Heuristic Algorithms for Solving Bounded Diameter Minimum Spanning Tree Problem and Its Application to Genetic Algorithm Development

Nguyen Duc Nghia and Huynh Thi Thanh Binh
Ha Noi University of Technology
Viet Nam

1. Introduction

The bounded diameter minimum spanning tree (BDMST) problem is a combinatorial optimization problem that appears in many applications such as wire-based communication network design when certain aspects of quality of service have to be considered, in ad-hoc wireless network (K. Bala, K. Petropoulos, T.E. Sterm, 1993) and in the areas of data compression and distributed mutual exclusion algorithms (K. Raymond, 1989; A. Bookstein, S. T. Klein, 1996). A more comprehensive discussion of the real-world applications of BDMST was given in Abdalla’s seminal dissertation (Abdalla, 2001).

Before the BDMST problem can be formally stated, we need some definitions relating to tree diameter and center. Given a tree \( T \), the maximal eccentricity of vertex \( v \) is the length (measured in the number of edges) of the longest path from \( v \) to other vertices. The diameter of a tree \( T \), denoted as \( \text{diam}(T) \), is the maximal eccentricity over all nodes in \( T \) (i.e the length of maximal path between two arbitrary vertices in \( T \)). Suppose that a diameter of a tree is defined by the path \( v_0, v_1, v_2, \ldots, v_{[k/2]}, v_{[k/2]+1}, \ldots, v_k \). If \( k \) is even then \( v_{[k/2]} \) is called a center of the tree. If \( k \) is odd then \( v_{[k/2]} \) and \( v_{[k/2]+1} \) are centers of the tree. In that case, the edge \( (v_{[k/2]}, v_{[k/2]+1}) \) is called a center edge.

Let \( G = (V, E) \) be a connected undirected graph with positive edge weights \( w(e) \). The BDMST problem can be formulated as follows: among spanning trees of \( G \) whose diameters do not exceed a given upper bound \( k \geq 2 \), find the spanning tree with the minimal cost (sum of the weights on edges of the trees). As in almost all studies of the BDMST problem, and without lost of generality, we will assume that \( G \) is a complete graph.

More precisely, the problem can be stated as:

Find a spanning tree \( T \) of \( G \) that minimizes

\[
\mathcal{W}(T) = \sum_{e \in T} w(e)
\]

subject to

\[
\text{diam}(T) \leq k .
\]
This problem is known to be NP-hard for \( 4 \leq k < |V|-1 \) (M.R.Garey & D.S.Johnson, 1979). In this chapter, we introduce the heuristic algorithms for solving BDMST: OTTC (Abdall, 2001), RGH (R.Raidl & B.A.Julstrom, 2003), RGH (Binh et.al, 2008a), RGH-I (A. Singh & A.K. Gupta, 2007), CBRC (Binh et.al., 2008b). In order to illustrate the effectiveness of the proposed algorithms, we apply them for initializing the population of our new genetic algorithm with multi-parent recombination operator for solving given problem. Then results of computational experiments are reported to show the efficiency of proposed algorithms.

The chapter is organized as follows. In the next section (section 2), we briefly overview works done in solving BDMST problems. Section 3 deals with new heuristic algorithm for solving BDMST problem. Section 4 describes our new genetic algorithm which uses heuristic algorithms that already presented in previous section to solve BDMST problem. The details of experiments and the comparative computational results are given and discussed in the last section of the chapter.

2. Previous work on the BDMST problem

Techniques for solving the BDMST problem may be classified into two categories: exact methods and inexact (heuristic) methods. Exact approaches for solving the BDMST problem are based on mixed linear integer programming (N.R.Achuthan et al., 1994), (L Gouveia et al., 2004). More recently, Gruber and Raidl suggested a branch and cut algorithm based on compact 0-1 integer linear programming (M. Gruber & G.R. Raidl, 2005). However, being deterministic and exhaustive in nature, these approaches could only be used to solve small problem instances (e.g. complete graphs with less than 100 nodes).

