

This is the peer reviewed version of the following article:

A divide and conquer matheuristic algorithm for the Prize-collecting Steiner Tree Problem / Akhmedov, Murodzhon; Kwee, Ivo; Montemanni, Roberto. - In: COMPUTERS & OPERATIONS RESEARCH. - ISSN 0305-0548. - 70:(2016), pp. 18-25. [10.1016/j.cor.2015.12.015]

*Terms of use:*

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

25/04/2024 06:05

(Article begins on next page)

# Author's Accepted Manuscript

A Divide and Conquer matheuristic algorithm for the prize-collecting Steiner tree problem

Murodzhon Akhmedov, Ivo Kwee, Roberto Montemanni



[www.elsevier.com/locate/caor](http://www.elsevier.com/locate/caor)

PII: S0305-0548(15)00301-9  
DOI: <http://dx.doi.org/10.1016/j.cor.2015.12.015>  
Reference: CAOR3906

To appear in: *Computers and Operation Research*

Received date: 12 June 2015  
Revised date: 5 December 2015  
Accepted date: 8 December 2015

Cite this article as: Murodzhon Akhmedov, Ivo Kwee and Roberto Montemanni A Divide and Conquer matheuristic algorithm for the prize-collecting Steiner tree problem, *Computers and Operation Research* <http://dx.doi.org/10.1016/j.cor.2015.12.015>

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting galley proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain

# A Divide and Conquer matheuristic algorithm for the prize-collecting Steiner tree problem

Murodzhon Akhmedov<sup>a,b</sup>, Ivo Kwee<sup>a,b</sup>, Roberto Montemanni<sup>a</sup>

<sup>a</sup>*Dalle Molle Institute for Artificial Intelligence  
(IDSIA-USI/SUPSI), Galleria 2,  
6928 Manno, Switzerland*

<sup>b</sup>*Institute of Oncology Research (IOR)  
Via Vela 6, 6500 Bellinzona, Switzerland*

---

## Abstract

The Prize-collecting Steiner Tree Problem (PCSTP) is a well-known problem in graph theory and combinatorial optimization. It has been successfully applied to solve real problems such as fiber-optic and gas distribution networks design. In this work, we concentrate on its application in biology to perform a functional analysis of genes. It is common to analyze large networks in genomics to infer a hidden knowledge. Due to the NP-hard characteristics of the PCSTP, it is computationally costly, if possible, to achieve exact solutions for such huge instances. Therefore, there is a need for fast and efficient matheuristic algorithms to explore and understand the concealed information in huge biological graphs. In this study, we propose a matheuristic method based on clustering algorithm. The main target of the method is to scale up the applicability of the currently available exact methods to large graph instances, without losing too much on solution quality. The proposed matheuristic method is composed of a preprocessing procedures, a heuristic clustering algorithm and an exact solver for the PCSTP, applied on sub-graphs. We examine the performance of the proposed method on real-world benchmark instances from biology, and compare its results with those of the exact solver alone, without the heuristic clustering. We obtain solutions in shorter execution time and with negligible optimality

---

*Email addresses:* [murodzhon@idsia.ch](mailto:murodzhon@idsia.ch) (Murodzhon Akhmedov), [ivo.kwee@ior.iosl.ch](mailto:ivo.kwee@ior.iosl.ch) (Ivo Kwee), [roberto@idsia.ch](mailto:roberto@idsia.ch) (Roberto Montemanni)

gaps. This enables analyzing very large biological networks with the currently available exact solvers.

*Keywords:* Prize-collecting Steiner Tree Problem (PCSTP); Combinatorial Optimization (CO); Matheuristics (M); Genomics (G)

---

## 1. Introduction

The Steiner Tree Problem (STP) is a well-know problem in graph theory and there exist different variants of the problem in the literature. The prize-collecting Steiner Tree problem (PCSTP) is a generalization of the STP. Given an undirected graph  $G = (V, E)$ , where the vertices are labeled with prizes  $p_j \geq 0$  and the edges are labeled with costs  $c_e > 0$ , the goal is to identify a sub-network  $G' = (V', E')$  which is also a tree. The target is to minimize the sum of prizes of the vertices in  $V \setminus V'$  and to minimize the total cost of the edges in  $E'$ . Thus, the corresponding problem is equivalent to the minimization of the following objective function, also known as *Goemans-Williamson minimization* problem [16]:

$$GW(G') = \min \sum_{e \in E'} c_e + \sum_{v \notin V'} p_v \quad (1)$$

The STP and the PCSTP have a wide range of applications, mainly in the design of utilities such as fiber optic networks and district heating networks. Recently applications in biological networks have appeared [7]. The relation between the PCSTP and genomic networks has been identified and it gathered the attention of researchers in [4]. Vertices and edges in these networks are given a score by using gene expression profiling (GEP) experimental data. GEP [1] measures the activity levels of thousands of genes simultaneously, and therefore it is able to provide a global overview of biological functions at cellular level. Genes have defined procedures to produce messenger RNAs (mRNA) [2]. A gene is considered to be active at a time if it produces mRNA, and passive otherwise. Thus, GEP measures the activity of each gene at a given point in time, and this can be related to the concurrent mRNA activities. In a gene

interaction network, every vertex is a gene and every edge between two vertices  
25 represents the genetic interaction between the two corresponding genes. Based  
on GEP data, each vertex in the network is given a prize, which is the mean  
value of the differential expression of the corresponding gene [5]. The differential  
expression is the amount of change in expression level of a gene in two different  
experimental conditions, for instance, tumor tissue versus control tissue. Also,  
30 each edge in the network is given a cost, which corresponds to the pairwise  
correlation level of the expression values of the corresponding gene pair [5].  
After having evaluated all vertices and edges in the network, the PCTSP is  
then used to detect a relevant sub-network (tree). In biological terms, the  
sub-network retrieved by the PCSTP has an important meaning [4], because  
35 it corresponds to a portion of the interaction network where many genes are  
highly correlated in terms of their functions and play an important role in  
the regulations of biological processes. Thus, the PCTSP can help to detect  
connected neighborhoods in interaction networks where the genes belong to the  
same biological pathways [5]. Based on these considerations, we develop in this  
40 work computational approaches for identifying functions of proteins or genes  
in cancer genomics relying upon PCSTP solutions. To this end, we construct  
gene-gene interaction networks by gathering information from genomic data. We  
use raw microarray data, that are commonly used to quantify gene expressions  
from different samples of tumor tissues and the corresponding control tissues.  
45 Subsequently, we apply the PCSTP on the generated gene interaction networks  
to perform functional analysis.

