Information Sciences 195 (2012) 226–240

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

New heuristics for two bounded-degree spanning tree problems

Shyam Sundar [a], Alok Singh [a,*], André Rossi [b]

[a] Department of Computer and Information Sciences, University of Hyderabad, Hyderabad 500 046, India
[b] Lab-STICC, Université de Bretagne-Sud, 56321 Lorient Cedex, France

ARTICLE INFO

Article history:
Received 8 December 2010
Received in revised form 28 December 2011
Accepted 29 January 2012
Available online 7 February 2012

Keywords:
Ant colony optimization
Branch vertex
Heuristic
Spanning tree
Swarm intelligence

ABSTRACT

A vertex v of a connected graph $G = (V,E)$ is called a branch vertex if its degree is greater than two. Pertaining to branch vertices, this paper studies two optimization problems having roots in the domain of optical networks. The first one, referred to as MBV, seeks a spanning tree T of G with the minimum number of branch vertices, whereas the second problem, referred to as MDS, seeks a spanning tree T of G with the minimum degree sum of branch vertices. Both MBV and MDS are $\mathcal{NP}$-Hard. Two heuristics approaches are presented for each problem. The first approach is a problem specific heuristic, whereas the latter one is a hybrid ant-colony optimization algorithm. Computational results show the effectiveness of our proposed approaches.

1. Introduction

Given a connected graph $G = (V,E)$, a vertex v of G is called a branch vertex if its degree is greater than two. Finding a spanning tree T of G with the minimum number of branch vertices (MBV) and finding a spanning tree T of G with the minimum degree sum of branch vertices (MDS) are among the several variants of the spanning tree problems which find applications in communication networks. This paper is concerned with MBV and MDS, both of which have roots in optical networks. We have used vertex and node interchangeably throughout this paper.

In an optical network, Wavelength Division Multiplexing (WDM) technology allows the transmission of multiple light beams of different wavelength concurrently through the same optical fiber cable. Multicasting in these networks is implemented using a light splitting switch that replicates the input optical signal on multiple paths. A *lightpath* is an optical path (data channel) that connects two nodes in the network and is created by the allocation of the same wavelength throughout the path [2]. A *light-tree* [12], an extension of the light path, is capable of optical multicasting, i.e., it transmits a signal from a single source node to a set of destination nodes in an optical WDM network. Multicasting is required for efficient implementations of many services such as worldwide web browsing, video conferencing and video on demand services. Zhang et al. [17] showed that multicasting can be supported at the WDM layer by letting WDM switches make copies of data packets in the optical domain via light splitting. Thus, *light-tree* enables all optical communications from a source node to a set of destination nodes. Switches with splitting ability are placed on the branch nodes. Each node with splitting ability is capable to transmit a number of copies of the optical signal equal to its neighbors, while other nodes only transmit a copy of it. However, a WDM optical network can have only a limited number of these sophisticated switches due to cost considerations. Therefore, the problem reduces to finding a spanning tree of the optical network with the minimum number of branch nodes, where switches can be placed so that all possible multicast connections can be ensured in the optical network.

* Corresponding author. Tel.: +91 4023134011.
E-mail addresses: mc08pc17@uohyd.ernet.in (S. Sundar), alokcs@uohyd.ernet.in (A. Singh), andre.rossi@univ-ubs.fr (A. Rossi).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2012.01.037

Gargano et al. [8] were the first to study the computational complexity of MBV. Let $s(G)$ be the smallest number of branch vertices in any spanning tree of G, then finding a spanning tree T of G with $s(G) = 0$ is same as finding a *Hamiltonian* path (a well known $\mathcal{NP}$-Complete problem) of G. A spanning tree T of G with $s(G) \leqslant 1$ is called a spanning spider. It is $\mathcal{NP}$-Complete to decide whether a graph G admits a spanning spider. More generally, the problem of deciding whether a given graph satisfies $s(G) \leqslant k$ is $\mathcal{NP}$-Complete problem, where k is a fixed non-negative integer. Therefore, unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial time algorithms for such type of problems. Therefore, any exact method is not practical even for moderately large instances. However, Gargano et al. [8] proposed a polynomial time algorithm for MBV on a class of graphs satisfying certain density conditions. But in real scenarios, it seldom happens that the network satisfies these density conditions. Therefore, such an algorithm cannot be used in real situations and heuristics are the only available options.

Cerulli et al. [1] proposed three heuristics for MBV. They also introduced and studied MDS in the same paper. Actually, many optical devices are capable of only duplicating an optical signal. Therefore, in order to replicate the optical signal on each edge, the number of devices to be placed on a branch node depends on the number of edges incident to it, i.e., one has to place $C[v] - 2$ different devices on a branch node v, where $C[v]$ is the degree of v in the spanning tree. Therefore, in order to minimize the number of devices, one has to minimize the degree sum of branch vertices. This was the motivation behind introducing MDS by Cerulli et al. [1]. Let $q(G)$ be the minimum degree sum of branch vertices of an optimal solution of MDS. Cerulli et al. [1] proved that if $\mathcal{P} \neq \mathcal{NP}$, then there is no polynomial time algorithm to check whether $q(G) \leqslant k$, where k is a fixed positive number. They also showed that MBV and MDS are related, but not the same. The optimal solution value $q(G)$ of MDS cannot always be directly derived from the optimal solution value $s(G)$ of MBV. They also proposed three heuristics for MDS.

In this paper, we propose a heuristic and a hybrid ant colony optimization approach to solve MBV and also a heuristic and a hybrid ant colony optimization approach to solve MDS. A special feature of our ant colony optimization approaches is the use of two pheromones (one for vertices and another for edges). We have compared our approaches with those presented in [1]. Computational results demonstrate the effectiveness of our approaches.

The remainder of this paper is organized as follows: Section 2 describes our heuristic approach for MBV, whereas Section 3 describes the ant colony optimization (ACO) approach for MBV. Sections 4 and 5 respectively describe the heuristic and ACO approach for MDS. Computational results are reported in Section 6. Finally, Section 7 contains some concluding remarks.

2. Heuristic for MBV

Our proposed heuristic for MBV (referred to as Heu_MBV) consists of two phases. It begins by selecting a vertex, say v_1, randomly and then it proceeds as follows:

Phase 1: It tries to repeatedly add an edge of G that can connect the last selected vertex to one of its adjacent unselected vertices of minimum degree into the partially constructed tree. If there are more than one such edges then ties are broken arbitrarily. If there exists no such edge and if local degree of v_1 in the partially constructed tree is still one, then again it tries to find an edge using v_1 in place of last selected vertex. Note that this is a one time measure, because once this measure is used the degree of v_1 will become two. This procedure is repeated again and again till it fails to find an edge. Upon failure we enter phase 2.

Phase 2: If the set of unselected vertices is nonempty, then a branch vertex needs to be created from among the selected non-branch vertices. Among all the selected non-branch vertices, a vertex having maximum number of adjacent unselected vertices (ties are broken arbitrarily) is chosen to become a branch vertex. All edges connecting this newly created branch vertex to its set S of unselected adjacent vertices are added into the partially constructed tree. All vertices in S are now sorted according to non-decreasing order of their degrees in G, and then, *phase* 1 is applied (without going into *phase* 2 upon failure) by setting each vertex in S as last selected vertex one-by-one. Once *phase* 1 has been applied to all vertices in S, then *phase* 2 restarts. This procedure is repeated again and again till the set of unselected vertices becomes empty.

Here, it is to be noted that by a selected vertex, we mean a vertex one of whose incident edge has been included in the partially constructed tree, and, by an unselected vertex, we mean a vertex none of whose incident edges have been included in the partially constructed tree. Also, note that if the spanning tree T of G is constructed without using phase 2, then this spanning tree T of G has no branch vertex and is actually a *Hamiltonian* path of G. In *phase* 1, instead of selecting a vertex randomly, we always select a vertex of minimum degree. This is based on the fact that such a selection will direct the search process to a spanning tree with longer diameter, which is expected to have lesser number of branch vertices. Same is the motivation behind ordering the vertices in S according to non-decreasing order of their degrees in *phase* 2. Similarly, in *phase* 2, a selected non-branch vertex with maximum number of adjacent unselected vertices is always chosen to become a branch vertex. Such a choice decreases the number of unselected vertices by maximum amount and gives *phase* 1 the maximum number of starting vertices to explore from. Both of these factors help in reducing the number of restarts of *phase* 2, thereby reducing the number of branch vertices. The idea of connecting all unselected vertices adjacent to newly created branch vertex is based on commonsense. MBV heuristics of Cerulli et al. [1] also use this idea.

