Balanced Connected Subgraph Problem in Geometric Intersection Graphs

Sujoy Bhore[1], Satyabrata Jana[2](✉), Supantha Pandit[3](✉), and Sasanka Roy[2]

[1] Algorithms and Complexity Group, TU Wien, Vienna, Austria
sujoy.bhore@gmail.com
[2] Indian Statistical Institute, Kolkata, India
satyamtma@gmail.com, sasanka.ro@gmail.com
[3] Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, Gujarat, India
pantha.pandit@gmail.com

Abstract. We study the Balanced Connected Subgraph (shortly, **BCS**) problem on geometric intersection graphs such as interval, circular-arc, permutation, unit-disk, outer-string graphs, etc. Given a *vertex-colored* graph $G = (V, E)$, where each vertex in V is colored with either "*red*" or "*blue*", the BCS problem seeks a maximum cardinality induced connected subgraph H of G such that H is *color-balanced*, i.e., H contains an equal number of red and blue vertices. We study the computational complexity landscape of the BCS problem while considering geometric intersection graphs. On one hand, we prove that the BCS problem is NP-hard on the unit disk, outer-string, complete grid, and unit square graphs. On the other hand, we design polynomial-time algorithms for the BCS problem on interval, circular-arc and permutation graphs. In particular, we give algorithms for the Steiner Tree problem on both interval and circular-arc graphs, and those algorithms are used as subroutines for solving the BCS problem on the same classes of graphs. Finally, we present a FPT algorithm for the BCS problem on general graphs.

Keywords: Balanced connected subgraph · Interval graphs · Permutation graphs · Circular-arc graphs · Unit-disk graphs · Outer-string graphs · NP-hard · Color-balanced · Fixed parameter tractable

1 Introduction

The *intersection graph* of a collection of sets is a graph where each vertex of the graph represents a set and there is an edge between two vertices if their corresponding sets intersect. Any graph can be represented as an intersection graph over some sets. Geometric intersection graph families consist of intersection graphs for which the underlying collection of sets are some geometric

S. Bhore—The author is supported by the Austrian Science Fund (FWF) grant P 31119.

objects. Some of the important graph classes in this family are interval graphs (intervals on real line), circular-arc graphs (arcs on a circle), permutation graphs (line segments with endpoints lying on two parallel lines), unit-disk graphs (unit disks in the Euclidean plane), unit-square graphs (unit squs in the Euclidean plane), outer-string graphs (curves lying inside a disk, with one endpoint on the boundary of the disk), etc. In the past several decades, geometric intersection graphs became very popular and extensively studied due to their interesting theoretical properties and applicability.

In this paper, we consider an interesting problem on general vertex-colored graphs called the Balanced Connected Subgraph (shortly, BCS) problem. A subgraph $H = (V', E')$ of G is called *color-balanced* if it contains an equal number of red and blue vertices.

Balanced Connected Subgraph (BCS) Problem
Input: A graph $G = (V, E)$, with node set $V = V_R \cup V_B$ partitioned into red nodes (V_R) and blue nodes (V_B).
Output: Maximum-sized color-balanced induced connected subgraph.

1.1 Previous Work

The BCS problem has been studied on various graph families such as trees, planar graphs, bipartite graphs, chordal graphs, split graphs, etc [2]. Most of the findings suggest that the problem is NP-hard for general graph classes, and it is possible to design polynomial time algorithms for the restricted classes with involved approaches. In [2], we have pointed out a connection between the BCS problem and graph motif problem (see, e.g., [6,7,11]). In the graph motif problem, we are given the input as a graph $G = (V, E)$, a color function $col : V \to \mathcal{C}$ on the vertices, and a multiset M, called motif, of colors of $\mathcal{C}$; the objective is to find a subset $V' \subseteq V$ such that the induced subgraph on V' is connected and $col(V') = M$. Note that, if $\mathcal{C} = \{\text{red, blue}\}$ and the motif has the same number of red and blues then, the solution of the graph motif problem gives a balanced connected subgraph. Indeed, a solution to the graph motif problem provides one balanced connected subgraph, with an impact of a polynomial factor in the running time. However, it does not guarantee the maximum size balanced connected subgraph. Nonetheless, the NP-hardness result for the BCS problem on any particular graph class implies the NP-hardness result for the graph motif problem on the same class. Graph motif problem has wide range of applications in bioinformatics [5], DNA physical mapping [8], perfect phylogeny [4], metabolic network analysis [12], protein–protein interaction networks and phylogenetic analysis [3]. This problem was introduced in the context of detecting patterns that occur in the interaction networks between chemical compounds and/or reactions [12].

1.2 Our Results

We present a collection of results on the BCS problem on geometric intersection graphs, that advances the study of this problem on diverse graph families.

