Soft Computing (2020) 24:2169–2186
https://doi.org/10.1007/s00500-019-04051-x

METHODOLOGIES AND APPLICATION

A hybrid genetic algorithm for the degree-constrained minimum spanning tree problem

Kavita Singh[1] · **Shyam Sundar**[1]

Published online: 14 May 2019

Abstract
Given an undirected, connected, edge-weighted graph G and a positive integer d, the degree-constrained minimum spanning tree (dc-MST) problem aims to find a minimum spanning tree T on G subject to the constraint that each vertex is either a leaf vertex or else has degree at most d in T, where d is a given positive integer. The dc-MST is $\mathcal{NP}$-hard problem for $d \geq 2$ and finds several real-world applications. This paper proposes a hybrid approach ($\mathcal{H}$SSGA) combining a steady-state genetic algorithm and local search strategies for the this problem. An additional step (based on perturbation strategy at a regular interval of time) in the replacement strategy is applied in order to maintain diversity in the population throughout the search process. On a set of available 107 benchmark instances, computational results show the superiority of our proposed $\mathcal{H}$SSGA in comparison with the state-of-the-art metaheuristic techniques.

Keywords Degree-constrained · Spanning tree · Steady-state genetic algorithm · Problem-specific crossover operator · Local search · Replacement strategy

1 Introduction

Given an undirected, connected and edge-weighted graph $G(V, E, w)$, where V is the set of nodes or vertices; E is the set of edges; and $w(i, j)$ is a positive weight that is associated with each edge $e_{ij} \in E$ whose end points are i and j vertices, the degree-constrained minimum spanning tree (dc-MST) problem aims to find a minimum spanning tree (T) of G such that each vertex is either a leaf vertex or else has degree at most d in T, where d is a given positive integer.

There is a rich literature related to dc-MST problem. For example, minimum branch vertices and minimum degree sum of branch vertices based spanning tree problems (Cerrone et al. 2014; Gargano et al. 2002; Marín 2015; Moreno et al. 2018; Silvestri et al. 2017; Sundar and Singh 2012), and a closely related version of dc-MST has been recently introduced in an uncertain random network, where some weights are uncertain variables and others are random variables (Gao et al. 2017).

The dc-MST is $\mathcal{NP}$-Hard for $d \geq 2$ (Garey and Johnson 1979). This problem finds several real-world applications, such as in the context of backplane wiring among pins, where any pin could be wrapped by at most a fixed number of wire-ends on the wiring panel (Boldon et al. 1996); in designing the road system that can be used to serve the suburbs with the constraint that no more than four roads may meet at any crossing (Savelsbergh and Volgenant 1985); in VLSI designing, where the number of transistors that can be driven by the output current of a transistor is the degree bound for VLSI routing trees (Boldon et al. 1996); in electrical circuits design (Narula and Ho 1980); and in communication networks where the maximum degree in a spanning tree is a measure of vulnerability to single-point failures (Ravi et al. 1993).

The dc-MST is a well-studied problem. Many approaches including exact as well as heuristic approaches have been developed for this problem. Among these approaches, Narula and Ho (1980) proposed a primal and a dual heuristic procedure and a branch-and-bound algorithm for this problem. Later, two general heuristic and a branch-and-bound algo-

Communicated by V. Loia.

✉ Shyam Sundar
ssundar.mca@nitrr.ac.in

Kavita Singh
ksingh.phd2015.mca@nitrr.ac.in

[1] Department of Computer Applications, National Institute of Technology Raipur, Raipur 492010, India

rithm were proposed (Savelsbergh and Volgenant 1985). Boldon et al. (1996) proposed four heuristic based on Prim's algorithm (Prim 1957).

Literature has also witnessed a number of metaheuristic approaches for this problem. Among metaheuristic approaches, various versions of genetic algorithm (GA) have been proposed. For example, Zhou and Gen (1997) presented a GA based on Prufer-encoding, Knowles and Corne (2000) presented a GA based on a $|V|\times$ (d-1) array encoding, and Raidl and Julstrom (2000) presented GA using a weight-coding. Later, Raidl and Julstrom (2003) further presented two versions (ES-EA and HES-EA) of evolutionary approach in which the first version (ES-EA) demonstrates the usefulness of the edge-set encoding, and the second version (HES-EA) which also uses edge-set encoding, but incorporates edge-cost heuristic in the initialization, crossover operator and mutation operator. The basic idea behind edge-cost heuristic in HES-EA (Raidl and Julstrom 2003) is to include low-cost edges into a candidate solution with higher probabilities than high-cost edges. Experimental results show that the edge-set encoded with edge-cost heuristic, i.e., HES-EA, performs better than the previous versions of GA.

In addition, various ant colony optimization approaches (Bau et al. 2005; Bui and Zrncic 2006; Doan 2007) as well as particle swarm optimization approaches (Binh and Nguyen 2008; Ernst 2010) have been proposed for the dc-MST problem. Bui et al. (2012) proposed an ant-based algorithm (ABA) for the dc-MST problem. In ABA, ants, while exploring the graph, identify a subset of edge-set so that a degree-constrained spanning tree can be constructed from this set of edges. ABA uses two local search strategies—2-EdgeReplacement and 1-EdgeReplacement—to further improve the solution quality of currently constructed solution. In 2-EdgeReplacement, two edges associated with the current feasible degree-constrained spanning tree (say T) are examined for replacement with the two new edges $\in E \setminus T$ without violating the degree constraint of T with the aim of further reduction in the weight of T. Whereas, in 1-EdgeReplacement, an edge in the current T is examined for replacement with a new edge $\in E \setminus T$ without violating the degree constraint of T with the aim of further reduction in the weight of T.

GA is a well-known evolutionary algorithm and has an array of various encodings, genetic operators (crossover and mutation operators) for combinatorial optimization problems. Even GA is flexible to integrate with problem-specific heuristic in order to find high-quality solutions to the combinatorial optimization problem under consideration, leading to various variants of GA or hybrid GAs for the same combinatorial optimization problem in the literature. For the dc-MST problem, many researchers have proposed many variants of GA or hybrid GA (Raidl and Julstrom 2000, 2003; Zhou and Gen 1997) in search of finding high-quality solutions. In this paper, we also develop a variant of hybrid genetic algorithm (hybrid approach) for the dc-MST problem. The motivation behind the development of hybrid approach is our new designed problem-specific crossover operator which is quite different from the existing crossover operators including crossover operator of HES-EA (Raidl and Julstrom 2003) (see Sect. 3.5). To make our hybrid approach effective and robust, we incorporate various strategies (such as local search strategies, if applied, are used to intensify the search around the generated child solution and an additional step (based on perturbation strategy at a regular interval of time) in the replacement strategy is applied in order to maintain diversity in the current population throughout the search process) that try to balance the trade-off between exploitation and exploration throughout the search. Hence, our hybrid approach combines a steady-state genetic algorithm and local search strategies for the dc-MST problem. On the available 107 benchmark instances, experimental results show the superiority of our proposed hybrid approach in comparison with the best-so-far hybrid GA (i.e., HES-EA Raidl and Julstrom 2003) and other state-of-the-art metaheuristic technique (ABA Bui et al. 2012). Hereafter, our proposed hybrid approach will be referred to as $\mathcal{H}$SSGA. Note that our proposed hybrid approach $\mathcal{H}$SSGA is quite different the hybrid approach (HES-EA) (Raidl and Julstrom 2003) on mainly three components—problem-specific crossover operator, local search strategies and an additional step in the replacement strategy. Also, note that local search strategies in $\mathcal{H}$SSGA which is based on *two-edges replacement* (referred to as 2ER) and *one-edge replacement* (referred to as 1ER) are applied conditionally. 2ER follows the idea of 2-EdgeReplacement local search strategy used in ABA (Bui et al. 2012), but 1ER which is common idea is different from 1-EdgeReplacement local search strategy used in ABA (Bui et al. 2012).

The organization of the remaining paper is as follows: Sect. 2 presents a brief discussion on steady-state genetic algorithm (SSGA); Sect. 3 presents our proposed hybrid approach ($\mathcal{H}$SSGA) for the dc-MST problem; Sect. 4 presents computational results; and Sect. 5 presents concluding remarks.

2 Steady-state genetic algorithm

Genetic algorithm (GA) is a stochastic search technique that is stem from the principles of natural evolution (Holland 1975). In nature, during the evolution of the population, individuals in the population compete with each other to survive. Individuals who are more fit remain intact, while less fit individuals do not survive. Similarly, in GA, more fit chromosomes (solutions) have higher chances to participate in genetic operators and to propagate their genes from one

generation to another. Genetic operators help GA in exploiting the promising regions of the search space as well as in exploring new region of the search space. Readers who are interested in a general introduction to GA and its applications may find in Cerrone et al. (2016), Goldberg (1989), Iordache et al. (2007) , Pop et al. (2013).

This paper presents a hybrid steady-state genetic algorithm for the dc-MST problem. Steady-state GA (SSGA) is different from generational GA (GGA) (Davis 1991), as GGA, in each generation, generates a population of new child solutions from the old population with the help of genetic operators and replaces usually the current parent population with the newly generated child population. While SSGA, in each generation, typically generates a single new child solution from the old population with the help of genetic operators and replaces an individual (solution) in the current population with the newly generated child solution.

3 Hybrid steady-state genetic algorithm for the dc-MST

Algorithm 1: The pseudocode of $\mathcal{H}$SSGA

Input : A connected, edge-weighted and undirected complete graph $G = (V, E, w)$, and a positive integer constant d
Output: A degree-constrained spanning tree T^{gb}

1 Generate a population of initial solutions $< T_1, T_2, \ldots, T_{pop} >$ of size pop;
2 $T^{gb} \leftarrow$ Best-so-far solution in the population;
3 **while** *Termination criterion is not met* **do**
  // $u01$ is a uniform variate
4 **if** $\mu 01 < P_c$ **then**
5 $p_1 \leftarrow BTS(T_1, T_2, \ldots, T_{pop})$;
6 $p_2 \leftarrow BTS(T_1, T_2, \ldots, T_{pop})$;
7 $T^C \leftarrow Xover(p_1, p_2)$;
8 **else**
9 $p_1 \leftarrow BTS(T_1, T_2, \ldots, T_{pop})$;
10 $T^C \leftarrow Mut(p_1)$;
  // See Sect. 3.7 for the local search
11 **if** $(\frac{F(T^{gb})}{F(T^C)} + \alpha \times dis(T^{gb}, T^C)) > 1)$ **then**
12 Apply LS on T^C;
13 **if** T^C *is Unique* **then**
  // Apply replacement strategy
14 **if** T^C *is better than* T^{gb} **then**
15 $T^{gb} \leftarrow T^C$;
16 Replace a solution of the current population, whose fitness is greater than the average fitness of the current population, with T^C;
17 **if** T^{gb} *does not improve a certain number of generations* **then**
18 Apply population update strategy (RS+); // See Sect. 3.8 for Replacement Strategy

This section discusses the framework of our proposed hybrid approach ($\mathcal{H}$SSGA) for the dc-MST problem that combines a steady-state genetic algorithm and local search strategies.

Algorithm 1 presents the pseudocode of $\mathcal{H}$SSGA for the dc-MST problem, where $< T_1, T_2, \ldots, T_{pop} >$ are feasible solutions of the population with population size pop; T^{gb} stores the best-so-far solution; $BTS(T_1, T_2, \ldots, T_{pop})$ is a function of binary tournament selection method that returns a parent solution; $Xover(p_1, p_2)$ is a function of crossover operator applied on two selected parent solutions (p_1 and p_2) and returns a child solution T^C; and $Mut(p_1)$ is a function of mutation operator applied on the selected parent solution (p_1) and returns a child solution T^C. Both crossover ($Xover$) and mutation (Mut) operators are applied in a mutually exclusive way. $u01$ is a uniform variate, and P_c is a probability parameter that is to be determined empirically. Once the child solution T^C is generated, local search strategies (LS) based on *two-edge replacement* (2ER) and *one-edge replacement* (1ER) methods are applied conditionally (See line no. 11 of Algorithm 1) in order to further improve the solution quality of T^C. Hereafter, if the current child solution T^C is found to be unique, then it is inserted into the current population by replacing an individual (solution) of the current population whose fitness is greater than the average fitness of its current population. If T^C is not unique, T^C is discarded. An additional step in the replacement strategy is applied in order to maintain diversity in the current population throughout the search process. Once the termination criterion is met, $\mathcal{H}$SSGA stops executing.