Algorithm 1. Heuristic for MBV

```
Algorithm 1: Heuristic for MBV
input  : A connected graph $G = (V, E)$
output: A spanning tree $T$
begin
 1   Initialize $T \leftarrow \emptyset$;
 2   for each vertex $i \in V$ do
 3       $C[i] \leftarrow 0$;
 4   Select a starting vertex $r \in V$ randomly;
 5   $v_1 \leftarrow r$; $us \leftarrow |V| - 1$;
 6   Find an unselected vertex $v_2$ of minimum degree adjacent to $v_1$;
 7   if ($v_2$ exists) then
 8       $T \leftarrow T \cup \{v_1, v_2\}$; $C[v_1]{+}{+}$; $C[v_2]{+}{+}$; $v_1 \leftarrow v_2$, $us{-}{-}$;
 9       goto step 6;
10   else
11       Find an unselected vertex $v_2$ of minimum degree adjacent to
         vertex $r$;
12       if ($v_2$ exists) then
13           $T \leftarrow T \cup \{r, v_2\}$; $C[r]{+}{+}$; $C[v_2]{+}{+}$; $r \leftarrow v_2$; $us{-}{-}$;
14           goto step 11;
15   while ($us \neq 0$) do
16       Select a vertex $v_3$ as a branch vertex among all selected vertices
         having maximum unselected vertices;
17       $S \leftarrow \emptyset$;
18       for each unselected vertex $i$ adjacent to $v_3$ do
19           $S \leftarrow S \cup i$;
20       for each vertex $i \in S$ do
21           $T \leftarrow T \cup \{v_3, i\}$; $C[v_3]{+}{+}$; $C[i]{+}{+}$; $us{-}{-}$;
22       sort all vertices $\in S$ into non_decreasing order based on their
         degrees in $G$ ;
23       for each vertex $i \in S$ do
24           $v_1 \leftarrow i$;
25           Find an unselected vertex $v_2$ of minimum degree adjacent to
             $v_1$;
26           if ($v_2$ exists) then
27               $T \leftarrow T \cup \{v_1, v_2\}$; $C[v_1]{+}{+}$; $C[v_2]{+}{+}$; $v_1 \leftarrow v_2$; $us{-}{-}$;
28               goto step 25;
end
```

Algorithm 1 gives the pseudo-code of aforementioned heuristic for MBV, where for each vertex *i*, *C*[*i*] represents the local degree of *i* in the tree. *C*[*i*] should not be confused with the degree *deg*[*i*] of *i* in *G*.

Figs. 1 and 2 explain the heuristic with the help of an example. The input graph containing 20 vertices and 28 edges is shown in Fig. 1 and the spanning tree obtained through MBV heuristic is given in Fig. 2a where branch vertices are shown in light shade and edges are named according to the order in which they are added to the spanning tree, i.e., edge e_i is the *i*th edge added to the tree. Our MBV heuristic begins by selecting a vertex randomly. Let us assume it begins by selecting vertex 15. Vertices 3, 4, 5, 7 are adjacent to vertex 15 in the graph. Out of these adjacent vertices, vertex 7 has minimum degree 2 so edge (15,7) is the first edge that is added to the tree (this is shown by e_1 in the Fig. 2a). Now, vertex 7 is the vertex that is selected last so we have to look into its adjacent unselected vertices. Vertex 4 is only such vertex, and therefore, edge (7,4) is added to the tree (this is shown by e_2 in Fig. 2a). Continuing in this way edges (4,6), (6,18), (18,11), (11,2) are added to the

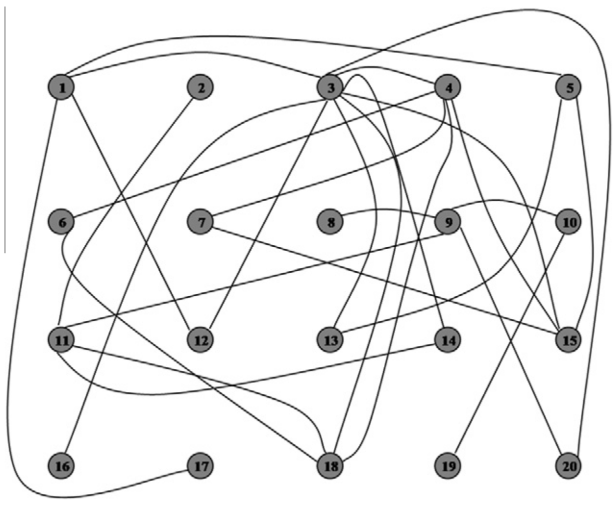

Fig. 1. Input graph.

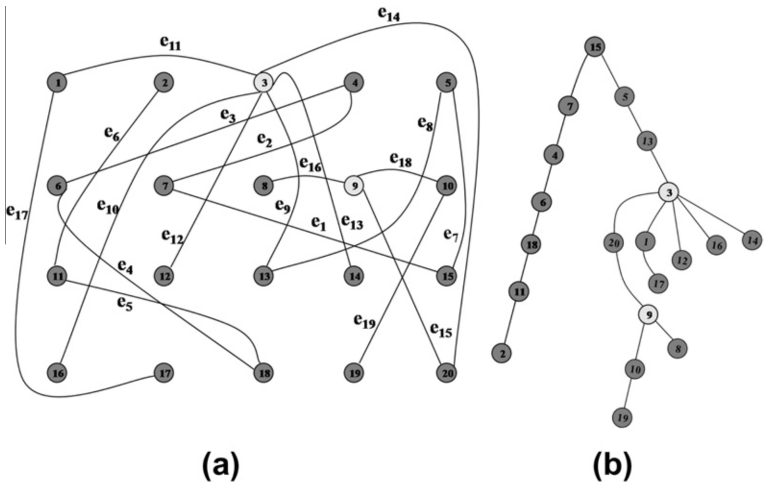

Fig. 2. Output of MBV heuristic.

tree. Now, 2 does not have any unselected adjacent vertices so heuristic restarts from first selected vertex, i.e., vertex 15. Now, vertex 5 is the lowest degree unselected adjacent vertex, so edge (15,5) is added to the tree. Then edges (5,13), (13,3), (3,16) are added into the tree. At this stage, we cannot proceed further using *phase* 1 of the heuristic, so we enter *phase* 2. Among all the selected vertices, vertex 3 has maximum number of unselected adjacent vertices, so vertex 3 is made a branch vertex and edges (3,1), (3,12), (3,14), (3,20) are added to the tree. Newly selected vertices 1, 12, 14 and 20 (set *S*) are sorted according to non-decreasing order of their degrees in *G* leading to order 12, 14, 20, 1. Now, *phase* 1 of the heuristic is restarted by setting vertex 12 as the last selected vertex, but it cannot go further because there is no unselected vertex adjacent to vertex 12. Same happens with vertex 14. Now, 20 becomes the last selected vertex and *phase* 1 adds edges (20,9) and (9,8) to the tree. Once again, *phase* 1 starts with vertex 1 as the last selected vertex and only one edge (1,17) is added to the tree. At this juncture, all vertices in *S* have been explored using *phase* 1, and therefore, *phase* 2 restarts again. This time vertex 9 is selected as a branch vertex and edge (9,10) is added to the tree. Again, we switch to *phase* 1 with vertex 10 as the last selected vertex. Now, *phase* 1 adds edge (10,19) to the tree and *phase* 2 restarts. Since there is no unselected vertex now, *phase* 2 ends and heuristic stops. So, we get a spanning tree with two branch vertices, which also happens to be the optimal number of branch vertices for this particular example. For the sake of clarity, the spanning tree of Fig. 2a is re-drawn in Fig. 2b.

3. Ant colony optimization

Ant colony optimization (ACO) is a nature-inspired metaheuristic that has been successfully applied to solve many $\mathcal{NP}$-Hard optimization problems. It is inspired by cooperative foraging behavior of real ant colonies. While walking, ants lay down on the ground a chemical substance called pheromone, forming in this way, a pheromone trail which is used for stigmergetic communication. Other ants can smell this pheromone, and its presence influences the choice of their paths, i.e., ants choose probabilistically the paths marked by strong pheromone concentrations. While searching for a path from nest to food source, ants tracing the shortest path will return sooner, hence immediately after their return, concentration of pheromone will be more on this path influencing other ants to follow this path. After some time, this will result in almost whole colony of ants following the same path. Thus pheromone trail allows the ants to find the shortest paths between their nest and food source. This feature of real ant colonies is exploited in artificial ant colony optimization algorithms to solve optimization problems.

In ACO algorithms, artificial ants can be regarded as stochastic constructive heuristics that construct better and better solutions to an optimization problem by using and updating pheromone trails. Since first ACO algorithm called Ant System was developed by Dorigo et al. [5,6], many diverse variants of the basic algorithm have been reported in the literature. Dorigo and Stützle [7] provides an excellent introduction to ant colony optimization along with a detailed survey of different variants of ACO algorithms and their applications. Some recent applications of ant colony optimization can be found in [3,9–11,16]. This paper is particularly concerned with one ACO variant called $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MMAS}$) which is proposed by Stützle and Hoos [14]. $\mathcal{MMAS}$ is one of the most successful ACO variants. The essential features of $\mathcal{MMAS}$ are following: (1) pheromone trail values are restricted to upper $[\tau_{max}]$ and lower $[\tau_{min}]$ trail limits to avoid stagnation, (2) pheromone trails values are initialized to τ_{max} to achieve better exploration of the search space at the start of the algorithm and (3) to exploit the best solutions, pheromone trails are updated by only one ant. This ant may be the one which built the iteration best solution and/or the global best solution.

3.1. ACO for MBV

It can be observed from the computational results (Section 6) that merely giving a heuristic approach to solve MBV is not enough to find even good solutions. To overcome this limitation, we attack MBV with an ant colony optimization (ACO) algorithm which is referred to as ACO_MBV subsequently in this paper. It is in essence a $\mathcal{MMAS}$. As MBV seeks a spanning tree with the minimum number of branch vertices, therefore, while solving this problem two factors will play crucial roles in finding good solutions. First one which edge should be selected and second one which vertex should be selected as a branch vertex, if there exists any, while constructing a spanning tree. Our ACO_MBV takes care of these two factors by incorporating them in the search process. For this, our ACO_MBV is empowered through the use of two different pheromone laying procedures, i.e., one for the edges and other for the vertices of the graph. The motivation behind using two pheromones is that pheromone laid on the edges helps in identifying good edges in constructing a spanning tree, whereas pheromone laid on the vertices helps in choosing promising vertices as branch vertices. We have incorporated iteration best pheromone trail update strategy as well as global best pheromone trail update strategy. We have coupled ACO_MBV with a local search to improve the solution quality further.

