

Clearing Paths with Minimum Movement

Daniel Apon
Anu Bandi

December 7, 2011

1 Lecture Notes

1.1 Introduction to Movement Problems

Consider a scenario where a group of automated robots with limited mobility, energy, and so on need to reorganize their joint position in order to form a reliable radio network connecting two locations. Since resources are scarce, we wish to minimize the movement required to establish the radio network. This, in turn, can take many forms. Do we want to minimize the total movement of all robots? Do we want to minimize the maximum movement of any single robot?

A striking feature of such movement-minimization problems is that many are generalizations of relatively simple, polynomial-time solvable problems, and yet when movement is introduced, the problem is suddenly NP-hard. The above example, minus the notion of pebbles and their movement, is nothing more than graph reachability and can be solved by simply computing the shortest path with Dijkstra's algorithm. However, it's not hard to see (at least intuitively) that deciding which pebbles go where along the path induces the type of combinatorial explosion traditionally associated with computational intractability.

1.1.1 Problems Studied in the Literature

We formally define the model used in the spirit of [2]. Unless otherwise stated, we assume that we are given a graph $G = (V, E)$ with $|V| = n$ vertices and m pebbles, with each pebble assigned to some (not necessarily distinct) vertex according to some *initial configuration*. We say any vertex with a pebble assigned to it is *occupied*. Further, we allow more than one pebble to be assigned to a single vertex. Moving a pebble involves assigning a path $\pi(p)$ over edges in the graph G for each pebble p , beginning with the vertex p occupies in the initial configuration and ending at some *target vertex*. We say that $|\pi(p)|$ is the *movement* of p . The set of target vertices and their associated pebbles form the *target configuration*. The goal is to minimize movement to achieve a given graph property according to some complexity measure, as detailed below.

In order to specify a particular computational problem in the class of movement problems, we give a pair (P, C) , where P is some graph property we are interested in and C is a complexity measure associated with any feasible solution. In [2], the authors consider five different properties:

1. CON (graph connectivity) – we want to minimize pebble movement such the graph induced by the pebbles’ final position is connected,
2. DIRCON (directed graph connectivity) – identical to CON, but G is directed,
3. PATH (s - t connectivity) – we want the pebbles to induce a path between two specified vertices $s, t \in G$,
4. IND (independent set) – we want no two pebbles to be assigned to the same vertex or adjacent vertices, and
5. MATCH (perfect matching) – we want a perfect matching of pebbles by grouping into adjacent pairs, where two pebbles are *adjacent* (for purposes of MATCH) *iff* their distance $d_G(p, q)$ in G is at most 1.

Similarly, the authors consider three different complexity measures:

1. MAX – we want to minimize the maximum movement of any single pebble required to achieve P ,
2. SUM – we want to minimize the total movement of all pebbles required to achieve P , and
3. NUM – we want to minimize the number of pebbles moved to achieve P .

This gives rise to a host of natural problems (namely, every combination of P and C above). For instance, we have CONMAX, CONSUM, CONNUM, DIRCONMAX, and so on.

1.1.2 Results of Demaine, et al

In Table 1, we summarize the results obtained in [2].

1.1.3 Example Result

Theorem. *Given a tree T and a configuration of k pebbles on T , CONMAX can be solved in polynomial time.*

Proof Sketch. To begin, we guess a vertex v of T occupied in the target configuration and the maximum movement k , $0 \leq k \leq n$ of OPT.

Then, we compute a set of *forced vertices* – or vertices that must be occupied in the final solution – in the following manner: For every pebble p , move it k steps along the unique path toward v (stopping if v is reached). Denote by x_p the final location of pebble p under such an operation. Then it must be the

	MAX	SUM	NUM
CON	$O(\sqrt{m/\text{OPT}})$	$O(\min\{n, m\}),$ $\Omega(n^{1-\epsilon})$	$O(m^\epsilon), \Omega(\log n)$
DIRCON	$\epsilon m, \Omega(n^{1-\epsilon})$	open	$O(m^\epsilon),$ $\Omega(\log^2 n)$
PATH	$O(\sqrt{m/\text{OPT}})$	$O(n)$	polynomial
IND	$1 + \frac{1}{\sqrt{3}}$ additive in \mathbb{R}^2	open	PTAS in \mathbb{R}^2
MATCH	polynomial	polynomial	polynomial

Table 1: Summary of results of Demaine, et al.

case that the vertices between x_p and v are occupied in the target configuration. Otherwise, the solution would not be feasible, for whichever p is chosen.

Once a set of forced vertices is found, define the bipartite graph $H = (U, V, E)$, where U is the set of pebbles, V is the set of forced vertices, and E is composed of edges that connect p to every forced vertex within k steps. Then a maximum-cardinality matching of H covers every $v \in V$ iff the pebbles can be moved to occupy the forced vertices. \square

1.2 Our Problem Definitions

For all problems we consider, we are given an undirected graph $G = (V, E)$ with $|V| = n$ vertices, m pebbles, and a property on configurations of pebbles over the vertices of G we are interested in satisfying. We augment the problems (where possible) with coincident-cost functions $\text{cost}_v(i) = \alpha_v(i - 1)$ – i.e. the second pebble to arrive at a vertex v pays α_v to be coincident; the third pebble pays $2\alpha_v$, and so on.

