Acta Technica *62* (2017), No. 5A, 575–578

A fast algorithm for freeze-tag problem

Suyuan Luo[2,3]

Abstract. The Freeze-tag problem arises as an important model in message spreading among all kinds of networks in recent years. We developed a novel algorithm for Freeze-tag problem which is a first $O(\sqrt{\log n})$-approximation algorithm for the general unmetric space in which a metric inequality is not assumed.

Key words. Freeze-tag problem, approximation algorithm, metric embedding, divide and conquer.

1. Freeze-Tag Problem

The Freeze-tag problem (FTP) arises as an important model in message spreading among all kinds of networks in recent years[1,2,3]. In the FTP we have n messengers, which are located at nodes in a distance space (e.g., the vertices of an edge-weighted graph). Initially, there is one awake or active messenger and all other messengers are asleep, i.e., in a stand-by mode. Our objective is to "wake up" all of the messengers as quickly as possible. In order for an active messenger to awaken a sleeping messenger, the awake messenger must travel to the location of the slumbering messenger. Once awake, this new messenger is available to assist in rousing other messengers. The objective is to minimize the *makespan*, that is, the time when the last messenger awakens.

The FTP is obviously important in information spreading on networks. Furthermore, what makes the FTP particularly intriguing is that any nonlazy strategy yields an $O(\log n)$-approximation(Proposition 1.1 of [4]), while the strategy for general metric spaces that yields an $o(\log n)$-approximation is difficult to achieve. Arkin et al. [4] showed that even simple versions of the problem (e.g., in star metrics) are NP-complete. They give an efficient polynomial-time approximation scheme (PTAS) for geometric instances on a set of points in any constant dimension δ. They also

[1] Acknowledgement - This paper is supported by National Natural Science Foundation of China under Grant No.11471205;supported by Innovative Team Supporting Project at Shanghai University of Finance and Economics

[2] Workshop 1 - School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai, China, 200433

[3] Corresponding author: Suyuan Luo; e-mail: suyuanluo@126.com

give a variety of results for star metrics (including an $O(1)$-approximation) and for ultrametrics, for which an $o(\log n)$-approximation is possible. Bender, Arkin, Ge[3] improved the result in several special cases. They obtain an $(\frac{L}{d}\log n)$-approximation where L is the longest edge and d is the diameter of the graph.

Recently, J. K nemann, A. Levin and A. Sinha[5,6] proposed an $O(\sqrt{\log n})$-approximation algorithm for the bounded degree minimum diameter spanning tree problem. K?nemann's paper claims the following fact: Suppose that there is a tree T with maximum node-degree B and diameter Δ in a given graph, then, Algorithm, $BDST(G,\Delta)$, produces a tree T^{apx} with maximum node-degree B and diameter $O(\sqrt{\log_B n}\cdot\Delta)$.

Noticing the fact an α-approximation for BDST implies an algorithm for FTP with the same performance ratio, the algorithm brings the first $O(\sqrt{\log n})$-approximation algorithm for FTP. The insight in this reduction derives from the fact a wake-up tree in optimal schedule waking procedure is actually an arboresence each of whose node has out-degree at most the number of messengers at that node.

The main idea in the algorithm relies on a combination of filtering and divide and conquer. They partition the node set of the graph G into clusters such that the diameter of each cluster is low. Then they retain one represent node for each cluster, and define an artificial degree bound for this representative node to account for the degree capacity of the entire cluster. The details of algorithm and performance ratio are both not easy in implementation.

1.1. MainContribution

In this work we developed a novel algorithm for FTP problem which has the same performance ratio for the general unmetric space in which a metric inequality is not assumed. The algorithm developed here is quite simple in both implementation and proving. The main insight behind this algorithm is that a good strategy is to always first wake up the nodes in the denser region greedily. And we also propose some possible direction to derive an algorithm with an approximation bound stronger than $o(\sqrt{\log n})$.

Also, as a direct implication of metric embedding, we notice that there is a simple algorithm which can bring a $O(\sqrt{\log\log n})$-approximation algorithm for FTP in Euclidean space, which is the best algorithm for this case currently.

2. Algorithm

The purpose of this algorithm, is to find an approximation algorithm, whose ratio is $O(\sqrt{\log n})$ or better, for a graph G.

We define n the number of messengers in the graph G. Start with m messengers at a node O. Here we define m to be the largest number of messengers on any single node in the graph G. D is the diameter of the graph G by checking the optimal wakeup tree.

For each node P in G, and a positive real number r, define $D(P,r)$ to be the subgraph of G, all nodes in it are within distance at most r to the node P. And let

n be the number of all messengers in the graph G.

Improved $O(\sqrt{\log n})$ algorithm:

1. Let $k{=}\sqrt{\log n/m}$. And for each $i \in N$, find the $G_i{=}D(P_i, r_i)$ with no less than $m(2^k)^i$ messengers, and r_i is the premium.

