Huynh Thi Thanh Binh, Nguyen Xuan Hoai, and R.I McKay

# A New Hybrid Genetic Algorithm for Solving the Bounded Diameter Minimum Spanning Tree Problem

***Abstract*— In this paper, a new hybrid genetic algorithm – known as HGA – is proposed for solving the Bounded Diameter Minimum Spanning Tree (BDMST) problem. We experiment with HGA on two sets of benchmark problem instances, both Euclidean and Non-Euclidean. On the Euclidean problem instances, HGA is shown to outperform the previous best two Genetic Algorithms (GAs) reported in the BDMST literature, while on the Non-Euclidean problem instance, HGA performs comparably with these two GAs.**

## I. Introduction

The bounded diameter minimum spanning tree (*BDMST*) problem is a combinatorial optimization problem that arises in many applications such as design of wire-based communication networks under quality of service requirements, in ad-hoc wireless networks [3], and in data compression and in distributed mutual exclusion algorithms [2, 6]. A more comprehensive discussion of the real-world applications of BDMST was given in Abdalla's seminal dissertation [10].

Before the BDMST problem can be formally stated, we need to define some concepts 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 $diam(T)$, 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 $v_1, v_2, \ldots, v_{\lfloor k/2 \rfloor}, v_{\lfloor k/2 \rfloor+1}, \ldots, v_k$. If $k$ is even then $v_{\lfloor k/2 \rfloor}$ is called a center of the tree. If $k$ is odd then $v_{\lfloor k/2 \rfloor}$ and $v_{\lfloor k/2 \rfloor+1}$ are centers of the tree. In that case, the edge ($v_{\lfloor k/2 \rfloor}$, $v_{\lfloor k/2 \rfloor+1}$) 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 $D \geq 2$, 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.

That is, we can formulate the problem as:

Find a spanning tree T = (V, E ) of G that minimize

subject to

$$diam(T) \leq D.$$

This problem is known to be *NP-hard* for $4 \leq D \leq |V|-1$ [1]. Moreover, the BDMST problem has been shown to be also approximate-hard, in that there is no polynomial time algorithm which could guarantee to find a solution that has a cost within log(|V|) of the optimum [22]. Therefore, heuristic and meta-heuristic techniques are currently the only method for improving the solution quality in solving the BDMST problem, especially when n is large.

In this paper, we introduce a new hybrid genetic algorithm (*HGA*) for solving BDMST problems. The new genetic algorithm use a multi-population, where each population is initialized with a different well known heuristic. The individuals in each population will subsequently compete for positions in a selection population, using a simulated annealing mechanism based on proportionate selection; in the selection population, they will combine and evolve toward the optimum. We compare our results with two other genetic algorithms on the same problem, namely, the genetic algorithm in [12] of Raidl and Julstrom (called RJ-ESEA in this paper), and the genetic algorithm of Alok and Gupta in [21] (called PEA-I).

The paper is organized as follows. In the next section (section 2), we briefly overview work done in solving BDMST problems, highlighting the motivations for our work. Section 3 contains the description of our new hybrid genetic algorithm for the BDMST problem. The details of our experiments are given in section 4, while the computational and comparative results are given and discussed in section 5 of the paper. The paper concludes with section 6, where we also describe some possible future extensions of this work.

## II. 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 [5, 15]. More recently, Gruber and Raidl suggested a branch and cut algorithm based on compact 0-1 integer linear programming [16]. 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. [8] presented a greedy heuristic algorithm, the *One Time Tree Construction* (OTTC) for solving the

Huynh Thi Thanh Binh is with the Department of Information Technology, Hanoi University of Technology, 1st Dai Co Viet St., Hanoi, Vietnam. E-mail: BinhHT@it-hut.edu.vn

Nguyen Xuan Hoai and R.I. McKay are with the school of Computer Science and Engineering, Seoul National University, Korea. E-mails: nxhoai@cse.snu.ac.kr; rim@cse.snu.ac.kr

*BDMST* problem. OTTC is based on Prim's algorithm in [23]. 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 [21].

Raidl and Julstrom proposed in [12] a modified version of OTTC, called *Random 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.

Raidl and Julstrom proposed a genetic algorithm for solving *BDMST* problems which used edge-set coded [12] and permutation-coded representations for individuals [13]. 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 [14], sharing many similarities with the permutation-coded evolutionary algorithms. In [17], 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, Raidl and Julstrom [18] re-used variable neighbourhood searches as in [17], 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, [20], 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 [20].

More recently, in [21], Alok and Gupta derived two improvements for RGH heuristics (given in [12]) and some new genetic algorithms for solving BDMST problems (notably the GA known as PEA-I). RGH-I in [21], denoted by $RGH_1$ in this paper, iteratively improves the solution found with RGH by using level change mutation. It was shown in [21] 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 [21]. In this paper, we implement another version of RGH, called $RGH_2$, $RGH_2$ 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.

Almost all genetic algorithms for solving the BDMST problem discussed above strongly depend on their particular heuristics, in that the heuristics were usually used to initialize GA populations and played an important role in the design of genetic operators (especially for mutation operators). However, it has been suggested in the literature that the behaviours of different heuristics vary over different classes of problem instances [21]. Therefore, our research approach is to build a new multiple-population GA, to employ different initial biases by using different heuristics for initialization, and to hybridize the individuals from these populations to promote the exploratory capacity of the GA. In the next section, we will outline such a GA.

