Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 22(3): 215-220, Sep. 2024 Regular paper

Optimal Terminal Interconnections Using Minimum Cost Spanning Tree of Randomly Divided Planes

Minkwon Kim, Yeonsoo Kim, Hanna Kim, and Byungyeon Hwang*, *Member, KIICE*

School of Computer Science and Information Engineering, the Catholic University of Korea, Bucheon 14662, Republic of Korea

Abstract

This paper presents an efficient method for expanding interconnections in scenarios involving the reconstruction of interconnections across arbitrarily divided planes. Conventionally, such situations necessitate rebuilding interconnections based on all targets, ensuring minimal cost but incurring substantial time expenditure. In this paper, we present a tinkered tree algorithm designed to efficiently expand interconnections within a Euclidean plane divided into m randomly generated regions. The primary objective of this algorithm is to construct an optimal tree by utilizing the minimum spanning tree (MST) of each region, resulting in swift interconnection expansion. Interconnection construction is applied in various design fields. Notably, in the context of ad hoc networks, which lack a fixed-wired infrastructure and communicate solely with mobile hosts, the heuristic proposed in this paper is anticipated to significantly reduce costs while establishing rapid interconnections in scenarios involving expanded connection targets.

Index Terms: Interconnection graph problem, Minimum cost spanning tree, Prim's algorithm, Tinkered tree

I. INTRODUCTION

Interconnection construction is an abstract problem in various industries, from networks to architecture [1-4]. The minimum spanning tree (MST) algorithm can be employed to connect all objects with the minimum length during interconnection construction [5,6]. If the target of the interconnection is expanded, it is necessary to reconstruct the inter-connection using the MST algorithm, which is time-consuming.

In this study, we address the scenario in which the interconnection target expands and the newly added components establish an MST within each subdivided plane. By leveraging this information, we present a heuristic approach that achieves faster terminal connections than the MST algorithm. The heuristic generates an optimal tree, referred to as the tinkered tree, which bears a resemblance to MST. In practical applications, this study can be employed to effectively construct a network service that supports the entire R2 based on the existing m network services that operate locally within a wide area R2. Furthermore, an efficient connected dominating set (CDS) configuration of a wireless ad hoc network consisting only of mobile hosts, without establishing a fixed wired network, can be expected [7-10]. In the following section, research related to this study is presented. In Section 3, we explain the heuristic proposed in this paper, and Section 4 proves the usefulness of the heuristic proposed in this paper through experiments. In Section 5, we describe the conclusions of this study.

II. RELATED WORKS

An important component of a computer system is its interconnected network. Several problems related to interconnected networks have been identified. Tripathy et al. [3]

Received 28 May 2023, **Revised** 5 October 2023, **Accepted** 14 October 2023
***Corresponding Author** Byungyeon Hwang (E-mail: byhwang@catholic.ac.kr, Tel: +82-2-2164-4363)
School of Computer Science and Information Engineering, the Catholic University of Korea, Bucheon 14662, Republic of Korea

Open Access https://doi.org/10.56977/jicce.2024.22.3.215 print ISSN: 2234-8255 online ISSN: 2234-8883

presented a genetic algorithm-based approach for solving the topology optimization problem of interconnection networks to optimize terminal reliability under predefined cost constraints. This approach is different from the heuristic used in this study, which uses existing interconnections to shorten the execution time and bring the cost closer to the optimum value. Kim et al. [4] considered the construction of a maximum interconnection modeled as a complete graph by setting the connection target as a terminal on the Euclidean plane and the connection cost of each edge as the distance between the terminals. When the number of objects to be connected is not fixed and not all objects of a given length can be interconnected, the problem of constructing a maximum interconnection is proven to be NP-hard, and a heuristic for constructing an efficient interconnection using a memetic genetic algorithm is proposed. Although it is common to address optimal interconnection with a planar connection target, the approach introduced by Kim et al. [4] differs in that a limit is imposed on the maximum length that can be connected. Wieselthier et al. [5] proposed a minimum energy broadcast problem that minimizes the total transmission energy required to broadcast messages from one node to all nodes within a network. In practical applications, these interconnections are constantly changing, and the cost of rebuilding them can be substantial. Therefore, we propose a tinkered tree that rapidly establishes interconnections between these ever-changing interconnections, while preserving the initially established connections, as part of our related work.