The following subsections discuss the details of various components of $\mathcal{H}$SSGA.

3.1 Encoding

To represent a chromosome (solution or spanning tree (T_i)), an edge-set encoding (Raidl and Julstrom 2003) is used. As per this encoding, each solution T_i consists of a set of $|V|$-1 edges. The reason to choose this encoding is that it not only offers high locality and heritability, but also adaptive to problem-specific genetic operators.

3.2 Generation of initial solutions of the population

$\mathcal{H}$SGGA follows Prim's algorithm (Prim 1957) to generate an initial solution of the population. Initially, a degree-constrained spanning tree (T_i) and the set S are empty; create a copy, say U, of V; label each vertex $v \in V$ *unmarked* ($mark[v] \leftarrow 0$); and set the degree of each vertex $v \in V$ of T_i to *zero* ($deg[v] \leftarrow 0$). Select a vertex $v_1 \in U$ randomly, and add this selected v_1 to S. Delete this selected v_1 from U. Label v_1 *marked* ($mark[v_1] \leftarrow 1$). Select a random edge $e_{v_1 v_2}$ that connects a vertex $v_1 \in S$ to a vertex $v_2 \in U$,

and add this selected edge $e_{v_1v_2}$ to T_i. Increment the value of $deg[v_1]$ and $deg[v_2]$ in T_i by one due to addition of an edge $e_{v_1v_2}$ to T_i. Add v_2 to S, and then delete v_2 from U. Label v_2 *marked* ($mark[v_2] \leftarrow 1$). Hereafter, iteratively at each step, search a random edge (say e_{ij}) that connects a vertex $i \in S$ ($deg[i] < d$) to an *unmarked* vertex $j \in U$. Add this searched e_{ij} to T_i and increment the value of $deg[i]$ and $deg[j]$ in T_i by one. Add j to S, and then delete j from U. Label j *marked* ($mark[j] \leftarrow 1$). This whole procedure is repeated again and again until U becomes empty. At this juncture, a feasible degree-constrained spanning tree (solution) T_i is constructed.

Hereafter, uniqueness of each generated T_i is checked against the initial solutions of the population generated so far. If the current generated initial solution is unique, it is included into the population, otherwise it is discarded.

3.3 Fitness

Once an initial solution T_i is generated, its fitness (say ($F(T_i)$) is computed as the sum of weight of edges in T_i.

3.4 Selection

$\mathcal{H}$SGGA follows binary tournament selection method to select a parent solution. This method is applied two times in order to select two parent solutions for the crossover operator, whereas this method is applied one time in order to select a parent solution for the mutation operator. As per this method, two different solutions are picked uniformly at random from the current population. With probability (P_b), fitter one between these two selected solutions is selected as a parent solution, otherwise the worse one is selected as a parent solution with probability (1 -P_b).

3.5 Crossover operator

Our proposed crossover operator (*Xover*) is a problem-specific crossover operator (say *Xover*) that tries to inherit good edges of parent individuals in the newly generated child solution (say T^C) as much as possible and at the same time *Xover* maintains the degree constraint of all non-leaf vertices of T^C. Algorithm 2 presents the pseudocode of *Xover* for $\mathcal{H}$SSGA whose description is as follows:

Xover starts with selecting two chromosomes (solutions) as parents (say, p_1 and p_2) from the population with the help of binary tournament selection method. Initially, consider the degree-constrained spanning tree T associated with an empty solution (say child solution T^C) as an empty set, and also consider a set (say S) as an empty set. Set the degree of each vertex $v \in V$ in T zero (i.e., $deg[v] \leftarrow 0\ \forall v \in V$), and label each vertex $v \in V$ *unmarked* ($mark[v] \leftarrow 0$).

Algorithm 2: The pseudocode of crossover operator (*Xover*) in $\mathcal{H}$SSGA

Input : Two different parent individuals (p_1 and p_2)
Output: A child solution T^C (a degree-constrained spanning tree T)

```
1  T ← ∅, S ← ∅, deg[i] ← 0 ∀i ∈ V in T, mark[i] ← 0 ∀i ∈ V;
2  Pick a vertex v1 ∈ V randomly;
3  S ← S ∪ v1, mark[v1] ← 1;
   // u01 is a uniform variate
4  if (u01 < Pb_{p1p2}) then
5      Pick an edge e_{v1v2}, connecting v1 ∈ S to v2 ∈ V \ S, randomly
       from p1;
6  else
7      Pick an edge e_{v1v2}, connecting v1 ∈ S to v2 ∈ V \ S, randomly
       from p2;
8  mark[v2] ← 1, S ← S ∪ v2, T ← T ∪ e_{v1v2}, deg[v1] + +,
   deg[v2] + +;
9  while (V \ S ≠ ∅) do
10     if (u01 < Pb_{p1p2}) then
11         for (each vertex i ∈ S) do
12             if (deg[i] < d) then
13                 Search a random edge e_ij (i ∈ S and j ∈ V \ S) from
                   p1;
14                 if (the search is successful) then
15                     mark[j] ← 1, S ← S ∪ j, T ← T ∪ e_ij,
                       deg[i] + +, deg[j] + +, break;
16         if (the search is not successful) then
17             for (each vertex i ∈ S) do
18                 if (deg[i] < d) then
19                     Search a random edge e_ij (i ∈ S and j ∈ V \ S)
                       from p2;
20                     if (the search is successful) then
21                         mark[j] ← 1, S ← S ∪ j, T ← T ∪ e_ij,
                           deg[i] + +, deg[j] + +, break;
22             if (the search is not successful) then
23                 Add an edge e_ij ∈ E \ T of minimum edge-weight,
                   connecting a vertex i ∈ S (deg[i] < d) to a vertex
                   j ∈ V \ S;
24                 mark[j] ← 1, S ← S ∪ j, T ← T ∪ e_ij,
                   deg[i] + +, deg[j] + +;
25     else
26         for (each vertex i ∈ S) do
27             if (deg[i] < d) then
28                 Search a random edge e_ij (i ∈ S and j ∈ V \ S) from
                   p2;
29                 if (the search is successful) then
30                     mark[j] ← 1, S ← S ∪ j, T ← T ∪ e_ij,
                       deg[i] + +, deg[j] + +, break;
31         if (the search is not successful) then
32             for (each vertex i ∈ S) do
33                 if (deg[i] < d) then
34                     Search a random edge e_ij (i ∈ S and j ∈ V \ S)
                       from p1;
35                     if (the search is successful) then
36                         mark[j] ← 1, S ← S ∪ j, T ← T ∪ e_ij,
                           deg[i] + +, deg[j] + +, break;
37             if (the search is not successful) then
38                 Add an edge e_ij ∈ E \ T of minimum edge-weight,
                   connecting a vertex i ∈ S (deg[i] < d) to a vertex
                   j ∈ V \ S;
39                 mark[j] ← 1, S ← S ∪ v2, T ← T ∪ e_{v1v2},
                   deg[v1] + +, deg[v2] + +;
```

At the beginning of *Xover*, add a vertex $v_1 \in V$, which is selected uniformly at random, to S. Label v_1 *marked* ($mark[v_1] \leftarrow 1$). With probability ($Pb_{p_1p_2}$), pick a random edge among all candidate edges in p_1, otherwise pick a random edge among all candidate edges in p_2 with probability (1-$Pb_{p_1p_2}$). Here a candidate edge is an edge that connects a vertex in S to a vertex $v_2 \in V \setminus S$; and the probability $Pb_{p_1p_2}$ is defined as $\frac{1/F(p_1)}{1/F(p_1)+1/F(p_2)}$ (Beasley and PC 1996). $F(p_1)$ and $F(p_2)$, respectively, are the fitness of p_1 and p_2. Such probability mechanism favors the fitter parent so that more and more number of edges of fitter parent would participate in constructing T of T^C in comparison with that of lesser fit parent. Add this selected edge $e_{v_1v_2}$ to T of T^C. Increment the value of $deg[v_1]$ and $deg[v_2]$ by one. Add v_2 to S, and label v_2 *marked* ($mark[v_2] \leftarrow 1$). Hereafter, iteratively at each step, search a random edge (say e_{ij}) that connects a vertex $i \in S$ ($deg[i] < d$) to an *unmarked* vertex $j \in V \setminus S$, from either p_1 with probability ($Pb_{p_1p_2}$) or p_2 with probability ($1 - Pb_{p_1p_2}$). If the search is successful, add this searched e_{ij} to T of T^C. Increment the value of $deg[i]$ and $deg[j]$ by one. Add j to S, and label j *marked* ($mark[j] \leftarrow 1$). If the search is not successful, *Xover* switches to other parent in order to search another random edge (say e_{ij}). If the search is successful, add this searched edge e_{ij} to T of T^C. Increment the value of $deg[i]$ and $deg[j]$ by one. Add j to S, and label j *marked* ($mark[j] \leftarrow 1$). If the search is still not successful, then search a minimum edge-weight edge (say e_{ij}) that connects a vertex $i \in S$ ($deg[i] < d$) to a vertex $j \in V \setminus S$ from E-{ edges of both parent solutions p_1 and p_2 }. Add this searched edge e_{ij} to T of T^C. Increment the value of $deg[i]$ and $deg[j]$ by one. Add j to S, and label j *marked* ($mark[j] \leftarrow 1$). This procedure is repeated until $V \setminus S$ becomes empty. At this juncture, a feasible degree-constrained spanning tree T of T^C (newly generated child solution) is constructed.

Though our *Xover* shares similar ideas of crossover operator used in ES-EA (Raidl and Julstrom 2003) and HES-EA (Raidl and Julstrom 2003), but the way of inheriting edges from parents is quite different. In ES-EA, the crossover operator selects edges of parents in a random order, whereas in HES-EA, the crossover operator uses edge-cost heuristic to select edges of parents. The basic idea of edge-cost heuristic is to include low-cost edges into a candidate solution with higher probabilities than high-cost edges. After considering edges from both parents in ES-EA and HES-EA, if the generated degree-constrained spanning tree T associated with the child solution is not feasible, then to make it feasible, edges are selected from the set of edges $E \setminus T$.

We present an example that illustrates the difference between crossover operator used in HES-EA (Raidl and Julstrom 2003) and our proposed *Xover*. For that we consider the dc-MST problem with $d = 3$. In this example, we

Vertices	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	–	1	6	5	3	2	7
v_2	–	–	7	2	2	3	3
v_3	–	–	–	3	3	4	4
v_4	–	–	–	–	4	2	2
v_5	–	–	–	–	–	1	3
v_6	–	–	–	–	–	–	2
v_7	–	–	–	–	–	–	–

Fig. 1 Weight matrix C_1 of graph G_1

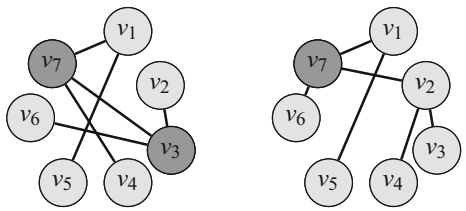


Fig. 2 Parents p_1 and p_2

consider an edge-weighted, undirected and connected graph $G_1(V, E, w)$, where $|V| = 7$; $|E| = 21$; and a weight (w) is associated with each edge $\in E$ (one can see Fig 1). Figure 1 presents the edge-weight matrix (C_1) of G_1. Figure 2a, b represents two parent solutions p_1 and p_2, respectively. The fitness of p_1 and p_2 is 27 and 24, respectively. To generate a child solution T^C, *Xover* for $\mathcal{H}$SSGA starts with selecting a vertex (say v_7) randomly (see Fig. 3a). *Xover* probabilistically selects an edge $e_{v_7v_6}$ that connects a vertex $v_7 \in S$ to a vertex $v_6 \in V \setminus S$ from parent p_2 (as per Algorithm 2) and adds this selected edge to the empty degree-constrained spanning tree (say T) of T^C (see Fig. 3b). Hereafter, iteratively, at each step, *Xover* selects an edge either from p_1 with probability ($Pb_{p_1p_2}$) or from p_2 with probability (1-$Pb_{p_1p_2}$). Once an edge is selected, it is added to T. Continuing this iterative process, an edge $e_{v_7v_2}$ is selected from parent p_2 and is added to T of T^C (Fig. 3c). In a similar way, $e_{v_7v_4}$ and $e_{v_6v_3}$, respectively, selected from parent p_1 and p_1 are added to T of T^C (see Fig. 3d, e). Figure 3e shows the situation of no candidate edge from the current parent p_1, then *Xover* switches to p_2 in order to select a candidate edge; however, *Xover* also fails to find a single candidate edge after switching to p_2. *Xover*, then, greedily selects a candidate edge $e_{v_6v_5} \in E \setminus T$ (see Fig. 3f). Similarly, edge $e_{v_1v_2} \in E \setminus T$ is also selected and is added to T of T^C (see Fig. 3g). Figure 3g presents a feasible generated T of T^C whose fitness is 13. Figure 4a–f illustrates how the crossover operator in HES-EA (Raidl and Julstrom 2003) is applied to generate a child solution T^C. Figure 4a

