Chiang Mai J. Sci. 2017; 44(1) : 279-286
http://epg.science.cmu.ac.th/ejournal/
Contributed Paper

An Improved Approximation Algorithm for the *s-t* Path Movement Problem

Wattana Jindaluang [a], Jakarin Chawachat [b], Varin Chouvatut [b], Jittat Fakcharoenphol*[a] and Sanpawat Kantabutra [c]

[a] Department of Computer Engineering, Kasetsart University, Bangkok, Thailand.
[b] Department of Computer Science, Chiang Mai University, Chiang Mai, Thailand.
[c] Department of Computer Engineering, Chiang Mai University, Chiang Mai, Thailand.
*Author for correspondence; e-mail: jittat@gmail.com

Received: 20 March 2015
Accepted: 3 July 2015

Abstract

This paper considers a movement problem that minimizes the maximum movement of pebbles on a graph to form a path from source vertex s to destination vertex t. The best known algorithm for this problem is a 7-approximation algorithm developed by Berman, Demaine, and Zadimoghaddam in 2011. We refine the analysis of Berman *et. al.* to obtain an $(3+\epsilon)$-approximation algorithm for any constant $\epsilon > 0$. This problem is a subroutine used by Berman *et. al.* for finding a solution to the connectivity movement problem. Using our improved algorithm as a subroutine, the approximation ratio for the connectivity movement problem improves from 136 to $104 + \epsilon$.

Keywords: movement problems, approximation algorithm, graph algorithm

1. Introduction

This paper considers a class of problems called movement problems first introduced in Demaine *et. al.* [1]. Movement problems consist of problems that involve planning the motion of a set of movable objects such that their final positions satisfy some property and their motions were measured under some criteria.

The movement problems have many practical applications. As an example, consider a group of firefighters surrounding a forest fire where each firefighter has a walkie-talkie with limited range connectivity. One problem arising from this situation is to arrange the firefighters in locations such that they can communicate with each other (directly or indirectly through message relaying by other firefighters). Thus, the goal of this problem is finding the minimum distance required for each firefighter to move so that they can form a communication network. In graph theoretical terms, if we have a graph representing all locations and their proximity, we want the subgraph induced by firefighter locations to be connected.

In this paper, we consider the *s-t path movement problem*. In this problem, we are given a graph $G = (V, E)$ with n vertices, m movable objects, called *pebbles*, which are placed on a subset of V, the starting vertex *s*, and the ending vertex *t*. The pebbles are objects that can be moved along the edges of the graph.

The *movement* of a pebble is the distance for which the pebble moves. We say that vertex v is *occupied* by pebble p if pebble p is placed on vertex v. The goal is to move a set of pebbles so that the graph induced by vertices occupied by some pebble contains a path from *s* to *t*. The objective of this movement is to minimize the maximum movement of any pebbles. This problem is referred to in Demaine *et. al.* [1] as PathMax, where it is shown that this problem cannot be approximate better than a factor of $2-\epsilon$, unless P = NP. The best known result for this problem is due to Berman, Demaine and Zadimoghaddam [2] who gave a 7-approximation algorithm. In this paper, we refine the analysis of [2] to yield a $(3+\epsilon)$ -approximation algorithm.

1.1 Previous Results

The *s-t* path movement problem is one of the movement problems defined under the framework of Demaine, Hajiaghayi, Mahini, Sayedi-Roshkhar, Oveisgharan, and Zadimoghaddam [1] where many approximation algorithms for movement problems are presented. (See Section 2 for a more complete review of movement problems.) They showed an $O(\sqrt{m/OPT})$-approximation algorithm for the *s-t* path movement problem where *OPT* is a maximum movement of an optimal solution. In 2011, Berman, Demaine and Zadimoghaddam [2] gave the first constant-factor approximation algorithm to the *s-t* path movement problem. This algorithm is a crucial subroutine they use to obtain the first constant-factor approximation algorithm to the connectivity movement problem that minimizes the maximum movement. In that paper, a 7-approximation algorithm for the *s-t* path movement problem is presented. They also show that given a γ-approximation to PathMax, one can obtain a (8γ+80)-approximation to the connectivity movement problem.

