On the Optimum Requirement Graph Problem

Bang Ye Wu
Dept. of Computer Science and Information Engineering, Shu-Te Univeristy,
YenChau, KaoShiung, Taiwan 824, R.O.C. Email: bangye@mail.stu.edu.tw

Kun–Mao Chao
Dept. of Life Science, National Yang-Ming University,
Taipei , Taiwan 112 , R.O.C. Email: kmchao@ym.edu.tw

Chuan Yi Tang
Dept. of Computer Science, National Tsing Hua University,
Hsinchu, Taiwan, R.O.C. Email: cytang@cs.nthu.edu.tw.

Abstract

Given a requirement a_{ij} for each pair of nodes i,$j \in \{1..n\}$, the optimum requirement graph problem looks for a k-edge graph H such that the routing cost $\sum_{i<j} a_{ij} d_H(i,j)$ is minimum, where $d_H(i,j)$ is the number of edges of the shortest path between i and j on H. In this paper, we show that the problem is NP-hard and give a constant ratio approximation algorithm. Furthermore, we define a sufficient condition for the problem to be polynomial-time solvable.

1 Introduction

Finding spanning subgraphs of a given graph is a classical problem of network design. Typically, we are given a nonnegative edge-weighted graph G. The weight on each edge represents the distance and reflects both the cost to install the link (building cost) and the cost to traverse it after the link is installed (routing cost). Let $G = (\{1..n\}, E, w)$ be an undirected graph, where w is a nonnegative edge weight function. The building cost of G is $w(G) = \sum_{e \in E} w(e)$, and the routing cost of G is $c(G) = \sum_{i<j} a_{ij} d_G(i,j)$, where a_{ij} is the requirement between the two vertices, and $d_G(i,j)$ is the length of the shortest path between the two vertices. For the case where all the requirements are one, the routing cost of a graph is the sum of all distances in the graph, and was studied under various name in graph theory. For example, it was named as the transmission of a graph in [15].

When considering the building cost only, we are looking for the minimum weight spanning subgraph. The optimal solution is the *minimum weight spanning tree (MST) or forest* of the graph. However, removing any edge from a graph will increase the routing cost unless such an edge can be replaced by a path with the same or smaller length. Therefore graphs with smaller routing cost might come up with larger building cost, and we often need to make a tradeoff between the two costs.

A large number of researchers have studied the problems of finding a spanning tree with minimum weight or minimum routing cost. Efficient polynomial time algorithms for the MST were developed (for example, see [4, 7]). Hu showed how to construct a spanning tree with minimum routing cost for two special cases in [10]. When the distances between every two vertices are the same, he gave a polynomial time algorithm to find the optimal solution. When all the requirements are the same, he gave a sufficient condition for the optimal tree to be a star. The optimal tree for a graph with identical requirements is referred as a *minimum routing cost spanning tree*(MRT), or a *shortest total path length spanning tree*. In [8, 11], the MRT problem has been shown to be NP-hard , and 2-approximation algorithms were given in [5, 16]. Approximation algorithms with better error ratios were developed in [17], and a *polynomial time approximation scheme* (PTAS) for the MRT problem was presented in [18]. Some extensions of the MRT problem were also investigated. Given nonnegative weights of vertices, the product-requirement (or the sum-requirement) MRT problem assumes that the requirement between two vertices is the product (or the sum, respectively) of their weights. Constant ratio approximation algorithms for these two generalized problems were given in [19], and a PTAS for the product-requirement MRT was shown in [20].

In this paper, we consider the problems of finding spanning subgraphs with small building and routing costs. Instead of the general problem with arbitrary requirements and arbitrary distances, we shall focus on the special case that the distances $d_G(i, j)$ are all equal to one, while the requirements a_{ij} are arbitrary. We shall call the optimum graph in this case the *optimum requirement graph* (ORG). An example is illustrated in Figure 1.

Let n be the number of vertices and k the building cost constraint. If $k = n-1$, the ORG problem is equivalent to the *optimum requirement spanning tree* problem in [10], and can be optimally solved in polynomial time. Here we consider a more general case when $n-1 \le k \le n(n-1)/2$ and obtain the following results.

- The NP-hardness of the ORG problem for general k.
- A sufficient condition such that the ORG can be found in polynomial time. Interestingly, it includes the product requirement and sum requirement cases.
- An approximation algorithm with error ratio $1 + (n-1)/k$. It should be noted that this error ratio is no more than 2.