➥ On the hardness side, in Sect. 2, we show that the BCS problem is NP-hard on unit-disk graphs, outer-string graphs, complete grid graphs, and unit square graphs.
➥ On the algorithmic side, in Sect. 3, we design polynomial-time algorithms for interval graphs ($\mathcal{O}(n^4 \log n)$ time), circular-arc graphs ($\mathcal{O}(n^6 \log n)$ time) and permutation graphs ($\mathcal{O}(n^6)$ time, and the result is described in the full version[1]). Moreover, we give an algorithm for the Steiner Tree problem on the interval graphs, that is used as a subroutine in the algorithm of the BCS problem for intervals graphs. Finally, we show that the BCS problem is fixed-parameter tractable for general graphs ($2^{\mathcal{O}(k)} n^2 \log n$) while parameterized by the number of vertices in a balanced connected subgraph.

2 Hardness Results

2.1 Unit-Disk Graphs

We show that the BCS problem is NP hard for the unit-disk graphs. We give a reduction from the Rectilinear Steiner Tree *(RST)* problem [9]. In this problem, we are given a set P of integer coordinate points in the plane and an integer L. The goal is to find a Steiner tree T (if one exists) of length at most L.

During the reduction, we first generate a geometric intersection graph from an instance $X(P, L)$ of the *RST* problem. The vertices of this graph are having integer coordinates and each edge is of unit length. Next, we show that this graph is a unit-disk graph.

Reduction: Suppose we have an instance $X(P, L)$ of the *RST* problem. For any point $p \in P$, let $p(x)$ and $p(y)$ denote the x- and y-coordinates of p, respectively. Let p_t, p_b, p_l, and p_r be the topmost (largest y-coordinate), bottom-most (smallest y-coordinate), leftmost (smallest x-coordinate), and rightmost (largest x-coordinate) points in P. We now take a unit integer rectangular grid graph D on the plane such that the coordinates of the lower-left grid vertex is $(p_l(x), p_b(y))$ and upper-right grid vertex is $(p_r(x), p_t(y))$. Now we associate each point p in P with a grid vertex d_p in D having the same x- and y-coordinates of p. Now we assign colors to the points in D. The vertices in D that are corresponding to the points in P are colored with red and the remaining grid vertices in D are colored with blue. We add some additional vertices to D as follows:

Observe that if there exists a Steiner tree T of length $L + 1 = |P|$ then T does not include any blue vertex in D. Further, if there exists a Steiner tree T of length $L + 1 = 2|P|$ then T contains equal number of red and blue vertices in D. Based on this observation we consider two cases to add some additional vertices (not necessarily forming a grid structure) to D.

[1] Some results are described in the full version, because of page limitations here.

Case 1. $[L + 1 \geq 2|P|]$**:** In this case the number of blue vertices in a Steiner tree T (if exists) is more than or equals to red vertices in D. We consider a path δ of $(L - 2|P| + 1)$ red vertices starting and ending with vertices r_1 and $r_{L-2|P|+1}$, respectively. The coordinates of r_i is $(p_l(x) - i, p_l(y))$, for $1 \leq i \leq L - 2|P| + 1$. We connect this path with D using an edge between the vertices r_1 and p_l. See Fig. 1(a) for an illustration of this construction. Let the resulting graph be $G_1 = D \cup \delta$.

Case 2. $[L+1 < 2|P|]$**:** In this case the number of red vertices in a Steiner tree T (if exists) is more than the number of blue vertices in D. We consider a path δ of $(2|P| - L)$ blue vertices starting and ending with vertices b_1 and $b_{2|P|-L}$, respectively. The coordinates of b_i is $(p_l(x) - i, p_l(y))$, for $1 \leq i \leq 2|P| - L$. We connect this path with D using an edge between the vertices b_1 and p_l. We add one more red vertex r' whose coordinates are $(p_{2|P|-L}(x) - 1, p_l(y))$ and connect it with $b_{2|P|-L}$ using an edge. See Fig. 1(b) for an illustration of this construction. Let the resulting graph be $G_2 = D \cup \delta \cup \{r'\}$

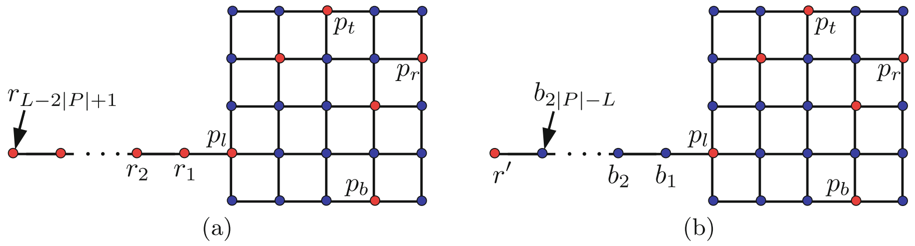

Fig. 1. (a) Construction of the instance G_1. (b) Construction of the instance G_2. (Color figure online)

This completes the construction. Clearly, the construction (either G_1 or G_2) can be done in polynomial time. Now we prove the following lemma.