III. HEURISTIC OF THIS PROBLEM

A. Preprocessing

To validate the effectiveness of this study, it was essential to develop a benchmark model that could be used for comparison with the results of this research. We employed the MST with the lowest cost encompassing all terminals as a reference point to demonstrate the advantages of our findings. After constructing an MST involving all terminal sets using Prim's algorithm, the resulting information was preserved. After constructing the MST based on the terminal subset in each partition, the information on the length and time used to construct the corresponding MST was stored.

B. Distance of each Partition

Definition 3-1. *A portal is located on the boundary between a partition and an adjacent partition, and is a benchmark used to find the closest pair connecting the two partitions.*

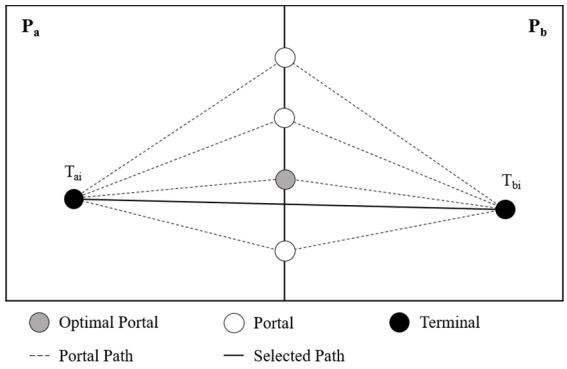

Fig. 1. Example of selected portal and path

Lemma 3-1-1 *When two partitions Pa and Pb exist, a total of i portals are evenly distributed as {Portal1, Portal2, ⋯, Portal i} on the line dividing the partition. The number of portals, i, is determined in a later experiment.*

To create an optimal tree similar to the MST without generating it for the entire set of terminals, it is necessary to identify the closest pair for connecting each partition with its adjacent partition. In this study, we employed the portal method as an effective approach to identify the closest pairs. Based on Portal i, terminal T_{ai} closest to P_a and terminal T_{bi} closest to P_b were found. The portal distance for Portal i is calculated as distance (T_{ai}, Portal i) + distance (T_{bi}, Portal i). Accordingly, the portal with the shortest distance was selected. Fig. 1 shows an example of the selected portal and its path.

In this case, the portal distance was calculated to be greater than or equal to the closest pair distance. Consequently, after identifying the closest pair for each partition using the portal method, the distance between the partitions was determined as the closest pair distance (Tai, Tbi).

Theorem 3-1. *Portal distance (Distance (T_{ai}, Portal i) + Distance (T_{bi}, Portal i)) uses a length that is greater than or equal to the closest pair distance (T_{ai}, T_{bi}).*

Proof. *The portal distance, which is a connection through a portal, consumes more cost than the closest pair distance, which is a connection that does not go through a portal. i.e., Distance (Tai, Portal i) + Distance (Tbi, Portal i) < Distance (Tai, Tbi). However, if there is a portal at the intersection of the boundary line dividing Pa and Pb and the line connecting Tai and Tbi, the distance is the same as Distance (Tai, Portal i) + Distance (Tbi, Portal i) = Distance (Tai, Tbi). Therefore, Distance (Tai, Portal i) + Distance (Tbi, Portal i) ≤Distance (Tai, Tbi).*

