Minimizing Movement

ERIK D. DEMAINE
MIT
MOHAMMADTAGHI HAJIAGHAYI
AT&T Labs — Research
HAMID MAHINI
AMIN S. SAYEDI-ROSHKHAR
SHAYAN OVEISGHARAN
and
MORTEZA ZADIMOGHADDAM
Sharif University of Technology

Abstract. We give approximation algorithms and inapproximability results for a class of movement problems. In general, these problems involve planning the coordinated motion of a large collection of objects (representing anything from a robot swarm or firefighter team to map labels or network messages) to achieve a global property of the network while minimizing the maximum or average movement. In particular, we consider the goals of achieving connectivity (undirected and directed), achieving connectivity between a given pair of vertices, achieving independence (a dispersion problem), and achieving a perfect matching (with applications to multicasting). This general family of movement problems encompass an intriguing range of graph and geometric algorithms, with several real-world applications and a surprising range of approximability. In some cases, we obtain tight approximation and inapproximability results using direct techniques (without use of PCP), assuming just that $P \neq NP$.

Categories and Subject Descriptors: F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Motion planning, pebble placement, graphs, Euclidean plane

---

A preliminary version of this article appeared in *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 2007, pages 258–267. This work was done while M. Hajiaghayi was at MIT and CMU. E. Demaine and M. Hajiaghayi were supported in part by NSF under grant number ITR ANI-0205445. M. Hajiaghayi was supported in part by Institute for Theoretical Physics and Mathematics (IPM) under grant number CS1384-6-01. H. Mahini was supported in part by Institute for Theoretical Physics and Mathematics (IPM) under grant number CS1384-2-01.
Author's addresses: E. Demaine, Computer Science and Artificial Intelligence Laboratory, MIT, 32 Vassar St., Cambridge, MA 02139, USA, e-mail: edemaine@mit.edu; M. Hajiaghayi, AT&T Labs — Research, 180 Park Ave., Florham Park, NJ 07932, USA, e-mail: hajiagha@research.att.com; H. Mahini, A. Sayedi-Roshkhar, S. Oveisgharan, M. Zadimoghaddam, Department of Computer Engineering, Sharif University of Technology, Azadi St., Tehran, Iran, e-mail: {mahini,sayedi,oveisgharan,zadimoghaddam}@ce.sharif.edu.

1 *Introduction*

Consider a group of firefighters surrounding a forest fire. Each firefighter is equipped with a reliable but short-range radio (walkie-talkie) as well as limited connectivity to a satellite (or other central location) for triangulating and sharing the approximate positions of firefighters. To form an effective communication network (for voice or data traffic), the firefighters' radios must form a connected graph. This scenario naturally leads to the following problem: given the current locations of the firefighters, find the minimum distance (time) required for each firefighter to move to reach a configuration that induces a connected radio network. More precisely, we wish to minimize the maximum movement of the firefighters such that, in their final positions, any two firefighters can talk to each other in the reliable radio network, possibly using multiple hops.

Of course, this playful description of the problem is rhetorical: in reality, the objects are not firefighters but are, say, autonomous robots with limited wireless connectivity and limited mobility in the field because of energy and resource constraints, so they wish to minimize the use of these resources to form a reliable radio network. See, for example, Corke et al. [2004a; 2004b] and Bredin et al. [2005] for descriptions of such practical scenarios.

The problem described above is one example of a natural broader family of problems, called *movement problems*, which we study systematically in this article. In particular, the firefighting problem can be abstracted into a problem called ConMax: minimize maximum movement to reach connectivity. This basic connectivity problem has many variations. For example, ConSum asks to minimize the total movement, which may be useful for reducing average power consumption; while ConNum asks to minimize the number of firefighters (robots) that have to move. In DirConMax, and analogously DirConSum and DirConNum, the radio connectivity is not necessarily symmetric and forms a directed network, for example, because different radios have different power levels, and the goal is to ensure that everyone can receive messages from a fixed root (the captain). In PathMax, PathSum, and PathNum, the goal is to re-arrange the objects to connect two specified locations.[1]

Many more variations arise from changing the desired property of the final configuration. In general, for a specified property P of configurations of objects, the goal of a movement problem is to minimize the (maximum or total/average) movement in a motion that ends with a configuration satisfying property P. The objects can be represented either as points or equivalently bodies that can only be translated, say in the plane, or as pebbles placed on the vertices of a graph that can move along edges. Many problems in this family arise naturally in the context of robotics, particularly in organizing the behavior of swarms of robots (see, e.g., Hsiang et al. [2003], LaValle [2006], Reif and Wang [1995], Schultz et al. [2003]).

A simple version of the movement problem is *collocation*, where the goal is to move all objects to a common location. In this case, we obtain two classic problems in graph algorithms: when minimizing the maximum movement, we have the 1-center problem; when minimizing the total movement, we have the 1-median

[1]For example, the firefighters might want to chain together their water hoses from a fire hydrant to the fire.

problem. These problems have well-known polynomial-time exact solutions. In the geometric plane, minimizing maximum movement becomes smallest enclosing disk, which is also polynomial, and minimizing total movement becomes the Weber problem, which can be solved up to error ε in time polynomial in n in $\lg(1/\varepsilon)$.

Another interesting version of the movement problem is *dispersion*, where the goal is to distribute the objects in order to guarantee a minimum pairwise separation between the objects. In the context of a radio network, this goal is equivalent to guaranteeing that the radio network forms an empty graph or an independent set. Thus, we refer to this problem as IndMax, IndSum, or IndNum, according to the objective function. This problem effectively asks to spread out the objects (e.g., robots) while keeping them as close as possible to their original locations. The problem also has applications to map labeling [Doddi et al. 1997; Jiang et al. 2004; Strijk and Wolff 2001; Jiang et al. 2003], where the goal is to find placements of labels as close as possible to the specified features of the map such that the labels do not overlap each other (so their centers are sufficiently separated).

Another version of the movement problem that arises in the context of broadcasting or multicasting is to move the objects into nearby pairs so that these pairs can exchange information. More precisely, in MatchMax, MatchSum, and MatchNum, the goal is to minimize the movement of the objects to a position having a perfect matching of the objects such that each matched pair can communicate (i.e., the objects are within distance 1 of each other). This problem is essentially a mobile version of the pseudo-matching problem (also known as path-matching) considered in the context of broadcasting and multicasting in cut-through routed networks [Cohen et al. 1998; Cohen 1998; Ghodsi et al. 2002]. The MatchMax problem is also closely related to one "round" of the freeze-tag problem [Arkin et al. 2002; Arkin et al. 2003; Sztainberg et al. 2004] in which a swarm of mobile robots must collectively "wake up", starting from a single awake robot, and moving awake robots next to sleeping robots to awaken them.

Several of the problems considered in this article can be viewed as considering the extent to which we exploit the mobility of existing resources to achieve desired global properties of the network such as connectivity. Related to this endeavor is work that considers how to augment networks (consisting of nonmobile sensors) by adding additional resources to achieve such global properties; see, for example, Bredin et al. [2005] and Corke et al. [2004a; 2004b]. In fact, we can view the class of movement problems as strictly more general than these augmentation problems, by imagining additional mobile resources initially "at infinity" and the goal is to minimize the total movement of these resources (and therefore minimize the number of resources moved).

1.1 Motion Problems and Model.

Before we describe our specific results, we formally define the model and the movement problems we consider.

The three general families of problems we consider are *minimum maximum movement to property* P, *minimum total movement to property* P, and *minimum number of movements to property* P. In all cases, we are given an (undirected or directed) graph $G = (V, E)$ with $|V| = n$ vertices, m *pebbles*, and a property P on "configurations". A *configuration* is a function assigning each pebble to a vertex of V; more than one pebble can be on a single vertex. We say that each such assigned vertex is *occupied* by a pebble. We are given an *initial configuration* for the pebbles. A *motion* assigns a path $\pi(p)$ in the graph G for each pebble p, starting at the vertex specified by the initial configuration and ending at some *target vertex*, also called the *target position*. (Thus, pebbles can move only along edges.) The length $|\pi(p)|$ of the path is the *movement* of p. The *maximum movement* of a motion is the maximum length of any path; the *total movement* is the total length of all paths; and the *number of movements* is the number of paths of nonzero length. The target vertices of pebbles define the *target configuration* of the motion. The goal is to find a motion that minimizes one of these three measures subject to the target configuration satisfying property P.

This graph-theoretic formulation of the movement problems also captures the geometric setting. For example, the *Euclidean plane* is defined by an infinite graph whose vertices correspond to points $p = (p_x, p_y)$, and edges connect two distinct vertices p and q whose Euclidean distance $d(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$ is less than 1. This definition models mobile nodes with unit communication radius. Because the graph is infinite, there is no notion of "n", so we define $n = m$, the number of pebbles.

We define the following properties P of interest and their associated problems of minimizing maximum movement. In most cases, we state a property P on graphs, implicitly referring to the subgraph of G induced by the vertices occupied by pebbles in the configuration.

(1) Minimum maximum movement to connectivity (ConMax): P is connectivity.

(2) Minimum maximum movement to connectivity in directed graphs (DirConMax): P is directed connectivity from every vertex to some root vertex.

(3) Minimum maximum movement to s-t connectivity (PathMax): P is having a path between two certain vertices s and t.

(4) Minimum maximum movement to independence (IndMax): P is that no two pebbles occupy the same or adjacent vertices.

(5) Minimum maximum movement to perfect matchability (MatchMax): P is the property that there is a perfect matching in the graph on pebbles in which two pebbles p and q are adjacent precisely if their distance $d_G(p, q)$ in G is at most 1.

Analogously, we define the problems of minimizing total movement (ConSum, DirConSum, IndSum, PathSum, and MatchSum) and minimizing the number of movements (ConNum, DirConNum, IndNum, PathNum, and MatchNum) to achieve the same properties. To our knowledge, none of these problems have been considered before in an algorithmic setting.

1.2 Our Results.

We prove several approximation and inapproximability results for the problems listed above, in many cases obtaining tight bounds (assuming just P $\neq$ NP). The various movement problems show a surprising range of difficulty, not consistent with the nonmovement (standard) version of each problem. For example, testing connectivity of a graph is trivial, but DirConMax and ConSum are $\Omega(n^{1-\varepsilon})$-inapproximable, while the best approximation so far for ConMax is $O(1 + \sqrt{m/\text{OPT}})$ in a graph or in the Euclidean plane, and we give evidence

	Max		Sum		Num	
Con	$O(1+\sqrt{m/\mathrm{OPT}})$	(§2.1)	$O(\min\{n \log n, m\})$ $\Omega(n^{1-\varepsilon})$	(§5.2) (§5.1)	$O(m^{\varepsilon})$ $\Omega(\log n)$	(§6.1) (§6.1)
Path	$O(1+\sqrt{m/\mathrm{OPT}})$	(§2.3)	$O(n)$	(§25)	polynomial	(§6.2)
DirCon	εm $\Omega(n^{1-\varepsilon})$	(§2.4) (§2.5)	open		$O(m^{\varepsilon})$ $\Omega(\log^2 n)$	§6.1) (§6.1)
Ind	$1+\frac{1}{\sqrt{3}}$ additive in $\mathbb{R}^2$	(§3)	open		PTAS in $\mathbb{R}^2$	(§6.3)
Match	polynomial	(§4)	polynomial	(§5.4)	polynomial	(§6.4)

Table I. A summary of our results.

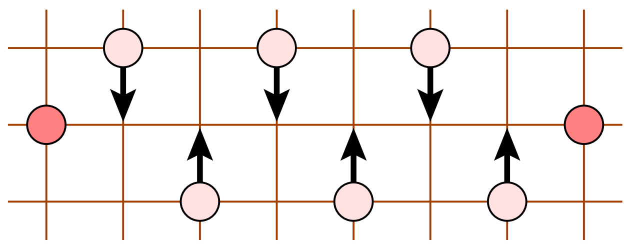

Fig. 1. Optimally moving the pebbles (drawn as disks) into a connected configuration requires a global solution (drawn with arrows).

that even the geometric scenario is difficult. On the other hand, we give an additive $O(1)$-approximation for IndMax in the Euclidean plane, even though the nonmovement version (independent set) is very hard for graphs and not known to be solvable exactly in the Euclidean plane. Yet some movement problems such as MatchMax turn out to have polynomial-time solutions. Our hardness results are particularly strong, yet they do not use techniques such as PCP and thus avoid any higher-level complexity assumptions, making them of independent interest.

We focus primarily on the maximum-movement problems, proving various approximability and inapproximability results in Sections 2, 3, and 4. We then consider the total-movement versions of the problems in Section 5. Section 6 considers the number-of-movements versions. Table I summarizes all of our results.

2 *Minimum Maximum Movement to Connectivity*

We begin with the problem of ConMax, a well-motivated problem as described in the Introduction. To provide some intuition about the problem, Figure 1 gives an example of a challenging instance. Here there is a "global" solution using maximum movement of 1, but any "local" solution (such as all pebbles approaching a common location) requires maximum movement of $\Omega(n)$.

