Computer Communications 32 (2009) 154–158

Contents lists available at ScienceDirect

Computer Communications

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

A dandelion-encoded evolutionary algorithm for the delay-constrained capacitated minimum spanning tree problem

Angel M. Pérez-Bellido [a], Sancho Salcedo-Sanz [a,*], Emilio G. Ortiz-García [a], Antonio Portilla-Figueras [a], Maurizio Naldi [b]

[a] Departament of Signal Theory and Communications, Universidad de Alcalá, Campus Universitario, Alcala de Henares, 28871 Madrid, Spain
[b] Dipartimento di Informatica, Sistemi e Produzione, Universitá di Roma "Tor Vergata", Rome, Italy

A R T I C L E   I N F O

Article history:
Received 4 June 2008
Accepted 25 September 2008
Available online 10 October 2008

Keywords:
Delay-constrained capacitated minimum spanning tree
Evolutionary algorithms
Dandelion encoding
Heuristics

A B S T R A C T

This paper proposes an evolutionary algorithm with *Dandelion-encoding* to tackle the Delay-Constrained Capacitated Minimum Spanning Tree (DC-CMST) problem. This problem has been recently proposed, and consists of finding several broadcast trees from a source node, jointly considering traffic and delay constraints in trees. A version of the problem in which the source node is also included in the optimization process is considered as well in the paper. The Dandelion code used in the proposed evolutionary algorithm has been recently proposed as an effective way of encoding trees in evolutionary algorithms. Good properties of locality has been reported on this encoding, which makes it very effective to solve problems in which the solutions can be expressed in form of trees. In the paper we describe the main characteristics of the algorithm, the implementation of the Dandelion-encoding to tackled the DC-CMST problem and a modification needed to include the source node in the optimization. In the experimental section of this article we compare the results obtained by our evolutionary with that of a recently proposed heuristic for the DC-CMST, the *Least Cost* (LC) algorithm. We show that our Dandelion-encoded evolutionary algorithm is able to obtain better results that the LC in all the instances tackled.

1. Introduction

Network topology consists of specifying the configuration of the computer connections in a given network [1,2]. The design of the topology is also a key point in the deployment of computers and communications networks. Topology affects to very important points such as communication costs, transmission speed, average time delay of the network, etc [1,2]. There are several research works about topology design, both in backbone and local networks [3–5]. Regarding local networks, these can be modeled as a backbone node (source node) and several trees that cover all end user nodes to satisfy the constraints of traffic volume [1,2].

In the literature, the works related to topology discovery of local networks can be classified in two main problems: Capacitated Minimum Spanning Tree (CMST) and Delay-Constrained Minimum Spanning Tree (DCMST) problems [6–10]. The CMST problem consists of finding a set of minimum spanning trees rooted at the source node which satisfies a set of traffic constraints [8]. This problem is known to be NP-complete [9], and several heuristic approaches have been applied to solve it [6,7]. On the other hand, the DCMST problem consists of finding the least cost broadcast and multi-cast trees rooted at the source node, which have the minimum total cost among all possible spanning trees, and also have a maximum end-to-end delay bounded by a given delay constraint Δ [10,11].

Recently, two innovative works have proposed the joint optimization of the network topology in terms of the traffic capacity and its mean time delay. This joint optimization allows obtaining reasonable quality of service (QoS) rates in the current communication networks. The joint optimization of the network taking into account traffic and delay constraints, produces the so called *Delay-Constrained Capacitated Minimum Spanning Tree* (DC-CMST) problem, which was first presented in [1], and successfully solved using an ad-hoc heuristic called *Least-cost* (LC) heuristic. The LC heuristic starts from the solution to the CMST problem provided by the Esau and William's algorithm [7] (EW solution), and then applies two different versions of the *Mean Delay Capacity Allocation Algorithm* (MDCAA) in order to obtain feasible solutions for the DC-CMST. In a more recent paper [2], the same authors proposed an exact algorithm for the DC-CMST, which provide the optimal solution to the problem. They show that this exact algorithm can compute the DC-CMST solution in reasonable time only for small networks, of less than 30 nodes. For real-size networks, in which the number of nodes is larger, exact algorithms are computationally inefficient, and thus heuristics approaches are usually a better option.