Lemma 1. *The instance X of the RST problem has a solution T if and only if*

- **For Case 1:** *the instance G_1 has a balanced connected subgraph H with* $2(L - |P| + 1)$ *vertices.*
- **For Case 2:** *the instance G_2 has a balanced connected subgraph T with* $2(|P| + 1)$ *vertices.*

Proof. We prove this lemma for Case 1. The proof of Case 2 is similar.

For Case 1: Assume that X has a Steiner tree T of length L, where $L+1 \geq 2|P|$. Let U be the set of those vertices in G_1 corresponds to the vertices in T. Clearly, U contains $|P|$ red vertices and $L - |P| + 1$ blue vertices. Since $L + 1 \geq 2|P|$, to make U balanced it needs $L - |P| + 1 - |P|$ more red vertices. So we can add the path δ of $L - 2|P| + 1$ red vertices to U. Therefore, $U \cup \delta$ becomes connected and balanced (contains $L - |P| + 1$ vertices in each color).

On the other hand, assume that there is a balanced connected subgraph H in G with $(L - |P| + 1)$ vertices of each color. We can observe that H is a tree and no blue vertex in H is a leaf vertex. The number of red vertices in G_1 is exactly $(L - |P| + 1)$. So the H must pick all the $(L - |P| + 1)$ blue vertices that connect the vertices in G_1 corresponding to P. We take the set A of all the grid vertices corresponding to the vertices in H except the vertices $\{r_i; 1 \leq i \leq (L - 2|P| + 1)\}$. We output the Steiner tree T that contains the vertex set A and edge set E_A connecting the vertices of A according to the edges in H. As $|A| = 2(L - |P| + 1) - (L - 2|P| + 1) = L + 1$, so we output a Steiner tree of length L. □

We now show that either G_1 or G_2 is a unit-disk graph. Let us consider the graph G_1. For each vertex v in G_1 we take a unit disk whose radius is $\frac{1}{2}$ and center is on the vertex v. Therefore from the Lemma 1, we conclude that,

Theorem 1. *The* BCS *problem is NP-hard for unit-disk graphs.*

Extensions: The above reduction can be extended to prove that the BCS problem is NP-hard for the unit square graphs and the complete grid graphs (see the full version of the paper).

2.2 Outer-String Graphs

We show that the BCS problem is NP-hard for the outer-string graphs. We give a reduction from the dominating set problem that is known to be NP complete on general graphs [10]. Given a graph $G = (V, E)$, the dominating set problem asks whether there exists a set $U \subseteq V$ such that $|U| \leq k$ and $N[U] = V$, where $N[U]$ denotes neighbours of U in G including U itself.

During the reduction, we first generate a geometric intersection graph $H = (R \cup B, E')$ from an instance $X(G, k)$ of the dominating set problem on general graph. Next, we show that H is an outer-string graph.

Reduction: Let $G = (V, E)$ be graph with vertex set $V = \{v_1, v_2, \ldots, v_n\}$. For each vertex $v_i \in V$ we add a red vertex v_i and a blue vertex v_i' in H. For each edge $(v_i, v_j) \in E$, we add two edges $(v_i, v_j'), (v_i', v_j)$ in E'. Take a path of k red vertices starting at r_1 and ending at r_k. Also take a path of n blue vertices starting at b_1 and ending at b_n. Add the edges $(b_n, r_k), (b_1, v_1)$ into E'. We add edges between all pair of vertices in $\{v_1', v_2', \ldots v_n'\}$. Our construction ends with adding n edges (v_i, v_i') into E' for $1 \leq i \leq n$. This completes the construction. See Fig. 2 for an illustration of this construction that can be made in polynomial time.

Lemma 2. *The instance X has a dominating set of size k if and only if H has a balanced connected subgraph T with $2(n + k)$ vertices.*

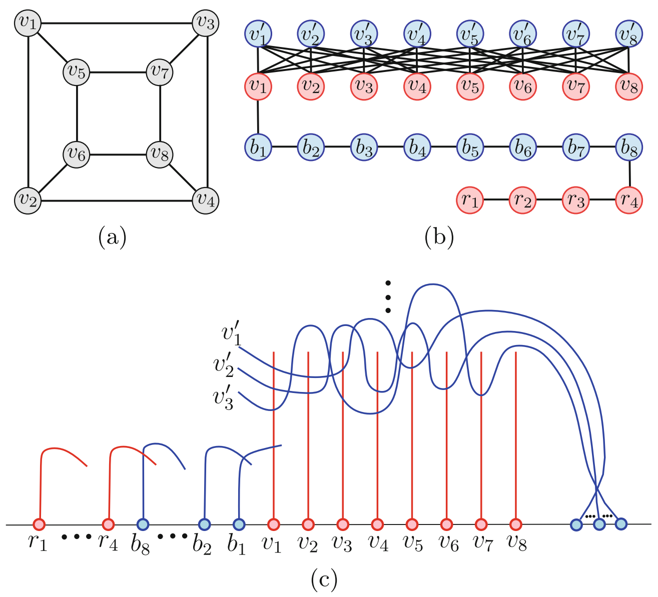
(a)
(b)
(c)
v_1'
v_2'
v_3'
r_1 r_4 b_8 b_2 b_1 v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8

