On the Robustest Spanning Tree Problem

Ruei-Yuan Chang, Sheng-Lung Peng*, Cheng-Yi Wang
Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien 97401, Taiwan
*corresponding author: slpeng@mail.ndhu.edu.tw

Abstract

Let $T = (V, F)$ be a spanning tree of a simple connected graph $G = (V, E)$. It forms a cycle C by adding an edge $e \in E \setminus F$ to T. Let $num(e) = |C| - 1$. We say that the robustness number of T is $R(T) = \sum_{e \in E \setminus F} num(e)$. The robustest spanning tree problem *of G is to find a spanning tree T with minimum $R(T)$ among all possible spanning trees of G. We propose linear-time algorithm for solving this problem on block graphs.*

1 Introduction

Let $G = (V, E)$ be a connected simple graph. A spanning tree $T = (V, F)$ of G is a tree containing all vertices of G. Spanning trees are important because we can obtain some information about the original graph from some special spanning trees. Spanning trees also play an important role in designing efficient routing algorithms and many problems can be solved approximately by using spanning trees. For various requirements, there are many different types of spanning trees, *e.g.*, minimum spanning trees, minimum diameter spanning trees, maximum leaf spanning trees, and so on. The minimum spanning tree problem is one of the most typical and well-known problems. It has applications in the computer and communication network design, wiring connections, telephone networks, transportation network linking, and so on [4, 5, 6, 16].

For network communication systems, messages are frequently routed along a minimum diameter spanning tree of the network [12]. The diameter of a graph is the longest shortest path among all possible shortest paths of the graph. The minimum diameter spanning tree problem is to find a spanning tree of G with the minimum diameter among all possible spanning trees. To solve this problem, a simple algorithm is to find the center c of G by using the Kariv-Hakimi algorithm [2, 3, 17], and then obtain a spanning tree by doing breadth-first search algorithm from c [9]. Figures 1 and 2 show an example of the graph G_1 and its minimum diameter spanning tree, respectively.

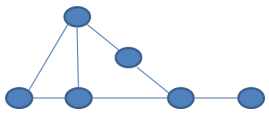

Figure 1: The graph G_1.

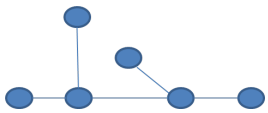

Figure 2: A minimum diameter spanning tree of G_1.

The maximum leaf spanning tree problem is to find a spanning tree whose number of leaves is maximum among all possible spanning trees. This problem has been widely studied and is known to be an NP-complete problem [7]. This problem has applications in networking [8], circuit layout [14] and an interesting application mentioned in [13]. Since this problem is NP-complete, several approximation algorithms have been proposed. In [15], Solis-Oba gave a linear-time 2-approximation algorithm. Also Gakuen-nishimachi and Nishi-ku proposed a branch-and-bound algorithm in [11]. Li and Toulouse try to solve this problem on grid graphs and make some contribution [10]. They showed that for an $n \times m$ grid graph where $n < 4$, the maximum number of leaves of a spanning tree is known. They also showed how to construct these maximum leaf spanning trees. They gave some constructions of spanning trees with a large number of leaves, and then showed the construction for grid graphs with 6 rows is optimal. There is a directed version of this problem [1]. Figures 3 and 4 show an example of graph G_2 and its maximum leaf spanning tree, respectively.

Note that in some cases, like the graph G_1, its minimum diameter spanning tree (Figure 2) is also its maximum leaf spanning tree. In some cases, the maximum leaf spanning tree is also equal to our robust spanning tree.

Since T is a spanning tree, by adding any edge

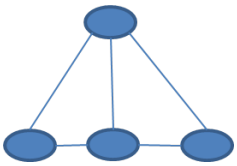

Figure 3: The graph G_2.

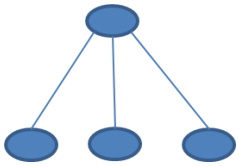

Figure 4: A maximum leaf spanning tree of G_2.