## III. PROPOSED HYBRID GA

Genetic algorithms (GAs) have proven to be effective tools for solving approximately *NP-hard* problems. Several GAs have been proposed for BDMST problems, notably, RJ-ESEA [12], JR-PEA [13], J-GPEA [14], and PEA-I [21] – (the names we used here are borrowed from [21] and each of the GAs has two versions, namely steady-state and generational). The above GAs differ from each other in individual representation and genetic operators, but have a common property that they use a single population based on a single heuristic. Our new hybrid GA employs multiple populations and is described as follows.

### *A. Individual Representations*

The problem of spanning tree representation has been studied extensively in the literature. [4, 9] and specially [19] contain substantial discussion and analysis of different representations from theoretical and practical perspectives. For the BDMST problem, three representations have been used for representing the spanning trees, namely, edge-set-coded [11, 12], permutation coded [13, 21], and random key representations [14]. In this paper, our new hybrid genetic algorithm (HGA) uses edge-set-coded representation as suggested in [11] for individuals. Implementing HGA on other spanning tree representations will be one of our near-future investigations.

### *B. HGA Population Structure*

HGA uses m+1 populations consisting of m populations, one for each individual GA, and a final population, for solving the BDMST problem. In our paper, m = 4 and the populations are called $GA_1$, $GA_2$, $GA_3$, $GA_4$ and GA_final.

### *C. Initialization*

$GA_1$, $GA_2$, $GA_3$, and $GA_4$ are initialized by using four different heuristics, in our experiments they are RGH, RGH1, RGH2, and OTTC. Given their different algorithm bias in and operational behaviours as reported in the literatures [21] and found in our experiments, it is hoped that they will provide the sufficient diversity for all individual GA ($GA_1$-$GA_4$) at the beginning. For GA_final, the initialization is done by uniformly randomly copying individuals from the individual GA ($GA_1$-$GA_4$) initial

populations.

*D.Evolutionary Process*

*D.1 Selection*

The process of selecting individuals in each individual GA ($GA_1$-$GA_4$), and in GA_final, to be included in the mating pools for the next generation, is similar to the selection process of a typical GA in the literature (i.e. any typical selection mechanism such as fitness-proportionate or tournament-based could be used). However selection in HGA has two stages. In the first, individuals from individual GA are selected to migrate into the GA_final mating pool. Our primary aim in this migration is to make GA_final a more competitive population, with good and diverse individuals, selected from different individual GA populations with different initial biases. The selection mechanism for GA_final at generation t is as follows.

The proportions of immigrants from each individual GA to the GA_final at a generation is selected based on the comparative average performance of this individual GA. The exact formula for computing the proportion of immigrants from $GA_i$ to GA_final at generation t is:

$$f_i^t = e^{-\frac{diff(i,t)}{T}}$$

with diff(i, t) above defined as:

$diff(i,t) = mean_i^t - \min\{min_j^t, min\_final^{t-1}\}$, with $1 \le j \le n$

where:

$min_j^t$ : is the minimal (best) fitness of the population of $GA_j$ at generation t.

$mean_i^t$: is the average fitness of the population of $GA_i$ at generation t.

$min\_final^{t-1}$: is the minimal (best) fitness of GA_final at generation t-1.

T is a parameter which plays a similar roles to the pseudo-temperature in simulated annealing, and was set to C×t, where C is a normalized constant factor (to ensure that $f_i^t$ is in [0,1]). As the generations increase, the difference between $GA_i$ and the others is expected to decrease (as they are assumed to converge), and therefore will have less impact on determining the proportion of its migrants to GA_final. Finally, the number of individuals from $GA_i$ to migrate to GA_final is

$$num\_select_i^t = Pop\_Size_i * f_i^t$$

($Pop\_Size_i$ is the population size of $GA_i$) .

In our current implementation of HGA, the actual individuals selected from the $GA_i$ population are the $num\_select_i^t$ best individuals (that is, trucation selection). Other selection strategies will be considered in future studies.

After selection, the migrants merge with GA_final individuals from generation t-1, and the best Pop_Size (population size of GA_final) individuals will form the mating pool for GA_final generation t. They are subsequently randomly selected and combined or modified using genetic operators (crossover and mutation). Procedure Migration(t) below summarizes the above migration process:

**Procedure** *Migration*(t)
**For i = 1 to m Do**
diff(i,t) = $mean_i^t - \min\{min_j^t, min\_final^{t-1}\}$
$f_i^t = e^{-\frac{diff(i,t)}{T}}$ ;
$num\_select_i^t = Pop\_size_i * f_i^t(dif)$;
**For** $j = 1$ **to** $num\_select_i^t$ **Do**
**Select**($GA_i[j]$);
**Endfor**
**Endfor**
**Merge**( *GA_final*, $GA_i$*selected*);
*Return (Best Pop_size(GA_final) trees);*

*E. Genetic Operators*

We use the genetic operators proposed in [12]. They consist of a specialized crossover operator derived from RGH, and four specialized mutation operators namely: center exchange, level exchange, edge exchange, and subtree optimization mutation. The full detailed implementations were given in [12, 18]. The application of genetic operators to each GA population is homogeneous, in the sense that all use the same set of genetic operators with the same application rate. Future research will investigate less homogenous strategies, in which each GA population could use different operators and/or with different application rates.