It is also not hard to see that the problem is NP-complete in general, even to approximate better than a factor of 2. We can reduce from the Hamiltonian path problem as follows. Given a graph $G = (V, E)$, we subdivide each edge in E into a path of three edges, and attach a new leaf vertex to each vertex in V. We place two pebbles on each vertex in V and we place one pebble on each added leaf. Any solution to this instance of ConMax of maximum movement 1 can move the pebble on each leaf to its neighboring vertex in V, and must move the two pebbles on each vertex in V toward neighboring vertices to induce a connected subgraph. Such a solution corresponds to a connected maximum-degree-2 subgraph in G that visits every vertex in V, that is, a Hamiltonian path in G.

2.1 $O(1+\sqrt{m/\mathrm{OPT}})$-Approximation for ConMax.

In this section we develop an $O(1+\sqrt{m/\mathrm{OPT}})$-approximation algorithm for ConMax, where m is the number of pebbles. (Note that m can be much smaller than n.) In particular, this algorithm is an $O(\sqrt{n})$-approximation algorithm if the initial configuration places at most one pebble on each vertex. We later show in Section 2.2 how to convert this approximation algorithm, or indeed any approximation algorithm for ConMax, to work in the Euclidean plane at a small extra cost.

THEOREM 1. *There is an $O(1+\sqrt{m/\mathrm{OPT}})$-approximation algorithm (and thus also an $O(\sqrt{m})$-approximation algorithm) for ConMax.*

Given a subset S of vertices in a graph G, the *dth power induced on S*, denoted by $G^d[S]$, has vertex set S and has an edge (u,v) between two vertices $u,v \in S$ if and only if there is a path in G between u and v with at most d edges.

LEMMA 2. *Consider an instance of the ConMax problem, consisting of a graph G and an initial configuration of m pebbles, with an optimal solution of maximum movement* OPT. *For any integer k between 0 and $m/2$, there is a subset S of vertices of G satisfying the following properties:*

(1) *Every vertex in S is occupied by a pebble in the initial configuration.*
(2) *The shortest-path distance between any two distinct vertices in S is greater than* $2k+4\mathrm{OPT}$.
(3) *The* $(2k+6\mathrm{OPT}+1)$*th power of G induced on S is connected.*
(4) *Every vertex v in S has at least $2k$ pebbles whose shortest-path distance to v is at most* $k+2\mathrm{OPT}$.
(5) *For every vertex w occupied by a pebble in the initial configuration, there is a vertex u in S whose shortest-path distance to w is at most* $3k+8\mathrm{OPT}+1$.

PROOF. We compute S via a greedy algorithm. Initially, S is the empty set, which satisfies Properties 1–4. In each step, if there is a vertex whose addition to S would still satisfy Properties 1–4, we add the vertex to S.

First we prove that the greedy algorithm computes a nonempty set S, that is, at the first step, there is a vertex we can add to S. Let T be a spanning tree of the (connected) graph induced by the target configuration in the optimal solution OPT. Define a *center* c of T to be a vertex of T that minimizes the maximum distance from c to any vertex of T. We claim that c is within distance k of at least $2k$ target positions of pebbles, and thus the initial position u of any pebble whose final position is c is within distance $k+2\mathrm{OPT}$ of at least $2k$ initial positions of pebbles, and therefore $S=\{u\}$ satisfies Properties 1–4. The proof of this claim divides into two cases. In Case 1, every vertex of T is within distance k of c, and thus the target positions of all m pebbles are within distance k of c, proving the claim. In Case 2, there is at least one vertex at distance exactly $k+1$ from c. In this case, we claim by induction on k that there are at least $2k$ vertices of T within distance k of c. Note that we remain in Case 2 even when considering smaller values of k. In the

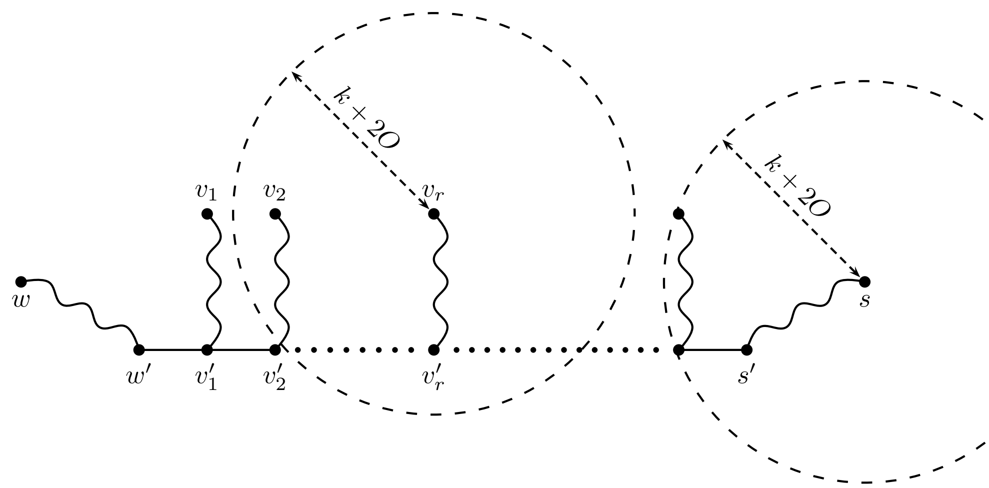

Fig. 2. Path between w and s.

base case, $k = 0$ and the claim is vacuous. In the general case $k > 0$, by induction, there are at least $2k - 2$ vertices of T within distance $k - 1$ of c. Because we are in Case 2, there is at least one vertex v at distance exactly k from c, and at least one vertex w at distance exactly $k+1$ from c. If there are at least two vertices at distance exactly k from c, then we have the claim. If vertex v is the only vertex at distance exactly k from c, then we argue that c cannot be a center. Moving c one step toward v decreases the distance from c to v, w, and any vertices of T with distance at least k, in particular decreasing w's distance of $k+1$, while the distance from c to all other vertices (which have distance at most $k-1$) increases by at most 1 and so the distance remains at most k. Therefore, this move decreases the maximum distance from c to any vertex of T, contracting centrality of c.

Now consider the maximal set S output by the greedy algorithm, and suppose for contradiction that some vertex w is not within distance $3k+8\text{OPT}+1$ of its nearest vertex s in S. (If there is more than one such vertex s, we choose one arbitrarily.) For any vertex v, let ND_v denote the distance between v and its nearest vertex in S. Thus, $ND_w > 3k + 8\text{OPT} + 1$. Let w' and s' denote the target positions for some pebble initially on vertex w and for some pebble initially on vertex s, respectively, in the optimal solution OPT. Refer to Figure 2. Let $P = \langle w' = v'_0, v'_1, v'_2, \ldots, v'_j = s' \rangle$ be a path between w' and s' in the (connected) graph induced by the target positions in OPT. Let v_i denote the initial position of some pebble whose target position is v'_i in OPT, making choices so that $v_0 = w$ and $v_j = s$. Thus, the distance between v_i and v'_i is at most OPT. Because v'_i and v'_{i+1} are adjacent in G, the distance between v_i and v_{i+1} is at most $2\text{OPT} + 1$. By the triangle inequality, $ND_{v_i} \leq ND_{v_{i+1}} + 2\text{OPT} + 1$. Because $ND_{v_0} = ND_w > 3k + 8\text{OPT} + 1$, because $ND_{v_j} = ND_s = 0$, and because ND_{v_i} decreases by at most $2\text{OPT} + 1$ each time we increment i, there is an index r such that $2k+4\text{OPT} < ND_{v_r} \leq 2k+6\text{OPT}+1$.

We claim that we can add v_r to S while satisfying Properties 1–4, contradicting the maximal choice of S. By our choice of v_r, we satisfy Properties 1–3. By the triangle inequality, the distance between w and v_r is at least $ND_w - ND_{v_r} > (3k + 8\mathrm{OPT} + 1) - (2k + 6\mathrm{OPT} + 1) = k + 2\mathrm{OPT}$. Thus, the distance between w' and v'_r, namely r, is at least $k + 2\mathrm{OPT} - 2\mathrm{OPT} = k$. Similarly, by the triangle inequality, the distance between v_r and s is at least $ND_{v_r} - ND_s > 2k + 4\mathrm{OPT}$. Thus, the distance between v'_r and s', namely $j-r$, is at least $2k+4\mathrm{OPT}-2\mathrm{OPT} = 2k+2\mathrm{OPT} > k$. Therefore, $v'_{r-k}, v'_{r-k+1}, \ldots, v'_r, \ldots, v'_{r+k-1}, v'_{r+k}$ are $2k+2$ vertices along the path P. The corresponding vertices $v_{r-k}, v_{r-k+1}, \ldots, v_r, \ldots, v_{r+k-1}, v_{r+k}$ are occupied by pebbles and have distance at most $k + 2\mathrm{OPT}$ from v_r. Hence, we satisfy Property 4. □

Lemma 3. *Given an instance of ConMax problem with m pebbles and with an optimal solution of maximum movement* OPT*, for any integer k between* 0 *and* $m/2$*, there is a polynomial-time algorithm to find a motion with maximum movement at most* $5k + 14\mathrm{OPT} + 2 + (6\mathrm{OPT} + 1)m/(2k)$.

Proof. The algorithm proceeds as follows.

(1) Find a subset S of vertices of G with the properties of Lemma 2.

(2) Move each pebble to its nearest vertex in S.

(3) Let H be the $(2k + 6\mathrm{OPT} + 1)$th power of G induced on S. By Property 3, H is connected, so let T be a spanning tree of H, and root it at an arbitrary vertex.

(4) For each vertex v in S other than the root, move all but one of the pebbles on v to occupy some of the vertices on the path in G corresponding to the edge between v and its parent in the tree T.

(5) Let T' be the tree in G obtained by combining the paths corresponding to the edges of T. For every vertex of T' that is unoccupied by a pebble, move all pebbles one step in T' toward that vertex.

By Property 5, Step 2 moves each pebble at most $3k + 8\mathrm{OPT} + 1$ steps. By Properties 2 and 4 of Lemma 2, for every vertex s in S, there are at least $2k$ pebbles that are closer to s than to any other vertex in S. Thus, after Step 2 of the algorithm, every vertex s in S is occupied by at least $2k + 1$ pebbles. By Property 3, Step 4 moves each pebble at most $2k+6\mathrm{OPT}+1$ steps. After Step 4, at most $6\mathrm{OPT} + 1$ vertices of each path corresponding to an edge of T lack a pebble. Thus the tree T' in G has at most $|S|(6\mathrm{OPT}+1)$ vertices that lack a pebble. Each iteration of the loop in Step 5 removes at least one of these vertices at a cost of 1. Thus Step 5 moves each pebble by at most $|S|(6\mathrm{OPT} + 1)$ steps. But $|S|$ is at most $m/(2k)$, because we assign at least $2k$ pebbles to each vertex in S and the total number of pebbles is m. Therefore the total cost is $(3k + 8\mathrm{OPT} + 1) + (2k + 6\mathrm{OPT} + 1) + (6\mathrm{OPT} + 1)m/(2k) = 5k + 14\mathrm{OPT} + 2 + (6\mathrm{OPT} + 1)m/(2k)$, proving the lemma. □

To prove Theorem 1, we first check whether OPT is zero, that is, whether the pebbles already induce a connected graph. Otherwise, we apply Lemma 3 with $k = \sqrt{m\,x}$ where x is a guessed value of OPT. With $k = \sqrt{m\,\mathrm{OPT}}$ (or with the best guess of x), we obtain an approximation ratio of $O(1 + \sqrt{m/\mathrm{OPT}})$.

Finally, it is worth mentioning that ConMax can be solved exactly on special classes of graphs. The following solution for the case of trees interestingly uses bipartite matching as its main tool, not the usual dynamic programming on trees.

THEOREM 4. *Given a tree T and a configuration of k pebbles on T, ConMax can be solved in polynomial time.*

PROOF. First we guess a vertex v of T that is occupied by a pebble in the target configuration. Second we guess the maximum movement k, $0 \leq k \leq n$, required by the optimal solution.

We compute a subset of vertices of T that must be occupied by pebbles in the target configuration. For each pebble in the initial configuration, consider moving it toward v by up to k steps (stopping if it reaches v). This vertex x is the closest the pebble could get to v in any solution with a maximum movement of k. Thus, every vertex along the path from x to v (including the endpoints) must be occupied by a pebble in the target configuration. We call these vertices *forced*.

To determine whether all forced vertices can indeed be occupied by pebbles (and thus whether this guess of v and k is valid), we use bipartite matching. Define the bipartite graph $H = (W_1, W_2, F)$ where W_1 is the set of pebbles, W_2 is the set of forced vertices, and edges in F connect a pebble to every forced vertex that is within distance k in G. A maximum-cardinality matching of this graph H covers every vertex in W_2 if and only if the pebbles can be moved to occupy the forced vertices.