Several results for trees realizing tradeoffs between weight and some distance requirements were studied before. In [12], Khuller *et al* showed that it is possible to construct a spanning tree balancing the minimum spanning tree and the short-path tree of a specified node. The NP-hardness of finding the minimum diameter subgraph with budget constraint was established in [14], while a polynomial time algorithm for finding the minimum diameter spanning tree of a graph with arbitrary edge weights was given in [9]. Considerable work has been done on the *spanner* of a graph. In general, a t-spanner of G is a low-weight subgraph of G such that, for any two vertices, the distance on the spanner is at most t times the distance in G. Some results of finding spanner of a weighted graph can be found in [1]. Obviously, the spanners can be used to approximate the minimum routing cost subgraph problem with arbitrary requirements. But since the criteria for a spanner are often much stricter, we may often come up with better results if we wish to minimize the routing cost only.

The remaining sections are organized as follows. Some basic assumptions are given in Section 2. The NP-hardness of the ORG problem is shown in Section 3, and the approximation algorithm and polynomial-time solvable cases are in Section 4.

2 Preliminaries

In this paper, a graph is a simple and undirected graph with unweighted edge. The requirements are assumed to be nonnegative. Let $G = (V, E)$ be a graph with nodes set V and edge set E, and n be the number of vertices of G. The distance $d_G(u, v)$ between vertices u and v in G is the number of edges of the shortest path in G between them. The weight of graph G is the total number of edges in G.

Definition 1. Let $G = (\{1..n\}, E)$ be a graph and a_{ij} be the requirement between vertices i and j. The routing cost of a graph G is defined as $c(G) = \sum_{i<j} a_{ij} d_G(i, j)$.

Note that the summation is over all *unordered pairs* of vertices, i.e. the routing cost between two vertices i and j is counted only once. In some other papers, the summation is over all pairs, and the routing cost is therefore exactly twice of that defined in this paper. Since we consider only undirected graph, the two definitions are equivalent. Also, the input requirements are assumed to be symmetry , i.e. $a_{ij} = a_{ji}$ for all i and j, and only one of them appeared in the definition. If the input requirements a_{ij} is not symmetry, we can set $b_{ij} = b_{ji} = a_{ij} + a_{ji}$. Since the path is undirected, the routing cost $\sum_{ij} a_{ij} d_G(i, j)$ is the same as $\sum_{i<j} b_{ij} d_G(i, j)$.

Definition 2. Given a set of nonnegative requirements $A = \{a_{ij} | 1 \le i < j \le n\}$ and an integer k, the *optimal requirement graph*, denoted by ORG, is a graph $G = (\{1..n\}, E)$ such that the routing cost of G is minimum among all graphs with k edges, where $n-1 \le k \le n(n-1)/2$.

If the vertex set can be partitioned into two subsets such that there is no requirement cross the two subsets, we may solve the problem individually. Here we assume that there is at least one positive requirement for any cut.

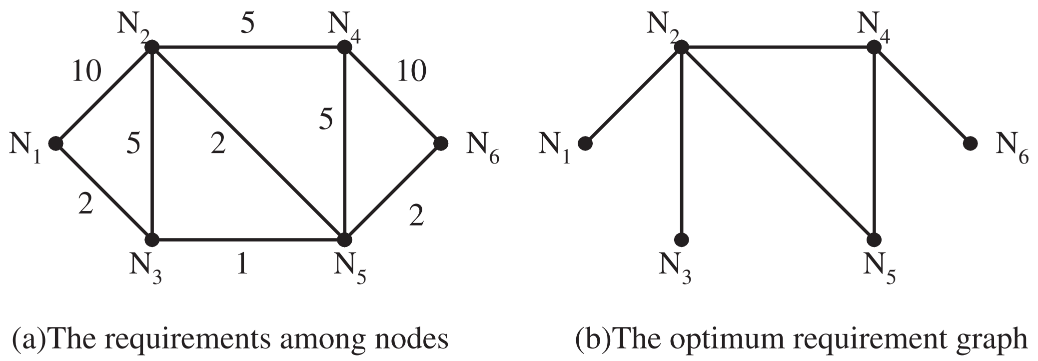

Figure 1: An instance of the optimum requirement graph problem. (a)The input requirements among nodes. Requirements not shown are assumed to be zero. (b)The optimum graph for $k = 6$. The routing cost is $(10 + 10 + 5 + 5 + 5 + 2) \times 1 + (2 + 2 + 1) \times 2 = 47$. The distances between N_1 and N_3, N_3 and N_5, N_5 and N_6 are twos; while distances between node pairs with nonzero requirements are all ones.

3 The NP-hardness

In this section, we shall show that the ORG problem is NP-hard by reducing the 2-spanner problem to it. The definition of the 2-spanner problem is as follows:

Definition 3. Let $G = (V, E)$ be a connected graph. A subgraph $H = (V, F)$, $F \subset E$, of G is a *2-spanner* of G if $d_H(i, j) \leq 2d_G(i, j)$ for all vertices i and j.

