Project Report for CMSC 858F (*Previous Work*)*

Rajesh Chitnis M. Reza Khani Vahid Liaghat

December 7, 2011

Abstract

In this report we give a brief view of some tractability and intractability results in the framework of *movement* problems. First we introduce the framework of Demaine et al. (SODA '07). We then see that how some polynomial time solvable problems become computationally hard if we introduce the concept of *movement*. There are two parameters : *property* desired in final configuration and *efficiency criteria* which we want to minimize. We give proofs the following results :

1. ConMax is NP-complete (even to approximate better than 2)
2. PathMax and PathSum are NP-hard
3. FacilityLocationMax has a 2-approximation algorithm.
4. FacilityLocationMax has no $(2-\varepsilon)$-approximation algorithm unless P=NP.
5. There is no constant $c > 0$ such that FacilityLocationMax has an additive c-approximation algorithm unless P=NP.

*Department of Computer Science , University of Maryland at College Park, USA. Email: {rchitnis, khani, vliaghat}@cs.umd.edu

1 Introduction

In many real-life scenarios we must plan the *coordinated motion* of *mobile agents* in order to carry out certain tasks. Examples are SWAT teams responding to emergency situations, firefighters responding to fires, organizing behaviour of swarms of robots [6], etc. In practice the number of agents is small and the terrain they must cover is very large. We desire the *most efficient* way for the agents to achieve the final desired configuration(s). Efficiency can be measured by various parameters like time taken, energy spent, etc. Also we can consider various *properties* for the choice of desired final configuration(s). Henceforth we will denote agents by pebbles.

Let us formalize the model now. In general, a *movement minimization* problem is defined by a set of final configuration(s) that we desire be formed by the pebbles and a movement objective function. The following examples are mentioned in Demaine et al. [2] who also gave some FPT results for various problems parameterized by the number of pebbles:

- The pebbles must move to form a connected subgraph so that the network is fault-tolerant.
- The pebbles must move to form an independent set. This has applications in map-labeling [3, 4].
- The pebbles must arrange themselves in a certain topological formation such as a grid which has a robust structure.
- The pebbles must augment an existing immobile structure to achieve a certain desired property [1].

2 Framework of Demaine et al. (SODA '07)

Demaine et al. [3] gave the first systematic study of problems in the framework of *movement*. They considered the following *efficiency criteria*:

1. **Max**: minimize the maximum movement of any pebble.
2. **Sum**: minimize the total movement of all pebbles.
3. **Num**: minimize the number of pebbles that move.

They considered the following *properties* desired in final configuration:

1. **Con**: the subgraph induced by the pebbles is connected.
2. **DirCon**: there is a directed path of pebbles from each vertex to some fixed root.
3. **Path**: there is a $s-t$ path of pebbles for some fixed vertices s and t.
4. **Ind**: the subgraph induced by the pebbles has no edges.
5. **Match**: the subgraph induced by the pebbles has a perfect matching.

Subsequent Work: Berman, Demaine and Zadimoghaddam [APPROX '11] gave constant-factor approximation algorithms for ConMax and PathMax (and some other new problems).

3 NP-hardness of ConMax

In this section we show the NP-hardness of "minimizing the maximum movement" version of the problem when we want the pebbles to induce a connected subgraph. We formally define the problems below.

CONMAX
Input : An undirected graph G, pebbles and their initial location on vertices of G.
Output : A final configuration of pebbles minimizing the maximum movement of any pebbles such that the subgraph induced by the pebbles is connected.

We reduce from Hamiltonian Path. Consider an instance $G = (V,E)$ of Hamiltonian Path where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2 \ldots, e_m\}$. We build an instance $G' = (V', E')$ of ConMax as shown in Figure:

	Max	Sum	Num
Con	$O(\sqrt{m/\mathrm{OPT}})$	$\tilde{O}(\min\{n,m\})$ $\Omega(n^{1-\varepsilon})$	$O(m^{\varepsilon})$ $\Omega(\log n)$
Path	$O(\sqrt{m/\mathrm{OPT}})$	$O(n)$	polynomial
DirCon	εm $\Omega(n^{1-\varepsilon})$	open	$O(m^{\varepsilon})$ $\Omega(\log^2 n)$
Ind	$1+\frac{1}{\sqrt{3}}$ additive in $\mathbb{R}^2$	open	PTAS in $\mathbb{R}^2$
Match	polynomial	polynomial	polynomial