Any extra pebbles not used in the matching can be moved to an arbitrary forced vertex: every pebble can be moved to some forced vertex, namely x. The forced vertices induce a connected subgraph of G, so we obtain a solution to ConMax with maximum movement k if this is possible. □

2.2 CONMAX IN THE EUCLIDEAN PLANE. In this section, we introduce a method for converting algorithms for movement problems in graphs to movement problems in the Euclidean plane, applied in particular to the ConMax problem. This method works for other properties but not for Path problems, because in path problems there may exist a solution while using this method yields a "no solution" answer.

THEOREM 5. *If we have a k-approximation algorithm for ConMax in graphs, then we have an $k(1+\varepsilon)$-approximation algorithm for ConMax in the Euclidean plane, for any $\varepsilon > 0$.*

PROOF. We partition the plane into squares of side length d. We define the graph $G = (V, E)$ where V is the set of the vertices of squares used for partitioning and edges in E connect two vertices if their Euclidean distance is not more than d. Next we show that a connected configuration of the ConMax problem on the Euclidean plane could be modified to a connected configuration on graph G defined above.

For a connected configuration of pebbles on the plane, we have to move them to the vertices of graph G such that they remain connected. We pick an arbitrary pebble r and move it to one of the four vertices of the square around it. Thus r moves at most $d\sqrt{2}$. For the pebbles which were adjacent to r, first we move them exactly the same as r to keep them adjacent, next we move each one to one of the four vertices of the square around it. Note that if we choose the nearest to r, among the four vertices around a pebble, the pebble remains adjacent to r. In this way the pebbles which were adjacent to r move at most $2d\sqrt{2}$.

For the pebbles which have distance i from r, first move them the same as the pebbles which have distance $i-1$ from r and then move them to one of the four vertices of the square around them to keep them connected to r. In this way the vertices with distance i from r move at most $(i+1)d\sqrt{2}$ to reach one vertex of G and remain connected to r. Since the farthest pebble from r has distance at most $m-1$ from it, all pebbles move less than or equal to $md\sqrt{2}$ where m is the number of pebbles. Thus if we have a connected configuration with cost OPT in the Euclidean plane, the best configuration in graph G has cost less than or equal to $\mathrm{OPT}+md\sqrt{2}$.

For an initial configuration of the pebbles, first we move each pebble to one of the four vertices around it which needs movement at most $d\sqrt{2}$. Then we solve the problem on graph G. The solution of ConMax on graph G is also a valid configuration for ConMax in the Euclidean plane. Since the solution of ConMax in graph G has cost less than or equal to $\mathrm{OPT}+md\sqrt{2}$, a k-approximation algorithm for ConMax in graphs yields to an algorithm with approximation factor $k(1+md\sqrt{2})+d\sqrt{2} \leq k(1+d[(m+1)\sqrt{2}])$. Therefore we have an $k(1+\varepsilon)$-approximation algorithm for ConMax in the Euclidean plane by setting $d=\varepsilon/[(m+1)\sqrt{2}]$. □

2.3 $O(1+\sqrt{m/\mathrm{OPT}})$-Approximation for PathMax. Our techniques can be extended to obtain the same approximation factor for connectivity between just two fixed vertices s and t. The previous approach does not apply directly to this problem because not all of the pebbles need to be involved in the solution; we can select an arbitrary subset of pebbles to use for our path. First we prove that PathMax is NP-hard via a reduction from the Hamiltonian path problem.

Theorem 6. *The PathMax problem is NP-Hard.*

Proof. Duplicate the vertices of a graph G into $n=|V(G)|$ levels, for a total of n^2 vertices, and adding edges between every pair of adjacent levels corresponding to edges of G, for a total of $2|E(G)|(n-1)$ edges. For each vertex v of G, add a path of length n from each copy of v to a common new vertex $\hat{v}$, at which we place a single pebble. Finally, connect the source s to every vertex in the first level, and connect the sink t to every vertex in the last level; and attach to each of s and t a path of length n, the end of which has a single pebble. G has a Hamiltonian path if and only if we can move each pebble to an instance of its corresponding vertex with a maximum movement of n, and construct a path from s to t. □

Theorem 7. *Given an instance of the PathMax problem with m pebbles and with an optimum solution of maximum movement* OPT*, for any integer k greater than* 2OPT*, there is a polynomial-time algorithm for finding a motion with maximum movement at most* $6k+(5+4m/k)\mathrm{OPT}$.

Proof. Because OPT is an integer between 1 and n, we can guess its value at the cost of a linear factor in running time.

Next we identify vertices that cannot be on our final path between s and t. Let $d_{v,u}$ denote the distance between vertex v and u through marked vertices. Initially, we mark all vertices. Then we run the following step for unmarking vertices, until no more vertices can be unmarked.

—Unmark v if there is an $i \leq \min\{d_{s,v}, d_{t,v}\}$ and there are less than $2i+1$ pebbles within distance $i + \mathrm{OPT}$ of v.

It is clear that all vertices on the path between s and t in an optimum solution, say Q, are still marked.

Delete each pebble for which the minimum distance between its initial position and any marked vertices is more than OPT and move each remaining pebble to its nearest marked vertex with maximum movement at most OPT. Let $P = \langle s = v_0, v_1, \ldots, v_{|P|} = t\rangle$ be a shortest path between s and t that goes through marked vertices and assume $|P| = (2k+1)r_1 + r_2$, $0 \leq r_2 < 2k+1$. Call vertices $v_k, v_{3k+1}, v_{5k+2}, \ldots, v_{(2k+1)(r_1-1)+k}$ *center* vertices. Thus we have r_1 center vertices.

—For every center vertex c, move to c all pebbles within distance k of c.

It is clear that no pebble is within distance k of two center vertices. We prove that there will be at least $2(k - 2\mathrm{OPT}) + 1$ pebbles on c. We know that c is a marked vertex. Thus there are at least $2(k - 2\mathrm{OPT}) + 1$ pebbles whose distance between their initial position and c is at most $k - \mathrm{OPT}$. Because we move each pebble to a marked vertex with maximum movement OPT, there are at least $2(k - 2\mathrm{OPT}) + 1$ pebbles within distance k of c. In the next phase, distribute all pebbles on center vertices along the path P to a vertex with no pebble on P. In this phase, we move each pebble at most k.

An *empty* vertex is a vertex of P with no pebble in this configuration. Suppose we have M empty vertices. It is clear that $M \leq 4\mathrm{OPT}r_1 + r_2$ and each pebble has moved at most $2k + \mathrm{OPT}$ up to now.

A *good pebble* is a pebble that participates in constructing the path between s and t in the optimum solution and it is now on P. A pebble whose target position is either s and t in the optimum solution is also a good pebble. A *bad pebble* is a pebble that participates in constructing the path between s and t in the optimum solution but it is not currently on P. Because the length of the path between s and t in the optimum solution is at least $|P|$, we have at least M bad pebbles. Each bad pebble has moved at most OPT up to now. Consider a bad pebble p_b and a good pebble p_g whose target positions are t_b and t_g in the optimum solution. Let H_{p_b,p_g} be the distance between t_b and t_g on the path Q. The distance between the initial position of p_b and its target position in the optimum solution is at most OPT. Also, the distance between the initial position of p_b and its recent position is at most OPT. We also know that the distance between the initial position of p_g and its target position in the optimum solution is at most OPT. Furthermore, the distance between the initial position of p_g and its recent position is at most $2k + \mathrm{OPT}$, Therefore the distance between the recent positions of p_g and p_b is at most $2k + 4\mathrm{OPT} + H_{p_b,p_g}$. Let $H_{p_b} = \min_{p_g}\{H_{p_b,p_g}\}$.

—For every bad pebble with $H_{p_b} \leq M$, we move p_b to place on a good pebble and then shift all pebbles between p_b and the nearest empty vertex of P to a pebble on an empty vertex of P.

We move each pebble along one edge in each shift. It is clear that we have at least M bad pebbles p_b with $H_{p_b} \leq M$. Therefore we have at least M pebbles that can be present on path P with maximum movement at most $2k+4\mathrm{OPT}+M$. We need to move each pebble at most M to fill empty vertices. We can fill all empty vertices with this strategy. The maximum movement in our algorithm is at most $\mathrm{OPT} + \max\{2k, 2k + 4\mathrm{OPT} + M\} + M \leq 2k + 5\mathrm{OPT} + 2M$. Because $M \leq 4 \cdot \mathrm{OPT} \cdot |P|/(2k) + 2k$ and $|P|$ is at most m, the maximum movement is at most $6k + (5 + 4m/k)\mathrm{OPT}$. □

Corollary 8. *There is an $O(1 + \sqrt{m/\mathrm{OPT}})$-approximation algorithm (and thus also an $O(\sqrt{m})$-approximation algorithm) for PathMax.*

Proof. If the optimum solution cost is greater than $m/4$, then we obtain a 3-approximation algorithm by placing pebbles on the shortest path between s and t. It is clear that the length of the shortest path between s and t is at most m. We can construct a solution by finding the minimum-weight matching between the initial position of the pebbles and the vertices of the shortest path. If we had moved the pebbles to the optimum solution, we could then move them to the vertices of the shortest path by at most $m/2$ additional moves, so there is always a matching of weight at most $\mathrm{OPT} + m/2$. Thus we obtain a 3-approximation algorithm because $\mathrm{OPT} > m/4$ so $\mathrm{OPT} + m/2 < 3\mathrm{OPT}$. On the other hand, if the optimal solution cost is at most $m/4$, then we can use Theorem 7 with $k = \sqrt{m\,\mathrm{OPT}}$, and obtain an $O(1 + \sqrt{m/\mathrm{OPT}})$-approximation algorithm for PathMax. □

2.4 εm-Approximation for DirConMax

Next we consider the directed version of ConMax, DirConMax, where we obtain nearly tight results: an εm-approximation and $m^{1-\varepsilon}$ inapproximability assuming $\mathrm{P} \neq \mathrm{NP}$. Our approximability result is based on another extension of our techniques from ConMax. As before, $G^d[S]$ denotes the dth power of a graph G induced on the subset S of vertices, where distances are measured along shortest directed paths.

Lemma 9. *Consider a directed tree T with m vertices in which every edge is directed toward the root r. For any integer k between 0 and $m-1$, there is a set S of vertices of T satisfying the following properties:*

(1) *The root r is in S.*

(2) *The size of S is at most $k+1$.*

(3) *Every vertex in $T^{\lfloor m/k \rfloor+1}[S]$ has a directed path to r.*

(4) *To every vertex v of S we can assign a set $\mathrm{Match}(v)$ of vertices of T such that*
   - (a) *if $v \neq r$, $|\mathrm{Match}(v)| \geq \lfloor m/k \rfloor + 1$;*
   - (b) *the distance from any vertex w in $\mathrm{Match}(v)$ to v is at most $\lfloor m/k \rfloor$; and*
   - (c) *every vertex of T belongs to exactly one of the Match sets.*

Proof. The algorithm for constructing S proceeds as follows:

(1) Set $S \leftarrow \{r\}$.

(2) While there remain vertices of depth more than $\lfloor m/k \rfloor$:
   - (a) Let u be a vertex of maximum depth in T.
   - (b) Let v be the $\lfloor m/k \rfloor$th ancestor of u.

(c) Set $S \leftarrow S \cup \{v\}$.
(d) Let Match(v) be the set of descendants of v, including v itself.
(e) Delete vertices in Match(v) from T.

(3) Set Match(r) to the set of remaining vertices in T.

It remains to prove that the resulting set S and the Match sets satisfy the required properties. Property 1 follows from Step 1. Property 2 follows from Property 4. Property 3 follows from Property 4 because the parent of any vertex v added to S must be in another Match set. Property 4(a) follows because Match(v) includes the path from u to v. Property 4(b) follows because, in each round, u is the vertex of maximum depth in T and thus the vertex of maximum depth within Match(v). Property 4(c) follows because every vertex in T is either deleted because it is in Match(v) (and a vertex can be deleted only once) or it belongs to Match(r). □

Now we use Lemma 9 to compress OPT into a small, guessable form:

Theorem 10. *For any integer constant k, there is an $n^{k+O(1)}$ algorithm that, given an instance of DirConMax with m pebbles and root r, finds a motion with maximum movement at most* $\mathrm{OPT} + 2\lfloor m/k \rfloor$. *(In particular, this algorithm is a $2\lfloor m/k \rfloor$-approximation.)*

Proof. First we guess the maximum movement l, $0 \le l \le n$, required by the optimal solution OPT. Second we guess the set S_{OPT} obtained by applying Lemma 9 to a spanning tree of OPT. We guess the size i of S_{OPT}, $1 \le i \le k+1$. Then we guess a subset S of $|S| = i$ vertices, including r, such that every vertex in $G^{\lfloor m/k \rfloor + 1}[S]$ has a directed path to the root r. The number of choices for these guesses is $O(kn^{k+1})$, which is polynomial for fixed k.

