A Crossover Algebra for Solving Minimum Spanning Tree in Network Design with Diameter Constraints

Ziyi Wang¹, Kai Shi¹*, Fuqiang Qiao² and Huanran Li¹

¹School of Computer Science and Engineering, Tianjin University of Technology
²College of Software and Communications, Tianjin Sino-German University of Applied Sciences Tianjin, China

*Corresponding author email: shikai0229@tjut.edu.cn

Abstract. The minimum spanning tree problem with diameter constraint is an important problem for many applications such as network design and reliability. The diameter constraint makes it different from the common minimum spanning tree problem, it is difficult to get the optimal solution, even the feasible solution can not be obtained by the traditional method. To address this problem, in this paper, a new crossover algebra is designed to ensure that the crossover tree still satisfies the diameter constraint. During the crossover process, structure of the subtree will be retained, thus avoiding the problem that the new generated tree is not a feasible solution. Experimental results show the effectiveness of the improved algorithm.

1. Introduction

Minimum Spanning Tree (MST) is a spanning tree of a connected graph with n nodes. It is a minimal connected sub-graph of the original graph, including all n nodes in the original graph, and has the least spanning edge to keep the graph connected¹⁻³. The minimum spanning tree problem is a classic problem in network design, which can be used in various network optimization applications. In practical application, such as the computer networks, the natural gas transmission etc., the diameter of the network communication these problems will be constrained. Therefore, the minimum spanning tree problem with diameter constraint⁴⁻⁶ has been attracted a lot of attentions. The problem of diameter constraint is an NP problem⁷, and difficult to solve.

At present, the more common algorithms for solving the minimum spanning tree with diameter constraint include genetic algorithm and ant algorithm⁸, etc. Therefore, the solving problem of intelligent algorithm⁹ has attracted great attention. Genetic algorithm¹⁰ is a kind of intelligent algorithm, but they all face the problem of tree representation. This problem leads to that the tree is no longer an effective tree after crossover, and thus can not be solved.

Crossover algorithm is the important content of genetic algorithm, and also the core content of this paper. In this paper, we improve the crossover algebra to ensure that feasible solution can be obtained. In this paper, the design goal of crossover algorithm is to make the crossover process of two subtrees without changing the basic structure of spanning tree. Another innovative design proposed in this paper is to change the local search operator and design it based on the sequence coding method, which makes a significant contribution to the improvement of the effectiveness of the algorithm.
2. Relate Work

The minimum spanning tree problem with diameter constraint is a famous problem, which has a very wide application background and practical significance, and has an important application value in many fields.

When solving the problem of minimum spanning tree with diameter constraint, we can use genetic algorithm. If we want to simulate the genetic way of biology to code, we must first solve the coding problem of chromosome genes. Common gene coding methods include binary coding, real coding, etc. Compared with binary encoding, real encoding directly uses real number to represent genes, so there is no need for decoding process, and it is easier to understand. The disadvantage is that premature convergence is easy to occur, thus falling into the situation of local optimal solution. Another innovative design proposed in this paper is to change the local search algebra and design it based on the sequential coding method, which makes a significant contribution to the improvement of the effectiveness of the algorithm.

In addition, Deo and Abdalla\cite{11} proposed a one-time tree construction algorithm for the diameter constrained minimum spanning tree problem. The algorithm constructs a tree in an improved greedy way, using heuristics algorithm to select edges to be added to the tree at each stage of tree construction. Lucena and Ribeiro\cite{12} proposed a hybrid fetching heuristic algorithm based on blind descent for the minimum spanning tree problem with diameter constraints, and achieved results closer to the optimal solution in data set tests.

Raidl and Julstrom\cite{13} proposed the random greedy heuristic algorithm\cite{14}. Compared with the traditional greedy algorithm, this algorithm has the characteristics of simpler and more effective. In addition, there are many algorithms for solving the BDMST problem, such as Center-based tree Construction (CBTC)\cite{15} and improved algorithms for RGH and OTTC\cite{16}. In Euclidean space, RGH has better results than OTTC and CBTC, while in other Spaces, the results are opposite.