(Abdalla et al., 2000) presented a greedy heuristic algorithm - the One Time Tree Construction (OTTC) for solving the BDMST problem. OTTC is based on Prim’s algorithm in (R. Prim, 1957). It starts with a set of vertices, initially containing a randomly chosen vertex. The set is then repeatedly extended by adding a new vertex that is nearest (in cost) to the set, as long as the inclusion of the new node does not violate the constraint on the diameter of the tree. This algorithm is time consuming, and its performance is strongly dependent on the starting vertex.

Raidl and Julstrom proposed in (G.R. Raidl & B.A. Julstrom, 2003) a modified version of OTTC, called Randomized Greedy Heuristics (RGH). RGH starts from a centre by randomly selecting a vertex and keeping it as the fixed center during the search. It then repeatedly extends the spanning tree from the center by adding a randomly chosen vertex from the remaining vertices, and connecting it to a vertex that is already in the tree via an edge with the smallest weight. The obtained results showed that on Euclidean instances RGH performs better than OTTC, whereas on non-Euclidean instances the situation is reversed.

RGH could be summarized in the following pseudo-code (G.R. Raidl & B.A. Julstrom, 2003)

\[
\begin{align*}
T & \leftarrow \emptyset; \\
U & \leftarrow V; \\
v0 & \leftarrow \text{random}(U); \\
U & \leftarrow V - \{v0\}; \\
C & \leftarrow \{v0\}; \\
\text{depth}[v0] & \leftarrow 0; \\
\text{if } (\text{odd}(k)) \{ \\
& \quad v1 \leftarrow \text{random}(U); \\
\}
\end{align*}
\]
T ← {(v0, v1)};
U ← U − {v1};
C ← C ∪ {v1};
depth[v1] ← 0;
}
while (U ≠ ∅) {
    v ← random(U);
    u ← argmin {c(x, v): x ∈ C};
    T ← T ∪ {(u, v)};
    U ← U − {v};
    depth[v] ← depth[u] + 1;
    if (depth[v] < \lfloor k/2 \rfloor)
        C ← C ∪ {v};
}
return T;

Raidl and Julstrom proposed a genetic algorithm for solving BDMST problems which used edge-set coded (G.R. Raidl & B.A. Julstrom, 2003) (JR-ESEA) and permutation-coded representations for individuals (B.A. Julstrom & G.R. Raidl, 2003) (JR-PEA). Permutation-coded evolutionary algorithms were reported to give better results than edge-set coded, but usually are much more time consuming. Another genetic algorithm, based on a random key representation, was derived in (B.A. Julstrom, 2004), sharing many similarities with the permutation-coded evolutionary algorithms. In (M. Gruber & G.R. Raidl, 2005), Gruber used four neighbourhood types to implement variable neighbourhood local search for solving the BDMST problem. They are: arc exchange neighbourhood, level change neighbourhood, node swap neighbourhood, and center change level neighbourhood. Later, (M. Gruber et al., 2006), re-used variable neighbourhood searches as in (M. Gruber & G.R. Raidl, 2005), embedding them in Ant Colony Optimization (ACO) and genetic algorithms for solving the BDMST problem. Both of their proposed algorithms (ACO and GA) exploited the neighbourhood structure to conduct local search, to improve candidate solutions. In (Nghia & Binh, 2007), Nghia and Binh proposed a new recombination operator which uses multiple parents to do the recombination in their genetic algorithm. Their proposed crossover operator helped to improve the minimum and mean weights of the evolved spanning trees. More recently, in (A. Singh & A.K. Gupta, 2007), Alok and Gupta derived two improvements for RGH heuristics (given in (G.R. Raidl & B.A. Julstrom, 2003)) and some new genetic algorithms for solving BDMST problems (notably the GA known as PEA-I). RGH-I in (A. Singh & A.K. Gupta, 2007) iteratively improves the solution found with RGH by using level change mutation. It was shown in (A. Singh & A.K. Gupta, 2007) that RGH-I has better results than all previously-known heuristics for solving the BDMST problem. PEA-I employs a permutation-coded representation for individuals. It uses uniform order-based crossover and swap mutation as its genetic operators. PEA-I was shown to be the best GA of all those tried on the BDMST problem instances used in (A. Singh & A.K. Gupta, 2007). In (Binh et al., 2008a), Binh et al., also implement another variant of RGH, which is called RGH1. RGH1 is similar to RGH, except that when a new vertex is added to the expanding spanning tree, it is chosen at random, and connected to a randomly chosen vertex that is already in the spanning tree.
3. New greedy heuristic algorithm (center-based recursive clustering)