To determine whether all pebbles can move to the vertices of S by at most $l + \lfloor m/k \rfloor$ movement, we use the version of maximum flow with minimum and maximum capacities on the edges, which can be solved in polynomial time; see West [2001, pages 186–187]. Define a bipartite graph $H = (W_1, W_2, F)$ where W_1 is the set of pebbles, W_2 is the set of vertices of S, and edges in F connect a pebble to a vertex u of S if the pebble can move to u by at most $l + \lfloor m/k \rfloor$ movements. These edges have a minimum and maximum capacity of 1. Add a source vertex s connected to all vertices of W_1, and add a sink vertex t connected from all vertices of W_2. The edges incident to s have a minimum and maximum capacity of 1, and the edges incident to t have a minimum capacity of $\lfloor m/k \rfloor$ and a maximum capacity of m. If the maximum flow "matches" every pebble to a vertex of S, then we obtain the desired Match sets; otherwise, S is invalid.

When we find a subset S and the Match sets with the properties of Lemma 9, we can construct a solution as follows. We move each pebble to its matched vertex in S, which leaves at least $\lfloor m/k \rfloor + 1$ pebbles on each vertex of S. Then, for each vertex u in S other than the root r, we move $\lfloor m/k \rfloor$ of the pebbles from u to occupy the vertices along a shortest path in G from u to its parent in a spanning tree of $G^{\lfloor m/k \rfloor + 1}[S]$. Thus each pebble moves at most $l + 2\lfloor m/k \rfloor$, as desired. □

2.5 $\Omega(n^{1-\varepsilon})$ Inapproximability for DirConMax. Next we prove that the εn-approximation algorithm from the previous section is essentially tight, assuming only that $\mathrm{P} \ne \mathrm{NP}$ without the use of PCP-type arguments:

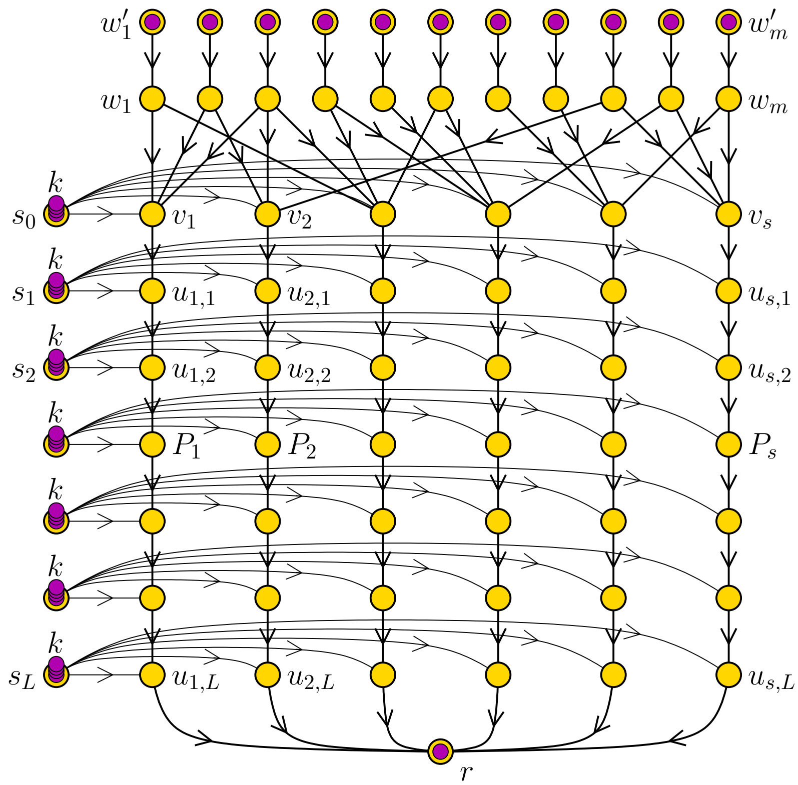

Fig. 3. Reduction from set cover in Theorem 11.

THEOREM 11. *For every constant ε, $0 < \varepsilon < 1$, it is NP-hard to approximate DirConMax within an $n^{1-\varepsilon}$ factor.*

PROOF. We prove that, if DirConMax can be approximated within $n^{1-\varepsilon}$, then set cover can be solved in polynomial time. Let $S = (E, C, k)$ be an instance of set cover, where $E = \{e_1, e_2, \ldots, e_m\}$ is the universe of elements, $C = \{c_1, c_2, \ldots, c_s\}$ is the set of subsets of E, and k is an integer. Without loss of generality, assume that $m \geq s$; otherwise, we can place $s - m$ dummy elements $e_{m+1}, \ldots, e_s$ in every subset in C.

We convert the set-cover instance S into a graph G as follows; refer to Figure 3. Let L be $m^{2/\varepsilon}$. We start with a root vertex r in G and then, for every subset $c_i \in C$, we add a vertex v_i to G. Then, for each v_i, we add a directed path P_i of length $L + 1$ from v_i to r. Label the vertices along path P_i from source to destination as $v_i, u_{i,1}, u_{i,2}, \ldots, u_{i,L}, r$. For each j, $1 \leq j \leq L$, we add a vertex s_j and we add an edge from s_j to $u_{i,j}$ for each i, $1 \leq i \leq s$. We also add a vertex s_0 and we add an edge from s_0 to v_i for each i, $1 \leq i \leq s$. For every element $e_i \in E$, we add two vertices w_i and w'_i to G, and we add an edge from w'_i to w_i. Finally, we add an edge from w_i to v_j precisely when $e_i \in c_j$. This graph has $n = 1 + (s+1)(L+1) + 2m = O(m^{2/\varepsilon+1})$ vertices. In our instance of DirConMax, we place one pebble on each w'_i and one pebble on r. We also place k pebbles on each s_i, $0 \le i \le L$.

If S has a set cover $C' = \{c_{p_1}, c_{p_2}, \ldots, c_{p_{k'}}\}$ of size $k' \le k$, then we can connect the pebbles in G using a maximum movement of 1. Namely, we move k' pebbles from each s_i, $1 \le i \le L$, to $u_{p_1,i}, u_{p_2,i}, \ldots, u_{p_{k'},i}$. Then we move the pebble from w'_j to w_j for each j, $1 \le j \le m$, and we move k' pebbles from s_0 to $v_{p_1}, v_{p_2}, \ldots, v_{p_{k'}}$.

Now we prove that, if S has no set cover of size at most k, then the maximum movement of any solution to this instance of DirConMax is at least $m^{2/\varepsilon-1}$. Consider a solution with maximum movement less than $m^{2/\varepsilon-1}$ and let L' be $m^{2/\varepsilon-1}$. Because the pebble at r can never move, the final positions of the pebbles must form a directed tree T rooted at r. We call a path P_i *semicompleted* if $u_{i,L'}$ is in T. Let $P' = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{k'}}\}$ be the set of semicompleted P_i's. We assert that the set $C'' = \{c_{i_1}, c_{i_2}, \ldots, c_{i_{k'}}\}$ is a set cover of size k' for the instance S. Let f_i be the final position of the pebble starting on w'_i. This vertex f_i cannot be $u_{i',j}$ for any i' and j with $1 \le i' \le s$ and $L' \le j \le L$. So the directed path from f_i to the root r must visit some vertex $u_{i_j,L'}$ along a semicompleted path P_{i_j} for some j, $1 \le j \le k'$. Thus, $e_i \in c_{i_j}$, and this property holds for all i, $1 \le i \le m$, so C'' is indeed a set cover of size k' for S. Now we prove that $k' \le k$, contradicting that S has no such set cover. For each j, $1 \le j \le k'$, we need at least $L - L'$ pebbles to occupy the vertices $u_{i_j,j'}$, $L' < j' \le L$. The total number of pebbles that can have a final position of $u_{i,j}$, where $1 \le i \le s$ and $L' < j \le L$, is less than kL. Thus $k'(L-L') < kL$, that is, $1 - 1/m < k/k'$. Because $k' \le s \le m$, $1 - 1/k' \le 1 - 1/m$, and therefore $1 - 1/k' < k/k'$, that is, $k' \le k$.

On the other hand, if there is a set cover of size at most k, then there is a solution with maximum movement 1. Thus any solution to DirConMax with maximum movement at least L' has an approximation ratio at least $L' = m^{2/\varepsilon-1}$, which is asymptotically larger than $m^{2/\varepsilon+1-2-\varepsilon} = (m^{2/\varepsilon+1})^{1-\varepsilon} = \Theta(n^{1-\varepsilon})$. Therefore, we can decide whether there is a set cover of size at most k by testing whether an $O(n^{1-\varepsilon})$-approximation algorithm for ConSum produces a solution of maximum movement less than $m^{2/\varepsilon-1}$. $\square$

3 *Minimum Maximum Movement to Independence*

It is NP-hard to decide whether IndMax even has a valid solution: an instance has a solution precisely if the graph has an independent set of size m, the number of pebbles. Thus, to obtain any approximability result, we must restrict our attention to special family of graphs.

In this section we focus on a particularly useful case of the *Euclidean plane*. This scenario has applications in the fields of map labeling and sensor networks, as described in the introduction. Recall that in this case we define $n = m$. We use the notation d for a more general notion of Euclidean distance: for a point p and a finite set Q of points, $d(p, Q)$ denotes the minimum distance $\min_{q \in Q} d(p, q)$.

Theorem 12. *There is a polynomial-time algorithm solving IndMax in the Euclidean plane using maximum movement at most the optimal plus* $1 + \frac{1}{\sqrt{3}}$.

The heart of our approximation algorithm is the triangular lattice, illustrated in Figure 4, in which every two distinct vertices have distance at least 1. Thus, these

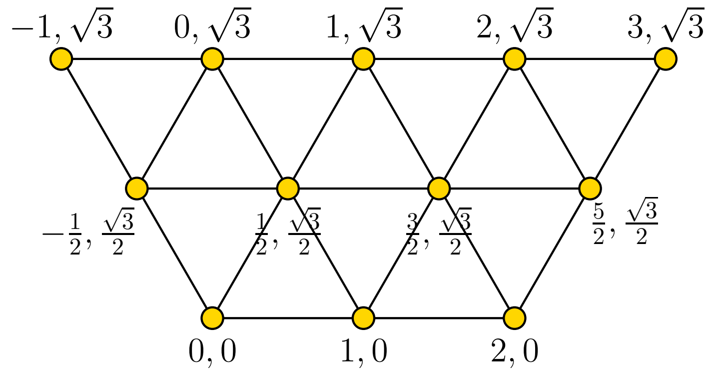

Fig. 4. Decomposition of the plane into equilateral triangles.

vertices induce an independent set of the plane. The vertex set is given by

$$A = \left\{ (i, j\sqrt{3}), (i + \tfrac{1}{2}, j\sqrt{3} + \tfrac{\sqrt{3}}{2}) \;\middle|\; i, j \in \mathbb{Z} \right\}.$$

Let C denote the decomposition of the plane into equilateral triangles with side length 1 induced by this lattice.

For a finite set R of points, we define two additional concepts. Let Neighbor(R) denote the set of points in A whose distance to R is at most $1 + \frac{1}{\sqrt{3}}$. Let Circle(R) denote the union of disks centered at points in R with radius $\frac{1}{2}$. In particular, if every two distinct points in R have distance at least 1, then Circle(R) has area $|R| \cdot \frac{\pi}{4}$.

Lemma 13. *The optimal solution has maximum movement at most* $2n - 2$.

Proof. Suppose for contradiction that there is a pebble x with initial position p and with target position q in the optimal solution, yet $d(p, q) > 2n - 2$. We define n points $r_0, r_1, \ldots, r_{n-1}$ on the line segment from p to q according to $d(p, r_i) = 2i$. The distance between any two of these points r_i and r_j, $i \neq j$, is at least 2, so any point can have distance less than 1 with at most one of these points $r_0, r_1, \ldots, r_{n-1}$. By the Pigeonhole Principle, there is at least one point r_i that is not within distance 1 of the target position of any pebble other than x. Thus we can change the target position of pebble x to r_i and obtain a valid solution in which the movement of x is at most $2n - 2$. By induction, we can reduce the movement of every pebble to at most $2n - 2$, giving us a solution with maximum movement at most $2n - 2$, contradicting optimality of the original solution.[2] □

Lemma 14. *The number of points in A within distance at most $2n - 2$ from an arbitrary point p in the plane is at most a polynomial function of n.*

Proof. Consider the square S centered at p and with side length $4n - 4$. All points of A within distance $2n - 2$ from p are in this square. Consider a decomposition of S into a grid of subsquares of side length $\frac{1}{2}$. Because the distance

[2]This argument can be improved to obtain a bound of $O(\sqrt{n})$ on the maximum motion, but it does not affect our main result.