If the objective of the problem is to minimize the sum of the movements, a problem referred to under the movement framework as PathSum, the best approximation ratio is $O(n)$, presented in [1]. Improving this algorithm remains an open problem.

1.2 Our Contributions

By refining the analysis of Berman *et. al.* [2], we obtain a $(3+\epsilon)$-approximation algorithm for the s-t path movement problem. This can be compared to a 7-approximation algorithm by [2].

Our main theorem is the following.

Theorem 1. *For any $\epsilon > 0$, there exists a polynomial-time $(3+\epsilon)$-approximation algorithm for the s-t path movement problem. That is, if there is a solution such that each pebble moves along a path of at most M edges to form an s-t path, the algorithm finds a solution such that each pebble moves along a path of at most $(3+\epsilon)$ M edges.*

Using this algorithm, we obtain a$(104+\epsilon)$ -approximation algorithm for the connectivity movement problem, an improvement from a 136-approximation algorithm in [2].

Our paper is organized as follows. Other related work is shown in section 2. Section 3 presents an overview of Berman, Demaine and Zadimoghaddam's approach. Section 4 presents our algorithm which is our main contribution, constructing the path from an (L,k)-locally consistent solutions.

2. RELATED WORK

In this section, we provide short reviews on movement problems. Motivated by many practical movement problems (see, e.g., [3],

*In [2], the researchers claim that there exists an 7- approximation algorithm for the s-t path movement problem which runs in polynomial time. However the full proofs of the main results and the running time of this algorithm were not showed in the manuscript.

[4], [5], [6], [7] and [8]), Demaine, Hajiaghayi, Mahini, Sayedi-Roshkhar, Oveisgharan, and Zadimoghaddam [1] defined a class of movement problems as the problems that plan to move a subset of movable objects to satisfy some property P. For example, property P can be that the set of vertices occupied by pebbles form a connected component, or the set of occupied vertices form a path from a particular vertex *s* to vertex *t*. For a specific property, there can be many objectives. Demaine *et. al.* [1] considered three possible objectives: (1) to minimize the maximum movement, (2) to minimize the total movement, or (3) to minimize the number of the objects are moved. In this pioneering paper, for the objective that aims to minimizes the maximum movement, Demaine *et. al.* presented an algorithm that finds solutions whose maximum movement is $O(\sqrt{m/OPT})$ if OPT is the optimal maximum movement for the connectivity movement problem. They also obtained an approximation algorithm with the same approximation ratio for the *s-t* path movement problem. Later on, Berman, Demaine and Zadimoghaddam [2] gave the first constant-factor approximation algorithm for the *s-t* path movement problem and the connectivity problem as we mentioned above.

3. OVERVIEW OF THE APPROACH BY BERMAN, DEMAINE AND ZADIMOGHADDAM

Our technique is based on the same technique which Berman, Demaine and Zadimoghaddam [2] used for the *s-t* path movement algorithm (later called BDZ algorithm). This section gives an overview of their algorithm.

The main ingredient of the BDZ algorithm is an intermediate solution of the s-t path movement called a *locally consistent solution*. This solution is intuitively a relaxed solution, for which a pebble can be used many times, but only at the positions which are far from each other. This relaxed constraint allows the use of dynamic programming to deal with the problem. After a locally consistent solution is found, a procedure based on a structural lemma works on the solution to remove multiple appearances of pebbles on the solution to obtain a feasible solution to the original s-t path movement problem.

To formally define a *locally consistent solution*, we first need a few definitions. Let $d(u, v)$ be the shortest distance from vertex u to vertex v in the graph, disk $D(u, R)$ be the set of vertices of distance at most R from u, i.e., $D(u, R) = \{v \in V \mid d(u, v) \leq R\}$, and a disk set $Peb(u, R)$ be the set of pebbles with their initial positions within $D(u, R)$.

We assume that we know the value M of the maximum movement of the optimal solution, since there are only n possible values to remove this assumption we can perform binary search over all the possible values.