Definition 4. Given a graph $G = (V, E)$, the minimum 2-spanner problem is to find the 2-spanner with minimum number of edges. Given an integer $k \geq n - 1$, the decision version of the minimum 2-spanner problem asks if there exists a 2-spanner with no more than k edges. The decision problem is referred to as the *2-spanner problem* in this paper.

Theorem 1. The ORG problem is NP-hard even for the case that all requirements are either 1 or 0.

Proof: We transform the 2-spanner problem to the ORG problem. Given an instance of the 2-spanner problem by the graph $G = (V, E)$ and an integer k, we construct an instance of the ORG problem as follows. For any edge $(i, j) \in E$, we set the requirement $a_{ij} = 1$ and $a_{ij} = 0$ otherwise. The edge constraint of the ORG problem is set to k. We claim that the ORG problem has a solution with routing cost at most $2a^* - k$ if and only if there is a 2-spanner of G with k edges, where $a^* = \sum_{i<j} a_{ij}$ is the total requirement.

Suppose that $H = (V, F)$ is a 2-spanner of G and $|F| = k$. Consider H as a solution of the ORG problem. Every edge in H corresponds to one requirement. There are k requirements are directly routed by one edge. Since H is a 2-spanner, the other requirements are routed by two edges. Consequently the routing cost of H is $k + 2(a^* - k) = 2a^* - k$. Conversely, suppose that the ORG problem has a solution H with routing cost $2a^* - k$. For any k-edge graph, there are at most k requirements can be routed by only one edge and the others needs at least two edges. The lower bound of the routing cost of a k-edge subgraph is $2a^* - k$, and such a routing cost is achieved only when k requirements are routed by one edge and all others by two edges. Therefore every edge in H is also an edge in G and $d_H(i, j) = 2$ for any edge (i, j) in G but not in H. It also implies that H is a 2-spanner of G. Since the 2-spanner problem is NP-complete [3, 13], by the above reduction, the ORG problem is NP-hard. □

4 Polynomial-time algorithms

In the following, we shall define a sufficient condition that the ORG problem can be solved in polynomial time.

Definition 5. For a set of requirements $\{a_{ij}\}$, a vertex m is a heavy vertex if $a_{im} \geq a_{ij}$ for all vertices i and j.

Theorem 2. The optimal requirement graph problem can be solved optimally in $O(n^2)$ time if the input contains a heavy vertex.

Proof: Without loss of the generality, we assume that vertex n is the heavy vertex. Let $Y = \{a_{in}|1 \leq i < n\}$ and X be the subset of the largest $k-(n-1)$ requirements of the set $\{a_{ij}|1 \leq i < j \leq n-1\}$. Let a^* be the total requirement, and x and y be the total requirement in X and Y respectively. For a graph with k edges, there are only k requirements may be routed directly and others will be routed by at least two edges. Since there is at least one edge incident to each vertex and n is the heavy vertex, for any graph G with k edges, the routing cost $c(G) \geq (x+y)+2(a^*-(x+y))$, and the equality holds when the requirement in $X \cup Y$ are routed directly and others are routed by exactly two edges. However, the graph with edge set $\{(i,j)|a_{ij} \in X \cup Y\}$ achieves the lower bound of the routing cost. Therefore such a graph is optimal and can be constructed in $O(n^2)$ time by a linear time algorithm [2] finding the k-th largest element in a set of $O(n^2)$ numbers. □

The next two corollaries can be proved similarly.

Corollary 3. Any graph with diameter 2 is an optimal solution of the ORG problem with identical requirements.

Corollary 4. Let A be a set of requirements over a vertex set $\{1..n\}$ and A_1 be the subset of the largest k requirements in A. Assume $H = (\{1..n\}, E)$, where $E = \{(i,j)|a_{ij} \in A_1\}$. If the diameter of H is 2, then H is the ORG.

Let q_i be a nonnegative weight of vertex i. As defined in [19], requirements are called *product requirements* (or *sum requirements*) if $a_{ij} = q_i q_j$ (or $a_{ij} = q_i + q_j$ respectively) for all vertices i and j.

Corollary 5. The ORG problem with product requirements or sum-requirements can be solved in $O(n \log n + k)$ time.

Proof: Let q_i be the given nonnegative weight of vertex i. Sort the vertices by their weights in nondecreasing order. We have $q_i \leq q_{i+1}$ for all $i < n$. The requirement matrix is a sorted matrix, *i.e.* all rows and columns are sorted. Obviously the vertex n is a heavy vertex. By theorem 2, all we need to do is to find the $k-(n-1)$ largest numbers in the set $\{q_{ij}|1 \leq i < j < n\}$. Since it is a sorted matrix, the selection can be done in $O(n)$ time by the algorithm in [6]. Then it takes $O(k)$ time to report the selected requirements , and the time complexity is $O(n \log n + k)$. □

We now turn to an approximation algorithm for the ORG problem with arbitrary requirements. The idea is to take a vertex m as if it was a heavy vertex. We connect all other vertices to it, i.e. m is the center of the constructed graph, and the remaining $k-(n-1)$ edges are added to the vertex pairs with larger requirements. Since the diameter of the constructed graph is two, the cost is within twice of that of the optimum. The next theorem gives us a slightly better result for large k.