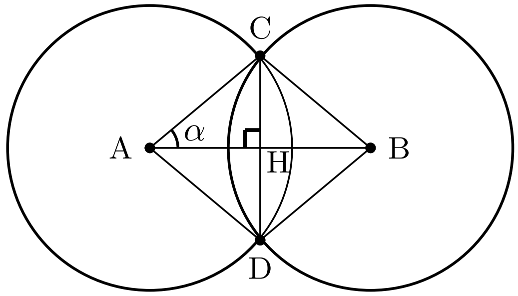

Fig. 5. Intersection of the circles C_A and C_B.

between each pair of points in such a subsquare is at most $1/\sqrt{2} < 1$, at most one point of A can be in each subsquare. Thus the number of subsquares is an upper bound on the number of points of A in S, which is an upper bound on the number of points of A within distance $2n-2$ from p. The number of subsquares is $(4n-4)^2/(\frac{1}{2})^2 = O(n^2)$. □

LEMMA 15. *Let C_A and C_B be two disks of radius $1/\sqrt{3}$ centered at points A and B, respectively. Let $d = d(A,B)$ be the distance between A and B. The area of intersection of the two disks C_A and C_B is $\frac{2}{3}\arccos(d\sqrt{3}/2) - d\sqrt{\frac{1}{3} - \frac{1}{4}d^2}$.*

PROOF. In Figure 5, we have $\overline{AC} = \overline{BC} = \overline{AD} = \overline{BD} = 1/\sqrt{3}$, $\overline{AB} = d$, $\angle BAC = \alpha$. Thus, $\cos\alpha = \overline{AH}/\overline{AC} = (d/2)/(1/\sqrt{3}) = d\sqrt{3}/2$, so $\alpha = \arccos(d\sqrt{3}/2)$. Hence, the area of the pie wedge of C_A given by the angle $\angle DAC$ is $\frac{2\alpha}{2\pi}\cdot\frac{\pi}{3} = \frac{1}{3}\alpha = \frac{1}{3}\arccos(d\sqrt{3}/2)$. By symmetry, the area of the pie wedge of C_B given by the angle $\angle CBD$ is the same. These pie wedges overlap at precisely the intersection of C_A and C_B. Their union is the quadrangle $ABCD$. Thus, the desired area of intersection is the sum of the areas of the pie wedges, $\frac{2}{3}\arccos(d\sqrt{3}/2)$, minus the area of the quadrangle $ABCD$. Now $\overline{CH} = \sqrt{\overline{AC}^2 - (\overline{AB}/2)^2} = \sqrt{\frac{1}{3} - \frac{1}{4}d^2}$, so the area of the quadrangle $ABCD$ is $\frac{1}{2}\overline{AB}\cdot\overline{CD} = \overline{AB}\cdot\overline{CH} = d\sqrt{\frac{1}{3} - \frac{1}{4}d^2}$. Therefore, the desired area of intersection of C_A and C_B is $\frac{2}{3}\arccos(d\sqrt{3}/2) - d\sqrt{\frac{1}{3} - \frac{1}{4}d^2}$, as desired. □

Now we are ready to prove the main theorem.

PROOF OF THEOREM 12. For $i \in \{1, 2, \ldots, n\}$, let p_i and q_i be the initial and target position of pebble i in the optimal solution OPT. Because OPT is a solution to IndMax, we have $d(q_i, q_j) \geq 1$ for all distinct $i, j \in \{1, 2, \ldots, n\}$. Furthermore, the optimal solution minimizes $\mathrm{OPT} = \max_{1\leq i\leq n} d(p_i, q_i)$ subject to this constraint. First we prove that there is a polynomial-time algorithm to move every pebble to a point of A such that no two pebbles move to the same point, and subject to minimizing the maximum movement M. Then we prove that we can move the pebbles from their target positions in OPT to points of A so that no two pebbles move to the same point and each pebble moves at most $1+\frac{1}{\sqrt{3}}$. Thus, our approximate solution of maximum movement M satisfies $M \le \mathrm{OPT}+1+\frac{1}{\sqrt{3}}$.

The algorithm constructs a complete weighted bipartite graph $H=(X,Y,E)$. For $i \in \{1,2,\ldots,n\}$, we place a vertex x_i in X representing pebble i. By Lemma 13, $\mathrm{OPT} \le 2n-2$. By the second part of the proof, the optimal movement M to points of A satisfies $M \le \mathrm{OPT}+1+\frac{1}{\sqrt{3}} \le 2n-2+1+\frac{1}{\sqrt{3}}$. Thus, in M, no pebble moves more than $2n$. For each point p of A within distance $2n-2$ from the set $\{p_1,p_2,\ldots,p_n\}$ of initial positions, we place a vertex in Y_p. By Lemma 14, the number of these points is polynomial in n, so the graph H has polynomial size. For each $x \in X$ and $y \in Y$, we set the weight $w(x,y)=d(x,y)$. The algorithm finds a perfect matching in H of minimum maximum weight. For each edge (x_i,y_p) in the matching, we move the ith pebble to point p of A. In this way, we move the pebbles to points of A such that no two pebbles move to the same point using the minimum maximum movement.

Now we reach the heart of the proof: we prove that we can transform OPT by moving each target position by at most $1+\frac{1}{\sqrt{3}}$ such that every new target position is a point of A and no two target positions are the same. We prove that there is a perfect matching from the set $Q=\{q_1,q_2,\ldots,q_n\}$ of target positions in OPT to the points of A such that the distance between matched points is at most $1+\frac{1}{\sqrt{3}}$. By Hall's Theorem, it suffices to show that, for each subset $R \subseteq Q$, $|R| \le |\,\mathrm{Neighbor}(R)|$.

Consider a subset $R=\{r_1,r_2,\ldots,r_m\} \subseteq Q$, and the region $W=\mathrm{Circle}(R)$. Because the distance between every two points in R is at least 1, $\mathrm{Circle}(R)$ has area $|R|\cdot\frac{\pi}{4}$. Consider the set $\mathrm{Neighbor}(R) \subseteq A$, and the region $V=\mathrm{Circle}(\mathrm{Neighbor}(R))$. Again V has area $|\,\mathrm{Neighbor}(R)|\cdot\frac{\pi}{4}$. We prove that the area of $\mathrm{Circle}(R)$ is at most the area of $\mathrm{Circle}(\mathrm{Neighbor}(R))$, which implies $|R| \le |\,\mathrm{Neighbor}(R)|$, completing the proof.

Consider a disk of radius $\frac{1}{\sqrt{3}}$ centered at each point of R. Define the region S to consist of the equilateral triangles of the decomposition C that intersect at least one of these disks. The vertices of the triangles in S are the points of $\mathrm{Neighbor}(R)$, because these vertices are the points of A within distance $1+\frac{1}{\sqrt{3}}$ from the points of R.

Next we prove that $\mathrm{Area}(\mathrm{Circle}(\mathrm{Neighbor}(R))) \ge \mathrm{Area}(S)\cdot\frac{\pi}{2\sqrt{3}}$. For each triangle T in S, there are only three circles of $\mathrm{Circle}(\mathrm{Neighbor}(R))$ that intersect with it, those whose centers are placed on the vertices of T; see Figure 6. These circles have area $\frac{\pi}{8}$ in common with T. Therefore, the ratio of this common area to the area of T is $\frac{\pi}{8}/\frac{\sqrt{3}}{4}=\frac{\pi}{2\sqrt{3}}$. Because this ratio is the same for every triangle in S, so is the ratio $\mathrm{Area}(\mathrm{Circle}(\mathrm{Neighbor}(R)) \cap S)/\,\mathrm{Area}(S)=\frac{\pi}{2\sqrt{3}}$. Therefore, $\mathrm{Area}(\mathrm{Circle}(\mathrm{Neighbor}(R)))/\,\mathrm{Area}(S) \ge \frac{\pi}{2\sqrt{3}}$ or $\mathrm{Area}(\mathrm{Circle}(\mathrm{Neighbor}(R))) \ge \mathrm{Area}(S)\cdot\frac{\pi}{2\sqrt{3}}$

Next we prove that $\mathrm{Area}(S)\cdot\frac{\pi}{2\sqrt{3}} \ge \mathrm{Area}(\mathrm{Circle}(R))$, which would prove the theorem. For each point r in R, we assign a region $\mathrm{Region}(r)$ contained in S of area at least $\frac{\sqrt{3}}{2}$ such that every two regions $\mathrm{Region}(r)$ and $\mathrm{Region}(s)$, $r \ne s$, are disjoint. Because these regions pack a subset of S, we obtain $\mathrm{Area}(S) \ge |R|\cdot\frac{\sqrt{3}}{2}$.

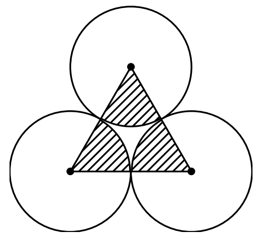

Fig. 6.

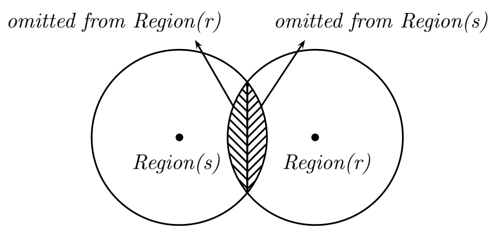

Fig. 7. The omitted region from the disks.

Therefore, $\mathrm{Area}(S) \cdot \frac{\pi}{2\sqrt{3}} \geq |R| \cdot \frac{\sqrt{3}}{2} \cdot \frac{\pi}{2\sqrt{3}} = |R| \cdot \frac{\pi}{4} = \mathrm{Area}(\mathrm{Circle}(R))$.

It remains to assign to each point r of R a region $\mathrm{Region}(r)$. We do so according to the following algorithm:

(1) For each point r in R, initially set $\mathrm{Region}(r)$ to the disk $\mathrm{Region}_0(r)$ of radius $\frac{1}{\sqrt{3}}$ centered at r. (Thus, $\mathrm{Region}_0(r)$ has area $\frac{\pi}{3}$.)

(2) For two arbitrary points r and s in R, if $\mathrm{Region}_0(r)$ intersects $\mathrm{Region}_0(s)$, omit half of their intersection from $\mathrm{Region}(r)$ and omit the other half from $\mathrm{Region}(s)$ according to the perpendicular bisector of r and s, as shown in the Figure 7.

Obviously, the resulting regions are pairwise disjoint and each region is contained in S.

We prove that the sum of the areas omitted from $\mathrm{Region}(r)$ is at most $\frac{\pi}{3} - \frac{\sqrt{3}}{2}$, for each point r in R; thus, $\mathrm{Region}(r)$ keeps an area of at least $\frac{\sqrt{3}}{2}$ as desired. Let $e(r,s) = \frac{1}{2}\,\mathrm{Area}(\mathrm{Region}_0(r) \cap \mathrm{Region}_0(s))$ be the area of $\mathrm{Region}(r)$ omitted because of $\mathrm{Region}(s)$. (This definition actually overestimates the omitted area if multiple overlapping regions are omitted.) Thus, our goal is to prove that $\sum_{s \neq r \in R} e(r,s) \leq \frac{\pi}{3} - \frac{\sqrt{3}}{2}$. For a fixed point r in R, consider the points s of R that have a nonzero value $e(r,s)$. Sort these points according to the angle of the ray from r to s with respect to the x axis, resulting in a sequence $s_1, s_2, \ldots, s_l$.

We prove that $e(r,s_i) + e(r,s_{i+1}) \leq \left(\frac{1}{3} - \frac{\sqrt{3}}{2\pi}\right) \angle s_i r s_{i+1}$. Let $a = d(r,s_i)$, $b = d(r,s_{i+1})$, and $c = d(s_i,s_{i+1})$. Because r, s_i, and s_{i+1} are points of R, we have $a,b,c \geq 1$, and in particular, $c^2 \geq 1$. Because $e(r,s_i), e(r,s_{i+1}) \neq 0$, we have $a,b \leq \frac{2}{\sqrt{3}}$. By the Law of Cosines, we have $c^2 = a^2 + b^2 - 2ab\cos(\angle s_i r s_{i+1})$, and thus $\cos(\angle s_i r s_{i+1}) \leq \frac{a^2+b^2-1}{2ab}$, so $\angle s_i r s_{i+1} \geq \arccos\left(\frac{a^2+b^2-1}{2ab}\right)$. By Lemma 15, we have

$$e(r,s_i)=\tfrac{1}{2}\left(\tfrac{2}{3}\arccos(a\sqrt{3}/2)-a\sqrt{\tfrac{1}{3}-\tfrac{1}{4}a^2}\right)$$

and

$$e(r,s_{i+1})=\frac{1}{2}\left(\frac{2}{3}\arccos(b\sqrt{3}/2)-b\sqrt{\frac{1}{3}-\frac{1}{4}b^2}\right).$$

We can check algebraically that

$$\begin{aligned}&\tfrac{1}{2}\left(\tfrac{2}{3}\arccos(a\sqrt{3}/2)-a\sqrt{\tfrac{1}{3}-\tfrac{1}{4}a^2}+\tfrac{2}{3}\arccos(b\sqrt{3}/2)-b\sqrt{\tfrac{1}{3}-\tfrac{1}{4}b^2}\right)\\ \le\ &\left(\frac{1}{3}-\frac{\sqrt{3}}{2\pi}\right)\arccos\left(\frac{a^2+b^2-1}{2ab}\right).\end{aligned}\tag{1}$$

(For the interested reader, a proof is in Appendix A.) Thus, $e(r,s_i)+e(r,s_{i+1})\le\left(\frac{1}{3}-\frac{\sqrt{3}}{2\pi}\right)\angle s_i r s_{i+1}$.