Our new greedy heuristics is based on RGH in (G.R. Raidl & B.A. Julstrom, 2003) and NRGH in (Nghia and Binh, 2007), called CBRC. We extend the concept of center to every level of the partially constructed spanning tree. The algorithm can be seen as recursively clustering the vertices of the graph, in that every in-node of the spanning tree is the center of the subgraph composed of nodes in the subtree rooted at this node. It is inspired from our observation (and other such as in (A. Abdalla et.al, 2000), (G.R. Raidl and B.A. Julstrom, 2003) that good solutions to the BDMST problem usually have “star-like structures” as can be seen (for a Euclidean graph) in Figure 1.

In a star-like structure, the vertices of the graph are grouped in clusters, and the clusters are connected by a link between their centers. Pseudocode for the new heuristic based on this observation, known as Center-Based Recursive Clustering (CBRC), is presented below:

1. \( T \leftarrow \emptyset; \)
   \( v_s \leftarrow \text{Choose a Center}(V) \)
   \( U \leftarrow V - \{v_s\}; \)
   \( C \leftarrow \{v_s\}; \)
   \( \text{depth}[v] \leftarrow 0; \)
   If \( k \) is odd then
   \{ \( v_i \leftarrow \text{Choose a Center}(U) \)
   \( T \leftarrow \{(v_s, v_i)\}; \)
   \( U \leftarrow U - \{v_i\}; \)
   \( C \leftarrow C \cup \{v_i\}; \)
   \( \text{depth}[v_i] \leftarrow 0; \)
   \} \( 2. //\text{Group vertices in } U \text{ into cluster(s)} \)
   //with centers at \( v_s \) or \( v_i \)
   For each node \( w \) in \( U \) do
   \{ \( \text{If } k \text{ is even then} \)
   \{ \( w \text{ becomes child of } v_s; \)
   \( \text{depth}[w]=1; \)
   \( T \leftarrow T \cup \{(w, v_s)\}; \)
   \} \( \text{Else} // k \text{ is odd} \)
   \( \text{If Distance}(w, v_s) \leq \text{Distance}(w, v_i) \text{ then} \)
   \{ \( w \text{ becomes child of } v_s; \)
   \( \text{depth}[w]=1; \)
   \( T \leftarrow T \cup \{(w, v_s)\}; \)
   \} \( \text{Else} \)
   \{ \( w \text{ becomes child of } v_i; \)
   \( \text{depth}[w]=1; \)
   \( T \leftarrow T \cup \{(w, v_i)\}; \)
   \} \) //end for
3. Loop
\( V \) = set of leaves in \( U \) with depths < \( \lfloor k/2 \rfloor \);
\( v \) = \( \text{Choose}_a\_\text{Center}(V) \);
if (\( v \) is empty)
    Break; // Jump out of the loop
\( U = U - \{v\} \);
For each leaf node \( w \) in \( U \) do
{
    If Distance(\( w, v \)) \leq \text{Distance}(w, \text{parent}(w)) \) then
        \( w \) becomes child of \( v \);
        \( \text{depth}[w] = \text{depth}[v] + 1; \)
        \( T = T - \{(w, \text{parent}(w))\} + \{(w, v)\} \);
}

The algorithm above is a general framework for \( \text{CBRC} \). It employs two abstract functions, namely, \( \text{Choose}_a\_\text{Center} \) and \( \text{Distance} \). The implementations of these functions are expected to affect the performance of the heuristics, and the best choice could depend on the problem instance. We propose below some possible implementations of these two functions.

