Parameterized Complexity of Locally Minimal Defensive Alliances

Ajinkya Gaikwad, Soumen Maity(✉), and Shuvam Kant Tripathi

Indian Institute of Science Education and Research, Pune, India
soumen@iiserpune.ac.in, {ajinkya.gaikwad, tripathi.shuvamkant}@students.iiserpune.ac.in

Abstract. A defensive alliance in a graph $G = (V, E)$ is a set of vertices S satisfying the condition that every vertex $v \in S$ has at least as many neighbours (including itself) in S as it has in $V \backslash S$. We consider the notion of local minimality in this paper. We are interested in locally minimal defensive alliance of maximum size. This problem is known to be NP-hard but its parameterized complexity remains open until now. We enhance our understanding of the problem from the viewpoint of parameterized complexity. The three main results of the paper are the following: (1) when the input graph happens to be a tree, Locally Minimal Strong Defensive Alliance can be solved in polynomial time, (2) Locally Minimal Defensive Alliance is fixed parameter tractable (FPT) when parametrized by neighbourhood diversity, and (3) Locally Minimal Defensive Alliance can be solved in polynomial time for graphs of bounded treewidth.

Keywords: Parameterized complexity · FPT · Treewidth

1 Introduction

During the last 20 years, the Defensive Alliance problem has been studied extensively. A defensive alliance in an undirected graph is a set of vertices with the property that each vertex has at least as many neighbours in the alliance (including itself) as neighbours outside the alliance. In 2000, Kristiansen, Hedetniemi, and Hedetniemi [10] introduced defensive, offensive, and powerful alliances, and further studied by Shafique [7] and other authors. In this paper, we will focus on defensive alliances. A defensive alliance is *strong* if each vertex has at least as many neighbours in the alliance (not counting itself) as outside the alliance. The theory of alliances in graphs have been studied intensively [2,5,6] both from a combinatorial and from a computational perspective. As

A. Gaikwad—The first author gratefully acknowledges support from the Ministry of Human Resource Development, Government of India, under Prime Minister's Research Fellowship Scheme (No. MRF-192002-211).
S. Maity—The author's research was supported in part by the Science and Engineering Research Board (SERB), Govt. of India, under Sanction Order No. MTR/2018/001025.

A. Mudgal and C. R. Subramanian (Eds.): CALDAM 2021, LNCS 12601, pp. 135–148, 2021.
https://doi.org/10.1007/978-3-030-67899-9_11

mentioned in [1], the focus has been mostly on finding small alliances, although studying large alliances do not only make a lot of sense from the original motivation of these notions, but was actually also delineated in the very first papers on alliances.

Note that defensive alliance is not a hereditary property, that is, a subset of defensive alliance is not necessarily a defensive alliance. Shafique [7] called an alliance a *locally minimal alliance* if the set obtained by removing any vertex of the alliance is not an alliance. Bazgan et al. [1] considered another notion of alliance that they called a *globally minimal alliance* which has the property that no proper subset is an alliance. In this paper we are interested in locally minimal alliances of maximum size. Clearly, the motivation is that big communities where every member still matters somehow are of more interest than really small communities. Also, there is a general mathematical interest in such type of problems, see [13].

2 Basic Notations

Throughout this article, $G = (V, E)$ denotes a finite, simple and undirected graph of order $|V| = n$. The subgraph induced by $S \subseteq V(G)$ is denoted by $G[S]$. For a vertex $v \in V$, we use $N_G(v) = \{u \ : \ (u, v) \in E(G)\}$ to denote the (open) neighbourhood of vertex v in G, and $N_G[v] = N_G(v) \cup \{v\}$ to denote the closed neighbourhood of v. The degree $d_G(v)$ of a vertex $v \in V(G)$ is $|N_G(v)|$. For a subset $S \subseteq V(G)$, we define its closed neighbourhood as $N_G[S] = \bigcup_{v \in S} N_G[v]$ and its open neighbourhood as $N_G(S) = N_G[S] \setminus S$. For a non-empty subset $S \subseteq V$ and a vertex $v \in V(G)$, $N_S(v)$ denotes the set of neighbours of v in S, that is, $N_S(v) = \{u \in S \ : \ (u, v) \in E(G)\}$. We use $d_S(v) = |N_S(v)|$ to denote the degree of vertex v in $G[S]$. The complement of the vertex set S in V is denoted by S^c.

Definition 1. A non-empty set $S \subseteq V$ is a defensive alliance in G if for each $v \in S$, $|N[v] \cap S| \geq |N(v) \setminus S|$, or equivalently, $d_S(v) + 1 \geq d_{S^c}(v)$.

A vertex $v \in S$ is said to be protected if $d_S(v) + 1 \geq d_{S^c}(v)$. A set $S \subseteq V$ is a defensive alliance if every vertex in S is protected.

Definition 2. A vertex $v \in S$ is said to be *marginally protected* if it becomes unprotected when one of its neighbour in S is moved from S to $V \setminus S$. A vertex $v \in S$ is said to be *strongly protected* if it remains protected even if one of its neighbours is moved from S to $V \setminus S$.

Definition 3. An alliance S is called a *locally minimal alliance* if for any $v \in S$, $S \setminus \{v\}$ is not an alliance.

