On the Clustered Shortest-Path Tree Problem
(Short Communication)

Mattia D'Emidio[1], Luca Forlizzi[2], Daniele Frigioni[2], Stefano Leucci[3], Guido Proietti[2,4]

[1] Gran Sasso Science Institute (GSSI), Viale F. Crispi 7, I–67100 L'Aquila, Italy.
mattia.demidio@gssi.infn.it
[2] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, Università degli Studi dell'Aquila, Via Vetoio, I–67100 L'Aquila, Italy.
{luca.forlizzi,daniele.frigioni,guido.proietti}@univaq.it
[3] Dipartimento di Informatica "Sapienza" Università di Roma, Viale R. Elena 295b, I–00161 Roma, Italy. leucci@di.uniroma1.it
[4] Istituto di Analisi dei Sistemi e Informatica "Antonio Ruberti", Consiglio Nazionale delle Ricerche, Via dei Taurini 19, I–00185 Roma, Italy.

Abstract. Given an n-vertex and m-edge non-negatively real-weighted graph $G = (V, E, w)$, whose vertices are partitioned into a set of k clusters, a *clustered network design problem* on G consists of finding a (possibly optimal) solution to a given network design problem on G, subject to some additional constraint on its clusters. In this paper, we focus on the classic *shortest-path tree* problem and summarize our ongoing work in this field. In particular, we analyze the hardness of a clustered version of the problem in which the additional feasibility constraint consists of forcing each cluster to form a (connected) subtree.

1 Introduction

In several network applications, the underlying set of nodes may be partitioned into *clusters*, with the intent of modeling some aggregation phenomena taking place among similar entities in the network. In particular, this is especially true in communication and social networks, where clusters may refer to local-area subnetworks and to communities of individuals, respectively. While on one hand the provision of clusters allows to represent the complexity of reality, on the other hand it may ask for introducing some additional constraints on a feasible solution to a given network design problem, with the goal of preserving a specific cluster-based property. Thus, on a theoretical side, given a vertex-partitioned input (possibly weighted) graph G, a *clustered* (a.k.a. *generalized*) *network design problem* on G consists of finding a (possibly optimal) solution to a given network design problem on G, subject to some additional constraint on its clusters.

V. Biló, A. Caruso (Eds.): ICTCS 2016, Proceedings of the 17th Italian Conference on Theoretical Computer Science, 73100 Lecce, Italy, September 7–9 2016, pp. 263–268 published in CEUR Workshop Proceedins Vol-1720 at http://ceur-ws.org/Vol-1720

One of the most intuitive constraint one could imagine is that of maintaining some sort of *proximity* relationship among nodes in a same cluster. This scenario has immediate practical motivations: for instance, in a communication network, this can be convincingly justified with the requirement of designing a network on a classic two-layer (i.e., local *versus* global layer) topology. In particular, if the foreseen solution should consist of a (spanning) tree T in G, then a natural setting is that of forcing each cluster to induce a (connected) subtree of T. For the sake of simplicity, in the following this will be referred to as a *clustered tree design problem* (CTDP), even if this is a slight abuse in the nomenclature. As a consequence, classic tree-based problems on graphs can be revisited under this new perspective, and while some of them do not actually exhibit, from a computational point of view, a significant misbehavior w.r.t. the ordinary (i.e., non-clustered) counterpart (for instance, the *minimum spanning tree* (MST) problem falls in this category, since we can easily solve its clustered version by first computing a MST of each cluster, then contracting these MSTs each to a vertex, and finally finding a MST of the resulting graph), some other will actually become much more complex, as it is the case for the problem of our interest in this paper, namely the *single-source shortest-path tree* (SPT) problem.

Related Work. Several classic tree/path-based problems have already been investigated in the framework of CTDPs. For instance, we refer the reader: (i) to [1,4] for studies that have focused on the clustered *traveling salesperson problem*; (ii) to [2,6] for works that have dealt with the clustered version of the *minimum Steiner tree problem*. Moreover, we mention a study that has tackled the clustered variant of the *minimum routing-cost spanning tree problem* [5], where the authors also present an inapproximability result for the clustered *shortest path problem*, which was in fact inspiring our present study. Finally, we refer the reader to the paper by Feremans *et al.* [3], where the authors review several classic network design problems in a clustered perspective, but with different side constraints on the clusters.

Our Contribution. In this paper, we focus on the clustered version of the SPT (say CluSPT in the following), and on its unweighted variant (say CluBFS in the following, where BFS refers to the *breadth-first search tree*). It is worth noticing that an SPT supports a set of communication primitives of primary importance, as for instance the broadcasting and the spanning tree protocol, and that in a non-clustered setting it can be computed in almost linear time by means of the classic Dijkstra's algorithm. Nevertheless, to the best of our knowledge nothing is known about its clustered variant, despite the fact that it is very reasonable to imagine a scenario where the aforementioned primitives are required to be applied locally and hierarchically within each cluster. In this work, we then try to fill this gap, by providing a set of results which allow to shed light on its computational complexity.

Graph Notation. Throughout the paper, we use the following graph notation. Let $G = (V, E, w)$ denote a generic *weighted* undirected graph with $|V| = n$ vertices

and $|E| = m$ edges, where $w : E \to \mathbb{R}_{\geq 0}$ is a weight function, associated with the graph, such that each edge $e = (u, v) \in E$ has a non-negative weight $w(e)$. We denote by $N(u)$ the set of neighbors of u in G, i.e., $N(u) = \{v \in V \mid (u, v) \in E\}$. Let $d_G(u, v)$ denote the *distance* between vertices u and v in G, that is the *length* of a shortest path $P_G(u, v)$ between u and v in G, which is given by the sum of the weights of the edges in $P_G(u, v)$. For a given spanning tree T of G, $d_T()$ will denote the corresponding distance function on T. Given a subset of vertices $S \subseteq V$ of G, we denote by $G[S]$ the subgraph of G induced by S. Finally, we denote by $V(G)$ and $E(G)$ the set of vertices and edges of G when we need to emphasize the dependence on the graph.

2 CluBFS

In this section, we formally introduce the CluBFS problem and then give our main results about it. The problem is defined as follows.

CluBFS	
***Input*:**	An unweighted undirected graph $G = (V, E)$, whose set of vertices is partitioned into a set of k (pairwise disjoint) clusters $\mathcal{V} = \{V_1, V_2, \ldots, V_k\}$, a distinguished source vertex $s \in V$.
***Solution*:**	A clustered BFS tree of G rooted at s, i.e., a spanning subgraph T of G such that: (i) T is a spanning tree of G rooted at s; (ii) for each $V_i \in \mathcal{V}$, $T[V_i]$ is connected.
***Measure*:**	The cost of T, i.e., $\text{COST}(T) = \sum_{v \in V} d_T(s, v)$.

In other words, a clustered BFS tree is a spanning tree T of G such that each subgraph $T_i = T[V_i]$ is connected and the sum of the hop distances in T from the source s towards all the other vertices is minimized. By a reduction from the NP-complete 3–CNF–SAT problem, we are able to prove the following result:

Theorem 1. CluBFS *is NP-hard.*

2.1 An approximation algorithm for CluBFS

In this subsection, we provide an approximation algorithm for the CluBFS problem. The main idea of the algorithm is that of minimizing the number of distinct clusters that must be traversed by any path from s to a vertex $v \in V$. If all the clusters are of low diameter then this leads to a good approximation for CluBFS. If at least one cluster has large diameter then it is possible to show that the optimal solution must be expensive and hence any solution for CluBFS will provide the sought approximation.

W.l.o.g., let V_1 be the cluster containing vertex s. The algorithm first considers each cluster $V_i \in \mathcal{V}$ and identifies all the vertices belonging to V_i into a single vertex ν_i, so as to obtain a graph G' in which (i) each vertex corresponds to a cluster and (ii) there is an edge (ν_i, ν_j) between two vertices in G' iff the set $E_{i,j} = \{(v_i, v_j) \in E(G) : v_i \in V_i \wedge v_j \in V_j\}$ is not empty. It then computes a

BFS tree T' of G' rooted at ν_1 and constructs the sought approximate solution $\widetilde{T}$ as follows: initially $\widetilde{T}$ contains all the vertices of G and the edges of a BFS tree of $G[V_1]$ rooted at s; then, for each edge (ν_i, ν_j) of T' where ν_i is the parent of ν_j in T', it adds to $\widetilde{T}$ a single edge $(v_i, v_j) \in E_{i,j}$ along with all the edges of a BFS tree of $G[V_j]$ rooted at v_j.

The analysis of the above algorithm allows us to prove the following result:

Theorem 2. *There exists a polynomial-time $O(n^{\frac{2}{3}})$-approximation algorithm for* CluBFS.

While CluBFS thus admits an $o(n)$-approximation algorithm, interestingly we proved (by a reduction from the NP-complete Exact–Cover–by–3-Sets problem) that the clustered *single-source to single-destination* shortest-path problem (on unweighted graphs) cannot be approximated in polynomial time within a factor of $n^{1-\epsilon}$, for any constant $\epsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$. This extends the inapproximability result (within any polynomial factor) that was given in [5] for the corresponding weighted version. Thus, establishing an $o(n^{2/3})$-inapproximability of CluBFS is a problem that we leave open.

2.2 Fixed-Parameter Tractability Results for CluBFS

In this subsection, we prove that CluBFS is fixed-parameter tractable w.r.t. two natural parameters by providing two different FPT algorithms. The notion of *fixed-parameter tractability* relaxes the classical notion of polynomial-time tractability, by admitting algorithms whose running time is exponential, but only in terms of some parameter of the problem instance that can be expected to be small in typical applications.