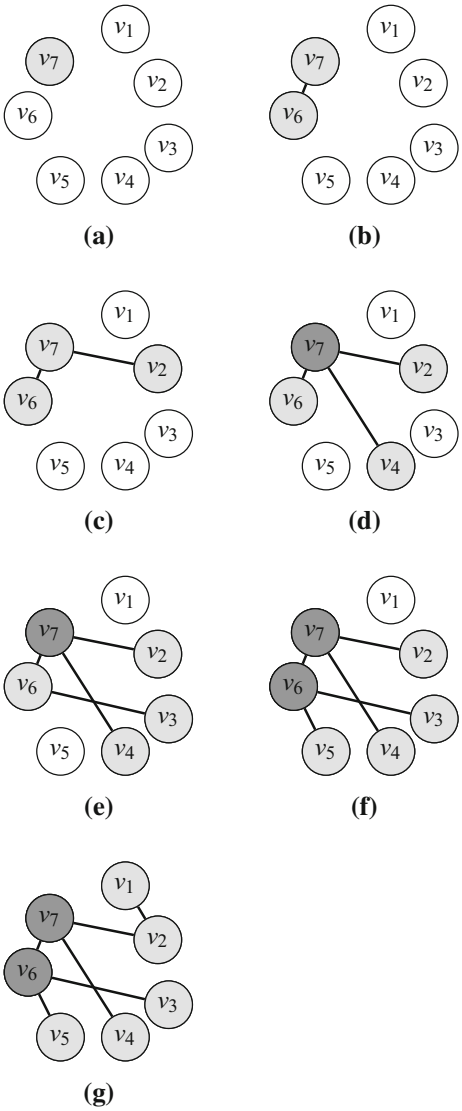


Fig. 3 *Xover* in $\mathcal{H}$SSGA

shows the set of edges ($E_{p_1p_2}$) that are common ($E_{p_1} \cap E_{p_2}$) to both parents (p_1 and p_2), where E_{p_1} and E_{p_2} are the set of edges of p_1 and p_2, respectively. Figure 4b shows the set of edges (say E') that includes $(E_{p_1} \cup E_{p_2}) - (E_{p_1} \cap E_{p_2})$ edges from both parents. According to crossover operator used in HES-EA (Raidl and Julstrom 2003), first it includes all edges from the set $E_{p_1p_2}$ to the empty degree-constrained spanning tree (T) of the child solution T^C (see Fig. 4c). Hereafter, iteratively, at each step, it greedily selects an edge from the set E' without violating the degree constraint of T of T^C. Continuing this iterative process, first an edge $e_{v_7v_6}$ is selected (see Fig. 4d). Hereafter, edges $e_{v_7v_4}$ and $e_{v_4v_2}$ are selected and are added to T of T^C (see Fig. 4e, f). Figure 4f denotes the resultant feasible T of T^C whose fitness is 23. ES-EA (Raidl and Julstrom 2003) follows the same procedure to generate T^C with the difference that it selects edges randomly from the set E'.

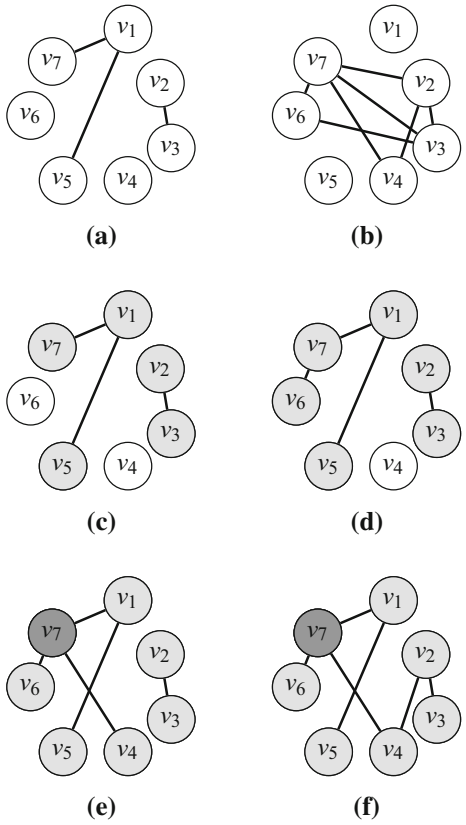


Fig. 4 Crossover in HES-EA (Raidl and Julstrom 2003)

3.6 Mutation operator

The role of mutation operator is used to provide diversity in the population. In $\mathcal{H}$SSGA, mutation operator (referred to as Mut) starts with a parent solution (p_1) selected from the population with the help of binary tournament selection method. Copy this selected parent solution to an empty child solution (say T^C). Hereafter, Mut performs *edge-deletion-insertion* operation on T^C. In this operation, first deletion of an edge (say e_{uv}) selected randomly from the spanning tree T of T^C is performed, leading to the partition of T into two different components (say T_u and T_v). To connect these two components, an edge (different from e_{uv}) is searched in $E \setminus T$ in such a way that the degree constraint of the resultant T does not get violated after insertion of this searched edge. Note that ES-EA (Raidl and Julstrom 2003) and HES-EA (Raidl and Julstrom 2003) follow *edge-insertion-deletion* in mutation operator, whereas Mut follows *edge-deletion-insertion*. HES-EA (Raidl and Julstrom 2003) inserts low-weight edge instead of a random edge like ES-EA.

Also note that similar to (Sundar 2014; Sundar and Singh 2015, 2017), crossover operator (*Xover*) and mutation operator (*Mut*) for $\mathcal{H}$SSGA are applied in a mutually exclusive way which is different from the way crossover and mutation operators are applied in ES-EA (Raidl and Julstrom 2003)

and HES-EA (Raidl and Julstrom 2003). With probability P_c, *Xover* is selected, otherwise Mut is selected with the probability (1-P_c). The reason behind this one is that *Xover* generates T^C with potentially good edges inherited from their parents, whereas Mut generates T^C based on *edge-deletion-insertion* operation. If Mut is applied after *Xover*, then the chances are high that the resultant child solution may lose some potentially good edges inherited from parent solutions of *Xover*.

3.7 Local search strategies (LS)

To further improve the quality of a currently generated child solution T^C, local search strategies (LS) based on *two-edges replacement* (referred to as 2ER) and *one-edge replacement* (referred to as 1ER) methods are applied conditionally. Edge replacement strategy is a common idea that is based on deletion of an edge $e_{uv} \in T$ of T^C and inclusion of a new edge $e_{xy} \in E$. LS are applied on T^C only when the following condition holds true:

$$\left(\frac{F(T^{gb})}{F(T^C)} + \alpha \times dis(T^{gb},\ T^C)\right) > 1; \tag{1}$$

where T^{gb} is the best-so-far generated solution; $F(T^{gb})$ and $F(T^C)$ are, respectively, the fitness of T^{gb} and T^C; α is a parameter to be determined empirically; and $dis(T^{gb},\ T^C)$ denotes the distance between T^{gb} and T^C in terms of fraction of edges of T^C that are not common to the edges of T^{gb}. The first part $\frac{F(T^{gb})}{F(T^C)}$ in conjunction with the second part $\alpha \times dis(T^{gb},\ T^C)$ of Eq. 1 relates to the *quality-and-distance* feature. The rationale behind using this condition (equation) is that this equation avoids applying LS on such generated child solution which is slightly inferior to $F(T^{gb})$ (*quality*), but is not sufficiently far from T^{gb} (*distance*). This way also saves the computational time. If the current child solution T^C satisfies equation 1, then the LS will be applied on T^C. LS is applied on T^C in the following order: *two-edges replacement* (2ER) $\rightarrow$ *one-edge replacement* (1ER). Descriptions of 2ER and 1ER are as follows:

2ER in $\mathcal{H}$SSGA follows the idea of 2-EdgeReplacement local search strategy (Bui et al. 2012); however, 2-EdgeReplacement local search strategy in Bui et al. (2012) uses $|V|/2$ iterations until no improvement is possible in Bui et al. (2012), whereas 2ER is applied at most three iterations. In each iteration of 2ER, a random edge (say $e_{uv} \in T$ of T^C) is picked, and among all candidate non-adjacent edges of e_{uv} in T of T^C, only that edge (say e_{wx}) is picked if replacement of e_{uv} and e_{wx} with two new edges $e_{uw} \in E \setminus T$ and $e_{vx} \in E \setminus T$ leads to the maximum improvement in the resultant T of T^C.

Algorithm 3: The pseudocode of 1ER of LS

Input : Degree-constrained spanning tree T of the current child solution T^C
Output: A new feasible spanning tree T of T^C, iff, 1ER takes place

1 $it \leftarrow 3$;
2 $count \leftarrow 0$;
3 $F_c \leftarrow F(T^C)$;
4 **while**($count < it$) **do**
  // First stage of 1ER
5 **for** *(each edge $e_{uv} \in T$ whose degree of at least one end vertex degree is equal to d (i.e., $deg[u] == d \mid\mid deg[v] == d$))* **do**
6 Search an edge $e_{xy} \in E \setminus T$ whose $w(x, y) \leq w(u, v)$ and $e_{xy} \cup T$ does not violate the degree constraint of the resultant T, if *edge-replacement* takes place;
7 **if** (*the search is successful*) **then**
8 $T \leftarrow (T - e_{uv}) \cup e_{xy}$;
9 $F_c \leftarrow (F_c - w(u, v)) + w(x, y)$;
  // Second stage of 1ER
10 **for** (*each edge $e_{uv} \in T$*) **do**
11 Search an edge $e_{xy} \in E \setminus T$ whose $w(x, y) < w(u, v)$ and $e_{xy} \cup T$ does not violate the degree constraint of the resultant T, if *edge-replacement* takes place;
12 **if** (*the search is successful*) **then**
13 $T \leftarrow (T - e_{uv}) \cup e_{xy}$;
14 $F_c \leftarrow (F_c - w(u, v)) + w(x, y)$;
15 **if** ($F(T^C) > F_c$) **then**
16 $F(T^C) \leftarrow F_c$;
17 $count \leftarrow count + 1$;

1ER in $\mathcal{H}$SSGA examines the edges of T of T^C for possible *edge-replacement* in two stages followed one-by-one. In the first stage of 1ER, for each edge $e_{uv} \in T$ whose degree of at least one end point (vertex) is equal to d (i.e., $deg[u] == d$ or $deg[v] == d$), search an appropriate edge $e_{xy} \in E \setminus T$ whose $w(x, y) \leq w(u, v)$ for *edge-replacement* without violating the degree constraint of the resultant T of T^C if such *edge-replacement* takes place. If the search is successful, then e_{uv} is replaced with e_{xy} in T (see Algo. 3, line no. 5–9). The idea behind this *edge-replacement* is that if the degree of the vertex v ($deg[v] == d$) in T is reduced, then in the second stage, there will be possibility that the edge with lesser weight may attach to v. Keeping this idea, after completion of the first stage, the second stage is applied. In the second stage, for each edge $e_{uv} \in T$, search an appropriate edge $e_{xy} \in E \setminus T$ whose edge-weight $w(x, y)$ is less than edge-weight $w(u, v)$ of e_{uv} for exchange. If the search is successful, replace e_{uv} with e_{xy} in T (see Algo. 3, line no. 10–14), resulting in the reduction of weight of the resultant T of T^C.