3.2. Construction of solution

In our ACO_MBV approach, we have incorporated major ideas of the heuristic of Section 2 in solution construction. During each iteration t, each ant k constructs a solution in a manner which is similar to the heuristic of Section 2 except the selection of an edge connecting the last selected vertex i to an unselected adjacent vertex and the selection of a branch vertex, which are done probabilistically with the help of pheromones and heuristic information. The probability $p^k_{e_{ij}}(t)$ of the selection of an edge e_{ij} is determined as follows:

$$p^k_{e_{ij}}(t) = \frac{[\tau_{e_{ij}}(t)]^{\alpha_e}[\eta_{e_{ij}}]^{\beta_e}}{\sum_{l\in N^k_i}[\tau_{e_{il}}(t)]^{\alpha_e}[\eta_{e_{il}}]^{\beta_e}}, \tag{1}$$

where $\eta_{e_{ij}} = \frac{1}{deg[j]}$ is a heuristic term that is available a priori and $\tau_{e_{ij}}$ is the pheromone trail on edge e_{ij}. α_e and β_e are two parameters which determine the relative influence of pheromone trail and the heuristic information, and N^k_i is the set of unselected vertices which are adjacent to the last selected vertex i.

Similarly, the probability $p^k_{v_i}(t)$ of the selection of a vertex v_i to become a branch vertex is determined as follows:

$$p^k_{v_i}(t) = \frac{[\tau_{v_i}(t)]^{\alpha_v}[\eta_{v_i}]^{\beta_v}}{\sum_{s\in S^k}[\tau_{v_s}(t)]^{\alpha_v}[\eta_{v_s}]^{\beta_v}}, \tag{2}$$

where η_{v_i} is a heuristic term that is set equal to the number of unselected vertices adjacent to v_i and τ_{v_i} is the pheromone trail on vertex v_i. α_v and β_v are two parameters which determine the relative influence of pheromone trail and the heuristic information, and S^k is the set of those non-branch selected vertices which are candidates to become a branch vertex.

3.3. Pheromone update

Pheromone update rule is used to augment the pheromone values on solution components that are present in high quality solutions and helps in guiding future ants towards better solutions. Once all ants have constructed their solutions, only the ant, which has constructed the iteration's best solution S^{ib}, is used to update the pheromone trails in the following way:

$$\tau_{e_{ij}}(t+1) = \rho\tau_{e_{ij}}(t) + \Delta\tau^{ib}_{e_{ij}}, \tag{3}$$

$$\Delta\tau^{ib}_{e_{ij}} = \begin{cases} P, & \text{if } e_{ij} \in S^{ib}, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

$$\tau_{v_i}(t+1) = \rho\tau_{v_i}(t) + \Delta\tau^{ib}_{v_i}, \tag{5}$$

$$\Delta\tau^{ib}_{v_i} = \begin{cases} P, & \text{if } v_i \text{ is a branch vertex in } S^{ib}, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

where ρ is persistence rate which is taken to be same for both edges and vertices of the graph $G(V,E)$, $\tau_{e_{ij}}(t)$ is the pheromone value of the edge $e_{i,j}$ at iteration $t, \tau_{v_i}(t)$ is the pheromone value of the vertex v_i at iteration t. $\Delta\tau^{ib}_{e_{ij}}$ and $\Delta\tau^{ib}_{v_i}$ are pheromone augmentation terms. P is a parameter to be determined empirically.

Moreover, we augment pheromone on the solution components of the global best solution (i.e., the overall best solution since the beginning of the ACO algorithm) S^{gb} once every GB_{it} iterations in the following way:

$$\tau_{e_{ij}} = \begin{cases} \tau_{e_{ij}} + Q, & \text{if } e_{ij} \in S^{gb} \text{ and either } i \text{ or } j \text{ is a branch vertex}, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

$$\tau_{v_i} = \begin{cases} \tau_{v_i} + Q, & \text{if } v_i \text{ is a branch vertex in } S^{gb}, \\ 0, & \text{otherwise}. \end{cases} \tag{8}$$

According to equation (7), a constant amount of pheromone Q is deposited on those edges of the global best solution S^{gb} whose at least one end point is a branch vertex, and according to Eq. (8), same amount of pheromone Q is deposited on the branch vertices of the global best solution S^{gb}. We have also tried with the strategy of augmenting the pheromone on every edge of the global best solution, but this strategy is always outperformed by the strategy of depositing pheromone on only those edges whose at least one end point is a branch vertex.

3.4. Local search

The literature on the variants of ACO algorithms to optimization problems suggests that a high quality solution is usually obtained by coupling a local search with a probabilistic solution construction by an ant. In our algorithm, we also couple a local search, which is a greedy approach, with ACO_MBV to obtain high quality solutions. This algorithm is referred to as ACO_MBV + LS. The description of ACO_MBV + LS is as follows: once each ant has constructed a solution as per the procedure described in Section 3.2, then a local search is applied to further improve the quality of solution. The pheromone trails are updated with this improved solution.

The purpose of this local search is to reduce the number of branch vertices as much as possible. Once an ant has constructed a solution, we have complete information about the local degree $C[i]$ for each vertex i in the spanning tree. The local search begins by sorting the branch vertices of the spanning tree according to non-decreasing order of their local degrees. Each branch vertex i with $C[i] = 3$ or $C[i] = 4$ or $C[i] = 5$ is of concern to the local search. The local search contains three separate cases to handle these three classes of branch vertices. All these cases begin by deleting an edge e_{ir} of the spanning tree, where i is a branch vertex, thereby, creating two components. Then an attempt is made to find an edge that can connect these components and is different from e_{ir} so that the degree of i can be reduced by one. Obviously, endpoints v_1 and v_2 of an edge (v_1, v_2) in G that can connect these two components must satisfy $C[v_1] < deg[v_1]$ and $C[v_2] < deg[v_2]$. This property is used to reduce the search space in all the three cases. Details of each of these cases are given below.

Case 1: This case tries to transform each branch vertex i with $C[i] = 3$ into a non-branch vertex. It consists of two procedures. If the first procedure (called *Procedure*(1_a)) is not successful in transforming a branch vertex i into a non-branch vertex, then we move to the second procedure (called *Procedure*(1_b)). In *Procedure*(1_a), an edge e_{ir} connecting the branch vertex i and the vertex r is deleted thereby creating the partition of the spanning tree into two components and making the solution infeasible. Let L_1 be the set of vertices belonging to the component containing i and L_2 be the set of vertices belonging to the component containing r. The values of $C[i]$ and $C[r]$ are updated to $C[i] - 1$ and $C[r] - 1$. To make this solution feasible again, an edge which can connect these two components and is different from e_{ir} is searched in G using the following rules:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $((C[v_1] = 1)$ *or* $(C[v_1] > 3))$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0)$ *or* $(C[v_2] > 3))$.
2. If first rule fails and if $(C[r] = 2)$, then find an edge connecting a vertex $v_1 \in L_1$ with $((C[v_1] = 1)$ *or* $(C[v_1] > 3))$ to the vertex $r \in L_2$.

If any one of these rules is satisfied, then an edge is found, which can connect these two components and one branch vertex, i.e., vertex i is transformed into a non-branch vertex and the next branch vertex is considered. If both of these rules fail, then the spanning tree is restored by adding the deleted edge e_{ir} into the tree. $C[i]$ and $C[r]$ get their previous values and the next edge incident to the branch vertex i is chosen for deletion and *Procedure*(1_a) is applied again.

If *Procedure*(1_a) is not successful even after trying all edges incident to i, i.e., vertex i is still a branch vertex, then we move to *Procedure*(1_b). Like *Procedure*(1_a), *Procedure*(1_b) deletes an edge e_{ir}. Then a search for another edge in G, which can make the solution feasible, is performed using the following rules:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $(C[v_2] = 3)$.
2. If first rule fails, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 3)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0)$ *or* $(C[v_2] = 3))$.

If the search is successful, then branch vertex i becomes a non-branch vertex and the next branch vertex is considered, otherwise the spanning tree is restored like *Procedure*(1_a) and the next edge incident to i is chosen for deletion. If *Procedure*(1_b) fails on all edges incident to i, then i cannot be transformed into a non-branch vertex and the next branch vertex is considered.

Case 2: This case tries to transform each branch vertex i with $C[i] = 4$ into a non-branch vertex. It consists of a single procedure called *Procedure*(2). Like *Procedure*(1_a), *Procedure*(2) deletes an edge e_{ir}. Then a search for another edge in G, which can make the solution feasible, is performed using the following rules:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $((C[v_1] = 1)$ *or* $(C[v_1] > 3))$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0)$ *or* $(C[v_2] > 3))$.
2. If first rule fails and if $(C[r] = 2)$, then find an edge connecting a vertex $v_1 \in L_1$ with $((C[v_1] = 1)$ *or* $(C[v_1] > 3))$ to the vertex $r \in L_2$.