* Corresponding author. Tel.: +34 91 885 6731; fax: +34 91 624 8749.
E-mail address: sancho.salcedo@uah.es (S. Salcedo-Sanz).

0140-3664/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.09.030

In this paper we propose an evolutionary algorithm to solve the DC-CMST, which uses a Dandelion encoding to represent the complete local network. The Dandelion encoding has been recently proposed as an effective method to represent trees in evolutionary algorithms [12], with good properties of locality (small changes in the representation make small changes in the tree). This locality produces a more effective evolutionary search than using other tree encodings such as Prüfer encoding [14]. In the paper we describe the main characteristics of the algorithm and the implementation of the Dandelion-encoding to tackle the DC-CMST problem. We also consider a different version of the DC-CMST problem, in which the source node is also part of the optimization process (in the traditional DC-CMST problem this source node is previously fixed). In the paper we discuss several modifications needed to tackle this new version of the problem. In the experimental section of this article, we compare the results obtained by our evolutionary with that of the *Least Cost* (LC) algorithm in [1].

The rest of the paper is structured as follows: next section introduces the DC-CMST problem following the description in [1]. Section 3 presents the main characteristics of the Dandelion-encoding evolutionary algorithm proposed in this paper. Special analysis of the encoding, initialization, evolutionary operators and adaptation to include the source node in the optimization problem are carried out. Section 4 shows the performance of the proposed approach by comparing the results obtained in different DC-CMST instances against that of the LC heuristic. Section 5 closes the paper giving some final remarks.

2. DC-CMST problem formulation

The DC-CMST problem has been recently proposed in [1]. In the formulation of the DC-CMST problem, a set of assumptions must be made:

1. There is only one source node, with unlimited capacity.
2. The traffic generated at a given node (q_i) cannot exceed the maximum traffic covered by one tree ($\max(q_i) \leqslant \varepsilon$, $i = 1,\cdots,n$).
3. The traffic at a given node is not splintered.
4. The total traffic exceeds the maximum traffic served by one of the trees conforming the network ($\sum_{i=1}^{n} q_i > \varepsilon$).
5. The arrival of packets is based on a Poisson distribution.
6. The service time of packets is exponentially distributed.

With these assumptions, our formulation of the DC-CMSTP follows the one given in [1]: Obtaining a tree network with minimum total link cost, which satisfies that the total network delay time (T) is bounded by a given time (Δ), and also that the traffic capacity limitation is less than a given value ε in each subtree. Fig. 1 shows

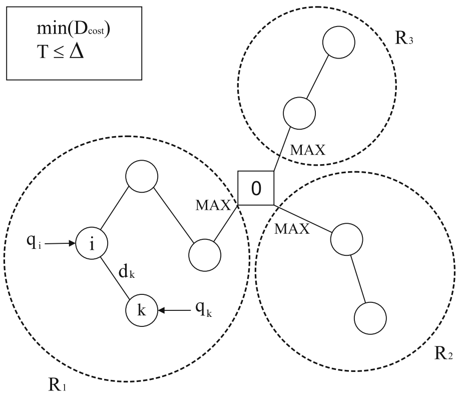

Fig. 1. An example of the DC-CMST problem.

an example of the problem definition. In order to provide a mathematical formulation of the DC-CMST problem, we need first to define several parameters of the problem. Let T be the mean time delay of the network, it can be defined as:

$$T = \frac{1}{v}\sum_{k=1}^{n} \lambda_k T_k \tag{1}$$

where v stands for the total traffic of the network (traffic between all source-destination pairs), λ_k is the traffic flow on link k in the current topology (packets/s) and T_k is the mean delay time of link k in the current topology. Since each link k can be regarded as an independent $M/M/1$ queue, the mean delay T_k of link k is given by the following equation [1]:

$$T_k = \frac{1}{(\mu \cdot C_k - \lambda_k)} \tag{2}$$

where $\frac{1}{\mu}$ is the average packet length (bits/packet), and C_k stands for the capacity of link k in the current topology (bits/s). If we include this expression in (1), we obtain the following equation for the total network mean time delay T:

$$T = \frac{1}{v}\sum_{k=1}^{n} \lambda_k \left(\frac{1}{(\mu \cdot C_k - \lambda_k)}\right) \tag{3}$$