Definition 4. An alliance S is *globally minimal alliance* or shorter *minimal alliance* if no proper subset is an alliance.

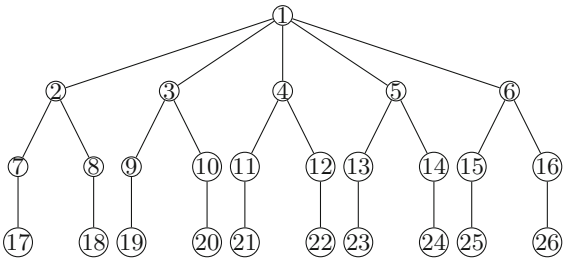

Fig. 1. A graph G with $A_L(G) = 10$ and $A(G) = 3$; $S = \{7, 2, 9, 3, 11, 4, 13, 5, 15, 6\}$ is a locally minimal defensive alliance of size 10 and $\{1, 2, 3\}$ is a globally minimal defensive alliance of size 3.

A defensive alliance S is connected if the subgraph induced by S is connected. An alliance S is called a *connected locally minimal alliance* if for any $v \in S$, $S \setminus \{v\}$ is not a connected alliance. Notice that any globally minimal alliance is also connected. As introduced in [1], we use $A_L(G)$ for the cardinality of the largest locally minimal defensive alliance in a graph G, and $A(G)$ for the cardinality of the largest globally minimal defensive alliance in a graph G (Fig. 1). In this paper, we consider Locally Minimal Defensive Alliance problem under structural parameters. We define the problem as follows:

Locally Minimal Defensive Alliance
Input: An undirected graph $G = (V, E)$ and an integer $k \leq |V(G)|$.
Question: Is there a locally minimal defensive alliance $S \subseteq V(G)$ such that $|S| \geq k$?

Our results are as follows:

- Locally Minimal Strong Defensive Alliance problem is polynomial time solvable on trees.
- Locally Minimal Defensive Alliance problem is FPT when parameterized by neighbourhood diversity of the input graph.
- Locally Minimal Defensive Alliance problem is polynomial time solvable for graphs with bounded treewith.

Known Results: The decision version for several types of alliances have been shown to be NP-complete. Carvajal et al. [3] proved that deciding if a graph contains a strong defensive alliance of size at most k is NP-hard. The defensive alliance problem is NP-complete even when restricted to split, chordal and bipartite graph [8]. Bazgan et al. [1] proved that deciding if a graph contains a locally minimal strong defensive alliance of size at least k is NP-complete, even when restricted to bipartite graphs with average degree less than 3.6; deciding if a graph contains a connected locally minimal strong defensive alliance or a connected locally minimal defensive alliance of size at least k is NP-complete,

even when restricted to bipartite graphs with average degree less than $2+\epsilon$, for any $\epsilon > 0$.

3 Locally Minimal Strong Defensive Alliance on Trees

Recall that a defensive alliance is *strong* if each vertex has at least as many neighbours in the alliance (not counting itself) as outside the alliance. Finding a locally minimal (strong) defensive alliance of maximum size is believed to be intractable [1]. However, when the graph happens to be a tree, we solve the problem in polynomial time, using dynamic programming. It may be observed that if S is a locally minimal strong defensive alliance, then for every vertex $v \in S$, at least one of its neighbours in S is marginally protected. A vertex $v \in S$ is said to be *good* if it has at least one marginally protected neighbour in S, otherwise it is called a *bad* vertex. Let v be a non-leaf node with children $v_1, v_2, \ldots, v_d$. Then $v \in S$ is *marginally unprotected* by its children if $\lceil \frac{d+1}{2} \rceil - 1$ of its children are in S; thus the parent of v must be in S for protection of v. Vertex v is *strongly protected* by its children if at least $\lceil \frac{d+1}{2} \rceil + 1$ of its children are in S. We define different possible states of a vertex v as follows:

- 0: vertex v is not in the solution.
- $\hat{1}_b$: vertex v is marginally unprotected by its children and none of its children are marginally protected.
- $\hat{1}_g$: vertex v is marginally unprotected by its children and if v has children then at least one of them is marginally protected.
- 1_{mg}: vertex v is marginally protected by its children and at least one of its children is marginally protected.
- 1_{sb}: vertex v is strongly protected by its children and none of the children is marginally protected.
- 1_{sg}: vertex v is strongly protected by its children and at least one of its children is marginally protected.

Here is the algorithm: Start by rooting the tree at any node v. Each node defines a subtree, the one hanging from it. This immediately suggests subproblems: $A_v(s)$ = the size of the largest locally minimal defensive alliance of the subtree rooted at v and the state of v is s. Our final goal is to compute $\max\left\{A_r(0), A_r(1_{sg}), A_r(1_{mg})\right\}$ where r is the root of T.

Leaf Node: For a leaf node v, we have $A_v(0) = 0$, $A_v(\hat{1}_b) = 1$; $A_v(\hat{1}_g) = A_v(1_{mg}) = A_v(1_{sb}) = A_v(1_{sg}) = -\infty$.

