in integer units of length and the uv path was not covered by the pebbles placed on u. So to connect T''s pebbles to u's pebbles, we had to move them 2 units and the approximation factor was at least 2.

This is different on geometrical ground as u's two pebbles can move to every point of the plane without any limitation. So, there would be a movement of them in which the minimum coverage of these pebbles over all 3 outgoing paths of u is $\frac{\sqrt{2}}{2}$(For example when they move in north-west and south-east direction with 45° slope). So the maximum of minimum coverage over these 3 paths by u's pebbles is at most $\frac{\sqrt{2}}{2}$ and again suppose that this minimum coverage is being happened for uv path. This completes the proof because in this situation the movement of T''s pebbles would be at least $3-1-\frac{\sqrt{2}}{2} = 2-\frac{\sqrt{2}}{2}$ and this leads to the approximation factor $2-\frac{\sqrt{2}}{2}$. □

2.2 $O(m)$ approximation for ConMax

In this section, we propose an $O(m)$ approximation algorithm for ConMax. The proposed algorithm uses a simple geometrical transformation called homothety.

Definition 5 Given a positive real number α and a point $o \in \mathbb{R}^2$, the α-homothety with respect to o is defined as the mapping $H_o^\alpha : \mathbb{R}^2 \to \mathbb{R}^2$ where

$$H_o^\alpha(x) = o + \alpha(x - o)$$

The following two lemmas will help us in the explanation and analysis of the algorithm.

Lemma 2 *An* α*-homothety multiplies all lengths by* α. *In other words* $|H_o^\alpha(x) - H_o^\alpha(y)|_2 = \alpha|x - y|_2$.

Lemma 3 *An α-homothety with respect to the center o, displaces the point x by at most $|1-\alpha| \cdot |x-o|_2$.*

Algorithm 1 $\mathcal{O}(m)$-approximation algorithm for ConMax

Compute the values $e_{ij} = \max(0, (|p_i - p_j| - 1)_2/2)$ for all $i, j \in \{1, \ldots, m\}$.
Put the pairs (i, j) in a non-decreasing order according to the values e_{ij}.
Let the ordered pairs be $(i_1, j_1), \ldots, (i_k, j_k)$.
$G \leftarrow (\{1, \ldots, m\}, \emptyset)$ (the empty graph on $1, \ldots, m$).
for $l \leftarrow 1$ to k **do**
  $G \leftarrow G \cup (i_l, j_l)$
  if G is connected **then**
    $\alpha \leftarrow \frac{1}{1+2e_{ij}}$
    For each $1 \leq r \leq m$ Move the robot r to $p_r^* = H_{p_1}^{\alpha}(p_r)$.
    return
  end if
end for

For a pair of pebbles (i, j), define the excess of the edge (i, j) as the value $e_{ij} = \max(0, (|p_i - p_j|_2 - 1)/2)$. Clearly for pebbles i and j to become directly connected, one of them must at least travel a distance of no less than e_{ij}. This simple observation leads to Algorithm 1.

Now let's prove that Algorithm 1 works and obtains a $\mathcal{O}(m)$-approximation of the optimum solution.

Theorem 3 *There is an $\mathcal{O}(m)$-factor approximation algorithm for* ConMax *on geometrical grounds.*

Proof Algorithm 1 starts with an empty graph on vertices $\{1, \ldots, m\}$ and iteratively adds edges to this graph. In each step the edge with the smallest excess which hasn't been covered yet, is added to the graph.

The algorithm stops as soon as the graph becomes connected. Then a proper homothety is found and it is applied to the robots' initial locations to obtain their final locations. The homothety is centered on one of the initial robot locations and has a coefficient just small enough to shrink the lengths of the edges covered by the algorithm to one.

Since after the application of the homothety, all edges covered by the algorithm become edges of the UDG, the resulting UDG is connected and hence a valid solution to the problem.

Let (i, j) be the last edge added by the algorithm. Clearly using edges with excess smaller than e_{ij}, we can not form a connected graph, because the algorithm didn't stop until it reached (i, j). So the optimum solution has at least one edge with excess greater than or equal to e_{ij}, which proves that $OPT \geq e_{ij}$.