In the application of PCSTP in genomics, it is possible to have very large  
networks. Nevertheless, PCSTP has NP-hard characteristics. Thus, it is com-  
putationally costly to find solutions for large instances in reasonable execution  
50 time. Taking this fact into account, one needs an efficient and fast matheuristic  
algorithms to compute the PCSTP solution for large networks. At the same  
time, we seek methods able to give high quality solutions for inferring the func-  
tional biology. Solution quality and running time speed of the algorithm must  
be considered simultaneously. In this study, we propose a matheuristic that

55 allows the current state-of-the-art PCSTP methods to efficiently scale up on large instances in biological applications. The matheuristic is composed of a preprocessing technique, a heuristic clustering algorithm and an exact solution method for PCSTP, applied on sub-graphs. In this work, we develop a solution strategy to efficiently solve the objective function in Eq. (1). The rest of the  
60 work is organized as follows: In the next section we describe related work in the literature. In Section 3, we present the proposed matheuristic. The experimental studies of matheuristic are reported in Section 4. Our concluding remarks and future work are delivered in the last section of the paper.

## 2. Related Work

65 The concept of the PCSTP was first introduced by Bienstock *et al* [6] and the authors contributed pioneering work by developing 3-approximation algorithm to this research area. Several exact solution methodologies were proposed by Ljubic *et al* [11] and [10]. In these studies, the authors formulated the problem by employing the mixed integer linear programming and a branch-and-cut algo-  
70 rithm was devised to solve the model. They have tested the performance of their methods on real-world benchmark instances. Besides, there are some very distinct studies in the stream of approximation and heuristic algorithms. Canuto *et al* [14] introduced a multi-start local search heuristic with perturbations. Goemans and Williamson [8] proposed a primal-dual  $2-1/(n-1)$  approximation  
75 algorithm with  $O(n^3 \log n)$  running time performance. Another matheuristic approach was presented in Klau *et al* [18], in which the authors studied an incorporation of integer programming into memetic algorithm. Lucena and Resende [9] proposed a method to obtain a lower bound on the problem using mixed integer linear programming.

80 Different versions of the PCSTP were analyzed by Johnson *et al* [16], and for the primal-dual 2-approximation algorithm a well-defined strong pruning rule was developed. Klau *et al* [19] studied the fractional prize-collecting Steiner tree problem on trees. They maximize the ratio of the vertex profits and the

edge costs plus a fixed cost. Another version of the PCSTP known as the *quota*  
85 *problem* was studied in Haouari and Chaouachi [20] and Haouari *et al* [20]. In  
this problem an additional quota constraint is inserted into the model to force  
the total prize of the nodes selected in the tree to be above a predetermined  
threshold  $Q$ . A further study on a *quota problem* was presented in Haouari *et al*  
[22] by considering penalties - different from vertex prizes - for vertices that are  
90 not covered by the output tree. Other related problems were finally discussed  
in Montemanni *et al* [23].

In the literature, the PCSTP has been successfully applied to similar bio-  
logical networks to perform functional analysis in [3, 4, 5]. The focus in these  
studies was to analyze the protein-protein interaction networks. As a result of  
95 these studies, the authors have detected unknown and unreported functions for  
some proteins. Moreover, the authors have justified their computational find-  
ings with experimental support from biological experiments. This suggests that  
the output of PCSTP is very promising and indeed valuable when it is applied  
to bio-genetical graphs.

100 Another PCSTP application was proposed in [17], in which a heuristic ap-  
proach was devised to handle huge biological networks. That implementation  
targeted large networks with a restricted number of vertices with prizes  $p_j > 0$ .  
In contrast, this study focuses on large biological networks with some different  
properties. It is common to encounter networks with these properties in ge-  
105 nomics [5]. Basically, all vertices of the network have positive prizes. Since the  
generated network is based on genomics, it is highly possible that every gene has  
non-zero mean value of differential expression. This implies that corresponding  
vertex has positive prize  $p_j > 0$ . Another difference of the targeted networks is  
the ratio of vertex-prizes to the edge-costs. This ratio usually is not extremely  
110 high. The final distinction is that  $p_{max} \ll D_{max}$ , where  $p_{max}$  is the maximum  
prize among all vertices and  $D_{max}$  is the maximum value in all-pairs shortest  
path matrix. Very similar PCSTP application to networks with same properties  
was proposed in [30]. The main difference is that a *single-commodity flow* mixed  
integer linear programming (MILP) formulation of the PCSTP was considered

115 there, where in this study we adopt a more efficient cut-based formulation. This  
 study aims to use the state-of-the-art formulation.

### 3. Methodology

In this section, we present our matheuristic algorithm for PSCTP. The ap-  
 proach is composed of three distinct components: a preprocessing procedure,  
 120 a heuristic clustering algorithm, and any available exact method to solve the  
 PCSTP separately on each cluster.

It is important to note that any exact state-of-the-art solution approach  
 can be incorporated into the proposed matheuristic. In this study, we use the  
 method proposed in Ljubic *et al* [10].

125 The main idea of the proposed matheuristic approach is to heuristically  
 divide the giant graph into smaller graphs that can be solved separately, before  
 merging back the results. Solving a single large problem instance of PCSTP is  
 potentially computationally costly compared to solving possibly many smaller  
 problem components of that instance. Therefore, the “divide and conquer”  
 130 paradigm is adopted while designing the matheuristic.