While our solution is a path, the BDZ algorithm will not work directly with the whole path. A set of vertices (including also s and t) in the path are denoted as *milestones*, which are equally spread out over the path. A locally consistent solution is defined in terms of milestones and pebbles needed to connect these milestones. The following definition is a generalization of the one in [2] where they set $L = M$ and $k = 7$. Formally, an (L, k) *-locally consistent solution* consists of the milestones $v_1, v_2, \ldots, v_x$ and set of pebbles $S_1, S_2, \ldots, S_x$ where

1. $d(v_1, s) = L - 1$ and $d(v_i, v_{i+1}) = 2L - 1$ for all $1 \leq i \leq x - 2$;

2. either $d(v_{x-1}, v_x) = 2L - 1$ and $d(v_x, t) < L$ or $v_x = t$ and $d(v_{x-1}, t) < 2L$;

3. $r_i = 2L - 1$ for all $1 \leq i \leq x - 1$ and $r_x = d(v_{x-1}, v_x) + d(v_x, t) - L + 1$;

4. $S_i \subset Peb(v_i, M + L - 1)$ and $|S_i| = r_i$;

5. $S_i \cap S_j = \emptyset$ for $i \neq j$ where $i < j < i + k$.

Note that the constraint (5) is the relaxed constraint; it allows pebbles to be used many times if they appear far enough on the path.

In [2] Berman *et. al.* proved that there

exists an (L,k) -locally consistent solution as Lemma 5 below.

Lemma 5. *There exists an* (L,k) *-locally-consistent solution.*

Note that Lemma 5 here appears as Lemma 2 in [2].

To obtain the solution to the original s-t path movement problem, Berman *et. al.* showed how to construct a consistent solution defined in the same way with a stronger guarantee that no pebbles can be used more than once (a new property 5, below):

5. $S_i \cap S_j = \emptyset$ for all $i \neq j$.

Lemma 6 below is a parameterized version of a lemma proved by Berman *et. al.* for $L = M$ and $k = 7$; it shows that an (L,k) -locally consistent solution can be constructed in polynomial time using dynamic programming.

Lemma 6. *An* (L,k) *-locally consistent solution can be found in* $n^{O(k)}$ *time where* n *is a number of vertices in* G.

The BDZ algorithm proceeds in 2 steps. First, it uses a dynamic programming algorithm to find an (*M*, *7*)-locally consistent solution where each pebble moves from its initial position by at most 3*M*-2 steps. In the second step, the solution is converted to a consistent solution. To see that this should be possible, consider a pebble v that appears in the solution more than once. Each vertex occupied by v is not too far from the pebble initial position (at most 3*M*-2 steps), however the distance between two vertices occupied by v on the locally consistent solution is large (more than 14*M* steps). This means that if we reroute the path to take a shortcut through v we should have enough pebbles to fill the gap. Formally, in this conversion, it is guaranteed that each pebble will move additionally by at most 4*M*-2steps. This gives a solution where all pebble move by at most 7*M*-4 steps; hence, they obtain a 7-approximation algorithm.

4. Our algorithm: constructing the solution from an (L,k) -locally consistent solution

We start with an observation on what contributes to an approximation factor of 7 in Berman *et. al.*'s result. In the first step, the additional movement caused by the transformation is roughly 2*M*; this depends crucially on the distance between milestones. If we allow milestones to get closer, we can save this additional movement. In our algorithm, the distance between a pair of consecutive milestones is 2*L*-1<2*M*-1.

In the second step, there is another dependency on the distance between milestones. Note that as we decrease the distance between consecutive milestones v_i and v_{i+1}, the number of pebbles that we can use in the shortcut gets smaller, unless we increase the number k, the number of consecutive sets $S_i, \ldots, S_{i+k}$ that share no pebbles. We use the trade-off between L and k to obtain the improvement on the approximation ratio.