Figure 1: Results of Demaine et al. (SODA '07)

1. Split each edge into a path of length 1. If $u-v$ is an edge we add two vertices $u'v$ and uv' in between with $u'v$ closer to u than v.
2. Attach a leaf β_i to v_i for each $i \in [n]$
3. For each $i \in [n]$ place two pebbles at v_i and one pebble at β_i

Theorem 3.1. *ConMax is NP-complete (even to approximate better than factor 2).*

Proof. Refer to Figure 2. We claim that G has a Hamiltonian path if and only if ConMax has a solution of cost at most 1. This claim proves our theorem since any solution for ConMax is integral and hence a $(2-\varepsilon)$ approximation for a solution of cost 1 would tell us that it is in fact 1 and we will be able to solve the Hamiltonian Path problem in polynomial time.

Suppose G has a Hamiltonian path say $v_{i_1} - v_{i_2} - \ldots - v_{i_n}$. For each $j \in [n]$ shift the pebble from β_j to v_j. For each $k \in [n-1]$ shift 1 pebble from v_{i_k} to $v'_{i_k}v_{i_{k+1}}$ and 1 pebble from $v_{i_{k+1}}$ to $v_{i_k}v'_{i_{k+1}}$. This clearly gives a solution of ConMax of cost 1.

Suppose ConMax has a solution of cost n. WLOG we can assume that each β_i shifts its pebble to v_i since each pebble can move at most 1 and this move only helps in the connectivity of induced subgraph. Note that each v_i has 3 pebbles now out of which 1 cannot be moved any more. Therefore all vertices of V must be connected in the subgraph induced by the pebbles. So there is a tree T containing all vertices of V. No vertex of V has degree ≥ 3 in this tree as that would imply that this vertex gave away all his 3 pebbles but this was not possible as the pebble which came from its leaf cannot move any further. So we have a Hamiltonian Path in G. □

4 NP-hardness of Path Problems

In this section we show the NP-hardness of two versions of $s-t$ path problem in the movement setting. We formally define the problems below.

PATHMAX
Input : An undirected graph G, pebbles and their initial location on vertices of G.
Output : A final configuration of pebbles minimizing the maximum movement of any pebbles such that there is a $s-t$ path which only contains vertices with pebbles.

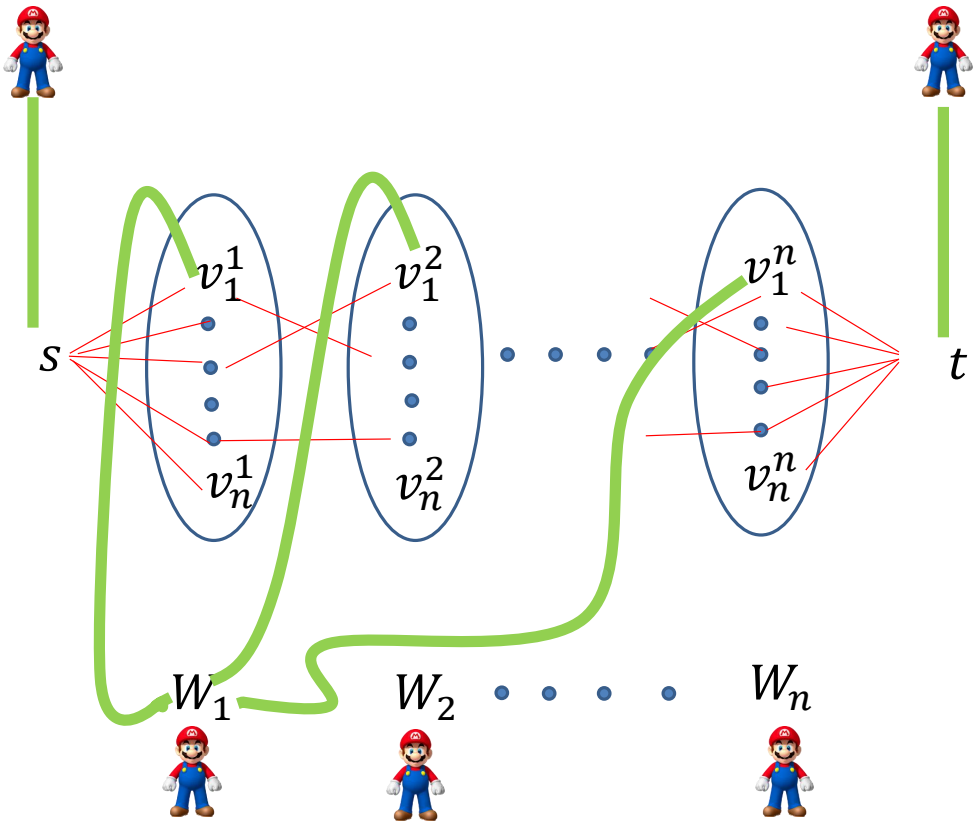

Figure 2: Path problems are NP-hard : Red paths are length 1 and green paths are length n