#### 3.1. Preprocessing

The main target of preprocessing is to reduce the size of given network in-  
 stance by removing the vertices and edges with some specific properties in the  
 graph. The overall preprocessing we adopt after some preliminary experiments  
 135 is composed of two basic procedures, that are available in the literature [11].  
 The main reason for choosing these procedures is they are simple and compu-  
 tationally efficient. There are some other more sophisticated procedures in the  
 literature, which require the calculation of all-pairs shortest paths repeatedly  
 [10]. Since this study focuses on large graphs, computing all-pairs shortest paths  
 140 continuously could be computationally expensive. We repeat these procedures  
 until they fail to reduce the graph size further.

**Degree-One Test** : Consider a vertex  $i$  that has only one edge, to vertex  
 $j$ . If  $p_i < c_{ij}$ , then the vertex  $i$  is removed together with its edge. If  $p_i > c_{ij}$ ,



then the prize of vertex  $j$  becomes  $p_j = p_j + p_i - c_{ij}$ , and vertex  $i$  is removed  
 145 together with its edge.

**Degree-Two Test** : Consider a vertex  $i$  which has only two edges, to  
 vertices  $j$  and  $k$  respectively. If there is no edge between vertices  $j$  and  $k$ , an  
 edge is added with cost  $c_{jk} = c_{ij} + c_{ik} - p_i$ . In case there already exists such an  
 edge, then the cost of the edge is set to  $\min(c_{jk}, c_{ij} + c_{ik} - p_i)$ . Finally, vertex  
 150  $i$  is removed from the graph together with its edges.

### 3.2. A Divide and Conquer Heuristic Clustering

The proposed heuristic clustering algorithm separates input graph  $G$  into  
 independent portions of smaller graphs  $G_s$  according to minimization objective  
 function (1). At first, the algorithm groups vertices into clusters. Two vertices  
 155 are clustered together if the shortest path distance between them is smaller than  
 the prize of one of the two vertices. Subsequently, smaller graphs are constructed  
 by considering all vertices belonging to same clusters. The heuristic clustering  
 uses the pre-calculated all-pairs shortest path matrix obtained by Johnson's  
 algorithm [16].

160 The performance of heuristic clustering is highly dependent on the structure  
 of input graph. The targeted graph instances in this study have particular  
 characteristics, as discussed in section 2. These make the clustering algorithm  
 practically suitable to be used on them.

The pseudocode of the heuristic clustering approach is provided in Algorithm  
 165 1. Before starting clustering, the algorithm computes all-pairs shortest path  
 distance matrix  $D$  of graph  $G$ , where  $D_{ij}$  is the shortest path distance between  
 the vertices  $i$  and  $j$ . The clustering of vertices is simply based on the comparison  
 of the shortest path distances against prizes. If the shortest path distance  
 between two vertices is smaller than the prize of one of these vertices, then those  
 170 two vertices are clustered together. More formally, vertices  $i$  and  $j$  are clustered  
 together if they satisfy the *clustering condition*, which is  $D_{ij} < p_i$  or  $D_{ij} < p_j$ .  
 The heuristic starts by arbitrarily selecting a vertex  $i$  and permanently labeling  
 it with cluster identifier  $ID$ . Then, all the other vertices are analyzed one by

one with respect to vertex  $i$ , and the vertices satisfying the *clustering condition*  
 175 are temporarily labeled with the same cluster identifier  $ID$  of vertex  $i$ , and put  
 into set  $K$ . Afterward, one randomly selected vertex in  $K$  is removed from  
 that set and permanently labeled with the cluster identifier  $ID$  of vertex  $i$ . The  
 selected vertex is then analyzed with respect to the remaining unlabeled vertices  
 according to the *clustering condition*.” The vertices satisfying that condition  
 180 are temporarily labeled with the cluster identifier  $ID$  of vertex  $i$ , and inserted  
 into  $K$ . The algorithm constructs the first cluster when the set of temporarily  
 labeled nodes  $K$  gets empty. At this stage, the input graph  $G$  is composed  
 of unlabeled and permanently labeled vertices. Then, the heuristic arbitrarily  
 selects one vertex from unlabeled vertices, and repeats the same procedures,  
 185 not considering at this stage the permanently labeled vertices. The algorithm  
 terminates when all the nodes are permanently labeled. Clusters are obtained  
 by grouping together vertices with the same cluster identifier.

The application of Algorithm 1 on an example graph is shown in Figure 1.  
 In this figure, permanently labeled vertices are colored in red, and temporarily  
 190 labeled vertices are colored in yellow. The algorithm computes all-pairs short-  
 est path distances in the initialization phase. These distances are shown on the  
 edges of the graph. The directed arcs show the steps of the algorithm at each  
 iteration in the figure. Note that the algorithm stores each temporary labeled  
 vertex in set  $K$ , and removes the vertex from that set when it is labeled as per-  
 195 manent. In the first iteration, the heuristic starts by arbitrarily selecting vertex  
 $a$  and permanently labels it, then it analyzes all the other vertices according  
 to the *clustering condition*. Vertex  $b$  satisfies this condition, so it is labeled as  
 temporary. Then, the heuristic selects vertex  $b$  and analyzes other unlabeled  
 nodes. Vertex  $c$  satisfies the *clustering condition*, therefore it is temporarily  
 200 labeled. In the next step, the algorithm selects vertex  $c$ , permanently labels it,  
 and investigates other unlabeled nodes. At this step, there does not exist any  
 vertex satisfying the *clustering condition* and set  $K$  is empty, so the first iter-  
 ation of the algorithm is terminated. The heuristic determines the first cluster  
 $G_1$  at the end of the first iteration. It is worthy to mention that the set of tem-