Fig. 2. (a) A graph G. (b) Construction of H from G with $k = 4$. For the clarity of the figure we omit the edges between each pair of vertices v_i' and v_j', for $i \neq j$. (c) Intersection model of H. (Color figure online)

Proof. Assume that G has a dominating set U of size k. Now we take the subgraph T of H induced by $\{v_i' \colon v_i \in U\}$ along with the vertices $\{r_i \colon 1 \leq i \leq k\} \cup \{b_j \colon 1 \leq j \leq n\} \cup \{v_i \colon 1 \leq j \leq n\}$ in H. Now clearly H is connected and balanced with $2(n + k)$ vertices.

On the other hand, assume that there is a balanced connected tree T in H with $(n + k)$ vertices of each color. The number of red vertices in H is exactly $(n+k)$. So the solution must pick the blue vertices $\{b_i \colon 1 \leq i \leq n\}$ that connect v_1 with r_1. As T has exactly $(n + k)$ blue vertices then it T should pick exactly k vertices from the set $\{v_i' \colon 1 \leq i \leq n\}$. The set of vertices in V corresponding to these k vertices gives us a dominating set of size k in G. $\square$

We now verify that H is an outer-string graph. For an illustration see Fig. 2(c). We draw a horizontal line $y = 0$. For each vertex $v_i \in H$, draw the line segment $L_i = \overline{(i, 0)(i, 1)}$. For each vertex $v_j' \in H$, we draw a curve C_j, having one endpoint on the line $y = 0$, in such a way that for each edge $(v_j', v_i) \in H$, C_j bend above L_i (the line segment corresponding to v_i) and intersects the lines L_i from top. Also all the curves C_j's intersect each other. Now we add the curves corresponding to $\{r_i \colon 1 \leq i \leq k\} \cup \{b_j \colon 1 \leq j \leq n\}$ having one endpoint on the line $y = 0$ with satisfying the adjacency. Finally, using Lemma 2, we conclude:

Theorem 2. *The* BCS *problem is NP-hard for the outer-string graphs.*

3 Algorithmic Results

3.1 Interval Graphs

In this section, we study the BCS problem on the connected interval graphs. Given an n vertex interval graph $G = (V, E)$, we order the vertices of G based on the left endpoints $\{l_v : \forall v \in V\}$ of their corresponding intervals. Consider a pair of vertices $u, v \in V$ with $l_u \leq l_v$, we define a set $S_{u,v} = \{w : w \in V, l_u \leq l_w < r_w \leq r_v\} \cup \{u, v\}$. We also consider the case when $u = v$, and define $S_{u,u}$ in a similar fashion. Let H be a subgraph of G induced by $S_{u,v}$ (resp. $S_{u,u}$) in G. For any $u, v \in V$, consider $S_{u,v}$ and let r and b be the number of red and blue vertices in $S_{u,v}$, respectively. Without loss of generality, we may assume that $b \leq r$. The goal is to find a BCS, with cardinality $2b$ in H, containing $u, v \in V(H)$. Let B be the set of all blue intervals and $T = B \cup \{u, v\}$.

We compute a connected subgraph that contains T and some extra red intervals. For that, we use an algorithm for Steiner tree problem on interval graphs.

Steiner Tree on Interval Graphs: Given a simple connected interval graph $G = (V, E)$ and a set $T \subseteq V$ of terminals, the minimum Steiner tree problem seeks a smallest tree that spans over T. The vertices in $S = V \setminus T$ are denoted as Steiner vertices. First we describe a greedy algorithm called *Select_Steiners*$(G = (V, E), T)$ that computes a minimum Steiner tree $T \cup D$ on G.

We first partition T into m ($m \in [n]$) components $\{C_1, \ldots, C_m\}$ sorted from left to right based on the right endpoints of the components (note that, the union of the intervals in each component is an interval on the real line). Let I_{C_i} be the rightmost interval of the i-th component C_i. We consider the first component C_1 and the neighborhood set $N(I_{C_1})$ of I_{C_1}. Let I_j be the rightmost interval in $N(I_{C_1})$. By rightmost, we mean that the interval having the rightmost endpoint. We add I_j in D. Now, we recompute the components based on $T \cup D$. Note, $C_1 \cup I_j$ is now contained in one component. We repeat this produce until $T \cup D$ becomes a single component. Finally we return $T \cup D$ as a solution.

Correctness: It is easy to verify that the graph induced by $T \cup D$ is connected. Now we prove the optimality of the Algorithm *Select_Steiners*$(G = (V, E), T)$ by induction. The base case is that we have to connect the first two components C_1 and C_2. We choose the rightmost interval from the neighborhood of I_{C_1}. Note, that we have to connect C_1 and therefore it is inevitable that we have to pick an interval from $N(I_{C_1})$. We choose the rightmost interval (say, (I_ℓ)). Now, if this choice already connects C_2, we are done. Otherwise, $C_1 = C_1 \cup I_\ell$. Now, let us assume that we have obtained an optimal solution until step i. At step $i + 1$, we have to connect the first two components. By applying the same argument as the base case, we choose the rightmost interval from the first component and proceed. It is easy to verify that this algorithm runs in $\mathcal{O}(n^2)$ time.