PATHSUM
Input : An undirected graph G, pebbles and their initial location on vertices of G.
Output : A final configuration of pebbles minimizing the total movement of all pebbles such that there is a $s-t$ path which only contains vertices with pebbles.

4.1 PathMax is NP-hard

We reduce from Hamiltonian Path. Consider an instance $G=(V,E)$ of Hamiltonian Path where $V=\{v_1,v_2,\ldots,v_n\}$ and $E=\{e_1,e_2\ldots,e_m\}$. We build an instance $G'=(V',E')$ of PathMax as shown in Figure 2:

1. For each $i\in[n]$ define $V_i=\{v_1^i,v_2^i,\ldots,v_n^i\}$
2. Add two new vertices s and t.
3. For each $i\in[n]$ add a vertex W_i.
4. For each $j\in[n]$ add paths of length n from W_j to $v_j^1,v_j^2,\ldots,v_j^n$.
5. From each of s and t add a path of length n.
6. Connect s to each vertex in V_1 and t to each vertex in V_n.
7. Between two consecutive layers V_i and V_{i+1} add an edge between v_j^i and v_k^{i+1} if and only if there is and edge between v_j and v_k in G

Theorem 4.1. *PathMax is NP-hard.*

Proof. Refer to Figure 2. We claim that G has a Hamiltonian path if and only if PathMax has a solution of cost at most n.

Suppose G has a Hamiltonian path say $v_{i_1} - v_{i_2} - \ldots - v_{i_n}$. Then we give a solution of PathMax of cost n as follows: Move the pebble from the path of length n adjacent to s to s. Similarly for t. Now move pebble at W_j to v_{i_j} for each $j \in [n]$. This gives a pebbled $s-t$ path with each pebble moving exactly n.

Suppose PathMax has a solution of cost n. Without loss of generality we can assume that the pebbles on the paths of length n attached to s and t are moved to s and t respectively since we are allowed movement of at most n and these two pebbles cannot do anything else. Now each W_i can move only on one path of length n. Also each W_i must move a path of length n to a vertex in some V_j as we have only n W-vertices and any pebbled $s-t$ path must have length at least n. So for each $j \in [n]$ let W_{i_j} be the vertex whose pebbled moved to a vertex in V_j. Then $v_{i_1} - v_{i_2} - \ldots - v_{i_n}$ form a Hamiltonian path in G and we are done. □

4.2 PathSum is NP-hard

We again reduce from Hamiltonian Path. We build the same instance for PathSum that we had built for PathMax in previous section.

Theorem 4.2. *PathSum is NP-hard.*

Proof. Refer to Figure 2. We claim that G has a Hamiltonian path if and only if PathSum has a solution of cost at most $n(n+2)$.

Suppose G has a Hamiltonian path say $v_{i_1} - v_{i_2} - \ldots - v_{i_n}$. Then we give a solution of PathMax of cost n as follows: Move the pebble from the path of length n adjacent to s to s. Similarly for t. Now move pebble at W_j to v_{i_j} for each $j \in [n]$. This gives a pebbled $s-t$ path with each pebble moving exactly n for a total cost of $n(n+2)$ since we had $(n+2)$ pebbles.

Suppose PathSum has a solution of cost $n(n+2)$. The pebbles on the paths of length n attached to s and t must move to s and t respectively since any pebbled $s-t$ path needs $(n+2)$ pebbles. This takes up $2n$ from the budget leaving n^2. Now each W_i must move a path of length n to a vertex in some V_j as we have only n W-vertices and any pebbled $s-t$ path must have length at least n. This takes up all the budget and hence this is the only scenario possible. So for each $j \in [n]$ let W_{i_j} be the vertex whose pebbled moved to a vertex in V_j. Then $v_{i_1} - v_{i_2} - \ldots - v_{i_n}$ form a Hamiltonian path in G and we are done. □

5 Facility Location with Movement

In the movement variant of the facility location problem we have two types of pebbles: *client* and *server*. We are given an initial location of the pebbles on vertices of an unweighted undirected graph. We allow both clients and servers to move and the final desired configuration is that each client be co-located with some server. Both client and server pebbles are allowed to move. We formally define the "minimizing the maximum movement" version of the facility location problem:

FACILITYLOCATIONMAX
Input : An undirected graph G, client and server pebbles and their initial location on vertices of G.
Output : A final configuration of pebbles minimizing the maximum movement of any pebble such that each client pebble is co-located with some server pebble.