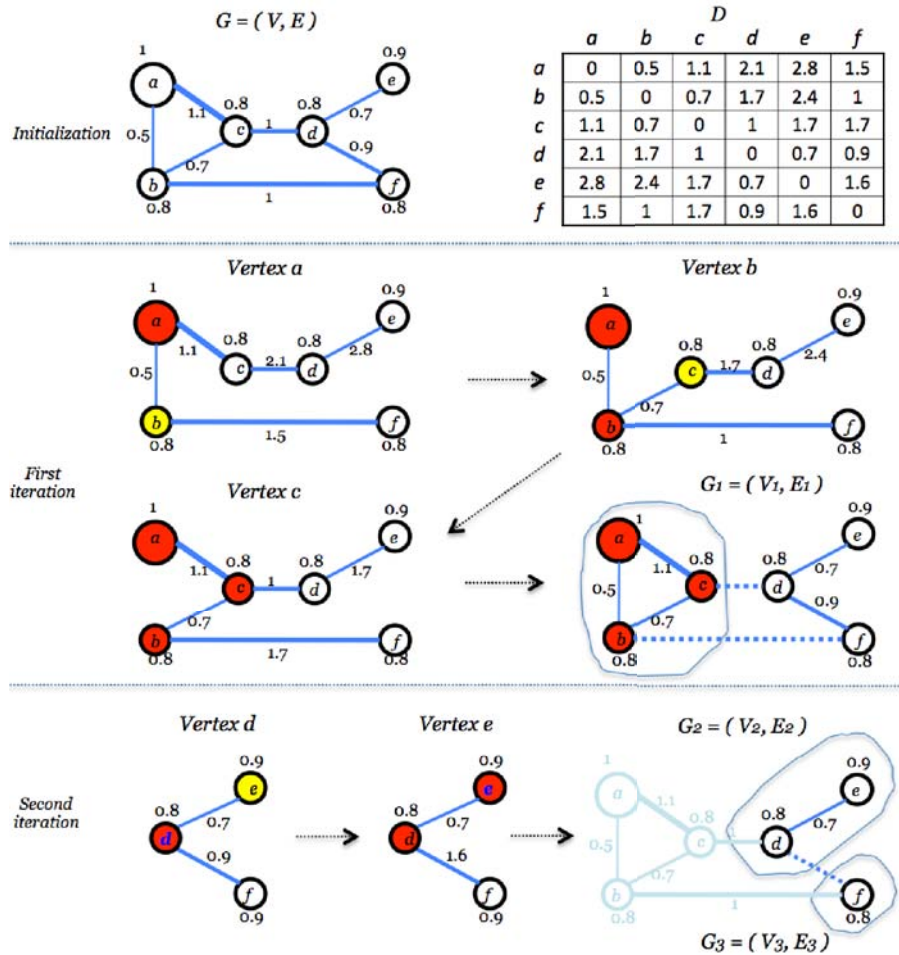


Figure 1: Example of application of the clustering algorithm. *Initialization*. All-pairs shortest path distance matrix  $D$  is calculated for given graph  $G$ . *First iteration*. The shortest path distances from the selected vertex to all the other vertices are represented on the edges. Permanently labeled vertices are colored in red, temporarily labeled vertices are colored in yellow. The heuristic obtains the first cluster  $G_1$ . *Second iteration*. The same procedure is repeated, and the algorithm obtains clusters  $G_2$  and  $G_3$ .

---

**Algorithm 1** : Heuristic Clustering Algorithm
 

---

**Require:** undirected graph  $G=(V, E)$ ; all-pairs shortest path matrix  $D$  of  $G$ ;

set  $S, K$ ; vector  $C$ ; int  $clusterID$

**Initialize:**  $S \leftarrow$  all vertices with  $p > 0$ ,

$clusterID \leftarrow 0, K \leftarrow \emptyset$ ;

**while**  $S$  is not  $\emptyset$  **do**

$clusterID \leftarrow clusterID + 1$ ;

Randomly select and remove a vertex  $i$  from  $S, C_i \leftarrow clusterID$ ;

**for**  $j \leftarrow 1 : |V|$  **do**

**if**  $(D_{ij} < p_i \parallel D_{ij} < p_j)$  and  $C_j = 0$  **then**

$C_j \leftarrow clusterID$  ;

**if**  $p_j > 0$  **then**

$K \leftarrow j$  and delete  $j$  from  $S$ ;

**end if**

**end if**

**end for**

**while**  $K$  is not  $\emptyset$  **do**

Randomly select and remove a vertex  $l$  from  $K, C_l \leftarrow clusterID$ ;

**for**  $m \leftarrow 1 : |V|$  **do**

**if**  $(D_{lm} < p_l \parallel D_{lm} < p_m)$  and  $C_m = 0$  **then**

$C_m \leftarrow clusterID$  ;

**if**  $p_m > 0$  **then**

$K \leftarrow m$  and delete  $m$  from  $S$ ;

**end if**

**end if**

**end for**

**end while**

**end while**

---

205 porary labeled vertices  $K$  only contains a single vertex at each step; however, it is not necessarily the case in general. Afterwards, the algorithm disregards the permanently labeled vertices from computation, and continues with the second iteration by arbitrarily selecting vertex  $d$  from the rest of the vertices, followed by vertex  $e$  later on. The heuristic performs the same procedures and obtains  
 210 the clusters  $G_2$  and  $G_3$  at the end of the second iteration.

The worst-case performance of the heuristic clustering is  $O(|V|^4)$  and the proof is straightforward.

### 3.3. Solving the PCSTP on the clusters

In the literature, there exist several MILPs and different techniques to solve  
 215 the PCSTP to optimality [11, 13]. The method proposed by Ljubic *et al* [10] is the most efficient one available to the best of our knowledge. In this study, we employ this tool to solve the sub-graphs generated by means of the clustering algorithm. The approach of [10] is based on a MILP and branch-and-cut. In the following paragraphs, we provide our MILP formulation that is slightly different  
 220 from [10]. These differences mainly arise within the constraints of the model due to the properties of the targeted networks in this study.