If the search is successful, then the searched edge is added to the partial spanning tree of the solution, which makes $C[i] = 3$, and subsequently the procedures of Case 1 are called for vertex i, otherwise the spanning tree is restored, and the next edge of the branch vertex i is tried like *Procedure*(1_a). If *Procedure*(2) fails on all edges incident to i, then the degree of i cannot be reduced and the next branch vertex is considered.

Case 3: This case considers each branch vertex i with $C[i] = 5$ and tries to transform it into a non-branch vertex. It consists of a single procedure called *Procedure*(3). *Procedure*(3) also deletes an edge e_{ir} and then searches for another edge in G, which can make the solution feasible, using the following two rules:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $((C[v_1] = 1)$ *or* $(C[v_1] > 4))$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0)$ *or* $(C[v_2] > 4))$.
2. If first rule fails and if $(C[r] = 4)$, then find an edge connecting a vertex $v_1 \in L_1$ with $((C[v_1] = 1)$ *or* $(C[v_1] > 4))$ to the vertex $r \in L_2$.

If the search is successful, then the searched edge is added to the partial spanning tree of the solution, which makes $C[i] = 4$, and subsequently *Procedure*(2) of Case 2 is called for vertex i, otherwise the spanning tree of the solution is restored and the next edge is considered. If *Procedure*(3) fails on all edges incident to i, then the degree of i cannot be reduced and the next branch vertex is considered.

We do not consider the branch vertices with local degree greater than 5, as limited computational experiments showed no significant improvements in results. Besides, computational cost of considering the branch vertices with local degree greater than 5 is relatively high.

4. Heuristic for MDS

It is known that MBV and MDS are *related* problems, but not the *same*. Therefore, our proposed heuristic for MDS (referred to as Heu_MDS) is similar to the heuristic for MBV and begins by selecting a vertex randomly. *Phase* 1 of our MDS heuristic is exactly same as the *phase* 1 of MBV heuristic, however, *phase* 2 differs in the manner the set S of unselected vertices adjacent to newly created branch vertex is processed. In *phase* 2 of the heuristic for MBV, whenever a vertex is made a branch vertex v_b, then all edges connecting v_b to vertices in set S are added to the partially constructed spanning tree. But this policy is not used in our heuristic for MDS. Instead, a vertex v_u having minimum degree (ties are broken arbitrarily) is selected from among the vertices in S and the edge connecting v_b and v_u is added into the partially constructed tree. After this, *phase* 1 is applied (without going into *phase* 2 upon failure) by setting v_u as the last selected vertex. S is re-computed to account for any change introduced due to the application of *phase* 1. Now, another vertex with minimum degree is chosen from S and processed in the same manner as v_u. *Phase* 2 restarts when S becomes empty. This new policy avoids unnecessary

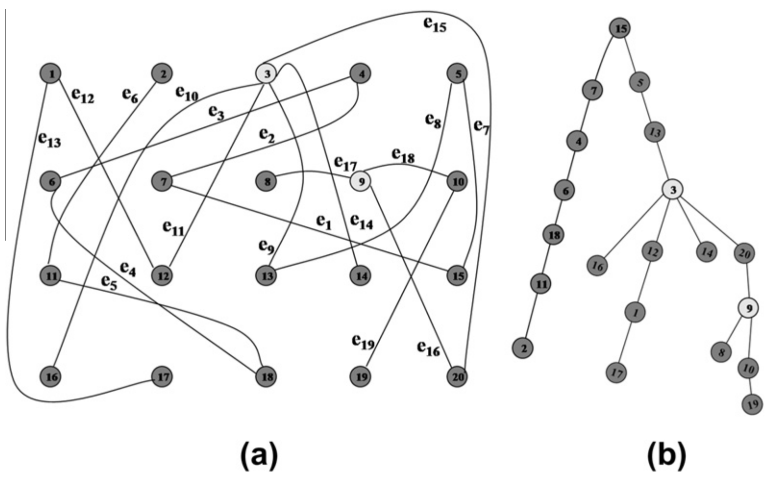

Fig. 3. Output of MDS heuristic.

increasing the degree of branch vertices. However, when it has to choose between increasing the degree by one of a branch vertex and creating a new branch vertex, it prefers the former. MDS heuristics of [1] also avoids as far as possible increasing the degree of branch vertices.

Fig. 3a shows the spanning tree obtained through MDS heuristic on the input graph of Fig. 1. Our MDS heuristic also begins by selecting a vertex randomly. To understand the difference between heuristics for MBV and MDS, let us assume our MDS heuristic also begins by selecting vertex 15. Exactly like MBV heuristic, edges (15,7), (7,4) (4,6), (6,18), (18,11), (11,2), (15,5), (5,13), (13,3) and (3,16) are added to the tree. At this stage, we cannot proceed further using *phase* 1 of the heuristic so we enter *phase* 2. Among all the selected vertices, vertex 3 has maximum number of unselected adjacent vertices so vertex 3 is made a branch vertex. Beginning at this juncture, MDS heuristic proceeds in a way different from MBV heuristic. Vertices 1, 12, 14 and 20 are unselected vertices adjacent to vertex 3. Of these vertices, 12, 14 and 20 have the minimum degree 2. Breaking the tie arbitrarily and selecting vertex 12 will add edge (3,12) to the tree. Now, *phase* 1 is restarted by setting vertex 12 as the last selected vertex. This leads to edges (12,1), (1,17) added to the tree and we return to *phase* 2. Next, we break the tie between 14 and 20 by selecting vertex 14 which leads to the addition of edge (3,14) to the tree. Now, *phase* 1 is called with vertex 14 as the last selected vertex, but there is no unselected vertex adjacent to vertex 14, so we again return to *phase* 2. Next edge (3,20) is added to the tree and *phase* 1 is called with vertex 20 as the last selected vertex. Now, edges (20,9), (9,8) are added to the tree. Then, *phase* 2 restarts again. This time vertex 9 is selected as a branch vertex and edge (9,10) is added to the tree. Again we call *phase* 1 with vertex 10 as the last selected vertex. Now, *phase* 1 adds edge (10,19) and *phase* 2 restarts. This time, since, there is no unselected vertex so the spanning tree is complete and heuristic stops. Note that MDS heuristic obtains a spanning tree with degree sum of branch vertices equal to 8, whereas the degree sum of branch vertices is 9 in the spanning tree obtained through MBV heuristic. Incidentally, in this particular example the tree obtained through MDS heuristic is optimal. For the sake of clarity, the spanning tree of Fig. 3a is redrawn in Fig. 3b.

5. ACO for MDS

Like ACO_MBV, we also attack MDS with the $\mathcal{MMAS}$ algorithm. This algorithm is referred to as ACO_MDS in this paper. ACO_MDS uses same ideas as ACO_MBV, but only after suitable modifications. Therefore, in this section we will describe only the differences and similarities between ACO_MDS and ACO_MBV. Like ACO_MBV, ACO_MDS also uses two pheromones and pheromones are updated using both iteration best and global best solutions. ACO_MDS is also coupled with a local search that is derived from the local search used in ACO_MBV.

5.1. Construction of solution

During each iteration *t*, each ant *k* constructs a solution in a manner which is similar to MDS heuristic of Section 4 except the selection of an edge connecting the last selected vertex *i* to an unselected vertex and the selection of a branch vertex, which are done probabilistically with the help of pheromones and heuristic information. The edge and branch vertex selection probabilities are still determined using Eqs. (1) and (2).

5.2. Pheromone update

Pheromone update rules for ACO_MDS are exactly same as ACO_MBV.

5.3. Local search

Local search tries to reduce the degree sum of branch vertices of the solution as much as possible. ACO_MDS with local search will be referred to as ACO_MDS + LS. Like the local search for MBV, here also the branch vertices of the spanning tree are sorted according to non-decreasing order of their local degrees. Also branch vertices with $C[i] = 3$ or $C[i] = 4$ or $C[i] = 5$ are considered one-by-one. Cases to handle each of these three classes of branch vertices are similar to respective cases of the local search for MBV, only the number of procedures and rules to select new edges differ to account for different structure of MDS. In addition, we have one more case to handle each branch vertex i with $C[i] > 5$. If a case has more than one procedure, then they are coupled in the same manner as in the case of the local search for MBV.

Case 1: This case is similar to Case 1 of the local search for MBV. Only the rules governing the selection of edges differ. This case consists of three procedures – *Procedure*(1_a), *Procedure*(1_b) and *Procedure*(1_c). These procedures are applied one after the other till one of them finds a suitable edge or all of them fail to find a suitable edge. *Procedure*(1_a) is similar to *Procedure*(1_a) of the local search for MBV except it uses the following two rules for edge selection:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0))$.
2. If rule 1 fails and if $(C[r] = 2)$, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to the vertex $r \in L_2$.

Procedure(1_b) is similar to *Procedure*(1_b) of the local search for MBV except on following two points: first, upon failure of *Procedure*(1_b) to find a suitable edge, here *Procedure*(1_c) is called. Second, it uses the following rules for edge selection.

Table 1
Results of MBV on Genmax instances.