5.1 2-approximation for FacilityLocationMax

We do not allow the servers to move. As only clients can move, the problem becomes trivial as each client just moves to his nearest server. We claim this is gives a 2-approximation for FacilityLocationMax.

Lemma 5.1. *The above algorithm gives a 2-approximation for the FacilityLocationMax problem.*

Proof. Let OPT be the optimum when both kinds of pebbles can be moved. Therefore no pebble moves more than OPT distance to achieve a final configuration, i.e., each client pebble is at a distance of at most 2.OPT from some server pebble which proves our claim. □

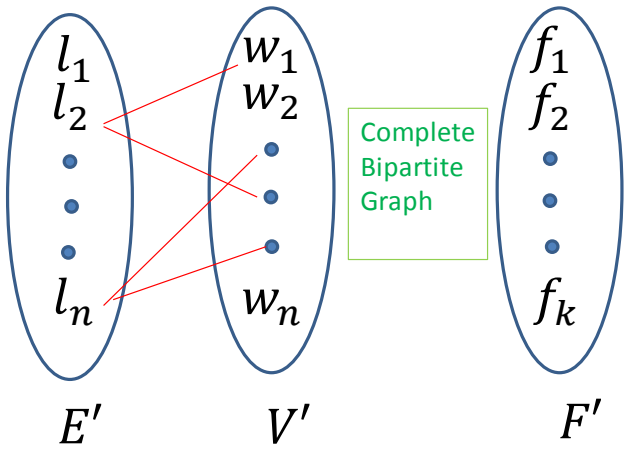

Figure 3: 2-approximation is tight for FacilityLocationMax

5.2 Tightness of 2-approximation for FacilityLocationMax

Friggstad and Salavatipour [5] showed that the FacilityLocationMax problem cannot have an approximation ratio better than 2. We reduce from Vertex Cover. Consider an instance $G = (V, E)$ of Vertex Cover where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2 \ldots, e_m\}$. We build an instance G' of FacilityLocationMax as shown in Figure 3.

1. $V' = \{w_1, w_2, \ldots, w_n\}$
2. $E' = \{\ell_1, \ell_2, \ldots, \ell_m\}$
3. $F' = \{f_1, f_2, \ldots, f_k\}$
4. Put an edge between w_i and ℓ_j if and only if v_i and e_j are incident in G.
5. Put a complete bipartite graph between V' and F'.
6. Facilities = F'
7. Clients = E'

Theorem 5.2. *There is no $(2-\varepsilon)$-approximation for FacilityLocationMax unless P=NP.*

Proof. We refer to Figure 3. Claim is G has a vertex cover of size at most k if and only if FacilityLocationMax has a solution of cost 1. This claim proves our theorem since any solution for FacilityLocationMax is integral and hence a $(2-\varepsilon)$ approximation for a solution of cost 1 would tell us that it is in fact 1 and we will hence be able to solve Vertex Cover problem in polynomial time.

Suppose G has a vertex cover of size k. Without loss of generality let it be $V_0 = \{v_1, v_2, \ldots, v_k\}$. For each $i \in [k]$ we move f_i to w_i at a cost of 1. Now since V_0 is a vertex cover each edge in E is incident to some v_i for $i \in [k]$, i.e., for every ℓ_j there is an edge to some v_i for $i \in [k]$. Move ℓ_j along this edge. So finally each client is co-located with a server and each pebble has moved exactly 1, i.e., there is a solution for FacilityLocationMax of cost 1.