## IV. EXPERIMENTS

*A. Problem Instances*

The problem instances used in our experiments are the BDMST benchmark problem instances used in [11, 12, 21]. They consist of two sets of Euclidean and Non-Euclidean instances. Euclidean instances are (complete) random graphs in the unit square. All can be downloaded from http://www.sc.snu.ac.kr/~xuan/BDMST.zip. We chose the first 5 instances of each problem size (number of vertices) n = 50, 100, 250, 500, and 1000, the bounds for diameters being 5, 10, 15, 20, 25 correspondingly (making up 25 problem instances in total).

For Non-Euclidean problem instances, five instances for each value of n= 100, 250, 500, 1000 were chosen. All of these instances are complete graphs with weights randomly generated in [0.01, 0.99]. The diameter bounds were set to 10, 15, 20, 25 respectively (making 20 problem instances in total).

*B. Experiment Setup*

We set up two sets of experiments. In the first, we compare the performance of HGA with RJ-ESEA and also PEA-I. We note that RJ-ESEA uses the same individual representation and operators as the individual and final GAs in HGA; our $GA_1$ uses RGH to create the initial population, so that it is essentially the same as RJ-ESEA, so we use the names $GA_1$ and RJ-ESEA interchangeably. We chose PEA-I

because it was reported in [21] as the best of several GAs for the benchmark problem instances that we use in our experiments. We note that PEA-I uses different individual representation (permutation based) and genetic operators from ours. In the second set of experiments, we run each individual GA ($GA_1$-$GA_4$) independently and compare with HGA, to fully confirm whether the use of multiple populations with migrations actually made a difference.

### C. System Settings

For HGA, the population sizes for $GA_1$, $GA_2$, $GA_3$, $GA_4$ and GA_final was 100, the number of generations was 200, giving 100000 as the total maximal number of fitness evaluations, i.e. the same as for RJ-ESEA and PEA-I in [12, 21]. $GA_1$, $GA_2$, $GA_3$, and $GA_4$ were initialized with RGH, $RGH_1$, OTTC, $RGH_2$. 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. In the second set of experiments, the population sizes of individual and independent $GA_1$, $GA_2$, $GA_3$, $GA_4$ population were increased to 500 so as to level out the number of fitness evaluation with HGA.

Each system was allocated 50 runs for each problem instance (so that the total number of runs in our experiments was 11250 runs), and all the programs were run on a machine with Pentium 4 Centrino 3.06 GHz CPU using 512MB RAM.

## V. Results and discussions

Tables 1-4 show the results of our experiments for all systems on the chosen problem instances. Table 1 shows the results of RJ-ESEA, PEA-I, and HGA on the 25 Euclidean problem instances in the first set of experiments. The results of PEA-I was taken from [21]. Table 2 depicts the results of independent individual GA (GA1-4) and HGA on the same 25 Euclidean problem instances. It is noted that $GA_1$ results was actually RJ-ESEA results as the two systems coincide. Tables 3 and 4 have similar meanings as tables 1 and 2 except that they show the results of the corresponding systems on Non-Euclidean instances. For tables 2 and 4, the columns of instance number have been omitted to fit the available space.

From the results in table 1, it can be seen that, on the Euclidean instances, HGA outperformed RJ-ESEA and PEA-I on almost all instances (rows in boldface and/or italic). In some cases, HGA outperformed both RJ-ESEA and PEA-I (rows in bold and italic face) statistically significantly (using test of the difference in mean between two binomial variables with $\alpha = 0.05$). In some other cases (rows in boldface), HGA found statistically significantly better solutions than either PEA-I or RJ-ESEA. By studying those rows, it can be seen that, when n is small, HGA statistically significantly outperformed PEA-I while its performance was still as good or better than RJ-ESEA (but not statistically significant). The situation is reversed when n is large (n>50), HGA was significantly better than RJ-ESEA, but only marginally better than PEA-I (except for a few cases where HGA significantly outperformed both other systems).

Table 2 shows that the use of multiple populations and migration in HGA was really useful (at least on those problem instances). HGA significantly outperformed each independent GA most of the time (rows in boldface) (again using test of the difference in mean between two binomial variables with $\alpha = 0.05$).

By contrast with the results on Euclidean instances, the results on non-Euclidean problem instances in table 3 do not give a clear margin of superiority of HGA over RJ-ESEA and PEA-I. On almost all problem instances, HGA was better, but not significantly, than the other two systems; in combination, these results suggest that there is an effect from the hybridization, but it is quite weak. This only marginally better performance could be explained by the results shown in table 4, where it appears that the use of multiple population and migration did not help HGA to perform better than the independent GAs ($GA_{1-4}$).

## VI. Conclusion

In this paper, we proposed a new hybrid genetic algorithm (HGA) for solving the BDMST problem. HGA employs multiple populations, which are initialized by using different well-known heuristics. The individuals from each population are then migrated to a final population called GA_final for merging and competing to breed. We have tested our new GA on a number of BDMST benchmark problem instances. The experimental results showed that on Euclidean problems, HGA outperformed the current best GAs, namely RJ-ESEA and PEA-I reported in [12, 21]; however it performed only very marginally better on non-Euclidean problems.