$e \in E \setminus F$ to T, there is a cycle C in the resulting graph. Let $num(e) = |C| - 1$. We say that the robustness number of T is $R(T) = \sum_{e \in E \setminus F} num(e)$. The *robustest spanning tree problem* is to find a spanning tree T with minimum $R(T)$ among all possible spanning trees of G. We denote $R(G)$ to be the minimum $R(T)$. Figures **??**, 5, and 6 show some examples. The meaning of the robustest spanning tree is that the edges of this spanning tree are robust so that they cannot be replaced easily.

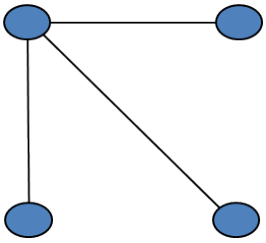

Figure 5: Spanning tree T_1 of K_4 with $R(T_1) = 6$.

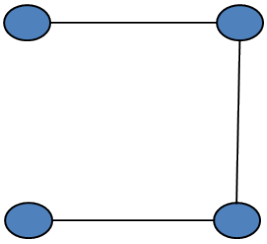

Figure 6: Spanning tree T_2 of K_4 with $R(T_2) = 7$.

When we study this problem, we find that the robustest spanning tree is very similar to the maximum leaf spanning tree in many graphs, like K_4 and K_5.

Thus, we are interested that whether these two spanning spanning tree problems are the same in any graph. However, it is not. A counter-example is shown in Figure 7. The maximum leaf spanning tree of G (Figure 9) is not the robustest spanning tree of G (Figure 8) because $R(G) = R(T_3) = 10$.

2 Block Graphs

A *block* of G is a biconnected component. A graph G is called a *block graph* if each block of G is a clique. Figure 10 shows an example. A vertex v in G is a *cut vertex* if the removal of v makes the resulting graph disconnected.

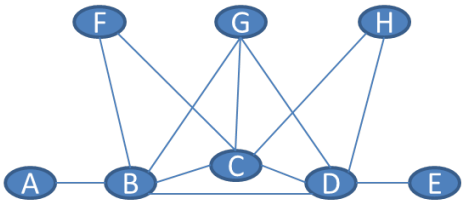

Figure 7: A graph G.

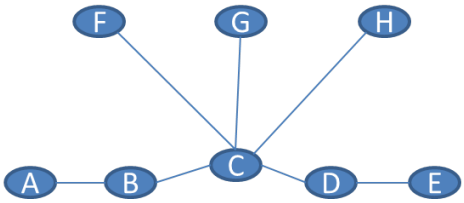

Figure 8: The robust spanning tree T_3 of G with $R(T_3) = 10$.

It is easy to check that a star is the robustest spanning tree for a clique. Thus our algorithm try to make each block of block graph G become a star in the resulting spanning tree. Our algorithm is as follows.

Algorithm *RST-Block*
Data: A block graph $G = (V, E)$
Result: A robustest spanning tree of G
Initially, all the vertices and edges are marked with 0;
Choose a vertex u arbitrarily and mark it with 1;
Initial an empty queue Q, and enqueue u into Q;
while $Q \neq \phi$ **do**
  Dequeue one vertex u from Q;
  Mark every 0-vertex v of $N(u)$ with 1 and the edges (u, v) with 1;
  if *v is a cut vertex* **then**
    Enqueue v into Q;
return the tree forming by all the 1-vertices and the 1-edges;

It is obvious that the time complexity is $O(n + m)$ where $n = |V|$ and $m = |E|$ because each vertex v will be visited at most $O(|N(v)|)$ times.

Theorem 1. *Algorithm* RST-Block *solves the robustest spanning tree problem on block graphs.*

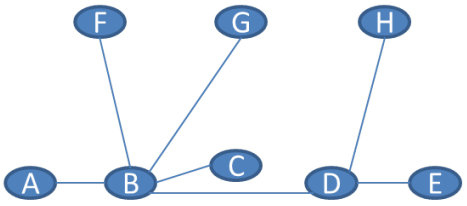

Figure 9: The maximum leaf spanning tree T_4 of G with $R(T_4) = 11$.