Suppose there is a solution for FacilityLocationMax of cost 1. F' has k vertices and solution has cost 1 $\Rightarrow$ these vertices move to at most k vertices in V'. Without loss of generality let $W_1 = \{w_1, w_2, \ldots, w_{k'}\}$ be the vertices in V' to which vertices of F' move to in the final solution for some $k' \leq k$. We claim that $V_1 = \{v_1, v_2, \ldots, v_{k'}\}$ forms a vertex cover of G. Note that each vertex of E' is a client. Since solution of FacilityLocationMax has cost 1, any vertex of F' can only move to a vertex of W_1. Thus each vertex has to be able to move to some vertex in W_1 with cost 1. Therefore for each $\ell_j \in E'$ there is a edge to some vertex in W_1, i.e., V_1 is a vertex cover of G. Since $k' \leq k$ we are done. □

5.3 No Additive Factor Approximation for FacilityLocationMax

This observation is due to Marek Cygan.

Theorem 5.3. *There is no constant $c > 0$ such that there is an additive c-approximation for FacilityLocationMax unless P=NP.*

Proof. We use the same construction as in Figure 3 except that each edge is now replaced by a path of length c. We now claim the following:

- If G has a vertex cover of size at most k then there is a solution of FacilityLocationMax of cost c
- If G does not have a vertex cover of size at most k then there is no solution of FacilityLocationMax of cost at most $(2c-1)$.

Note that this proves our theorem since $c+c = 2c > (2c-1)$ and hence we would be able to decide the Vertex Cover problem in polynomial time.

Suppose G has a vertex cover of size at most k. Without loss of generality let it be $V_0 = \{v_1, v_2, \ldots, v_{k'}\}$ for some $k' \leq k$. For each $i \in [k']$ we move f_i to w_i at a cost of c. For $k' < j \leq k$ move f_i to w_1. Now since V_0 is a vertex cover each edge in E is incident to some v_i for $i \in [k]$, i.e., for every ℓ_j there is a path of length c to some v_i for $i \in [k]$. Move ℓ_j along this path. So finally each client is co-located with a server and each pebble has moved exactly c, i.e., there is a solution for FacilityLocationMax of cost c.

Suppose G does not have a vertex cover of size at most k. We want to show that there is no solution to FacilityLocationMax of cost at most $(2c-1)$. Suppose to the contrary that there is such a solution. Color all vertices red on paths between E' and V'. Color all vertices green on paths between V' and F'. For $i \in [k]$ we have only three cases when we allow maximum movement of $(2c-1)$ for each server pebble:

1. f_i ends up at a vertex in V'.
2. f_i ends up at a green vertex. Hence f_i has either traveled through exactly one vertex of V' or moved a distance less than c. In the first case, we take f_i back to the vertex of V' which it had crossed. In the second case, we move f_i along the green path and let it meet the unique vertex of V' at which the path of length c ends.
3. f_i ends up at a red vertex. We move f_i back to the unique vertex of V' which it had crossed.

For each $j \in [m]$ we have only three cases when we allow maximum movement of $(2c-1)$ for each client pebble:

1. e_j ends up at a vertex in V'.
2. e_j ends up at a red vertex. Hence e_j has either traveled through exactly one vertex of V' or moved a distance less than c. In the first case, we take e_j back to the vertex of V' which it had crossed. In the second case, we move e_j along the red path and let it meet the unique vertex of V' at which the path of length c ends.
3. e_j ends up at a green vertex. We move e_j back to the unique vertex of V' which it had crossed.

It is easy to see that this modifications give a solution of cost at most $(2c-1)$ where all pebbles are now located in V'. Since we have exactly k server pebbles they can occupy k' vertices in V' for some $k' \leq k$. It is easy to see that these k' vertices form a vertex cover in G which contradicts the fact that G does not have a vertex cover of size at most k. □

References

[1] J. Bredin, E. D. Demaine, M. T. Hajiaghayi, and D. Rus. Deploying sensor networks with guaranteed capacity and fault tolerance. In *MobiHoc*, pages 309–319, 2005.

[2] E. Demaine, M. Hajiaghayi, and D. Marx. Minimizing movement: fixed-parameter tractability. In *In Proceedings of the 17th Annual European Symposium on Algorithms (ESA)*, 2009.

[3] E. D. Demaine, M. T. Hajiaghayi, H. Mahini, A. S. Sayedi-Roshkhar, S. O. Gharan, and M. Zadimoghaddam. Minimizing movement. In *SODA*, pages 258–267, 2007.

[4] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. E. Moret, and B. Zhu. Map labeling and its generalizations. In *SODA*, pages 148–157, 1997.

[5] Z. Friggstad and M. R. Salavatipour. Minimizing movement in mobile facility location problems. In *FOCS*, pages 357–366, 2008.

[6] S. M. LaValle. Planning algorithms, 2004.