— CLEARMAX —

Input: Two specified vertices s, t , and a coincident-cost function cost_v for every $v \in G$.

Output: A prescribed set of moves for the pebbles such that a path from $s - t$ is totally unoccupied by pebbles afterward, and such that *maximum sum of movement plus induced coincident-cost of any single pebble* is minimized. That is,

$$\min \left\{ \max_{p_i: i \in [m]} \left\{ \sum_{(u,v) \in E: u, v \in \mathbb{P}_i} w(u, v) + \text{cost}_{v'_i}(i_{p_i, v}) \right\} \right\}$$

where $w(u, v)$ denotes the weight of edge (u, v) , where v'_i is the target vertex of p_i 's movement, and where $i_{p_i, v}$ is the index of pebble p_i with respect to the coincident-cost assignment of v .

— CLEARSUM —

Input: Two specified vertices s, t , and a coincident-cost function cost_v for every $v \in G$.

Output: A prescribed set of moves for the pebbles such that a path from $s-t$ is totally unoccupied by pebbles afterward, and such that *total sum* of movement plus induced coincident-cost *over all pebbles* is minimized. That is,

$$\min \left\{ \sum_{p_i: i \in [m]} \left(\sum_{(u,v) \in E: u, v \in \mathbb{P}_i} w(u, v) + \text{cost}_{v'_i}(i_{p_i, v}) \right) \right\}$$

where the $w(u, v)$, v'_i , and $i_{p_i, v}$ are as above.

— CLEARNUM —

Input: Two specified vertices s, t .

Output: A prescribed set of moves for the pebbles such that a path from $s-t$ is totally unoccupied by pebbles afterward, and such that *number* of moved pebbles is minimized.

1.3 Our Results

In the full version, we prove the following:

Theorem. CLEARSUM is NP-hard.

Theorem (1). CLEARMAX is NP-hard.

Theorem (2). CLEARMAX on trees without coincident-cost functions is polynomial-time solvable.

Theorem. There is a polytime, $O(\log n)$ -approx. algorithm for CLEARMAX (on general graphs and with coincident-cost functions).

Theorem (3). CLEARNUM is polynomial-time solvable.

Due to space concerns, we only provide proofs in what follows for the theorems labeled with numbers above.

1.3.1 Hardness of CLEARMAX

Proof of Theorem (1). We reduce HAMILTONIAN PATH to CLEARMAX. The proof is essentially a proof by picture; refer to Figure 1 for the following.

On input a graph G from an instance of HAMILTONIAN PATH: We begin by constructing n^2 vertices in a square. In particular, there are n levels (or rows) of vertices, each composed of the n distinct vertices $v_1, v_2, \dots, v_n \in V$. We then construct a vertex labeled s above the grid, and a vertex labeled t below.

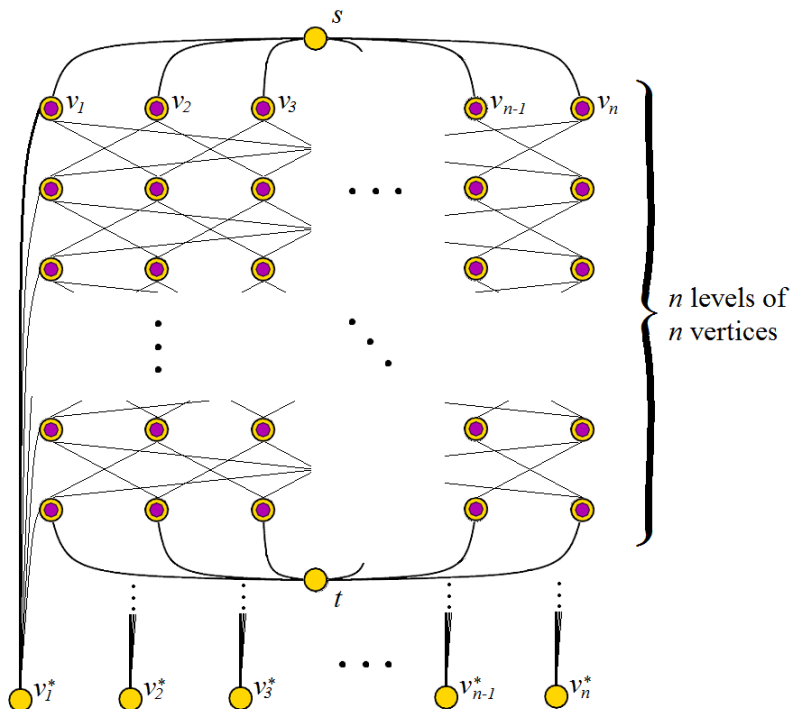


Figure 1: Gadget for CLEARSUM and CLEARMAX reductions

Finally, we construct n special vertices v_1^*, \dots, v_n^* that we will associate with the n corresponding length- n columns of v_1 's, \dots , v_n 's.