Fig. 1. A “star-like” structure of a typical solution to the BDMST problem.

Implementations of \( \text{Choose}_a\_\text{Center} \) function:
- \( v \) is a center of \( U \) if \( \sum_{w \in U} \text{Distance}(v, w) \to \min \). If there is more than one such \( v \) then choose from them randomly.
- Rank all vertices in \( U \) according to \( \sum_{w \in U} \text{Distance}(v, w) \), then choose \( v \) randomly from the first \( h\% \) of the vertices.
- Conduct \( h\)-tournament selection, \( \sum_{w \in U} \text{Distance}(v, w) \) as the vertex for \( v \).
- Choose \( v \) randomly (i.e. it does not depend on Distance at all).

Implementations of the Distance function:
- \( \text{Distance}(u, v) = c(u, v) \).
- \( \text{Distance}(u, v) = \text{cost the of shortest path between } u \) and \( v \) (used for Non-Eclidean graphs).

It can be seen from the pseudo-code of \( \text{CBRC} \) that none of the combinations of \( \text{Distance} \) and \( \text{Choose}_a\_\text{Center} \) from the above implementations increase the asymptotic computational complexity of the heuristic to more than \( O(n^2) \). It is also possible to apply post-
improvement, as proposed in (A. Singh and A.K. Gupta, 2007) to CBRC just as for RGH. The resulting heuristic is known as CBRC-I. In the next section, CBRC is tested on some benchmark Euclidean instances of the BDMST problem.

4. Proposed genetic algorithm

Genetic algorithm has proven effective on NP-hard problem. Much works research on NP-hard problem, particularly in problems relating to tree have been done. Several studies proposed representations for tree (J.Gottlieb et al., 2000), (G.R.Rafl & B.A.Julstrom, 2003), (B.A.Julstrom & G.R.Raild, 2003), (B.A.Julstrom, 2004), (Martin Gruber et al., 2006), (Franz Rothlauf, 2006). This section presents the genetic algorithm for solving BDMST problem.

4.1 Initialization

Use OTTC, RGH₁, CBRC, RGH heuristic algorithms described above for initializing population and edge list for chromosome code.

4.2 Recombination operator

Using k-recombination operator as in (Nghia and Binh, 2007).

4.3 Mutation operator


5. Computational results

5.1 Problem instances

The problem instances used in our experiments are the BDMST benchmark problem instances used in (G.R. Rafl & B.A. Julstrom, 2003), (A. Singh & A.K. Gupta, 2007), (Nghia & Binh, 2007), (Binh et al., 2008a). They are Euclidean instances. All can be downloaded from http://www.sc.snu.ac.kr/~xuan/BDMST.zip. Euclidean instances are complete random graphs in the unit square. We chose the first five instances of each problem size on Euclidean instances (number of vertices) \( n = 100, 250, 500, \) and 1000, the bounds for diameters being 10, 15, 20, 25 correspondingly (making up 20 problem instances in total).

5.2 Experiment setup

We created two sets of experiments. In the first set of experiment, we compare the performance of the heuristic algorithms: OTTC, RGH, RGH₁, CBRC. The detail of the comparison between other heuristic algorithm for solving BDMST problem such as CBTC, RGH-I, CBRC-I can be referred to (Binh et al., 2008b), (A. Singh and A.K. Gupta, 2007).

There are several heuristic algorithms for solving BDMST problem as mentioned above but no research has concerned with their effectiveness in application to develop hybrid genetic algorithm. Therefore, in second set of experiment, we will try to fix this problem.