3. Mechanism

In order to solve the problem of minimum spanning tree with diameter constraint, a new crossover algebra is designed to ensure that the crossover tree can still satisfy the diameter constraint and avoid the problem that the traditional algorithm can not get a feasible solution. We will introduce it in the following parts.

The first thing that needs to be solved is the coding problem. The random number generation function is used to generate any set of numbers within a reasonable interval to represent the Prufer\cite{17} coded sequence of the tree. Prufer coding can transform a rootless tree into a number sequence, and the number sequence obtained by this coding method is unique, that is, the original rootless tree can be restored. The advantage of Prufer coding is that the tree after crossover process is still a tree, but the disadvantage is that the tree produced is not necessarily feasible, and the new tree is with great difference from the original tree.

Secondly, there is an important problem involved here, that is, how to ensure that the randomly generated chromosome individuals meet the diameter constraint. For this reason, the solution adopted in this paper is to design a judgment function to check the diameter constraint of chromosomes. If it is not satisfied, a new chromosome will be generated to replace the old one that does not meet the requirements, until the newly generated chromosome meets the requirements. As for the problem of finding the diameter, this paper adopts the depth-first search method. First, select a node randomly as the root node to perform a depth-first search to get the depth of each node. The maximum node depth found again is the diameter of the tree.

Then the crossover operator is designed to ensure that the basic structure of spanning tree is kept unchanged to completing the crossover of two subtrees. The process of cross function is relatively complex, and two spanning trees need to be processed first. Set the number of nodes as \( n \). For chromosome A =\{0, 4, 1, 1, 0, 3\} and chromosome B =\{1,1,0,2,2\}, it is first transformed into a tree. The specific structure of the tree is shown in the figure 1 and figure 2.
Through the Prüfer number, the chromosome array can be transformed into the edge sequence pair array of the tree. A two-dimensional array is used to store the edge order pairs of the two trees, and then the edge order pairs of the trees are converted into an adjacency matrix, as shown in the tables 1-4.

**Table 1.** Edge order pairs of tree a.  **Table 2.** Edge order pairs of tree b.

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 2 | 0 | 3 | 1 |
| 1 | 4 | 1 | 5 | 4 | 6 | 3 |

**Table 3.** Matrix of tree a.  **Table 4.** Matrix of tree b.

$$
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

The next step is to find the size and depth of each node of the two trees, and record the parent-child relationship between nodes, which are stored in four arrays respectively, as show in the tables 5 - 8.

**Table 5.** Size of node.  **Table 6.** Depth of node.

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| a | 3 | 2 | 0 | 1 | 1 | 0 |
| b | 2 | 1 | 1 | 0 | 0 | 0 |

**Table 7.** The parent node of each node in tree a.  **Table 8.** The parent node of each node in tree b.

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 2 | 0 | 3 | 0 |
| 1 | 4 | 1 | 5 | 4 | 6 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 3 |