Now, we go back to the BCS problem. Recall that, for any pair of intervals $u, v \in V$, our objective is to compute a BCS with vertex set T of cardinality $2b$ (if exists), where b and r is the number of red and blue intervals in $S_{u,v}$,

Algorithm 1. $BCS\_Interval(H)$

```
1: $T \leftarrow B \cup \{u, v\}$
2: $D = Select\_Steiners(H, T)$
3: $r' \leftarrow$ number of red vertices in $D \cup \{u, v\}$
4: $b' \leftarrow$ number of blue vertices in $T$
5: if $r' > b'$ then
6:     Return $\phi$
7: if $r' = b'$ then
8:     Return $G[D \cup T]$
9: if $r' < b'$ then
10:    Return $G[D \cup T \cup X]$
       where $X \subset V(H)$ is the set of $(b' - r')$ red vertices with $X \cap (D \cup T) = \phi$.
```

respectively, and $b \leq r$. Let H be the subgraph of G induced by $S_{u,v}$. Let R and B denote the set of red and blue intervals in $S_{u,v}$, respectively. We describe this process in Algorithm 1. We repeat this procedure for every pair of intervals and report the solution set with the maximum number of intervals.

Correctness: We prove that our algorithm yields an optimum solution. Let G' be such a solution. Let u and v be the intervals with leftmost endpoint and rightmost endpoint of G', respectively. Now we show that $V(G') = \min\{2r, 2b\}$, where r and b be the number of red and blue color vertices is $S_{u,v}$, respectively. Let us assume $V(G') \neq \min\{2r, 2b\}$. Then there exists at least one blue interval z and one red interval z' that belong to $S_{u,v} \setminus V(G')$. However, we know that $S_{u,v}$ induces an intersection graph of intervals corresponding to the vertices $\{w \colon w \in V, l_u \leq l_w < r_w \leq r_v\} \cup \{u, v\}$, and G' contain both u and v. So, $N[z] \cap G' \neq \phi$, $N[z'] \cap G' \neq \phi$. Therefore $V(G') \cup \{z, z'\}$ induces a balanced connected subgraph in G. It contradicts our assumption and hence the proof.

Time Complexity: Here we use the algorithm $Select\_Steiners(G = (V, E), T)$ as a subroutine. Using range tree the set $S_{u,v}$ can be obtained in $\mathcal{O}(\log n)$ time. Hence total running time is $\mathcal{O}(n^4 \log n)$.

Theorem 3. *Let G be an interval graph whose n vertices are colored either red or blue. Then* BCS *problem on G can be solved in $\mathcal{O}(n^4 \log n)$ time.*

3.2 Circular-Arc Graphs

We study the BCS problem on circular-arc graphs. We are given a bicolored the circular-arc graph $G = (V_R \cup V_B, E)$, where the set V_R and V_B contains a set of red and blue arcs, respectively. With out loss of generality we assume that the given input arcs fully cover the circle. Otherwise it becomes an interval graph and we use the algorithm of the interval graph to get an optimal solution.

Let us assume that H be a resulting maximum balanced connected subgraph of G, and let S denote the set of vertices in H. Since H is a connected subgraph of G, H covers the circle either partially or entirely. We propose an algorithm that computes a maximum size balanced connected subgraph H of G in polynomial time. Without loss of generality we assume that $V_B \leq V_R$. For any arc $u \in V$, let $l(u)$ and $r(u)$ denote the two endpoints of u in the clockwise order of the endpoints of the arc. To design the algorithm, we shall concentrate on the following two cases – **Case A** and **Case B**. In Case A, we check all possible instances while the the output set does not cover the circle fully. Then, in Case B, we handle all those instances while the output covers the entire circle. Later, we prove that the optimum solution lies in one of these instances. The objective is to reach the optimum solution by exploiting these instances exhaustively.

Case A: H covers the circle partially: In this case, there must be a clique of arcs K ($|K| \geq 1$) that is not present in the optimal solution. We consider any pair of arcs $u, v \in V$ such that $r(u) \prec l(v)$ in the clockwise order of the endpoints of the arcs, and consider the vertex set $S_{u,v} = \{w\colon w \in V, l_v \leq l_w < r_w \leq r_u\} \cup \{u, v\}$. Then, we use the Algorithm 1 to compute maximum BCS on $G[S_{u,v}]$. This process is repeated for each such pair of arcs, and report the BCS with maximum number of arcs.