We continue in this section to describe an algorithm that converts a locally consistent solution to a feasible solution to the *s-t* path problem. This first part assumes that we can find an (L,k) -locally consistent solution by using the results of Berman *et. al.* (Lemma 5 and Lemma 6).

In this paper, as in [2], we shall start by finding an (L,k) -locally consistent solution to the problem. However, the notion based on clusters and milestones are closely related to the dynamic programming algorithm used to find an (L,k) -locally consistent solution. We find it easier to work with another formulation based on the actual *s-t* path.

An (L,k) *-locally consistent path* from s to t is an s-t path $\mathcal{P} = u_1, u_2, \ldots, u_l$ with a mapping $p: \mathcal{P} \rightarrow Peb$ such that

1. $u_1 = s$ and $u_l = t$;

2. for every vertex $u_i \in \mathcal{P}$, $p(u_i)$ is the pebble that moves to u_i in the solution, and

3. if $p(u_i)$=$p(u_i)$, i.e., the same pebble moves to two vertices, then $|i-j| > k(2L-1)$.

Note that this definition does not guarantee the distance that a pebble has to move to some vertex in the path. To obtain an (L,k)-locally consistent path from an $(L,k+1)$-locally consistent solution, we use pebbles in S_i to create paths between each consecutive pair of milestones. The following lemma states the property of this construction.

Lemma 1. *Given an* $(L,k+1)$*-locally consistent solution, one can construct an* (L,k)*-locally consistent path whose maximum movement for any pebbles is at most* $M+2L-2$.

Proof: First note that the maximum distance from each pebble to its associate milestones in an $(L,k+1)$-locally consistent solution is $M+L-1$. To form a path connecting milestones, a pebble might have to move additionally with the maximum distance of $L-1$; therefore, the maximum movement is $M+2L-2$. To see that condition 3 holds, first note that in an $(L,k+1)$-locally consistent solution, a pebble can move to two milestones whose indices differ by at least $k+1$. This implies that milestones are of distance at least $(k+1)(2L-1)$ apart. To create a locally consistent path, each pebble may move for at most $L-1$ steps. Therefore the minimum distance between two vertices sharing the same pebble is at least $(k+1)(2L-1)-2(L-1) = k(2L-1) + 1 > k(2L-1)$, as claimed.

Before we prove the main theorem, we need a few definitions. At any point, we maintain an (L,k)-locally consistent path $\mathcal{P} = u_1, u_2, \ldots, u_l$.

Vertex ordering. For any two vertices u_i and u_i we say that u_i appears *before* u_j if $i \leq j$, and u_i appears *after* u_j if $i \geq j$. We say u_i appears *strictly* before or after u_j if $i \neq j$.

Black and white vertices. A vertex $u_i \in \mathcal{P}$ sharing a pebble with other vertices is colored *black*. Other vertices in $\mathcal{P}$ are *white*.

Shortcut vertices. For any black vertex u_i, we define a *shortcut vertex* of u_i as vertex u_j that share some pebble with u_i with maximum index j. Let pebble p be the pebble shared between u_i and u_j. A *shortcut path* of u_i is a shortest path from u_i to p and from p to u_j, a shortcut vertex of u_i. Note that if u_i shares a pebble with more than one vertices, its shortcut vertex is the one that appears after all other vertices sharing a pebble with u_i.

Usable vertices. Vertex u_l is called a *usable vertex* for u_i if u_l is white or satisfies following two conditions: (1) a shortcut vertex of u_l appears before a shortcut vertex of u_i or it has the same shortcut vertex as u_i, and (2) vertex u_l does not share any pebble with any vertices which appear before it.

We are ready to consider the main technical theorem. Given a locally consistent path $\mathcal{P}$. Let u_a be the first black vertex which shares some pebble with the other vertices where a is minimum. We call the prefix path $u_1, u_2, \ldots, u_{a-1}$ (which can be empty) *a consistent prefix of path*.

We shall maintain the following invariant during our conversion algorithm:

- **Invariant 1:** *Every pebble that moves to the consistent prefix will not be moved again.*
- **Invariant 2:** *The distance that every pebble moves to each vertex in the consistent prefix is at most* $3(M+2L-2)$. *The distance that every pebble moves to other part of* $\mathcal{P}$ *is at most* $M+2L-2$.