Note that our proposed 1ER for $\mathcal{H}$SSGA is different from 1-EdgeReplacement (Bui et al. 2012). 1ER consists of two stages and is based on first-fit improvement strategy, whereas 1-EdgeReplacement (Bui et al. 2012) consists of only one stage and is based on best-fit improvement. Also, 1-EdgeReplacement in Bui et al. (2012) is applied repeatedly until no improvement is possible, whereas 1ER is applied at most three iterations (see the pseudocode of 1ER of LS in Algorithm 3).

3.8 Replacement strategy (RS)

Uniqueness of each newly generated child solution T^C is checked against all individuals of the current population. If T^C is found to be unique, then T^C replaces a randomly selected solution of the current population whose fitness is greater than the average fitness of the current population.

In addition to the above replacement strategy (RS), we follow one more step in this replacement strategy (referred to as RS+). This strategy is followed only when T^C is unique. The need of RS+ was felt intuitively during our initial experimentations that the search process of $\mathcal{H}$SSGA without RS+ often got trapped into local optimum, indicating the lack of carrying out diversity in the population generation-over-generation in the search space effectively. RS+ is applied only when this indication is identified. Our assumption of this identification is based on this fact if the best-so-far generated solution (T^{gb}) stops emerging (in terms of better solution quality) for a certain number of generations, say PR_{pop} (PR_{pop} is set to 500 empirically after a large number of trials) in the course of search. In RS+, a perturbation strategy is applied on the current population at a regular interval of time (i.e., PR_{pop}). In this perturbation strategy, a subset (say rs) of individuals (solutions) of the current population is selected randomly, where rs is a parameter to be determined empirically. Each solution (say $T_i \in rs$) is perturbed with the help of mutation operator Mut (see Sect. 3.6). If the perturbed solution $T^{'}$ of T_i is unique against all individuals of the current population, then $T^{'}$ is included into the current population by replacing its own old solution T_i. Otherwise, $T^{'}$ is discarded. Applying perturbation strategy on a set of random solutions (rs) force the population to be diversified throughout the search process, helping in finding high-quality solutions.

4 Computational results

$\mathcal{H}$SSGA is implemented in C and executed on a Linux-based operating system with the configuration of Intel Core *i5* processor 3.3 GHz $\times$ 4 with 4 GB RAM. In all our experiments with $\mathcal{H}$SSGA, we have used $pop = 300$ (population of solutions), $P_b = 0.90$ (see Sect. 3.4) and $P_c = 0.50$ (crossover probability, see Sect. 3.6), $\alpha = 0.10$ (see Sect. 3.7), and $rs = 0.05$ (5% solutions of pop are perturbed (see Sect. 3.8)). All these parameters are set empirically after a large number of trials. Although these parameter values provide good results on most of the instances, they may not be optimal for all instances. We have tested $\mathcal{H}$SSGA on the available 107 benchmark instances which were also used for the recent one ant-based algorithm (ABA) (Bui et al. 2012). Some of these instances were also used for HES-EA (Raidl and Julstrom 2003). These instances can be classified into two groups—Euclidean and non-Euclidean instances. Euclidean instances consist of three different sets, i.e., CRD, SYM and STR, whereas non-Euclidean instances consist of three different sets, i.e., SHRD, random hard (R) and misleading hard (M). These benchmark instances can be downloaded from the link https://turing.cs.hbg.psu.edu/benchmarks/file_instances/spanning_tree/. The descriptions of 107 benchmark instances that are classified into six different data sets with varying sizes from 15 to 500 vertices (nodes) are as follows:

- **CRD data set** This data set consists of graphs whose sizes vary from 50 to 100 vertices. Vertices in such a graph are generated by using a uniform distribution in a two-dimensional plane and edge-weight is the Euclidean distance between two vertices.
- **SYM data set** This data set consists of graphs whose sizes vary from 50 to 70 vertices. These graphs are analogous to the CRD instances except vertices are generated by using a uniform distribution in a higher dimensional Euclidean space. Edge-weight is the Euclidean distance between two vertices.
- **STR data set** This data set consists of graphs whose sizes vary from 50 to 100 vertices. Vertices in such a graph are randomly distributed points in a higher dimensional space grouped together as cluster and edge-weight is the Euclidean distance between two vertices.
- **SHRD data set** This data set consists of graphs whose sizes vary from 15 to 30 vertices. These graphs are generated by assignment of non-Euclidean distances to the graph edges in such a way that the number of optimal solutions is limited (Krishnamoorthy et al. 2001).
- **Random-hard (R) data set** This data set consists of graphs whose sizes vary from 50 to 200 vertices. These are non-Euclidean graph instances and edge-weights are randomly generated from a pre-defined interval by using a uniform distribution.
- **Misleading-hard (M) data set** This data set consists of graphs whose sizes vary from 50 to 500 vertices. These are non-Euclidean graph instances and edge-weights are randomly generated from a pre-defined interval by using a uniform distribution. Graphs of M-data sets are designed to mislead greedy algorithms.

We compare our approach $\mathcal{H}$SSGA with two state-of-the-art metaheuristic techniques, i.e., ant-based algorithm (ABA) (Bui et al. 2012) and HES-EA (Raidl and Julstrom 2003). Since authors (Bui et al. 2012) carried out all their experiments on a system based on Intel Core 2 Duo E8600 at 3.33 GHz with 6 GB RAM and executed ABA for 50 runs on each instance. Note that authors (Bui et al. 2012) used available 97 benchmark instances, but results of ABA on 97 instances reported by authors are *partial* in terms of average solution

quality and average total execution time. Out of 97 instances, authors (Bui et al. 2012) reported the best value obtained over 50 runs on all instances, but did not report the average solution quality and average total execution time over 50 runs for 56 and 40 instances, respectively. Also, the computer system used for $\mathcal{H}$SSGA is different from that of ABA (Bui et al. 2012). For HES-EA (Raidl and Julstrom 2003), authors performed 50 runs on each considered instance and used a termination criterion when the best-so-far solution obtained does not improve over 1,00,000 generations on considered instance, but did not report computational time on each considered instance. Authors carried out their experiments on a system based on Pentium-III/800-MHz PC, which is different from the computer system used for $\mathcal{H}$SSGA. Hence, we find difficulty to analyze exact comparison with ABA due to different computer platform and the way the computational results are reported in their paper as well as HES-EA due to different computer platform and unavailability of computational time on each considered instance. Looking at all these aspects of difficulties, we have implemented HES-EA and ABA using same values of parameters mentioned in their respective papers for the purpose of giving a fair comparison with $\mathcal{H}$SSGA. Like $\mathcal{H}$SSGA, we have implemented ABA and HES-EA in C and executed on a Linux-based operating system with the configuration of Intel Core *i5* processor 3.3 GHz $\times$ 4 with 4 GB RAM. All approaches ($\mathcal{H}$SSGA, HES-EA and ABA) have been executed for 50 independent runs on each instance in order to test their robustness. We have set the same stopping criterion (in terms of computational time) for each approach ($\mathcal{H}$SGGA, HES-EA and ABA) as per the instance size (1 s for $|V| < 100$, 10 s for $|V| == 100$, 60 s for $|V| == 200$, 200 s for $|V| == 300$, 600 s for $|V| == 400$, and 1000 s for $|V| == 500$).

Subsequent subsections discuss a detailed comparison of $\mathcal{H}$SSGA with HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012).

4.1 Comparison of $\mathcal{H}$SSGA with HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012)

$\mathcal{H}$SSGA has been compared with HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012) on a set of available 107 benchmark instances. Since ABA uses the two local search strategies based on 2-EdgeReplacement and 1-EdgeReplacement in order to further improve the solution quality of the currently constructed solution, and our proposed approach $\mathcal{H}$SSGA combines a steady-state genetic algorithm (SSGA) and local search strategies based on 1ER and 2ER. Therefore, in addition to analyzing the comparison of $\mathcal{H}$SSGA with ABA, we have also analyzed the individual effect of 1ER and 2ER that combines with only SSGA (including RS+ step) part of $\mathcal{H}$SSGA (referred to as, respectively, (SSGA)+1ER and (SSGA)+2ER) on benchmark instances.

Tables 1, 2 and 3 report the results of HES-EA, ABA, (SSGA)+2ER, (SSGA)+1ER and $\mathcal{H}$SSGA on Euclidean and non-Euclidean instances. In these tables, column *Instance* denotes the name of instance; column $|V|$ denotes the number of vertices corresponding to its instance; column d denotes the degree constraint on its corresponding instance; each next four columns *Best*, *Avg*, *SD* and *ATET*, respectively, denote the best value, the average solution quality, standard deviation and the average total execution time obtained by HES-EA, ABA, (SSGA)+2ER, (SSGA)+1ER and $\mathcal{H}$SSGA over 50 runs. For each instance, the best value (*Best*) and the best average solution quality (*Avg*) among HES-EA, ABA, (SSGA)+2ER, (SSGA)+1ER and $\mathcal{H}$SSGA are highlighted in bold.

Table 1 reports the results of 27 Euclidean instances. Comparing with HES-EA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 19, equal on 6 and is worse on 2 instances; $\mathcal{H}$SSGA, in terms of *Avg*, is better on 18, equals on 1 and is worse on 8 instances. Comparing with HES-EA, (SSGA)+1ER, in terms of *Best*, is better on 20, equals on 6 and is worse on 1 instances; (SSGA)+1ER, in terms of *Avg*, is better on 21, equals on 1 and is worse on 5 instances. Similarly, comparing with HES-EA, (SSGA)+2ER, in terms of *Best*, is better on 8, equals on 4 and is worse on 15 instances; (SSGA)+2ER, in terms of *Avg*, is better on 2, equals on 1 and is worse on 24 instances. Comparing with ABA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 23 and equals on 4 instances; $\mathcal{H}$SSGA, in terms of *Avg*, is better on 26 and equals on 1 instances. Comparing with ABA, (SSGA)+1ER, in terms of *Best*, is better on 22 and equals on 5; (SSGA)+1ER, in terms of *Avg*, is better on 26 and equals on 1 instances. Similarly, comparing with ABA, (SSGA)+2ER, in terms of *Best*, is better on 20, equals on 2 and is worse on 5; (SSGA)+2ER, in terms of *Avg*, is better on 17 equals on 1 and is worse on 9 instances.

Table 2 reports the results of 52 instances. Comparing with HES-EA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 11 and equals on 41 instances; $\mathcal{H}$SSGA, in terms of *Avg*, is better on 26, equals on 24 and is worse on 2 instances. Comparing with HES-EA, (SSGA)+1ER, in terms of *Best*, is better on 11, equals on 40 and is worse on 1 instances; (SSGA)+1ER, in terms of *Avg*, is better on 20, equals on 28 and is worse on 4 instances. Similarly, comparing with HES-EA, (SSGA)+2ER, in terms of *Best*, is better on 5, equals on 29 and is worse on 18 instances; (SSGA)+2ER, in terms of *Avg*, is better on 11, equals on 7 and is worse on 34 instances. Comparing with ABA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 19 and equals on 33 instances; $\mathcal{H}$SSGA, in terms of *Avg*, is better on 31 and equals on 21 instances. Comparing with ABA, (SSGA)+1ER, in terms of *Best*, is better on 19 and equals on 33 instances; (SSGA)+1ER, in terms of *Avg*, is better on 29, equals on 22 and worse on 1 instances. Simi-

Table 1 Results of HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012) and $\mathcal{H}$SSGA for the CRD and SYM instances