By summing the previous inequality, we obtain

$$\begin{aligned}e(r,s_1)+e(r,s_2)&\le\left(\tfrac{1}{3}-\tfrac{\sqrt{3}}{2\pi}\right)\angle s_1 r s_2\\ e(r,s_2)+e(r,s_3)&\le\left(\tfrac{1}{3}-\tfrac{\sqrt{3}}{2\pi}\right)\angle s_2 r s_3\\ &\ \vdots\quad\vdots\\ e(r,s_l)+e(r,s_1)&\le\left(\tfrac{1}{3}-\tfrac{\sqrt{3}}{2\pi}\right)\angle s_l r s_1\\ \Longrightarrow\qquad 2\sum_{i=1}^{l}e(r,s_i)&\le\left(\tfrac{1}{3}-\tfrac{\sqrt{3}}{2\pi}\right)2\pi\end{aligned}$$

Therefore, $\sum_{i=1}^{l}e(r,s_i)\le\left(\frac{\Pi}{3}-\frac{\sqrt{3}}{2}\right)$ as desired.

In summary, for each $R\subseteq Q$, we have $|R|\le|\operatorname{Neighbor}(R)|$, so there is a perfect matching from Q to Neighbor(Q); thus, we can move each pebble to a unique point in A such that the maximum movement is at most $1+\frac{1}{\sqrt{3}}$. □

4 *Minimum Maximum Movement to Perfect Matchability*

In contrast to the difficult problems of ConMax and IndMax, we show that minimizing movement does not make perfect matching much harder in graphs: there is a polynomial-time algorithm for MatchMax.

LEMMA 16. *For a given graph G, if two pebbles p and q are within distance 1 in the target configuration, then $|\pi(p)|+|\pi(q)|\ge d_G(p,q)-1$, and thus $\max\{|\pi(p)|,|\pi(q)|\}\ge\left\lceil\frac{d_G(p,q)-1}{2}\right\rceil$.*

PROOF. Each step in the motion path of p or q may decrease $d_G(p,q)$ by at most 1. Therefore the sum of the movements of p and q must be at least their original distance $d_G(p,q)$ minus their target distance of 0 or 1. □

THEOREM 17. *There is a polynomial-time algorithm solving MatchMax.*

Proof. We assume that the number of pebbles in each connected component of G is even; otherwise, no solution exists. We can also consider each connected component separately, so we assume without loss of generality that G is connected. Let $p_1, p_2, \ldots, p_{2n}$ denote the pebbles.

Define the weighted complete undirected graph H as follows. For each pebble p_i we make a vertex v_i in graph H. For each edge $e = \{v_i, v_j\}$ in H, we set its weight $w(e)$ to $\lceil \frac{d_G(p_i,p_j)-1}{2} \rceil$. Define the *maximum weight* $w(M) = \max_{e \in M} w(e)$ of a perfect matching M of H to be the maximum weight of its edges.

Our algorithm computes a perfect matching M in H of minimum maximum weight $w(M)$ (in polynomial time), and converts this matching into a motion as follows. For each edge $\{v_i, v_j\}$ in the matching M, we move p_i by $\left\lceil \frac{d_G(p_i,p_j)-1}{2} \right\rceil$ steps toward p_j along a shortest path from p_i to p_j in G, and we move p_j by $\left\lceil \frac{d_G(p_i,p_j)-1}{2} \right\rceil$ steps toward p_i along the same shortest path. (Note that $\left\lceil \frac{d_G(p_i,p_j)-1}{2} \right\rceil \geq 0$.) Thus, after the motion, p_i and p_j are at distance at most 1 in G. The maximum movement in this motion is the maximum weight of such a matched edge $\{v_i, v_j\}$, which is precisely $w(M)$.

Now we argue that no solution to MatchMax has maximum movement less than $w(M)$. By definition of MatchMax, any solution induces a perfect matching M' in the graph H (i.e., on the pebbles) with the property that, in the target configuration, every two matched pebbles have distance at most 1 in G. For every edge $e = \{v_i, v_j\}$ in this matching M', by Lemma 16, we have that $\max\{|\pi(p_i)|, |\pi(p_j)|\} \geq \left\lceil \frac{d_G(p,q)-1}{2} \right\rceil = w(e)$. Therefore, the maximum movement in the solution must be at least $\max_{e \in M'} w(e) = w(M')$. But M was chosen to minimize this lower bound $w(M)$, so every solution must have maximum movement at least $w(M)$, proving optimality of our strategy of maximum movement $w(M)$. □

5 *Minimum Total Movement*

In this section, we consider the variations of the movement problems in which the goal is to minimize total movement instead of maximum movement. For both ConSum and MatchSum, we obtain tight results.

5.1 Connectivity: $\Omega(n^{1-\varepsilon})$ Inapproximability

Lemma 18. *Suppose the graph G is a path of length n with vertices $v_0, v_1, \ldots, v_{n-1}$, and initially there are $n-1$ pebbles, one on each vertex $v_1, v_2, \ldots, v_{n-1}$. To occupy v_0 with a pebble and maintain connectivity, we need motion with total movement at least $n-1$.*

Proof. In the initial configuration, the sum of the distances from each pebble to v_0 is $d_1 = \sum_{i=1}^{n-1} i = n(n-1)/2$. In the target configuration, this sum is at most $d_2 = (n-1)(n-2)/2$. Thus the total motion must be at least the difference between these two values, that is, $d_1 - d_2 = n-1$. □

Theorem 19. *For every constant ε, $0 < \varepsilon < 1$, it is NP-hard to approximate ConSum within an $n^{1-\varepsilon}$ factor.*

Proof. We prove that, if the ConSum problem can be approximated within $n^{1-\varepsilon}$, then set cover can be solved in polynomial time. Let $S = (E, C, k)$ be an instance of set cover, where $E = \{e_1, e_2, \ldots, e_m\}$ is the universe of elements, $C = \{c_1, c_2, \ldots, c_s\}$ is the set of subsets of E, and k is an integer. Without loss of generality, assume that $m \geq s$; otherwise, we can place $s - m$ dummy elements $e_{m+1}, \ldots, e_s$ in every subset in C.

We convert the set-cover instance S into a graph G as follows. For every element $e_i \in E$, we add to G a path P_i of length $m^{3/\varepsilon-2}$ with one endpoint labeled v_i. For every subset $c_j \in C$, we add a vertex w_j to G and we connect vertices v_i and w_j precisely when $e_i \in c_j$. We also add a vertex v^* to G connected to $w_1, w_2, \ldots, w_s$. Finally, we attach v^* to one endpoint of a path P_0 of length $m^{3/\varepsilon-2}$. This graph G has $n = 1 + s + (m+1)m^{3/\varepsilon-2} = \Theta(m^{3/\varepsilon-1})$ vertices. In our instance of ConSum, we place one pebble on every vertex of every path P_i, $0 \leq i \leq m$, and we place $k+1$ pebbles on v^*.

If S has a set cover $C' = \{c_{p_1}, c_{p_2}, \ldots, c_{p_{k'}}\}$ of size $k' \leq k$, then we can connect the pebbles in G using k' total movement by moving one pebble each on vertex v^* to vertices $w_{p_1}, w_{p_2}, \ldots, w_{p_{k'}}$. Conversely, by Lemma 18, if we move pebbles on one of paths P_i to connect with an adjacent vertex, then we need at least $|P_i| = m^{3/\varepsilon-2}$ total movement. Thus, any solution with less than $m^{3/\varepsilon-2}$ total movement does not move a pebble off of any path P_i, $0 \leq i \leq m$. Furthermore, any such solution can only move k of the pebbles on vertex v^*; if it moved all $k+1$ pebbles from vertex v^*, it would disconnect P_0 from the remaining pebbles. It helps to move these k pebbles only to vertices w_j, so any such solution must move $k' \leq k$ pebbles to a set of vertices $w_{p_1}, w_{p_2}, \ldots, w_{p_{k'}}$ that covers all v_i vertices, which corresponds to a set cover $\{c_{p_1}, c_{p_2}, \ldots, c_{p_{k'}}\}$ of size at most k.

On the other hand, if there is a set cover of size at most k, then we claim that any solution to ConSum with total movement at least $m^{3/\varepsilon-2}$ has an approximation ratio $\omega(n^{1-\varepsilon})$. Because $k \leq s \leq m$, the approximation ratio is at least $m^{3/\varepsilon-2}/m = m^{3/\varepsilon-3}$, which is asymptotically larger than $m^{3/\varepsilon-3-1+\varepsilon} = (m^{3/\varepsilon-1})^{1-\varepsilon} = \Theta(n^{1-\varepsilon})$ because $\varepsilon < 1$. Therefore, we can decide whether there is a set cover of size at most k by testing whether an $O(n^{1-\varepsilon})$-approximation algorithm for ConSum produces a solution of total movement less than $m^{3/\varepsilon-2}$. □

5.2 Connectivity: $O(\min\{n \log n, m\})$ Approximation. Note that $O(nm)$-approximation is trivial for ConSum, where n is the number of vertices and m is the number of pebbles. If the pebbles already induce a connected graph, then there is nothing to do. Otherwise, the optimal solution has total motion at least 1, and we can move all m pebbles to any particular vertex using at most $m(n-1)$ total movement.

The $O(\min\{n \log n, m\})$-approximation algorithm consists of two parts, one for each term of the min.

Theorem 20. *There is an $O(n \log n)$-approximation algorithm for ConSum.*

Proof. Consider an instance of the ConSum problem, which consists of a graph G and a configuration of pebbles $p_1, p_2, \ldots, p_k$. Let $\mathrm{OPT}_{\mathrm{ConSum}}$ be the optimal solution for this instance. Let U be the subset of vertices of G occupied by at least one pebble in the initial configuration. We consider the instance of the node

Steiner tree problem defined by graph G and vertex subset U. Let $\mathrm{OPT}_{\mathrm{NSteiner}}$ be the optimal solution to this instance, that is, the minimum set W of vertices in G such that $U \cup W$ induces a connected graph. It is known how to approximate this problem within an $O(\lg |U|) = O(\lg n)$ factor, and that no better approximation is possible unless P = NP [Klein and Ravi 1995; Guha and Khuller 1999].

First we prove that $\mathrm{OPT}_{\mathrm{NSteiner}} \leq \mathrm{OPT}_{\mathrm{ConSum}}$. Let $\pi(p_i)$ denote the motion path of p_i in the optimal solution $\mathrm{OPT}_{\mathrm{ConSum}}$. We can form a solution to $\mathrm{OPT}_{\mathrm{NSteiner}}$ by including all vertices in $\pi(p_i)$ except the initial position of p_i. Because the target positions of the pebbles induce a connected graph, attaching the paths $\pi(p_i)$ to this connected graph also results in a connected graph. The size of this solution is precisely the total length of the paths $\pi(p_i)$, which is the total motion $\mathrm{OPT}_{\mathrm{ConSum}}$. Therefore $\mathrm{OPT}_{\mathrm{NSteiner}} \leq \mathrm{OPT}_{\mathrm{ConSum}}$.

Next we show how to construct a solution to ConSum whose total movement is at most $n\mathrm{OPT}_{\mathrm{NSteiner}}$. Let T be a spanning tree of the connected subgraph induced by $U \cup \mathrm{OPT}_{\mathrm{NSteiner}}$. Root this tree T at an arbitrary node $r \in U$. We define a motion of pebbles by repeatedly performing the following step. Pick a pebble on a vertex $v \neq r$ such that no descendant vertices of v in T are occupied by pebbles. Let $v, v_1, v_2, \ldots, v_j, r$ denote the path in T from v to r. We move a pebble on v to the first vertex v_i that is connected to r via other pebbles, that is, for which $v_{i+1}, v_{i+2}, \ldots, v_j$ are occupied by pebbles. Thus, the target position of pebble v is a vertex without a pebble, so it is in $\mathrm{OPT}_{\mathrm{NSteiner}}$. Therefore, the number of moved pebbles is at most $\mathrm{OPT}_{\mathrm{NSteiner}}$, and the motion of each pebble is at most $n-1$, for a total cost of $\mathrm{OPT}_{\mathrm{NSteiner}}(n-1)$. □

Note that this analysis is tight. Consider a graph consisting of just a path $v_1, v_2, \ldots, v_{2n-1}$ such that v_n has no pebble, but every other vertex has exactly one pebble. By Lemma 18, $\mathrm{OPT}_{\mathrm{ConSum}} = \Theta(n)$, but $\mathrm{OPT}_{\mathrm{NSteiner}} = 1$.

THEOREM 21. *There is an $O(m)$-approximation algorithm for ConSum.*

PROOF. Consider an instance of the ConSum problem, which consists of a graph G and a configuration of pebbles $p_1, p_2, \ldots, p_m$. Let $\mathrm{OPT}_{\mathrm{ConSum}}$ be the optimal solution for this instance. We fix the pebble p_m and try to connect other pebbles to it. For a path P in G, we set its weight $W(P)$ as the number of unoccupied vertices within P. For each pebble $p_i (1 \leq i \leq m)$ define distance $\mathrm{dist}(p_i)$, the weight of the path who has the minimum among all the paths from p_i to p_m. Since G is connected, each pebble has some finite distance.