Returning to the grid, we use the adjacency matrix of G to construct corresponding edges between vertices in each level of the grid. We then connect every vertex in the top level to s , and every vertex in the bottom level to t . Finally, we connect all vertices in each column i to their special vertex v_i^* .

To wrap up, we place n^2 pebbles on the grid vertices, which leaves precisely $n + 2$ empty spaces on the grid – the n special vertices, s , and t . Finally, we give every edge adjacent to a special vertex a weight of n and every other edge a weight of n^3 . We also set $\alpha_v := \infty, \forall v$ so that no pebbles may be coincident.

By construction, there is a solution to the CLEARMAX instance with maximum cost for any pebble of n if and only if there is a Hamiltonian path in G . \square

1.3.2 CLEARMAX, no coincident-cost functions, on trees

Proof of Theorem (2). Given some tree T , WLOG, assume s is a leaf of the tree and t is the root of the tree, or vice versa. Notably, we make the assumption

that no pebbles may move through s or t (after all, we are trying to clear a path in which, presumably, objects at s and/or t would be moving!). Then it's not hard to see that we can always prune the tree so that this is the case (up to an arbitrary labeling of whether s or t is the root or the leaf, respectively). In particular, since there is a unique, valid path from u to v , for $u, v \in T$ by definition of a tree, we can **completely ignore** any vertices v and edges e such that

1. $v \neq s$ and/or e are contained in a subtree T_s of T , rooted at s , and
2. $v \neq t$ and/or e are not contained in the subtree $T_{t'}$, rooted at t' , where t' is t 's child in the direction of s .

Moreover, by the definition of coincident-cost functions, we can actually do (much) better. In particular, consider the path from s to t in T . By construction, this path is a sequence of parent vertices, each of the previous vertices in the sequence respectively. The only adjacent vertices that are eligible target vertices are the children of the vertices intermediate in the path from s to t (but not including s or t 's distinct children, by the above assumption). However, we can also rule out of consideration as a target vertex any vertex v where v is neither in the $s - t$ path or an immediate child of any vertex $v' \neq s, t$ in the path.

That is, let v_1 be in the path, v_2 be a child of v_1 not in the path, and v_3 be a child of v_2 , and consider moving a pebble p from its starting vertex to v_3 . Upon arriving at v_2 , it's easy to see that there is no motivation to pay the additional movement cost of $w(v_2, v_3)$ to move p to v_3 . On the one hand, p is already off the path (and so is at a valid, final location), and on the other, there can be no savings in cost by moving again (since p has already incurred the coincident-cost associated with v_2 for moving "through" it, as v_2 is not on the $s - t$ path. Therefore, we might decide to move pebbles to vertices like v_2 , but will never move pebbles to vertices like v_3 or elsewhere further away from the $s - t$ path.

Finally since there are no coincident-cost functions, the movement of the pebbles are independent of each other (namely, the starting vertex, chosen path, and chosen target vertex of some pebble p_i has no influence on the choices made for another pebble p_j), so we perform a simple greedy search at each vertex $v \in \mathbb{P}$ to find the closest $v' \notin \mathbb{P}$.

This gives a polynomial-time algorithm for the relaxation of CLEARMAX without coincident-cost functions, giving Theorem 3. \square

1.3.3 Algorithm for CLEARNUM

Proof of Theorem 5. On input an instance of CLEARNUM, color the given pebbles black, and then place a white pebble on every unoccupied vertex in G . Then run the PATHNUM dynamic programming algorithm of [2] (treating black pebbles as empty spaces) to find the path \mathbb{P} from s to t with the minimum number of vertices *unoccupied* by white pebbles. Then \mathbb{P} also has the minimum number of vertices *occupied* by black pebbles. Therefore, \mathbb{P} is the optimal solution to the CLEARNUM instance. \square

References

- [1] Piotr Berman, Erik D. Demaine, and Morteza Zadimoghaddam. $O(1)$ -approximations for maximum movement problems. In *Proceedings of the 14th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2011)*, pages 62–74, Princeton, New Jersey, August 17–19 2011.
- [2] Erik D. Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, Amin S. Sayedi-Roshkhar, Shayan Oveisgharan, and Morteza Zadimoghaddam. Minimizing movement. *ACM Transactions on Algorithms*, 5(3):Article 30, July 2009.
- [3] Erik D. Demaine, MohammadTaghi Hajiaghayi, and Dániel Marx. Minimizing movement: Fixed-parameter tractability. In *Proceedings of the 17th Annual European Symposium on Algorithms (ESA 2009)*, volume 5757 of *Lecture Notes in Computer Science*, pages 718–729, Copenhagen, Denmark, September 7–9 2009.
- [4] Adrian Dumitrescu and Minghui Jiang. Constrained k -center and movement to independence. *Discrete Appl. Math.*, 159:859–865, April 2011.
- [5] Zachary Friggstad and Mohammad R. Salavatipour. Minimizing movement in mobile facility location problems. *ACM Trans. Algorithms*, 7:28:1–28:22, July 2011.