Case B: H covers the circle entirely: In this case, S must contains $2|V_B|$ number of arcs and in fact that is the maximum number of arcs S can opt. In order to compute such a set S, first we add the vertices in V_B to S, then consider the vertices in V_B as a set T of terminal arcs and we need to find a minimum number of red arcs $D \in V_R$ to span over T. We further assume two cases.

B.1. [$T \cup D$ covers the circle partially] This case is similar to Case A without some extra red arcs that would be added afterwards to ensure that S contains $2|V_B|$ arcs. Similar to Case A, we again try all possible subsets obtained by pair of vertices u, v with $r(u) \prec l(v)$ and $S_{u,v}$ contains all blue vertices and we find optimal Steiner tree by using Algorithm $Select\_Steiners(G = (V, E), T)$. Then, we add $(|V_R| - |D|)$ (where D is the set of Steiner arcs) red arcs from V_R to S.

B.2. [$T \cup D$ covers the circle entirely] First, we obtain a set $\mathcal{C}$ of m (for some $m \in [n]$) components from T. We may see each component $C \in \mathcal{C}$ as an arc and the neighborhood set $N(C)$ as the union of the neighborhoods of the arcs contained in C. Observe that, for any component $C \in \mathcal{C}$, D contains either one arc from $N(C)$ that covers C, or two arcs from $N(C)$ where none of them individually covers C. Let us consider one component $C \in \mathcal{C}$. Let $l(C)$ and $r(C)$ be the left and right end points of C, respectively. If $|N(C) \cap D| = 1$, we consider each arc from $N(C)$ separately that contains C. For each such arc $I(C) \in N(C)$, we do the following three step operations –(1) include $I(C)$ in D, (2) remove $N(C)$ from the graph, (3) include two blue arcs $(l(I(C)), r(C))$ and $(r(C), r(I(C)))$ in the vertex set of the graph. Now, $T = T \cup \{[l(I(C)), r(C)), (r(C), r(I(C))]\}$. We give this processed graph that is an interval graph, as an input of the Steiner tree (Algorithm $Select\_Steiners(G = (V, E), T)$) and look for a tree with at most $(|D| - 1)$ Steiner red arcs. Else, when $|N(C) \cap D| = 2$, we take the arcs C_ℓ and C_r from $N(C)$ with leftmost and rightmost endpoints, respectively, in D. We do the same three step operations –(1) include C_ℓ and C_r in D, (2) remove $N(C)$ from the graph, (3) include two blue arcs $(l(C_\ell), l(C))$,

$(r(C), r(C_r)))$. Now, $T = T \cup \{[l(C_\ell), l(C))), (r(C), r(C_r))]\}$. We give this processed graph that is an interval graph, as an input of the Steiner tree (Algorithm $Select\_Steiners(G = (V, E), T)$) and look for a tree with at most $(|D| - 2)$ Steiner red arcs. This completes the procedure.

Correctness: We prove that our algorithm yields an optimum solution. The proof of correctness follows from the way we have designed the algorithm. The algorithm is divided into two cases. For case A, the primary objective is to construct the instances from a circular-arc graph to some interval graph. Thereafter, we can solve it optimally. Now, Case B is further divided into two sub-cases. Here we know that all blue vertices present in optimum solution. Therefore, our goal is to employ the Steiner tree algorithm with terminal set $T = V_B$. Note, for B 1 we again try all possible subsets obtained by pair of vertices u, v where $r(u) \prec l(v)$ and $S_{u,v}$ contains all blue vertices. Note, $G[S_{u,v}]$ is an interval graph and V_B is the set of terminals (since we assumed, w.l.o.g, $V_B \leq V_R$). Therefore we directly apply the Steiner tree procedure and obtain the optimum subset for each such pair. Indeed, this process reports the maximum BCS. In the case B 2 we modify the input graph in three step operations. Moreover, we update the expected output size to make it coherent to the modified input instance. This process is done for one arbitrary component of T (of size ≥ 1), which gives an interval graph. Clearly, the choice of this component makes no impact on the size of BCS. Thereafter, we follow the Steiner tree algorithm on this graph. Moreover, the algorithm exploits all possible cases and reduce the graph into interval graph without affecting the size of the BCS. Thereby, putting everything together, we conclude the proof.

Time Complexity: For case A, we try all pairs of arcs that holds certain condition and consider the subset (note, such subset can be computed in $\mathcal{O}(\log n)$ time given the clockwise order of the vertex set and a range tree where the arcs are stored). For each such subset we use the algorithm for interval graph to compute the maximum BCS. This whole process takes $\mathcal{O}(n^6 \log n)$ time. The complexity of Case A dominates complexity of Case B and we get the total running time $\mathcal{O}(n^6 \log n)$.