LEMMA 22. $\max_{1 \leq i \leq m} \mathrm{dist}(p_i) \leq \mathrm{OPT}_{\mathrm{ConSum}}$.

PROOF. Suppose that p_i has the maximum distance. Assume that in the optimal solution, p_i is moved to u by m_i movements and p_m is moved to v by m_m movements. Consider the path P in the target configuration which connects p_i to p_m. Because all vertices of P must be occupied, the other pebbles should have at least $W(P)$ movements in total. Therefore $\mathrm{OPT}_{\mathrm{ConSum}} \geq m_i + m_m + W(P)$. Consider the path P' including P and the vertices that each of p_i and p_m have traversed. P' may not have more than $m_i + m_m$ unoccupied vertices other than P, thus $W(P') \leq m_i + m_m + W(P)$. By definition $\mathrm{dist}(p_i) \leq W(P')$, therefore $\mathrm{OPT}_{\mathrm{ConSum}} \geq \mathrm{dist}(p_i)$, as desired. □

Lemma 23. *There is a polynomial-time algorithm that connects the pebbles such that no one moves more than its distance.*

Proof. Define the weighted undirected graph H as follows. For each pebble p_i we make a vertex v_i in graph H. For each pair of pebbles p_i and p_j, find the minimum-weight path $P_{i,j}$, that connects them and has no pebble in its intermediate vertices. If such a path exists, we set the weight of the edge $e = \{v_i, v_j\}$ in H, $w(e) = W(P_{i,j})$. By dividing a path P of G between two pebbles, to some $P_{i,j}$s, you may find the corresponding path P' in H such that the weight of P is the same as the length of P'. By use of this correspondence the distance of a pebble p_i, is equal to the shortest path from v_i to v_m in H.

Run the Dijkstra algorithm on H from v_m. Consider the Dijkstra tree T. For each vertex v_i consider the shortest path from v_i to v_m in T and find the corresponding path P_{p_i} from p_i to p_m. Since P_{p_i} has the minimum unoccupied vertices, we have $W(P_{p_i}) = \mathrm{dist}(p_i)$. Sorting the pebbles according to their distance yields the order $p'_1, p'_2, \ldots, p'_m$. Use an incremental algorithm. At step l move the pebble p'_l through the path $P_{p'_l}$ until it reaches an occupied vertex on $P_{p'_l}$.

By induction after step l of the incremental algorithm the pebbles $p'_1, p'_2, \ldots, p'_l$ are connected, and at last all the pebbles are connected. Since the shortest path from v_i contains the shortest paths from its ancestors, P_{p_i} contains their paths. Thus each pebble p_i moves at most $W(P_{p_i})$ steps because the pebbles that their corresponding vertices are ancestors of v_i do not leave the P_{p_i} in the described algorithm. Since $W(P_{p_i}) = \mathrm{dist}(p_i)$, no pebble moves more than its distances. □

By applying the algorithm given in Lemma 23, the sum of the movements of the pebbles are at most $\sum_{i=1}^{m} \mathrm{dist}(p_i)$. Moreover, by applying Lemma 22, we have $\sum_{i=1}^{m} \mathrm{dist}(p_i) \leq m\mathrm{OPT}_{\mathrm{ConSum}}$.

5.3 Path Connectivity: $O(n)$ Approximation

Theorem 24. *The PathSum problem is NP-Hard.*

Proof. Apply the same construction as proof of Theorem 6. Then G has a Hamiltonian path if and only if PathSum has a solution of total movement $n(n+2)$. □

Theorem 25. *There is an $O(n)$-approximation algorithm for PathSum.*

Proof. Consider an instance S of PathSum problem with optimum solution OPT. Let k pebbles change their initial position in optimum solution. So $k \leq \mathrm{OPT}$. We consider the instance of PathNum defined by S. Let $\mathrm{OPT}_{\mathrm{PathNum}}$ be the optimal solution to this instance, that is, the minimum number of pebbles which must change their initial position to connect s and t. By moving these pebbles we can connect s and t with total movement of at most $n\mathrm{OPT}_{\mathrm{PathNum}}$. Because $\mathrm{OPT}_{\mathrm{PathNum}} \leq k \leq \mathrm{OPT}$. Therefore we have an $O(n)$-approximation algorithm for PathSum. □

5.4 Perfect Matchability.

Like MatchMax, the MatchSum variation can also be solved in polynomial time:

Theorem 26. *There is a polynomial-time algorithm solving MatchSum.*

Proof. As in Theorem 17, we assume without loss of generality that the graph G is connected and has an even number of pebbles, $p_1, p_2, \ldots, p_{2n}$. Define the weighted complete undirected graph H with vertex set $\{p_1, p_2, \ldots, p_{2n}\}$. For each edge $e = \{v_i, v_j\}$ in H, we set its weight $w(e)$ to $\max\{0, d_G(p_i, p_j) - 1\}$. Define the *total weight* $w(M) = \sum_{e \in M} w(e)$ of a perfect matching M of H to be the total weight of its edges.

Our algorithm computes a perfect matching M in H of minimum total weight $w(M)$ (in polynomial time), and converts this matching into a motion as follows. For each edge $e = \{v_i, v_j\}$ in the matching M, we move p_i by $\max\{0, d_G(p_i, p_j) - 1\} = w(e)$ steps toward p_j along a shortest path from p_i to p_j in G. Thus, after the motion, p_i and p_j are at distance at most 1 in G. The total movement in this motion is $\sum_{e \in M} w(e) = w(M)$.

Now we argue that no solution to MatchSum has total movement less than $w(M)$. By definition of MatchSum, any solution induces a perfect matching M' in the graph H (i.e., on the pebbles) with the property that, in the target configuration, every two matched pebbles have distance at most 1 in G. For every edge $e = \{v_i, v_j\}$ in this matching M', by Lemma 16, we have that $|\pi(p_i)| + |\pi(p_j)| \geq d_G(p_i, p_j) - 1$, and thus $|\pi(p_i)| + |\pi(p_j)| \geq \max\{0, d_G(p_i, p_j) - 1\} = w(e)$. Therefore, the total movement in the solution must be at least $\sum_{e \in M'} w(e) = w(M')$. But M was chosen to minimize this lower bound $w(M)$, so every solution must have total movement at least $w(M)$, proving optimality of our strategy of total movement $w(M)$. $\square$

6 *Minimizing the Number of Movements*

6.1 ConNum and DirConNum.

It is easy to see that ConNum (DirConNum) is at least as hard to approximate as the (directed) Steiner tree problem, and thus is $\Omega(\log n)$-inapproximable ($\Omega(\log^2 n)$-inapproximable). For consider an instance of (directed) Steiner tree consisting of a graph G and terminal set T. Attach a path of length $|V(G)|^2$ to each terminal in T (directed toward the terminal), and place one pebble on each vertex of each such path. Any $o(n)$-approximation must leave at least one pebble on each path, and thus the ConNum (DirConNum) problem becomes to find a minimum set of nonterminal vertices in G to connect together the terminals, which is exactly the (directed) Steiner tree problem.

The remainder of this section considers approximability of ConNum and DirConNum.

We can canonicalize the graph so that there is at most one pebble on each vertex. For otherwise, for each vertex v with $l > 1$ pebbles on it, we replace v with vertices $v_1, v_2, \ldots, v_l$. Then, for each $1 \leq i, j \leq l$, we put a directed edge from v_i to v_j. Also, for each incoming edge (u, v) in the previous graph, we put directed edges (u, v_i); and, for each edge outgoing (v, u) in the previous graph, we put directed edges (v_i, u), $1 \leq i \leq l$.

Definition Given a directed weighted graph G, a root r, an integer k, and a set $X \subseteq V(G)$ of terminals with $|X| \geq k$, the k-DSteiner(r, X) problem is to construct a tree rooted at r, spanning any k terminals in X, and of minimum weight.

The following lemma is proved by Charikar et al. [1999].

Lemma 27. [Charikar et al. 1999] *There is an $O(|X|^{\varepsilon})$-approximation for the* *k-DSteiner problem.*

Definition Given a directed graph G, a root r, and a subset X of *terminal vertices*. The *node-weighted directed k-Steiner tree problem*, k-DNSteiner, is to find a tree rooted at r that spans at least k terminals and has the minimum possible number of nonterminal vertices.

THEOREM 28. *There is an $O(|X|^{\varepsilon})$-approximation for k-DNSteiner.*

PROOF. Consider an instance of the k-DNSteiner problem. Given a graph G with a root r, an integer k, and a set $X \subseteq V(G)$ of terminals. We construct an instance G' of the k-DSteiner problem as follows. For each vertex $v \in V(G)$, we place two vertices v^- and v^+ in G' and put a directed edge from v^- to v^+. We put a directed edge from u^+ to v^- if and only if $(u, v) \in E(G)$. For each $v \notin X$, define the weight of edge (v^-, v^+) to be 1. Define the weight of all other edges to be 0. Specify vertex v^+ to be a terminal in the k-DSteiner instance if and only if v is a terminal in the k-DNSteiner instance.

For each solution of k-DSteiner(k, r^+, U) in G' with weight l, there is a solution to our instance of the k-DNSteiner problem with l nonterminal vertices and vice versa. Now applying Lemma 27, we obtain an $O(|X|^{\varepsilon})$-approximate solution for the k-DNSteiner instance. □

LEMMA 29. *Given a directed tree T rooted at r, let U be the set of occupied vertices. Then there is a solution for this instance of DirConNum whose cost (moved pebbles) is at most $|V_T - U|$.*

PROOF. We define a motion of pebbles by repeatedly performing the following step. Pick a pebble on vertex $v \neq r$ such that no descendant vertices of v in T are occupied by pebbles. Let $v, v_1, v_2, \ldots, v_j, r$ denote the path in T from v to r. We move a pebble on v to the first vertex v_i that is connected to r via other pebbles, that is, for which $v_{i+1}, v_{i+2}, \ldots, v_j$ are occupied by pebbles. Thus, the target position of pebble v is a vertex without a pebble, so it is in $V_T - U$. Therefore, the number of moved pebbles is at most $|V_T - U|$. □

THEOREM 30. *An f-approximation for k-DNSteiner implies an $(1 + f)$-approximation for DirConNum.*

PROOF. We want to find a $(1 + f)$-approximation for instance I of DirConNum with graph G. Let OPT denote the optimum solution for instance I. We make an instance of the k-DNSteiner problem from I with the same graph G. The set X of terminal vertices of I consists of the initial positions of the pebbles in I. Let k be $m - \mathrm{OPT}$ where m is the number of pebbles in instance I. The optimum solution for instance I is a solution for our instance of the k-DNSteiner problem with OPT nonterminal vertices. So the optimum solution for this instance has at most OPT nonterminal vertices. According to the f-approximation for the k-DNSteiner problem we can find a tree T with at most $f \cdot \mathrm{OPT}$ nonterminal vertices and that spans at least $m - \mathrm{OPT}$ terminal. Now using Lemma 29 and moving at most $f \cdot \mathrm{OPT}$ pebbles of these $m - \mathrm{OPT}$ pebbles, we can form these $m - \mathrm{OPT}$ pebbles into a tree T' rooted at r. We still have $m - (m - \mathrm{OPT}) = \mathrm{OPT}$ pebbles that are not present in tree T'. We move these OPT pebbles to r. Now all pebbles are present in tree T' and we have moved at most $f \cdot \mathrm{OPT} + \mathrm{OPT}$ pebbles. By testing all possible values of OPT, we have an $(1+f)$-approximation for the DirConNum problem. □

Corollary 31. *There is a polynomial-time $O(m^{\varepsilon})$-approximation algorithm for DirConNum.*

By applying Theorem 31 to all possible root vertices r, we obtain the same approximation factor for ConNum:

Corollary 32. *There is a polynomial-time $O(m^{\varepsilon})$-approximation algorithm for ConNum.*

6.2 PathNum

Theorem 33. *There is a polynomial-time algorithm that solves PathNum for a given graph G and two given vertices s and t in G.*

Proof. We use dynamic programming to find a path from s to t that has the minimum number of unoccupied vertices. For every vertex v in G and any integer l, $1 \leq l \leq |V(G)|$, let $C_{v,l}$ be the minimum number of unoccupied vertices that a path of length l from s to v could have. Then $C_{v,l}$ can be computed using $C_{u,l-1}$ where u iterates over all neighbors of v. At last we find the minimum of $C_{t,i}$ where i ranges from 0 to the total number of pebbles, which is the solution of the PathNum problem. □

6.3 IndNum.

Like IndMax, it is NP-hard to decide whether an IndNum instance has a valid solution: an instance has a solution precisely if the graph has an independent set of size m, the number of pebbles.