From Lemma 1, we know that both invariants hold initially after we construct a locally consistent path.

We shall modify the path so that the pebble that moves to u_a only moves to one vertex in the path. Let u_z be a shortcut vertex for u_a.

Let vertex u_y, where $a \le c \le y$, be a vertex with minimum index y that shares a pebble with some vertex before it. (Note that it is possible that $y = z$.) Let u_b be a vertex on $\mathcal{P}$ whose index is minimum that shares a pebble with u_y. (Again, it is possible that $b = a$.) Finally, let u_c be a black vertex whose index satisfies $a \le c \le y$ that maximize the index r of its shortcut vertex u_r; i.e., the shortcut vertex of u_c appears after the shortcut vertex of every black vertex in the set $\{u_a, u_{a+1}, \dots, u_y\}$.

Note that vertices u_b, u_c, u_r, u_y exist because u_a and u_z are feasible candidates for them. We prove a few useful facts before proving the main theorem.

Lemma 2. *The following statements are true.*

1. $a \le b < y \le z$.

2. Since $\mathcal{P}$ *is locally consistent, we have that* $y \ge b + k(2L - 1)$.

3. For every black vertex u_i *such that* $a \le i < y$, *its shortcut vertex* u_j *appears after* u_y *but before* u_r.

Proof: Statement 1 is true from the choices of u_y and u_b. We choose u_y which appears between vertex u_a and u_z, i.e., $a \le y \le z$. To see that $a \le b <$, note that, by definition, u_b must be a black vertex and u_a is the first black vertex in $\mathcal{P}$.

Statement 2 is true from the definition of $\mathcal{P}$.

We shall show that statement 3 is true. Consider vertex u_i such that $a \le i < y$, and its shortcut vertex u_j. Not that index $j \ge y$ from the choice of u_y. Also, $j \le r$ because u_i is also a candidate for u_c and we choose u_c that maximizes index r of its shortcut vertex.

Lemma 3. *Every vertex* u_i, *such that* $c < i < y$, *is usable for* u_c.

Proof: If u_i is a white vertex, we are done. Let's assume that u_i is a black vertex. Thus, we must show that u_i satisfy two conditions. The first condition is that a shortcut vertex of u_i appears before a shortcut vertex u_r of u_c; this is true because of our choice for u_r. Another condition requires that u_i does not share a pebble with any vertices before it. This is true because u_y is the first of such vertices.

Lemma 4. *Consider vertices* u_i *and* u_j, *both of which are not in the consistent prefix of* $\mathcal{P}$. *If Invariant 2 holds and vertex* u_i *shares a pebble with vertex* u_j, *there is a path of length at most* $2(M + 2L - 2)$ *between* u_i *and* u_j.

Proof: Let pebble p be the pebble that moves to u_i and u_j. The path from u_i to p to u_j is of length $2(M + 2L - 2)$ as required.

We are ready to state the main technical theorem.

Theorem 2. *If* $k \ge \frac{6(M + 2L - 2)}{2L - 1}$, *we can obtain* $\mathcal{P}'$ *from* $\mathcal{P}$ *such that*

(1) *the consistent prefix of* $\mathcal{P}$ *is a proper subset of the consistent prefix of* $\mathcal{P}'$,

(2) *every pebble that moves for an additional distance moves to the consistent prefix of* $\mathcal{P}'$, *and*

(3) *the maximum movement of every pebble that moves to the consistent prefix of* $\mathcal{P}'$ *is at most* $3(M + 2L - 2)$.

Proof: There are 2 cases to consider.

Case 1: When $c \le b + 4(M + 2L - 2)$ In this case, since $y - b > k(2L - 1) \ge 6(M + 2L - 2)$, we have that $y - c > 2(M + 2L - 2)$. From Lemma 2, c has at least $2(M + 2L - 2)$ usable vertices. We can then use these pebbles to form a shortcut path for u_c and modify the solution to obtain $\mathcal{P}'$. These pebbles have to move extra distances of $2(M + 2L - 2)$; therefore, the distance that they move, in total, is at most $3(M + 2L - 2)$, implying property (3).