Let p_l be an arbitrary robot. There is a path of at most $m-1$ edges, each having an excess of at most e_{ij}, connecting p_l and p_1. So $|p_l - p_1|_2 \leq (m-1)(1+2e_{ij})$. Using Lemma 3, we have

$$|p_l^* - p_l|_2 \leq |1 - \frac{1}{1+2e_{ij}}| \cdot (m-1) \cdot (1+2e_{ij})$$
$$= 2(m-1)e_{ij} \leq 2(m-1)OPT$$

Hence, the solution produced by the algorithm is at most $2(m-1)$ times worse than the optimum solution. A step-by-step explanation of the algorithm can be presented in Algorithm 1. □

2.3 $O(m)$ approximation for ConSum

The algorithm is slightly more complicated than the one used to approximate ConMax, but the two have similarities.

We will use the following lemma.

Lemma 4 *Any connected UDG has a spanning tree for which the degree of any vertex is at most* 5.

Proof It can be shown that a depth-first search obtains such a tree. Assume that T is a spanning tree generated by a depth-first search on a given UDG and u is a vertex with more than 5 neighboring vertices. Define S as the set of u's neighboring vertices in T. Consider a unit circle C with center u. Since $|S| \geq 6$, C contains at least 7 vertices, so by the pigenhole principle, there exist at least two vertices $x, y \in S$ whose distance is less than 1 and thus they are connected in the UDG. Assume that the depth-first search visited x sooner. Since y is one of x's neighbors in the UDG, the edge xy is discovered sooner than the edge uy and thus y is a neighbor of x in T. So this is a contradiction. □

The algorithm, uses the same notation, excess of the edge, as the one used to describe the $O(m)$-approximation for ConMax. That is $e_{ij} = \max(0, (|p_i - p_j|_2 - 1)/2)$.

The first step of the algorithm is to find a Minimum Spanning Tree (MST) in a complete weighted graph defined on the vertices $\{1, \ldots, m\}$ where the weight of the edge between i and j is the same as e_{ij}.

The observation that leads to this choice for the algorithm is the following.

Lemma 5 *Given a spanning tree T on vertices $\{1, \ldots, m\}$, with maximum vertex-degree Δ, the sum of the distances the robots move needs to be at least the following amount in order to have T as a subgraph of the resulting UDG.*

$$2\frac{\sum_{(i,j)\in T} e_{ij}}{\Delta}$$

Proof Let d_i be the distance robot i travels in an optimal solution. For an edge $(i, j) \in T$, using the triangle inequality we have

$$d_i + d_j \geq 2e_{ij}$$

Adding all equations of the above form, each d_i appears at most Δ times on the left hand side. So we get

$$\Delta \sum_i d_i \geq 2 \sum_{(i,j)\in T} e_{ij}$$

Together with Lemma 4, this shows that OPT is at least two fifths of the weight of the chosen MST.

Now using operations that we call edge-contractions, we can find an admissible solution with the sum of movements at most $O(m)OPT$.

Definition 6 A contraction on edge (i, j) of a tree T can be defined as the following operation.

Removing the edge from the tree gives us two connected components T_1 and T_2. The contraction is defined as translating the vertices of T_1 and T_2 along the line segment $p_i p_j$, each by a distance of e_{ij}. Note that the directions of translations are chosen in a way that p_i and p_j become closer to each other.

Note that after a contraction, the edge (i, j) becomes part of the UDG, and other edges of the tree will remain in the UDG if they were so already.

Another noteworthy point is that a contraction on edge (i, j) contributes at most me_{ij} to the value SUM because each vertex is moved by at most e_{ij}.

Algorithm 2 $O(m)$-approximation algorithm for ConSum

1: Let e_{ij} be defined as $\max(0, (|p_i - p_j|_2 - 1)/2)$.
2: Let G be the complete graph with vertices $\{1, \ldots, m\}$ and edge weights e_{ij}.
3: Compute a Minimum Spanning Tree of G. Name it T.
4: **for** each edge (i, j) in T **do**
5: Let T_1 and T_2 be the components of T that appear after a removal of (i, j). Assume that $i \in T_1$ and $j \in T_2$.
6: Let u_1 be a 2-dimensional vector with the same direction as $p_j - p_i$ but with a magnitude of e_{ij}.
7: Let u_2 be a 2-dimensional vector with the same direction as $p_i - p_j$ but with a magnitude of e_{ij}.
8: Translate all pebbles in T_1 by the vector u_1.
9: Translate all pebbles in T_2 by the vector u_2.
10: **end for**

Now the algorithm is clear (See Algorithm 2). On the MST obtained, we one by one contract all edges. After all contractions are done, we get an admissible solution because the MST becomes a subgraph of the UDG. The parameter SUM is at most

$$\text{SUM} \leq m \sum_{(i,j)\in MST} e_{ij} \leq \frac{2m}{5} OPT$$

which proves that the algorithm is an $O(m)$-approximation.

Theorem 4 *There is an $O(m)$-factor approximation algorithm for* ConSum *on geometrical grounds.*

3 Predetermined topology

Assume that we are given m different points $p_1, \ldots, p_m \in \mathbb{R}^2$. The goal is to make the UDG defined on these points have certain properties. One of the properties that might be desirable for the UDG to have, is to have it contain a certain predetermined topology.

Clearly one can assume that the given topology E is connected; otherwise, the problem can be solved for each connected component separately, and the solutions can be combined together. Hence, from now on we will assume that E is connected.

The main result we obtain is that there is a FPTAS for each one of these problems. Our FPTAS's use the ELLIPSOID method as a blackbox.

Remark 1 The ELLIPSOID method works with a separation oracle defined on a convex set; that is an oracle which when given a point p determines whether it's inside the convex set, and if the answer is false, returns a hyperplane separating the point and the convex set. Given a convex body $C \subset \mathbb{R}^n$ and an initial ellipsoid E_0 containing C, and an arbitrary positive number $\underline{V}$, the ELLIPSOID method either finds a point in C, or finds out that the volume of C is less than $\underline{V}$. The time it takes for the ELLIPSOID method to run is bounded by a polynomial in n and $\log(\text{Vol}(C)/\underline{V})$.

3.1 FPTAS for TOPOLMAX

In this section, we will show how TOPOLMAX can be approximated using the ELLIPSOID method. Our algorithm uses some of the results and tools from the $\mathcal{O}(m)$ approximation algorithm for CONMAX, including the definition and properties of the geometrical transformation homothety.

For two given points p_i and p_j to become at most 1 unit apart (in the Euclidean metric), one should be moved by at least $e_{ij} = \max(0, (|p_i - p_j|_2 - 1)/2)$. Now given an instance of TOPOLMAX define $\underline{O}$ to be $\max_{\{i,j\} \in E} e_{ij}$. Clearly $\underline{O}$ is a lower-bound for OPT. The main idea used behind the proof is exactly the same as the one used in CONMAX, namely the use of homotheties. Let's formulate TOPOLMAX as a linear programing. This linear programing is exact, but unfortunately has infinitely many constraints. The following simple lemma forms the basis of this linear programing.

Lemma 6 *For a vector $v \in \mathbb{R}^2$, the inequality $|v|_2 \leq d$ holds if and only if for each unit vector $u \in S^1$ (S^1 is the unit circle), the inequality $u \cdot v \leq d$ holds.*

Now let's formulate our problem as a non-linear programming, and then convert it to a linear programming. We can define the variables $x_1, \ldots, x_m$ and $y_1, \ldots, y_m$ to be the final coordinates of the points; i.e. $p_i^* = (x_i, y_i)$. Our problem can be formulated like the following

$$\begin{aligned} \text{Minimize} \quad & s \\ \text{Subject To} \quad & |p_i - p_i^*|_2 \leq s, \ \forall i \in \{1, \ldots, m\} \\ & |p_i^* - p_j^*|_2 \leq 1, \ \forall \{i, j\} \in E. \end{aligned}$$