Non-leaf Node: Let v be a non-leaf node with the set $\mathcal{C} = \{v_1, v_2, \ldots, v_d\}$ of children. Suppose we know $A_{v_i}(s)$ for all children v_i of v. How can we compute $A_v(s)$? We now consider the following cases:

Case 1: Let the state of v be 0. Then

$$A_v(0) = \sum_{x \in \mathcal{C}} \max\left\{A_x(0), A_x(1_{sg}), A_x(1_{mg})\right\}$$

Case 2: Let the state of v be $\hat{1}_b$ or 1_{sb}. Let $(v_1, v_2, \ldots, v_d)$ be a descending ordering of $\mathcal{C}$ according to values $\max\{A_{v_i}(1_{sg}), A_{v_i}(1_{sb})\}$, that is,

$$\max\{A_{v_1}(1_{sg}), A_{v_1}(1_{sb})\} \geq \ldots \geq \max\{A_{v_d}(1_{sg}), A_{v_d}(1_{sb})\}.$$

Let $\mathcal{C}_{\lceil \frac{d+1}{2} \rceil - 1} = \{v_1, v_2, \ldots, v_{\lceil \frac{d+1}{2} \rceil - 1}\}$ and $\mathcal{C}_{\lceil \frac{d+1}{2} \rceil + 1} = \{v_1, v_2, \ldots, v_{\lceil \frac{d+1}{2} \rceil + 1}\}$. Then

$$A_v(\hat{1}_b) = 1 + \sum_{x \in \mathcal{C}_{\lceil \frac{d+1}{2} \rceil - 1}} \max\Big\{A_x(1_{sg}), A_x(1_{sb})\Big\} + \sum_{x \in \mathcal{C} \setminus \mathcal{C}_{\lceil \frac{d+1}{2} \rceil - 1}} A_x(0),$$

and

$$\begin{aligned} A_v(1_{sb}) = 1 &+ \sum_{x \in \mathcal{C}_{\lceil \frac{d+1}{2} \rceil + 1}} \max\Big\{A_x(1_{sg}), A_x(1_{sb})\Big\} \\ &+ \sum_{x \in \mathcal{C} \setminus \mathcal{C}_{\lceil \frac{d+1}{2} \rceil + 1}} \max\Big\{A_x(0), A_x(1_{sg}), A_x(1_{sb})\Big\} \end{aligned}$$

Thus, in this case, v must have at least $\lceil \frac{d+1}{2} \rceil + 1$ non-leaf children, otherwise, $A_v(1_{sb}) = -\infty$.

Case 3: Let the state of v be $\hat{1}_g$, 1_{mg} or 1_{sg}. Let $(v_1, v_2, \ldots, v_d)$ be a descending ordering of $\mathcal{C}$ according to values $\max\{A_{v_i}(\hat{1}_g), A_{v_i}(\hat{1}_b), A_{v_i}(1_{sg}), A_{v_i}(1_{sb})\}$. Let $\mathcal{C}_{k,i}$ be the set of first k children from the ordering $(v_1, v_2, \ldots, v_d)$ except vertex v_i. We have the following recurrence relations:

$$\begin{aligned} A_v(\hat{1}_g) = \max_{v_i \in \mathcal{C}} \Big\{ &1 + \max\{A_{v_i}(\hat{1}_g), A_{v_i}(\hat{1}_b)\} \\ &+ \sum_{x \in \mathcal{C}_{\lceil \frac{d+1}{2} \rceil - 2, i}} \max\{A_x(\hat{1}_g), A_x(\hat{1}_b), A_x(1_{sg}), A_x(1_{sb})\} \\ &+ \sum_{x \in \mathcal{C} \setminus \left(\mathcal{C}_{\lceil \frac{d+1}{2} \rceil - 2, i} \cup \{v_i\}\right)} A_x(0) \Big\}, \end{aligned}$$

for $d \geq 2$, and $A_v(\hat{1}_g) = 1$ for $d = 1$. Here $v \in S$ is good and *marginally unprotected* by its children, that is, exactly $\lceil \frac{d+1}{2} \rceil - 1$ of its children are in S and at least one of them is labelled $\hat{1}_g$ or $\hat{1}_b$ so that v is adjacent to at least one marginally protected child. Next, we have

$$\begin{aligned} A_v(1_{mg}) = \max_{v_i \in \mathcal{C}} \Big\{ &1 + \max\{A_{v_i}(\hat{1}_g), A_{v_i}(\hat{1}_b)\} \\ &+ \sum_{x \in \mathcal{C}_{\lceil \frac{d+1}{2} \rceil - 1, i}} \max\{A_x(\hat{1}_g), A_x(\hat{1}_b), A_x(1_{sg}), A_x(1_{sb})\} \\ &+ \sum_{x \in \mathcal{C} \setminus \left(\mathcal{C}_{\lceil \frac{d+1}{2} \rceil - 1, i} \cup \{v_i\}\right)} A_x(0) \Big\} \end{aligned}$$

