Effective Heuristics for the Bi-objective Euclidean Bounded Diameter Minimum Spanning Tree Problem

V. Prem Prakash[1(✉)], C. Patvardhan[1], and Anand Srivastav[2]

[1] Dayalbagh Educational Institute (Deemed University), Agra 282005, India
vpremprakash@acm.org
[2] Christian-Albrechts-Universitat zu Kiel, Kiel, Germany

Abstract. The Euclidean Bounded Diameter Minimum Spanning Tree (BDMST) Problem aims to find the spanning tree with the lowest cost, or weight, under the constraint that the diameter does not exceed a given integer D, and where the weight of an edge is the Euclidean distance between its two end points (vertices). Several well-known heuristic approaches have been applied to this problem. The bi-objective version of this problem aims to minimize two conflicting objectives, weight (or cost), and diameter. Several heuristics for the BDMST problem have been recast for the bi-objective BDMST problem (or BOMST problem) and their performance studied on the entire range of possible diameter values. While some of the extant heuristics are seen to dominate other heuristics over certain portions of the Pareto front of solutions, no single heuristic performs well over the entire range. This paper presents a hybrid tree construction heuristic that combines a greedy approach with a heuristic strategy for constructing effective tree "backbones". The performance of the proposed heuristic is shown to be consistently superior to the other extant heuristics on a standard benchmark suite of dense Euclidean graphs widely used in the literature.

Keywords: Multi-objective · BOMST · Spanning tree · Bounded diameter

1 Introduction

Given a connected, weighted, undirected graph G and an integer bound D, a bounded-diameter spanning tree (BDST) is a spanning tree on G whose diameter, that is, the maximum number of edges along any path in the tree, does not exceed D. The Bounded Diameter Minimum Spanning Tree (BDMST) Problem aims to find a BDST on G of minimum weight (the weight of a BDST is the sum of its edge weights). The BDMST problem is known to be NP-hard [1] for $4 \leq D < n - 1$. The Euclidean version of the BDMST problem deals with graph instances whose edge weights are the Euclidean distances between the connected vertices.

Applications from several domains map to this problem: routing problems in VLSI often require minimum spanning trees that bound the sink-source delays (diameter bound) and total wire length (minimum cost/weight); large bitmap data structures are often clustered and compressed as MSTs – fast retrieval such structures of such structures requires the MST to have a low diameter [2]; the work by Raymond [3] presents an application of the BDMST problem for minimizing communication costs while performing distributed mutual exclusion in large scale distributed systems and ad-hoc networks.

The Bi-objective Bounded Diameter Minimum Spanning Tree (BOMST) Problem [4] aims to find BDSTs such that both the diameter and weight of the tree are minimized over the entire range of diameter values. Thus the single-objective BDMST Problem is a special case of the more general BOMST Problem as defined here. The Euclidean BOMST Problem (e-BOMST) restricts the domain to that of Euclidean instances.

Achuthan and Caccetta give two exact algorithms for the BDMST Problem in [5, 6]. Multiple variants of multi-commodity flow (MCF) formulations for the BDMST problem are given by Gouveia and Magnanti [7], which obtain very tight LP bounds. However, these algorithms are only able to solve very small problem instances. This has motivated the search for algorithms that are able to approximate low cost BDSTs well for sufficiently large problem sizes within reasonable time. Several such heuristics abound in the literature of the BDMST problem.

Abdalla and Deo [8] give a construction heuristic based on Prim's algorithm [14] called the *One-time tree construction* (OTTC) heuristic that runs in $O(n^4)$ time and produces low cost BDSTs when the diameter constraint is small. They also give two iterative refinement (IR) algorithms that iteratively decrease the lengths of long paths in an unconstrained MST until the diameter constraint is satisfied. A more effective Prim's-based approach is presented in the *Center-based tree construction* (CBTC) heuristic given by Julstrom [9], which constructs the BDST as a height-restricted tree rooted either at a central vertex, if the diameter limit is even, or a central edge, if it is odd. This heuristic, which takes $O(n^3)$ time, outperforms OTTC both in terms of solution quality and running time. The *Randomized tree construction heuristic* (RTC) [9] builds the BDST by selecting vertices in a random order and appending each vertex to the tree at the lowest cost possible. This heuristic also requires $O(n^3)$ computation time. The *Center-based Least Sum-of-Costs* (CBLSoC) heuristic given by Patvardhan and Prakash [10] builds a low cost BDST in $O(n^3)$ time by repeatedly appending the non-tree vertex with the lowest mean cost to all the remaining non-tree vertices in the graph. Parallel versions of the CBTC, RTC and CBLSoC heuristics are given in [16] and their performance studied on several benchmark problems. A recursive, clustering-based heuristic called *Center-based Recursive Clustering* heuristic (CBRC) is given for the problem by Nghia and Binh [11].

Kumar and Saha [4, 12] adapt several extant BDMST heuristics for the bi-objective MST formulation of the problem and compare their performance on standard problem instances. The results obtained are further improved using a bi-objective meta-heuristic algorithm seeded with different heuristic solutions.

This paper presents a comprehensive comparison of some well known extant heuristics with a hybrid heuristic that is adapted to the Euclidean version of the BOMST problem. The performance of all the heuristics is obtained on much larger problem instances than in earlier work on the BOMST problem, and the proposed heuristic is shown to give superior performance on a benchmark suite comprised of several dense Euclidean graph instances used widely in the literature.

The rest of the paper is organized as follows. Section 2 describes two extant heuristics that were shown to obtain superior Pareto fronts on Euclidean graphs in an earlier work [4], and another well known heuristic for the BDMST problem which is recast for the bi-objective version of the problem. Section 3 presents a hybrid heuristic that computes low cost BDSTs across the diameter range in O(n^3) time. The performance of the heuristics on the benchmark suite is presented and discussed in Sect. 4, and concluding remarks made in Sect. 5.

2 Three Extant Heuristics

Several heuristics have been developed in the literature for the BDMST problem, some of which were recast for the BOMST problem in [4, 12]. Of these, the Center-based Tree Construction (CBTC) and Randomized Tree Construction (RTC) were found to generally obtain superior results on Euclidean instances over different ranges of the Pareto front [12]. Therefore both of these are taken as baselines for comparing the performance of the heuristics presented in this work with. The CBLSoC heuristic has also been shown to perform well on Euclidean instances [15], and has been adapted here for the eBOMST Problem. Following is a brief description of each of these heuristics.

2.1 Center-Based Tree Construction (CBTC)

In a tree with diameter D, no vertex is more than D/2 hops or edges from the root vertex of the tree [13]. The Center-Based Tree Construction (CBTC) heuristic [9] uses this idea to build a BDST starting with an arbitrary graph vertex as the center of the tree, and repeatedly appending to the BDST, the graph vertex with the lowest cost edge to the partial tree. The heuristic dynamically maintains the depth information of each tree node and ensures that the depth of any leaf node of the tree is at most D/2. The center of the tree comprises of a single vertex if D is even, and one edge if D is odd. The algorithm repeatedly appends to the growing BDST, the edge with the lowest-cost/weight that adds a new vertex to the tree, while not violating the diameter bound. In order to obtain a low cost BDST, this process is repeated n times, taking a different graph vertex as the root node in each iteration.

Lemma 1. The running time of Center-based Tree Construction heuristic is O(n^3).

Proof. The heuristic builds a BDST from its center, keeping track of the depth of each incoming vertex and ensuring that node depth is strictly less than D/2. By using an O(n) space data structure to dynamically keep track of the tree node of depth $\leq$ D/2 closest to each graph vertex, the computational cost of appending each incoming node to the growing BDST becomes a linear time operation. Appending $n - 1$ vertices (or $n - 2$ vertices to the BDST if D is odd) results in O(n^2) time for building one BDST. As the heuristic repeats this process for each vertex, the total computation time required is O(n^3).

2.2 Randomized Tree Construction (RTC)

The Randomized Tree Construction (RTC) heuristic sets a randomly chosen vertex (or edge, depending on whether D is even or odd, respectively) as the root of the BDST. All subsequent vertices are chosen at random and appended to the tree via the lowest cost edge that does not violate the diameter bound D. As with the CBTC, this process is repeated n times, and the lowest cost BDST is returned.

Lemma 2. The running time of Randomized Tree Construction heuristic is $O(n^3)$.

Proof. If a partially constructed BDST has k nodes (represented by the set T), then there are $n - k$ graph vertices (represented as the set U) that are not part of the BDST. Choosing a vertex u at random from U and appending it greedily to the vertex $v \in T$ such that cost (u, v) is minimal would require O(n) time. Appending $n - 1$ vertices in this manner to complete the BDST would therefore require $O(n^2)$ time. Repeating this process n times and returning the best BDST would thus need a total of $O(n^3)$ time.

2.3 Center-Based Least Sum-of-Costs (CBLSoC) Heuristic

The CBLSoC heuristic tries to construct the BDST in a relatively "less greedy" manner by repeatedly appending to the partial tree, the graph vertex with the lowest average cost to all other graph vertices, via the edge with smallest cost that does not violate the diameter bound. This process is repeated starting from each graph vertex, and the lowest cost BDST obtained is returned by the algorithm.

Lemma 3. The running time of the CBLSoC heuristic is $O(n^3)$.

Proof. For each vertex u, the sum of costs (and hence the mean cost) to all other $n - 1$ vertices in the graph G can be computed in O(n) time. Hence the time to compute the sum-of-costs for all vertices in the graph would take $O(n^2)$ time. The BDST is initially empty, i.e., $T = \varphi$. Starting with a center based approach, identifying the graph vertex u with lowest mean cost to all other vertices in G would require O(n) time. Appending u to vertex $v \in T$ such that cost (u, v) is minimal would also take O(n) time. As explained in the proofs of Lemmas 1 and 2, appending $n - 1$ vertices in this manner to complete the BDST would require $O(n^2)$ time; repeating this process n times and returning the best BDST would therefore result in a total of $O(n^3)$ time.

3 Hybrid Tree Construction (Hyb-TC) Heuristic

The CBTC heuristic loses out to other heuristics (notably the RTC heuristic) for small values of D, but its performance improves quickly as the diameter bound is relaxed (cf. Table 1). This is because the greediness inherent in the CBTC heuristic constrains it to always choose the vertex that can be appended to the tree at the lowest cost possible. In Euclidean BDSTs, this often results in backbones comprised of a small number of low-depth vertices that are very close to each other, forcing the remaining vertices to be appended to these vertices via higher cost edges and thereby returning BDSTs with high total cost, or weight. As the diameter bound is gradually increased, the BDST returned by CBTC tends to shape up like an unconstrained MST, resulting in lower cost BDSTs. Another fast heuristic strategy that proves very effective on small D in Euclidean instances is the greedy Quadrant-centers based Heuristic (gQCH) [15], which empirically breaks up the Euclidean space of graph vertices into different numbers of equal sized sub-spaces, or quadrants, and tries to build a backbone of tree vertices that can in turn form the roots for low cost sub-trees obtained by greedily appending the remaining graph vertices to the BDST. In each quadrant, the vertex with the lowest average cost to all other vertices (in that quadrant) is set as a backbone node. When the number of quadrants is 1, this heuristic "collapses" into a single run of the CBTC heuristic with the root node(s) set as the vertex (if the diameter limit is even) or pair of vertices (if the diameter limit is odd) with the lowest mean cost to all other graph vertices.

The proposed Hybrid Tree Construction (Hyb-TC) heuristic combines these two strategies in order to obtain better results over the entire range of diameter bounds. The heuristic starts in the same manner as the CBTC does, and builds n BDSTs starting once each from each vertex/vertex pair. Thereafter, it tries to quickly build effective BDST backbones using a subset of the vertex set to which the remaining vertices may be appended greedily at a lower cost on average, thereby leading to lower tree costs. With this objective, the heuristic chooses the graph vertex/vertices with the lowest mean cost(s) to all other graph nodes as the central, or root vertex/vertices (depending respectively on whether D is even or odd), and segregates the remaining graph vertices into the "quadrants" of a uniform $K \times K$ matrix in the two dimensional Euclidean plane, for $2 \leq K \leq \sqrt{n}$. Within each quadrant, the vertex with the lowest mean cost to all other vertices within the same quadrant is set as a tree backbone node of depth 1. Once the backbone has been constructed in this manner, the remaining vertices are appended to the tree greedily, as in CBTC. Another $\sqrt{n} - 1$ BDSTs are constructed in this manner, and the algorithm returns the lowest cost tree from amongst the $n + \sqrt{n} - 1$ BDSTs thus constructed.

Lemma 4. The running time of the Hyb-TC heuristic is $O(n^3)$.

Proof. In order to build the backbone, the heuristic computes the mean cost of each vertex to every other vertex within its designated quadrant. Computing the mean cost of each vertex v over K^2 quadrants comprising a total of n vertices would require at most $O(n^2)$ time. Over $\sqrt{n-1}$ iterations, this would require a total of $O(n^2\sqrt{n})$ time. In each iteration, the remaining vertices are then appended greedily to the partial BDST – if the backbone consists of m vertices, then n – m graph vertices need to be appended to the tree – this operation would take another $O(n^2)$ time. Thus the total computation time to compute the $\sqrt{n-1}$ BDSTs would be $O(n^2\sqrt{n})$ time. The remaining n BDSTs are generated using a greedy approach starting from each vertex. As shown in the proof for Lemma 1, constructing each BDST would require an additional $O(n^2)$ time, and repeating this process for n vertices would take $O(n^3)$ time. Thus the total computation time of the heuristic is the sum of these two components: $O(n^3)$ time for finding the lowest cost BDST starting once from each vertex, and $O(n^2\sqrt{n})$ for computing $\sqrt{n} - 1$ BDSTs using the backbone construction heuristic, thus resulting in a total computation time of $O(n^3)$.

4 Experiments

The benchmark graphs used in this work were taken from the Euclidean Steiner Problem data sets given in Beasley's OR-Library [17]. These data sets contain fifteen instances each of completely connected graphs with 50, 100, 250, 500 and 1000 vertices, working out to a total of seventy five graphs. The x- and y- co-ordinates of random points in the unit square form the vertices of the graph, and the Euclidean distance between these vertices their edge weights. These instances have been used in the literature for benchmarking heuristics and algorithms for the BOMST Problem.

The heuristics presented in this paper were tested on the first five instances of 50, 100, 250 and 500 vertex dense graphs of the Euclidean Steiner data sets, a total of twenty problem instances, and the BDST costs returned by the heuristics were obtained for diameter D, $2 \leq D \leq D_{max}$, where the upper limit for the D values, D_{max} was chosen as the diameter of an unconstrained MST on each graph instance. Pareto fronts were obtained for the heuristics in this paper on each test instance, and plots of the Pareto fronts obtained for the first instance of each size (50, 100, 250 and 500) of benchmark graph are shown in Figs. 1, 2, 3 and 4 respectively. All the heuristics were implemented in C on a Dell Precision T5500 workstation with 12 Xeon (2.4-GHz) processor cores and 11 GB of RAM running RHEL 6. The BDST costs obtained by

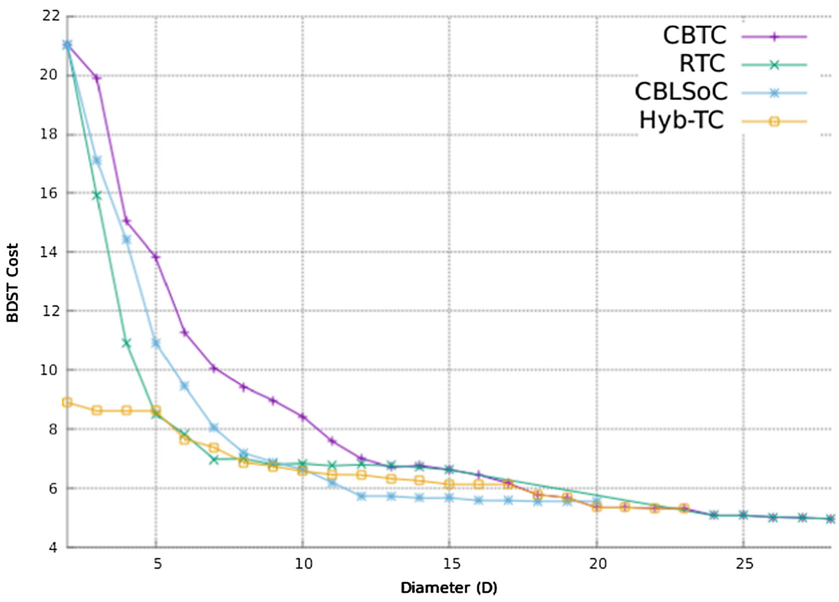

Fig. 1. Pareto fronts obtained over the entire diameter range for the first 50 vertex dense Euclidean graph for the CBTC, RTC, CBLSoC and Hyb-TC heuristics, up to D = 28.

each heuristic for five selected diameter limits on the first instance of each of the 50, 100, 250 and 500 vertex benchmark graphs are given in Table 1.

Table 1. BDST costs obtained for selected values of diameter bound on dense graphs of 50, 100, 250 and 500 vertices

n	D	BDST cost/weight			
		CBTC	RTC	CBLSoC	Hyb-TC
50	5	13.04	8.53	10.94	**8.63**
	10	8.44	6.84	6.66	**6.59**
	15	6.64	6.63	**5.68**	6.14
	20	**5.35**	–	5.56	**5.35**
	25	**5.08**	–	–	**5.08**
100	5	27.12	15.13	23.16	**12.97**
	15	11.41	8.80	8.39	**8.33**
	25	7.07	–	**7.03**	7.07
	35	**6.79**	–	–	**6.79**
	45	**6.61**	–	–	**6.61**
250	5	75.68	32.34	60.58	**25.63**
	20	26.74	15.11	17.70	**13.55**
	35	14.35	–	12.17	**11.71**
	50	**11.29**	–	–	**11.29**
	65	**10.65**	–	–	**10.65**
500	5	153.44	65.42	128.40	**42.22**
	35	39.89	21.50	24.00	**17.29**
	70	16.46	–	15.98	**15.92**
	105	**15.10**	–	–	**15.10**
	140	**14.86**	–	–	**14.86**

The Pareto fronts for the various heuristics show that on small-to-medium range of diameter bound, the CBLSoC heuristic significantly outperforms the CBTC, but is in turn dominated by the RTC heuristic. However, as D is further increased, the random choices made by the RTC lead it astray, often resulting in stagnation of improvements in the cost of BDST returned (such results are indicated in Table 1 by a blank entry). The Hyb-TC heuristic invariably obtains superior results, dominating all the other heuristics in this range of the diameter bound, on all instances. This is clearly seen from the better Pareto fronts obtained by the Hyb-TC heuristic on up to 500 vertex dense graphs (Figs. 1, 2, 3 and 4 respectively). On medium range diameter bounds, CBLSoC returns superior BDSTs vis-à-vis the other heuristics on 50 and 100 vertex graphs (Figs. 1 and 2 respectively), and remains competitive on larger instances. However, CBLSoC fails to obtain further improvements in the best trees returned when the diameter bound is large. With the exception of a few cases on 50 and 100 vertex graphs where the CBTC heuristic obtains better trees, the Hyb-TC heuristic outperforms all the other heuristics over the medium range of diameter bound. As the diameter bound becomes large, the low cost BDST returned by the Hyb-TC heuristic tends towards an

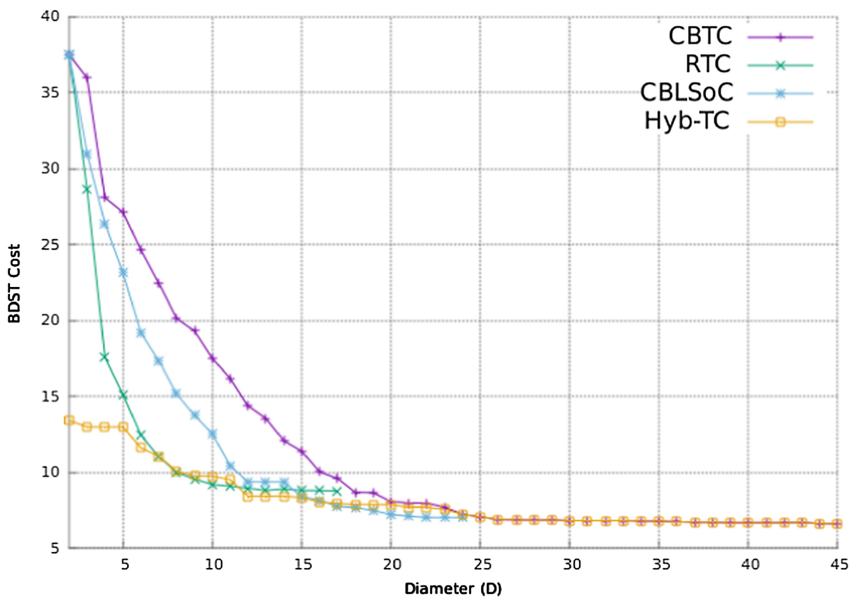

Fig. 2. Pareto fronts obtained over the entire diameter range for the first 100 vertex Euclidean benchmark instance for the CBTC, RTC, CBLSoC and Hyb-TC heuristics, up to D = 45.