The PCSTP sub-network  $G_s = (V_s, E_s, c_s, p_s)$  is transformed into a Steiner arborescence instance  $G'_s = (V'_s, E'_s, c'_s)$ , where  $V'_s$  is formed by combining  $V_s$  with artificial root vertex  $r$ . For every undirected edge  $(i, j) \in E$ , two directed  
 225 edges  $(i, j)$  and  $(j, i)$  are added to  $E'_s$  with the costs  $c'_{ij} = c_{ij} - p_j$  and  $c'_{ji} = c_{ji} - p_i$ , respectively, where  $c_{ij}$  is equal to  $c_{ji}$ . In addition, for every vertex in  $V$ , an edge  $(r, j)$  is added to  $E'_s$  from root vertex  $r$  with corresponding cost of  $-p_j$ . This transformation is illustrated with an example in Figure 2. The subsequent goal is to find a final tree subgraph  $T = (V_t, E_t)$  for the transformed Steiner  
 230 arborescence instance  $G'_s$ .

The following model uses decision variables  $x_{ij} \in \{0, 1\}$  and  $y_i \in \{0, 1\}$  which are defined as follows:

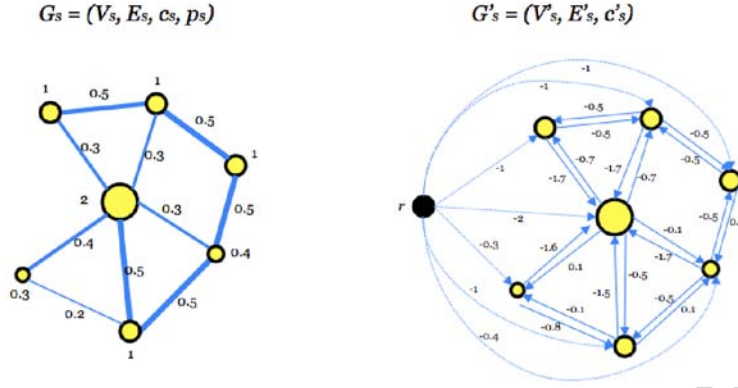


Figure 2: The transformation of the PCSTP instance into the Steiner Arborescence instance.

$$x_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E_t, \\ 0 & \text{otherwise} \end{cases} \quad \forall (i, j) \in E'_s$$

$$y_i = \begin{cases} 1 & \text{if } (i) \in V_t, \\ 0 & \text{otherwise} \end{cases} \quad \forall (i) \in V'_s$$

240

The following notation is also employed:  $S \subset V'_s$  is a set of vertices, and  $S' = V'_s/S$  is the complement of that set. The  $\delta^+(S) = \{(i, j) | i \in S, j \in S'\}$  and  $\delta^-(S) = \{(i, j) | i \in S', j \in S\}$  are two set of directed cuts induced by  $S$  and  $S'$ , and  $x(A) = \sum_{ij \in A} x_{ij}$ .

245

It should be noted that the PCSTP has a feasible solution only if the root vertex has a single outgoing edge. Accordingly, the mathematical model for the problem defined on the Steiner aborescence is [11]:

$$\text{Minimize } \sum_{ij \in E'_s} c'_{ij} x_{ij} + \sum_{i \in V'_s} p_i \quad (2)$$

$$\text{s. t. } \sum_{j \in E'_s} x_{ij} = y_i \quad \forall i \in V'_s \setminus \{r\} \quad (3)$$

$$x(\delta^-(S)) \geq y_i \quad \forall i \in V'_s \quad (4)$$

250

$$y_r = 1 \quad (5)$$

$$\sum_{ri \in E'_s} x_{ri} = 1 \quad (6)$$

$$x_{rj} \leq 1 - y_i \quad \forall i < j, i \in V'_s \quad (7)$$

$$x_{ij} + x_{ji} \leq y_i \quad \forall i \in V'_s \setminus r \quad (8)$$

$$x_{ij} \in \{0, 1\}, y_i \in \{0, 1\} \quad \forall i \in V'_s \setminus \{r\}, (i, j) \in E'_s \quad (9)$$

255

The objective function (2) minimizes the sum of total edge costs in the solution tree and the constant term. The role of the constant term in the objective function is for obtaining a consistent solution cost with respect to original PCSTP instance. The constraint set (3) enforces every selected vertex to have one directed incoming edge. Each vertex in a solution tree must have a directed path from artificial root vertex and this condition is ensured by the constraint set (4). Constraint (5) maintains the root vertex to be included into final tree and constraint (6) ensures the root to have a single outgoing edge. The following constraints are added in order to strengthen the model further. The constraint set (7) creates a bijection between the solutions of PCSTP and arborescence problem, and it eliminates a huge number of solutions for PCSTP that correspond to same solution for arborescence problem. The constraint set (8) forces the model to select one edge within the pair of directed edges between two vertices. Finally, the constraint set (9) ensures that the decision variables are binary. The interested reader may refer to [10] for further details and explanation about the solution approach.

270

## 4. Experimental Results

The matheuristic approach is evaluated on two different problem sets and is compared to the exact method described in [10] (used already as an inner solver  
 275 by our approach) on those problem sets. We have re-implemented the whole approach discussed in [10]. In order to justify the correctness and performance of our re-implementation, we have tested it as a preliminary step.

### 4.1. Results of our re-implementation of the exact approach [10]

We have re-implemented the approach discussed in [10] in C++ and have  
 280 employed the Boost Graph Library [27] for handling large graphs efficiently. IBM ILOG CPLEX 12.6 [28] was used as a MILP solver. The computational studies have been performed on a server equipped with an Intel(R) Xeon(R) CPU E5320 1.86GHz processors and 32 GB of shared memory. A single core was used for the experiments. On the other hand, the experimental results  
 285 of [10] had been obtained on a Pentium IV 2.8 GHz computer with 2GB of memory, and by employing version 8.1 of ILOG CPLEX.

By employing our re-implementation, we have been able to reproduce the same results for all the instances considered in [10]. For illustrative purposes, we report the results for the Cologne I instances in Table 1, where a comparison  
 290 with findings reported in [10] is provided. The first column of the table contains instance names. The second and the third columns report the size of the instances. The “DHEA [10] time(s)” column corresponds to the running time of approach reported in [10]. The “DHEA time(s)” column contains the running time of our re-implementation of the same approach.

