MMST of Moving Points

Sobyasachi Chatterjee
under the supervision of Prof. Anil Maheswari[1]

[1]Ramakrishna Mission Vivekananda Educational and Research Institute
[2]Belur, India

[*]*Correspondence to* sobyasachichatterjee@gmail.com

Abstract

Background In this project we will try to address the question that has been once attempted successfully to find a better approximation ratio to a NP-Hard problem . In this project we will attempt to find a better approximation ratio to a specialised problem. *In this project we experimented on more than 25000 graphs and used a clever priority technique to get to a better approximation ratio Here, we show. . .* that the approximation ratio of 1.5 never fail in almost all cases. The problem above is inspired by the fact that people tend to move closer to other be it due to work or some other activities.Here we consider one node moving towards other and find a better way to approximate.

1 Introduction

The problem *While the approximation ratio of 2 was found by a renowned researchers of Canada, there are no more researches carried out in this area.*.Here we will try to approach a problem where nodes few nodes in 2-dimensional space that moves towards the other while we find the minimum weight spanning tree whose weight of the spanning tree is the max over the sum of weights of all the time stamps.

More formally,

A moving point p in the plane is a continuous function p : [0, 1] → R^2. We assume that p moves on a straight line segment in R^2. We say that p is at p(t) at time t. We are given a set of points $S = a_1, ...a_n, b_1, ...b_n$ in the plane where a_i moves to b_i . A moving spanning tree T of S has S as its vertex set and weight function w_T : $[0, 1] \rightarrow$ R defined as $w_T(t) = \sum_{pq \epsilon T} \|p(t)q(t)\|$. Let T (S) denote the set of all moving spanning trees of S. Let $w(T) = sup_t w_T(t)$ be the weight of the moving spanning tree T . A minimum moving spanning tree (MMST) of S is a moving spanning tree of S with minimum weight. In other words an MMST is in

$$\arg\min_{T(S)}(w(T))$$

Old results *The Conference paper published on 2021 revolves on the O(n log n) time (2+ϵ approximation of a more generalised MMST finding algorithm. It had a O(n^2) time 2-approximation algorithm which we are going heavily use in this project.* Infact this will be our starting tool for the entire project. So we present some results from the conference paper,

- The distance function d is convex where the points move linearly.
- This leads to a corollary that the largest distance between two moving points is attained either at the start time or at the finish time.
- Let G be the complete graph on points of S where the points of S where the weight w(pq) of every edge pq is the largest distance between p and q during time interval[0,1], that is w(pq) = $\|pq\|_{max}$.
  The weight of any MST of G is at most two times the weight of any MMST of S.
- The decision version of the MMST problem is weakly NP-hard.

2 Results

2.1 Declarative result statement

Context Tightness were proved for the more general problem where they proved the approximation ratio is 2 but for the specialized problem , a kind of similar spread of the points had a new MMST given by their 2-approximation algorithm