Theorem 4. *Given an n vertex circular-arc graph G whose vertices are colored either red or blue, the* BCS *problem on G can be solved in $\mathcal{O}(n^6 \log n)$ time.*

3.3 FPT Algorithm

In this section, we show that the BCS problem is fixed-parameter tractable for general graphs while parameterized by the solution size. Let $G = (V, E)$ be a simple connected graph, and let k be a given parameter. A family $\mathcal{F}$ of functions from V to $\{1, 2, \ldots, k\}$ is called a *perfect hash family* for V if the following condition holds. For any subset $U \subseteq V$ with $|U| = k$, there is a function $f \in \mathcal{F}$ which is injective on U, i.e., $f|_U$ is one-to-one. For any graph of n vertices and a positive integer k, it is possible to obtain a perfect hash family of size $2^{\mathcal{O}(k)} \log n$ in $2^{\mathcal{O}(k)} n \log n$ time; see [1]. Now, the k-BCS problem can be defined as follows.

Given a bicolored (red and blue) graph $G = (V, E)$, and a parameter k, decide if G contains a BCS of size k.

We employ two methods to solve the k-BCS problem: (i) *color coding technique*, and (ii) *batch procedure*. Our approach is motivated by the approach of Fellows et al. [7], where they have used these techniques to provide a FPT-algorithm for the graph motif problem. Suppose H is a solution to the k-BCS problem and $\mathcal{F}$ is a perfect hash family for V. This ensures us that at least one function of $\mathcal{F}$ assigns vertices of H with k distinct labels. Therefore, if we iterate through all functions of $\mathcal{F}$ and find the subsets of V of size k that are distinctly labeled by our hashing, we are guaranteed to find H. Now, we try to solve the following problem: Given a hash function $f\colon V \to \{1, 2, \ldots, k\}$ from perfect hash family $\mathcal{F}$, decide if there is a subset $U \subseteq V$ with $|U| = k$ such that $G[U]$ is a balanced connected subgraph of G and $f|_U$ is one-to-one.

First, we create a table, denoted by M. For a label $L \subseteq \{1, 2, \ldots, k\}$ and a color-pair (b, r) of non-negative integers where $b + r = |L|$, we put $M[v\ ;\ L,\ (b, r)] = 1$ if and only if there exists a subset $U \subseteq V$ of vertices such that the conditions holds: (i) $v \in U$, (ii) $f|_U = L$, (iii) $G[U]$ is connected, and (iv) U consisting exactly b blue vertices and r red vertices.

Notice that, the total number of entries of this table is $\mathcal{O}(2^k kn)$. If we can fill all the entries of the table M, then we can just look at the entries $M[v\ ;\ \{1, 2, \ldots, k\},\ (\frac{k}{2}, \frac{k}{2})]$, $\forall v \in V$, and if any of them is one then we can claim that the k-BCS problem has a solution. Now we use the batch procedure to compute $M[v\ ;\ L,\ (b, r)]$ for each subset $L \subseteq \{1, 2, \ldots k\}$, and for each pair (b, r) of non-negative integers such that $(b + r) = |L|$. Now, we explain the batch procedure. Without loss of generality we assume that $L = \{1, 2, \ldots, t\}$, $f(v) = t$, and the color of v is red.

Batch Procedure $(v, L, (b, r))$:

(1) Initialize: Construct the set $\mathcal{S}$ of pairs $(L', (b', r'))$, where $b' + r' = |L'|$ such that $L' \subseteq \{1, 2, \ldots, t-1\}$, $b' \leq b$, $r' \leq r - 1$ and $M[u\ ;\ L',\ (b', r')] = 1$ for some neighbour u of v.
(2) Update: If there exists two pairs $\{(L_1, (b_1, r_1)), (L_2, (b_2, r_2))\} \in \mathcal{S}$ such that $L_1 \cap L_2 = \phi$ and $(b_1, r_1) + (b_2, r_2) \leq (b, r-1)$, then add $(L_1 \cup L_2, (b_1 + b_2, r_1 + r_2))$ into $\mathcal{S}$. Repeat this step until unable to add any more.
(3) Decide: Set $M[v\ ;\ L,\ (b, r)] = 1$ if $(\{1, 2, \ldots, t-1\}, (b, r-1)) \in \mathcal{S}$, else 0.

Lemma 3. *The batch procedure correctly computes $M[v\ ;\ L,\ (b, r)]$ for all v, $L \subseteq \{1, 2, \ldots, k\}$ with $b + r = |L|$.*

Proof. Without loss of generality we assume that $L = \{1, 2, \ldots, t\}$, $f(v) = t$ and color of v is red. We have to show that $M[v\ ;\ L,\ (b, r)] = 1 \Leftrightarrow$ there exists a connected subgraph, containing v, and having exactly b blue vertices and r red vertices. Firstly, we assume that $M[v\ ;\ L,\ (b, r)] = 1$. So, $(\{1, 2, \ldots, t-1\}, (b, r-1)) \in \mathcal{S}$. So, there must exist some neighbours $\{v'_1, v'_2, \ldots, v'_l\}$ of v such that $M[v'_1;\ L_1,\ (b_1, r_1)] = M[v'_2;\ L_2,\ (b_2, r_2)] = \cdots = M[v'_l;\ L_l,\ (b_l, r_l)] = 1$ with $\bigcup_{i=1}^{l} L_i = L \setminus \{t\}$, $\sum_{i=1}^{l} b_i = b$, $\sum_{i=1}^{l} r_i = r - 1$ where $L_1, L_2, \ldots, L_l$ are pairwise disjoint. Thus, there exists a connected subgraph containing $\{v, v'_1, \ldots, v'_l\}$ having exactly b blue vertices and r red vertices. Other direction of the proof follows from the same idea.

Lemma 4. *Given a hash function $f\colon V \to \{1, 2, \ldots, k\}$, batch procedure fill all the entries of table M in $\mathcal{O}(2^{4k}k^3n^2)$ time.*

Proof. The initialization depends on the number of the search process in the entries correspond to the neighbour of v. Now, this number is bounded by the size of M. The first step takes $\mathcal{O}(2^kkn)$ time. Now the size of $\mathcal{S}$ can be at most 2^kk. Each update takes $\mathcal{O}(2^{2k}k^2)$ time. So step 2 takes $\mathcal{O}(2^{3k}k^3)$ time. Now the value of $M[v\ ;\ L,\ (b, r)]$ can be decided in $\mathcal{O}(2^kk)$ time. As the number of entries in M is $\mathcal{O}(2^kkn)$, so the total running time is $\mathcal{O}(2^{4k}k^3n^2)$. □

The algorithm for the k-BCS problem is following:

1. Construct a perfect hash family $\mathcal{F}$ of $2^{\mathcal{O}(k)} \log n$ functions in $2^{\mathcal{O}(k)}n \log n$ time.

2. For each function, $f \in \mathcal{F}$ build the table M using batch procedure. For each function $f \in \mathcal{F}$ it takes $\mathcal{O}(2^{4k}k^3n^2)$ time.

3. As each $f \in \mathcal{F}$ is perfect, by an exhaustive search through all function in $\mathcal{F}$ our algorithm correctly decide whether there exists a balanced connected subgraph of k vertices. We output yes, if and only if there is a vertex v and $f \in \mathcal{F}$ for which the corresponding table M contains the entry one in $M[v\ ;\ \{1, 2, \ldots, k\},\ (\frac{k}{2}, \frac{k}{2})]$.

Theorem 5. *The k-BCS problem can be solved in time $2^{\mathcal{O}(k)}n^2 \log n$ time.*

Acknowledgement. We thank Joseph S. B. Mitchell for his useful suggestions.

References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM **42**(4), 844–856 (1995)
2. Bhore, S., Chakraborty, S., Jana, S., Mitchell, J.S.B., Pandit, S., Roy, S.: The balanced connected subgraph problem. In: Pal, S.P., Vijayakumar, A. (eds.) CALDAM 2019. LNCS, vol. 11394, pp. 201–215. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11509-8_17
3. Bodlaender, H.L., Fellows, M.R., Langston, M.A., Ragan, M.A., Rosamond, F.A., Weyer, M.: Quadratic kernelization for convex recoloring of trees. In: Lin, G. (ed.) COCOON 2007. LNCS, vol. 4598, pp. 86–96. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73545-8_11
4. Bodlaender, H.L., Fellows, M.R., Warnow, T.J.: Two strikes against perfect phylogeny. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 273–283. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-55719-9_80
5. Bodlaender, H.L., de Fluiter, B.: Intervalizing k-colored graphs. In: Fülöp, Z., Gécseg, F. (eds.) ICALP 1995. LNCS, vol. 944, pp. 87–98. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-60084-1_65
6. Bonnet, É., Sikora, F.: The graph motif problem parameterized by the structure of the input graph. Discret. Appl. Math. **231**, 78–94 (2017)

7. Fellows, M.R., Fertin, G., Hermelin, D., Vialette, S.: Upper and lower bounds for finding connected motifs in vertex-colored graphs. J. Comput. Syst. Sci. **77**(4), 799–811 (2011)
8. Fellows, M.R., Hallett, M.T., Wareham, H.T.: DNA physical mapping: three ways difficult. In: Lengauer, T. (ed.) ESA 1993. LNCS, vol. 726, pp. 157–168. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57273-2_52
9. Garey, M.R., Johnson, D.S.: The rectilinear steiner tree problem is NP-complete. SIAM J. Appl. Math. **32**(4), 826–834 (1977)
10. Kikuno, T., Yoshida, N., Kakuda, Y.: The NP-completeness of the dominating set problem in cubic planer graphs. IEICI Trans. (1976–1990) **63**(6), 443–444 (1980)
11. Lacroix, V., Fernandes, C.G., Sagot, M.: Motif search in graphs: application to metabolic networks. IEEE/ACM Trans. Comput. Biol. Bioinf. **3**(4), 360–368 (2006)
12. Lacroix, V., Fernandes, C.G., Sagot, M.F.: Motif search in graphs: application to metabolic networks. IEEE/ACM Trans. Comput. Biol. Bioinform. (TCBB) **3**(4), 360–368 (2006)