Here $v \in S$ is good and *marginally protected* by its children, that is, exactly $\lceil \frac{d+1}{2} \rceil$ of its children are in S and at least one of them is labelled $\hat{1}_g$ or $\hat{1}_b$ so that v is adjacent to at least one marginally protected child. Finally, we have

$$A_v(1_{sg}) = \max_{\text{non-leaf } v_i \in \mathcal{C}} \Big\{ 1 + \max\{A_{v_i}(\hat{1}_g), A_{v_i}(\hat{1}_b)\} + \sum_{\text{non-leaf } x \in \mathcal{C}_{\lceil \frac{d+1}{2} \rceil, i}} \max\{A_x(\hat{1}_g), A_x(\hat{1}_b), A_x(1_{sg}), A_x(1_{sb})\} + \sum_{\text{non-leaf } x \in \mathcal{C} \setminus \left(\mathcal{C}_{\lceil \frac{d+1}{2} \rceil, i} \cup \{v_i\}\right)} \max\{A_x(0), A_x(\hat{1}_g), A_x(\hat{1}_b), A_x(1_{sg})\} \Big\}.$$

Here $v \in S$ is good and *strongly protected* by its children, that is, at least $\lceil \frac{d+1}{2} \rceil + 1$ of its children are in S and at least one of these $\lceil \frac{d+1}{2} \rceil + 1$ children is labelled $\hat{1}_g$ or $\hat{1}_b$ so that v is adjacent to at least one marginally protected child. It may be noted that if v_i is a leaf node then v_i cannot be in S. The reason is this; v_i's only neighbour is its parent v, which is strongly protected, therefore v_i will never have a marginally protected neighbour. Thus, in this case, v must have at least $\lceil \frac{d+1}{2} \rceil + 1$ non-leaf children, otherwise, $A_v(1_{sg}) = -\infty$. For computation of $A_r(1_{mg})$ and $A_r(1_{sg})$, we replace d by $d-1$ in the above recurrence relations as the root node r with d children has degree d, whereas other non-leaf node with d children has degree $d+1$.

The running time of this algorithm is easy to analyze. At each node $v \in V(T)$, we compute $A_v(s)$ where s is a state of v. The time required to get descending ordering of the children of v is $O(d \log d)$, where d is the number of children of vertex v. The number of subproblems is exactly the number of vertices in T. The total running time is therefore equal to $c \sum d_i \log d_i \leq c \log n \sum d_i = cn \log n = O(n \log n)$, where c is a constant.

4 FPT Algorithm Parameterized by Neighbourhood Diversity

In this section, we present an FPT algorithm for Locally Minimal Defensive Alliance problem parameterized by neighbourhood diversity. We say two vertices u and v have the same type if and only if $N(u) \setminus \{v\} = N(v) \setminus \{u\}$. The relation of having the same type is an equivalence relation. The idea of neighbourhood diversity is based on this type structure.

Definition 5. [11] The neighbourhood diversity of a graph $G = (V, E)$, denoted by $\mathtt{nd}(G)$, is the least integer k for which we can partition the set V of vertices into k classes, such that all vertices in each class have the same type.

If neighbourhood diversity of a graph is bounded by an integer k, then there exists a partition $\{C_1, C_2, \ldots, C_k\}$ of $V(G)$ into k type classes. It is known that such a minimum partition can be found in linear time using fast modular decomposition algorithms [14]. Notice that each type class could either be a clique or

an independent set by definition. For algorithmic purpose it is often useful to consider a *type graph* H of graph G, where each vertex of H is a type class in G, and two vertices C_i and C_j are adjacent iff there is complete bipartite clique between these type classes in G. It is not difficult to see that there will be either a complete bipartite clique or no edges between any two type classes. The key property of graphs of bounded neighbourhood diversity is that their type graphs have bounded size. In this section, we prove the following theorem:

Theorem 1. *The* Locally Minimal Defensive Alliance *problem is fixed-parameter tractable when parameterized by the neighbourhood diversity.*

Let G be a connected graph such that $\mathtt{nd}(G) = k$. Let $C_1, \ldots, C_k$ be the partition of $V(G)$ into sets of type classes. We assume $k \geq 2$ since otherwise the problem becomes trivial. Next we guess $|C_i \cap D|$ and whether the vertices in C_i are marginally or strongly protected, where D is a locally minimal defensive alliance. We make the following guesses:

- *Option 1:* $|C_i \cap D| = 0$.
- *Option 2:* $|C_i \cap D| = 1$ and the vertices in C_i are marginally protected.
- *Option 3:* $|C_i \cap D| = 1$ and the vertices in C_i are strongly protected.
- *Option 4:* $|C_i \cap D| > 1$ and the vertices in C_i are marginally protected.
- *Option 5:* $|C_i \cap D| > 1$ and the vertices in C_i are strongly protected.

There are at most 5^k choices for the tuple $(C_1, C_2, \ldots, C_k)$ as each C_i has 5 options as given above. Finally we reduce the problem of finding a locally minimal defensive alliance of maximum size to an integer linear programming optimization with k variables. Since integer linear programming is fixed parameter tractable when parameterized by the number of variables [12], we conclude that our problem is FPT when parameterized by the neighbourhood diversity.