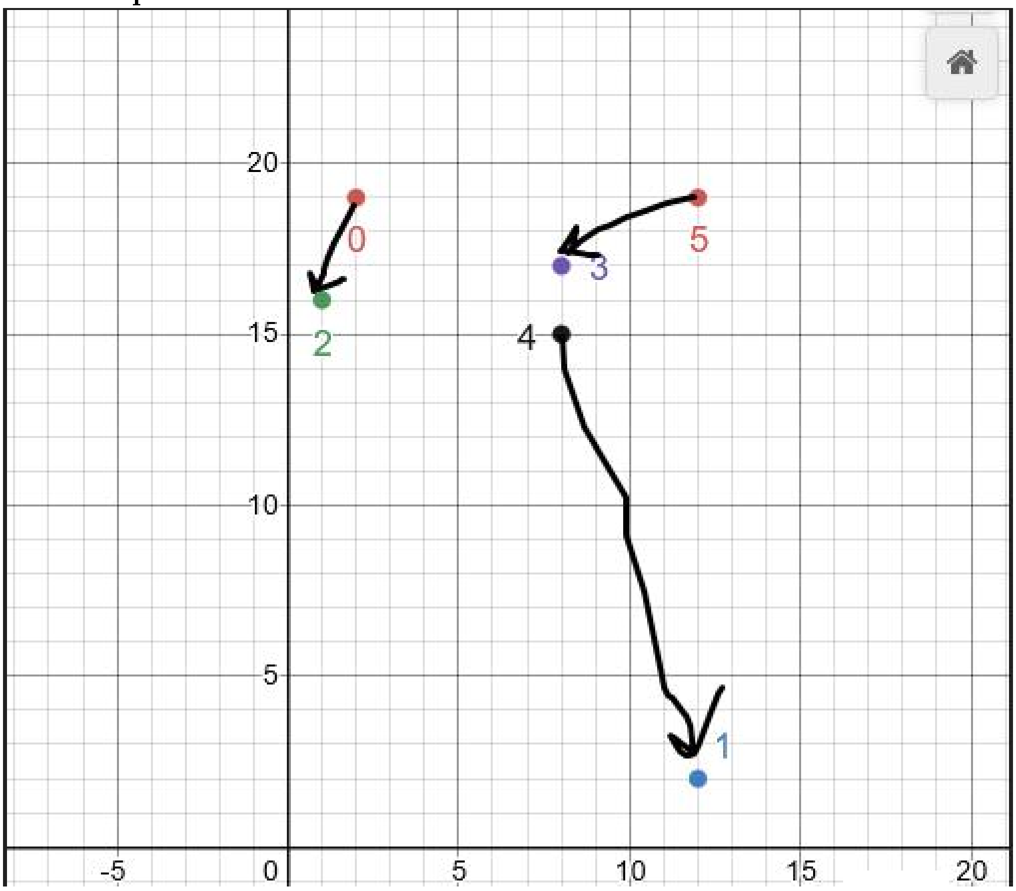
Approach We divided the time span from 0 to 1 second in 20 steps.At each step we compute the cost (summing over the distance for the particular spanning tree).The max cost is the cost of that spanning tree.So for experimentation we had to find graphs of size up to 8 because the number of spanning trees can be at most n^{n-2}

Result $p\{w(T) <= 1.5w(T')\} <= \epsilon$ where T' is the best possible MMST and T is the algorithm generated MST

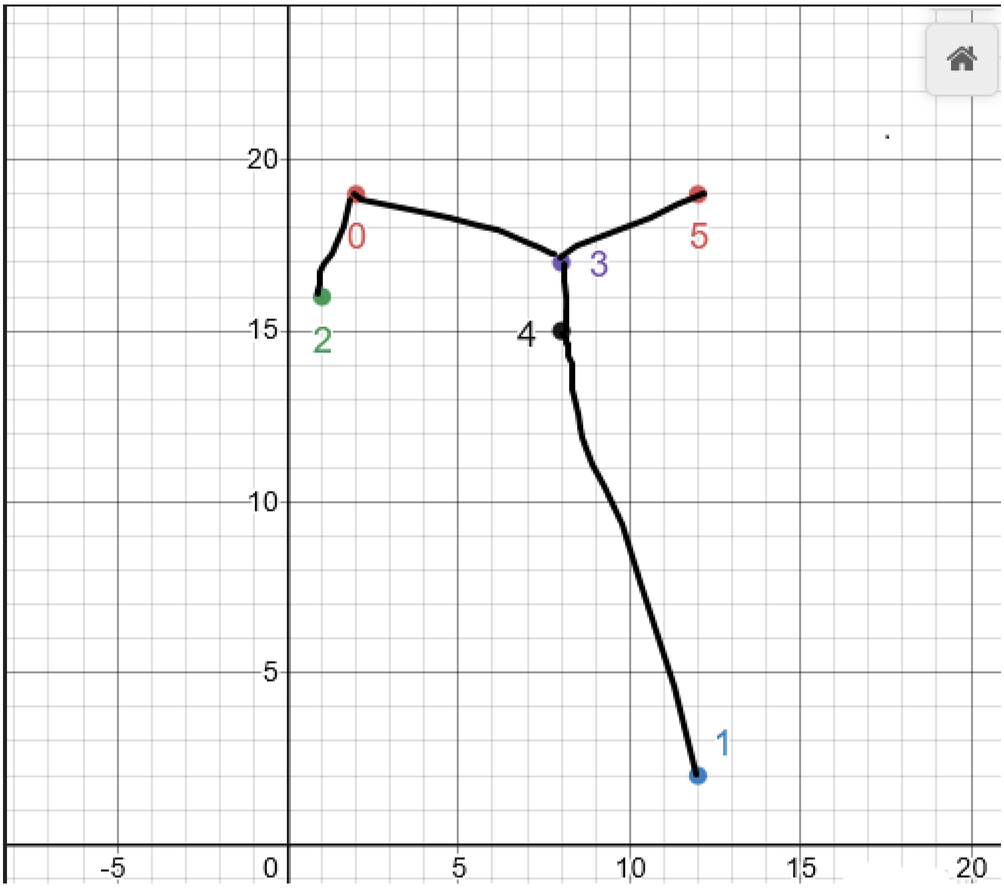
Details Find a graph G where G has the same vertex set as S and the weight of e(pq) = max(p(0)q(0),p(1)q(1)). It is here we remember the corollary stated by the conference paper that the largest distance between two moving points is attained either at the start time or at the finish time which they use . We see that such an implementation of G and find the MST of the graph G using the Kruskal's Algorithm where we saw $p\{w(T) <= 1.5w(T')\} <= c$ where c ¡= 0.5 for the edges that Kruskal's Algorithm gives.