It is difficult to explain the differences between Euclidean and non-Euclidean problems.

In future work, we are planning to make a serious investigation into the course of the ineffectiveness of our HGA multi-population usage and migration strategies on solving Non-Euclidean problem instances. Other migration strategies and use of genetic operators for each individual GA population (such as the non-homogenous use of genetic operators) will be studied. We will also try our HGA algorithm on other tree representation and genetic operator sets such as those used in PEA-I. Moreover, using multi-parent crossover as in [20] for combining individuals in GA_final is also currently under our investigation.

## Acknowledgment

We would like to thank Brian Julstrom and Alok Singh for providing us with the BDMST problem instances (both Euclidean and Non-Euclidean), as well as sending us the materials related to their works on BDMST problems. Huynh Thi Thanh Binh is partly supported by a national research grant for fundamental sciences, grant number: 203106, for doing this work.

## References

[1] M.R. Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.

[2] K. Raymond, "A Tree-based Algorithm for Distributed Mutual Exclusion", *ACM Transactions on Computer Systems*, **7** (1), 1989, pp. 61-77.

[3] K. Bala, K. Petropoulos, and T. E. Stern, "Multicasting in a linear Lightwave Network", in *Proceedings of IEEE INFOCOM'93*, 1993, pp. 1350–1358.

[4] C.C. Palmer and A. Kershenbaum, "Representing Trees in Genetic Algorithms", in *Proceedings of The First IEEE Conference on Evolutionary Computation*, 1994, pp. 379-384.

[5] N.R.Achuthan, L.Caccetta, P.Caccetta, and A. Geelen, "Computational Methods for the Diameter Restricted Minimum Weight Spanning Tree Problem", *Australian Journal of Combinatorics*, **10**, 1994, pp.51-71.

[6] A. Bookstein and S. T. Klein, "Compression of Correlated Bit-Vectors", *Information Systems*, **16** (4), 1996, pp. 387-400.

[7] G. Kortsarz and D. Peleg, "Approximating Shallow-light Trees", in *Proceedings of the 8th Symposium on Discrete Algorithms*, 1997, pp. 103-110.

[8] A. Abdalla, N. Deo, and P. Gupta, "Random-tree Diameter and the Diameter Constrained MST", in *Proceedings of Congress on Numerantium*, 2000, pp. 161-182.

[9] J.Gottlieb, B.A.Julstrom, F.Rothlauf, and G.R.Raidl, "Prufer Numbers: A Poor Representation of Spanning Trees for Evolutionary Search", in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, 2001.

[10] A. Abdalla, "Computing a Diameter-constrained Minimum Spanning Tree", *PhD Dissertation,* The School of Electrical Engineering and Computer Science, University of Central Florida, 2001.

[11] G.R. Raidl and B.A. Julstrom, "Edge-sets: An Effective Evolutionary Coding of Spanning Trees", *IEEE Transactions on Evolutionary Computation*, **7**, 2003, pp.225-239.

[12] G.R. Raidl and B.A. Julstrom, "Greedy Heuristics and an Evolutionary Algorithm for the Bounded-Diameter Minimum Spanning Tree Problem", in *Proceeding of the ACM Symposium on Applied Computing*, 2003, pp. 747-752.

[13] B.A. Julstrom, G.R. Raidl, "A Permutation Coded Evolutionary for the Bounded Diameter Minimum Spanning Tree Problem, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)*, 2003, pp.2-7.

[14] B.A. Julstrom, "Encoding Bounded Diameter Minimum Spanning Trees with Permutations and Random Keys", in *Proceedings of Genetic and Evolutionary Computational Conference (GECCO'2004)*, 2004.

[15] L Gouveia, T.L. Magnanti and C. Requejo, "A 2-Path Approach for Odd Diameter Constrained Minimum Spanning and Steiner Trees", *Network*, **44** (4), 2004, pp. 254-265.

[16] M. Gruber and G.R. Raidl, "A New 0-1 ILP Approach for the Bounded Diameter Minimum Spanning Tree Problem, in *Proceedings of the 2nd International Network Optimization Conference,* 2005.

[17] M. Gruber and G.R. Raidl, "Variable Neighbourhood Search for the Bounded Diameter Minimum Spanning Tree Problem, in *Proceedings of the 18th Mini Euro Conference on Variable Neighborhood Search*, Spain, 2005.

[18] M. Gruber, J. Hemert, and G.R. Raidl, "Neighbourhood Searches for the Bounded Diameter Minimum Spanning Tree Problem Embedded in a VNS, EA and ACO", in Proceedings of *Genetic and Evolutionary Computational Conference (GECCO'2006)*, 2006.

[19] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, 2nd Edition, Springer-Verlag, 2006.

[20] Nguyen Duc Nghia and Huynh Thi Thanh Binh, "A New Recombination Operator for Solving Bouded Diameter Minimum Spanning Tree Problem", in Proceedings of RIVF'2007, LNCS, 2007.

[21] A. Singh and A.K. Gupta, "An Impoved Heuristic for the Bounded Diameter Minimum Spanning Tree Problem, *Journal of Soft Computing,* **11**, 2007, pp. 911-921.