Note that this formulation can be completely written in terms of $x_1, \ldots, x_m$ and $y_1, \ldots, y_m$; we can simply replace each p_i^* by (x_i, y_i). Now applying the previous lemma to this formulation, we can rewrite it like the following

$$\begin{array}{ll} \text{Minimize} & s \\ \text{Subject To} & (p_i^* - p_i) \cdot u \leq s,\ \forall i \in \{1, \ldots, m\}, u \in S^1 \\ & (p_i^* - p_j^*) \cdot u \leq 1,\ \forall \{i, j\} \in E, u \in S^1. \end{array}$$

The new formulation is a linear programming (although, with infinitely many constraints), since inner product is a bilinear operator. To use the ELLIPSOID method on this new formulation, we should first remove s. For each $s \in \mathbb{R}^{\geq 0}$, define L_s to be the convex set in $\mathbb{R}^{2m}$ defined by the constraints

$$\begin{aligned} &(p_i^* - p_i) \cdot u \leq s,\ \forall i \in \{1, \ldots, m\}, u \in S^1 \\ &(p_i^* - p_j^*) \cdot u \leq 1,\ \forall \{i, j\} \in E, u \in S^1. \end{aligned}$$

L_s is the intersection of infinitely many half-planes. Hence, it is convex. The optimum solution of TOPOLMAX is the minimum s for which L_s is nonempty.

Because of the constraints $(p_i^* - p_i) \cdot u \leq s$, we can find a sphere surrounding L_s. This sphere is centered at the point $(p_1, \ldots, p_m) \in \mathbb{R}^{2m}$, and its radius is $\sqrt{m}s$. That is because

$$|(p_1^*, \ldots, p_n^*) - (p_1, \ldots, p_m)|_2 = \sqrt{\sum_{i \in \{1,\ldots,n\}} |p_i^* - p_i|_2^2} \leq \sqrt{ms^2} = \sqrt{m}s.$$

Since this sphere can be surrounded by a hypercube with a side length of $2\sqrt{m}s$, the volume of this sphere is at most $(2\sqrt{m}s)^{2m}$.

Note that using the previous lemma, existence of a separation oracle for L_s becomes obvious. In fact, we just have to check the unit vectors u which are parallel to the vectors $(p_i^* - p_i)$ and the vectors $(p_i^* - p_j^*)$.

We have all of the things we need for the ELLIPSOID method, except V, the lower-bound on the volume of L_s. Note that $L_s \subseteq L_t$ for $s \leq t$. So if we obtain a lower-bound on the volume of L_s for one s, that lower-bound also works for every L_t for which $t \geq s$. Let OPT denote the optimum solution of TOPOLMAX. Let $s^* = (1 + \delta)OPT$. Our goal is to derive a lower-bound on the volume of L_{s^*}.

Lemma 7 *The volume of L_{s^*} is greater than or equal to $(\frac{\delta OPT}{2m})^{2m}$.*

Proof Since L_{OPT} is nonempty, one can find a point $(q_1, \ldots, q_m) \in L_{OPT} \subseteq L_{s^*} \subset \mathbb{R}^{2m}$.

Let H^α denote the α-homothety with respect to q_1. Consider the points $H^\alpha(q_1), \ldots, H^\alpha(q_m)$. Since each q_i can be reached from q_1 by a path consisting only of the edges in E, we have $|q_i - q_1|_2 \leq m - 1$. So

$$|H^\alpha(q_i) - q_i|_2 = (1 - \alpha)|q_i - q_1|_2 \leq (1 - \alpha)(n - 1).$$

Since $|q_i - p_i| \leq OPT$, we have $|H^\alpha(q_i) - p_i| \leq OPT + (1 - \alpha)(m - 1)$.

Because of the properties of homotheties, for each $\{i, j\} \in E$, we have $|H^\alpha(q_i) - H^\alpha(q_j)|_2 \leq \alpha$. Now let $r_1, \ldots, r_m$ be some arbitrary points for which we have $|r_i - H^\alpha(q_i)|_2 \leq (1-\alpha)/2$. For each $\{i, j\} \in E$, we have

$$\begin{aligned} |r_i - r_j|_2 &\leq |H^\alpha(q_i) - H^\alpha(q_j)|_2 + |r_i - H^\alpha(q_i)|_2 + |r_j - H^\alpha(q_j)|_2 \\ &\leq \alpha + \frac{1-\alpha}{2} + \frac{1-\alpha}{2} = 1. \end{aligned}$$

We also have

$$\begin{aligned} |r_i - p_i| &\leq |r_i - H^\alpha(q_i)| + |H^\alpha(q_i) - p_i| \\ &\leq \frac{1-\alpha}{2} + OPT + (1-\alpha)(n-1) \\ &\leq OPT + (1-\alpha)m. \end{aligned}$$

This shows that there is a copy of $\underbrace{B_{(1-\alpha)/2} \times \cdots \times B_{(1-\alpha)/2}}_{m}$ inside $L_{OPT+(1-\alpha)m}$, where B_x shows a 2-dimensional ball of radius x. Since $\mathrm{Vol}(B_x) = \pi x^2 \geq x^2$, we have

$$\mathrm{Vol}(L_{OPT+(1-\alpha)m}) \geq \left(\frac{1-\alpha}{2}\right)^{2m}.$$

We want α to be chosen in such a way that $OPT + (1-\alpha)m \leq s^*$. This can be obtained by setting $\alpha = 1 - (s^* - OPT)/m$. For this α, we have $1 - \alpha = (s^* - OPT)/m = \delta OPT/m$. Therefore

$$\mathrm{Vol}(L_{s^*}) \geq \left(\frac{\delta OPT}{2m}\right)^{2m}.$$

□

Using the lower-bound $(\frac{\delta OPT}{2m})^{2m}$ as the parameter $\underline{\mathrm{V}}$ of Ellipsoid, one can see that the ellipsoid method is able to find a point inside L_{s^*} in time bounded by a polynomial of m and

$$\begin{aligned} \log \frac{(2\sqrt{m}(1+\delta)OPT)^{2m}}{\left(\frac{\delta OPT}{2m}\right)^{2m}} &= \log \left(4m\sqrt{m}\frac{1+\delta}{\delta}\right)^{2m} \\ &= 2m \log \left(4m\sqrt{n}\frac{1+\delta}{\delta}\right) = \mathcal{O}\left(\mathrm{poly}(m, 1/\delta)\right). \end{aligned}$$

We don't know OPT, so we can't actually set the parameter $\underline{\mathrm{V}}$ of Ellipsoid to the above lower bound; this is not a problem, as we can just run the Ellipsoid method for the time bound we have obtained (which depends only on m and δ).

Using the previous lemmas it's easy to see that Algorithm 3 is a $(1+\epsilon)$-approximation for TopolMax. If OPT resides in an interval $[(1+\delta)^i \underline{\mathrm{O}}, (1+\delta)^{i+1} \underline{\mathrm{O}}]$,

then $\overline{s} = (1+\delta)^{i+2}\underline{O}$ is definitely larger than $s^* = (1+\delta)O$. Hence, ELLIPSOID finds a point of $L_{\overline{s}}$ in the time limit given. But we have the following inequality (we're assuming without loss of generality that $\epsilon \leq 1$)

$$\begin{aligned}\overline{s} &\leq (1+\delta)^2(1+\delta)^i\underline{O} \leq (1+\delta)^2 OPT \\ &= (1+2\delta+\delta^2)OPT \\ &\leq (1+2\delta+\delta)OPT = (1+\epsilon)OPT.\end{aligned}$$

So the solution found by Algorithm 3 is a $(1+\epsilon)$-approximation.