The DC-CMST problem can now be mathematically stated, in the following way:

$$\min\left(D_{\text{cost}} = \sum_{k=1}^{n} d \cdot C_k \cdot d_k\right) \tag{4}$$

subject to:

$$\frac{\lambda_k}{\mu} \leqslant C_k \tag{5}$$

$$T = \frac{1}{v}\sum_{k=1}^{n} \lambda_k \left(\frac{1}{(\mu \cdot C_k - \lambda_k)}\right) \leqslant \Delta \tag{6}$$

$$\sum_{i \in R_j} q_i \cdot x_{ij} \leqslant \varepsilon \tag{7}$$

$$\sum_{i,j} x_{ij} = n \tag{8}$$

$$x_{ij} \in \{0,1\} \tag{9}$$

where, the objective function of the DC-CMSTP is given by Eq. (4), and consists of finding a collection of trees with minimal link cost (D_{cost}). Note that D_{cost} depend on the values of C_k (capacity of link k in the current topology (in seconds), d unit cost of link capacity and d_k defined as the distance between node k in a given tree and its antecessor node in that tree (note that this distance may be different depending on the considered tree). The constraints of the DC-CMSTP are the following: the average traffic flow on a link must be smaller than the capacity of the link, Eq. (5). The mean delay of the network has to be dropped within allowable mean delay time (Δ), Eq. (6). The total traffic flow in one tree (R_j) must be below a value ε, Eq. (7). Finally, Eq. (8) ensures that n nodes are included in the final solution. Note that variable x_{ij} is 1 if there is a link between nodes i and j in the current topology, and 0 otherwise, as Eq. (9) states.

2.1. DC-CMST problem with source node optimization

In this paper we consider a modification of the DC-CMST problem consisting of including the selection of the optimal source node in the DC-CMST problem. In the DC-CMST definition giving in [1 and 2], the source node was previously fixed, and it is not considered in the optimization process. In fact, if the network to be optimized is denoted as a graph $G(V,E)$, with V the set of nodes and E the set of edges, the number of nodes involved in the DC-CMST is $n = |V| - 1$, since the node 0 is considered to be the source node [1,2]. Note that this version of the DC-CMST is equivalent to launch $n + 1$ algorithms for the classical DC-CMST problems. Note also that, for large values of n, the computation time of algorithms for this new version of the problem may be unaffordable.

The mathematical formulation of this new version of the problem is quite similar to the traditional DC-CMST, but in this case the parameters d_k, C_k, λ_k, v depend on the specific tree and also on its root r (in the traditional DC-CMST definition these parameters only depend on the current topology considered, since the root node is always the node 0).

The DC-CMST problem with source node optimization can be stated in the following way (remaking the specific dependence on root node r):

$$\min\left(D_{\text{cost}} = \sum_{k=0}^{n} d \cdot C_k^r \cdot d_k^r\right) \tag{10}$$

subject to:

$$\frac{\lambda_k^r}{\mu} \leqslant C_k^r \tag{11}$$

$$T = \frac{1}{v^r}\sum_{k=1}^{n}\lambda_k^r\left(\frac{1}{(\mu \cdot C_k^r - \lambda_k^r)}\right) \leqslant \Delta \tag{12}$$

$$\sum_{i \in R_j^r} q_i \cdot x_{ij} \leqslant \varepsilon \tag{13}$$

$$\sum_{i,j} x_{ij} = n \tag{14}$$

$$x_{ij} \in \{0, 1\} \tag{15}$$

3. A Dandelion-encoded evolutionary algorithm for the DC-CMST problem

3.1. Algorithm encoding

The Dandelion code is a *Cayley-like* encoding [12] which has been recently described and used for encoding trees in genetic algorithms [12,13]. There are several decoding algorithm (string to tree) for the Dandelion code. In this paper we use the so-called *fast algorithm*, proposed by Piccioto in [15], which has been also used in [13]:

- **Input:** A Dandelion code $C = (c_2, c_3, \cdots, c_{n-1})$.
- **Output:** The tree $T \in \mathscr{T}_n$ corresponding to C.
- **Step 1:** Define the function ϕ_C:[2,$n - 1$] → [1,n] such that $\phi_C(i) = c_i$ for each $i \in [2, n - 1]$.
- **Step 2:** Calculate the cycles associated to the function ϕ_C, $Z_1, Z_2, \cdots, Z_k$. Let b_i be the maximum element in cycle Z_i. We assume that the cycles are recorded such that b_i is the rightmost element of Z_i, and that $b_i < b_j$ if $i < j$.
- **Step 3:** Form a single list π of the elements in $Z_1, Z_2, \cdots, Z_k$, in the order they occur in this cycle list, form the first element of Z_1 to the last element of Z_k.
- **Step 4:** Construct the tree $T \in \mathscr{T}_n$ corresponding to C in the following way: take a set of n isolated vertices (labeled with the integers from 1 to n), create a path from vertex 1 to vertex n by following the list π from left to right, and then create the edge (i, c_i) for every $i \in [2, n - 1]$ which does not occur in the list π.

We will illustrate this fast algorithm using the example in Fig. 2. This figure proposes the Dandelion code $C = (4,6,2,5,9,1,12,6,2,9)$. Note that there are three cycles in this case, $Z_1 = (6,9)$, $Z_2 = (5)$ and $Z_3 = (2,4)$. Note also the order in which we have recorded these cycles follows the indications in the step 2 of the fast decoder algorithm. We form then the list π=[6,9,5,2,4], and construct the first part of the tree T starting from vertex 1, ending in vertex 12, and following the numbers in π. The rest of the tree is constructed by creating the corresponding edges (i, c_i) for i which are not in the list π, in this case the vertices 7,3,11,10 and 8.

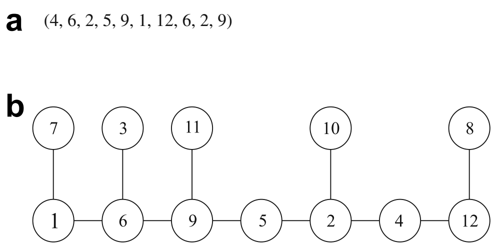

Fig. 2. (a) Example of a Dandelion code; (b) The final tree after the decoding process.

3.2. Initialization of the evolutionary algorithm

In order to initialize the algorithm, we start from the EW solution, given by the algorithm in [7]. Recall that this ensures a feasible solution for the CMST. The next step is to convert this EW tree to its corresponding Dandelion code, following the tree-to-code procedure described in [12]:

- **Input:** The EW solution (from the algorithm in [7]) for the CMST problem.
- **Output:** A Dandelion code $C_{EW} = (c_2, c_3, \cdots, c_{n-1})$ corresponding to the EW solution.
- **Step 1:** Find the unique path from one to n in T, and let π be the ordered list of intermediate vertices.
- **Step 2:** Recover the cycles $\{Z_i\}$ by writing π in a left-to-right list, and closing the cycle immediately to the right of each right-to-left minimum.
- **Step 3:** The Dandelion code corresponding with T is the unique code $C_{EW} = (c_2, c_3, \cdots, c_{n-1})$ such that: a) the cycles of the function $\phi_C(i) = c_i$ are $\{Z_i\}$, where $\phi_C(i)$ stands for the value of the ith position in string C_{EW}; b) for each $i \in [2, n - 1]$ which does not occur in π the first vertex on the path from vertex i to vertex n in the tree T is c_i.

Using this code C_{EW}, we generate the initial population of our Dandelion-based evolutionary algorithm by applying mutations to C_{EW} (we call C'_{EW} to a given mutation of C_{EW}), in such a way that $C'_{EW}(i) \in [2, n - 1]$. The delay constraint to obtain a solution to the DC-CMST is tackled by the objective function of the algorithm, as we will show later.