295 It is difficult to compare the running times of the approaches fairly due to the usage of different machines, solvers and tools in the implementation. The machines are comparable with a little disadvantage for our computer [26]. Taking this fact into account, we can argue that our re-implementation is a little bit advantageous in running time. However, this certainly depend on the  
 300 upgraded version of CPLEX software and the usage of most recent libraries for graphs.



Table 1: Results obtained by our re-implementation of the exact approach [10].

Instance	V	E	DHEA [10] time(s)	DHEA time(s)
i01M1	768	69077	2.9	3.0
i01M2	768	69077	487.8	167.3
i01M3	768	69077	1195.8	694.1
i02M1	769	69140	2.9	3.9
i02M2	769	69140	598.2	306.5
i02M3	769	69140	1810.9	813
i03M1	771	69100	3.1	5.0
i03M2	771	69100	326.8	110.3
i03M3	771	69100	755.9	603.9
i04M1	761	68907	2.8	1.3
i04M2	761	68907	22.6	9.3
i04M3	761	68907	77.7	26.2
i05M1	761	68934	2.8	1.0
i05M2	761	68934	122.9	60.6
i05M3	761	68934	399.4	325

#### 4.2. Testing the matheuristic approach on real-life biological instances

The first set of instances used to evaluate the performance of our matheuristic method is composed of real-world biological benchmark instances from the literature. These medium sized protein-protein interaction graphs are taken from Dittrich *et al* [5] to perform computational analysis. These network instances provide an important information for lymphoma cancer and the detailed description about them can be found in [5].

The second set of instances has been generated by us based on gene expression profiling data of lymphoma cancer patients available in Gene Expression Omnibus repository<sup>1</sup>. In particular, this study focuses on Diffuse Large B-Cell Lymphoma (DLBCL) cancer. There exist two DLBCL cancer types which are an activated B cell (ABC) and the germinal center B cell (GCB). The target is to identify signatures that are the group of genes relevant for cancer and have discriminative effects in type classification. In order to achieve this goal,

<sup>1</sup><http://www.ncbi.nlm.nih.gov/geo/>

Table 2: Preprocessing results of Dittrich *et al* [5] test instances.

Instance	V	E	V'	E'	Preprocessing time(s)
HCMV	3863	29293	3102	28331	4.62
met_mic1	3523	4345	1788	2731	1.54
met_mic2	3514	4332	1472	2402	1.94
met_mic3	2853	3335	881	1479	1.35
lymphoma	2034	7756	1563	7285	1.75
dros.001	5226	93394	4412	92580	32.07
dros.005	5226	93394	4412	92580	31.63
dros.0075	5226	93394	4412	92580	31.69

we have constructed gene-gene interaction graphs, where vertex  $j$  represents gene  $j$ . The node prize is associated with the differential expression value of two cancer types for each gene  $p_j = |E_{ABC} - E_{GCB}|$ , where  $E_{ABC}$  and  $E_{GCB}$  are equivalent to the mean of gene expression values of ABC and GCB cancer patients for the corresponding gene, respectively. Edge-cost  $c_{ij}$  is associated with the correlation value  $r_{ij}$  of gene expression values between two genes  $i$  and  $j$ . Each edge is added to the network if the corresponding correlation value is bigger than some threshold value. We have chosen the threshold values as  $r_{ij} = \{0.5, 0.6, 0.7\}$ . These instance are bigger than the previous ones in the field, and they consist of graphs with upto 21049 vertices and 293113 edges. It is common to encounter networks with such sizes in genomics. These instances can be made available to other researchers upon request.

We have applied the preprocessing techniques described in Section 3.1 on each test instance before feeding it to the algorithms. Table 2 summarizes the preprocessing results for the test instances discussed in [5]. The first column corresponds to the name of instances. The second and third columns report the original size of input graphs. The fourth and fifth columns show the size of the residual graphs after preprocessing. The last column contains the execution time of the preprocessing procedures.

Table 3 reports the computational results for the instances described in [5]. The first and second columns contain instance names and the objective value

Table 3: Results of Dittrich *et al* [5] test instances with and without preprocessing.

Instance	With Preprocessing					Without Preprocessing			
	OPT	DHEA(s)	MATH(s)	Clusters	OPT-Gap(%)	DHEA(s)	MATH(s)	Clusters	OPT-Gap(%)
HCMV	7371.53	85.01	13.26	2429	0.000	154.94	12.18	2791	0.000
met_mic1	11346.93	26.42	15.12	944	0.000	42.79	21.02	1749	0.000
met_mic2	16250.24	14.81	4.96	900	0.424	22.07	3.91	1238	0.424
met_mic3	16919.62	3.96	1.95	420	0.022	7.78	2.05	713	0.182
lymphoma	3341.89	3.44	3.03	950	0.000	2.85	2.64	1395	0.000
dros_001	8273.98	28136.72	7756.16	3143	0.056	30144.22	12058.76	3940	0.056
dros_005	8121.31	38278.76	34838.19	999	0.000	20130.25	29386.68	1567	0.000
dros_0075	8039.86	12725.58	9530.6	750	0.000	8036.81	10546.51	4277	0.000

of each optimal solution. We provide the results obtained by the proposed matheuristic approach both with and without preprocessing. The “DHEA(s)” column contains the running time data of the approach discussed in [10], and it also includes the preprocessing time (if applied) in order to have a fair comparison baseline. The “MATH(s)” column reports the execution time of the matheuristic clustering approach (embedding the exact approach in [10] as an inner solver). The number of clusters obtained by applying the procedure described in Section 3.2 is indicated in the “Clusters” column for each instance. The number of clusters also includes the clusters consisting of just a single vertex. The execution time for the matheuristic includes the preprocessing time (if applied), the time consumed by the heuristic clustering algorithm, and the total time needed to solve all the clusters with the method described in [10]. The “OPT-Gap(%)” column indicates the optimality gap percentage between the solutions provided by the matheuristic method and the exact approach [10].