[22] G. Kortsarz and D. Peleg, "Approximating the Weight of Shallow Steiner Trees", *Discrete Application Mathematics*, **93**, 1999, pp. 265-285.

[23] R. Prim, "Shortest Connection Networks and Some Generalization", *Bell System Technical Journal*, **36**, 1957, pp. 1389-1401.

TABLE 1. RESULTS OF RJ-ESEA, PEA-I, and HGA ON 25 EUCLIDEAN INSTANCES.

| Instance | | | RJ-ESEA | | | PEA-I | | | HGA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| n | k | Number | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD |
| 50 | 5 | 1 | 7.60 | 7.93 | 0.22 | 7.60 | 7.64 | 0.10 | 7.60 | 7.64 | 0.06 |
| 50 | 5 | 2 | **7.68** | **7.87** | **0.14** | **7.75** | **7.75** | **0.01** | **7.68** | **7.75** | **0.01** |
| 50 | 5 | 3 | 7.24 | 7.51 | 0.15 | 7.25 | 7.27 | 0.05 | 7.24 | 7.27 | 0.04 |
| 50 | 5 | 4 | 6.59 | 6.75 | 0.15 | 6.62 | 6.63 | 0.02 | 6.59 | 6.63 | 0.02 |
| 50 | 5 | 5 | **7.32** | **7.49** | **0.09** | **7.39** | **7.42** | **0.04** | **7.32** | **7.42** | **0.03** |
| 100 | 10 | 1 | **8.00** | **8.30** | **0.12** | **7.76** | **7.82** | **0.03** | **7.62** | **7.80** | **0.02** |
| 100 | 10 | 2 | ***8.10*** | ***8.41*** | ***0.16*** | ***7.85*** | ***7.89*** | ***0.04*** | ***7.71*** | ***7.80*** | ***0.04*** |
| 100 | 10 | 3 | ***8.22*** | ***8.61*** | ***0.19*** | ***7.90*** | ***7.97*** | ***0.04*** | ***7.73*** | ***7.82*** | ***0.03*** |
| 100 | 10 | 4 | ***8.27*** | ***8.57*** | ***0.17*** | ***7.98*** | ***8.04*** | ***0.03*** | ***7.72*** | ***7.94*** | ***0.03*** |
| 100 | 10 | 5 | ***8.48*** | ***8.72*** | ***0.15*** | ***8.16*** | ***8.21*** | ***0.03*** | ***7.92*** | ***8.02*** | ***0.02*** |
| 250 | 15 | 1 | **12.93** | **13.36** | **0.19** | **12.24** | **12.36** | **0.05** | **12.22** | **12.32** | **0.05** |
| 250 | 15 | 2 | ***12.86*** | ***13.25*** | ***0.20*** | ***12.04*** | ***12.13*** | ***0.04*** | ***12.00*** | ***12.08*** | ***0.05*** |
| 250 | 15 | 3 | **12.69** | **13.6** | **0.20** | **12.03** | **12.11** | **0.05** | **12.00** | **12.06** | **0.05** |
| 250 | 15 | 4 | **13.22** | **13.65** | **0.19** | **12.42** | **12.57** | **0.05** | **12.40** | **12.47** | **0.03** |
| 250 | 15 | 5 | **13.02** | **13.40** | **0.19** | **12.28** | **12.39** | **0.05** | **12.25** | **12.30** | **0.05** |
| 500 | 20 | 1 | **18.33** | **18.77** | **0.29** | **16.96** | **17.13** | **0.06** | **16.93** | **17.01** | **0.04** |
| 500 | 20 | 2 | **18.17** | **18.60** | **0.19** | **16.81** | **16.99** | **0.07** | **16.78** | **16.85** | **0.05** |
| 500 | 20 | 3 | **18.33** | **18.76** | **0.28** | **16.89** | **17.04** | **0.06** | **16.83** | **16.89** | **0.03** |
| 500 | 20 | 4 | **18.32** | **18.74** | **0.18** | **16.96** | **17.10** | **0.06** | **16.84** | **16.94** | **0.05** |
| 500 | 20 | 5 | ***17.80*** | ***18.40*** | ***0.28*** | ***16.58*** | ***16.72*** | ***0.06*** | ***16.47*** | ***16.55*** | ***0.06*** |
| 1000 | 25 | 1 | **26.13** | **26.72** | **0.31** | **23.97** | **24.19** | **0.10** | **23.92** | **24.20** | **0.12** |
| 1000 | 25 | 2 | ***26.14*** | ***26.58*** | ***0.27*** | ***23.70*** | ***23.98*** | ***0.13*** | ***23.65*** | ***24.09*** | ***0.13*** |
| 1000 | 25 | 3 | **25.47** | **26.21** | **0.29** | **23.61** | **23.76** | **0.08** | **23.55** | **23.70** | **0.08** |
| 1000 | 25 | 4 | **26.13** | **26.65** | **0.22** | **24.04** | **24.16** | **0.07** | **24.04** | **24.16** | **0.07** |
| 1000 | 25 | 5 | **25.91** | **26.29** | **0.27** | **23.75** | **23.90** | **0.07** | **23.75** | **23.89** | **0.07** |