In $\mathcal{P}'$, note that all vertices in its prefix up to u_r do not share any pebbles with other vertices. This means that its consistent prefix

includes vertex u_a; thus, it is a proper superset of $\mathcal{P}$'s consistent prefix, proving property (1). Also note that property (2) is true because every pebble moved in this step appears in the consistent prefix of $\mathcal{P}'$. This case is shown in Figure 1.

Case 2: When $c > b + 4(M + 2L - 2)$. Lemma 4 ensures that there is a path $\mathcal{P}_{by}$ from u_b to u_y of length at most $2(M + 2L - 2)$ and a path $\mathcal{P}_{cr}$ from u_c to u_r of length at most $2(M + 2L - 2)$. Since all vertices u_i such that $c < i < y$ are usable from Lemma 3, we shall use pebbles that moves to vertices $u_{b+1}, u_{b+2}, \ldots, u_{b+2(M+2L-2)}$ to form a shortcut path for u_b and use pebbles that moves to vertices $u_{c-2(M+2L-2)}, u_{c-2(M+2L-2)+1}, \ldots, u_{c-1}$ to form a shortcut path for u_c. The new path $\mathcal{P}'$ consists of four subpaths $(u_1, u_2, \ldots, u_b), \mathcal{P}_{by}, (u_y, u_{y-1}, u_{y-2}, \ldots, u_c)$ and $\mathcal{P}_{cr}$. This case is shown in Figure 2.

Since $c > b + 4(M + 2L - 2)$, we have enough pebbles. The extra distance each pebbles has to move is at most $2(M + 2L - 2)$. This means that the maximum distance each of these pebbles has to move is at most $3(M + 2L - 2)$, and property (3) is true. We can use the same arguments as in Case 1 to prove that properties (1) and (2) are also true.

Theorem 2 clearly implies that if Invariants 1 and 2 hold before we construct $\mathcal{P}'$, they remain valid after the construction procedure. We can obtain Theorem 1, our main result, by repeated applications of Theorem 2.

Theorem 1. *Assume that for any constant k, we can find an (L, k)-locally consistent solution in polynomial*

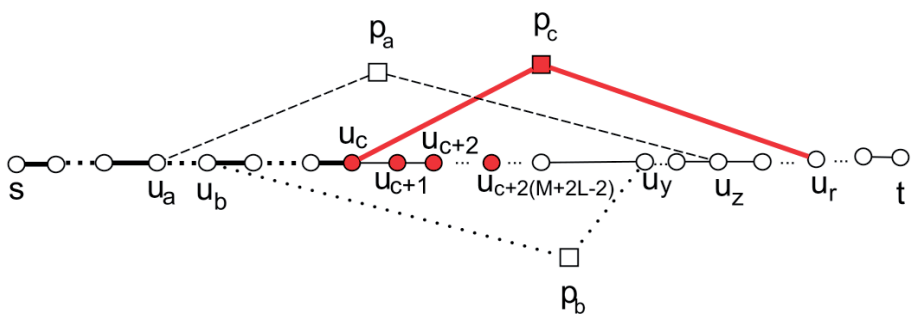

Figure 1. How to form a shortcut path in Case 1. Pebbles are moved from red vertices to form a red shortcut path.