Instance	$\|V\|$	d	HES-EA				ABA				(SSGA)+2ER				(SSGA)+1ER				$\mathcal{H}$SSGA			
			Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET
CRD502	50	2	**5480**	5489.94	10.36	1.00	5663	5777.88	40.01	1.00	**5480**	5510.02	25.06	1.00	**5480**	5486.36	8.84	1.00	**5480**	**5483.66**	3.25	1.00
CRD503	50	2	**5079**	**5079.00**	0.00	1.00	**5079**	**5079.00**	0.00	1.00	**5079**	**5079.00**	0.00	1.00	**5079**	**5079.00**	0.00	1.00	**5079**	**5079.00**	0.00	1.00
CRD702	70	2	6777	6864.42	44.17	1.00	7186	7466.98	128.42	1.00	6772	6918.44	79.39	1.00	6771	6824.96	42.81	1.00	**6763**	**6798.50**	33.32	1.00
CRD704	70	2	6380	6443.38	22.97	1.00	6620	6620.00	0.00	1.00	6467	6625.52	100.79	1.00	6400	6460.20	40.02	1.00	**6370**	**6430.10**	35.19	1.00
CRD706	70	2	**6671**	**6673.60**	5.31	1.00	6994	7222.00	70.26	1.00	6745	6872.70	81.87	1.00	**6671**	6713.14	39.15	1.00	**6671**	6707.20	32.37	1.00
CRD707	70	2	6470	6512.48	33.79	1.00	6789	7296.76	159.68	1.00	6489	6677.12	91.85	1.00	**6467**	6520.48	33.42	1.00	**6467**	**6492.10**	30.77	1.00
CRD100	100	2	7096	7142.84	28.33	10.00	7080	7183.98	172.73	10.00	7080	7202.30	74.37	10.00	**7044**	7079.20	38.08	10.00	**7044**	**7075.16**	33.69	10.00
CRD101	100	2	7723	7800.38	50.70	10.00	7737	7905.34	203.85	10.00	7726	7836.66	76.71	10.00	**7697**	7734.64	32.52	10.00	**7697**	**7714.40**	24.93	10.00
CRD102	100	2	7623	7674.50	24.30	10.00	7589	7671.18	116.45	10.00	7575	7703.12	71.16	10.00	**7535**	7610.60	36.68	10.00	**7535**	**7588.90**	22.14	10.00
CRD103	100	2	7703	7754.30	25.22	10.00	**7660**	7775.82	104.29	10.00	7690	7841.20	97.70	10.00	**7660**	7702.26	23.48	10.00	**7660**	**7677.38**	24.19	10.00
CRD104	100	2	7571	7645.06	36.03	10.00	**7541**	7661.42	190.69	10.00	7544	7713.00	101.22	10.00	**7541**	7562.08	24.71	10.00	**7541**	**7552.54**	23.80	10.00
CRD105	100	2	7123	7169.10	30.34	10.00	7077	7160.16	192.03	10.00	7080	7230.58	130.54	10.00	7077	7111.16	34.94	10.00	**7075**	**7095.00**	19.54	10.00
CRD107	100	2	7721	7734.42	8.94	10.00	7728	7833.00	184.36	10.00	7756	7866.88	83.13	10.00	**7720**	7733.74	21.42	10.00	**7720**	**7729.78**	17.18	10.00
CRD108	100	2	7304	7534.84	129.73	10.00	7154	7245.68	220.79	10.00	7153	7256.32	78.55	10.00	**7137**	7165.08	33.18	10.00	**7137**	**7155.58**	16.77	10.00
CRD109	100	2	7771	8069.40	113.81	10.00	**7286**	7409.02	327.97	10.00	7321	7455.92	87.46	10.00	**7286**	7305.00	37.65	10.00	**7286**	**7297.64**	22.82	10.00
SYM502	50	2	**2116**	2135.60	9.08	1.00	2273	2415.42	62.14	1.00	**2116**	2142.30	15.30	1.00	**2116**	**2124.38**	8.74	1.00	**2116**	2131.14	11.15	1.00
SYM503	50	2	**1965**	1971.90	10.54	1.00	2162	2384.82	92.97	1.00	**1965**	1974.40	15.22	1.00	**1965**	**1965.92**	4.51	1.00	**1965**	1968.40	8.45	1.00
SYM700	70	2	2037	2093.92	26.42	1.00	2556	2805.08	121.09	1.00	2068	2178.28	58.70	1.00	**2012**	**2059.06**	22.18	1.00	**2012**	2069.64	22.71	1.00
SYM701	70	2	1940	1989.04	24.53	1.00	2487	2692.42	88.25	1.00	1998	2100.14	70.99	1.00	**1931**	**1944.28**	13.11	1.00	**1931**	1959.18	26.60	1.00
SYM702	70	2	**1864**	**1899.60**	12.16	1.00	2092	2386.00	72.10	1.00	1938	2043.18	68.38	1.00	**1864**	1910.64	27.82	1.00	1870	1926.34	32.59	1.00
SYM703	70	2	1503	**1510.52**	4.00	1.00	1674	1928.44	86.81	1.00	1522	1625.12	42.13	1.00	**1497**	1514.28	11.83	1.00	1503	1527.40	20.47	1.00
SYM704	70	2	2032	2085.92	19.89	1.00	2470	2728.02	96.45	1.00	2080	2200.48	68.42	1.00	**2017**	**2050.94**	16.13	1.00	**2017**	2066.22	25.24	1.00
SYM705	70	2	2157	2187.06	13.97	1.00	2507	2657.34	81.20	1.00	2227	2338.60	64.68	1.00	**2152**	**2185.22**	26.45	1.00	**2152**	2196.36	30.52	1.00
SYM706	70	2	1495	1509.58	10.71	1.00	1656	1897.04	90.66	1.00	1510	1623.34	54.85	1.00	**1489**	**1507.66**	12.77	1.00	**1489**	1517.72	16.96	1.00
SYM707	70	2	2384	2412.46	16.78	1.00	2666	2998.80	110.27	1.00	2447	2579.76	56.51	1.00	**2371**	**2408.46**	22.08	1.00	2388	2419.88	22.76	1.00
SYM708	70	2	1823	1840.94	8.27	1.00	2136	2319.34	93.89	1.00	1851	1939.48	53.68	1.00	**1816**	**1834.66**	10.79	1.00	**1816**	1843.26	17.14	1.00
SYM709	70	2	1749	1777.18	19.62	1.00	2103	2382.16	104.93	1.00	1799	1940.76	69.53	1.00	**1749**	**1769.36**	25.55	1.00	**1749**	1783.74	27.37	1.00

Table 2 Results of HES-EA (Raidl and Julstrom 2003), ABA (Bui et al. 2012) and $\mathcal{H}$SSGA for the CRD, SYM, STR and SHRD instances

Instance	$\|V\|$	d	HES-EA				ABA				(SSGA)+2ER				(SSGA)+1ER				$\mathcal{H}$SSGA			
			Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET
CRD501	50	2	**5553**	5605.58	38.36	1.00	5706	6158.14	148.32	1.00	**5553**	5561.46	16.86	1.00	**5553**	5566.64	17.23	1.00	**5553**	**5553.66**	4.62	1.00
CRD501	50	3	**5126**	**5126.00**	0.00	1.00	**5126**	**5126.00**	0.00	1.00	**5126**	5166.74	37.31	1.00	**5126**	**5126.00**	0.00	1.00	**5126**	**5126.00**	0.00	1.00
CRD700	70	2	6366	6436.60	34.86	1.00	6933	7050.88	32.50	1.00	6334	6525.76	121.36	1.00	6314	6422.98	47.96	1.00	**6308**	**6365.44**	38.86	1.00
CRD700	70	3	**5789**	**5789.00**	0.00	1.00	**5789**	**5789.00**	0.00	1.00	6047	6153.16	68.34	1.00	**5789**	**5789.00**	0.00	1.00	**5789**	**5789.00**	0.00	1.00
CRD100	100	3	**6196**	**6196.00**	0.00	10.00	**6196**	**6196.00**	0.00	10.00	6201	6263.82	40.75	10.00	**6196**	**6196.00**	0.00	10.00	**6196**	**6196.00**	0.00	10.00
SYM500	50	2	**1759**	1777.26	8.36	1.00	1863	2070.54	79.78	1.00	1762	1781.24	11.27	1.00	**1759**	**1768.54**	6.94	1.00	**1759**	1769.44	7.47	1.00
SYM500	50	3	**1156**	**1156.00**	0.00	1.00	**1156**	**1156.00**	0.00	1.00	**1156**	1160.84	9.78	1.00	**1156**	**1156.00**	0.00	1.00	**1156**	**1156.00**	0.00	1.00
SYM500	50	4	**1105**	**1105.00**	0.00	1.00	**1105**	**1105.00**	0.00	1.00	**1105**	1115.62	10.23	1.00	**1105**	**1105.00**	0.00	1.00	**1105**	**1105.00**	0.00	1.00
SYM500	50	5	**1098**	**1098.00**	0.00	1.00	**1098**	**1098.00**	0.00	1.00	**1098**	1118.90	18.25	1.00	**1098**	**1098.00**	0.00	1.00	**1098**	**1098.00**	0.00	1.00
SYM508	50	2	**1861**	1875.04	13.01	1.00	1985	2156.06	77.45	1.00	**1861**	1869.12	10.33	1.00	**1861**	**1864.30**	2.37	1.00	**1861**	1866.04	5.12	1.00
SYM701	70	3	**1270**	1271.12	1.80	1.00	**1270**	1295.30	12.10	1.00	1331	1394.86	34.62	1.00	**1270**	**1270.00**	0.00	1.00	**1270**	**1270.00**	0.00	1.00
SYM701	70	4	**1198**	**1198.00**	0.00	1.00	**1198**	1199.34	2.22	1.00	1257	1330.04	34.51	1.00	**1198**	**1198.00**	0.00	1.00	**1198**	**1198.00**	0.00	1.00
SYM701	70	5	**1186**	**1186.00**	0.00	1.00	**1186**	**1186.00**	0.00	1.00	1233	1348.08	59.02	1.00	**1186**	**1186.00**	0.00	1.00	**1186**	**1186.00**	0.00	1.00
SYM708	70	2	1823	1841.42	8.37	1.00	2128	2313.70	92.30	1.00	1869	1957.04	57.79	1.00	**1816**	**1837.30**	11.41	1.00	1822	1846.20	15.92	1.00
STR500	50	2	**4420**	4422.28	2.88	1.00	4430	4472.96	12.01	1.00	**4420**	4422.44	7.83	1.00	**4420**	4425.28	7.25	1.00	**4420**	**4420.00**	0.00	1.00
STR500	50	3	**4118**	**4118.00**	0.00	1.00	**4118**	**4118.00**	0.00	1.00	**4118**	4127.42	5.44	1.00	**4118**	**4118.00**	0.00	1.00	**4118**	**4118.00**	0.00	1.00
STR500	50	4	**3956**	**3956.00**	0.00	1.00	**3956**	**3956.00**	0.00	1.00	3967	3985.34	9.87	1.00	**3956**	**3956.00**	0.00	1.00	**3956**	**3956.00**	0.00	1.00
STR500	50	5	**3807**	**3807.00**	0.00	1.00	**3807**	**3807.00**	0.00	1.00	3828	3849.44	13.04	1.00	**3807**	**3807.00**	0.00	1.00	**3807**	**3807.00**	0.00	1.00
STR502	50	2	**6126**	**6127.56**	2.09	1.00	6159	6181.10	10.81	1.00	**6126**	6132.38	4.75	1.00	**6126**	6132.40	4.59	1.00	**6126**	6127.92	2.31	1.00
STR700	70	2	4700	4713.74	6.28	1.00	4744	4803.56	18.01	1.00	**4693**	4718.94	11.90	1.00	4701	4732.76	13.64	1.00	**4693**	**4704.48**	6.32	1.00
STR700	70	3	**4397**	**4397.00**	0.00	1.00	**4397**	**4397.00**	0.00	1.00	4465	4501.38	15.99	1.00	**4397**	**4397.00**	0.00	1.00	**4397**	**4397.00**	0.00	1.00
STR700	70	4	**4245**	**4245.00**	0.00	1.00	**4245**	**4245.00**	0.00	1.00	4341	4390.80	23.83	1.00	**4245**	**4245.00**	0.00	1.00	**4245**	**4245.00**	0.00	1.00
STR700	70	5	**4100**	**4100.00**	0.00	1.00	**4100**	**4100.00**	0.00	1.00	4203	4280.36	38.00	1.00	**4100**	**4100.00**	0.00	1.00	**4100**	**4100.00**	0.00	1.00
STR1000	100	2	**5000**	5021.10	8.91	10.00	5004	5030.96	31.67	10.00	**5000**	5024.64	17.35	10.00	**5000**	5029.34	24.08	10.00	**5000**	**5004.50**	6.65	10.00
STR1000	100	3	**4702**	4702.08	0.27	10.00	**4702**	4702.70	0.45	10.00	4708	4719.64	7.53	10.00	**4702**	4702.02	0.14	10.00	**4702**	**4702.00**	0.00	10.00
STR1000	100	4	**4546**	**4546.00**	0.00	10.00	**4546**	**4546.00**	0.00	10.00	4569	4597.60	14.01	10.00	**4546**	**4546.00**	0.00	10.00	**4546**	**4546.00**	0.00	10.00