3.3. Evolutionary operators

Our evolutionary algorithm is structured in the traditional form of a classic genetic algorithm [16], with procedures of Selection, Crossover and Mutation. The Selection procedure is the standard roulette wheel, in which the probability of survival of a given individual is proportional to its fitness value. A two-points Crossover operator is applied, where the parents are randomly chosen among the individuals in the population. The Mutation operator changes specific values of C, substituting its value by a different one in [2, $n - 1$]. Regarding the parameters of the algorithm, we have implemented the standard values of Crossover ($P_c = 0.6$) and Mutation ($P_m = 0.01$) operators. The population size is 100 and the generations number has been fixed to 1000, after which we keep the best tree encountered so far.

In addition to the standard operators, we include a local search consisting in applying the Prim's algorithm [17] over each sub-tree of the individuals, in every generation of the algorithm. This makes that each sub-tree in the network is a MST. Since the objective function consists of a sum involving the distances between nodes belonging to the same sub-tree, it is expected that this procedure improves the quality of the final solution of the algorithm.

3.4. Constraints requirements of the DC-CMST and objective function

In [1] is shown that if we consider a restriction to the DC-CMST as:

$$\min\left(D_{\text{cost}} = \sum_{k=1}^{n} d \cdot C_k \cdot d_k\right) \tag{16}$$

subject to:

$$\frac{1}{v}\sum_{k=1}^{n}\left(\frac{\lambda_k}{\mu \cdot C_k - \lambda_k}\right) = \Delta \tag{17}$$

values of C_k can be calculated as

$$C_k = \frac{\lambda_k}{\mu}\left(1 + \frac{\sum_{j=1}^{n}\sqrt{d \cdot \lambda_j \cdot d_j}}{v \cdot \Delta \cdot \sqrt{d \cdot \lambda_k \cdot d_k}}\right), \tag{18}$$

note that if we use these values of C_k, we ensure that the generated tree always fulfils Constraints (5) and (6), and we only have to manage Constraint (7) in our evolutionary algorithm.

The objective function of our Dandelion evolutionary algorithm is given next. First we define the following parameter:

$$\mathscr{I} = \prod_{j}\left(\frac{1}{\varepsilon}\sum_{i \in R_j} q_i \cdot x_{ij}\right), \tag{19}$$

where j is the set of values such that $\sum_{i \in R_j} q_i \cdot x_{ij} > \varepsilon$. Note that this parameter measures the excess of traffic in sub-tree R_j. The idea is to include it in the objective function, in order to correct the traffic excess in sub-trees by means the evolution process. The objective function considered for the DC-CMST is then:

$$f(T) = \sum_{k=1}^{n} d \cdot C_k \cdot d_k \cdot \mathscr{I} \tag{20}$$

3.5. Tackling the source node optimization

The Dandelion-encoded evolutionary algorithm proposed can be adapted to solve the DC-CMST with source node optimization, by including the root node into the algorithm's encoding. In this case, a given individual of the algorithm can be encoded as $\mathbf{x}=[C,r]$, where C is a dandelion code representing the tree, and $r \in [0,n]$ is a number representing the node which must be used as source node. This way we avoid having to launch the evolutionary algorithm n times, once per possible root node. Of course, the search space in which the evolutionary must search for is larger now than in the traditional DC-CMST.

4. Experiments and results

In order to test the performance of our proposal, we have done a computational experience similar to the one shown in [1]. For comparison purposes we have implemented the LC algorithm following its description in [1], and we compare our results with that of this approach. A set of 15 DC-CMST instances have been constructed, in networks of 30, 50 and 70 nodes (5 networks of each case), following the instance construction procedure given in [1].

Our first experimental analysis consists of comparing the Dandelion EA proposed in this paper with the LC algorithm in the traditional DC-CMST. Table 1 reports the details of this comparison. Note that the LC algorithm is deterministic, so one value for each instance is obtained. On the other hand, our EA algorithm has been launched 30 times, and we provide the best, mean and standard deviation values for each instance. Our EA has been run using a population of 100 individuals during 1000 generations.