Instance		Xpress-MP		C.A.		Heu_MBV		ACO_MBV-EP		ACO_MBV		ACO_MBV + LS	
V	d	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET
20	1.5	1.40	0.09	3.00	0.00	2.20	0.00	1.40	0.25	1.40	0.26	1.40	0.65
20	2	0.60	0.10	2.20	0.00	1.00	0.00	0.60	0.13	0.60	0.22	0.60	0.30
20	4	0.00	0.10	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	10	0.00	0.13	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	15	0.00	0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	1.5	2.40	0.18	5.20	0.00	3.40	0.00	2.40	0.49	2.40	0.56	2.40	1.47
30	2	0.60	0.40	3.60	0.00	1.60	0.00	0.60	0.33	0.60	0.32	0.60	0.65
30	4	0.40	0.41	2.80	0.00	1.40	0.00	0.40	0.19	0.40	0.20	0.40	0.39
30	10	0.00	0.83	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	15	0.00	0.51	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	1.5	2.00	0.31	5.60	0.00	3.00	0.00	2.20	0.72	2.00	0.63	2.00	1.98
40	2	0.60	0.47	4.20	0.00	1.60	0.00	0.60	0.34	0.60	0.43	0.60	1.71
40	4	0.00	0.59	1.80	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	10	0.00	0.88	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	15	0.00	1.60	0.80	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	1.5	3.20	0.70	8.00	0.00	5.00	0.00	3.40	0.96	3.20	1.07	3.20	2.67
50	2	0.80	0.71	6.00	0.00	2.20	0.00	1.00	0.93	0.80	0.83	0.80	1.07
50	4	0.00	0.83	3.20	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	10	0.00	1.51	1.40	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	15	0.00	1.68	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	1.5	5.80	12.11	15.20	0.00	9.80	0.00	6.40	3.00	6.20	2.66	6.00	9.51
100	2	2.00	11.22	12.60	0.00	4.80	0.00	2.20	2.87	2.00	2.72	2.00	4.43
100	4	0.00	4.78	6.00	0.00	1.80	0.00	0.00	0.01	0.00	0.05	0.00	0.01
100	10	0.00	14.42	2.40	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	15	0.00	13.66	1.20	0.01	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
300	1.5	18.20*	dnf	47.80	0.00	32.80	0.00	25.00	24.91	21.20	19.08	20.40	74.15
300	2	4.80*	dnf	34.80	0.01	17.00	0.00	11.80	22.49	7.40	15.59	6.80	38.94
300	4	0.00	92.09	18.00	0.03	4.80	0.00	1.20	14.80	0.40	10.73	0.20	12.71
300	10	0.00	554.34	8.40	0.11	2.20	0.00	0.40	8.57	0.00	3.49	0.00	0.58
300	15	0.00	1345.46	5.80	0.13	1.60	0.00	0.00	0.07	0.00	0.29	0.00	0.03
500	1.5	28.80*	dnf	79.40	0.02	53.40	0.00	45.00	59.58	35.00	58.64	33.40	251.31
500	2	7.60*	dnf	58.60	0.05	26.20	0.00	21.40	58.48	11.80	47.57	10.60	99.51
500	4	0.20	1162.97	28.60	0.09	9.00	0.00	4.20	45.46	0.60	26.80	0.60	49.57
500	10	0.00	2537.12	14.40	0.24	3.20	0.00	1.20	33.30	0.00	12.91	0.00	15.02
500	15	0.00	2728.29	7.60	0.37	2.20	0.00	0.40	23.03	0.00	10.28	0.00	5.87
1000	1.5	59.00*	dnf	159.80	0.13	106.40	0.00	95.20	291.69	78.20	270.74	72.20	1289.14
1000	2	16.60*	dnf	115.20	0.18	56.80	0.00	50.20	245.81	28.40	175.24	24.60	485.64
1000	4	0.60	3410.95	60.60	0.40	16.60	0.00	12.20	145.52	1.20	102.42	1.00	203.92
1000	10	0.00*	dnf	22.20	1.02	6.60	0.00	3.40	122.06	0.00	53.21	0.00	88.37
1000	15	0.00*	dnf	15.00	1.66	4.80	0.00	2.40	113.32	0.00	55.89	0.00	57.95

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $(C[v_2] > 2)$.
2. If first rule fails, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] > 2)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0)$ *or* $(C[v_2] > 2))$.

Procedure(1_c) is similar to *Procedure*(1_b) of local search for MBV except the rule for edge selection is as follows.

1. If $(C[r] > 1)$, then find an edge connecting a vertex $v_2 \in L_2$ with $(C[v_2] = 1)$ to the vertex $i \in L_1$.

Case 2: This case considers each branch vertex i with $C[i] = 4$. It consists of two procedures viz. *Procedure*(2_a) and *Procedure*(2_b). *Procedure*(2_a) is similar to *Procedure*(2) of the local search for MBV except on the following points: first, upon finding a new edge, it calls procedures of Case 1 of MDS. Second, upon failure, *procedure*(2_b) is called here. Third, following rules are used for edge selection:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0))$.
2. If rule 1 fails and if $(C[r] > 1)$, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to the vertex $r \in L_2$.
3. If both rule 1 and 2 fail and if $(C[r] > 1)$, then find an edge connecting a vertex $v_2 \in L_2$ with $(C[v_2] = 1)$ to the vertex $i \in L_1$.

Procedure(2_b) is similar to *Procedure*(2_a) except on two counts. First, upon failure, the degree of vertex i cannot be reduced. Second, it uses the following rules for edge selection:

Table 2
Results of MBV on Netgen instances.

Instance		Xpress-MP		C.A.		Heu_MBV		ACO_MBV-EP		ACO_MBV		ACO_MBV + LS	
V	d	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET
20	1.5	0.00	0.05	2.40	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	2	0.00	0.11	1.40	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	4	0.00	0.16	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	10	0.00	0.20	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	15	0.00	0.32	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	1.5	0.00	0.09	3.20	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	2	0.00	0.17	2.60	0.00	1.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	4	0.00	0.42	1.20	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	10	0.00	0.67	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	15	0.00	0.70	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	1.5	0.00	0.20	4.40	0.00	1.60	0.00	0.20	0.09	0.00	0.00	0.00	0.00
40	2	0.00	0.40	3.00	0.00	1.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	4	0.00	0.68	2.40	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	10	0.00	1.03	1.00	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	15	0.00	1.22	1.00	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	1.5	0.00	0.19	4.00	0.00	1.20	0.00	0.00	0.01	0.00	0.00	0.00	0.00
50	2	0.00	0.56	4.40	0.00	2.20	0.00	0.00	0.02	0.00	0.00	0.00	0.00
50	4	0.00	1.09	2.40	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	10	0.00	1.47	1.40	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	15	0.00	1.24	1.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	1.5	0.00	1.09	10.00	0.00	3.80	0.00	0.40	0.67	0.00	0.06	0.00	0.18
100	2	0.00	2.75	9.20	0.00	4.40	0.00	0.20	0.80	0.00	0.18	0.00	0.36
100	4	0.00	4.19	5.20	0.00	2.00	0.00	0.20	0.77	0.00	0.05	0.00	0.02
100	10	0.00	8.95	2.60	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	15	0.00	11.53	2.00	0.01	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
300	1.5	0.00	8.24	26.20	0.00	8.80	0.00	3.80	12.28	0.20	2.17	0.00	5.01
300	2	0.00	21.96	24.40	0.00	11.20	0.00	4.40	18.56	0.20	3.97	0.00	10.63
300	4	0.00	104.53	17.80	0.03	4.80	0.00	1.60	16.56	0.00	3.78	0.00	9.66
300	10	0.00	504.14	7.80	0.10	1.80	0.00	0.00	0.94	0.00	2.45	0.00	0.36
300	15	0.00	1675.10	7.00	0.14	1.20	0.00	0.00	0.06	0.00	0.04	0.00	0.01
500	1.5	0.00	22.10	45.80	0.03	18.40	0.00	9.00	27.51	1.00	15.25	0.40	25.07
500	2	0.00	100.59	43.40	0.06	19.20	0.00	11.20	48.86	0.60	19.24	0.20	43.24
500	4	0.00	646.94	26.40	0.11	9.40	0.00	4.60	44.12	0.20	18.90	0.00	29.48
500	10	0.00	2101.29	15.00	0.23	3.20	0.00	0.80	26.67	0.00	11.69	0.00	11.47
500	15	0.00	3376.00	12.00	0.38	2.20	0.00	0.20	10.17	0.00	9.56	0.00	2.11
1000	1.5	0.00	121.11	96.80	0.10	36.00	0.00	23.60	133.90	2.20	58.10	1.40	155.71
1000	2	0.00	934.20	85.60	0.17	34.20	0.01	27.40	167.79	1.60	86.61	0.60	213.57
1000	4	0.00	3375.60	50.40	0.39	17.40	0.01	13.00	156.24	0.40	85.30	0.00	160.54
1000	10	0.00*	dnf	30.60	1.06	5.20	0.00	2.80	110.05	0.00	56.06	0.00	86.81
1000	15	0.00*	dnf	22.80	1.57	4.40	0.00	1.60	117.65	0.00	60.81	0.00	57.13

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $(C[v_2] > 3)$.
2. If rule 1 fails, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] > 3)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0)$ *or* $(C[v_2] > 3))$.

Case 3: This case considers each branch vertex *i* with $C[i] = 5$. It also consists of two procedures viz. *Procedure*(3_a) and *Procedure*(3_b). *Procedure*(3_a) is similar to *Procedure*(3) of MBV except on the following points: first, upon finding a new edge it calls procedures of Case 2 of MDS. Second, upon failure, *procedure*(3_b) is called here. Third, following rules are used for edge selection:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0))$.
2. If rule 1 fails and if $(C[r] > 1)$, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to the vertex $r \in L_2$.
3. If both rules 1 and 2 fail and if $(C[r] > 1)$, then find an edge connecting a vertex $v_2 \in L_2$ with $(C[v_2] = 1)$ to the vertex $i \in L_1$.