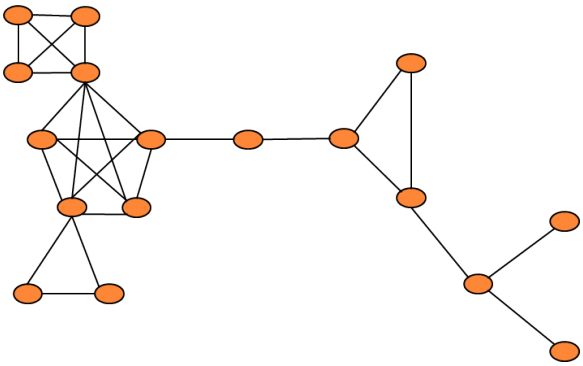

Figure 10: A block graph G.

Proof. For each block B, there are $|B|-1$ edges in the spanning tree exactly. In our algorithm, any extra edge produces a C_3 which is the best possible for an optimal solution. Therefore, our algorithm solve the problem for block graphs. □

Now, we use an example to explain our algorithm. Given a block graph shown in Figure 10, we process cliques one by one.

First, we process the first block. We choose an arbitrary vertex to be the center of star as shown in Figure 11. This star will be a part of our spanning tree.

Then we process the second and third blocks. The results are shown in Figures 12 and 13, respectively.

Note that some blocks contain only two vertex. It means that these blocks are already a star. So we can directly process the final block as shown in Figure 14.

3 Conclusion

In this paper, we propose a new problem called the robustest spanning tree problem. We design a linear-time algorithm for solving this problem on

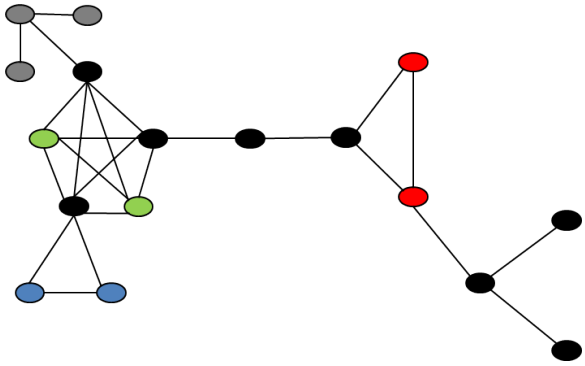

Figure 11: Form a star for the first block.

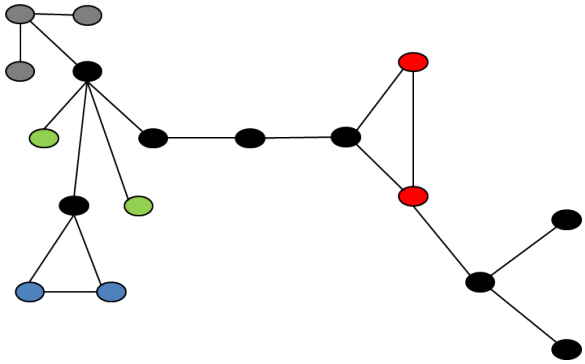

Figure 12: Form a star for the second block.

block graphs. Our algorithm is based on the process of a clique. However, we do not know how to extend it to interval graphs. Further, we conjecture that the robustest spanning tree problem is NP-hard on general graphs.

Acknowledgements

This work was partially supported by the National Science Council of Taiwan, under contract NSC 102-2221-E-259-018.

References

[1] D. Binkele-Raible and H. Fernau, A Faster Exact Algorithm for the Directed Maximum Leaf Spanning Tree Problem, *Lecture Notes in Computer Science*, Vol. 6072, pp. 328-339 (2010)

[2] R. E. Burkard , Y. Lin and G. Rote, The Obnoxious Center Problem on a Tree, *SIAM*

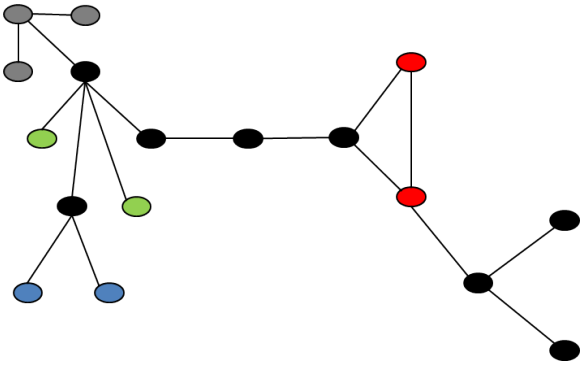

Figure 13: Form a star for the third block.