The results showed in Table 1 confirm the good performance of our approach. The Dandelion-encoded EA is able to obtain better results than the LC algorithm in all cases considered. In all the instances tackled, the best value found by our Dandelion-encoded EA is better than the result obtained by the LC algorithm. Moreover, we find that in all instances considered but one (instance 15), the mean value of the 30 runs with the EA is better than the result obtained by the LC algorithm. The differences between the proposed EA and the LC algorithm are larger as the instances' size grows. This means that our EA is more scalable than the LC algorithm. Fig. 3 shows the evolution of the best run for DC-CMST instance #1. Note that the algorithm's evolution has the form of steps, with flat zones between improvements. This behavior is consistent with the algorithm proposed: recall that we include a MST local search in each sub-branch of the tree, so small improvements in the objective function (associated with small changes in sub-

Table 1
Results of the Dandelion-encoded EA proposed, and comparison with the results obtained by the LC algorithm in [1].

Instance #	Size (nodes)	LC algorithm in [1]	Dandelion EA		
			Best	Mean	Std. Dev.
1	30	2180.86	2139.52	2162.58	15.65
2	30	1926.01	1751.49	1772.10	20.66
3	30	1855.45	1671.56	1721.30	32.31
4	30	2147.29	2086.38	2120.09	20.87
5	30	1844.00	1787.00	1787.24	0.76
6	50	3136.46	3015.61	3085.15	40.73
7	50	3550.55	3449.32	3503.89	22.47
8	50	2956.75	2709.80	2797.22	63.07
9	50	2488.86	2414.22	2452.38	16.11
10	50	2479.62	2299.15	2356.25	51.82
11	70	5403.27	5293.44	5320.54	14.77
12	70	3899.74	3693.69	3736.63	24.94
13	70	4960.57	4800.59	4847.00	33.23
14	70	3842.11	3639.38	3695.26	55.49
15	70	5826.47	5707.86	5835.26	68.71

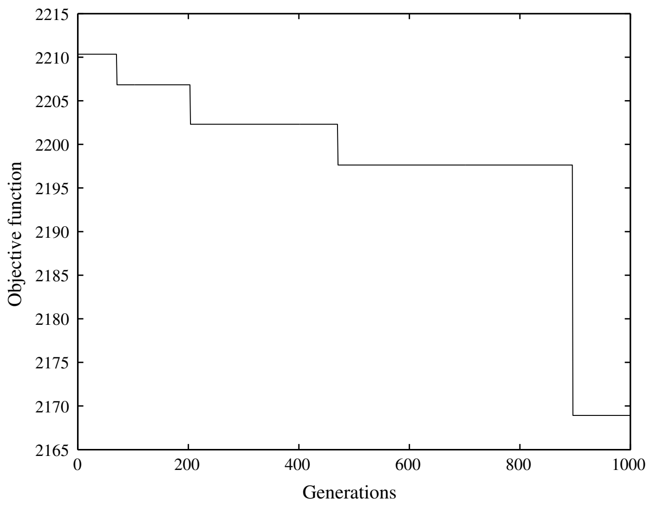

Fig. 3. Evolution of the EA in DC-CMST instance #1.

Table 2
Results of the Dandelion-encoded EA with source node optimization (sno). A comparison with the Dandelion-encoded EA without sno optimization is provided.

Instance #	Dandelion EA (sno) Best	Dandelion EA (sno) Mean	Dandelion EA Best	Dandelion EA Mean
1	2000.50	2019.03	2139.52	2162.58
2	1445.07	1511.57	1751.49	1772.10
3	1527.69	1557.42	1671.56	1721.30
4	1523.44	1561.22	2086.38	2120.09
5	1758.50	1771.94	1787.00	1787.24
6	2956.55	2999.85	3015.61	3085.15
7	2511.06	2603.81	3449.32	3503.89
8	2530.11	2558.49	2709.80	2797.22
9	2153.20	2216.13	2414.22	2452.38
10	2138.65	2204.32	2299.15	2356.25
11	3983.96	4073.21	5293.44	5320.54
12	3352.02	3415.24	3693.69	3736.63
13	4088.08	4141.13	4800.59	4847.00
14	3483.83	3629.22	3639.38	3695.26
15	4097.51	4271.95	5707.86	5835.26

branches) are discarded. The improvements are obtained when individuals with a better structure appear, which will be quite different from existing individuals.