Fig. 2 shows the pseudocode for finding the closest pair using a portal and determining the distance between partitions.

```
# DEFINE 1. Partition: Any Partition
# DEFINE 2. AdjPartition: Another partition adjacent to the partition

GET_DISTANCE_BETWEEN_PARTITION (portal, partition, adjPartion)
    distanceOfPartion ← INF
    FOR i = 1 to Number Of Terminal In partition
        distanceBox ← CALCAULTATE_DISTANCE(partition.terminal[i], portal)
        IF (distanceOfPartition > distanceBox) THEN
            distanceOfPartition ← distanceBox
            indexOfPartition ← i
        END IF
    END FOR

    distanceOfAdjPartion ← INF
    FOR j = 1 to Number Of Terminal In adjPartition
        distanceBox ← CALCAULTATE_DISTANCE(adjPartition.terminal[j], portal)
        IF (distanceOfAdjPartion > distanceBox) THEN
            distanceOfAdjPartion ← distanceBox
            indexOfAdjPartition ← j
        END IF
    END FOR
    RETURN CALCAULTATE_DISTANCE(
        partition.terminal[indexOfPartition],
        adjPartition.terminal[indexOfAdjPartition]
    )
END FUNCTION
```

Fig. 2. Pseudocode to obtain the distance between partitions using a portal

C. Tinkered Tree Heuristic

There are m partition sets, denoted by P = {P1, P2, …, Pm}, with the MST information of each terminal subset stored within every partition. Let the length of the terminal subset MST for Partition Pi be denoted by length (Pi). In the previous section, the shortest distance between all the partitions was determined using the closest pair algorithm through the portal. The heuristic approach introduced in this study, known as the tinkered tree, treats the MST of each partition as an individual component. A tinkered tree was then constructed by connecting these components based on the shortest distance between the closest pairs. Fig. 3 shows the progress of the tinkered tree heuristic when the number of partitions was three.

In Fig. 3, the closest pair of Partition0 and Partition1 is terminals a and c, the closest pair of Partition0 and Partition2 is terminals b and e, and the closest pair of Partition1 and Partition2 is terminals d and f. Currently, if the MST is generated by the distance between partitions using each closest pair, d and f, which have the longest distance, are not connected, and all partitions are connected by the connection between a and c and the connection between b and e. In this way, the tinkered tree uses the closest pair between the partitions to create an MST that connects the three partitions, effectively constructing an interconnection without damaging the existing terminal subsets.

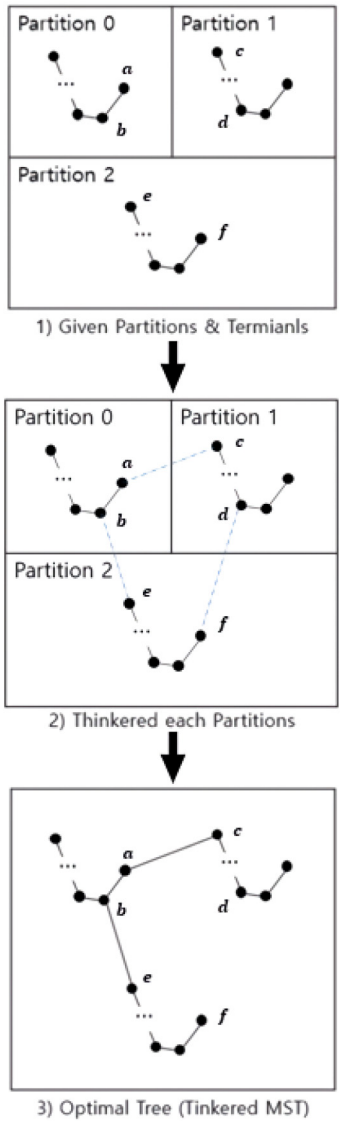

Fig. 3. Tinkered tree heuristic

IV. EXPERIMENTS

A. Create Instance and Benchmark Model

This method was used to create the necessary variables (Portal, Partition, and Terminal) for the study. The instance in this study is not just a method of arranging uniform terminals in the partition on the Euclidean plane, but is designed for an instance that is more suitable for practical applications. Therefore, the instance was self-made, and the self-made instance settings were as follows: (1) given in 100000 × 100000 Euclidean plane; (2) Portals are located at the