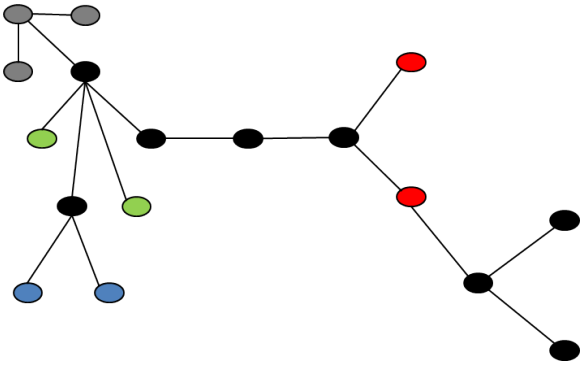

Figure 14: Form a star for the final block.

Journal on Discrete Mathematics, Vol. 14, pp. 498-509 (2001)

[3] B. Ben-moshe, B. Bhattacharya and Q. Shi, Efficient Algorithms for The Weighted 2-center Problem in a Cactus Graph, *Lecture Notes in Computer Science*, Vol. 3827, pp. 693-703 (2005)

[4] W. Chou and A. Kershenbaum, A Unified Algorithm for Designing Multidrop Teleprocessing Networks, *Proceedings of the third ACM symposium on Data communications and Data networks: Analysis and Design*, pp. 148-156 (1973)

[5] J. A. Dei Rossi, R. S. Heiser and N. S King, A Cost Analysis of Minimum Distance TV Networking for Broadcasting Medical Information, *RM-6204-NLM*, Rand Corporation (1970)

[6] L. R. Esau and K. C. Williams, On Teleprocessing System Design: Part II, *IBM Systems Journal*, Vol. 5, pp. 142-147 (1966)

[7] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, *W. H. Freeman*, NY, USA (1979)

[8] S. Guha and S. Khuller, Approximation Algorithms for Connected Dominating Sets, *Proceedings of the Fourth Annual European Symposium on Algorithms*, Vol. 20, pp. 374-387 (1998)

[9] R. Hassin and A. Tamir, On The Minimum Diameter Spanning Tree Problem, *Information Processing Letters*, Vol. 53, pp. 109-111 (1998)

[10] B. Li and M. Toulouse, Maximum Leaf Spanning Tree Problem for Grid Graphs, *Journal of Combinatorial Mathematics and Combinatorial Computing*, Vol. 73, pp. 181-193 (2010)

[11] T. Fujie, An Exact Algorithm for The Maximum Leaf Spanning Tree Problem, *Journal Computers and Operations Research*, Vol. 30, pp. 1931-1944 (2003)

[12] E. Nardelli, G Proietti1 and Peter Widmayer, Finding All the Best Swaps of a Minimum Diameter Spanning Tree Under Transient Edge Failures, *Lecture Notes in Computer Science*, Vol. 1461, pp. 55-66 (1998)

[13] C. Payan, M. Tchuente, and N. H. Xuong, Arbres avec un Nombres Maximum de Sommet Pendants , *Discrete Mathematics*, Vol. 49, pp. 267-273 (1984)

[14] J. A. Storer, Constructing Full Spanning Trees for Cubic Graphs, *Computers and Operations Research*, Vol. 13, pp. 8-11 (1981)

[15] S. Solis-Oba, 2-approximation Algorithm for Finding a Spanning Tree with Maximum Number of Leaves, *Lecture Notes in Computer Science*, Vol. 1461, pp. 441-452 (1998)

[16] R. G. Saltman, G. R. Bolotsky and Z. G. Ruthberg, Heuristic Cost Optimization of The Federal Telpaknetwork, *Tech. Note 787*, National Bureau of Standards, Washington, D.C. (1973)

[17] A. Tamir, Improved Complexity Bounds for Center Location Problems on Networks by Using Dynamic Date Structures, *SIAM Journal of Disrete Mathematics*, Vol. 1, pp. 377-396 (1988)