Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

RALF BORNDÖRFER
ZIENA ELIJAZYFER
STEPHAN SCHWARTZ

Approximating Balanced Graph Partitions

ZIB Report 19-25 (June 2019)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Approximating Balanced Graph Partitions

Ralf Borndörfer
Ziena Elijazyfer
Stephan Schwartz

Abstract

We consider the problem of partitioning a weighted graph into $k \in \mathbb{N}$ connected components of similar weight. In particular, we consider the two classical objectives to maximize the lightest part or to minimize the heaviest part. For a partitioning of the vertex set and for both objectives, we give the first known approximation results on general graphs. Specifically, we give a Δ-approximation where Δ is the maximum degree of an arbitrary spanning tree of the given graph. Concerning the edge partition case, we even obtain a 2-approximation for the min-max and the max-min problem, by using the claw-freeness of line graphs.

1 Introduction

Partitioning a graph into connected subgraphs is a problem arising in many application areas, such as parallel processing, road network decomposition, and image processing [LPS93; Möh+07; Bul+16].

The two fundamental ways to partition a graph are vertex partition and edge partition. The problem of partitioning the vertex set into $k \in \mathbb{N}$ connected components is called *connected-vertex-k-partition problem* (CVP_k). If k is part of the input, the problem is denoted by CVP. Analogously, the *connected-edge-k-partition problem* (CEP_k) as well as CEP are defined.

For balancing given node weights (or resp. edge weights) among the parts, the two classical objectives are to maximize the total weight of the minimum part or to minimize the weight of the maximum part. The CVP with one of these objectives is also known as the *balanced connected partition problem* (BCP). On general graphs both variants of CVP_k are NP-hard [CGM83], and to the best of our knowledge no approximation results are known unless for certain classes of graphs [PS81; Bec+01].

In this paper, we give the first approximation result for the CVP on general graphs. Our contribution is a Δ-approximation for the min-max CVP, where Δ is the maximum degree of an arbitrary spanning tree of the given graph. For the max-min CVP we

Table 1: Contributions of the present paper.

	min-max	max-min
CVP	Δ-approximation	Δ-approximation (if $w_{\max} \leq \frac{w(G)}{\Delta k}$)
CEP	2-approximation	2-approximation (if $w_{\max} \leq \frac{w(G)}{2k}$)

obtain the same result, but only for instances where the maximum node weight is bounded. The approach is based on an algorithm that partitions a tree into subtrees whose weight is bounded from below and above. Depending on the parameters with which this algorithm is called, it generates an approximate solution for the min-max CVP or for the max-min CVP.

We also prove that our algorithm can be extended to provide a 2-approximation, if the given graph is a line graph. We show that the maximum degree of a spanning tree for a line graph can be bounded by 3. By exploiting the fact that a connected edge partition on a graph is equivalent to a connected vertex partition on the corresponding line graph, we obtain a 2-approximation for both the min-max CEP and the max-min CEP. Again, for the latter, we impose a constraint on the maximum edge weight. While we have not found any approximation results for both versions of the CEP, the 2-approximation we achieve also follows from the work of Chu, Wu, and Chao [CWC13].

The rest of this paper is organized as follows. In Section 2 we formally describe the CVP and CEP and review related work. In Section 3 we give a generalized approximation scheme for the CVP, and present an algorithm to obtain a Δ-approximation for both versions of the CVP. Finally, in Section 4 we use line graphs to transform the CEP into a CVP. Using the claw-freeness of line graphs, we provide a 2-approximation for both versions of the CEP.

2 Problem formulation

Problem formulation All graphs considered in this paper are assumed to be connected. The main problem we consider is the *connected-vertex-partition problem (CVP)* which is defined as follows: Let $G = (V, E)$ be a graph with node weights $w : V \to \mathbb{R}_{>0}$ and let $k \in \mathbb{N}$. For $V' \subseteq V$ we define $w(V') := \sum_{v \in V'} w(v)$ and for a subgraph $S \subseteq G$ we write $w(S) := w(V(S))$. Furthermore, we denote by $w_{\max} := \max_{v \in V} w(v)$ the maximum node weight in G. The min-max CVP is to find a partition $(V_1, \dots, V_k)$ of V such that $G[V_i]$ is connected for $i \in [k]$ while minimizing $\max_{i \in [k]} w(V_i)$. Equivalently we are to find trees $T_1, \dots, T_k \subseteq G$ such that

$V = \dot{\bigcup}_{i\in[k]} V(T_i)$ and $\max_{i\in[k]} w(T_i)$ is minimized. The max-min CVP is defined analogously but with the objective to maximize $\min_{i\in[k]} w(T_i)$.

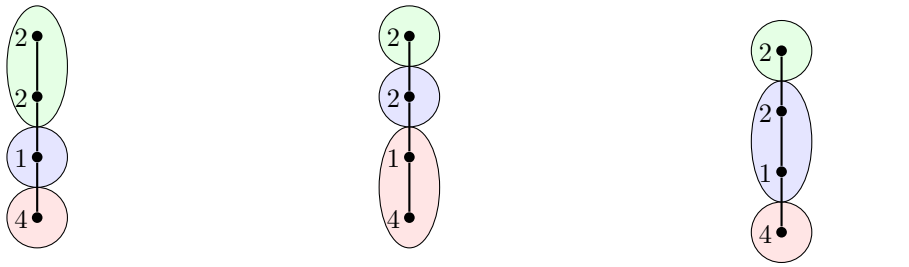
For the *connected-edge-partition problem (CEP)* we consider an edge-weighted graph and a positive integer k. We assume the weight function w to operate on the edge set E, and adopt the notation from the node case. The CEP is to find connected subgraphs $G_1, \dots, G_k \subseteq G$ such that $E = \dot{\bigcup}_{i\in[k]} E(G_i)$. Again, we aim to minimize $\max_{i\in[k]} w(G_i)$ for the min-max CEP, or respectively, to maximize $\min_{i\in[k]} w(G_i)$ for the max-min CEP.

In both versions of the CVP and CEP we explicitly allow empty sets (or empty graphs, respectively) in the partition. However, the objective function has a natural aversion towards empty parts, since for the max-min version this would lead to a worst possible objective value of 0, and for the min-max version one could potentially split the heaviest component to reduce the maximum weight.

Related work The literature on partition problems is vast. We are focussed on the aforementioned CVP and CEP with min-max or max-min objective.

The two versions of CVP are closely related. In fact, for CVP_2 the two versions obviously coincide. For larger k though, optimal solutions of the two problems *can* differ. A simple example for this is presented in Figure 1. While for this instance one can find a solution that simultaneously optimizes the min-max and max-min objective, there are also instances that do not admit such a solution (cf. [LPS93]).

Concerning the complexity, already CVP_2 is NP-hard [CGM83], even on unit-weighted bipartite graphs [DF85]. Moreover, it is also hard to approximate. Chataigner, Salgado, and Wakabayashi [CSW07] proved that CVP_2 does not admit a PTAS and that there is no α-approximation for CVP with $\alpha < \frac{6}{5}$, unless P = NP. Earlier, Chlebíková [Chl96] had given a $\frac{4}{3}$-approximation for CVP_2.



(a) optimal solution for min-max CVP_3 but not optimal for max-min CVP_3

(b) optimal solution for max-min CVP_3 but not optimal for min-max CVP_3

(c) optimal solution for min-max CVP_3 and max-min CVP_3

Figure 1: Comparison of solutions for min-max CVP_k and max-min CVP_k for an exemplary instance. The numbers indicate node weights.

Focussing on the max-min CVP, more approximation results are known. For CVP_3 and CVP_4 the best known result is a 2-approximation for 3-connected, or respectively 4-connected graphs [CSW07].

The max-min CVP for special graph classes allows for even better results. For instance, CVP_k is polynomially solvable for trees. For the max-min CVP_k this was proved in [PS81] and for the min-max CVP_k in [BPS80]. Earlier, Kundu and Misra [KM77] had considered a closely related problem and provided an algorithm that can be employed to solve the min-max CVP_k in polynomial time. In 1991, Frederickson [Fre91] gave linear-time algorithms for both problems. These are particularly important, since there are different heuristics that transform the original instance onto a tree to efficiently solve the problem there [CWC13; Zho+19].

Another graph class for which CVP was investigated are grid graphs. While it was shown that CVP_k is NP-hard for arbitrary grid graphs [Bec+98], the max-min CVP_k can be solved in polynomial time for ladders, i.e., grid graphs with two rows and an arbitrary number of columns [Bec+01].

For the class of series-parallel graphs, CVP_k is also NP-hard. Ito, Zhou, and Nishizeki [IZN06] give a pseudo-polynomial-time algorithm for both, the min-max and max-min CVP. They also show that their algorithm can be extended to graphs with bounded tree-width.

The min-max CVP has received less attention. It was studied by Zhou et al. [Zho+19] who give a mixed integer linear program to solve the problem using a flow formulation to ensure the connection of the subgraphs. Since this approach is only feasible for small instances, they also propose a genetic algorithm as a heuristic approach. The main idea is to find a proper spanning tree whose optimal partition will also be optimal for the given graph. While the authors of [Zho+19] perform a computational study to show that their heuristic performs well, there is no theoretical result that guarantees the quality of this solution.

The literature on CEP is also sparse. Jünger, Reinelt, and Pulleyblank [JRP85] were the first to consider a problem related to the min-max CEP with unit weights. They study the problem of partitioning the edge set into connected subgraphs of equal size s (except for the last subgraph which may have size $\leq s$).

The min-max CEP_k and max-min CEP_k with unit weights were studied by Wu et al. [Wu+07], who show that these problems are NP-hard. For trees with unit weights Chu et al. [Chu+10] give an algorithm that guarantees that the biggest tree has at most twice as many edges as the smallest tree.

Chu, Wu, and Chao [CWC13] then showed that this result carries over to a general graph G with non-negative integer edge weights, if $w_{\max} \leq \frac{w(G)}{2k}$. They present an algorithm to find a given number of connected subgraphs, such that the weight of the heaviest subgraph is at most twice as large as the weight of the lightest subgraph. While this is not mentioned by the authors, one can show that this result can be used to obtain a 2-approximation for the min-max CEP as well as the max-min CEP, if the edge weights of the respective instance satisfy the mentioned condition.

3 Approximating the CVP

In this section we will give an algorithm to obtain approximative solutions for the min-max CVP as well as for the max-min CVP. For the latter however, we need to assume an additional bound for the maximum node weight $w_{\max}$. Interestingly, we use the same algorithm for both problems, while we only adjust a single input parameter for the respective problem.

In particular, we give a Δ-approximation for both cases where Δ is the maximum degree of a spanning tree of the given graph. This means that if W is the objective value of our algorithm and W^* is the optimal objective value, we guarantee that $W \leq \Delta W^*$ for the min-max CVP, and $W \geq \frac{1}{\Delta} W^*$ for the max-min CVP. To the best of our knowledge this is the first approximation result for CVP on general graphs.

3.1 A general approximation scheme for CVP

We start with a more general approach for the approximation and consider connected partitions where the weight of every part is bounded from below and above (except for the last one, which is only bounded from above). This approach is similar to the one of Ito et al. [Ito+12], but their method is quite different and requires integer node weights. Specifically, we consider $[\lambda, c\lambda)$-partitions which are defined as follows.

Definition 1. *Let G be a graph with positive node weights w and let $\lambda > 0,\ c > 1$. A connected partition $T_1, \ldots, T_m$ of $V(G)$ is called* $[\lambda, c\lambda)$-partition of G *if*

i) $w(T_i) \in [\lambda, c\lambda) \quad \forall i \in [m-1]$,

ii) $w(T_m) \in (0, c\lambda)$.

We will see that for certain choices of λ, a $[\lambda, c\lambda)$-partition leads to a c-approximation for the CVP. The following theorem gives the details.

Theorem 2. *Let (G, w, k) be an instance of CVP and let $T_1, \ldots, T_m$ be a $[\lambda, c\lambda)$-partition of G.*

a) If $\lambda := \max\{w_{\max}, \frac{w(G)}{k}\}$, then $T_1, \ldots, T_m$ is a c-approximation for the min-max CVP on (G, w, k).

b) If $\lambda := \frac{w(G)}{ck}$, then $T_1, \ldots, T_m$ leads to a c-approximation for the max-min CVP on (G, w, k).

Proof.

a) Let $T_1^*, \ldots, T_k^*$ be an optimal partition for the min-max CVP on (G, w, k) and let $W^* := \max_{i \in [k]} w(T_i^*)$. Then $w(G) = \sum_{i \in [k]} w(T_i^*) \leq kW^*$ and hence $\lambda \leq W^*$ because $w_{\max} \leq W^*$ is obvious.

Note that $w(T_m) > 0$ and hence

$$w(G) \;=\; w(T_m) + \sum_{i=1}^{m-1} w(T_i) \;\geq\; w(T_m) + (m-1)\lambda \;>\; \frac{m-1}{k} w(G),$$

which implies that $m-1 < k$ or, equivalently, that $m \leq k$. As a result $T_1, \ldots, T_m$ is a feasible k-partition and $w(T_i) < c\lambda \leq cW^*$ for any $i \in [m]$, which completes the proof of the first statement.

b) Now let $T_1^*, \ldots, T_k^*$ be an optimal partition for the max-min CVP on (G, w, k) and let $W^* := \min_{i\in[k]} w(T_i^*)$. Then $w(G) = \sum_{i\in[k]} w(T_i^*) \geq kW^*$ and consequently $\lambda \geq \frac{1}{c}W^*$.

Since $w(T_i) < c\lambda$ for every $i \in [m]$, we know that $w(G) = \sum_{i\in[m]} w(T_i) < mc\lambda$, and hence $k = \frac{w(G)}{c\lambda} < m$.

We can easily relabel the trees $T_1, \ldots, T_{m-1}$ such that $\bigcup_{j\in\{k,\ldots,m\}} T_j$ is connected. Afterwards we define

$$S_i := \begin{cases} T_i & , i \in [k-1], \\ \bigcup_{j\in\{k,\ldots,m\}} T_j & , i = k. \end{cases}$$

From $w(T_i) \geq \lambda \; \forall i \in [m-1]$ we obtain $w(S_i) \geq \lambda \geq \frac{1}{c}W^*$ for arbitrary $i \in [k]$ and hence $S_1, \ldots, S_k$ is the sought approximation. □

3.2 A Δ-approximation for CVP

The approximation for the CVP instance (G, w, k) is performed in two steps. First we find a spanning tree T of G, specify a root node r, and denote the maximum degree of T by Δ. Then we use a partition algorithm which operates on a rooted tree with node weights to find a Δ-approximation.

While we discuss the choice of the spanning tree later, we are now concerned with the problem of partitioning a tree into connected parts of similar weight. Lukes [Luk74] was the first to consider partition problems on trees, but did not demand connectedness of the parts. Kundu and Misra [KM77] give an algorithm to partition a tree with node weights into subtrees of bounded total weight. Since then, various other approaches have been proposed, e.g. in [BPS80; Fre91; ABP93; Ito+12], but none of them were employed to construct approximation algorithms for the corresponding problem on general graphs.

In order to simplify the description of our algorithm, we start by introducing some notation concerning rooted trees. Let T^r be the tree rooted at node r. With $N^+(v)$ we denote the set of child nodes of the node v and $N^-(v)$ is the (unique) parent node of $v \neq r$ in T^r. For $v \in T^r$ the graph T_v^r is the subtree of T^r rooted at v. For instance, in Figure 2 we have $T_2 = T_e^a$. Note that both T_v^r and $T^r \setminus T_v^r$ are trees and that their respective node sets are disjoint, i.e., the edge between $v \neq r$ and its parent does neither belong to T_v^r nor to $T^r \setminus T_v^r$.

The tree partitioning routine is described in Algorithm 1 and Figure 2 illustrates how it works. We successively split off rooted subtrees from bottom to top, while manipulating the original tree T^r (cf. line 9). After considering all other nodes, we process the root node and set T_m to be the remaining tree T^r.

With the preceding observations, the correctness of Algorithm 1 is clear. Let us now briefly analyze its computational complexity. Breadth-first search on T^r runs in

Algorithm 1: BalancedTreePartition(T^r, w, λ)

Input: rooted tree T^r,
$w : V(T^r) \to \mathbb{R}_{>0}$,
$\lambda \geq w_{\max}$

Output: trees $T_1, \dots, T_m \subseteq T^r$ with $V(T^r) = \dot{\bigcup}_{i \in [m]} V(T_i)$

1 $\mathcal{T} \leftarrow \varnothing$; $\quad w_v^r \leftarrow 0 \quad \forall v \in V(T^r)$
2 $Q \leftarrow$ list of vertices of T^r in reversed BFS order
3 **for** $v \in Q$ **do**
4 **if** $v = r$ **then**
5 **return** $\mathcal{T} \cup T^r$
6 $w_v^r \leftarrow w(v) + \sum_{x \in N^+(v)} w_x^r$
7 **if** $w_v^r \geq \lambda$ **then**
8 $\mathcal{T} \leftarrow \mathcal{T} \cup T_v^r$
9 $T^r \leftarrow T^r \setminus T_v^r$

$\mathcal{O}(|V(T^r)|)$ and afterwards, we process every node in amortized constant time. Thus, our algorithm has a runtime linear in $|V(T)|$.

The following result is the cornerstone for the Δ-approximation.

Proposition 3. *Let T^r be a rooted tree with node weights w and let $\deg(r) < \Delta$, where Δ is the maximum degree of T^r. Then, for every $\lambda \geq w_{\max}$ the output $T_1, \dots, T_m$ of BalancedTreePartition(T^r, w, λ) is a $[\lambda, \Delta\lambda)$-partition of T^r.*

Proof. For some $i \in [m]$ consider the root v of T_i and observe that $w_{x_j}^r < \lambda$ for all child nodes $x_1, \dots, x_l$ of v in T_i. Due to the choice of r we have $l \leq \Delta - 1$ and hence

$$w(T_i) \;=\; w_v^r \;=\; w(v) + \sum_{j=1}^{l} w_{x_j}^r \;<\; w_{\max} + l\lambda \;\leq\; \Delta\lambda.$$

The fact that $w(T_i) \geq \lambda \;\; \forall i \in [m-1]$ comes from line 7 of the algorithm. □

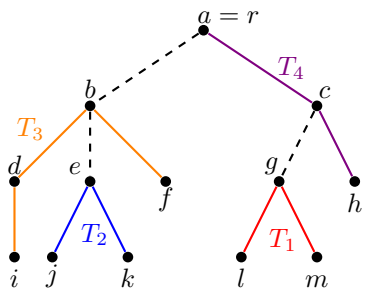


Figure 2: Balanced tree decomposition for an exemplary tree rooted at node a with unit weights and $\lambda = 3$.

Figure 3: Instance for min-max CVP where Algorithm 1 yields a worst-possible solution. For $k := |V|$ we obtain $\lambda = w_{\max} = 1 + \varepsilon$ and the algorithm produces the solution on the left with objective value $\Delta + \varepsilon$. On the right, the optimal solution with objective value $1 + \varepsilon$ is depicted.

By combining the results from Theorem 2 and Proposition 3, we obtain the following.

Corollary 4. *Let (G, w, k) be an instance of CVP and let T^r be a rooted spanning tree of G. Also assume that Δ is the maximum degree of T^r and that $\deg(r) < \Delta$. Then,*

a) for $\lambda := \max\{w_{\max}, \frac{w(G)}{k}\}$, the output of BalancedTreePartition(T^r, w, λ) is a Δ-approximation for the min-max CVP on (G, w, k).

b) if $\lambda := \frac{w(G)}{\alpha k} \geq w_{\max}$, BalancedTreePartition$(T^r, w, \lambda)$ leads to a Δ-approximation for the max-min CVP on (G, w, k).

In the following we will show that our algorithm does not admit a better approximation ratio. Figure 3 shows an example where the approximation bound for the min-max CVP is tight. Note however, that the example exploits that $m \ll k$ for the output of the algorithm, i.e., we only obtain two trees instead of $|V|$ trees. It remains open if a second stage optimization where we iteratively partition the heaviest tree can improve the theoretic bound further.

Concerning the max-min CVP, Figure 4 shows an instance where the algorithm yields a partition with $\min_{i \in [k]} w(T_i) = 1$ while $W^* = \Delta$. The example can easily be extended for arbitrary k.

Figure 4: Instance for max-min CVP where Algorithm 1 yields a worst-possible solution. For $k := 2$ and unit weights we obtain $\lambda = \frac{2\Delta}{2\Delta} = 1$ and the algorithm produces the solution on the left with objective value 1. On the right, the optimal solution with objective value Δ is depicted.

To end this section, we will briefly discuss the choice of the spanning tree. To obtain a best possible approximation result, we are interested in finding a spanning tree whose maximum degree is as small as possible. This problem is known as *minimum degree spanning tree problem*. While the problem is NP-hard, Fürer and Raghavachari [FR92] give a polynomial time algorithm to find a spanning tree of degree at most $\Delta^* + 1$, where Δ^* is the maximum degree of an optimal spanning tree. For special classes of graphs better results can be achieved. For instance, in the next section we will show that every line graph admits a spanning tree of degree at most 3 (cf. Lemma 5).

4 A 2-Approximation for CEP

In this section we present a 2-approximation algorithm for the min-max CEP as well as for the max-min CEP. We exploit the fact that a connected edge partition on a graph is equivalent to a connected vertex partition on the corresponding line graph. Recall that for an undirected graph $G = (V, E)$ the line graph $L(G)$ has a vertex for each edge $e \in E$ and two vertices in $L(G)$ are adjacent iff the corresponding edges are incident in G.

In 1970 Beinecke [Bei70] proved a characterization of line graphs in terms of forbidden subgraphs. One result is that a line graph cannot contain a claw, i.e., a $K_{1,3}$, as induced subgraph. This has an interesting implication concerning the minimum degree spanning tree of line graphs.

Proposition 5. *Every line graph has a spanning tree T with $\Delta(T) \leq 3$. Furthermore, every tree found with depth-first search (DFS) has this property.*

Proof. Let L be a line graph of some simple graph and let T be a tree obtained by depth-first search in L. Assume that T has a vertex v of degree ≥ 4, then v has at least three child nodes a, b, c. Since T is a DFS tree, we know that $ab, ac, bc \notin E(L)$. Thus, L contains an induced claw on the nodes $\{v, a, b, c\}$, contradicting the assumption that L is a line graph. □

With a transformation of the problem to the line graph, this result immediately gives us a 3-approximation for the min-max CEP and max-min CEP. Of course, the condition for the max-min CVP translates to the requirement $\frac{w(G)}{3k} \geq w_{\max}$ on the edge-weighted graph.

However, we can use the fact that $L(G)$ is claw-free for an even stronger result. Namely, we obtain a 2-approximation for both variants of the CEP. The following lemma lays the foundation.

Lemma 6. *Let L be a line graph, let $w : V(L) \to \mathbb{R}_{>0}$ and let λ be a number satisfying $w(L) \geq \lambda \geq w_{\max}$.*
Then there exists a tree $S \subseteq L$ with $w(S) \in [\lambda, 2\lambda)$ such that $L \setminus S$ is connected.

Proof. The proof is constructive. Consider a depth-first search spanning tree T^r of L which is rooted at some node r. As L is claw-free, $\deg(r) < 3$. From Proposition 5 we know that $\Delta(T^r) \leq 3$ and hence, every node in T^r has at most two child nodes.

(a) line graph L

(b) DFS spanning tree of L with highlighted subtree $T_y^r + vy$

Figure 5: Exemplary transformation from line graph to DFS spanning tree.

If there exists some $v \in T^r$ such that $w(T_v^r) \in [\lambda, 2\lambda)$ we are done since $L \setminus T_v^r$ is connected. In the remaining case there has to exist a node v with child nodes x and y such that $w(T_v^r) \geq 2\lambda$ while $w(T_x^r) < \lambda$ and $w(T_y^r) < \lambda$. In particular, note that $w(T_l^r) \leq \lambda$ for every leaf l in T^r and that v with $w(T_v^r) \geq 2\lambda$ cannot have a single child node x with $w(T_x^r) < \lambda$. Consequently, we have

$$\lambda = 2\lambda - \lambda \leq w(T_v^r) - w(T_y^r) = w(T_x^r) + w(v) < \lambda + \lambda = 2\lambda$$

to prove $w(T_x^r) + w(v) \in [\lambda, 2\lambda)$, and analogously we obtain $w(T_y^r) + w(v) \in [\lambda, 2\lambda)$. So if $v = r$, we choose the tree consisting of T_y^r extended with the edge vy, which we denote by $S = T_y^r + vy$, and find that $L \setminus S$ is still connected. If $v \neq r$, then v has a parent node u. Because T^r is a DFS tree we know that $xy \notin E(L)$. Now recall that L as a line graph is claw-free which implies that $ux \in E(L)$ or $uy \in E(L)$. W.l.o.g. we assume the former and conclude that $S = T_y^r + vy$ is a sought tree in L. □

This result can be used to define an algorithm for tree partitioning of line graphs as listed in Algorithm 2. By showing that it yields a $[\lambda, 2\lambda)$-partition, we obtain 2-approximations for both versions of the CEP.

Algorithm 2: BalancedLineGraphPartition(L, w, λ)

Input: line graph L
$w : V(L) \to \mathbb{R}_{>0}$,
$\lambda \geq w_{\max}$
Output: trees $T_1, \ldots, T_m \subseteq L$ with $V(L) = \dot{\bigcup} V(T_i)$

```
1 T := ∅
2 while w(L) ≥ 2λ do
3     S := tree in L as constructed in Lemma 6
4     T := T ∪ S
5     L := L \ S
6 S := spanning tree of L
7 return T ∪ S
```

Proposition 7. *Let L be a line graph with positive node weights w and let $\lambda \geq w_{\max}$. Then, the output of BalancedLineGraphPartition(L, w, λ) is a $[\lambda, 2\lambda)$-partition of L.*

Proof. Let $T_1, \ldots, T_m$ be the output of BalancedLineGraphPartition(L, w, λ). From Lemma 6 we know that $w(T_i) \in [\lambda, 2\lambda)$ for all $i \in [m-1]$ and due to the condition in line 2 of the algorithm we have $w(T_m) < 2\lambda$. □

This result paves the way for the 2-approximation of the CEP. First, we transform a CEP into a CVP on the line graph to apply Algorithm 2. Since every connected vertex k-partition on a line graph can easily be converted into a connected edge k-partition of the original graph, we are done. The following corollary summarizes the results.

Corollary 8. *Let (G, w', k) be an instance of CEP and let (L, w, k) be the corresponding CVP on the line graph $L = L(G)$. Then,*

a) for $\lambda := \max\{w_{\max}, \frac{w(L)}{k}\}$, Algorithm 2 leads to a 2-approximation for the min-max CEP on (G, w', k).

b) if $\lambda := \frac{w(L)}{2k} \geq w_{\max}$, Algorithm 2 leads to a 2-approximation for the max-min CEP on (G, w', k).

We end this part with an analysis of the computational complexity of Algorithm 2. In the present form we have to compute a number of DFS trees depending on λ. However, one can modify the algorithm to work in a similar way as Algorithm 1, such that its runtime is linear in the size of the line graph. Note that the size of the line graph can be quadratic in the size of the original graph of the CEP. For implementational purposes, however, the line graph can be handled implicitly to achieve a linear running time for the approximative solution of the CEP.

References

[ABP93] Eliezer Agasi, Ronald I Becker, and Yehoshua Perl. "A shifting algorithm for constrained min-max partition on trees". In: *Discrete Applied Mathematics* 45.1 (1993), pp. 1–28.

[BPS80] Ronald I Becker, Yehoshua Perl, and Stephen R Schach. "A shifting algorithm for min-max tree partitioning". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 1980, pp. 64–75.

[Bec+01] R Becker, Isabella Lari, Mario Lucertini, and Bruno Simeone. "A polynomial-time algorithm for max-min partitioning of ladders". In: *Theory of Computing Systems* 34.4 (2001), pp. 353–374.

[Bec+98] Ronald Becker, Isabella Lari, Mario Lucertini, and Bruno Simeone. "Max-min partitioning of grid graphs into connected components". In: *Networks: An International Journal* 32.2 (1998), pp. 115–125.

[Bei70] Lowell W. Beineke. "Characterizations of derived graphs". In: *Journal of Combinatorial Theory* 9.2 (1970), pp. 129 –135.

[Bul+16] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. "Recent advances in graph partitioning". In: *Algorithm Engineering*. Springer, 2016, pp. 117–158.

[CGM83] Paolo M Camerini, Giulia Galbiati, and Francesco Maffioli. "On the complexity of finding multi-constrained spanning trees". In: *Discrete Applied Mathematics* 5.1 (1983), pp. 39–50.

[CSW07] Frédéric Chataigner, Liliane RB Salgado, and Yoshiko Wakabayashi. "Approximation and inapproximability results on balanced connected partitions of graphs". In: *Discrete Mathematics and Theoretical Computer Science* 9.1 (2007), pp. 177–192.

[CWC13] An-Chiang Chu, Bang Ye Wu, and Kun-Mao Chao. "A linear-time algorithm for finding an edge-partition with max-min ratio at most two". In: *Discrete Applied Mathematics* 161.7-8 (2013), pp. 932–943.

[Chl96] Janka Chlebíková. "Approximating the maximally balanced connected partition problem in graphs". In: *Information Processing Letters* 60.5 (1996), pp. 225–230.

[Chu+10] An-Chiang Chu, Bang Ye Wu, Hung-Lung Wang, and Kun-Mao Chao. "A tight bound on the min-ratio edge-partitioning problem of a tree". In: *Discrete Applied Mathematics* 158.14 (2010), pp. 1471–1478.

[DF85] Martin E Dyer and Alan M Frieze. "On the complexity of partitioning graphs into connected subgraphs". In: *Discrete Applied Mathematics* 10.2 (1985), pp. 139–153.

[FR92] Martin Fürer and Balaji Raghavachari. "Approximating the minimum degree spanning tree to within one from the optimal degree". In: *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 1992, pp. 317–324.

[Fre91] Greg N Frederickson. "Optimal algorithms for tree partitioning". In: *SODA*. Vol. 91. 1991, pp. 168–177.

[IZN06] Takehiro Ito, Xiao Zhou, and Takao Nishizeki. "Partitioning a graph of bounded tree-width to connected subgraphs of almost uniform size". In: *Journal of discrete algorithms* 4.1 (2006), pp. 142–154.

[Ito+12] Takehiro Ito, Takao Nishizeki, Michael Schröder, Takeaki Uno, and Xiao Zhou. "Partitioning a weighted tree into subtrees with weights in a given range". In: *Algorithmica* 62.3-4 (2012), pp. 823–841.

[JRP85] Michael Jünger, Gerhard Reinelt, and William R Pulleyblank. "On partitioning the edges of graphs into connected subgraphs". In: *Journal of Graph Theory* 9.4 (1985), pp. 539–549.

[KM77] Sukhamay Kundu and Jayadev Misra. "A linear tree partitioning algorithm". In: *SIAM Journal on Computing* 6.1 (1977), pp. 151–154.

[LPS93] Mario Lucertini, Yehoshua Perl, and Bruno Simeone. “Most uniform path partitioning and its use in image processing”. In: *Discrete Applied Mathematics* 42.2-3 (1993), pp. 227–256.

[Luk74] Joseph A. Lukes. “Efficient algorithm for the partitioning of trees”. In: *IBM Journal of Research and Development* 18.3 (1974), pp. 217–224.

[Möh+07] Rolf H Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. “Partitioning graphs to speedup Dijkstra’s algorithm”. In: *Journal of Experimental Algorithmics (JEA)* 11 (2007), pp. 2–8.

[PS81] Yehoshua Perl and Stephen R. Schach. “Max-Min Tree Partitioning”. In: *J. ACM* 28.1 (Jan. 1981), pp. 5–15.

[Wu+07] Bang Ye Wu, Hung-Lung Wang, Shih Ta Kuan, and Kun-Mao Chao. “On the uniform edge-partition of a tree”. In: *Discrete Applied Mathematics* 155.10 (2007), pp. 1213–1223.

[Zho+19] Xing Zhou, Huaimin Wang, Bo Ding, Tianjiang Hu, and Suning Shang. “Balanced connected task allocations for multi-robot systems: An exact flow-based integer program and an approximate tree-based genetic algorithm”. In: *Expert Systems with Applications* 116 (2019), pp. 10 –20.