**Table 9.** Array of tree a.  **Table 10.** Array of tree b.

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 4 | 1 | 0 | 0 |
| 1 | 2 | 6 | 0 | 5 | 0 | 5 |
| 2 | 3 | 5 | 1 | 4 | 0 | 3 |
The tree selected as the root node has a parent node set to MAX to indicate that it does not exist. Then solve the problem of finding the root node of the subtree for crossover. We start by finding nodes of the same size in both trees. The diameters of the subtrees with these nodes as the root nodes are then calculated separately. The two nodes with the most similar diameters are selected from the two arrays and recorded. We then generate the array we need to swap subtrees, and since we know each node and its parent, we can find out how many children each node has. We use an array to record the number of nodes in the subtree and the number of children of each node. Note when recording that subsequent operations must be performed in the order of breadth traversal. The array is shown in the tables 9-10. These two nodes are the root nodes of the subtree used for crossover, and the size of the node is the size of the subtree used for crossover. Depending on whether the diameters found are the same or not, different crossover methods are selected. If the diameter size is the same, the swap operation is carried out directly. If the diameter sizes are different, the conversion from large diameter to small diameter can be done directly, but the conversion from small diameter to large diameter needs to be improved. The improvement is to connect the extra diameter to the root node of the subtree. Then, the specific crossover process is introduced. In the crossover process, the number of nodes in the subtree and the parent node of the root node of the subtree should not be changed. Firstly, the first subtree is operated to re-establish the association relation of the node of the subtree, and the root node of the subtree is linked to the parent node, that is, node 1 is linked to node 0 in the first adjacency matrix. Use the degree of the second subtree to update: take out the first node 1 in the first subtree, and read out the first degree 3 of the second subtree. Connect node 1 to the next 3 nodes in the first subtree that are not connected, namely connections 1 and 4, 5, 6. Then, take out the second node 4 in the first subtree, read out the second degree 0 in the second subtree, degree 0 is no child node, end the operation. At this time, all the nodes of the first subtree have been reconnected, and the establishment of the subtree has been completed. Similarly, the second subtree is constructed, and the root node of the subtree is associated with the parent nodes. Use the degree of the first subtree to update: take out the first node 1 in the second subtree, and read out the first degree 2 of the first subtree. Connect node 1 to the next 2 nodes in the first subtree that are not connected, namely connections 1, 3 and 4. Then we continue to extract the second node 3 in the second subtree and read the second degree 1 in the first subtree. If 3 and 5 are connected, the diameter of the subtree will become larger, so connect node 5 directly to the root node 1 of the second subtree. At this point, all the nodes of the second subtree have been reconnected, and the establishment of the subtree has been completed.

The structure of the transformed tree is shown in the figure 3 and 4.

![Figure 3. The structure of tree a.](image)

![Figure 4. The structure of tree b.](image)

**4. Experiment Result**

The number of iterations for this test was 4000, the number of nodes was 30, the population was 200, and the maximum diameter was 20. Convergence times represent the algebra at which the function value last converges. Running time is the accumulated running time of the program. Running time records the number of CPU clocks timing units. Figure 5 depicts the results of a data test with different diameter constraints for a tree with 30 nodes. It can be seen that different diameter constraints have a certain degree of influence on the minimum spanning tree weight. Overall, the data tended to be stable, but with the enhancement of the diameter constraint, the value of the optimal function increased. This is determined by the nature of the CRD dataset, because the spanning tree usually has a larger diameter with a smaller weight in the CRD
dataset. Therefore, strengthening the diameter constraint will affect the optimal function value of spanning tree and make the optimal function value larger.

**Figure 5.** Optimal function values under different diameter constraints.

Figure 6 is an illustration of the optimal function values tested in this paper with different number of nodes, that is, the minimum spanning tree weights. It can be seen clearly from the figure that for 10 groups of data sets with the same node size. Their test results have relatively high stability and will not fluctuate greatly. This can also indirectly prove that the results of the algorithm are credible and relatively stable.

Figure 7 shows the cumulative running time of 10 groups of test data using this algorithm. It is obvious that the single running time of the algorithm is relatively stable. There is a linear relationship between the cumulative running time and the number of running data sets, and the overall trend of steady growth is presented. Moreover, the running time of the algorithm in the test increases obviously with the diameter constraint strengthened, which proves that it takes more time to ensure the diameter constraint of the tree.

Figure 8 is a description of the cumulative elapsed time of tests with 50 nodes and 70 node trees respectively, when the diameter constraints are all set to 30. As a whole, it shows a linear correlation and a steady growth trend. From another point of view, it can be proved that the running time of the algorithm increases obviously with the enhancement of the diameter constraint.