TABLE 2. RESULTS OF $GA_1$, $GA_2$, $GA_3$, $GA_4$, AND HGA ON 25 EUCLIDEAN INSTANCES

| Instance | | $GA_1$ | | | $GA_2$ | | | $GA_3$ | | | $GA_4$ | | | HGA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | k | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD |
| 50 | 5 | 7.60 | 7.93 | 0.22 | 7.60 | 7.85 | 0.14 | 7.60 | 7.93 | 0.22 | 7.60 | 8.12 | 0.47 | 7.60 | 7.64 | 0.06 |
| 50 | 5 | 7.68 | 7.87 | 0.14 | 7.68 | 7.82 | 0.11 | 7.68 | 7.87 | 0.14 | 7.68 | 8.13 | 0.37 | 7.68 | 7.75 | 0.01 |
| 50 | 5 | 7.24 | 7.51 | 0.15 | 7.24 | 7.46 | 0.10 | 7.24 | 7.51 | 0.15 | 7.24 | 7.99 | 0.36 | 7.24 | 7.27 | 0.04 |
| 50 | 5 | 6.59 | 6.75 | 0.15 | 6.59 | 6.70 | 0.06 | 6.59 | 6.75 | 0.15 | 6.59 | 7.32 | 0.35 | 6.59 | 6.63 | 0.02 |
| 50 | 5 | 7.32 | 7.49 | 0.09 | 7.32 | 7.45 | 0.05 | 7.32 | 7.49 | 0.09 | 7.32 | 8.12 | 0.34 | 7.32 | 7.42 | 0.03 |
| 100 | 10 | **8.00** | **8.30** | **0.12** | **7.90** | **8.14** | **0.13** | **7.92** | **8.14** | **0.05** | **7.90** | **8.63** | **0.36** | **7.62** | **7.80** | **0.02** |
| 100 | 10 | 8.10 | 8.41 | 0.16 | 7.98 | 8.11 | 0.12 | 7.98 | 8.45 | 0.23 | 8.00 | 8.76 | 0.33 | 7.71 | 7.80 | 0.04 |
| 100 | 10 | **8.22** | **8.61** | **0.19** | **8.00** | **8.34** | **0.11** | **8.02** | **8.67** | **0.23** | **8.03** | **8.82** | **0.36** | **7.73** | **7.82** | **0.03** |
| 100 | 10 | **8.27** | **8.57** | **0.17** | **8.10** | **8.66** | **0.21** | **8.10** | **8.66** | **0.21** | **8.12** | **9.12** | **0.39** | **7.72** | **7.94** | **0.03** |
| 100 | 10 | 8.48 | 8.72 | 0.15 | 8.11 | 8.48 | 0.19 | 8.13 | 8.78 | 0.33 | 8.16 | 9.21 | 0.36 | 7.92 | 8.02 | 0.02 |
| 250 | 15 | **12.93** | **13.36** | **0.19** | **12.67** | **13.14** | **0.24** | **12.79** | **13.56** | **0.33** | **12.73** | **13.81** | **0.45** | **12.22** | **12.32** | **0.05** |
| 250 | 15 | **12.86** | **13.25** | **0.20** | **12.50** | **13.00** | **0.17** | **12.55** | **13.01** | **0.23** | **12.29** | **13.79** | **0.44** | **12.00** | **12.08** | **0.05** |
| 250 | 15 | **12.69** | **13.60** | **0.20** | **12.43** | **12.87** | **0.14** | **12.48** | **13.01** | **0.25** | **12.47** | **13.43** | **0.47** | **12.00** | **12.06** | **0.05** |
| 250 | 15 | **13.22** | **13.65** | **0.19** | **13.01** | **13.59** | **0.17** | **13.00** | **13.78** | **0.21** | **13.05** | **13.99** | **0.38** | **12.40** | **12.47** | **0.03** |
| 250 | 15 | **13.02** | **13.40** | **0.19** | **13.00** | **13.56** | **0.17** | **12.98** | **13.89** | **0.25** | **13.02** | **13.99** | **0.33** | **12.25** | **12.30** | **0.05** |
| 500 | 20 | **18.33** | **18.77** | **0.29** | **17.70** | **18.95** | **0.17** | **17.74** | **18.78** | **0.34** | **17.63** | **19.01** | **0.37** | **16.93** | **17.01** | **0.04** |
| 500 | 20 | **18.17** | **18.60** | **0.19** | **17.75** | **18.74** | **0.34** | **17.70** | **18.72** | **0.35** | **17.60** | **18.89** | **0.41** | **16.78** | **16.85** | **0.05** |
| 500 | 20 | **18.33** | **18.76** | **0.28** | **17.88** | **18.76** | **0.35** | **17.92** | **18.71** | **0.35** | **17.88** | **18.82** | **0.36** | **16.83** | **16.89** | **0.03** |
| 500 | 20 | **18.32** | **18.74** | **0.18** | **17.83** | **18.56** | **0.27** | **17.85** | **18.68** | **0.32** | **17.88** | **18.89** | **0.32** | **16.84** | **16.94** | **0.05** |
| 500 | 20 | **17.80** | **18.40** | **0.28** | **17.54** | **18.32** | **0.26** | **17.67** | **18.68** | **0.37** | **17.64** | **18.81** | **0.42** | **16.47** | **16.55** | **0.06** |
| 1000 | 25 | **26.13** | **26.72** | **0.31** | **24.78** | **25.02** | **0.18** | **24.99** | **25.97** | **0.31** | **25.02** | **26.45** | **0.32** | **23.92** | **24.20** | **0.12** |
| 1000 | 25 | **26.14** | **26.58** | **0.27** | **24.98** | **25.34** | **0.18** | **25.11** | **25.79** | **0.22** | **25.23** | **26.34** | **0.41** | **23.65** | **24.09** | **0.13** |
| 1000 | 25 | **25.47** | **26.21** | **0.29** | **25.11** | **25.49** | **0.19** | **25.12** | **25.98** | **0.21** | **25.10** | **26.23** | **0.32** | **23.55** | **23.70** | **0.08** |
| 1000 | 25 | **26.13** | **26.65** | **0.22** | **25.57** | **26.12** | **0.18** | **25.63** | **26.45** | **0.31** | **25.89** | **26.99** | **0.33** | **24.04** | **24.16** | **0.07** |
| 1000 | 25 | **25.91** | **26.29** | **0.27** | **25.15** | **25.78** | **0.16** | **25.11** | **26.00** | **0.31** | **25.34** | **28.12** | **0.40** | **23.75** | **23.89** | **0.07** |