For case of the Euclidean plane, our problem is equivalent to finding a minimum vertex cover in unit-disk graphs: by moving the pebbles of minimum vertex cover to infinity, we reach an independent set of pebbles and unmoved pebbles in our problem must induce an independent set. Because minimum vertex cover has a PTAS in unit-disk graphs [Hunt et al. 1998], IndMax has a PTAS in the case of the Euclidean plane.

6.4 MatchNum.

MatchNum can be solved in polynomial time as follows. Find a maximum matching whose edges have both endpoints in occupied vertices. Moving an unmatched pebble near another unmatched pebble can increase the size of the matching by one, and we cannot get better by moving one. So moving unmatched pebbles one by one leads to a perfect matching with the minimum number of movements.

7 *Conclusion*

This article makes a systematic study of movement problems which, despite connections to several practical problems, have not been studied before in theoretical computer science. Among the problems we consider, we highlight one open problem of primary concern: the approximability of ConMax and PathMax. For directed graphs, we proved essentially tight approximability and inapproximability results for DirConMax, of roughly $\widetilde{\Theta}(n)$. However, for undirected graphs, we obtained an $O(1+\sqrt{m/\mathrm{OPT}}) = O(\sqrt{m})$-approximation for ConMax and PathMax. Can these approximations be improved, or are there matching inapproximability results? Figure 1 shows a difficult example which might be extended to prove an inapproximability result for both problems.

It would also be interesting to consider more problems in the practical scenario of the Euclidean plane, either for improved approximation ratios compared to general graphs or for problems that cannot be solved on general graphs. In particular, in the latter category, we obtained an additive $O(1)$-approximation for IndMax, but even the existence of a multiplicative $O(1)$-approximation for IndMax remains open. (Specifically, when $\mathrm{OPT} = o(1)$, we lack good approximations.)

Several other movement problems fit into our general framework. One variation changes the notion of the graph induced by a set of pebbles to include edges between pebbles within a given fixed distance d. This variation models the situation in which pebbles can communicate within a fixed distance d, but they still move one unit at a time (so we cannot simply take the dth power of the graph). Another variation, the *facility-location movement problem*, introduces two types of pebbles, namely clients and servers, and the target property is that every client is collocated with some server. If only the clients are permitted to move, this problem is trivial: each client moves to its nearest server. If both clients and servers can move, this solution is a 2-approximation to the maximum-movement version, but can we do better? What about the other versions of the problem, for example, total movement?

A *Proof of Inequality 1*

In this appendix, we prove Inequality 1 from the proof of Theorem 12:

$$\begin{aligned}
&\tfrac{1}{2}\left(\tfrac{2}{3}\arccos(a\sqrt{3}/2) - a\sqrt{\tfrac{1}{3}-\tfrac{1}{4}a^2} + \tfrac{2}{3}\arccos(b\sqrt{3}/2) - b\sqrt{\tfrac{1}{3}-\tfrac{1}{4}b^2}\right)\\
&\le \left(\frac{1}{3}-\frac{\sqrt{3}}{2\pi}\right)\arccos\left(\frac{a^2+b^2-1}{2ab}\right). \qquad (1)
\end{aligned}$$

First we note that the inequality is symmetric with respect to a and b. Let $g(a,b)$ and $f(a,b)$ be the left- and right-hand sides of the inequality, respectively. Define $h(x) = \frac{2}{3}\arccos(x\sqrt{3}/2) - x\sqrt{\frac{1}{3}-\frac{1}{4}x^2}$, so that $g(a,b) = \frac{1}{2}(h(a)+h(b))$.

We first prove the following lemma to show that it is sufficient to prove the inequality just for the boundary values of a and b.

LEMMA 34. *For sufficiently small $\varepsilon > 0$, if $1 \le a-\varepsilon$, $a \le b$, and $b+\varepsilon \le \frac{2}{\sqrt{3}}$, then we have $f(a-\varepsilon, b+\varepsilon) \le f(a,b)$ and $g(a-\varepsilon,b+\varepsilon) \ge g(a,b)$.*

PROOF. Because arccos is a decreasing function and $\frac{(a-\varepsilon)^2+(b+\varepsilon)^2-1}{2(a-\varepsilon)(b+\varepsilon)} > \frac{a^2+b^2-1}{2ab}$, we have $f(a-\varepsilon,b+\varepsilon) \le f(a,b)$.

For the other part, we must show that $h(a)-h(a-\varepsilon) \le h(b+\varepsilon)-h(b)$ or equivalently (because ε tends to zero), $h'(a) \le h'(b)$. It is sufficient to prove that

$h''(x) \geq 0$ for any $1 \leq x \leq \frac{2}{\sqrt{3}}$.

$$\begin{aligned} h'(x) &= \tfrac{2}{3} \cdot \frac{-\frac{1}{2}\sqrt{3}}{\sqrt{1-\frac{3}{4}x^2}} - \sqrt{\tfrac{1}{3}-\tfrac{1}{4}x^2} - x\frac{-\frac{1}{2}x}{2\sqrt{\frac{1}{3}-\frac{1}{4}x^2}} \\ &= \frac{1}{\sqrt{\frac{1}{3}-\frac{1}{4}x^2}}(-\tfrac{1}{3}-\tfrac{1}{3}+\tfrac{1}{4}x^2+\tfrac{1}{4}x^2) = -2\sqrt{\tfrac{1}{3}-\tfrac{1}{4}x^2} \end{aligned} \tag{2}$$

Now, we have $h''(x) = a \Big/ \left(2\sqrt{\frac{1}{3}-\frac{1}{4}a^2}\right)$, which is clearly nonnegative. □

According to Lemma 34 and the mentioned symmetry of Inequality 1, it remains to prove the inequality just for $b = 1$ and $b = \frac{2}{\sqrt{3}}$.

First suppose that $b = \frac{2}{\sqrt{3}}$. Then we have

$$f(a,b) = \left(\tfrac{1}{3} - \tfrac{\sqrt{3}}{2\pi}\right) \arccos \frac{a^2+\frac{1}{3}}{4a/\sqrt{3}}.$$

Because $\frac{a^2+1/3}{4a/\sqrt{3}} \leq \frac{5}{8}$ and arccos is a decreasing function, $f(a,b) \geq (\frac{1}{3} - \frac{\sqrt{3}}{2\pi}) \arccos \frac{5}{8}$. On the other hand, we have

$$g(a,b) = \tfrac{1}{2}\left(\tfrac{2}{3}\arccos(a\sqrt{3}/2) - a\sqrt{\tfrac{1}{3}-\tfrac{1}{4}a^2}\right) = \tfrac{1}{2}h(a).$$

By Equation 2, function $h(x)$ reaches its maximum at $x = 1$, where $h(1) = \frac{\pi}{9} - \frac{1}{2\sqrt{3}}$. Thus,

$$g(a,b) \leq \tfrac{1}{2}\left(\tfrac{\pi}{9} - \tfrac{1}{2\sqrt{3}}\right) \leq \left(\tfrac{1}{3} - \tfrac{\sqrt{3}}{2\pi}\right)\arccos\tfrac{5}{8} \leq f(a,b).$$

In the other case, $b = 1$. Here we have

$$\begin{aligned} f(a,b) &= \left(\tfrac{1}{3} - \tfrac{\sqrt{3}}{2\pi}\right)\arccos\tfrac{1}{2}a; \quad \text{and} \\ g(a,b) &= \tfrac{1}{2}\left(\tfrac{2}{3}\arccos(a\sqrt{3}/2) - a\sqrt{\tfrac{1}{3}-\tfrac{a^2}{4}} + \left(\tfrac{\pi}{9} - \tfrac{1}{2\sqrt{3}}\right)\right). \end{aligned}$$

The function $f(a,b)$ reaches its minimum at $a = \frac{2}{\sqrt{3}}$, where $f(a,b) = \left(\frac{1}{3} - \frac{\sqrt{3}}{2\pi}\right)\arccos\frac{1}{\sqrt{3}}$. Similar to the previous case,

$$g(a,b) \leq \tfrac{1}{2}\left(\tfrac{\pi}{9} - \tfrac{1}{2\sqrt{3}} + \tfrac{\pi}{9} - \tfrac{1}{2\sqrt{3}}\right) = \tfrac{\pi}{9} - \tfrac{1}{2\sqrt{3}} \leq \left(\tfrac{1}{3} - \tfrac{\sqrt{3}}{2\pi}\right)\arccos\tfrac{1}{\sqrt{3}} \leq f(a,b).$$

This inequality completes the proof of Inequality 1.

Acknowledgments

We thank the anonymous referees for helpful comments on this article.

REFERENCES

Arkin, E. M., Bender, M. A., Fekete, S. P., Mitchell, J. S. B., and Skutella, M. 2002. The freeze-tag problem: how to wake up a swarm of robots. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, California, 568–577.

Arkin, E. M., Bender, M. A., and Ge, D. 2003. Improved approximation algorithms for the freeze-tag problem. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms and Architectures*. San Diego, California, USA, 295–303.

Bredin, J. L., Demaine, E. D., Hajiaghayi, M., and Rus, D. 2005. Deploying sensor networks with guaranteed capacity and fault tolerance. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2005)*. Urbana-Champaign, Illinois, 309–319.

Charikar, M., Chekuri, C., Cheung, T.-Y., Dai, Z., Goel, A., Guha, S., and Li, M. 1999. Approximation algorithms for directed steiner problems. *Journal of Algorithms 33,* 1, 73–91.

Cohen, J. 1998. Broadcasting, multicasting and gossiping in trees under the all-port line model. In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures*. Puerto Vallarta, Mexico, 164–171.

Cohen, J., Fraigniaud, P., König, J.-C., and Raspaud, A. 1998. Optimized broadcasting and multicasting protocols in cut-through routed networks. *IEEE Transactions on Parallel and Distributed Systems 9,* 8, 788–802.

Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., and Sukhatme, G. 2004a. Autonomous deployment of a sensor network using an unmanned aerial vehicle. In *Proceedings of the 2004 International Conference on Robotics and Automation*. New Orleans, USA.

Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., and Sukhatme, G. 2004b. Deployment and connectivity repair of a sensor net with a flying robot. In *Proceedings of the 9th International Symposium on Experimental Robotics*. Singapore.

Doddi, S., Marathe, M. V., Mirzaian, A., Moret, B. M. E., and Zhu, B. 1997. Map labeling and its generalizations. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*. New Orleans, LA, 148–157.

Ghodsi, M., Hajiaghayi, M., Mahdian, M., and Mirrokni, V. S. 2002. Length-constrained path-matchings in graphs. *Networks 39,* 4, 210–215.

Guha, S. and Khuller, S. 1999. Improved methods for approximating node weighted steiner trees and connected dominating sets. *Information and Computation 150,* 1, 57–74.

Hsiang, T.-R., Arkin, E. M., Bender, M. A., Fekete, S. P., and Mitchell, J. S. B. 2003. Algorithms for rapidly dispersing robot swarms in unknown environments. In *Algorithmic Foundations of Robotics V*, J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds. Springer Tracts in Advanced Robotics, vol. 7. Springer-Verlag, 77–94.

Hunt, III, H. B., Marathe, M. V., Radhakrishnan, V., Ravi, S. S., Rosenkrantz, D. J., and Stearns, R. E. 1998. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *Journal of Algorithms 26,* 2, 238–274.

Jiang, M., Bereg, S., Qin, Z., and Zhu, B. 2004. New bounds on map labeling with circular labels. In *Proceedings of the 15th International Symposium on Algorithms and Computation*. Lecture Notes in Computer Science, vol. 3341. Hong Kong, China, 606–617.

Jiang, M., Qian, J., Qin, Z., Zhu, B., and Cimikowski, R. 2003. A simple factor-3 approximation for labeling points with circles. *Information Processing Letters 87,* 2, 101–105.

Klein, P. N. and Ravi, R. 1995. A nearly best-possible approximation algorithm for node-weighted steiner trees. *Journal of Algorithms 19,* 1, 104–115.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge University Press. http://msl.cs.uiuc.edu/planning/.

Reif, J. H. and Wang, H. 1995. Social potential fields: a distributed behavioral control for autonomous robots. In *Proceedings of the Workshop on Algorithmic Foundations of Robotics*. 331–345.

Schultz, A. C., Parker, L. E., and Schneider, F. E., Eds. 2003. *Multi-Robot Systems: From Swarms to Intelligent Automata*. Springer. Proceedings from the 2003 International Workshop on Multi-Robot Systems.

Strijk, T. and Wolff, A. 2001. Labeling points with circles. *International Journal of Computational Geometry & Applications 11,* 2, 181–195.

Sztainberg, M. O., Arkin, E. M., Bender, M. A., and Mitchell, J. S. B. 2004. Theoretical and experimental analysis of heuristics for the "freeze-tag" robot awakening problem. *IEEE Transactions on Robotics and Automation 20,* 4, 691–701.

West, D. B. 2001. *Introduction to Graph Theory*, Second ed. Prentice Hall Inc., Upper Saddle River, NJ.

RECEIVED FEBRUARY 2007; REVISED DECEMBER 2007; ACCEPTED MARCH 2008