The results show that the matheuristic method with preprocessing is able to find solutions in shorter running times than DHEA. On the other hand, optimality gaps are negligible, especially for bioinformatics practitioners. The contribution of the preprocessing procedures to the total execution time data of DHEA is favorable for all the instances except for the last two cases, where it is advantageous not to apply preprocessing for faster running times. For some instances the matheuristic method without preprocessing is faster compared to

Table 4: Presprocessing results of DLBCL test instances

Instance	V	E	V'	E'	Preprocessing time(s)
GSE4732	2407	24392	1385	23370	5.67
GSE4475	13211	81023	4753	72565	162.39
GSE22470	13211	129103	5665	121557	234.90
GSE10172	13211	191646	13202	191637	2.99
GSE10846	21049	293113	3576	275640	1185.53
GSE19246	21049	126208	2567	107726	532.50
GSE31312	21049	166598	4800	150335	649.78
GSE23501	21049	142301	3535	124787	703.94
GSE19246g	42450	1001206	8070	966826	8163.01
GSE10846g	42450	1046485	12601	1016636	8289.28
GSE23501g	42450	1112354	14565	1084469	7617.63

the variant with preprocessing. Preprocessing is therefore not always convenient. Overall optimality gaps are lower than 1 % for all test cases.

360 Table 4 summarizes the results of the preprocessing procedures on the second set of instances, , while Table 5 reports the complete results, using the same format as in Table 3. For the first 8 instances of Table 5, a maximum computation time of 50000 seconds is imposed. The results show that the matheuristic clustering, both with and without preprocessing, is significantly  
365 faster than standard DHEA. For DHEA, contribution of the preprocessing to total running time is significant on these large graphs and without preprocessing it could not solve three instances within the time or memory limits. For the last 3 instances, we set the computational time limit to 100000 seconds since the instances are much larger. DHEA was not able to solve these large instances,  
370 either with or without preprocessing, due to time or memory limits. In these cases, apart from GSE23501g (where an upper bound of 12929.78 – anyway worse than that provided by the matheuristic method – was found), DHEA was not able to retrieve a feasible solution, while lower bounds were provided. These lower bounds, together with the cost of the heuristic solutions provided by the  
375 matheuristic approach are provided in column “OPT” for these instances. It is however important to observe that the gap between the available lower and

Table 5: Results of DLBCL test instances with and without preprocessing.

Instance	With Preprocessing					Without Preprocessing			
	OPT	DHEA(s)	MATH(s)	Clusters	OPT-Gap(%)	DHEA(s)	MATH(s)	Clusters	OPT-Gap(%)
GSE4732	398.86	134.80	8.18	1241	0.839	253.44	6.25	2263	0.839
GSE4475	988.25	16344.49	218.52	4583	0.000	28855.08	231.73	13041	0.000
GSE22470	1006.45	42549.55	342.52	5504	0.009	**	330.15	13050	0.009
GSE10172	354.82	30280.01	541.85	12375	0.108	**	545.26	12384	0.108
GSE10846	1480.23	7021.58	1660.88	3052	0.070	18237.53	1481.27	20507	0.070
GSE19246	1641.12	5801.55	2803.58	734	0.008	*	4163.46	19216	0.008
GSE31312	3350.63	8312.06	2332.31	3659	0.022	24221.80	6165.62	19464	0.022
GSE23501	980.08	15301.89	844.08	3250	0.038	49239.31	692.40	20695	0.038
GSE19246g	[9812.04; 9815.79]	*	27125	5016	$\leq 0.038$	**	75790	40102	$\leq 0.038$
GSE10846g	[8914.83; 8946.50]	*	24117	9055	$\leq 0.355$	**	70033	39061	$\leq 0.355$
GSE23501g	[12789.95; 12817.36]	**	61612	1245	$\leq 0.214$	**	97106	40089	$\leq 0.214$

\*\* Insufficient memory

\* CPLEX run out of time

upper bounds is always below 0.4%).

Interestingly, the matheuristic method with and without preprocessing found the same solutions for all test cases. For some instances the matheuristic method without preprocessing is remarkably faster. An intuitive explanation can be the following one: graphs associated with biological networks present the properties described in Section 2, and on those graphs the clustering algorithm (that is significantly effective) tends to mimic the preprocessing itself, which is therefore automatically incorporated into the main algorithm.

Finally, it is worth noting that optimality gaps for the matheuristic algorithm are always below 1%, in spite of faster execution times. Moreover, the matheuristic method was able to provide solutions for 2 (large) instances, while the exact solver was not.

## 5. Conclusions and Future Work

We have proposed a matheuristic approach for solving the PCSTP based on a clustering algorithm. The main purpose of the method is to scale up the applicability of the currently available exact methods to large instances, without losing too much on solution quality. The proposed approach has been tested on

real-world instances constructed by combining protein-protein interaction net-  
works with data obtained from micro-array technology. In addition, we have  
395 also tested the overall matheuristic approach on networks generated from real  
lymphoma data obtained again by micro-array technology. The performance  
of the matheuristic approach has been compared with the state-of-the-art to  
show the effectiveness of the heuristic clustering phase. On large graphs, the  
400 matheuristic method has obtained solutions in shorter execution time with neg-  
ligible optimality gaps.

In this study, Ljubic *et al* [10] exact approach has been used as an inner  
solver for the matheuristic method. For example, a recent study [29] reports a  
new algorithm, which can be seen as an updated version of previous approach.  
405 As a future work, we are interested in testing such an updated method as an  
inner solver of the proposed matheuristic method.

### Acknowledgment

The authors would like to thank Martin Luipersbeck and Ivana Ljubic for  
the useful discussions on their approaches and the area editor together with  
410 the reviewers for their valuable suggestions. M. A. is supported by Swiss Na-  
tional Science Foundation through project 205321-147138/1: “Steiner trees for  
functional analysis in cancer system biology”.