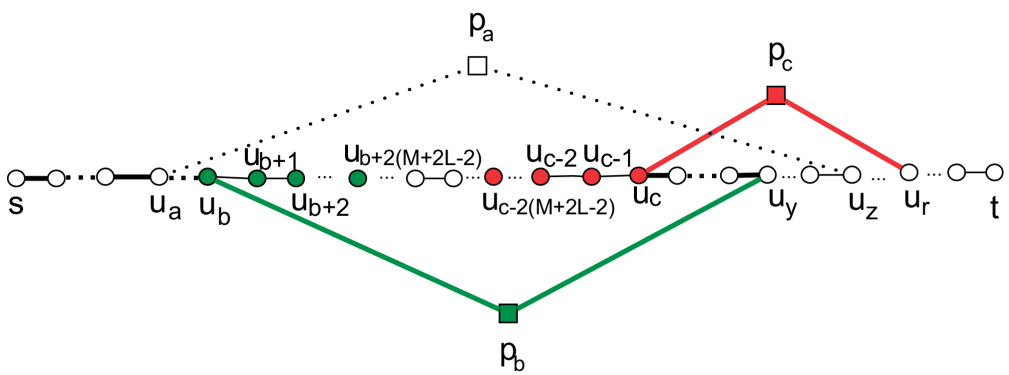

Figure 2. How to form shortcut paths in Case 2. Pebbles in green vertices form a green shortcut path, while pebbles in red vertices form a red shortcut path.

time. For any $\epsilon > 0$*, there exists a polynomial-time* $(3+\epsilon)$*-approximation algorithm for the s-t path movement problem. That is, if there is a solution such that each pebble moves along a path of at most M edges to form an s-t path, the algorithm finds a solution such that each pebble moves along a path of at most* $(3+\epsilon)M$ *edges.*

Proof: Assume first that we have an (L,k) -locally consistent path with the right parameters. Since Invariants 1 and 2 hold throughout and we make progress by increasing the consistent prefix of the locally consistent path the algorithm terminates and every pebble moves by the distance at most $3(M+2L-2)$.

We now verify the approximation ratio. If we set $L=\epsilon M/6$, we have that $3(M+2L-2) < (3+\epsilon)M$ as required. This implies that we have to start with an $(L,k+1)$ -locally consistent solution where

$$k \geq \frac{6\left(M+\frac{2\epsilon M}{6}-2\right)}{(\frac{2\epsilon M}{6})-1}.$$

Assume that $\epsilon M/6 \geq 1$, this is true when $k \geq 18/\epsilon$.

Regarding the running time, from the assumption, since k is a constant, we can obtain the required $(L,k+1)$-consistent solution in polynomial time. The other steps in the conversion algorithm clearly runs in polynomial time. Therefore, we have an algorithm that runs in polynomial time.

5. Acknowledgement

This research is supported by Office of the Higher Education Commission, Thailand. This research has also been partially supported by the Kasetsart University Research and Development Institute, grant number: S-T(I) 69.54.

References

[1] Demaine E., Hajiaghayi M., Mahini H., Sayedi-roshkhar A., Oveisgharan S. and Zadimoghaddam M., *ACM Transactions on Algorithms,* 2009; **5**: 30:1-30:30. DOI10.1145/1541885.1541891.

[2] Berman P., Demaine E. and Zadimoghaddam M., *Proceedings of the 14th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2011)*, Princeton, New Jersey, U.S., 17-19 August 2011; 62-74; DOI 10.1007/978-3-642-22935-0_6.

[3] Croke P., Hrabar S., Peterson R., Rus D., Saripalli S. and Sukhatme G., *Proceedings of the 2004 International Conference on Robotics and Automation*, New Orleans, U.S., 2014; 3602-3608; DOI 10.1109/ROBOT.2004.1308811.

[4] Atay N. and Bayazit B., *Algorithmic Foundations of Robotics,* 2010; **8**: 35-49. DOI 10.1007/978-3-642-00312-7_3.

[5] Ramanna P., Gaikwad P. and Vidyadharan S., *Proceedings of the 12th International Conference on Networks (ICN 2013)*, Seville, Sapin, 2013; 7-14.

[6] Khan M., Hasbullah H. and Nazir B., *AASRI Procedia,* 2013; **5**: 85-91. DOI 10.1016/j.aasri.2013.10.062.

[7] Idoudi H., Houai dia C., Saidane L. and Minet P., *J. Network Technol.,* 2012; **3**: 1-12.

[8] Fan G. and Jin S., *J. Network,* 2010; **5**: 1033-1040. DOI 10.4304/jnw.5.9.1033-1040.