In the first algorithm, we choose as our first "natural" parameter the number of clusters of $\mathcal{V}$. Notice that every solution T for CluBFS induces a *cluster-tree* $\widetilde{T}$ obtained from T by identifying the vertices belonging to the same cluster. The algorithm first guesses the cluster-tree $\widetilde{T}^*$ of an optimal solution T^*, and then it reconstructs T^* by using a dynamic programming approach. Notice that our first FPT algorithm is efficient when the number of clusters of the CluBFS instance is small. On the other hand, the classical BFS tree problem can be seen as a special instance of CluBFS where $\mathcal{V} = \{\{v\} : v \in V\}$, i.e., each cluster contains only one vertex. This problem can clearly be solved in polynomial time, but the complexity of the above algorithm becomes super-exponential! This suggests that, for the case in which $\mathcal{V}$ consists of many singleton clusters, there must be another parametrization yielding a better complexity. Following this observation, we are able to develop another FPT algorithm parameterized in the total number of vertices, say h, that belong to clusters of size at least two. The idea of the algorithm is that of guessing the *cluster-root* r_i of each cluster $V_i \in \mathcal{V}$, i.e., a vertex of V_i closer to the source s in an optimal solution T^* to the CluBFS instance. It can be shown that $T^*[V_i]$ must be a BFS tree of $G[V_i]$ rooted at r_i, and this, along with the knowledge of the cluster-roots, allows us to efficiently reconstruct the optimal tree T^*. Thus, overall, we can give the following result:

Theorem 3. CluBFS *can be solved in* $O(\min\{n\,k^{k-2}, h^h\}\cdot(m+n))$ *time and* $O(m)$ *space.*

3 CluSPT

In this section, we give our results on the CluSPT problem. Regarding the formal definition of CluSPT, it can be simply derived as the weighted version of CluBFS. In more details, the main differences w.r.t. CluBFS are then two: (i) the given graph $G = (V, E, w)$ is weighted by a weight function $w : E \to \mathbb{R}_{\geq 0}$; (ii) the measure that we are willing to minimize (i.e. the cost of T) is expressed in terms of distances (instead of hop distances, as in the unweighted case). In other words, in this case, a clustered SPT is a spanning tree T of G such that each subgraph $T_i = T[V_i]$ is connected, and the total length of all paths in T emanating from the source s is minimized. By elaborating on the reduction we used to prove the NP-hardness of CluBFS, we are able to prove the following:

Theorem 4. CluSPT *cannot be approximated, in polynomial time, within a factor of* $n^{1-\epsilon}$ *for any constant* $\epsilon \in (0, 1]$*, unless* $\mathsf{P} = \mathsf{NP}$*.*

The above result is easily seen to be (essentially) tight, since we can provide a simple $O(n)$-approximation algorithm, as follows. First it computes a multigraph G' from G by identifying each cluster $V_i \in \mathcal{V}$ into a single vertex ν_i. When doing this, it associates each edge of G' with the corresponding edge of G. Then it computes a *minimum spanning tree* (MST from now on) T' of G', and k MSTs $T_1, \ldots, T_k$ of $G[V_1], \ldots, G[V_k]$, respectively. Finally, the algorithm returns the spanning tree $\widetilde{T}$ of G which contains all the edges in $E' \cup \bigcup_{i=1}^{k} E(T_i)$, where E' denotes the set of edges of G associated with an edge in $E(T')$.

3.1 Fixed-Parameter Tractability Results for CluSPT

The two fixed-parameter algorithms for CluBFS, presented in Section 2.2, can be easily extended to CluSPT. In particular, the theorem below can be easily derived by Theorem 3 basically by replacing, in both the algorithms for CluBFS, the BFS algorithm with the Dijkstra's algorithm, when the part of the solution to the problem inside each cluster has to be computed.

Theorem 5. CluSPT *can be solved in* $O(\min\{n\,k^{k-2}, h^h\}\,(m+n\log n))$ *time and* $O(m)$ *space.*

References

1. Xiaoguang Bao and Zhaohui Liu. An improved approximation algorithm for the clustered traveling salesman problem. *Inf. Process. Lett.*, 112(23):908–910, 2012.
2. Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6, 2013.

3. Corinne Feremans, Martine Labbé, and Gilbert Laporte. Generalized network design problems. *European Journal of Operational Research*, 148(1):1–13, 2003.
4. Nili Guttmann-Beck, Refael Hassin, Samir Khuller, and Balaji Raghavachari. Approximation algorithms with bounded performance guarantees for the clustered traveling salesman problem. *Algorithmica*, 28(4):422–437, 2000.
5. Chen-Wan Lin and Bang Ye Wu. On the minimum routing cost clustered tree problem. *J. Comb. Optim.*, 31(1):1–16, 2016.
6. Bang Ye Wu and Chen-Wan Lin. On the clustered Steiner tree problem. *J. Comb. Optim.*, 30(2):370–386, 2015.