In the second set of experiment, we tested six genetic algorithm algorithms for solving BDMST problem. All of the genetic algorithms use recombination and mutation operator mentioned in section 4 but initialized by different heuristic algorithm. \( GA₁, GA₂, GA₃ \) uses
CBRC, OTTC, RGH₁ algorithm correspondent for initializing the population. GA₄ uses CBRC, OTTC, RGH₁, RGH for initialization the population with the same rate for each heuristic. GA₅ uses RGH₁, CBRC for initializing, the rate of them in the population are 30 and 70. GA₆ uses RGH₁, OTTC, CBRC for initializing, the rate of them in the population are 35, 35 and 30.

<table>
<thead>
<tr>
<th></th>
<th>GA₁</th>
<th>GA₂</th>
<th>GA₃</th>
<th>GA₄</th>
<th>GA₅</th>
<th>GA₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBRC</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>25%</td>
<td>70%</td>
<td>30%</td>
</tr>
<tr>
<td>RGH</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>25%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>OTTC</td>
<td>0%</td>
<td>100%</td>
<td>0%</td>
<td>25%</td>
<td>0%</td>
<td>35%</td>
</tr>
<tr>
<td>RGH₁</td>
<td>0%</td>
<td>0%</td>
<td>100%</td>
<td>25%</td>
<td>30%</td>
<td>35%</td>
</tr>
</tbody>
</table>

Fig. 2. The rate of the heuristic algorithms use for initialization of the population in each experiment genetic algorithm

5.3 System setting
In the first experiment, the system was run 300 times for each instances. In the second experiment, the population size for GA₁, GA₂, GA₃, GA₄, GA₅, GA₆ was 100. The number of generations was 500. All GAs populations used tournament selection of size 3 and crossover rate of 0.5. The mutation rates for center level change, center move, greedy edge mutation, and subtree optimize mutation were 0.7, 0.2, 0.8, and 0.5 respectively.

Each system was allocated 20 runs for each problem instance. All the programs were run on a machine with Pentium 4 Centrino 3.06 GHz CPU using 512MB RAM.

5.4 Results of computational experiments
The experiment shows that:
- Figure 3, 4, 5, 6, 7, 8, 9, 10, 11 show that the proposed heuristic algorithm, called CBRC have the best result than RGH, OTTC, RGH₁. It means that the solution found by CBRC algorithm is the best solution in comparison with the other known heuristic algorithm for solving BDMST problem on all the instances with \( n = 100, 250, 500 \) and 1000 (\( n \) is the number of vertices).
- Figure 15 shows that the best solution found by GA₁ have better result about 22% than the CBRC which is used for initialization the population in GA₁ on all 20 problem instances.
- Figure 16 shows that sum up of the best solution found by GA₂ have better result about approximately four times than the OTTC which is used for initialization the population in GA₂ on all 20 problem instances.
- Figure 17 shows that sum up of the best solution found by GA₃ have better result about over 10 times than the RGH₁ which is used for initialization the population in GA₃ on all 20 problem instances.
- Figure 11 shows that sum up of the best solution found by CBRC have better result about 6.5 times than the OTTC and 17 times than RGH₁ while the the figure 18 shows that sum up of the best solution found by GA₁ have better result about 0.8% times than the GA₂ and approximately 2% than GA₃.
Fig. 3. The best solution found by the heuristics: OTTC, RGH and CBRC on the problem instance with \( n = 250 \) and \( k = 15 \), test 1:
(a) OTTC, weight=42.09; (b) RGH, weight=15.14; (c) CBRC, weight = 13.32.

Fig. 4. The best solution found by the four heuristics: CBRC, RGH, OTTC, RGH\(_1\) on the problem instance with \( n = 100 \) and \( k = 10 \).
Fig. 5. The best solution found by the four heuristics: CBRC, RGH, OTTC, RGH₁ on the problem instance with \( n = 250 \) and \( k = 15 \).

Fig. 6. The best solution found by the four heuristics: CBRC, RGH, OTTC, RGH₁ on the problem instance with \( n = 500 \) and \( k = 20 \).