Table 2 continued

Instance	$\|V\|$	d	HES-EA				ABA				(SSGA)+2ER				(SSGA)+1ER				$\mathcal{H}$SSGA			
			Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET
STR1000	100	5	**4403**	**4403.00**	0.00	10.00	**4403**	**4403.00**	0.00	10.00	4434	4466.50	16.25	10.00	**4403**	**4403.00**	0.00	10.00	**4403**	**4403.00**	0.00	10.00
STR1001	100	2	**5000**	5022.56	7.56	10.00	5006	5027.92	29.65	10.00	**5000**	5023.62	21.19	10.00	**5000**	5029.10	23.96	10.00	**5000**	**5004.80**	7.52	10.00
STR1002	100	2	7094	7103.70	7.56	10.00	7094	7127.42	30.6823	10.00	7094	7120.20	16.97	10.00	**7089**	7094.78	4.30	10.00	**7089**	**7091.96**	3.38	10.00
STR1003	100	2	7093	7104.24	8.55	10.00	7105	7129.18	31.74	10.00	7093	7119.52	15.28	10.00	**7089**	7095.06	5.03	10.00	**7089**	**7091.98**	2.75	10.00
STR1004	100	2	8705	8730.94	11.13	10.00	8732	8760.24	22.1617	10.00	8701	8744.08	20.24	10.00	**8699**	8705.32	4.31	10.00	**8699**	**8703.00**	2.99	10.00
STR1005	100	2	8703	8731.30	10.51	10.00	8727	8761.96	24.60	10.00	8702	8740.44	17.61	10.00	**8699**	8706.06	4.42	10.00	**8699**	**8703.64**	4.66	10.00
STR1006	100	2	10543	10555.96	8.59	10.00	10532	10559.18	26.3034	10.00	10547	10570.58	12.60	10.00	**10529**	10539.84	5.32	10.00	**10529**	**10535.50**	4.17	10.00
STR1007	100	2	10532	10555.20	8.49	10.00	10536	10558.10	26.3668	10.00	10536	10569.34	15.80	10.00	**10529**	10539.96	5.46	10.00	**10529**	**10534.76**	4.16	10.00
STR1008	100	2	12446	12474.90	12.24	10.00	12485	12534.92	27.6412	10.00	12439	12475.96	19.68	10.00	12428	12449.80	12.01	10.00	**12427**	**12433.52**	5.58	10.00
STR1009	100	2	12412	12431.14	9.23	10.00	12440	12480.56	29.15	10.00	12420	12457.86	21.40	10.00	**12401**	12414.74	6.33	10.00	**12401**	**12412.42**	5.98	10.00
SHRD159	15	2	**904**	**904.00**	0.00	1.00	**904**	904.90	1.0817	1.00	**904**	904.04	0.28	1.00	**904**	**904.00**	0.00	1.00	**904**	**904.00**	0.00	1.00
SHRD159	15	3	**597**	**597.00**	0.00	1.00	**597**	**597.00**	0.00	1.00	**597**	**597.00**	0.00	1.00	**597**	**597.00**	0.00	1.00	**597**	**597.00**	0.00	1.00
SHRD159	15	4	**430**	**430.00**	0.00	1.00	**430**	**430.00**	0.00	1.00	**430**	**430.00**	0.00	1.00	**430**	**430.00**	0.00	1.00	**430**	**430.00**	0.00	1.00
SHRD159	15	5	**332**	**332.00**	0.00	1.00	**332**	**332.00**	0.00	1.00	**332**	**332.00**	0.00	1.00	**332**	**332.00**	0.00	1.00	**332**	**332.00**	0.00	1.00
SHRD200	20	2	**1679**	**1679.00**	0.00	1.00	**1679**	1679.10	0.30	1.00	**1679**	**1679.00**	0.00	1.00	**1679**	**1679.00**	0.00	1.00	**1679**	**1679.00**	0.00	1.00
SHRD200	20	3	**1088**	**1088.00**	0.00	1.00	**1088**	**1088.00**	0.00	1.00	**1088**	**1088.00**	0.00	1.00	**1088**	**1088.00**	0.00	1.00	**1088**	**1088.00**	0.00	1.00
SHRD200	20	4	**802**	**802.00**	0.00	1.00	**802**	**802.00**	0.00	1.00	**802**	**802.00**	0.00	1.00	**802**	**802.00**	0.00	1.00	**802**	**802.00**	0.00	1.00
SHRD200	20	5	**627**	627.04	0.20	1.00	**627**	**627.00**	0.00	1.00	**627**	**627.00**	0.00	1.00	**627**	**627.00**	0.00	1.00	**627**	**627.00**	0.00	1.00
SHRD259	25	2	**2714**	2717.80	9.90	1.00	**2714**	2717.02	2.11	1.00	**2714**	**2714.00**	0.00	1.00	**2714**	2714.00	0.00	1.00	**2714**	**2714.00**	0.00	1.00
SHRD259	25	3	**1756**	1756.06	0.42	1.00	**1756**	1756.62	1.26	1.00	**1756**	**1756.00**	0.00	1.00	**1756**	**1756.00**	0.00	1.00	**1756**	**1756.00**	0.00	1.00
SHRD259	25	4	**1292**	**1292.00**	0.00	1.00	**1292**	1292.06	0.23	1.00	**1292**	**1292.00**	0.00	1.00	**1292**	**1292.00**	0.00	1.00	**1292**	**1292.00**	0.00	1.00
SHRD259	25	5	**1016**	1016.02	0.14	1.00	**1016**	**1016.00**	0.00	1.00	**1016**	**1016.00**	0.00	1.00	**1016**	**1016.00**	0.00	1.00	**1016**	**1016.00**	0.00	1.00
SHRD300	30	2	**3992**	3995.98	6.27	1.00	3996	4002.84	3.18	1.00	**3992**	**3992.00**	0.00	1.00	**3992**	**3992.00**	0.00	1.00	**3992**	**3992.00**	0.00	1.00
SHRD300	30	3	**2592**	2592.22	0.41	1.00	**2592**	2594.34	1.36	1.00	**2592**	2592.20	0.40	1.00	**2592**	**2592.00**	0.00	1.00	**2592**	**2592.00**	0.00	1.00
SHRD300	30	4	**1905**	1905.18	0.74	1.00	**1905**	1907.42	1.60	1.00	**1905**	**1905.00**	0.00	1.00	**1905**	**1905.00**	0.00	1.00	**1905**	**1905.00**	0.00	1.00
SHRD300	30	5	**1504**	1504.26	0.44	1.00	**1504**	1506.20	1.46	1.00	**1504**	1504.04	0.28	1.00	**1504**	**1504.00**	0.00	1.00	**1504**	**1504.00**	0.00	1.00

larly, comparing with ABA, (SSGA)+2ER, in terms of *Best*, is better on 16, equals on 22 and is worse on 14 instances; (SSGA)+2ER, in terms of *Avg*, is better on 23, equals on 6 and is worse on 23 instances.

Table 3 reports the results of 28 non-Euclidean instances. Comparing with HES-EA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 5 and equal on 23 instances; $\mathcal{H}$SSGA, in terms of *Avg*, is better on 12 and equals on 16 instances. Comparing with HES-EA, (SSGA)+1ER, in terms of *Best*, is better on 4 and equals on 24 instances; (SSGA)+1ER, in terms of *Avg*, is better on 8, equals on 15 and worse on 5 instances. Similarly, comparing with HES-EA, (SSGA)+2ER, in terms of *Best*, is better on 3 and worse on 25 instances; (SSGA)+2ER, in terms of *Avg*, is better on 3 and worse on 25 instances. Comparing with ABA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 6 and equals on 22 instances; $\mathcal{H}$SSGA, in terms of *Avg*, is better on 16, equals on 11 and worse on 1 instances. Comparing with ABA, (SSGA)+1ER, in terms of *Best*, is better on 3, equals on 22 and worse on 3 instances; (SSGA)+1ER, in terms of *Avg*, is better on 10, equals on 10 and worse on 8 instances. Similarly, comparing with ABA, (SSGA)+2ER, in terms of *Best*, is worse on all 28 instances; (SSGA)+2ER, in terms of *Avg*, is worse on all 28 instances.

One can observe clearly from the results of Tables 1, 2 and 3 that $\mathcal{H}$SSGA and (SSGA)+1ER are superior to both HES-EA and ABA in terms of both *Best* and *Avg*.

4.2 Analyses on the two key components of $\mathcal{H}$SGGA

In this subsection, we carry out analyses on the two key components of $\mathcal{H}$SGGA: (i) effectiveness of *quality-and-distance* feature in the local search strategies (ii) effectiveness of RS+ in the replacement strategy. In addition, we also carry out statistical analyses about significant differences between $\mathcal{H}$SGGA vs HES-EA and $\mathcal{H}$SGGA vs ABA.

4.2.1 Effectiveness of quality-and-distance feature in the local search strategies

To analyze the effectiveness of *quality-and-distance* feature in the local search strategies (LS), we have performed experiments based on (i) $\mathcal{H}$SSGA that uses *quality-and-distance* feature in the LS and (ii) $\mathcal{H}$SSGA that does not use *quality-and-distance* feature (i.e., $\mathcal{H}$SSGA - {quality-and-distance}) in the LS for the dc-MST problem. In these experiments, we consider two instances (SYM702 and SYM704 for $d = 2$) selected randomly. The stopping criterion of $\mathcal{H}$SSGA and $\mathcal{H}$SSGA - *{quality-and-distance}* is set to 1 s for SYM702 and SYM704 instances. Figure 5a, b depicts the evolution of average solution quality (*Avg*) based on 50 runs) over average total execution time (*ATET*). Y-axis represents the *Average Solution Quality*, whereas X-axis represents the *Average Total Execution Time*. The curves in Fig. 5a, b clearly demonstrate that $\mathcal{H}$SSGA that uses *quality-and-distance* feature in the LS finds better solution qualities as well as converges faster than that of $\mathcal{H}$SSGA - {quality-and-distance}. Hence, such experiments justify the usefulness of *quality-and-distance* feature in the LS.

4.2.2 Effectiveness of RS+ in the replacement strategy

To analyze the effect of RS+, we have performed experiments based on (i) $\mathcal{H}$SSGA that uses RS+ and (ii) $\mathcal{H}$SSGA without RS+ (i.e., $\mathcal{H}$SSGA - {RS+}) for the dc-MST problem. In these experiments, we consider two instances (SYM702 and SYM704 for $d = 2$) selected randomly. Figure 6a, b exhibit the average solution quality (*Avg*) over 50 runs versus the number of generations on considered instances. In these experiments, $\mathcal{H}$SSGA and $\mathcal{H}$SSGA - {RS+} are allowed to execute over 50000 generations. X-axis represents the *Generation*, while Y-axis represents *Average Solution Quality*. The curves in Fig. 6a, b clearly demonstrate that $\mathcal{H}$SSGA that uses RS+ finds better solution qualities as well as converges faster than that of $\mathcal{H}$SSGA - {RS+}. Hence, such experiments justify the usefulness of RS+ in the replacement strategy.

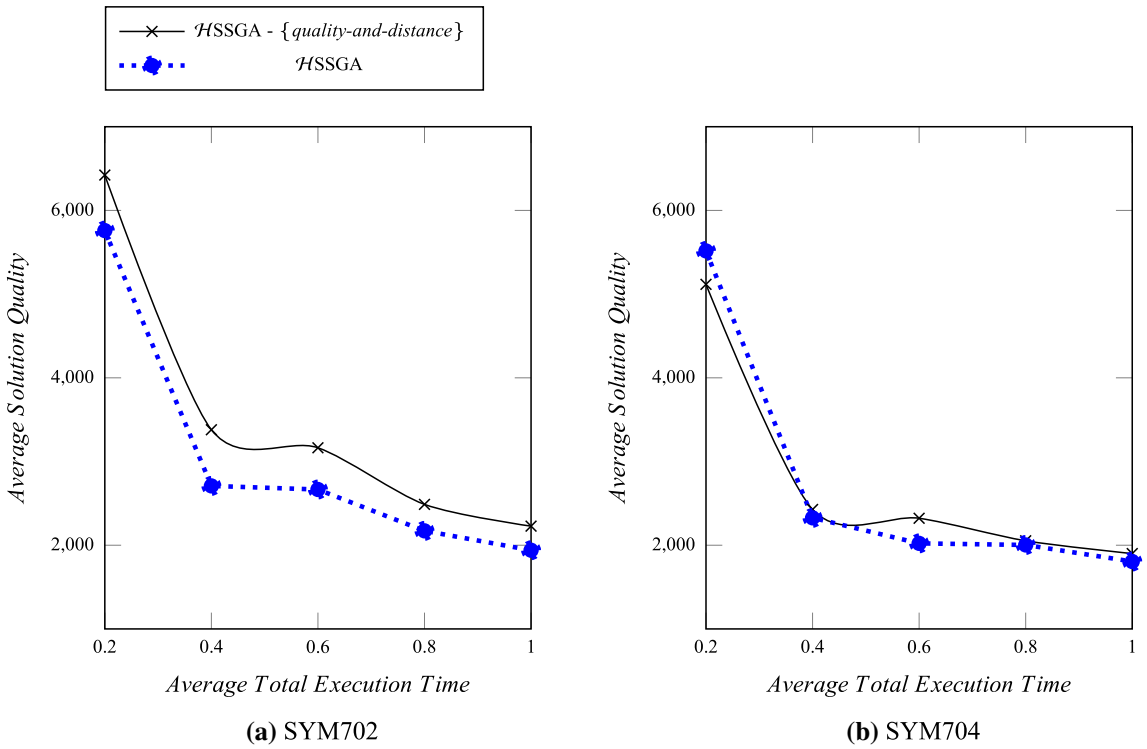
4.2.3 Statistical analysis

For statistical analysis, we perform nonparametric *Wilcoxon's signed-rank test* (García et al. 2009) on each group of instances (Euclidean and non-Euclidean) in order to compare $\mathcal{H}$SSGA with HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012) in terms of the best (*Best*) and average solution quality (*Avg*). We use *Wilcoxon's signed-rank test calculator* available at the link http://www.socscistatistics.com/tests/signedranks/Default2.aspx. For this statistical test, we first calculate the difference between the results obtained by each two compared approaches (HES-EA vs. $\mathcal{H}$SSGA and ABA vs. $\mathcal{H}$SSGA) on each group, and then, rank them according to its absolute value. For each group, R^+ is the sum of ranks in which the second approach outperforms the first, while R^- denotes the sum of ranks for the opposite case. If $min\{R^+; R^-\}$ is less than or equal to the critical value, then this test detects significant difference between the two compared approaches. The critical values are taken from the statistical table available at the link http://users.stat.ufl.edu/athienit/Tables/tables. Table 4 and 5 report the results of *Wilcoxon's signed-rank test* with a level of significance $\alpha = 0.05$ for the *Best* and *Avg* over 50 runs, respectively.

In Tables 4 and 5, column *Group* denotes the name of each groups; column *Comparison* denotes the name of two compared approaches; the next five columns *Sample Size*, *Critical Value*, R^+, R^- and *Significant*, respectively, denote the sample size, critical value, R^+, R^- and significant difference between the two compared approaches ("**yes**" if there exists

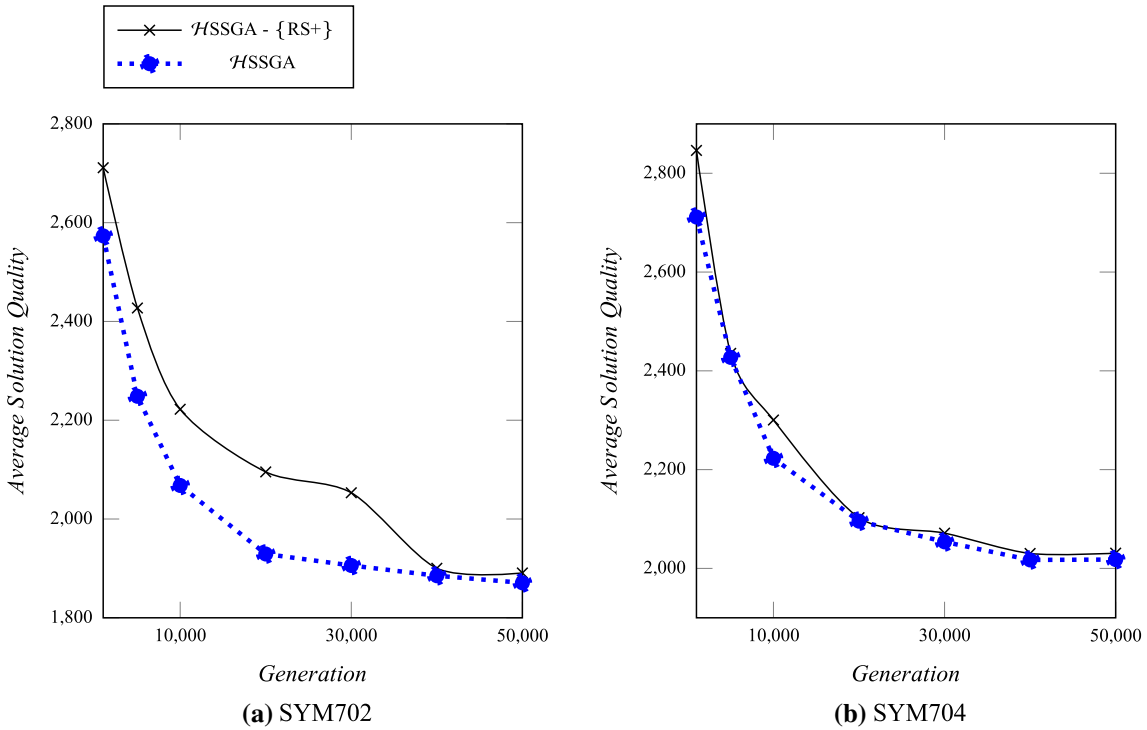
Table 3 Results of HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012) and $\mathcal{H}$SSGA for random-hard (R) and misleading-hard (M) instances

Instance	$\|V\|$	d	HES-EA				ABA				(SSGA)+2ER				(SSGA)+1ER				$\mathcal{H}$SSGA			
			Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET	Best	AVG	SD	ATET
R50n1	50	5	**4.04**	**4.04**	0.00	1.00	**4.04**	**4.04**	0.00	1.00	4.09	4.15	0.03	1.00	**4.04**	**4.04**	0.00	1.00	**4.04**	**4.04**	0.00	1.00
R50n2	50	4	**4.26**	**4.26**	0.00	1.00	**4.26**	**4.26**	0.00	1.00	4.31	4.37	0.03	1.00	**4.26**	**4.26**	0.00	1.00	**4.26**	**4.26**	0.00	1.00
R50n2	50	5	**3.94**	**3.94**	0.00	1.00	**3.94**	**3.94**	0.00	1.00	3.99	4.05	0.03	1.00	**3.94**	**3.94**	0.00	1.00	**3.94**	**3.94**	0.00	1.00
R50n3	50	5	**3.92**	3.93	0.00	1.00	**3.92**	**3.92**	0.00	1.00	3.97	4.03	0.04	1.00	**3.92**	**3.92**	0.00	1.00	**3.92**	**3.92**	0.00	1.00
R100n1	100	4	**8.06**	**8.06**	0.00	10.00	**8.06**	**8.06**	0.00	1.00	8.36	8.48	0.07	10.00	**8.06**	**8.06**	0.00	10.00	**8.06**	**8.06**	0.00	10.00
R100n1	100	5	**7.51**	**7.51**	0.00	10.00	**7.51**	**7.51**	0.00	10.00	7.82	7.92	0.07	10.00	**7.51**	**7.51**	0.00	10.00	**7.51**	**7.51**	0.00	10.00
R100n2	100	4	**8.06**	**8.06**	0.00	10.00	**8.06**	8.07	0.01	10.00	8.34	8.50	0.07	10.00	**8.06**	**8.06**	0.00	10.00	**8.06**	**8.06**	0.00	10.00
R100n2	100	5	**7.58**	**7.58**	0.00	10.00	**7.58**	**7.58**	0.00	10.00	7.87	7.99	0.06	10.00	**7.58**	**7.58**	0.00	10.00	**7.58**	**7.58**	0.00	10.00
R100n3	100	4	**8.02**	**8.02**	0.00	10.00	**8.02**	8.05	0.02	10.00	8.26	8.45	0.07	10.00	**8.02**	**8.02**	0.00	10.00	**8.02**	**8.02**	0.00	10.00
R100n3	100	5	**7.50**	**7.50**	0.00	10.00	**7.50**	**7.50**	0.00	10.00	7.78	7.89	0.07	10.00	**7.50**	**7.50**	0.00	10.00	**7.50**	**7.50**	0.00	10.00
R200n1	200	4	**16.09**	**16.09**	0.00	60.00	**16.09**	16.10	0.00	60.00	17.28	17.70	0.16	60.00	**16.09**	**16.09**	0.00	60.00	**16.09**	**16.09**	0.00	60.00
R200n1	200	5	**15.05**	15.06	0.00	60.00	**15.05**	15.06	0.00	60.00	16.27	16.71	0.20	60.00	**15.05**	**15.05**	0.00	60.00	**15.05**	**15.05**	0.00	60.00
R200n2	200	4	**15.69**	**15.69**	0.00	60.00	**15.69**	15.70	0.00	60.00	17.06	17.33	0.16	60.00	**15.69**	**15.69**	0.00	60.00	**15.69**	**15.69**	0.00	60.00
R200n2	200	5	**14.65**	14.66	0.00	60.00	**14.65**	14.66	0.00	60.00	15.89	16.30	0.19	60.00	**14.65**	**14.65**	0.00	60.00	**14.65**	14.66	0.00	60.00
R200n3	200	4	15.63	**15.63**	0.00	60.00	15.63	15.64	0.01	60.00	16.86	17.27	0.14	60.00	**15.62**	**15.63**	0.00	60.00	**15.62**	**15.63**	0.00	60.00
R200n3	200	5	**14.59**	**14.59**	0.00	60.00	**14.59**	**14.59**	0.00	60.00	15.95	16.24	0.14	60.00	**14.59**	**14.59**	0.00	60.00	**14.59**	**14.59**	0.00	60.00
m050n1	50	5	**6.60**	**6.60**	0.01	1.00	**6.60**	6.63	0.01	1.00	6.68	6.83	0.07	1.00	**6.60**	6.61	0.01	1.00	**6.60**	**6.60**	0.00	1.00
m050n2	50	5	**5.78**	5.85	0.17	1.00	**5.78**	5.79	0.01	1.00	5.81	5.91	0.09	1.00	**5.78**	5.85	0.19	1.00	**5.78**	**5.78**	0.00	1.00
m050n3	50	5	**5.50**	**5.50**	0.00	1.00	**5.50**	**5.50**	0.00	1.00	5.58	5.69	0.05	1.00	**5.50**	**5.50**	0.00	1.00	**5.50**	**5.50**	0.00	1.00
m100n1	100	5	**11.08**	11.26	0.28	10.00	**11.08**	11.23	0.31	10.00	11.47	11.70	0.18	10.00	**11.08**	11.52	0.43	10.00	**11.08**	**11.09**	0.00	10.00
m100n2	100	5	**11.33**	11.74	0.40	10.00	**11.33**	11.79	0.42	10.00	11.66	11.92	0.12	10.00	**11.33**	11.40	0.20	10.00	**11.33**	**11.35**	0.12	10.00
m100n3	100	5	**10.19**	10.43	0.37	10.00	10.20	10.52	0.36	10.00	10.45	10.74	0.11	10.00	**10.19**	10.66	0.45	10.00	**10.19**	**10.19**	0.00	10.00
m200n1	200	5	**18.33**	18.67	0.40	60.00	**18.33**	**18.42**	0.19	60.00	20.48	21.22	0.41	60.00	**18.33**	19.09	0.55	60.00	**18.33**	18.57	0.32	60.00
m200n2	200	5	19.18	20.87	0.96	60.00	19.19	19.45	0.33	60.00	20.62	21.30	0.36	60.00	19.18	20.57	0.75	60.00	**19.16**	**19.23**	0.24	60.00
m200n3	200	5	**16.13**	**16.13**	0.00	60.00	**16.13**	16.14	0.09	60.00	17.89	18.67	0.40	60.00	**16.13**	16.32	0.30	60.00	**16.13**	**16.13**	0.00	60.00
m300n1	300	5	44.53	46.44	0.81	200.00	40.71	40.85	0.26	200.00	43.48	43.99	0.31	200.00	41.29	42.53	0.70	200.00	**40.68**	**40.83**	0.26	200.00
m400n1	400	5	67.99	70.85	1.20	600.00	54.96	58.11	1.85	600.00	58.09	59.22	0.66	600.00	57.79	60.37	1.12	600.00	**54.64**	**55.01**	0.43	600.00
m500n1	500	5	97.47	100.03	1.36	1000.00	79.57	82.75	1.61	1000.00	83.50	85.79	0.90	1000.00	87.49	89.58	1.18	1000.00	**79.31**	**80.40**	0.57	1000.00