Algorithm 3 TOPOLMAX

1: Calculate $\underline{O}$ using the formula $\max\{(|p_i - p_j|_2 - 1)/2 \mid \{i, j\} \in E\}$.
2: Let $\delta = \epsilon/3$. Divide the interval $[\underline{O}, 2(m-1)\underline{O}]$ into $\mathcal{O}(\log m/\log(1+\epsilon))$ intervals of the form $[(1+\delta)^i\underline{O}, (1+\delta)^{i+1}\underline{O}]$. Sort the interval endpoints in an increasing order.
3: **for** each interval endpoint like a **do**
4: Run the ELLIPSOID method on L_a using the initial bounding sphere of radius $\sqrt{m}a$ around the origin. Run this method until it finds an answer or the upper-bound on the execution time we found earlier passes.
5: If the ELLIPSOID method finds a solution point, then stop the algorithm and return that solution.
6: **end for**
7: If no solution is found, return the solution found from our previous $\mathcal{O}(m)$-approximation algorithm.

Theorem 5 *There is a* FPTAS *for the problem* TOPOLMAX.

3.2 TOPOLSUM

The same method used in the previous section can be slightly modified to work for TOPOLSUM. One can again find similar bounds on the volume of the convex body and again show that the ELLIPSOID method works in polynomial time.

Theorem 6 *There is a* FPTAS *for the problems* TOPOLSUM.

4 Concluding remarks

In this paper, the hardness of the CONMAX problem in geometrical settings is proved and then $O(m)$-factor approximation algorithms for each of the CONMAX and CONSUM problems are proposed. Thereafter it is shown that a FPTAS for each of the problems TOPOLMAX and TOPOLSUM exists once the target UDG is known, i.e. adjacent vertices are specified.

Considering other types of properties such as obtaining an independent set of a given size or considering a bigger class of graphs like disc graphs are good research directions to follow. We conjecture that both CONMAX and CONSUM are approximable within constant factors. More than that, as an open problem we conjecture that TOPOLSUM and TOPOLMAX are both NP-hard problems.

References

Basagni S, Carosi A, Melachrinoudis E, Petrioli C, Wang ZM (2008a) Controlled sink mobility for prolonging wireless sensor networks lifetime. Wirel Netw 14(6):831–858

Basagni S, Carosi A, Petrioli C, Phillips CA (2008b) Moving multiple sinks through wireless sensor networks for lifetime maximization. In: MASS, pp. 523–526

Basagni S, Carosi A, Petrioli C (2009) Heuristics for lifetime maximization in wireless sensor networks with multiple mobile sinks. In: ICC'09. IEEE International Conference on Communications, 2009, pp. 1–6

Berman P, Demaine ED, Zadimoghaddam M (2011) O (1)-approximations for maximum movement problems. In: Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. Springer, Berlin, pp. 62–74

Burda Z, Jurkiewicz J, Krzywicki A (2004) Network transitivity and matrix models. Phys Rev E 69(2):026106

Callaway DS, Newman MEJ, Strogatz SH, Watts DJ (2000) Network robustness and fragility: percolation on random graphs. Phys Rev Lett 85(25):5468

Demaine ED, Hajiaghayi M, Mahini H, Sayedi-Roshkhar AS, Oveisgharan Shayan, Zadimoghaddam Morteza (2009a) Minimizing movement. ACM Trans Algorithms (TALG) 5(3):30

Demaine ED, Hajiaghayi M, Marx D (2009b) Minimizing movement: fixed-parameter tractability. In: Algorithms-ESA 2009. Springer, pp. 718–729

Garey MR, Johnson DS, Tarjan RE (1976) The planar hamiltonian circuit problem is np-complete. SIAM J Comput 5(4):704–714

Itai A, Papadimitriou CH, Szwarcfiter JL (1982) Hamilton paths in grid graphs. SIAM J Comput 11(4):676–686

Kleinberg J (2007) Cascading behavior in networks: algorithmic and economic issues. Algorithmic Game Theory 24:613–632

Philips TK, Panwar Shivendra S, Tantawi AN (1989) Connectivity properties of a packet radio network model. IEEE Trans Inf Theory 35(5):1044–1047

Valiant LG (1981) Universality considerations in vlsi circuits. IEEE Trans Comput 100(2):135–140