Procedure(3_b) is similar to *Procedure*(3_a) except on two counts. First, upon failure, the degree of vertex *i* cannot be reduced. Second, it uses the following rules for edge selection:

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $(C[v_2] > 4)$.
2. If rule 1 fails, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] > 4)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0)$ *or* $(C[v_2] > 4))$.

Case 4: This case has only one procedure viz. *Procedure*(4), which considers each branch vertex *i* with $C[i] > 5$. Like other procedures, *Procedure*(4) deletes an edge e_{ir} which makes the solution infeasible. To make this solution feasible again, an edge is searched in *G*, which can connect these two components, in the following way:

Table 3
Results of MBV on Random instances.

Instance		Xpress-MP		C.A.		Heu_MBV		ACO_MBV-EP		ACO_MBV		ACO_MBV + LS	
V	d	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET
20	1.5	1.00	0.12	2.40	0.00	1.80	0.00	1.00	0.19	1.00	0.21	1.00	0.55
20	2	0.00	0.13	2.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	4	0.00	0.16	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	1.5	2.60	0.32	5.00	0.00	3.20	0.00	2.60	0.48	2.60	0.62	2.60	1.72
30	2	0.20	0.13	3.20	0.00	1.00	0.00	0.20	0.08	0.20	0.10	0.20	0.21
30	4	0.00	0.48	1.40	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	10	0.00	1.05	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	1.5	3.20	0.45	6.00	0.00	4.60	0.00	3.20	0.78	3.20	0.81	3.20	2.48
40	2	1.20	0.76	4.40	0.00	2.20	0.00	1.40	0.76	1.20	0.84	1.20	1.45
40	4	0.00	0.70	2.40	0.00	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	10	0.00	0.99	0.40	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	15	0.00	2.59	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	1.5	3.00	0.60	7.60	0.00	5.20	0.00	3.60	1.08	3.00	0.97	3.00	2.64
50	2	1.00	1.26	4.80	0.00	2.00	0.00	1.00	0.90	1.00	0.97	1.00	1.57
50	4	0.00	0.09	2.80	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	10	0.00	2.27	0.80	0.00	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	15	0.00	3.11	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	1.5	6.60	11.41	16.20	0.00	11.00	0.00	7.20	3.25	6.80	2.97	6.60	10.38
100	2	2.00	69.32	11.00	0.00	5.00	0.00	2.40	2.82	2.00	2.62	2.20	4.11
100	4	0.00	7.03	5.00	0.00	1.40	0.00	0.00	0.01	0.00	0.04	0.00	0.01
100	10	0.00	13.60	2.00	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	15	0.00	25.65	1.00	0.01	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
300	1.5	18.00*	dnf	49.00	0.00	31.80	0.00	25.00	22.79	21.00	21.40	20.20	89.76
300	2	5.20	3424.07	35.60	0.01	17.00	0.00	11.80	22.56	7.40	18.15	6.60	46.09
300	4	0.00	99.72	16.60	0.03	4.60	0.00	1.20	15.79	0.00	4.43	0.00	6.58
300	10	0.00	1104.37	7.40	0.08	2.80	0.00	0.60	10.63	0.00	3.17	0.00	0.71
300	15	0.00	1307.45	5.80	0.14	1.40	0.00	0.20	3.43	0.00	0.80	0.00	0.05
500	1.5	28.00*	dnf	78.20	0.02	51.60	0.00	43.20	71.81	34.60	60.86	32.60	250.44
500	2	8.60*	dnf	57.80	0.03	28.80	0.00	23.20	52.17	13.80	42.68	11.80	124.08
500	4	0.20	1549.24	28.80	0.11	7.60	0.00	4.20	45.90	0.80	30.11	0.60	50.43
500	10	0.00	3216.41	11.80	0.27	4.20	0.00	1.40	36.55	0.00	10.73	0.00	16.12
500	15	0.00	2900.89	8.40	0.42	2.60	0.00	1.00	38.12	0.00	10.55	0.00	3.78
1000	1.5	58.00*	dnf	151.20	0.11	104.40	0.00	94.60	265.30	76.00	222.84	70.20	1371.29
1000	2	15.40*	dnf	115.00	0.17	55.20	0.01	50.00	235.61	27.40	186.69	23.20	509.28
1000	4	0.20	3446.78	60.40	0.37	14.40	0.00	12.80	171.65	0.80	100.47	0.60	206.81
1000	10	0.00*	dnf	23.20	1.24	7.20	0.00	4.00	113.99	0.00	47.05	0.00	75.68
1000	15	0.00*	dnf	16.60	2.09	5.00	0.00	2.00	106.29	0.00	52.47	0.00	71.28

1. Find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to a vertex $v_2 \in L_2$ with $((C[v_2] = 1)$ *or* $(C[v_2] = 0))$.
2. If rule 1 fails and if $(C[r] > 1)$, then find an edge connecting a vertex $v_1 \in L_1$ with $(C[v_1] = 1)$ to the vertex $r \in L_2$.
3. If both rule 1 and 2 fail and if $(C[r] > 1)$, then find an edge connecting a vertex $v_2 \in L_2$ with $(C[v_2] = 1)$ to the vertex $i \in L_1$.

If the search is successful, then the searched edge is added to the partial spanning tree of the solution and the local degree of the branch vertex *i* or the branch vertex *r* (if *r* also happens to be a branch vertex) or both in the spanning tree is reduced by one, otherwise the spanning tree of the solution is restored by inserting edge e_{ir} and the search proceeds like other procedures.

6. Computational results

Our heuristics for MBV and MDS have been implemented in C and executed on a Linux based 3.2 GHz Pentium 4 system with 1 GB RAM. In all our computational experiments with ACO_MBV + LS and ACO_MDS + LS, we have used a colony of 25 ants. We have set $\alpha_e = 2$, $\beta_e = 2$, $\alpha_v = 1$, $\beta_v = 2$, $\rho = 0.98$, $P = 0.1$, $Q = 0.1$ and $GB_{it} = 20$. All pheromone values are constrained in the interval [0.01, 5] and are initialized to 5 at the beginning. We have allowed ACO_MBV + LS and ACO_MDS + LS to execute for 3000 iterations. These approaches may terminate before 3000 iterations, if a solution without a branch vertex is found. All these parameter values are chosen empirically after a large number of trials. These parameter values provide good results though they may not be optimal for all instances.

As the instances that were used in [1] are unavailable, therefore, to compare our approaches with the heuristics of [1], we have generated problem instances by using the same three graph instance generators viz., *Netgen*, *Genmax* and *Random* as used in [1]. All these generators are available from DIMACS (http://dimacs.rutgers.edu/). *Netgen* and *Genmax* are network

Table 4
Results of MDS on Genmax instances.

Instance		Xpress-MP		E.W.		Heu_MDS		ACO_MDS-EP		ACO_MDS		ACO_MDS + LS	
V	d	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET
20	1.5	5.20	0.06	9.60	0.00	9.60	0.00	5.40	0.26	5.20	0.28	5.20	0.98
20	2	2.00	0.20	4.80	0.00	4.00	0.00	2.00	0.15	2.00	0.20	2.00	0.52
20	4	0.00	0.16	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	10	0.00	0.21	5.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	15	0.00	0.32	5.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	1.5	9.20	0.27	15.80	0.00	14.00	0.00	9.40	0.48	9.20	0.53	9.20	2.31
30	2	2.60	0.35	9.00	0.00	7.80	0.00	2.80	0.23	2.60	0.37	2.60	0.77
30	4	1.80	0.23	7.80	0.00	5.60	0.00	1.80	0.16	1.80	0.24	1.80	0.48
30	10	0.00	0.64	9.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	15	0.00	0.65	7.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	1.5	8.20	0.25	18.60	0.00	18.20	0.00	8.60	0.67	8.20	0.66	8.20	2.27
40	2	1.80	25.73	9.60	0.00	7.60	0.00	1.80	0.36	1.80	0.41	1.80	0.97
40	4	0.00	0.56	9.00	0.00	1.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	10	0.00	1.05	13.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	15	0.00	1.68	12.00	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	1.5	14.20	0.87	25.60	0.00	27.00	0.00	14.40	1.00	14.40	1.11	14.20	5.05
50	2	4.00	19.17	15.00	0.00	16.20	0.00	5.20	0.95	4.00	0.74	4.00	1.60
50	4	0.00	0.80	7.20	0.00	2.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	10	0.00	1.77	15.00	0.00	1.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	15	0.00	1.79	14.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	1.5	28.60	27.25	55.00	0.00	54.00	0.00	30.20	2.96	29.60	2.81	28.80	15.58
100	2	10.80	738.78	29.40	0.00	34.40	0.00	13.20	2.60	11.20	2.68	11.20	8.25
100	4	0.00	3.33	21.00	0.00	10.40	0.00	0.00	0.00	0.00	0.05	0.00	0.02
100	10	0.00	9.05	22.20	0.01	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	15	0.00	6.60	25.80	0.03	1.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
300	1.5	94.40*	dnf	174.60	0.01	180.60	0.00	114.80	21.70	104.40	18.82	99.20	172.70
300	2	35.80*	dnf	103.20	0.01	107.80	0.00	59.20	19.81	41.60	19.02	40.20	83.19
300	4	0.00	23.91	58.20	0.05	24.80	0.00	7.00	9.07	0.00	5.59	0.00	13.15
300	10	0.00	110.05	76.40	0.15	14.80	0.00	0.60	5.04	0.00	2.56	0.00	0.74
300	15	0.00	276.30	88.20	0.22	9.40	0.00	0.00	0.16	0.00	0.93	0.00	0.04
500	1.5	153.80*	dnf	292.40	0.03	297.40	0.00	200.40	67.56	172.20	55.69	164.80	533.67
500	2	59.60*	dnf	169.40	0.06	185.20	0.00	104.80	52.48	72.00	42.74	66.80	210.87
500	4	0.80	910.99	88.80	0.14	58.00	0.00	19.80	34.67	1.00	20.80	1.00	45.98
500	10	0.00	1573.46	146.60	0.39	27.00	0.00	5.80	30.61	0.00	10.82	0.00	16.18
500	15	0.00	1198.72	130.00	0.65	16.60	0.00	1.20	24.89	0.00	11.58	0.00	4.00
1000	1.5	320.60*	dnf	606.40	0.15	603.20	0.01	427.40	253.48	375.00	250.94	352.80	2810.33
1000	2	128.00*	dnf	353.60	0.22	386.80	0.03	236.20	234.31	167.00	188.31	151.20	1116.61
1000	4	2.00*	dnf	183.60	0.49	118.60	0.01	54.20	160.32	3.20	97.76	2.60	273.84
1000	10	0.00	2027.56	272.20	1.78	51.40	0.01	20.00	121.72	0.00	44.99	0.00	109.59
1000	15	0.00*	dnf	297.40	2.92	39.20	0.01	12.60	102.66	0.00	51.94	0.00	90.03