(a) SYM702 **(b)** SYM704

Fig. 5 Improvement of average solution quality over average total execution time



(a) SYM702 **(b)** SYM704

Fig. 6 Improvement of average solution quality over successive generations

Table 4 Results of statistical comparison for the best value (Best)

Group	Comparison	Best				
		Sample size	Critical value	R^+	R^-	Significant
Euclidean	HES-EA versus $\mathcal{H}$SSGA	31	147	479	17	Yes
	ABA versus $\mathcal{H}$SSGA	41	279	861	0	Yes
Non-Euclidean	HES-EA versus $\mathcal{H}$SSGA	5	Unknown	15	0	Not applicable
	ABA versus $\mathcal{H}$SSGA	7	2	28	0	Yes

Table 5 Results of statistical comparison for the average solution (avg) over 50 runs

Group	Comparison	Avg				
		Sample size	Critical value	R^+	R^-	Significant
Euclidean	HES-EA versus $\mathcal{H}$SSGA	46	361	925	156	Yes
	ABA versus $\mathcal{H}$SSGA	47	378	1128	0	Yes
Non-Euclidean	HES-EA versus $\mathcal{H}$SSGA	19	46	190	0	Yes
	ABA versus $\mathcal{H}$SSGA	26	98	337	14	Yes

Table 6 Overall comparison of $\mathcal{H}$SSGA with HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012)

Approaches	Best		Avg	
	HES-EA	ABA	HES-EA	ABA
Better	35	48	56	73
Equal	70	59	41	33
Worse	2	0	10	1

a significant difference between two compared approaches, otherwise "**no**") for each *Best* and *Avg* on each group. On comparison with HES-EA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 3 out of 4 statistical tests, and $\mathcal{H}$SSGA, in terms of *Avg* is better on 4 out of 4 statistical tests. The result of test HES-EA vs. $\mathcal{H}$SSGA for the *Best* is unknown because the sample size is not big enough to return a critical value at the level of significance $\alpha = 0.05$. On comparison with ABA, $\mathcal{H}$SSGA, in terms of *Best*, is better on 4 out of 4 statistical tests, and $\mathcal{H}$SSGA, in terms of *Avg* is better on 4 out of 4 statistical tests. This test discloses significant differences between HES-EA vs. $\mathcal{H}$SSGA and ABA vs. $\mathcal{H}$SSGA. Tables 4 and 5 clearly show that $\mathcal{H}$SSGA is superior to HES-EA and ABA.

4.3 Collective picture

This subsection presents a collective picture that describes an overall comparison of $\mathcal{H}$SSGA with HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012) on 107 benchmark instances.

Table 6 gives an overall comparison of $\mathcal{H}$SSGA with HES-EA (Raidl and Julstrom 2003) and ABA (Bui et al. 2012) on both best value (*Best*) and average solution quality (*Avg*). One can observe clearly from the results of Table 6 that $\mathcal{H}$SSGA is superior to HES-EA and ABA in terms of both *Best* and *Avg*.

5 Conclusion

In this paper, we have presented a hybrid approach ($\mathcal{H}$SSGA) combining a steady-state genetic algorithm and local search strategies for the degree-constrained minimum spanning tree (dc-MST) problem. $\mathcal{H}$SSGA is quite different from the hybrid approach (HES-EA) (Raidl and Julstrom 2003) which is the best one among all existing variants of genetic algorithm for dc-MST problem, particularly on three components—problem-specific crossover operator, local search strategies and an additional step in the replacement strategy. The way problem-specific crossover operator (*Xover*) inherits good edges of parent individuals in the newly generated child solution (say T^C) as much as possible and at the same time *Xover* maintains the degree constraint of all non-leaf vertices of T^C makes it quite different from the existing crossover operators designed for this problem. The role of local search strategies if applied on newly generated child solution is used to intensify the search around the generated child solution, whereas the role of an additional step (based on perturbation strategy at a regular interval of time) in the replacement strategy is used to maintain diversity in the current population throughout the search process. All components of $\mathcal{H}$SSGA effectively coordinate with each other and

help in making $\mathcal{H}$SSGA more effective and robust in finding high-quality solutions. On a set of 107 benchmark instances, computational results show that $\mathcal{H}$SSGA is overall superior to state-of-the-art metaheuristic techniques (HES-EA Raidl and Julstrom 2003 and ABA Bui et al. 2012). We have also performed experimental analyses that justify the usefulness of *quality-and-distance* feature in the local search strategies and RS+ in the replacement strategy in $\mathcal{H}$SSGA.

In future work, *quality-and-distance* feature in the local search strategies as well as RS+ in the replacement strategy in $\mathcal{H}$SSGA can be applied to develop variants of hybrid genetic algorithm for other $\mathcal{NP}$-hard spanning tree problems as well as other $\mathcal{NP}$-hard combinatorial optimization problems. Even *quality-and-distance* feature in the local search strategies in $\mathcal{H}$SSGA can be applied to develop other metaheuristic techniques for $\mathcal{NP}$-hard combinatorial optimization problems. The idea used in designing problem-specific crossover operator for the dc-MST problem can be also applied to develop variants of hybrid genetic algorithm for other $\mathcal{NP}$-hard spanning tree problems.

Acknowledgements This work is supported in part by a grant (Grant Number YSS/2015/000276) from the Science and Engineering Research Board—Department of Science & Technology, Government of India.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

Bau Y, Ho CK, Ewe HT (2005) An ant colony optimization approach to the degree-constrained minimum spanning tree problem. In: Proceedings of the computational intelligence and security, international conference (CIS 2005), Xi'an, China, December 15–19, 2005, Part I, pp 657–662

Beasley JE, PC C (1996) A genetic algorithm for the set covering problem. Eur J Oper Res 94:394–404

Binh HTT, Nguyen TB (2008) New particle swarm optimization algorithm for solving degree constrained minimum spanning tree problem. In: Proceedings of the PRICAI 2008: trends in artificial intelligence, 10th Pacific rim international conference on artificial intelligence, Hanoi, Vietnam, December 15–19, 2008, pp 1077–1085

Boldon B, Deo N, Kumar N (1996) Minimum-weight degree-constrained spanning tree problem: heuristics and implementation on an SIMD parallel machine. Parallel Comput 22(3):369–382

Bui TN, Zrncic CM (2006) An ant-based algorithm for finding degree-constrained minimum spanning tree. In: Proceedings of the genetic and evolutionary computation conference, GECCO 2006, Seattle, Washington, USA, July 8–12, 2006, pp 11–18

Bui TN, Deng X, Zrncic CM (2012) An improved ant-based algorithm for the degree-constrained minimum spanning tree problem. IEEE Trans Evol Comput 16(2):266–278

Cerrone C, Cerulli R, Raiconi A (2014) Relations, models and a memetic approach for three degree-dependent spanning tree problems. Eur J Oper Res 232(3):442–453

Cerrone C, Cerulli R, Gaudioso M (2016) OMEGA one multi ethnic genetic approach. Optim Lett 10(2):309–324

Davis L (1991) Handbook of genetic algorithms. Van Nostrand Reinhold, New York

Doan MN (2007) An effective ant-based algorithm for the degree-constrained minimum spanning tree problem. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2007), 25–28 September 2007, Singapore, pp 485–491

Ernst AT (2010) A hybrid Lagrangian particle swarm optimization algorithm for the degree-constrained minimum spanning tree problem. In: Proceedings of the IEEE congress on evolutionary computation, CEC 2010, Barcelona, Spain, 18–23 July 2010, pp 1–8

Gao X, Jia L, Kar S (2017) Degree-constrained minimum spanning tree problem of uncertain random network. J Ambient Intell Humaniz Comput 8(5):747–757

García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heuristics 15(6):617–644

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco

Gargano L, Hell P, Stacho L, Vaccaro U (2002) Spanning trees with bounded number of branch vertices. In: Proceedings of 29th international colloquium, ICALP 2002 Malaga, Spain, vol 2380, pp 355–365

Goldberg DE (1989) Genetic algorithms in search optimization and machine learning. Addison-Wesley, Reading

Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications in biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor

Iordache GV, Boboila MS, Pop F, Stratan C, Cristea V (2007) A decentralized strategy for genetic scheduling in heterogeneous environments. Multiagent Grid Syst 3(4):355–367

Knowles JD, Corne D (2000) A new evolutionary approach to the degree-constrained minimum spanning tree problem. IEEE Trans Evol Comput 4(2):125–134

Krishnamoorthy M, Ernst AT, Sharaiha YM (2001) Comparison of algorithms for the degree constrained minimum spanning tree. J Heuristics 7(6):587–611

Marín A (2015) Exact and heuristic solutions for the minimum number of branch vertices spanning tree problem. Eur J Oper Res 245(3):680–689

Moreno J, Frota Y, Martins S (2018) An exact and heuristic approach for the d-minimum branch vertices problem. Comput Opt Appl 71(3):829–855

Narula SC, Ho CA (1980) Degree-constrained minimum spanning tree. Comput OR 7(4):239–249

Pop F, Cristea V, Bessis N, Sotiriadis S (2013) Reputation guided genetic scheduling algorithm for independent tasks in inter-clouds environments. In: 27th International conference on advanced information networking and applications workshops, WAINA 2013, Barcelona, Spain, March 25–28, 2013, pp 772–776

Prim R (1957) Shortest connection networks and some generalizations. Bell Syst Tech J 36:1389–1401

Raidl GR, Julstrom BA (2000) A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In: Proceedings of the 2000 ACM symposium on applied computing, Villa Olmo, Via Cantoni 1, 22100 Como, Italy, March 19–21, 2000, vol 1, pp 440–445

Raidl GR, Julstrom BA (2003) Edge sets: an effective evolutionary coding of spanning trees. IEEE Trans Evol Comput 7:225–239

Ravi R, Marathe M, Ravi S, Rosenkrantz D, Hunt H III (1993) Many birds with one stone: multi-objective approximation algorithms. In: Proceedings of the 25th annual ACM STOCS, pp 438–447

Savelsbergh MWP, Volgenant T (1985) Edge exchanges in the degree-constrained minimum spanning tree problem. Comput OR 12(4):341–348

Silvestri S, Laporte G, Cerulli R (2017) A branch-and-cut algorithm for the minimum branch vertices spanning tree problem. Comput Oper Res 81:322–332

Sundar S (2014) A steady-state genetic algorithm for the dominating tree problem. In: Proceedings of the simulated evolution and learning—10th international conference, SEAL 2014, Dunedin, New Zealand, December 15–18, 2014, pp 48–57

Sundar S, Singh A (2012) New heuristics for two bounded-degree spanning tree problems. Inf Sci 195:226–240

Sundar S, Singh A (2015) Metaheuristic approaches for the blockmodel problem. IEEE Syst J 9(4):1237–1247

Sundar S, Singh A (2017) Two grouping-based metaheuristics for clique partitioning problem. Appl Intell 47(2):430–442

Zhou G, Gen M (1997) A note on genetic algorithms for degree-constrained spanning tree problems. Networks 30(2):91–95