### References

- [1] A. L. Tarca, R. Romero, and S. Draghici. Analysis of microarray experiments  
415 of gene expression profiling. *American journal of obstetrics and gynecology*,  
195(2):373–388, 2006.
- [2] M. B. Gerstein, C. Bruce, J. S. Rozowsky, D. Zheng, J. Du, J. O. Korbelt,  
O. Emanuelsson, Z. D. Zhang, S. Weissman, and M. Snyder. What is a  
gene, post-ENCODE? History and updated definition. *Genome Research*,  
420 17: 669–681, 2007.

- [3] N. Tuncbag, S. McCallum, S. C. Huang, and E. Fraenkel. SteinerNet: a web server for integrating omic data to discover hidden components of response pathways. *Nucleic Acids Research*, 40(Web Server issue):W505–9, 2012.
- [4] M. B. Becheta, C. Borgsb, A. Braunsteinc, J. Chayesb, A. Dagkessaman-  
425 skaiad, J. M. Franoisd, and R. Zecchina. Finding undetected protein associations in cell signaling by belief propagation. *PNAS*, 108:882–887, 2010.
- [5] M. T. Dittrich, G. W. Klau, A. Rosenwald, T. Dandekar, and T. Mueller. Identifying functional modules in protein-protein interaction networks: an integrated exact approach. *Bioinformatics*, 26:223–231, 2008.
- 430 [6] D. Bienstock, M. X. Goemans, D. Simchi–Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59:413–420, 1993.
- [7] N. Tuncbag, A. Braunstein, A. Pagnani, S. C. Huang, J. Chayes, C. Borgs, R. Zecchina, and E. Fraenkel. Simultaneous reconstruction of multiple sig-  
435 naling pathways via the prize–collecting Steiner forest problem. *Journal of Computational Biology*, 20(2):124–136, 2013.
- [8] M. X. Goemans, and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems, *D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems*,  
440 144–191,1997.
- [9] A. Lucena, and M. G. C. Resende. Strong lower bounds for the prize–collecting Steiner problem in graphs. *Discrete Applied Mathematics*, 141:277–294, 2004.
- [10] I. Ljubic, R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel,  
445 and M. Fischetti. An algorithmic framework for the exact solution of the prize–collecting Steiner tree problem. *Mathematical Programming*, 105(2):427–449, 2006.

- [11] I. Ljubic, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. Solving the prize–collecting Steiner tree problem to optimality. *Proceedings of ALENEX, Seventh Workshop on Algorithm Engineering and Experiments*, 6876, 2005.
- [12] I. Ljubic. Exact and Memetic Algorithms for Two Network Design Problems. PhD thesis, Faculty of Computer Science, Vienna University of Technology, Chapter 4:91–151, 2004.
- 450 [13] M. Haouari, S. B. Layeb, and H. D. Sherali. Strength of three MIP formulations for the prize collecting Steiner tree problem with a quota constraint. *Electronic Notes in Discrete Mathematics*, 36:495–5021, 2010.
- [14] S. A. Canuto, M. G. C. Resende, and C. C. Ribeiro. Local search with perturbation for the prize- collecting Steiner tree problem in graphs. *Networks*, 460 38:50–58, 2001.
- [15] J. E. Beasley. OR–Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 1069–1072, 1990
- [16] D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting Steiner tree problem: theory and practice. *Proc. 11th ACM–SIAM Symp. on Discrete* 465 *Algorithms*, 760–769, 2000
- [17] M. Akhmedov, I. Kwee and R. Montemanni. A Fast Heuristic for the Prize-collecting Steiner Tree Problem. *Lecture Notes in Management Science*, 6:207–216, 2014.
- [18] G. W. Klau, I. Ljubic, A. Moser, P. Mutzel, P. Neuner, U. Pferschy, G. 470 Raidl, and R. Weiskircher. Combining a memetic algorithm with integer programming to solve the prize-collecting Steiner tree problem. *Genetic and Evolutionary Computation GECCO 2004*, 3102:1304–1315, 2004.
- [19] G. W. Klau, I. Ljubic, P. Mutzel, U. Pferschy, and R. Weiskircher. The fractional prize–collecting Steiner tree problem on trees. *Extended Abstract, ESA 2003*, 691–702, 2003. 475



- [20] M. Haouari, and J. Chaouachi. A hybrid Lagrangian genetic algorithm for the prize-collecting Steiner tree problem. *Computers and Operations Research*, 33:1274–1288, 2006.
- [21] M. Haouari, S. Layeb, and H.D. Sherali. The prize collecting Steiner tree  
480 problem: Models and Lagrangian dual optimization approaches. *Computational Optimization and Applications*, 40:13–39, 2008.
- [22] M. Haouari, S. Layeb, and H.D. Sherali. Algorithmic expedients for the Prize Collecting Steiner Tree Problem. *Discrete Optimization*, 7:32–47, 2010.
- 485 [23] R. Montemanni, V. Leggieri and C. Triki. Mixed integer formulations for the probabilistic minimum energy broadcast problem in wireless networks. *European Journal of Operational Research*, 190(2):578-585, 2008.
- [24] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- 490 [25] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 1389–1401, 1957.
- [26] J. J. Dongarra. Performance of various computers using standard linear equations software. Technical report CS–89–85, University of Manchester, 2014.
- 495 [27] J. Siek, L.Q. Lee and A. Lumsdaine. Boost Graph Library, 2000, <http://www.boost.org>
- [28] IBM ILOG CPLEX Optimization Studio, <http://www.cplex.com>
- [29] M. Fischetti, M. Leitner, I. Ljubic, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, and M. Sinnl, 2014 December, Thinning out Steiner trees: a  
500 node–based model for uniform edge costs. *Mathematical Programming Computations*, 2015. Special Issue: 11th DIMACS Implementation Challenge on Steiner Tree Problems, submitted.

- [30] M. Akhmedov, I. Kwee and R. Montemanni. A Matheuristic Algorithm for the Prize–collecting Steiner Tree Problem. *The 3<sup>rd</sup> International Conference on Information and Communication Technology IEEE*, 411–415, 2015.

505

Accepted manuscript