flow problem instance generators which generates problem instances with random edges and uniform capacity. Like [1], we have discarded the edge capacity and transformed the network from directed to undirected. For each generator, we have also taken eight different values of $|V|$ into account, i.e., $|V| = 20, 30, 40, 50, 100, 300, 500, 1000$. For each value of $|V|$, we have taken five different values of density (the ratio of the number of edges to the number of vertices) into account, i.e., $d = 1.5, 2, 4, 10, 15$. However, we are not able to generate instances with $|V| = 20$, $d = 10$; $|V| = 20$, $d = 15$; $|V| = 30$, $d = 15$ using *Random* instance generator due to some internal restrictions in the generator program. Thus, we have 40 different combinations of V and d for each generator except *Random* for which we have 37 combinations. For each combination, a set of five instances were generated leading to overall 117 sets. Similar to [1], for each set we report the average of these five instances.

In addition to generating instances, for the purpose of comparison, we have implemented best heuristics for MBV and MDS presented in [1] viz. Heuristic C.A. (combined approach of edge weighting and node coloring) for MBV and heuristic E.W. (edge weighting approach) for MDS. Moreover, in order to get exact solution or lower bound for each instance, the mathematical formulations presented in [1] for MBV and MDS are solved using Xpress-MP solver (http://www.dashoptimization.com/). Results of different approaches for MBV are reported in Tables 1–3, whereas for MDS they are reported in Tables 4–6. For each instance set and each method these tables report the average solution quality and average execution time (AET) in seconds.

We have allowed Xpress-MP solver to execute for at most one hour time on each instance. When Xpress-MP solver cannot find a solution in one hour, we will get a lower bound on the solution. Such cases are reported in the tables with a '*' in the value column and *dnf* (did not finish) in AET column.

In subsequent subsections, we compare the performances of different approaches for MBV and MDS.

Table 5
Results of MDS on Netgen instances.

Instance		Xpress-MP		E.W.		Heu_MDS		ACO_MDS-EP		ACO_MDS		ACO_MDS + LS	
V	d	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET
20	1.5	0.00	0.02	1.80	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	2	0.00	0.09	2.40	0.00	1.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	4	0.00	0.06	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	10	0.00	0.29	3.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	15	0.00	0.25	5.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	1.5	0.00	0.10	5.40	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	2	0.00	0.20	8.40	0.00	4.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	4	0.00	0.28	3.00	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	10	0.00	0.75	7.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	15	0.00	0.39	8.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	1.5	0.00	0.12	9.60	0.00	5.00	0.00	0.60	0.11	0.00	0.00	0.00	0.00
40	2	0.00	0.42	9.00	0.00	6.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	4	0.00	0.44	6.60	0.00	1.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	10	0.00	1.07	10.20	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	15	0.00	1.78	11.40	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	1.5	0.00	0.15	9.00	0.00	4.20	0.00	0.00	0.00	0.00	0.00	0.00	0.01
50	2	0.00	0.46	10.20	0.00	8.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	4	0.00	0.89	11.40	0.00	2.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	10	0.00	2.22	12.00	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	15	0.00	1.13	11.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	1.5	0.00	0.85	15.00	0.00	13.60	0.00	0.60	0.43	0.00	0.06	0.00	0.18
100	2	0.00	1.79	22.20	0.00	15.20	0.00	0.60	0.60	0.00	0.15	0.00	0.32
100	4	0.00	3.37	19.20	0.00	13.00	0.00	0.60	0.78	0.00	0.06	0.00	0.01
100	10	0.00	9.00	27.00	0.00	2.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	15	0.00	22.35	27.60	0.02	2.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
300	1.5	0.00	9.35	52.80	0.00	44.20	0.00	14.00	10.82	0.60	2.32	0.00	7.18
300	2	0.00	39.55	54.60	0.01	52.80	0.00	17.80	15.09	0.00	3.43	0.00	12.72
300	4	0.00	73.84	66.00	0.03	32.00	0.00	6.80	16.34	0.00	3.89	0.00	10.04
300	10	0.00	47.79	82.80	0.15	13.20	0.01	0.60	2.62	0.00	3.14	0.00	0.42
300	15	0.00	187.73	53.40	0.20	7.20	0.00	0.00	0.12	0.00	0.27	0.00	0.02
500	1.5	0.00	18.55	82.20	0.03	80.00	0.00	30.40	30.08	2.20	12.55	0.60	24.18
500	2	0.00	252.68	96.60	0.06	88.40	0.00	42.60	44.12	0.00	13.61	0.00	45.12
500	4	0.00	210.41	97.20	0.14	62.20	0.00	20.40	38.80	0.00	14.09	0.00	44.46
500	10	0.00	1539.98	131.80	0.39	24.80	0.00	3.20	28.68	0.00	10.83	0.00	14.49
500	15	0.00	882.76	129.40	0.56	14.00	0.00	0.80	12.19	0.00	9.79	0.00	2.14
1000	1.5	0.00	233.06	168.00	0.13	151.00	0.01	80.60	115.17	6.40	63.35	3.00	151.71
1000	2	0.00	2021.92	180.60	0.22	187.20	0.03	101.40	180.67	2.00	71.09	0.60	223.90
1000	4	0.00	2764.78	204.60	0.52	119.60	0.02	53.80	159.24	0.00	63.20	0.00	182.61
1000	10	0.00	905.48	274.80	1.72	40.60	0.01	13.20	92.35	0.00	57.91	0.00	96.96
1000	15	0.00	3140.60	282.00	2.82	33.60	0.01	8.60	103.33	0.00	45.86	0.00	68.12

6.1. Comparison of our approaches with C.A. heuristic for MBV

We first compare our heuristic Heu_MBV with C.A. heuristic [1]. Tables 1–3 clearly show the superiority of our Heu_MBV over C.A. in terms of both solution quality as well as execution time. Heu_MBV always performs as good as or better than C.A. in terms of both these parameters. Solutions obtained by Heu_MBV are better than C.A. on 113 instance sets and equal on four remaining sets. Heu_MBV is much faster than C.A. on larger instances.

However, comparing the results obtained by Heu_MBV with those of Xpress-MP solver clearly show that Heu_MBV, in itself, is not so robust. Moreover, the difference in solution quality grows with increase in instances size. Therefore, to get good solutions even on larger instances, we have proposed ACO_MBV + LS as described in Section 3. ACO_MBV + LS is able to find 94 optimal solutions out of a total of 100 instance sets for which we know the optimal solutions. We have idea about the optimal solution values of only 100 instance sets, since Xpress-MP solver is able to find 100 optimal solutions out of 117 instance sets, therefore, results of ACO_MBV + LS are quite good.

To see the effect of local search, we also run our ACO approach without local search (referred to as ACO_MBV) while keeping all ACO parameters unchanged. ACO_MBV is able to find optimal solutions for 90 sets in comparison to 94 sets for ACO_MBV + LS. Overall, on all 117 sets, the results of ACO_MBV + LS is better than ACO_MBV on 26 sets, whereas it is worse on one set. As expected, ACO_MBV is much faster than ACO_MBV + LS.

Since, we have used two types of pheromone (pheromone on edges and pheromone on vertices) which is different from the traditional way of using only one pheromone. To our knowledge, Cordone et al. [4] is the only work where concept of multiple pheromones was used. To support our approach, we have also experimented with a single pheromone version of our ACO algorithm where we have deposited pheromone only on vertices. In this situation, instead of selecting an edge e_{ij} connecting the last selected vertex i in the partially constructed tree to an unselected adjacent vertex j based on pheromone concentration on edges, an edge e_{ij} is selected exactly like *phase* 1 of Heu_MBV. The resulting approach is referred to as ACO_MBV-EP. It is clear from Tables 1–3 that the results obtained by ACO_MBV-EP are worse or equal to ACO_MBV on all instances. There are 55 instances where ACO_MBV-EP performs worse than ACO_MBV. Most of these instances are large in-

Table 6
Results of MDS on Random instances.