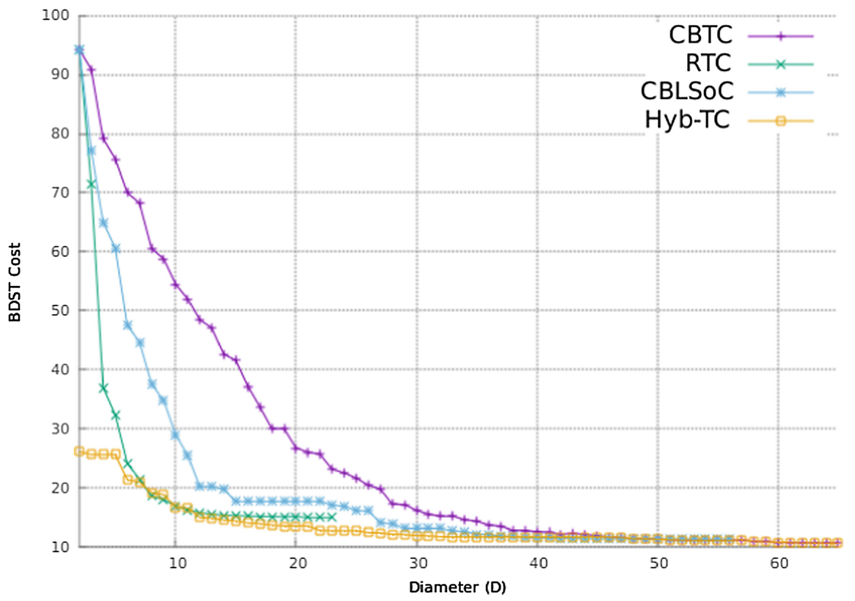

Fig. 3. Pareto fronts obtained over the entire diameter range for the first 250 vertex Euclidean benchmark graph for the CBTC, RTC, CBLSoC and Hyb-TC heuristics, up to D = 65.

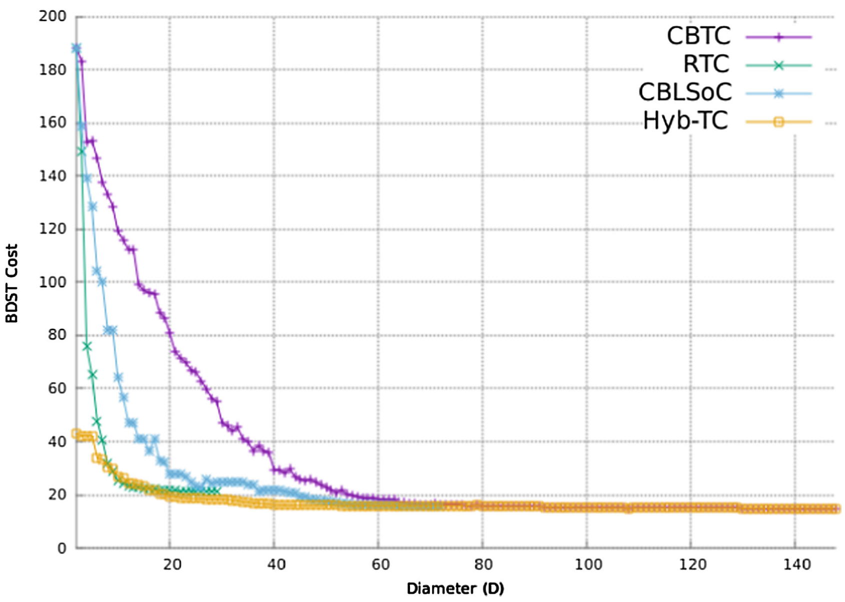

Fig. 4. Pareto fronts obtained over the entire diameter range for the first 500 vertex Euclidean benchmark graph for the CBTC, RTC, CBLSoC and Hyb-TC heuristics, up to D = 145.

unconstrained MST, thereby leading to good, low cost BDSTs as in the CBTC heuristic. Thus it is seen that in general, the Hyb-TC dominates all of the other heuristics across the entire Pareto front of solutions.