boundary of the partition and distributed at equal intervals according to the number; (3) Partitions are given in the form of an initial Hanan grid and are created as unequal partitions through merge [11]; (4) The coordinates of each terminal are rounded to two decimal places; (5) The number of terminals in the partition was distributed as evenly as possible, and if the partition and the terminal were not divided, they were randomly generated. The heuristic presented in this paper aims to efficiently link all provided terminals using the shortest feasible distance. A criterion was required to assess the effectiveness of the heuristic. The task of connecting all terminals with the least possible length can be addressed using the MST algorithm. Therefore, using Prim's algorithm as a control, the ratio of execution time to the total length of connections used by Prim's algorithm was compared with that of the proposed heuristics.

B. Environment for Experiments

The implementation environment for the experiment was the same as that in [4], and the heuristic proposed in this study was written in Java. The heuristic demonstrates variations in performance based on the number of portals, partitions, and terminals. Consequently, it is essential to determine the optimal values of these three variables to achieve the most favorable outcomes in the final experimental results. In the experiment, for each variable, the time and length of the heuristic were expressed as a ratio compared with the benchmark model according to the variable value, and the optimal value was determined among the values. To determine the number of portals, the numbers were compared by conducting an experiment on cases in which the number of portals ranged from 1 to 1,000. The experimental result for the number of portals in each case was the average of the time and length of the number of portals after 10 experiments for different inputs. Fig. 4 shows a graph of the time and length for interconnection according to the number of portals in the heuristic proposed in this study.

As shown in Fig. 4, as the number of portals increased, the time increased, and the length gradually decreased, converging to approximately 102.04%. Therefore, when the number of portals was 50, it was judged that most of them converged; in the final experiment, the number of portals was fixed at 50, and the experiment was carried out. To determine the optimal number of partitions, the numbers were compared across cases ranging from 11 to 5,010 partitions. Similar to the experiment used to determine the number of portals, the experimental result for each case was the average of the time and length of the number of partitions after 10 experiments for different inputs. Fig. 5 shows a graph of the time and length for interconnection according to the number of partitions in the heuristic proposed in this study.

Therefore, in the final experiment, the number of partitions was fixed to 110, and the experiment was carried out. To establish the appropriate number of terminals for the final experiment, we conducted trials spanning terminal counts ranging from 10,000 to 500,000. The experimental results for each case were obtained by averaging the time and length of the number of terminals through 10 experiments using different inputs. Fig. 6 shows a graph of the time and length