TABLE 3. RESULTS OF RJ-ESEA, PEA_I, AND HGA ON 20 NON-EUCLIDEAN BDMST PROBLEM INSTANCES.

| Instance | | | RJ-ESEA | | | PEA-I | | | HGA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| n | K | Number | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD |
| 100 | 10 | 1 | 2.33 | 2.49 | 0.07 | 2.32 | 2.38 | 0.03 | 2.32 | 2.38 | 0.03 |
| 100 | 10 | 2 | 2.20 | 2.45 | 0.06 | 2.20 | 2.22 | 0.02 | 2.18 | 2.22 | 0.02 |
| 100 | 10 | 3 | 2.42 | 2.78 | 0.07 | 2.40 | 2.43 | 0.04 | 2.40 | 2.41 | 0.04 |
| 100 | 10 | 4 | 2.19 | 2.59 | 0.06 | 2.18 | 2.23 | 0.02 | 2.16 | 2.21 | 0.04 |
| 100 | 10 | 5 | 2.40 | 2.79 | 0.06 | 2.35 | 2.42 | 0.04 | 2.34 | 2.40 | 0.03 |
| 250 | 15 | 1 | 3.78 | 3.99 | 0.11 | 3.73 | 3.79 | 0.03 | 3.72 | 3.76 | 0.03 |
| 250 | 15 | 2 | 3.80 | 3.99 | 0.08 | 3.79 | 3.83 | 0.03 | 3.77 | 3.82 | 0.03 |
| 250 | 15 | 3 | 3.69 | 3.99 | 0.08 | 3.69 | 3.76 | 0.03 | 3.69 | 3.77 | 0.03 |
| 250 | 15 | 4 | 3.78 | 4.02 | 0.08 | 3.76 | 3.82 | 0.03 | 3.75 | 3.80 | 0.03 |
| 250 | 15 | 5 | 3.98 | 4.44 | 0.06 | 3.88 | 3.95 | 0.04 | 3.87 | 3.96 | 0.04 |
| 500 | 20 | 1 | 6.26 | 6.46 | 0.07 | 6.24 | 6.29 | 0.03 | 6.23 | 6.27 | 0.02 |
| 500 | 20 | 2 | 6.30 | 6.60 | 0.06 | 6.30 | 6.36 | 0.03 | 6.28 | 6.35 | 0.03 |
| 500 | 20 | 3 | 6.17 | 6.37 | 0.06 | 6.16 | 6.22 | 0.03 | 6.16 | 6.24 | 0.03 |
| 500 | 20 | 4 | 6.25 | 6.55 | 0.06 | 6.25 | 6.32 | 0.03 | 6.23 | 6.32 | 0.04 |
| 500 | 20 | 5 | 6.25 | 6.54 | 0.06 | 6.25 | 6.29 | 0.04 | 6.22 | 6.27 | 0.04 |
| 1000 | 25 | 1 | 11.32 | 11.63 | 0.06 | 11.26 | 11.31 | 0.03 | 11.25 | 11.30 | 0.03 |
| 1000 | 25 | 2 | 11.28 | 11.52 | 0.05 | 11.30 | 11.34 | 0.03 | 11.27 | 11.32 | 0.03 |
| 1000 | 25 | 3 | 11.29 | 11.49 | 0.04 | 11.30 | 11.35 | 0.03 | 11.28 | 11.34 | 0.04 |
| 1000 | 25 | 4 | 11.23 | 11.45 | 0.05 | 11.22 | 11.26 | 0.03 | 11.22 | 11.25 | 0.03 |
| 1000 | 25 | 5 | 11.39 | 11.69 | 0.04 | 11.39 | 11.42 | 0.03 | 11.37 | 11.41 | 0.03 |

TABLE 4. RÉSULTS OF GA1, GA2, GA3, GA4, AND HGA ON 20 NON-EUCLIDEAN BDMST PROBLEM INSTANCES.