5 Conclusions

A comprehensive comparison of several well known extant heuristics for the eBOMST problem with a proposed hybrid heuristic is made in this work. While some of the existing heuristics obtain good results for low diameter bounds and others perform better as the diameter is relaxed, none of the extant heuristics obtains low cost BDSTs across the entire range of diameter values. The proposed Hyb-TC heuristic combines the strengths of some of the extant heuristics and obtains consistently superior BDSTs across the entire range of possible diameter values. The performance of the heuristics is studied on a wide range of dense graphs of up to 500 vertices, and the hybrid heuristic is shown to outperform all the other extant heuristics.

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP Completeness. Freeman, New York, W.H (1979)
2. Bookstein, A., Klein, S.T.: Compression of correlated bit-vectors. Inf. Syst. **16**(4), 110–118 (1996)

3. Raymond, K.: A tree-based algorithm for distributed mutual exclusion. ACM Trans. Comput. Syst. **7**(1), 61–77 (1989)
4. Saha, S., Aslam, M., Kumar, R.: Assessing the performance of bi-objective MST for Euclidean and non-Euclidean instances. In: Ranka, S., Banerjee, A., Biswas, K.K., Dua, S., Mishra, P., Moona, R., Poon, S.-H., Wang, C.-L. (eds.) IC3 2010. CCIS, vol. 94, pp. 229–240. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14834-7_22
5. Achuthan, N.R., Caccetta, L.: Minimum weight spanning trees with bounded diameter. Australas. J. Comb. Univ. Queensland Press **5**, 261–276 (1992)
6. Achuthan, N.R., Caccetta, L., Caccetta, P., Geelen, J.F.: Computational methods for the diameter restricted minimum weight spanning tree problem. Australas. J. Comb. Univ. Queensland Press **10**, 51–71 (1994)
7. Gouveia, L., Magnanti, T.L.: Network flow models for designing diameter constrained minimum spanning and Steiner trees. Netw. **41**(3), 159–173 (2003)
8. Deo, N., Abdalla, A.: Computing a diameter-constrained minimum spanning tree in parallel. In: Bongiovanni, G., Petreschi, R., Gambosi, G. (eds.) CIAC 2000. LNCS, vol. 1767, pp. 17–31. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-46521-9_2
9. Julstrom, B.A.: Greedy heuristics for the bounded diameter minimum spanning tree problem. J. Exp. Algorithmics **14**(1), 1–14 (2009)
10. Patvardhan, C., Prakash, V.P.: Novel deterministic heuristics for building minimum spanning trees with constrained diameter. In: Chaudhury, S., Mitra, S., Murthy, C.A., Sastry, P.S., Pal, Sankar K. (eds.) PReMI 2009. LNCS, vol. 5909, pp. 68–73. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-11164-8_12
11. Binh, H., Nghia, N.: New multi-parent recombination in genetic algorithm for solving bounded diameter minimum spanning tree problem. In: First Asian Conference on Intelligent Information and Database Systems, pp. 283–288 (2009)
12. Saha, S., Kumar, R.: Bounded-diameter MST instances with hybridization of multi-objective EA. Int. J. Comput. Appl. **18**(4), 17–25 (2011). (0975 – 8887)
13. Handler, G.Y.: Minimax location of a facility in an undirected graph. Transp. Sci. **7**, 287–293 (1978)
14. Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. **36**, 1389–1401 (1957)
15. Patvardhan, C., Prakash, V.P., Srivastav, A.: Fast heuristics for large instances of the Euclidean bounded diameter minimum spanning tree problem. Informatica **39**(2015), 281–292 (2015)
16. Patvardhan, C., Prakash, V.P., Srivastav, A.: Parallel heuristics for the bounded diameter minimum spanning tree problem. In: India Conference (INDICON), 2014 Annual IEEE, 11–13 December 2014, pp. 1–5. IEEE Press (2014)
17. Beasley's OR Library, Department of Mathematical Sciences, Brunel University, UK. http://people.brunel.ac.uk/mastjjb/orlib/files. Accessed 21 Jan 2017