2. Now we send the messengers to P_1, then wake G_1 naively.

3. After waking G_i, send all the messengers to P_{i+1}, then wake G_{i+1} naively.

Lemma 1 *This algorithm is $O(\sqrt{logn})$-approximation.*

Proof It is because of the following reasons:

1. r_i at the first step can be found by binary search in polynomial time. We just need to begin binary search from D, and use the traverse of a graph at most n times at each step of binary search. Note that the number of messengers in a traversed subgraph will increase at most m when one more node is added in, so it is feasible to get G_i as we define. It is easy to restrict the number of messengers in G_i is less than $2{*}m(2^k)$.

2. Let's first look at the optimal wakeup tree (we don't know which one it is, but there exists one for the purpose of analysis). The time to wake up all messengers following the order of wakeup tree is optimal. We represent it by OPT. Define t_i to be the optimal time by which the number of awaked messengers grows exactly from $m(2^k)^{i-1}$ to $m(2^k)^i$.

Note that each t_i could overlap with t_{i+1} in the optimal wakeup tree. But the overlapping can only happy at most once for each part, so we get $2 * OPT \geq \int_i t_i$.

3. For each i, the time a_i spend in the algorithm on G_{i-1} to G_i satisfies:

$$a_i \leq D + (k+1)(2 * r_i) \leq D + 4(k+1)t_i \leq D + 8kt_i,$$

i.e., the time spent in step 3 in algorithm.

Note: by naive method, define $m_i = m(2^k)^i$, then we can see $\frac{m_i}{m_{i-1}} = 2^k$, then the activated nodes will be at least doubled at each further step, so all nodes will be activated in at most k steps. and each step will take at most $2 * r_i$ cost.

Note: To wake $m(2^k)^i$ many nodes in time less than $r_i/2$, all the waked messengers would be inside distant $r_i/2$ from the starting point, which contradicts to the definition of r_i. So $t_i \geq r_i/2$.

4. Sum up the above 2 results, we know that the total time is at most $\int_i a_i \leq kD + 8k(2 * OPT)$, which is obviously $18k * OPT$, i.e., $O(\sqrt{\log n})$-approximation. $(OPT \geq D/2)$.

3. Ideas for Improved Algorithm:

Notice in the above algorithm, we wake each G_i only by the simple algorithm. So if we set the number of level to be $(\log n)^{1/3}$, and try some better algorithm for each G_i, the end result would possibly be a $O((\log n)^{1/3})$-approximation algorithm.

There is some gap we can continue to explore. It seems that we should not implement $\sqrt{\log n}$-opt to G_i since it cannot guarantee that $k * r_i$ part, i.e., we cannot wake up all nodes in $\sqrt{\log n} * r_i$ time. The best we can do is just to wake up G_i in $\sqrt{\log n}$-opt. we consider avoiding it by embedding correctly.

If we can make it successfully by a right embedding, then by applying this algorithm to G_i, and reducing levels, we can get $O((\log n)^{1/4}$ approximation. If we do it multiple times, we can get $O(\log n)^1$ approximation for any S.And if we apply it loglog$n$ times, by some calculation, we could see this becomes a $O(\log\log n)$-approximation hopefully. Currently we are working on it.

4. Euclidean space

If n points are in Euclidean space, we already have 3+-opt algorithm if the dimension is less than $o(\log/\log\log n)$. And note that there is a well known embedding theorem by Johnson-Lindenstrauss[7]: we can embed one n-points structure in euclidean space to $O(\frac{1}{2}\log n)$-dim space with distortion 1+. So here if we let $=\sqrt{\log\log n}$, we can get $O(\sqrt{\log\log n})$-approximation for general Euclidean case.

Lemma 2 *One $O(\sqrt{loglogn})$-approximation algorithm for euclidean space is direct by applying Johnson-Lindenstrauss embedding.*

References

[1] S. Albers, M. R. Henzinger: *Exploring unknown environments.* In Proc. 29th ACM Sympos (1997) 416–425.

[2] S. Albers, K. Kursawe, S. Schuierer: *Exploring unknown environments with obstacles.* In Proc. 10th ACM-SIAM Sympos. Discrete algorithms (1999) 842–843.

[3] E. M. Arkin, M. A. Bender, P. A. A. Laura: *Improved approximation algorithm for the freeze-tag problem.* SPAA (2003).

[4] E. M. Arkin, M. A. Bender, S. P. Fekete: *he Freeze-Tag Problem: How to wake up a swarm of messengers.* In Proc. 13th ACM-SIAM Sympos (2002) 568–577.

[5] J. Konemann, A. Levin, A. Sinha: *Approximating the degree-bounded minimum diameter spanning tree problem. In Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX).* Lecture Notes in Computer Science *2676* (2003) 109–121.

[6] J. Konemann, A. Levin, A. Sinha: *Approximating the degree-bounded minimum diameter spanning tree problem.* Algorithmica *41* (2004), No. 2, 117–129.

[7] W. B. Johnson, J. Lindenstrauss: *Extensions of Lipschitz mappings into a Hilbert space.* Contemporary mathematics *26* (1984), No. 1, 189–206.

Received November 16, 2017