ILP Formulation: Given a particular choice P of options for $(C_1, C_2, \ldots, C_k)$, our goal here is to find a locally minimal defensive alliance of maximum size. For each C_i, we associate a variable x_i that indicates $|D \cap C_i| = x_i$. Clearly, $x_i = 0$, if C_i is assigned Option 1; $x_i = 1$ if C_i is assigned Option 2 or 3; and $x_i > 1$ if C_i is assigned Option 3 or 4. Because the vertices in C_i have the same neighbourhood, the variables x_i determine D uniquely, up to isomorphism. Let $S_1 = \{C_i \mid x_i = 1\}$, $S_{>1} = \{C_i \mid x_i > 1\}$ and $S = S_1 \cup S_{>1}$. Let $H[S]$ be the subgraph of H induced by S. Now we label the vertices of $H[S]$ as follows: vertex C_i is labelled c_1 if it is a clique and Option 2 is assigned to C_i; vertex C_i is labelled $c_{>1}$ if it is a clique and Option 4 is assigned to C_i; vertex C_i is labelled `ind` if it is an independent set and Option 2 or 4 is assigned to C_i; vertex C_i is labelled s if it is a clique or an independent set, and Option 3 or 5 is assigned to C_i. To ensure local minimality of defensive alliance, the induced subgraph must satisfy the following conditions:

- Every vertex labelled s in the induced graph must have at least one neighbour labelled $c_1, c_{>1}$ or `ind`.
- Every vertex labelled c_1 in the induced graph must have at least one neighbour labelled $c_1, c_{>1}$ or `ind`.
- Every vertex labelled `ind` in the induced graph must have at least one neighbour labelled c_1, $c_{>1}$ or `ind`.

Above conditions ensure local minimality of the solution because when we remove a vertex from the solution, we make sure at least one of its neighbours gets unprotected. This happens because every vertex in the solution has at least one neighbour which is marginally protected. If the induced subgraph $H[S]$ satisfies all the above conditions then we proceed for the ILP, otherwise not. Let C be a subset of S consisting of all type classes which are cliques; $I = S \setminus C$ and $R = \{C_1, \ldots, C_k\} \setminus S$. Let n_i denote the number of vertices in C_i. We consider two cases:

Case 1: Suppose $v \in C_j$ where $C_j \in I$. Then the degree of v in D satisfies

$$d_D(v) = \sum_{C_i \in N_H(C_j) \cap S} x_i \tag{1}$$

Thus, including itself, v has $1 + \sum_{C_i \in N_H(C_j) \cap S} x_i$ defenders in G. Note that if $C_i \in D$, then only x_i vertices of C_i are in D and the remaining $n_i - x_i$ vertices of C_i are outside D. The degree of v outside D satisfies

$$d_{D^c}(v) = \sum_{C_i \in N_H(C_j) \cap S} (n_i - x_i) + \sum_{C_i \in N_H(C_j) \cap R} n_i \tag{2}$$

Case 2: Suppose $v \in C_j$ where $C_j \in C$. The degree of v in D satisfies

$$d_D(v) = \sum_{C_i \in N_H[C_j] \cap S} x_i \tag{3}$$

The degree of v outside D satisfies

$$d_{D^c}(v) = \sum_{C_i \in N_H[C_j] \cap S} (n_i - x_i) + \sum_{C_i \in N_H[C_j] \cap R} n_i \tag{4}$$

In the following, we present an ILP formulation of locally minimal defensive alliance problem, where a choice of options for $(C_1, \ldots, C_k)$ is given:

Maximize $\sum_{C_i \in S} x_i$

Subject to

$$1 + \sum_{C_i \in N_H(C_j) \cap S} 2x_i > \sum_{C_i \in N_H(C_j)} n_i, \quad \text{for all } C_j \in I, \text{ labelled } s,$$

$$\sum_{C_i \in N_H(C_j) \cap S} 2x_i - \sum_{C_i \in N_H(C_j)} n_i = 0 \text{ or } -1, \quad \text{for all } C_j \in I, \text{ labelled } \texttt{ind},$$

$$\sum_{C_i \in N_H[C_j] \cap S} 2x_i > \sum_{C_i \in N_H[C_j]} n_i, \quad \text{for all } C_j \in C, \text{ labelled } s,$$

$$\sum_{C_i \in N_H[C_j] \cap S} 2x_i - \sum_{C_i \in N_H[C_j]} n_i = 0 \text{ or } 1, \quad \text{for all } C_j \in C, \text{ labelled } c_1 \text{ or } c_{>1},$$

$x_i = 1$ for all $i \ : \ C_i \in S_1$;

$x_i \in \{2, 3, \ldots, |C_i|\}$ for all $i \ : \ C_i \in S_2$.

Solving the ILP: Lenstra [12] showed that the feasibility version of p-ILP is FPT with running time doubly exponential in p, where p is the number of variables. Later, Kannan [9] proved an algorithm for p-ILP running in time $p^{O(p)}$. In our algorithm, we need the optimization version of p-ILP rather than the feasibility version. We state the minimization version of p-ILP as presented by Fellows et. al. [4].

p-Variable Integer Linear Programming Optimization (p-Opt-ILP): Let matrices $A \in Z^{m \times p}$, $b \in Z^{p \times 1}$ and $c \in Z^{1 \times p}$ be given. We want to find a vector $x \in Z^{p \times 1}$ that minimizes the objective function $c \cdot x$ and satisfies the m inequalities, that is, $A \cdot x \geq b$. The number of variables p is the parameter. Then they showed the following:

Lemma 1. [4] p-Opt-ILP can be solved using $O(p^{2.5p+o(p)} \cdot L \cdot log(MN))$ arithmetic operations and space polynomial in L. Here L is the number of bits in the input, N is the maximum absolute value any variable can take, and M is an upper bound on the absolute value of the minimum taken by the objective function.