Instance		Xpress-MP		E.W.		Heu_MDS		ACO_MDS-EP		ACO_MBV		ACO_MDS + LS	
V	d	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET	Value	AET
20	1.5	3.40	0.10	6.60	0.00	6.20	0.00	3.40	0.19	3.40	0.22	3.40	0.66
20	2	0.00	0.07	1.80	0.00	2.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	4	0.00	0.06	3.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	1.5	10.20	0.32	16.00	0.00	15.60	0.00	10.40	0.48	10.20	0.59	10.20	2.33
30	2	0.60	0.54	6.60	0.00	3.80	0.00	0.60	0.08	0.60	0.10	0.60	0.29
30	4	0.00	0.22	5.40	0.00	1.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	10	0.00	0.93	8.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	1.5	13.20	0.68	23.60	0.00	23.80	0.00	13.40	0.71	13.20	0.88	13.20	4.05
40	2	4.80	36.44	14.40	0.00	11.60	0.00	5.00	0.76	4.80	0.73	4.80	2.08
40	4	0.00	0.60	7.20	0.00	1.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	10	0.00	1.00	14.40	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	15	0.00	2.73	14.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	1.5	13.40	1.31	25.80	0.00	26.00	0.00	15.40	0.99	13.60	1.03	13.40	4.20
50	2	3.60	2.13	15.60	0.00	12.00	0.00	4.20	0.85	3.60	0.89	3.60	2.08
50	4	0.00	0.60	9.60	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	10	0.00	2.05	12.00	0.00	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	15	0.00	2.38	10.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	1.5	30.60	110.73	59.40	0.00	59.20	0.00	32.40	3.15	31.40	3.14	30.80	18.94
100	2	11.20	842.16	36.60	0.00	29.80	0.00	13.00	2.36	11.80	2.56	11.40	6.97
100	4	0.00	3.67	19.20	0.00	8.40	0.00	0.00	0.01	0.00	0.04	0.00	0.01
100	10	0.00	7.76	27.00	0.01	2.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	15	0.00	13.12	34.80	0.03	0.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00
300	1.5	91.00*	dnf	171.20	0.01	181.80	0.00	111.80	23.52	98.80	22.32	96.20	171.48
300	2	33.20*	dnf	92.40	0.02	114.40	0.00	55.00	19.62	39.80	16.24	37.20	76.63
300	4	0.00	37.83	51.60	0.06	29.60	0.00	5.60	12.91	0.00	3.57	0.00	10.56
300	10	0.00	50.23	77.20	0.15	18.00	0.00	1.80	9.89	0.00	3.27	0.00	0.67
300	15	0.00	471.79	90.80	0.23	8.20	0.00	0.60	5.31	0.00	1.06	0.00	0.06
500	1.5	151.80*	dnf	291.60	0.03	306.00	0.00	197.60	60.72	169.20	56.56	161.60	558.23
500	2	63.60*	dnf	174.60	0.04	190.80	0.00	111.60	60.12	76.00	47.30	73.00	248.26
500	4	0.40	1057.76	88.80	0.14	49.20	0.00	20.20	36.72	1.20	23.14	0.60	52.61
500	10	0.00	1817.40	133.20	0.40	29.80	0.00	5.80	30.98	0.00	12.26	0.00	19.61
500	15	0.00	1161.14	144.80	0.61	22.80	0.00	4.00	41.96	0.00	14.38	0.00	3.79
1000	1.5	309.80*	dnf	581.40	0.15	585.60	0.01	418.80	250.74	367.20	244.13	337.80	2762.58
1000	2	118.80*	dnf	334.80	0.22	370.60	0.03	233.00	245.86	158.20	173.53	142.40	1096.66
1000	4	0.40*	dnf	186.00	0.49	110.60	0.02	53.60	137.00	1.20	85.42	0.60	232.04
1000	10	0.00	2550.20	257.40	1.80	52.80	0.01	18.80	121.37	0.00	56.52	0.00	95.26
1000	15	0.00	2226.30	296.00	3.19	38.60	0.01	11.00	107.05	0.00	46.12	0.00	69.85

stances. Even AETs of ACO_MBV-EP on most of the instance sets are higher than ACO_MBV. Hence, this experimentation provides the justification for using two pheromones instead of one in our ACO approach for MBV.

As far as comparison of solution quality of different ACO variants with Heu_MBV and C.W. is concerned, all ACO variants obtain better results except on few smaller instances where results are same.

6.2. Comparison of our approaches with E.W. heuristic for MDS

We first compare Heu_MDS with E.W. heuristic [1]. Tables 4–6 clearly show the superiority of our Heu_MDS over E.W. in terms of solution quality as well as execution time. Solutions obtained by Heu_MDS are better than E.W. on 98 sets, worse on 18 sets and equal on 1 set. AETs for Heu_MDS are as good as or better than E.W. On larger instances our heuristic is much faster than E.W.

Similar to ACO_MBV + LS, ACO_MBV and ACO_MBV-EP, here also we have three ACO variants viz. ACO_MDS + LS, ACO_MDS and ACO_MDS-EP. ACO_MDS + LS and ACO_MDS are able to find 94 and 91 optimal results respectively. There are 102 instance sets for which Xpress-MP solver is able to find optimal results, so again ACO_MDS + LS and ACO_MDS perform quite well. Overall, there are 24 instances on which ACO_MDS + LS is better than ACO_MDS. Here also single pheromone version ACO_MDS-EP performs worse both in terms of solution quality and execution time thereby justifying once again the use of two pheromones. There are 61 instances where ACO_MDS-EP performs worse than ACO_MDS. Similar to MBV, here also all ACO variants obtain as good as or better quality solutions in comparison to Heu_MDS and E.W.

7. Conclusions

In this paper, we have developed heuristic approaches for two spanning tree problems viz. MBV and MDS having relevance in optical network design. For each of these two problems, our first approach is a problem specific heuristic whereas the second approach is based on ACO. Our problem specific heuristic approaches outperform the previously proposed best problem specific heuristic approaches for these problems. As the results obtained by all these problem specific heuristics are inferior to exact approaches and the difference in solution quality grows with instance size, we have proposed the hybrid ACO approaches. As our ACO approaches are the first metaheuristic approaches for MBV and MDS, therefore, they will serve as the baseline approaches for any future metaheuristic approaches for these problems. A significant feature of our ACO approaches is the use of two pheromones. Similar ACO approaches with multiple pheromones can be designed for other problems also.

As a future work, we intend to develop an artificial bee colony algorithm based approaches for MBV and MDS by combining the ideas presented here with those in [13,15].

References

[1] R. Cerulli, M. Gentili, A. Iossa, Bounded-degree spanning tree problems: models and new algorithms, Computational Optimization and Applications 42 (2009) 353–370.
[2] I. Chlamtac, A. Ganz, G. Karmi, Lightpath communications: an approach to high bandwidth optical WANs, IEEE Transactions on Communications 40 (1992) 1171–1182.
[3] J. Christmas, E. Keedwell, T.M. Frayling, J.R.B. Perry, Ant colony optimisation to identify genetic variant association with type 2 diabetes, Information Sciences 181 (2011) 1609–1622.
[4] R. Cordone, F. Cordone, F. Maffioli, Coloured ant system and local search to design local telecommunication networks, Lecture Notes in Computer Science, vol. 2037, Springer-Verlag, Berlin, 2001.
[5] M. Dorigo, V. Maniezzo, A. Colorni, Positive Feedback as a Search Strategy, Technical Report 91-016, 1991, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy.
[6] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man and Cybernetics, Part B 26 (1996) 29–42.
[7] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.
[8] L. Gargano, P. Hell, L. Stacho, U. Vaccaro, Spanning trees with bounded number of branch vertices, Lecture Notes in Computer Science, vol. 2380, Springer-Verlag, Berlin, 2002.
[9] T.-P. Hong, Y.-F. Tung, S.-L. Wang, Y.-L. Wu, M.-T. Wu, A multi-level ant-colony mining algorithm for membership functions, Information Sciences 182 (2012) 3–14.
[10] H.-J. Kim, S. Kang, Communication-aware task scheduling and voltage selection for total energy minimization in a multiprocessor system using ant colony optimization, Information Sciences 181 (2011) 3995–4008.
[11] Z.-G. Ren, Z.-R. Feng, A.-M. Zhang, Fusing ant colony optimization with Lagrangian relaxation for the multiple-choice multidimensional knapsack problem, Information Sciences 182 (2012) 15–29.
[12] L.H. Sahasrabuddhe, B. Mukherjee, Light-trees: optical multicasting for improved performance in wavelength-routed networks, IEEE Communications Magazine 37 (1999) 67–73.
[13] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, Applied Soft Computing 9 (2009) 625–631.
[14] T. Stützle, H. Hoos, $\mathcal{MAX}-\mathcal{MIN}$ ant system, Future Generation Computer Systems 16 (2009) 889–914.
[15] S. Sundar, A. Singh, A swarm intelligence approach to the quadratic minimum spanning tree problem, Information Sciences 180 (2010) 3182–3191.
[16] L. Zhang, Q. Cao, A novel ant-based clustering algorithm using the kernel method, Information Sciences 181 (2011) 4658–4672.
[17] X. Zhang, J.Y. Wei, C. Qiao, Constrained multicast routing in WDM networks with sparse light splitting, Journal of Lightwave Technology 18 (2000) 1917–1927.