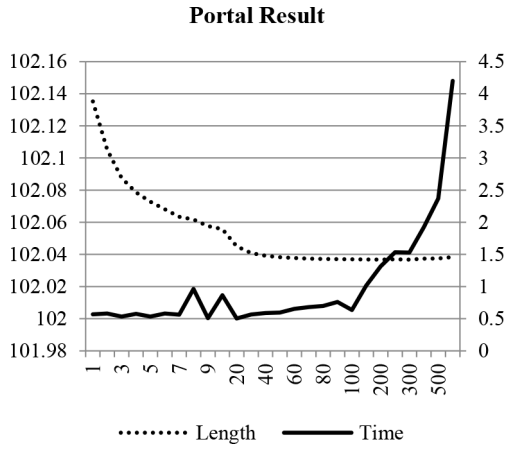

Fig. 4. Results according to the number of portals

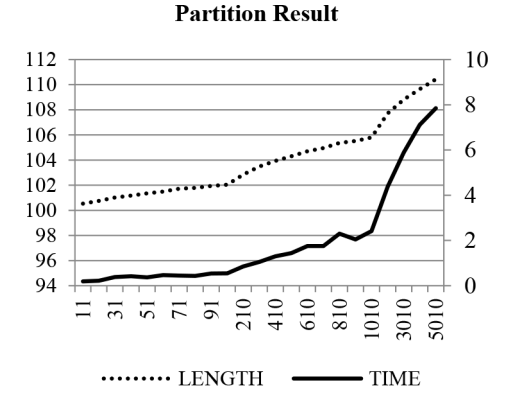

Fig. 5. Results according to the number of partitions

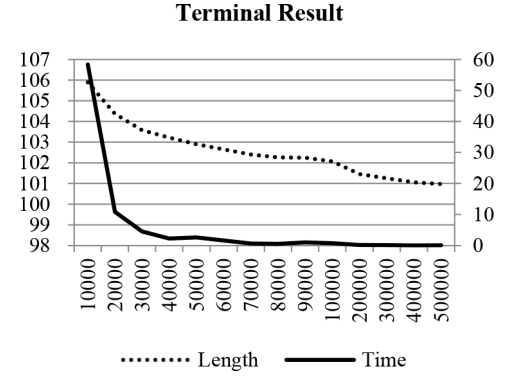

Fig. 6. Results according to the number of terminals

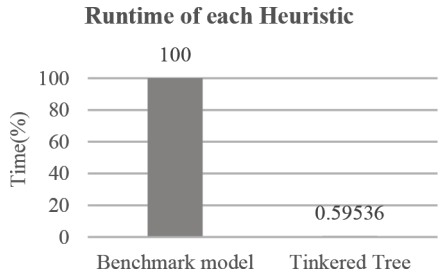

Fig. 7. Runtime of each heuristic

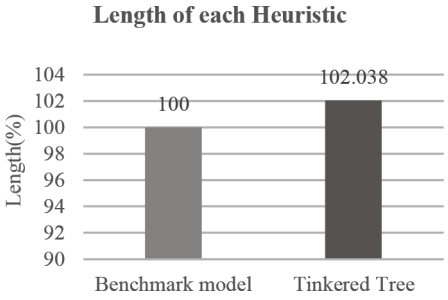

Fig. 8. Length of each heuristic

for interconnection according to the number of terminals in the heuristic proposed in this study. As shown in Fig. 6, as the number of terminals within all partitions increased, the time and length decreased. The time and length decreased rapidly when the number of terminals was 100,000. Therefore, in the final experiment, the number of terminals within all partitions was fixed to 100,000, and the experiment was carried out.

D. Performance Comparison

In the previous experiment, the optimal values of the three variables—Portal, Partition, and Terminal—were determined and selected to yield favorable outcomes in the final experiment. In this experiment, we aimed to determine how good the values were compared with the benchmark model when the determined values were applied. The test result was the average value obtained in the same manner with 10 different inputs. Fig. 7 shows a graph comparing the time of the tinkered tree proposed in this study with the benchmark model as a ratio, and Fig. 8 shows a graph comparing the length of the benchmark model with that of the tinkered tree proposed in this paper as a ratio.

As shown in Fig. 7, the tinkered tree proposed in this study exhibits a time reduction of 99.4% compared with the benchmark model. As shown in Fig. 8, the tinkered tree proposed in this study exhibits an increase in length of 2.0% compared with the benchmark model. In other words, the results show that the tinkered tree proposed in this study works efficiently by shortening the execution time to 99.4% instead of increasing the length by 2.0% compared with the benchmark model.

V. CONCLUSIONS

In this study, we introduced a heuristic tinkered tree that efficiently establishes interconnections using a given MST of partitions randomly divided across the Euclidean plane. This approach eliminates the need to reconstruct the MST for the entire plane while effectively utilizing the information provided. Although this problem does not guarantee minimum cost, as in the MST algorithm, it can be expected to perform well in ad hoc networks where dynamic changes in the network topology make it difficult to maintain paths. Additionally, one can expect an effect when combining raw data from the online store with data computed from the offline store within a big data platform. As a prospective research avenue, we propose a methodology that is applicable across diverse domains. This entails extending the problem to scenarios where factors such as the distribution of terminals within partitions or alterations in partition coordinates come into play.