**Figure 7.** Cumulative running time under different diameter constraints.

**Figure 8.** Cumulative elapsed time for data sets of different node sizes.

5. Conclusion

In this paper, the network reliability algorithm based on diameter constraint minimum spanning tree is studied. The genetic algorithm is used to solve the minimum spanning tree with diameter constraint. The solution of the minimum spanning tree problem and the genetic algorithm are connected, and the design task of the algorithm is completed. Multiple data sets with the number of nodes of 30, 50 and 70 were tested, and corresponding test results were obtained. Finally, the results were classified, and the influences of different diameter constraints and different number of nodes on the results were analyzed.

**Acknowledgements**

This work was supported in part by a Tianjin Science and Technology Project under grant No. 19JCTPJC43300.
References

[1] Nana Wang and Wei Liu. Application of extension analytic hierarchy process in spanning tree. Journal of Physics: Conference Series, 1624:032031, Oct 2020.

[2] Jing Li and Zaifu Guo. Solve constrained minimum spanning tree by cross-entropy (CE) method. Journal of Physics: Conference Series, 1325:012120, Oct 2019.

[3] Jiang Xiaojuan, Zhang An, Chen Yong, Chen Guangting. Research on Minimum Spanning Tree Problem Algorithm with Constrained Internal Nodes [J]. Computer Engineering and Applications, 2017, 53(10):35-37.

[4] Wolfgang Steitz. New Heuristic Approaches for the Bounded-Diameter Minimum Spanning Tree Problem [J]. INFORMS Journal on Computing, 2015, 27(1).

[5] Bryant A. Julstrom. Greedy heuristics for the bounded diameter minimum spanning tree problem. Journal of Experimental Algorithmics, 2010, 14:1.1-1.14.

[6] Kavita Singh, Shyam Sundar. A Heuristic for the Bounded Diameter Minimum Spanning Tree Problem. 2018: 84-88.

[7] Javad Akbari Torkestani, Zahra Rezaei. A learning automata based approach to the bounded-diameter minimum spanning tree problem [J]. Journal of the Chinese Institute of Engineers, 2013, 36(6).

[8] Yan W M, Wu W M. Data Structure. 2nd ed. Beijing: Tsinghua University Press, 1992. 168-174

[9] Li Hui, Shi K, Li H et al. A New Crossover Algebra of GA for Solving the Degree Constrained Minimum Spanning Tree Problems [J]. Journal of Physics: Conference Series, 2019, 1187:042085.

[10] LIANG Ping, LIU Mingzhou. A Niche Genetic Algorithm for Multi-chromosome Crossing [J]. Computer Engineering and Applications, 2016, 52(18):162-166.

[11] Deo N, Abdalla A. Computing a Diameter Constrained Minimum Spanning Tree in Parallel[C]. Algorithms and Complexity, 2000.

[12] Lucena A, Ribeiro C, Santos A. A hybrid heuristic for the diameter constrained minimum spanning tree problem [M]. 2010.

[13] Günther R. Raidl, Julstrom B. Greedy Heuristics and an Evolutionary Algorithm for the Bounded-Diameter Minimum Spanning Tree Problem [C]. Acm Symposium on Applied Computing, 2003.

[14] Edvin Berglin, Gerth Stolting Brodal. A Simple Greedy Algorithm for Dynamic Graph Orientation [J]. Algorithmica, 2020, 82(2).

[15] Huynh Thi Thanh Binh, Nguyen Xuan Hoai, R. I. McKay. A New Hybrid Genetic Algorithm for Solving the Bounded Diameter Minimum Spanning Tree Problem[J].Proceedings of IEEE World Congress on Evolutionary Computation, 2008, 24(5):3128-3134.

[16] Xuanguang male, Cheng Runwei (book).Yu xinjie, Zhou Gengui (trans.).Genetic Algorithm and Engineering Optimization [M]. Beijing: Tsinghua University Press.2003.1-10.

[17] Liu Qingxue, Hu Wei, Li Heng, Liu Xiaofeng. Application of genetic algorithm based on Prufers coding in construction of contracted tree [J]. Electronic Technology and Software Engineering, 2015(19):186.