In the formulation for Locally Minimal Defensive Alliance problem, we have at most k variables. The value of objective function is bounded by n and the value of any variable in the integer linear programming is also bounded by n. The constraints can be represented using $O(k^2 \log n)$ bits. Lemma 1 implies that we can solve the problem with the guess P in FPT time. There are at most 5^k choices for P, and the ILP formula for a guess can be solved in FPT time. Thus Theorem 1 holds.

5 Graphs of Bounded Treewidth

In this section we prove that Locally Minimal Defensive Alliance problem can be solved in polynomial time for graphs of bounded treewidth. In other words, this section presents XP-time algorithm for Locally Minimal Defensive Alliance problem parameterized by treewidth. We now prove the following theorem:

Theorem 2. *Given an n-vertex graph G and its nice tree decomposition T of width at most k, the size of a maximum locally minimal defensive alliance of G can be computed in $8^k n^{O(2^{k+1})}$ time.*

Let $(T, \{X_t\}_{t \in V(T)})$ be a nice tree decomposition rooted at node r of the input graph G. For a node t of T, let V_t be the union of all bags present in the subtree of T rooted at t, including X_t. We denote by G_t the subgraph of G induced by V_t. For each node t of T, we construct a table $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta) \in \{\text{true, false}\}$ where $A \subseteq X_t$; $\mathbf{x}$ and $\mathbf{y}$ are vectors of length n; a, α and β are integers between 0 and n. We set $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta) =$ true if and only if there exists a set $A_t \subseteq V_t$ such that:

1. $A_t \cap X_t = A$
2. $a = |A_t|$
3. the ith coordinate of vector $\mathbf{x}$ is

$$x(i) = \begin{cases} d_{A_t}(v_i) & \text{for } v_i \in A \\ 0 & \text{otherwise} \end{cases}$$

4. α is the number of vertices $v \in A_t$ that are protected, that is, $d_{A_t}(v) \geq \frac{d_G(v)-1}{2}$.
5. A vertex $v \in A$ is said to be "good" if it has at least one marginally protected neighbour in $A_t \setminus A$. A vertex $v \in A$ is said to be "bad" if it has no marginally protected neighbours in $A_t \setminus A$. Here $\mathbf{y}$ is a vector of length n, and the ith coordinate of vector $\mathbf{y}$ is

$$y(i) = \begin{cases} g & \text{if } v_i \in A \text{ and } v_i \text{ is a good vertex} \\ b & \text{if } v_i \in A \text{ and } v_i \text{ is a bad vertex} \\ 0 & \text{otherwise} \end{cases}$$

6. $\mathbf{z}$ is a 2^k length vector, where the entry $z(S)$ associated with subset $S \subseteq A$ denotes the number of common bad neighbours of S in $A_t \setminus A$. The z vector considers the power set of A in lexicographic order. For example, let $A = \{a, b, c\}$, then $\mathbf{z} = \Big(z(\{a\}), z(\{a, b\}), z(\{a, b, c\}), z(\{a, c\}), z(\{b\}), z(\{b, c\}), z(\{c\})\Big)$.
7. β is the number of good vertices in A_t.

We compute all entries $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ in a bottom-up manner. Since $tw(T) \leq k$, at most $2^k n^k (n+1)^3 2^k n^{2^k} = 4^k n^{O(2^k)}$ records are maintained at each node t. Thus, to prove Theorem 2, it suffices to show that each entry $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ can be computed in $2^k n^{O(2^k)}$ time, assuming that the entries for the children of t are already computed.

Leaf Node: For a leaf node t we have that $X_t = \emptyset$. Thus $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ is true if and only if $A = \emptyset$, $\mathbf{x} = \mathbf{0}$, $a = 0$, $\alpha = 0$, $\mathbf{y} = \mathbf{0}$, $\mathbf{z} = \mathbf{0}$, $\beta = 0$. These conditions can be checked in $O(1)$ time.

Introduce Node: Suppose t is an introduction node with child t' such that $X_t = X_{t'} \cup \{v_i\}$ for some $v_i \notin X_{t'}$. Let A be any subset of X_t. We consider two cases:

Case (i): Let $v_i \notin A$. In this case $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ is true if and only if $dp_{t'}(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ is true.

Case (ii): Let $v_i \in A$. Here $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ is true if and only if there exist $A', \mathbf{x}', a', \alpha', \mathbf{y}', \mathbf{z}'$, and β' such that $dp_{t'}(A', \mathbf{x}', a', \alpha', \mathbf{y}', \mathbf{z}', \beta')$=true, where

1. $A = A' \cup \{v_i\}$;
2. $x(j) = x'(j) + 1$, if $v_j \in N_A(v_i)$, $x(i) = d_A(v_i)$, and $x(j) = x'(j)$ if $v_j \in A \setminus N_A[v_i]$;
3. $a = a' + 1$;
4. $\alpha = \alpha' + \delta$; here δ is the cardinality of the set

$$\left\{ v_j \in A \mid x'(j) < \frac{d_G(v_j) - 1}{2}; x(j) \geq \frac{d_G(v_j) - 1}{2} \right\}.$$

   That is, to compute α from α' we need to add the number δ of those vertices not satisfied in $(A', \mathbf{x}', a', \alpha', \mathbf{y}', \mathbf{z}', \beta')$ but satisfied in $(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$.
5. $y(i) = b$ and $y(j) = y'(j)$ for all $j \neq i$.
6. $z(S) = z'(S)$ if $v_i \notin S$; $z(S) = 0$ if $v_i \in S$.
7. $\beta = \beta'$.