Conclusion As there are several MST$_s$ we can get close with the constant c so we use priority evaluation on them *This suggests...*

Example



Suppose we want to find the MMST of this euclidean points where the movement pairs are shown by arrows.By a little speculation we can see this to the MMST of the graph.



2.2 Example results section to demonstrate figure references and placement

This is the table of cost for the Kruskal's Algorithm for the above example.We can see the amount of duplicates and we show that the amount of alternatives leads to demise of the ratio.

	0	1	2	3	4	5
0		19.72	3.16	7.07	17.8	10
1			17.8	15.52	13.6	17
2				7.07	17.80	11.40
3					15.52	4.47
4						15.52
5						

3 Methods

3.1 Results

We will refer to the approximation ratio as r

Nodes	4	6	8
Graphs	10000	10000	1946
Average	1.021	1.070	1.095
Graphs with $r = 1$	8698	4224	279
Graphs with $1 <= r <= 1.1$	9299	7154	1149
Graphs with $r > 1.5$	15	27	0
max value of r	1.76	1.76	1.48
Percentage graphs exceeding 1.5	0.0015	0.0027	0

3.2 Priority used

This is the part where we have used a clever priority evaluation of the MMST. We take the best of 4 graphs

1. The MST of the graph G mentioned in Sec 2.1
2. The MST of the graph G prioritising the distance at time 0. Essentially we found through all the experiments that mostly the structure of the graph is depicted by the initial time stamp
3. The MST of the graph G prioritising the minimum distance.As the cost is going to decrease somewhere , so we prioritise it to the get the minimum .
4. The MST of the graph G prioritising w.r.t cost at time 0 and minimum distance with more priority on at time 0

We could have implemented the Kruskal's algorithm, used priority queue but instead we use a clever change in the cost to see a huge improve in the value of ϵ

3.3 Claim : There can never be a graph with $r > 1.5$ and that have an unique MST

Of course our results highly suggest of such a possibility We represent l_i as the movement line of a_i to b_i . Either one of a_i or b_i has same distance to each of other a_j or b_j . So there are n*(n-1) . Now as the $r > 1.5$ there are only there are nC_2 possibilities. Thus in one of the MST calculation there should be an alternative in which one can use the priority.

3.4 Code availability

The publicly available code for analysis are available in the following repositories:

The custom code we generated are available in the repository "[MST of Moving Points]" available here [https://github.com/Sobyasachi/MST-of-Moving-Points.git].
The csv files marked by best follows directly from the use of the algorithm.

3.5 References

Akitaya, H.A. et al. (2021). The Minimum Moving Spanning Tree Problem. In: Lubiw, A., Salavatipour, M., He, M. (eds) Algorithms and Data Structures. WADS 2021. Lecture Notes in Computer Science(), vol 12808. Springer, Cham. https://doi.org/10.1007/978-3-030-83508-8-2

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society, 7(1), 48–50.