| Instance | | GA1 | | | GA2 | | | GA3 | | | GA4 | | | HGA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| n | k | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD | Best | Avg | SD |
| 100 | 10 | 2.33 | 2.49 | 0.07 | 2.33 | 2.45 | 0.07 | 2.33 | 2.59 | 0.09 | 2.33 | 2.79 | 0.23 | 2.32 | 2.38 | 0.03 |
| 100 | 10 | 2.20 | 2.45 | 0.06 | 2.20 | 2.42 | 0.06 | 2.22 | 2.60 | 0.09 | 2.23 | 2.78 | 0.24 | 2.18 | 2.22 | 0.02 |
| 100 | 10 | 2.42 | 2.78 | 0.07 | 2.41 | 2.66 | 0.06 | 2.42 | 2.90 | 0.11 | 2.42 | 3.11 | 0.11 | 2.40 | 2.41 | 0.04 |
| 100 | 10 | 2.19 | 2.59 | 0.06 | 2.18 | 2.55 | 0.06 | 2.19 | 2.79 | 0.07 | 2.19 | 2.88 | 0.09 | 2.16 | 2.21 | 0.04 |
| 100 | 10 | 2.40 | 2.79 | 0.06 | 2.39 | 2.69 | 0.06 | 2.40 | 3.01 | 0.08 | 2.40 | 3.23 | 0.12 | 2.34 | 2.40 | 0.03 |
| 250 | 15 | 3.78 | 3.99 | 0.11 | 3.77 | 3.94 | 0.06 | 3.77 | 4.02 | 0.08 | 3.79 | 4.15 | 0.14 | 3.72 | 3.76 | 0.03 |
| 250 | 15 | 3.80 | 3.99 | 0.08 | 3.79 | 3.87 | 0.06 | 3.79 | 4.01 | 0.08 | 3.80 | 4.23 | 0.14 | 3.77 | 3.82 | 0.03 |
| 250 | 15 | 3.69 | 3.99 | 0.08 | 3.69 | 3.98 | 0.06 | 3.69 | 4.15 | 0.12 | 3.70 | 4.35 | 0.16 | 3.69 | 3.77 | 0.03 |
| 250 | 15 | 3.78 | 4.02 | 0.08 | 3.78 | 4.03 | 0.07 | 3.79 | 4.27 | 0.17 | 3.79 | 4.45 | 0.25 | 3.75 | 3.80 | 0.03 |
| 250 | 15 | 3.98 | 4.44 | 0.06 | 3.95 | 4.29 | 0.05 | 3.96 | 4.69 | 0.06 | 3.97 | 4.48 | 0.07 | 3.87 | 3.96 | 0.04 |
| 500 | 20 | 6.26 | 6.46 | 0.07 | 6.25 | 6.62 | 0.05 | 6.25 | 6.78 | 0.07 | 6.25 | 6.98 | 0.12 | 6.23 | 6.27 | 0.02 |
| 500 | 20 | 6.30 | 6.60 | 0.06 | 6.30 | 6.54 | 0.03 | 6.31 | 6.75 | 0.07 | 6.31 | 6.96 | 0.11 | 6.28 | 6.35 | 0.03 |
| 500 | 20 | 6.17 | 6.37 | 0.06 | 6.16 | 6.34 | 0.05 | 6.16 | 6.72 | 0.07 | 6.17 | 6.78 | 0.12 | 6.16 | 6.24 | 0.03 |
| 500 | 20 | 6.25 | 6.55 | 0.06 | 6.25 | 6.50 | 0.05 | 6.25 | 6.80 | 0.07 | 6.26 | 6.98 | 0.10 | 6.23 | 6.32 | 0.04 |
| 500 | 20 | 6.25 | 6.54 | 0.06 | 6.24 | 6.52 | 0.05 | 6.24 | 6.83 | 0.07 | 6.25 | 6.93 | 0.10 | 6.22 | 6.27 | 0.04 |
| 1000 | 25 | 11.32 | 11.63 | 0.06 | 11.31 | 11.66 | 0.06 | 11.31 | 11.99 | 0.07 | 11.32 | 12.12 | 0.12 | 11.25 | 11.30 | 0.03 |
| 1000 | 25 | 11.28 | 11.52 | 0.05 | 11.28 | 11.53 | 0.06 | 11.29 | 11.98 | 0.07 | 11.30 | 12.01 | 0.09 | 11.27 | 11.32 | 0.03 |
| 1000 | 25 | 11.29 | 11.49 | 0.04 | 11.30 | 11.56 | 0.04 | 11.30 | 11.98 | 0.07 | 11.29 | 12.00 | 0.09 | 11.28 | 11.34 | 0.04 |
| 1000 | 25 | 11.23 | 11.45 | 0.05 | 11.22 | 11.55 | 0.06 | 11.22 | 11.78 | 0.07 | 11.22 | 11.98 | 0.09 | 11.22 | 11.25 | 0.03 |
| 1000 | 25 | 11.39 | 11.69 | 0.04 | 11.38 | 11.68 | 0.04 | 11.38 | 11.98 | 0.07 | 11.38 | 12.02 | 0.08 | 11.37 | 11.41 | 0.03 |

Number: the number of instance
K: bounded diameter
Best: Min weight of the best trees obtained by the algorithm over the runs.
Avg: Mean weight of the best tree obtained by the algorithm over the runs.
SD: Standard deviation of Best