Theorem 6. The ORG problem with arbitrary requirements can be approximated with error ratio $1+(n-1)/k$ in $O(n^2)$ time.

Proof: Let A_1 be the summation of the largest k requirements and the remaining total requirement be A_2. Obviously the routing cost of the optimal graph is no less than $A_1 + 2A_2$. Let A_3 be the summation of the largest $k-(n-1)$ requirements. For the constructed graph, since its diameter is two and the largest $k-(n-1)$ requirements are routed directly, its routing cost is no more than $A_3 + 2(A_1 - A_3 + A_2)$. Since $A_3 \geq ((k-n+1)/k)A_1$, the relative error ratio is $1+(n-1)/k$. The time complexity is dominated by the time to select the largest $k-(n-1)$ requirements. By a linear time algorithm for the selection problem, the time complexity is $O(n^2)$ since there are $O(n^2)$ requirements. □

In fact, there exists a vertex m such that there are at least $\lceil 2k/n \rceil$ of the largest k requirements incident to m. By choosing m as the center, we may get a little better ratio but it is not asymptotically better.

References

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, On sparse spanners of weighted graphs, *Discrete and Computational Geometry*, 9(1993), pp. 81–100.

[2] M. Blum, R.W. Floyd, V.R. Pratt, R.L. Rivest, abd R.E. Tarjan, Time bounds for selection, *J. Comput. Syst. Sci.*, 7(1972), pp. 448–461.

[3] L. Cai, NP-completeness of minimum spanner problems, *Discrete Applied Math.*, 48(1994), pp. 187–194.

[4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, the MIT Press, 1994.

[5] R.C. Entringer, D.J. Kleitman, and L.A. Szekely, A note on spanning trees with minimum average distance, *Bulletin Inst. Comb. Appl.*, 17(1996), pp. 71–78.

[6] G.N. Frederickson and D.B. Johnson, Generalized selection and ranking: sorted matrices, *SIAM J. Computing*, 13(1984), pp. 14–30.

[7] M.L. Fredman and R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. of the ACM*, 34(1987), pp. 596–615.

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, San Fransisco, 1979.

[9] R. Hassin and A. Tamir, On the minimum diameter spanning tree problem, *Information Processing Letters*, 53(1995), pp. 109–111.

[10] T.C. Hu, Optimum communication spanning trees, *SIAM J. Computing*, 3(1974), pp. 188–195.

[11] D.S. Johnson, J.K. Lenstra, and A.H.G. Rinnooy Kan, The complexity of the network design problem, *Networks*, 8(1978), pp. 279–285.

[12] S. Khuller, B. Raghavachari, and N. Young, Balancing minimum spanning trees and shortest-path trees, *Algorithmica*, 14(1995), pp. 305–321.

[13] D. Peleg and A.A. Schäffer, Graph spanners, *J. Graph Theory* 13(1989), pp. 99–116.

[14] J. Plesnik, The complexity of designing a network with minimum diameter, *Networks*, 11(1981), pp. 77–85.

[15] J. Plesnik, On the sum of all distances in a graph or digraph, *J. of Graph Theory*, 8(1984), pp. 1–21.

[16] R. Wong, Worst-case analysis of network design problem heuristics. *SIAM J. Algebraic Discrete Methods*, 1(1980), pp. 51–63.

[17] B.Y. Wu, K.-M. Chao, and C.Y. Tang, Approximation algorithms for the shortest total path length spanning tree problem, *Discrete Applied Math.*, 105(2000), pp. 273–289.

[18] B.Y. Wu, G. Lancia, V. Bafna, K.-M. Chao, R. Ravi, and C.Y. Tang, A polynomial time approximation scheme for minimum routing cost spanning trees, *SIAM J. Computing*, 29(2000), pp. 761–778. A preliminary version appeared in the *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, pp. 21–32, 1998.

[19] B.Y. Wu, K.-M. Chao, and C.Y. Tang, Approximation algorithms for some optimum communication spanning tree problems, *Discrete Applied Math.*, 102(2000), pp. 245–266. Partial results of this paper appeared in the *Proceedings of the Ninth Annual International Symposium on Algorithm and Computation (ISAAC'98)*, Lecture Notes in Computer Science, (Springer Verlag), Vol. 1533, pp.407-416, 1998.

[20] B.Y. Wu, K.-M. Chao, and C.Y. Tang, A polynomial time approximation scheme for optimal product-requirement communication spanning trees, *J. Algorithms*, 36(2000), pp. 182–204.