REFERENCES

[1] K. Zhou and J. Chen, "Simulation DNA algorithm of set covering problem," *Applied Mathematics & Information Sciences*, vol. 8, pp. 139-144, Jan. 2014. DOI: 10.12785/amis/080117.

[2] A. Mariano, D. Lee, A. Gerstlauer, and D. Chiou, "Hardware and software implementations of Prim's algorithm for efficient minimum spanning tree computation," in *International Embedded Systems Symposium*, Berlin, Germany, pp. 151-158, 2013. DOI: 10.1007/978-3-642-38853-8_14.

[3] P. Tripathy, R. K. Dash, and C. R. Tripathy, "A genetic algorithm based approach for topological optimization of interconnection networks," *Procedia Technology*, vol. 6, pp. 196-205, Nov. 2012. DOI: 10.1016/j.protcy.2012.10.024.

[4] J. Kim, J. Oh, M. Kim, Y. Kim, J. Lee, S. Han, and B. Hwang, "Maximum node interconnection by a given sum of Euclidean edge lengths," *Journal of Information and Communication Convergence Engineering*, vol. 17, no. 4, pp. 246-254, Dec. 2019. DOI: 10.6109/jicce.2019.17.4.1.

[5] J. E. Wieselthier, G. D. Nguyen, A. Ephremides, "On the construction of energy-efficient broadcast and multicast trees in wireless networks," in *Proceeding of IEEE Infocom 2000*, Tel Aviv, IL, pp. 585-594, 2000. DOI: 10.1109/INFCOM.2000.832232.

[6] D. Hu, P. Dai, K. Zhou, and S. Ge, "Improved particle swarm optimization for minimum spanning tree of length constraint problem," in *Proceeding of International Conference on Intelligent Computation Technology and Automation*, Nanchang, CN, pp. 474-477, 2015. DOI: 10.1109/icicta.2015.124.

[7] C. E. Perkins, "Ad hoc networking in the IETF," in *IEEE International Workshop on Broadband Convergence Networks*, Vancouver, CA, 2006. DOI: 10.1109/BCN.2006.1662294.

[8] S. Ren, P. Yi, D. Hong, Y. Wu, and T. Zhu, "Distributed construction

of connected dominating sets optimized by minimum-weight spanning tree in wireless ad-hoc sensor networks," in *IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, CN, pp. 901-908, 2014, DOI: 10.1109/cse.2014.183.

[9] X. Zhang and X. Zhang, "A binary artificial bee colony algorithm for constructing spanning trees in vehicular ad hoc networks" *Ad Hoc Networks*, vol. 58, pp. 198-204, Apr. 2017. DOI: 10.1016/j.adhoc.2016.07.001.

[10] J. J. Kponyo, Y. Kuang, E. Zhang, and K. Domenic, "VANET cluster-on-demand minimum spanning tree (MST) prim clustering algorithm," in *International Conference on Computational Problem-Solving*, Jiuzhai, CN, 2013. DOI: 10.1109/ICCPS.2013.6893585.

[11] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM Journal of Applied Mathematics*, vol. 14, no. 2, pp. 255-265, Mar. 1966. DOI: 10.1137/0114025.

Minkwon Kim

received his B.S. degree in computer science and information engineering from the Catholic University of Korea in 2021. His research interests include database and algorithm.

Yeonsoo Kim

received his B.S. degree in computer science and information engineering from the Catholic University of Korea in 2021. His research interests include database and algorithm.

Hanna Kim

is an undergraduate majoring in computer science and information engineering at the Catholic University of Korea since 2017. Her research interests include database and algorithm.

Byungyeon Hwang

received his B.S. degree in computer engineering from Seoul National University, Republic of Korea in 1986, and MS and PhD degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 1989 and 1994, respectively. He is a professor in the School of Computer Science and Information Engineering at the Catholic University of Korea. His research interests include database, social network analysis, and approximation algorithms.