The second round of experiments performed consists of solving the DC-CMST with source node optimization. Using the 15 DC-CMST instances considered before, we apply the Dandelion EA modified to tackled this problem, and compare the results with the proposed EA algorithm without source node optimization (sno). Table 2 shows these results. As expected, the optimization of the source node within the algorithm provides better networks in terms of the objective function given by Eq. (20). The computation time of the EA algorithm with sno optimization is similar to the EA without sno, since both algorithms are launched with the same number of individuals in the population and same number of generations. Note that the alternative to the EA with sno (i.e. launching one EA for each possible source node) is computationally infeasible in the majority of cases, and the EA with sno is a good alternative.

5. Conclusions

In this paper we have presented a Dandelion-encoded evolutionary algorithm for the Delay-Constrained Capacitated Minimum Spanning Tree (DC-CMST) problem. This problem arises in the topological design of communication networks, and is complicated because it considers optimization of the network topology in terms of the traffic capacity and its mean time delay. Due to this complexity, only heuristic approaches have been applied to this problem. Specifically, the best algorithm to solve the DC-CMST is known as the Least-Cost heuristic (LC). In the paper we show that the Dandelion-encoded evolutionary algorithm we propose is able to improve the results of the LC heuristic in several DC-CMST problem instances.

References

[1] Y.J. Lee, M. Atiquzzaman, Least cost heuristic for the delay-constrained capacitated minimum spanning tree problem, Comput. Commun. 28 (2005) 1371–1379.
[2] Y.J. Lee and M. Atiquzzaman, Exact algorithm for delay-constrained capacitated minimum spanning tree network, IET Communications 1(6) 1238-1247.
[3] T. Thomadsen, J. Larsen, A hub location problem with fully interconnected backbone and access networks, Comput. Operations Res. 34 (2007) 2520–2531.
[4] I. Astic, O. Festor, A hierarchical topology discovery sevice for IPv6 networks, in: Proceedings of the IEEE/IFIP Network Operations and Management Symposium, 2002, pp. 497–510.
[5] Y. Bejerano, M. Breitbart, R. Rastogi, Physical topology discovery for large multi subnet networks, in: Proceedings of IEEE INFOCOM'03, 2003, pp. 342–352.
[6] B. Gavish, Parallel savings heuristic for the topological design of local access tree networks, in: Proceedings of IEEE INFOCOM'03, 1986, pp. 130–139.
[7] L. Esau, K. Williams, On teleprocessing system design, part II, IBM Syst. J. 3 (1996) 142–147.
[8] K.M. Chandy, T. Lo, The capacitated minimum spanning tree, Networks 3 (1973) 173–181.
[9] C. Papadimitriou, The complexity of the capacitated minimum spanning tree problem, Networks 8 (1978) 217–230.
[10] A. Karaman, H. Hassanein, DCMC–delay constrained multipoint communication with multiple sources, in: Proceedings of the IEEE International Symposium on computers and Communications, 2003.
[11] D.S. Reeves, H.F. Salama, A distributed algorithm for delay-constrained unicast routing, IEEE/ACM Trans. Netw. 8 (2) (2000) 239–250.
[12] T. Paulden, D.K. Smith, From the Dandelion code to the rainbow code: a class of bijective spanning tree representations with linear complexity and bounded locality, IEEE Trans. Evol. Comput. 10 (2) (2006) 108–123.
[13] E. Thompson, T. Paulden, D.K. Smith, The Dandelion code: a new coding of spanning trees for genetic algorithms, IEEE Trans. Evol. Comput. 11 (1) (2007) 91–100.
[14] N. Deo, P. Micikevicius, Prüfer-like codes for labeled trees, Congressus Numerantium 151 (2001) 65–73.
[15] S. Piccioto, How to encode a tree, Ph.D. dissertation, Univ. California, San Diego, 1999.
[16] [16] D. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, MA, 1989.
[17] R.C. Prim, Shortest connection networks and some generalisations, Bell Syst. Tech. J. 36 (1957) 1389–1401.