For an introduce node t, $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ can be computed in $O(1)$ time. This follows from the fact that there is only one candidate of such tuple $(A', \mathbf{x}', a', \alpha', \mathbf{y}', \mathbf{z}', \beta')$.

Forget Node: Suppose t is a forget node with child t' such that $X_t = X_{t'} \setminus \{v_i\}$ for some $v_i \in X_{t'}$. Let A be any subset of X_t. Here $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ is true if and only if either $dp_{t'}(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ is true (this corresponds to the case that A_t does not contain v_i) or $dp_{t'}(A', \mathbf{x}', a', \alpha', \mathbf{y}', \mathbf{z}', \beta')$=true for some $A', \mathbf{x}'$, a', α', $\mathbf{y}'$, $\mathbf{z}'$, β' with the following conditions (this corresponds to the case that A_t contains v_i):

1. $A' = A \cup \{v_i\}$;
2. $x(j) = x'(j)$ for all $j \neq i$ and $x(i) = 0$;
3. $a = a'$;
4. $\alpha = \alpha'$;

We now consider four cases:
Case 1: v_i is not marginally protected and v_i is a good vertex.

5. $y(j) = y'(j)$ for all $j \neq i$ and $y(i) = 0$;
6. $z(S) = z'(S)$ for all $S \subseteq A$;
7. $\beta = \beta'$.

Case 2: v_i is not marginally protected and v_i is a bad vertex.

5. $y(j) = y'(j)$ for all $j \neq i$ and $y(i) = 0$;
6. $$z(S) = \begin{cases} z'(S) + 1 & \text{if } S \subseteq N_A(v_i) \\ z'(S) & \text{otherwise} \end{cases}$$
7. $\beta = \beta'$.

Case 3: v_i is marginally protected and v_i is a good vertex.

5. $$y(j) = \begin{cases} g & \text{if } v_j \in N_A(v_i) \\ y'(j) & \text{if } v_j \in A \setminus N_A(v_i) \end{cases}$$
6. $z(S) = z'(S) - z'(S \cup \{v_i\})$ for all $S \subseteq A$;
7. $\beta = \beta' + z'(\{v_i\}) + |\Big\{j \ : \ y'(j) = b; y(j) = g\Big\}|$.

Case 4: v_i is marginally protected and v_i is a bad vertex.

5. $$y(j) = \begin{cases} g & \text{if } v_j \in N_A(v_i) \\ y'(j) & \text{if } v_j \in A \setminus N_A(v_i) \end{cases}$$
6. $$z(S) = \begin{cases} z'(S) - z'(S \cup \{v_i\}) + 1 & \text{if } S \subseteq N_A(v_i) \\ z'(S) - z'(S \cup \{v_i\}) & \text{for all other subsets } S \subseteq A \end{cases}$$
7. $\beta = \beta' + z'(\{v_i\}) + |\Big\{j \ : \ y'(j) = b; y(j) = g\Big\}|$.

For a forget node t, $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ can be computed in $n^{O(2^k)}$ time. This follows from the fact that there are $n^{O(2^k)}$ candidates of such tuple $(A', \mathbf{x}', a', \alpha', \mathbf{z}', \beta')$, and each of them can be checked in $O(1)$ time.

Join Node: Suppose t is a join node with children t_1 and t_2 such that $X_t = X_{t_1} = X_{t_2}$. Let A be any subset of X_t. Then $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ is true if and only if there exist $(A_1, \mathbf{x_1}, a_1, \alpha_1, \mathbf{y_1}, \mathbf{z_1}, \beta_1)$ and $(A_2, \mathbf{x_2}, a_2, \alpha_2, \mathbf{y_2}, \mathbf{z_2}, \beta_2)$ such that $dp_{t_1}(A_1, \mathbf{x_1}, a_1, \alpha_1, \mathbf{y_1}, \mathbf{z_1}, \beta_1) =$ true and $dp_{t_2}(A_2, \mathbf{x_2}, a_2, \alpha_2, \mathbf{y_2}, \mathbf{z_2}, \beta_2) =$ true, where

1. $A = A_1 = A_2$;
2. $x(i) = x_1(i) + x_2(i) - d_A(v_i)$ for all $i \in A$, and $x(i) = 0$ if $i \notin A$;

3. $a = a_1 + a_2 - |A|$;
4. $\alpha = \alpha_1 + \alpha_2 - \gamma + \delta$; γ is the cardinality of the set

$$\left\{v_j \in A \mid x_1(j) \geq \frac{d_G(v_i)-1}{2};\ x_2(j) \geq \frac{d_G(v_i)-1}{2}\right\}$$

and δ is the cardinality of the set

$$\left\{v_j \in A \mid x_1(j) < \frac{d_G(v_i)-1}{2};\ x_2(j) < \frac{d_G(v_i)-1}{2};\ x(j) \geq \frac{d_G(v_i)-1}{2}\right\}.$$

To compute α from $\alpha_1 + \alpha_2$, we need to subtract the number of those v_j which are satisfied in both the branches and add the number of vertices v_j not satisfied in either of the branches t_1 and t_1 but satisfied in t.