- Figure 18 shows that among \( GA_1, GA_2, GA_3, GA_4, GA_5, GA_6 \), sum up of the best solution found by \( GA_6 \) have bestest result than the other, otherwise \( GA_3 \) have worsest result.
- Figure 19 shows that \( GA_1 \) have smallest sum of standard deviation otherwise \( GA_3 \) have largest sum of standard deviation.
- Figure 20 shows that among \( GA_1, GA_2, GA_3, GA_4, GA_5, GA_6 \), the number of instances found best result by \( GA_5 \) and \( GA_6 \) are biggest otherwise the number of instances found best result by \( GA_2 \) and \( GA_3 \) are smallest.
Fig. 7. The best solution found by the four heuristics: CBRC, RGH, OTTC, RGH₁ on the problem instance with \( n = 1000 \) and \( k = 25 \).

Fig. 8. Comparision of the best solution found by the four heuristics: CBRC, RGH, OTTC, RGH₁ on all the problem instance with \( n = 100 \) (5 instances), \( k = 10 \).
Heuristic Algorithms for Solving Bounded Diameter Minimum Spanning Tree Problem and Its Application to Genetic Algorithm Development

Fig. 9. Comparison between the best solution found by the four heuristics: CBRC, RGH, OTTC, RGH₁ on all the problem instance with \( n = 250 \) (5 instances), \( k = 15 \)

Fig. 10. Comparison between the best solution found by the four heuristics: CBRC, RGH, OTTC, RGH₁ on all the problem instance with \( n = 500 \) (5 instances), \( k = 20 \)
Fig. 11. Comparision between the best solution found by the four heuristics: CBRC, RGH, OTTC, RGH₁ on all the problem instance with $n = 1000$ (5 instances), $k = 25$

Fig. 12. The best solution found by the CBRC and GA₁ on all the problem instance with $n = 250$, $k = 15$
Fig. 13. The best solution found by the OTTC and GA₂ on all the problem instance with $n = 250, k = 15$

Fig. 14. The best solution found by the RGH₁ and GA₃ on all the problem instance with $n = 250, k = 15$
Fig. 15. Sum up the best solution found by the CBRC and GA\textsubscript{1} on all the problem instances (20 instances)

Fig. 16. Sum up the best solution found by the OTTC and GA\textsubscript{2} on all the problem instances (20 instances)
Fig. 17. Comparison between the best solution found by the RGH1 and GA3 on all the problem instances (20 instances)

Fig. 18. Comparison between the best solution found by GA1, GA2, GA3, GA4, GA5, GA6 on all the problem instance (20 instances)
Fig. 19. Comparison between the standard deviation of the solution found by GA1, GA2, GA3, GA4, GA5, GA6 on all the problem instance (20 instances)

Fig. 20. Number of instances found best result by GA1, GA2, GA3, GA4, GA5, GA6 on all the problem instance (20 instances)
6. Conclusion

We have introduced the heuristic algorithm for solving BDMST problem, called CBRC. The experiment shows that CBRC have best result than the other known heuristic algorithm for solving BDMST problem on Euclidean instances. The best solution found by the genetic algorithm which uses best heuristic algorithm or only one heuristic algorithm for initialization the population is not better than the best solution found by the genetic algorithm which uses mixed heuristic algorithms (randomized heuristic algorithm and greedy randomized heuristic algorithm). The solution found by the genetic algorithm which uses mixed heuristic algorithm for initialization always is the best result.

7. References


Huynh Thi Thanh Binh, Nguyen Xuan Hoai, R.I Ian McKay (2008a), “A New Hybrid Genetic Algorithm for Solving the Bounded Diameter Minimum Spanning Tree Problem”, Proceeding of IEEE World Congress on Computational Intelligence, Hong Kong, LNCS

Each chapter comprises a separate study on some optimization problem giving both an introductory look into the theory the problem comes from and some new developments invented by author(s). Usually some elementary knowledge is assumed, yet all the required facts are quoted mostly in examples, remarks or theorems.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following: