Parameterized Complexity of Graph Partitioning into Connected Clusters

Ankit Abhinav¹, Susobhan Bandopadhyay¹, Aritra Banik¹, and Saket Saurabh²

¹National Institute of Science, Education and Research, An OCC of Homi Bhabha National Institute, Bhubaneswar 752050, Odisha, India., {ankit.abhinav, susobhan.bandopadhyay, aritra}@niser.ac.in
²The Institute of Mathematical Sciences, HBNI, Chennai, India, saket@imsc.res.in

February 25, 2022

Abstract

Given an undirected graph $G$ and $q$ integers $n_1, n_2, \ldots, n_q$, balanced connected $q$-partition problem ($BCP_q$) asks whether there exists a partition of the vertex set $V$ of $G$ into $q$ parts $V_1, V_2, V_3, \ldots, V_q$ such that for all $i \in [1, q]$, $|V_i| = n_i$ and the graph induced on $V_i$ is connected. A related problem denoted as the balanced connected $q$-edge partition problem ($BCEP_q$) is defined as follows. Given an undirected graph $G$ and $q$ integers $n_1, n_2, n_3, \ldots, n_q$, $BCEP_q$ asks whether there exists a partition of the edge set of $G$ into $q$ parts $E_1, E_2, E_3, \ldots, E_q$ such that for all $i \in [1, q]$, $|E_i| = n_i$ and the graph induced on the edge set $E_i$ is connected. Here we study both the problems for $q = 2$ and prove that $BCP_q$ for $q \geq 2$ is $W[1]$-hard. We also show that $BCP_2$ is unlikely to have a polynomial kernel on the class of planar graphs. Coming to the positive results, we show that $BCP_2$ is fixed parameter tractable (FPT) parameterized by treewidth of the graph, which generalizes to FPT algorithm for planar graphs. We design another FPT algorithm and a polynomial kernel on the class of unit disk graphs parameterized by $\min(n_1, n_2)$. Finally, we prove that unlike $BCP_2$, $BCEP_2$ is FPT parameterized by $\min(n_1, n_2)$.

1 Introduction

The following combinatorial question was raised by Andras Frank in his doctoral dissertation [10].

Given a $q$-connected graph $G$, $q$ vertices $\{v_1, v_2, \cdots, v_q\} \subseteq V(G)$, and $q$ positive integers $n_1, n_2, \cdots, n_q$ such that $\sum_{i=1}^{q} n_i = |V(G)|$, does there exists a partition $V_1, \cdots, V_q$ of $V(G)$ such that $v_i \in V_i$, $|V_i| = n_i$ and $G[V_i]$ is connected for all $1 \leq i \leq q$?
The question was partially answered by the author and the statement was proved for the case when $q = 2$. The original conjecture was settled separately by Győri [11] and Lovász [12] and it is famously known as the Győri-Lovász theorem. The proof given by Győri, was constructive. A polynomial time algorithm can be constructed to find a $q$ partition of a $q$-connected graph.

Surprisingly, when the graph is not $q$-connected, the problem becomes NP-hard, for $k \geq 2$ [3]. The general problem is known as the Balanced Connected $q$-Partition Problem ($\text{BCP}_q$) and can be formally stated as follows.

**Balanced Connected $q$-Partition Problem ($\text{BCP}_q$)**

**Input:** An undirected graph $G$, an integer $q$ and $q$ positive integers $n_1, n_2, \cdots, n_q$ such that $\sum_{i=1}^{q} n_i = |V(G)|$.

**Question:** Does there exist a partition $V_1, \cdots, V_q$ of $V(G)$ such that $|V_i| = n_i$ and $G[V_i]$ is connected for all $1 \leq i \leq q$?

The approximation version of the problem is called the Max Balanced Connected $q$-Partition Problem. The objective of the problem is to maximize the minimum partition size. The best known approximation algorithm known is due to Chlebíková [5] and it presents a $\frac{4}{3}$-factor algorithm. It is also known that $\text{BCP}_q$ does not admit an $\alpha$-approximation algorithm with $\alpha < 6/5$, unless $P = \text{NP}$ [4].

In this paper, we initiate the study of fixed parameter tractability of $\text{BCP}_q$. We have proved the following results.

**Our Results.** We have shown that $\text{BCP}_q$, for $q \geq 2$ is W[1]-hard. In Section 3 we have shown that $\text{BCP}_2$ is fixed parameter tractable when parameterized by the treewidth. More specifically we have constructed a $O^*(k^{O(k)})$ algorithm for $\text{BCP}_2$ where $k$ is the treewidth of the given graph.

In Section 4 we considered the case when the underlying graph is planar. We have presented a $O^*(k^{O(k)})$ algorithm for $\text{BCP}_2$ where $k = \min(n_1, n_2)$. In Section 4.2 we have proved that under standard complexity theoretic assumptions there is no poly-kernel for $\text{BCP}_2$ in planar graphs. In Section 5 we have proved that when the underlying graph is unit disk graph, there is a poly-kernel for $\text{BCP}_2$ parameterized by $k = \min(n_1, n_2)$. In Section 6 we have introduced the "edge" variant of the $\text{BCP}_2$ which we is defined as follows.

**Balanced Connected $q$-Edge Partition Problem ($\text{BCEP}_q$)**

**Input:** An undirected graph $G$, an integer $q$ and $q$ positive integers $n_1, n_2, \cdots, n_q$ such that $\sum_{i=1}^{q} n_i = |E(G)|$.

**Question:** Does there exist a partition $E_1, \cdots, E_q$ of $E(G)$ such that $|E_i| = n_i$ and $G[E_i]$ is connected for all $1 \leq i \leq q$?

In section 6 we have proved that unlike $\text{BCP}_2$, $\text{BCEP}_2$ is FPT parameterized by $\min(n_1, n_2)$. 

In section 6 we have proved that unlike $\text{BCP}_2$, $\text{BCEP}_2$ is FPT parameterized by $\min(n_1, n_2)$.
1.1 Notations

**Graph Notations.** Let $G$ be a graph. We use $V(G)$ and $E(G)$ to denote the set of vertices and edge of $G$, respectively. Throughout the paper we use $n$ and $m$ to denote $|V(G)|$ and $|E(G)|$, respectively. For a vertex $v$, we use $N(v)$ to denote set of its neighbors, and use $\deg(v)$ to denote $|N(v)|$. For any graph $G$ and a set of vertices $V' \subseteq V(G)$, we denote the subgraph of $G$ induced by $V'$ by $G[V']$. For any set of vertices $X \in V(G)$, we denote $V(G) \setminus X$ by $\tilde{X}$. Most of the symbols and notations used of graph theory are standard and taken from Diestel [8].

**Parameterized Complexity and Algorithms.** The goal of parameterized complexity is to find ways of solving NP-hard problems more efficiently than brute force: here the aim is to restrict the combinatorial explosion to a parameter that is hopefully much smaller than the input size.

Formally, a parameterization of a problem is assigning a positive integer parameter $k$ to each input instance and we say that a parameterized problem is fixed-parameter tractable (FPT) if there is an algorithm that solves the problem in time $f(k) \cdot |I|^{O(1)}$, where $|I|$ is the size of the input and $f$ is an arbitrary computable function that depends only on the parameter $k$. Such an algorithm is called an FPT algorithm and such a running time is called FPT running time. There is also an accompanying theory of parameterized intractability using which one can identify parameterized problems that are unlikely to admit FPT algorithms. These are essentially proved by showing that the problem is W-hard.

Another major research field in parameterized complexity is kernelization. Kernelization is a technique that formalizes the notion of preprocessing. A kernelization algorithm for a parameterized problem replaces, in polynomial time, the given instance by an equivalent instance whose size is a function of the parameter. The reduced instance is called a kernel for the problem. Once we have a kernel, a brute force algorithm on the resulting instance answers the problem in FPT time. If the kernel size is a polynomial function of the input, then the problem is said to have a polynomial kernel.

**Definition 1** (Composition [2]). A composition algorithm (also called OR-composition algorithm) for a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $((x_1,k), \ldots, (x_t,k))$, with $(x_i,k) \in \Sigma^* \times \mathbb{N}$ for each $1 \leq i \leq t$, uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y,k') \in \Sigma^* \times \mathbb{N}$ with (a) $(y,k') \in \Pi \iff (x_i,k) \in \Pi$ for some $1 \leq i \leq t$ and (b) $k'$ is polynomial in $k$. A parameterized problem is compositional (or OR-compositional) if there is a composition algorithm for it.

It is unlikely that an NP-complete problem has both a composition algorithm and a polynomial kernel as suggested by the following theorem.

**Theorem 1** ([2] [9]). Let $\Pi$ be a compositional parameterized problem whose unparameterized version $\tilde{\Pi}$ is NP-complete. Then, if $\Pi$ has a polynomial kernel then $\text{co-NP} \subseteq \text{NP/poly}$.

**Treewidth.** A tree decomposition of a graph $G$ is a pair $(T,B)$ consisting of a tree $T$ and a family
\( B = \{ B_t : t \in V(T) \} \) of sets \( B_t \subseteq V(G) \), called bags, satisfying the following three conditions:

(I1) \( V(G) = \bigcup_{t \in V(T)} B_t \)

(I2) For every edge \((u, v) \in E(G)\), there exists a \( t \in V(T) \) such that \( u, v \in B_t \)

(I3) For every vertex \( v \in V(G) \) the set, \( T_v = \{ t \in V(T) | v \in B_t \} \), induces a connected subtree of \( T \).

The width of a tree decomposition \((T, B)\) is \( |B_t| - 1 : t \in V(T) \). The treewidth of \( G \) is the minimum width over all tree decompositions of \( G \). A tree decomposition \((T, B)\) where \( T \) is a rooted tree with root node \( r \), is nice if \( B_r = \emptyset \) and \( B_t = \emptyset \) for every leaf node \( l \) of \( T \). Further every non-leaf node of \( T \) is of one of the following three types:

**Introduce vertex node**: a node \( t \) with exactly one child \( t' \) such that \( B_t = B_{t'} \cup \{ v \} \) for some \( v \notin B_{t'} \). We say that \( v \) is introduced at \( t \).

**Introduce edge node**: a node \( t \), labeled with an edge \((u, v) \in E(G)\) such that \( u, v \in B_t \), and with exactly one child \( t' \) such that \( B_t = B_{t'} \). We say that edge \((u, v)\) is introduced at \( t \).

**Forget node**: a node \( t \) with exactly one child \( t' \) such that \( B_t = B_{t'} \setminus \{ v \} \) for some \( v \in B_{t'} \). We say that \( v \) is forgotten at \( t \).

**Join Node** a node \( t \) with exactly two child \( t_1 \) and \( t_2 \) such that \( B_t = B_{t_1} = B_{t_2} \).

### 2 Hardness of BCP\(_2\)

In the following theorem, we show that the above problem is \( \text{NP}\)-hard as well as \( \text{W}[1]\)-hard.

**Theorem 2.** The BCP\(_2\) is \( \text{W}[1]\)-hard parameterized by the solution size.

**Proof.** We show a reduction from the CLIQUE. Let \( I(G, k) \) be an instance of the CLIQUE. We create a new instance \( I'(G', k) \) of BCP\(_2\) as follows.

\[
V(G') = V' = V \cup U \cup W \text{ where } V = V(G) \text{ induces a clique. For every non-neighbor } v_i, v_j \in V(G) \text{ we have a vertex } u_{ij} \in U \text{ where } v_i \text{ and } v_j \text{ are connected to } u_{ij} \text{ by an edge. For each vertex } u_{ij} \in U \text{ we have } k + 1 \text{ many vertices } \{ w_{ij}^1, w_{ij}^2, \ldots, w_{ij}^{k+1} \} \text{ in } W \text{ where there is an edge between each } w_{ij}^l \text{ and } u_{ij} \text{ where } 1 \leq l \leq k + 1 \text{ (see Figure 1). Formally } U = \{ u_{ij} | (v_i, v_j) \notin E(G) \} \text{ and } W = \{ w_{ij}^l | u_{ij} \in U \text{ and } 1 \leq l \leq k + 1 \}. E(G') = E_1 \cup E_2 \cup E_3 \text{ where } E_1 = \{ (v_i, v_j) | (v_i, v_j) \in V \}, E_2 = \{ (v_a, u_{a,b}) | u_{a,b} \in U \} \cup \{ (v_b, u_{a,b}) | u_{a,b} \in U \} \text{ and } E_3 = \{ (u_{ij}, w_{ij}^l) | u_{ij} \in U, 1 \leq l \leq k + 1 \}.
\]

**Claim 1.** The \( G \) contains a clique of size \( k \) if and only if there exists a set \( X \subseteq V(G') \) of cardinality \( k \) such that \( G'[V' \setminus X] \text{ and } G'[X] \) both are connected.

\( ^2 \)Theorems marked with \( * \) are true for a more constrained version of the BCP\(_2\), i.e \( G[X] \) is not only connected but also contains a path on \( k \) vertices.
Figure 1: Construction of the Connected Modulator to Connected Set Problem instance

Proof. Assume, \( G \) contains a clique \( S \) of size \( k \). Observe that \( S \), also, induces a clique of size \( k \) in the graph \( G' \). Thus, \( G[S] \) is connected. Now we show that \( G'[V' \setminus S] \) is connected. For the sake of contradiction, assume in \( G'[V' \setminus S] \), there exist two vertices \( x \) and \( y \) which are not connected in \( G'[V' \setminus S] \).

Case 1: \( x \in V \setminus S \) and \( y \in V \setminus S \). As \( V \setminus S \) induces a clique in \( G' \), there is an edge between \( x \) and \( y \). Therefore, contradiction.

Case 2: \( x \in U \) and \( y \in V \setminus S \) or vice versa. Let \( x = u_{i,j} \). As \( v_i, v_j, y \in V \) they form a clique. Therefore, if either \( v_i \) or \( v_j \) is in \( V \setminus S \) say \( v_i \), then there is a path \( (x, v_i, y) \) between \( x \) and \( y \) in \( G'[V' \setminus S] \). On the other hand, \( v_i \) and \( v_j \) are nonadjacent in \( G \) (by definition of \( u_{i,j} \)). Thus, if \( v_i, v_j \in S \) then it contradicts the fact that \( S \) induces a clique in \( G \).

Case 3: \( x \in U \) and \( y \in U \). Let \( x = u_{a,b} \) and \( y = u_{c,d} \). Observe that as \( S \) is a clique in \( G \), and \( v_a \) and \( v_b \) are non-neighbours in \( G \), \( |S \cap \{v_a, v_b\}| \leq 1 \). Without loss of generality assume \( v_a \in V \setminus S \). Similarly, without loss of generality assume \( v_c \in V \setminus S \). Now we have a path \( (x, v_a, v_c, y) \) between \( x \) and \( y \) in \( G[V' \setminus S] \). Thus, contradiction.

Case 4: At least one of \( x \) or \( y \) in \( W \). Let \( x = w_{i,j}^l \). Observe that from the previous two cases (Case 2 and Case 3) we know that \( u_{i,j} \) is connected to every vertex in \( U \) and every vertex in \( V \setminus S \). Hence, \( u_{i,j} \) is connected to any vertex in \( W \). Hence, \( x \) is connected to all the vertices in \( U \cup \{V \setminus S\} \cup W \). This concludes the proof.

Now we prove the reverse direction. Assume that in \( G' \), there exists \( X \subseteq V' \) of cardinality \( k \) such that \( G'[V' \setminus X] \) and \( G'[X] \) both are connected. Observe that for all \( u_{i,j} \in U \), there are \( k + 1 \) pendant vertices adjacent with \( u_{i,j} \). Thus, there exist at least one vertex \( w_{i,j}^l \notin X \). Hence, \( u_{i,j} \notin X \) otherwise \( w_{i,j}^l \) will be isolated in \( G[V' \setminus X] \).
Observe, as \( X \) is connected, if there exist a vertex \( w_{ij}^l \in X \) then \( u_{ij} \in X \). We have ruled out that possibility, and hence \( W \cap X = \emptyset \). Hence, \( X \subseteq V \). Next, we show that \( X \) forms a clique in \( G \). Assume that two non-neighbors \( v_i, v_j \in X \). Observe that \( u_{ij} \) along with the vertices \( \{w_{ij}^l | 1 \leq l \leq k + 1\} \) is disconnected from the rest of the graph in \( G[V' \setminus X] \). This contradicts the fact that \( G[V' \setminus X] \) is connected. This concludes the proof.

Observe that this reduction is a parameter preserving reduction where the parameters in both the cases are the same as \( k \). Thus, this problem is \( \mathsf{W[1]} \)-hard, parameterized by the solution size.

### 3 BCP\(_2\) on Graphs with Bounded Treewidth

Treewidth is a measure to determine the tree-like structure of a graph \( G \). Many computationally hard problems become polynomial time solvable for graphs with bounded treewidth. Therefore, treewidth is a natural parameter and many hard problems are shown to be \( \mathsf{FPT} \) parameterized by the treewidth of the underlying graph. In this section, we present a \( \mathsf{FPT} \) algorithm for BCP\(_2\) parameterized by the treewidth of the given graph. More specifically, we will consider the following problem.

**Balanced Connected 2-Partition Problem (BCP\(_2\))**

**Parameter:** \( k \)

**Input:** An undirected graph \( G \), two positive integers \( n_1, n_2 \) and nice tree decomposition \((T, B)\) of \( G \) of width at most \( k \) such that \( n_1 + n_2 = |V(G)| \).

**Question:** Does there exists a partition \( \{V_1, V_2\} \) of \( V(G) \) such that \( |V_i| = n_i \) and \( G[V_i] \) is connected for all \( 1 \leq i \leq 2 \) ?

Due to the famous Courcelle’s theorem \([6]\), it is also known that any problem expressible in \( \mathsf{MSO}_2 \) is \( \mathsf{FPT} \) parameterized by treewidth.

**Theorem 3.** (Courcelle’s theorem, \([6]\)). Assume that \( \varphi \) is a formula of \( \mathsf{MSO}_2 \) and \( G \) is a \( n \)-vertex graph equipped with evaluation of all the free variables of \( \varphi \). Suppose, moreover, that a tree decomposition of \( G \) of width \( t \) is provided. Then there exists an algorithm that verifies whether \( \varphi \) is satisfied in \( G \) in time \( f(||\varphi||, t) \cdot n \), for some computable function \( f \).

Observe that BCP\(_2\) can be defined by \( \mathsf{MSO}_2 \) formula as follows.

\[
\text{con-ver-per}(X) = \text{con}(X) \land \text{con}(V \setminus X)
\]

where, for a graph \( G \) on the set \( V \) of vertices and \( X \subseteq V \), it checks whether both \( G[X] \) and \( G[V \setminus X] \) are connected. We adopt the definition of \( \text{con}(A) \) from \([7]\).

\[
\text{con}(A) = \forall B \subseteq V \left[ (\exists a \in A \land a \in B) \land \exists (b \in A \land b \notin B) \right] \implies \left( \exists e \in E \exists a \in A \exists b \in A \ \text{INC}(a, e) \land \text{INC}(b, e) \land a \in B \land b \notin B \right)
\]
where, given a graph $G = (V, E)$ and a subset $A$ of $V$, it checks whether $G[A]$ is connected.

INC$(u, e)$ checks whether the edge $e$ is incident on the vertex $u$.

Therefore, by Theorem 3, BCP$_2$ is FPT. Although Courcelle’s theorem provides an excellent tool to prove many problems to be FPT parameterized by treewidth, in practice $f(||\varphi||, t)$ is a rapidly increasing function. Even for very simple MSO$_2$ formulas, this function can have super-exponential dependence on the treewidth of the graph [14]. Thus, efforts have been made to construct a dynamic programming-based algorithm with a single exponential dependency on the treewidth. In the rest of the section, we provide an efficient dynamic programming-based algorithm for BCP$_2$ with improved runtime.

Observe that if $n_1 = |V(G)|$ or $n_1 = 0$, BCP$_2$ can be trivially solved. For ease of explanation from now onward, we assume that we are given two vertices $a$ and $b$ and the objective is to find a subset $V_t$ of size $n_1$ such that $a \in V_t, b \notin V_t$ and $G[V_t]$ and $G - V_t$ both are connected.

We add $a$ and $b$ to every bag of $\mathcal{B}$. Towards the construction of the dynamic programming, we define the following subproblem for each bag.

For a node $t$ of $T$, let $V_t$ be the union of all the bags present in the subtree of $T$ rooted at $t$, including $B_t$. With each node $t$ of the tree decomposition, we associate a subgraph $G_t$ of $G$ defined as follows:

$$G_t = (V_t, E_t) \text{ where } E_t = \{e \mid e \text{ is introduced in the subtree rooted at } t\}$$

We say the connected components $C_1, C_2, \cdots, C_b$ of a graph $G$ induce a partition $\mathcal{P} = \{P_1, P_2, \cdots, P_b\}$ of a vertex subset $X$ if $\forall i, \ V(C_i) \cap X = P_i$. For a bag $B_t$ and an arbitrary subset of vertices $V' \subseteq V(G)$, we define the following. $V_t^{V'} = V_t \cap V'$, $B_t^{V'} = B_t \cap V'$ and $G_t^{V'} = G_t[V']$.

For any solution $(A, B)_t$, $G_t^A$ and $G_t^B$ will contain one or many connected components. Observe that each such connected component must intersect $B_t$, as $B_t$ is a separator between, $V_t \setminus B_t$ and the rest of the graph. We will capture this idea in the following recursive definition.

For a bag $B_t$, a set of vertices $B_t^U$, a partition $\mathcal{P} = \{P_1, P_2, \cdots, P_p\}$ of $B_t^U$, a partition $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_q\}$ of $B_t^W = B_t \setminus B_t^U$ and an integer $1 \leq n_u \leq n_1$, $\eta(t, B_t^U, B_t^W, \mathcal{P}, \mathcal{Q}, n_u)$ is true if there exists a partition $(U, W)$ of $V_t$ where

- $U$ induces exactly $p = |\mathcal{P}|$ connected components in, $G_t[V_t]$ and those connected components induce the partition $\mathcal{P} = \{P_1, P_2, \cdots, P_p\}$ of $B_t^U = B_t \cap U$.
- $W$ induces exactly $q = |\mathcal{Q}|$ connected components in, $V_t$ and those connected components induce the partition $\mathcal{Q} = \{Q_1, Q_2, \cdots, Q_q\}$ of $B_t^W = B_t \cap W$.
- $|V_t^U| = n_u$

Please note that $B_t^W$ is a redundant parameter, it is kept in the definition for ease of understanding. Note that there is a solution of size $(n_1, n_2)$ if $\eta(r, \{a\}, \{b\}, \{\{a\}\}, \{\{b\}\}, n_1)$ is true. Now we present a bottom up recursive computation of $\eta$ as follows
Leaf Node. If \( t \) is a leaf node, set \( \eta(t, \{a\}, \{b\}, \{\{a\}\}, \{\{b\}\}, 1) = \text{true} \)

Introduce vertex node. Let \( v \) be the vertex introduced at \( B_i \) and \( t \) has only one child \( t' \). As none of the edges incident on \( v \) is introduced yet, \( v \) must form a singleton set either at \( \mathcal{P} \) or \( Q \). We present here the case when \( \{v\} \in \mathcal{P} \). Set \( \eta(t, B_i^U, B_i^W, \mathcal{P}, Q, n_u) = \eta(t, B_i^U \setminus \{v\}, B_i^W, \mathcal{P} \setminus \{\{v\}\}, Q, n_u - 1) \) if and only if both belongs to \( B_i^U \) or both belongs to \( B_i^W \). In the second case one of the \( u \) and \( v \) belongs to \( B_i^U \) and another in \( B_i^W \). We consider the Case 1 first. Without loss of generality assume \( u, v \in B_i^U \).

\[
\eta(t, B_i^U, B_i^W, \mathcal{P}, Q, n_u) = \left[ \bigvee_{P} \eta(t', B_i^U, B_i^W, \mathcal{P} \setminus P \cup \mathcal{P}', Q, n_u) \right] \bigvee \eta(t', B_i^U, B_i^W, P, Q, n_u)
\]

Next we consider the case when one of the \( u \) and \( v \) belongs to \( B_i^U \) and another in \( B_i^W \). In this case

\[
\eta(t, B_i^U, B_i^W, \mathcal{P}, Q, n_u) = \eta(t', B_i^U, B_i^W, \mathcal{P}, Q, n_u)
\]

Forget Node. Let \( t \) be the forget node with \( B_i = B_t \setminus \{w\} \).

\[
\eta(t, B_i^U, B_i^W, \mathcal{P}, Q, n_u) = \left[ \bigvee_{P} \eta(t', B_i^U, B_i^W, \mathcal{P} \setminus \{w\}, P', Q, n_u) \right] \bigvee \eta(t', B_i^U, B_i^W, \mathcal{P}, Q', n_u)
\]

Here \( \mathcal{P}' \) is a partition of \( S_t \cup \{w\} \) where the vertex \( w \) is being added to one of the parts in \( \mathcal{P} \) and \( \mathcal{Q}' \) is a partition of \( R_t \cup \{w\} \) where the vertex \( w \) is being added to one of the parts in \( \mathcal{Q} \).

Join Node. Suppose \( t \) is a join node with children \( t_1 \) and \( t_2 \). Recall that \( B_t = B_{t_1} = B_{t_2} \). Our objective here is to merge two partial solutions: one originating from \( G_{t_1} \) and the second originating from \( G_{t_2} \). Towards that, we define the following. For any set of vertices \( A \) and a partition of \( A, A = \{A_1, A_2, \cdots, A_u\} \) we define a graph \( G_A \) with vertex set \( A \) as follows. There is an edge between two vertices \( a_1 \) and \( a_2 \) if and only if \( a_1 \) and \( a_2 \) belongs to same partition of \( A \). For any set of vertices, \( A \) we say two partitions of \( A, A_1 \) and \( A_2 \) represents a third partition \( A \) if \( G_A = G_{A_1} \cup G_{A_2} \).

\[
\eta(t, B_i^U, B_i^W, \mathcal{P}, Q, n_u) = \bigvee_{P_1, P_2, Q_1, Q_2, n_u} [\eta(t_1, B_i^U, B_i^W, P_1, Q_1, n_u^1) \land \eta(t_2, B_i^U, B_i^W, P_2, Q_2, n_u^2)]
\]

The OR runs over all possible choice of \( P_1, P_2, Q_1, Q_2, n_u^1 \) where \( P_1, P_2 \) represents \( \mathcal{P}, Q_1, Q_2 \) represents \( Q, 0 \leq n_u^1 \leq n_u \) and \( n_u^2 = n_u - n_u^1 \). This concludes the dynamic programming description.
Runtime. The number of subproblems is determined by the number of different partition of $S_t$ and $R_t$ which is essentially bounded by the number of partition of $B_t$. The size of $B_t$ is bounded by $k_2$ and thus the number of partitions is bounded by $k^{O(k)}$. Time required to solve each sub-problem is $k^{O(k)}$. Hence, the total runtime is $k^{O(k)}$.

Note. Runtime of the algorithm can be improved to $O(c^k)$ for some constant $c$ using standard techniques using representative sets.

4 BCP$_2$ on Planar Graphs

4.1 FPT Algorithm for BCP$_2$ on Planar Graphs

In this section, we give an FPT algorithm for BCP$_2$. Let us first consider a restricted version of the BCP$_2$, where we are given a vertex $v$ such that $v \in X$. The problem definition is as follows.

**Restricted BCP$_2$**

**Input:** An undirected graph $G(V, E)$ with $n$ vertices, an integer $k$ and a vertex $v$.

**Question:** Does there exist $X \subseteq V$ of cardinality $k$ such that $G[X]$ and $G[V \setminus X]$ is connected and $v \in X$?

Next, we show a relation between the original version and the restricted version of the problem in the following lemma.

**Lemma 1.** If Restricted BCP$_2$ can be solved in FPT time then the BCP$_2$ can also be solved in time FPT.

**Proof.** Assume that the Restricted BCP$_2$ can be solved in time FPT. Observe that there are $n$ many choices for $v$, where $n$ is the number of vertices in the given graph. Thus given a graph we can create $n$ many restricted problem instances for this problem. Hence BCP$_2$ also can be solved in FPT time.

Let us first focus on the restricted version of the problem. Let the given vertex be $v$. Define $V_k$ as the set of the vertices we reach till $k$-th level if we start doing breath first search (BFS) from the the vertex $v$. Since $|X| = k$ and $v_i \in X$, $X \subseteq V_k$. Observe that the graph $G[V \setminus V_k]$ might have several connected components say, $G_1, G_2, \ldots, G_{\ell}$. Contract each connected component $G_i$ into a single vertex $u_i$. Let $U = \{u_i|i \leq \ell\}$. Next we consider the graph $G' = G[V_k \cup U]$ and make $N(u_i) = N(V(G_i))$ in $G'$. Observe that the diameter of $G'$ is $O(k)$. It is well known fact that a planar graph with diameter $d$ has treewidth $O(d)$. Thus treewidth of $G'$ is also bounded by $O(k)$.

Now, using the algorithm described in Section 3 we get the following theorem.

**Theorem 4.** The BCP$_2$ on planar graphs can be solved in time $k^{O(k)}$. 
4.2 No Polynomial Kernels for BCP in Planar Graph

In this section, we prove that no polynomial sized kernel exists for BCP on planar graphs. Towards that, we show a composition for Connected Modulator to Connected Set Problem on planar graphs in the following theorem.

**Theorem 5.** BCP on planar graph parameterized by the solution size does not have a polynomial kernel.

**Proof.** Let we are given $t$ many planar graphs $G_1, G_2, \ldots, G_t$ where each $G_i$ contains $n_i$ many vertices. We construct a new graph $G'$ as follows. We create $k + 1$ many copies for each planar graph $G_i$. Since each planar graph has a planar embedding keeping any vertex in the outer face, let us assume $G_1^t, G_2^t, \ldots, G_{k+1}^t$ be the embeddings of $G_i$ keeping $v_i^1, v_i^2, \ldots, v_i^{k+1}$ on the outer face, respectively. Now add an edge between $v_i^j$ and $v_i^{j+1}$ for all $1 \leq j \leq k$. Also, add an edge between $v_i^{k+1}$ and $v_{i+1}^1$ for all $1 \leq i \leq t - 1$.

**Claim 2.** $G'$ is a YES instance if and only if at least one of the $G_i$ is a YES instance.

**Proof.** We prove the forward direction first. Assume that in $G'$, there exists $X \subseteq V(G')$ of cardinality $k$ such that $G[X]$ and $G[V(G') \setminus X]$ both are connected. Since, deleting any vertex $v_i^j$ from $G_i^j$ where $j \in [k+1]$ makes the graph $G'$ disconnected. Thus $X$ can not contain any vertex $v_i^j$ from $G_i^j$ where $j \in [k+1]$. Hence, we can assume that $X \cap \{v_i^j\} = \emptyset$ in $G_j$ for $1 \leq j \leq k + 1$, i.e $X$ is contained completely inside any one of $G_i^j$ without containing any vertex at the outer face that shares edges with other copies of $G_i$. Therefore, $G[X]$ is a connected graph on $k$ vertices inside $G_i^j$ and $G[V(G_i^j) \setminus X]$ is connected. Hence $G_i$ is a YES instance.

Now, we prove the reverse direction. Assume that there is a graph, $G_i$ in which there exists $X \subseteq V(G_i^j)$ of cardinality $k$ such that $G[X]$ and $G[V(G_i^j) \setminus X]$ both are connected. Without loss of generality, assume $X = \{v_1, v_2, \ldots, v_k\}$. Since $G_i$ contains $X$, all the copies of $G_i$ in $G'$ also contain $X$. Now, we show that there exists a copy of $G_i$, say $G_i^n$ in $G'$, that does not contain any vertex of $X$ on the outer face which shares edges with other copies of $G_i$. As $|X| = k$, observe that there can be at most $k$ many different copies of $G_i$ in $G'$ which has a vertex of $X$ on the outer faces which shares edges with the other copies. Hence by pigeon hole principle, there exists at least a copy of $G_i$ in $G'$ which has no vertex on the outer face that shares edges with other copies. Thus, deleting $X$ from that copy will not disconnect any $G_i^n$ or $G_i^j$ in the graph $G'$. Therefore, $G'$ is also a YES instance.

5 BCP on Unit Disk Graphs

Let we are given an unit disk graph $G(V, E)$ and its representation $(D, C)$. Let $V = \{v_1, v_2, \ldots, v_n\}$ and for each $v_i$, $D_i$ be the corresponding disk centered at $c_i$. Here $D = \{D_i\} 1 \leq i \leq n$ and
\[ C = \{ c_i \mid 1 \leq i \leq n \}. \]

**Reduction Rule 1.** If the graph \( G \) has more than two connected components the return NO.

**Lemma 2.** Reduction Rule 1 is safe and can be implemented in polynomial time.

**Proof.** Assume \( G \) has at least three connected components. After deleting any subset \( X \) of cardinality \( k \) from the graph such that \( G[X] \) is connected, the graph remains disconnected. The number of components in the given graph can be checked in polynomial time. \( \square \)

Let us assume that the graph has two connected components. Observe that the given instance is a YES instance if and only if at least one of the two components contains exactly \( k \) vertices. Otherwise, the instance is a NO instance. The number of components in the given graph and the number of vertices in each component can be checked in polynomial time. Thus now onward we assume that \( G \) is connected. Let us consider a \( (\frac{1}{2} \times \frac{1}{2}) \) square grid on the plane. Assume \( V_S \) be the set of centers of the disks that are contained inside the cell \( S \); more formally, \( V_S = \{ v_i \mid c_i \in S \} \).

We also define \( N(V', S) \) as the set of vertices in the cell \( S \) that are neighbors of vertices in \( V' \). For any cell \( S \) in the grid, next, we prove that if \( S \) contains at least \( k + 24 \) centers then the given instance is a YES instance.

**Reduction Rule 2.** If there exists a cell \( S \) containing at least \( k + 24 \) many centers then return YES.

**Lemma 3.** Reduction Rule 2 is safe and can be implemented in polynomial time.

**Proof.** Select an arbitrary cell \( S \) with at least \( k + 24 \) centers. Observe that any vertex \( v \in V_S \) can have a neighbor inside the circle drawn centering \( v \) with radius one. That circle intersects at most 25 cells including \( S \). Observe that, these 25 cells form a \( 5 \times 5 \) grid keeping \( S \) as the center. Let \( A_S \) be the set of cells where a vertex \( v \in V_S \) can have neighbors. Now, for each such cell \( S' \in A_S \) if \( N(V_S, S) \neq \emptyset \), then mark an arbitrary vertex in \( N(V_S, S) \). As \( v \) can have neighbors in other 24 cells, we can mark at most 24 vertices in \( S \). Thus there are at least \( k \) unmarked vertices in \( S \).

Let \( X \) be a set of any arbitrary \( k \) many unmarked vertices. Observe that, any two disks having a center inside a cell intersect. Thus, all the centers in a cell form a clique. Hence \( G[X] \) is also a clique on \( k \) vertices. Observe that to show \( G[V \setminus X] \) is connected it is enough to show that all the vertices in \( N[X] \) are connected in \( G[V \setminus X] \). Let \( x \) and \( y \) be any two vertices from \( N[X] \) in \( G[V \setminus X] \). We show that there exists a path between \( x \) and \( y \) in \( G[V \setminus X] \). There can be three cases based on where the vertices \( x \) and \( y \) lie. First, both \( x, y \) in \( S \). As all the vertices in \( S \) form a clique, after deletion of \( X \) remaining vertices in \( S \) also induce a clique. Thus \( x \) and \( y \) are adjacent. The second case is when exactly one of \( x \) and \( y \) is in \( S \). Let \( x \in S \) and \( y \in S' \). If any vertex \( q \) in \( S' \) was adjacent to at least one vertex \( p \) in \( S \) then there is marked vertex \( p \) in \( S \) which is not in \( X \). Observe that, \( p \) is adjacent to \( x \) and \( q \) is adjacent to \( y \). Hence there exists a path between \( x \) and \( y \) in \( G[V \setminus X] \). If \( S' \) does not have any vertex adjacent to a vertex in \( S \), then \( y \) is connected to \( x \) via
vertices in other cell $S''$. Similarly, we can prove that any vertex in $S''$ is connected to $x$. Hence $x$ and $y$ are connected by a path. Now we consider the last case where both $x$ and $y$ are not in $S$. Assume $y \in V_{S'}$ and $x \in V_{S''}$. If any vertex $q$ in $S'$ was adjacent to at least one vertex $p$ in $S$ then there is marked vertex $p$ in $S$ which is not in $X$. Observe that, $q$ is adjacent to $y$. Again, if there exists a vertex $r$ in $S''$ that is adjacent to at least a vertex $s$ in $S$ then we can argue similarly and conclude that there exists a path between $x$ and $s$ in $G[V \setminus X]$. Observe that, $r$ and $p$ are adjacent in $G[V \setminus X]$, which connects $x$ and $y$ in $G[V \setminus X]$. Moreover, we can find such a subset $X$ of vertices in polynomial time. This concludes the proof.

Now onward we can assume that each cell has at most $k + 23$ centers. Next, we give a turing kernel for BCP$_2$ on unit disk graphs in the following theorem. Toward doing so we first prove the following theorem.

**Theorem 6.** *The Restricted BCP$_2$ has a $O(k^3)$ kernel on unit disk graphs.*

**Proof.** Let $v \in X$ be the given vertex. Observe that all the vertices in $X$ lie inside the circle drawn centering $v$ with radius $k$. Thus there are at most $O(k^2)$ many cells which can contain the other vertices of $X$. By Reduction Rule 2, no cell contains more than $k + 23$ vertices. Thus there can be at most $O(k^3)$ vertices, which can be the other endpoint of the path. Therefore we have $O(k^3)$ kernel.

Observe that from the graph $G$, for each $v_i$ we can create a restricted problem. From Theorem 6, Restricted BCP$_2$ has a $O(k^3)$ kernel. Thus we have the following corollary.

**Corollary 1.** *The BCP$_2$ on unit disk graphs admits a polynomial turing kernel.*

### 6 FPT Algorithm for BCEP$_2$

In this section, we present a FPT algorithm to solve the BCEP$_2$. The idea used here is similar to the FPT algorithm for the Longest path problem [1]. Before we proceed with the algorithm, let us introduce few well-known results which will be essential for our algorithm.

Matroid theory has been used widely to develop many FPT algorithms. In this section, we use representative sets defined on co-graphic matroid to design FPT algorithm for BCEP$_2$. We present a brief introduction to matroid and representative sets for the sake of completeness. For a more detailed discussion on these topics please refer to [7,13].

**Definition 2 ([13]).** A matroid $M$ is an ordered pair $(E,I)$ consisting of a finite set $E$ and a collection $I$ of subsets of $E$ having the following three properties:

1. $\emptyset \in I$.  
2. If $I \in I$ and $I' \subseteq I$, then $I' \in I$.

12
Theorem 7. There is an algorithm that, given a matrix $M$ over a field $GF(s)$, representing a matroid $M = (U,F)$ of rank $k$, a $p$-family $A$ of independent sets in $M$, and an integer $q$ such that $p + q = k$, computes a $q$-representative family $A' \subseteq_{rep} A$ of size at most $(p+q)$ using at most $O(|A|((p+q)p^q + (p+q)\omega^{-1}))$ operations over $GF(s)$.

Let $G$ be a graph. Let us define $CG(G) = (E, \mathcal{I})$ where $\mathcal{I} = \{E \subseteq E(G) | G - E$ is connected$\}$. $CG(G)$ is called the co-graphic matroid. For any vertex $v$, let $E_v^p$ be a set of subset of $p$ edges $X$ such that $G[X]$ is connected, $G[X]$ contains $v$ and $G - X$ is connected. Formally

$$E_v^p = \{X | X \subseteq E(G), |X| = p, G[X]$ connected and $G - X$ connected$\}$$

Observe that for all $v \in V$ and $1 \leq p \leq k$, $E_v^p$ is a $p$-family independent set in $CG(G)$. We can define $q$-representative family for $E_v^p$ with respect to $CG(G)$.

$$\mathcal{F}_v^{p,q} \subseteq_{rep} E_v^p$$

Next we present a dynamic programming based algorithm to compute $\mathcal{F}_v^{p,q}$ for all $v \in V(G)$, $1 \leq p \leq k$ and $1 \leq q \leq k - p$.

Theorem 8. There is an algorithm which computes for all $v \in V(G)$, $1 \leq p \leq k$ $1 \leq q \leq k - p$, $\mathcal{F}_v^{p,q}$ in time $O^*(5.2^k)$.

Proof. Observe that $E_v^1 = \{(u,v) : (u,v) \in E(G)\}$. We set $\mathcal{F}_v^{1,k-1} = E_v^1$. We compute $\mathcal{F}_v^{p,q}$ in a bottom up fashion using the following recursive definition. We create an auxiliary set $\overline{E}_v^q$ as follows

$$\overline{E}_v^q = \{E' \cup (u,v) | (u,v) \in E(G), X \in F_{u,v}^{p-1,q+1} and G - \{E' \cup (u,v)\}$ connected$\}$$

Observe that cardinality of $\overline{E}_v^q$ is at most $(p+q)|E(G)|$ and $\overline{E}_v^q$ is also a $p$-family independent set in $CG(G)$. Thus we can define a $q$-representative family $\mathcal{F}_v^{p,q} \subseteq_{rep} \overline{E}_v^q$. We can compute $\mathcal{F}_v^{p,q}$ using the algorithm of Theorem 7 in time $|E_{uv}^{p-1,q+1}|(p+q)^q n^{O(1)}$ which is equals to $(p+q)^q n^{O(1)}$. Note that as $q$-representation is transitive [7], $\mathcal{F}_v^{p,q}$ will $q$-represent $\mathcal{F}_v^{p,q}$. We set $\mathcal{F}_v^{p,q} = \mathcal{F}_v^{p,q}$. Overall runtime of the algorithm is $\sum_{p=2}^{k} (p+q)^q n^{O(1)}$ which is $O^*(2^k n)$. Due to a recent result [1], it is known that $\omega \leq 2.37286$. Therefore the runtime of the algorithm is bounded by $O^*(5.2^k)$. 

$\square$
References

[1] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 522–539. SIAM, 2021.

[2] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[3] Paolo M Camerini, Giulia Galbiati, and Francesco Maffioli. On the complexity of finding multi-constrained spanning trees. *Discrete Applied Mathematics*, 5(1):39–50, 1983.

[4] Frédéric Chataigner, Liliane RB Salgado, and Yoshiko Wakabayashi. Approximation and inapproximability results on balanced connected partitions of graphs. *Discrete Mathematics and Theoretical Computer Science*, 9(1):177–192, 2007.

[5] Janka Chlebíková. Approximating the maximally balanced connected partition problem in graphs. *Information Processing Letters*, 60(5):225–230, 1996.

[6] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, March 1990.

[7] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[8] Reinhard Diestel. Graph theory, volume 173 of *Graduate texts in mathematics*, page 7, 2012.

[9] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *STOC*, pages 133–142, 2008.

[10] A Frank. Combinatorial algorithms, algorithmic proofs. *Doctoral Thesis, Eotvos University*, 1975.

[11] Ervin Gyori. On division of graphs to connected subgraphs, combinatorics. In *Colloq. Math. Soc. Janos Bolyai*, 1976, 1976.

[12] László Lovász. A homology theory for spanning tress of a graph. *Acta Mathematica Hungarica*, 30(3-4):241–251, 1977.

[13] James G. Oxley. *Matroid Theory (Oxford Graduate Texts in Mathematics)*. Oxford University Press, Inc., USA, 2006.

[14] Wolfgang Thomas. *Languages, Automata, and Logic*, page 389–455. Springer-Verlag, Berlin, Heidelberg, 1997.