5.
$$y(j) = \begin{cases} g & \text{if } y_1(j) = g \text{ or } y_2(j) = g \\ b & \text{otherwise} \end{cases}$$

6. $z(S) = z_1(S) + z_2(S)$ for all $S \subseteq A$;
7. $\beta = \beta_1 + \beta_2 - |\left\{j \ :\ y_1(j) = g, y_2(j) = g\right\}|$.

For a join node t, there are n^k possible pairs for $(\mathbf{x_1}, \mathbf{x_2})$ as $\mathbf{x_2}$ is uniquely determined by $\mathbf{x_1}$; $n+1$ possible pairs for (a_1, a_2); $n+1$ possible pairs for (α_1, α_2); there are 2^k possible pairs for $(\mathbf{y_1}, \mathbf{y_2})$ as $\mathbf{y_2}$ is uniquely determined by $\mathbf{y_1}$; there are n^{2^k} possible pairs for $(\mathbf{z_1}, \mathbf{z_2})$ as $\mathbf{z_2}$ is uniquely determined by $\mathbf{z_1}$; and $n+1$ possible pairs for (β_1, β_2). In total, there are $2^k n^{O(2^k)}$ candidates, and each of them can be checked in $O(1)$ time. Thus, for a join node t, $dp_t(A, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta)$ can be computed in $2^k n^{O(2^k)}$ time.

At the root node r, we look at all records such that $dp_r(\emptyset, \mathbf{x}, a, \alpha, \mathbf{y}, \mathbf{z}, \beta) =$ true, and $a = \alpha = \beta$. The size of a maximum locally minimal defensive alliance is the maximum a satisfying $dp_r(\emptyset, \mathbf{x}, a, a, \mathbf{y}, \mathbf{z}, a)=$ true.

6 Conclusion

The main contributions in this paper are that the Locally Minimal Defensive Alliance problem is FPT when parameterized by neighborhood diversity, the problem is polynomial time solvable on trees, and XP in treewidth. We list some nice problems emerge from the results here: is the problem FPT in treewidth, and does it admit a polynomial kernel in neighborhood diversity? Also, noting that the result for neighborhood diversity implies that the problem is FPT in vertex cover, it would be interesting to consider the parameterized complexity with respect to twin cover. The modular width parameter also appears to be a natural parameter to consider here, and since there are graphs with bounded modular-width and unbounded neighborhood diversity; we believe this is also an interesting open problem. The parameterized complexity of the Locally Minimal Defensive Alliance problem remains unsettle when parameterized by other important structural graph parameters like clique-width.

Acknowledgement. We are grateful to the referees for thorough reading and constructive comments that have made the paper better readable.

References

1. Bazgan, C., Fernau, H., Tuza, Z.: Aspects of upper defensive alliances. Discret. Appl. Math. **266**, 111–120 (2019)
2. Cami, A., Balakrishnan, H., Deo, N., Dutton, R.: On the complexity of finding optimal global alliances. J. Comb. Math. Comb. Comput. **58**, 23–31 (2006)
3. Carvajal, R., Matamala, M., Rapaport, I., Schabanel, N.: Small alliances in graphs. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 218–227. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74456-6_21
4. Fellows, M.R., Lokshtanov, D., Misra, N., Rosamond, F.A., Saurabh, S.: Graph layout problems parameterized by vertex cover. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 294–305. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-92182-0_28
5. Fernau, H., Rodriguez-Velazquez, J.A.: A survey on alliances and related parameters in graphs. Electron. J. Graph Theory Appl. **2**(1) (2014)
6. Fricke, G., Lawson, L., Haynes, T., Hedetniemi, M., Hedetniemi, S.: A note on defensive alliances in graphs. Bull. Inst. Comb. Appl. **38**, 37–41 (2003)
7. Hassan-Shafique, K.: Partitioning a graph in alliances and its application to data clustering (2004)
8. Jamieson, L.H., Hedetniemi, S.T., McRae, A.A.: The algorithmic complexity of alliances in graphs. J. Comb. Math. Comb. Comput. **68**, 137–150 (2009)
9. Kannan, R.: Minkowski's convex body theorem and integer programming. Math. Oper. Res. **12**(3), 415–440 (1987)
10. Kristiansen, P., Hedetniemi, M., Hedetniemi, S.: Alliances in graphs. J. Comb. Math. Comb. Comput. **48**, 157–177 (2004)
11. Lampis, M.: Algorithmic meta-theorems for restrictions of treewidth. Algorithmica **64**, 19–37 (2012)
12. Lenstra, H.W.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**(4), 538–548 (1983)
13. Manlove, D.: Minimaximal and maximinimal optimisation problems: a partial order-based approach (1998)
14. Tedder, M., Corneil, D., Habib, M., Paul, C.: Simpler linear-time modular decomposition via recursive factorizing permutations. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5125, pp. 634–645. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70575-8_52