The Algorithmic Complexity of Alliances in Graphs

Lindsay H. Jamieson and Stephen T. Hedetniemi
Department of Computer Science
Clemson University
Clemson, SC, USA
lch@cs.clemson.edu, hedet@cs.clemson.edu

Alice A. McRae
Department of Computer Science
Appalachian State University
Boone, NC, USA
aam@appstate.edu

Abstract

In 2001 Kristiansen, Hedetniemi and Hedetniemi [9] first defined the concept of a defensive alliance in a graph, to be a subset $S \subset V$ of a graph $G = (V, E)$ having the property that every vertex $v \in S$ has at most one more neighbor in V-S than it has in S (i.e. $|N[v] \cap S| \geq |N[v] - S|$). Since then several other types of alliances have been defined and studied including strong, offensive, global, powerful, and secure alliances. To date, no algorithms or complexity analyses have been developed for alliances in graphs. This is the subject of this paper.

Keywords: alliances, global alliances, complexity, algorithms

AMS Subject Classification: 05C69

1 Introduction

In order to define an alliance in a graph, we need the following definitions. Let $G = (V, E)$ be a graph. For any vertex $v \in V$, the open neighborhood of v is the set $N(v) = \{u : uv \in E\}$, while the closed neighborhood of v is the set $N[v] = N(v) \cup \{v\}$. The open and closed neighborhoods of a set $S \subset V$ are defined as $N(S) = \cup_{v \in S} N(v)$ and $N[S] = N(S) \cup S$, respectively. The boundary of a set S is the set $\delta(S) = \cup_{v \in S}(N(v) - S)$. A non-empty set of vertices $S \subset V$ is called a defensive alliance if and only if $\forall v \in S, |N[v] \cap S| \geq |N[v] - S|$. This is equivalent to saying that every vertex in S has at least as many neighbors in the alliance S (including itself) as it has neighbors not in the alliance. The defensive alliance number of a graph G equals the minimum cardinality of a defensive alliance in G and is denoted a(G). A defensive alliance is called strong if this inequality is strict, i.e. $\forall v \in S, |N[v] \cap S| > |N[v] - S|$. A non-empty set $S \subset V$ is an offensive alliance if $\forall v \in \delta S, |N[v] \cap S| \geq |N[v] - S|$. The offensive alliance number $a_o(G)$ equals the minimum cardinality of an offensive alliance in G. An offensive alliance is called strong if $\forall v \in \delta S, |N[v] \cap S| > |N[v] - S|$. An alliance which is both defensive and offensive is called a powerful alliance, ($a_p(G)$ denotes the powerful alliance number of a graph G) while an alliance S for which $\delta S = V - S$ is called a global alliance. First defined in 2001 [9] by Kristiansen, Hedetniemi and Hedetniemi, alliances have been studied mathematically in [2], [1], [3], [4], [5], [6], [7], [8] and [10]. To date, however, no algorithms or complexity analyses have been developed for alliances in graphs. In this paper, we present linear algorithms for finding a minimum cardinality powerful alliance and global powerful alliance in an arbitrary tree T, using an interesting adaptation of the dynamic programming method developed by Wimer [11]. We also establish the NP-completeness of the decision problems corresponding to a minimum defensive and a minimum powerful alliance in an arbitrary graph.

2 Powerful Alliances in Trees

In this section we present a linear time algorithm for finding the powerful alliance number of an arbitrary tree which is based on the dynamic programming method of Wimer et al [11]. However, Wimer's method must be modified in a new way in order to record the balance between neighbors in a powerful alliance and neighbors not in a powerful alliance at every vertex. Although having to remember this information does not change the complexity of the algorithm, it does require the use of conditionals in the composition tables.

In order to compute the powerful alliance number of a tree, we must determine the possible types of subtrees that could be created by a minimum cardinality powerful alliance, denoted p.a. Let S be the set of vertices in a minimum powerful alliance. Let T be the tree for which we are computing $a_p(T)$. Let r be the root of the current subtree. There are 6 cases for the subtree at any point in the algorithm, with the current difference between the number of neighbors that are in and not in S for each case being indicated by the value i, that is i = |N[r] ∩ S | -

|N[r]∩(V-S)|.

[1:i] = $\{r \in S, S \cap T \neq \oslash$ is a p. a. in $T\}$ /* i ≥ 0 */
[2:i] = $\{r \in S, S \cap T \neq \oslash$ is not a p. a. in T, but adding neighbors in S to r can create a p. a.$\}$ /* i < 0 */
[3:i] = $\{r \notin S, S \cap T \neq \oslash$ is exactly a p. a. in T (the balance is 0), $S \cap N[r] \neq \oslash\}$ /* i = 0 */
[4:i] = $\{r \notin S, S \cap T \neq \oslash$ is a p. a. in T but the p. a. is not minimal, $S \cap N[r] \neq \oslash\}$ /* i ≥ 1 */
[5:i] = $\{r \notin S, S \cap T \neq \oslash$ is a p. a. in T, $S \cap N[r] = \oslash\}$ /* i < 0 */
[6:i] = $\{r \notin S, S \cap T = \oslash$ and S is not a p. a. in $T\}$ /* i ¡ 0 */
[7:i] = $\{r \notin S, S \cap T \neq \oslash$ and S is not a p. a. in T, but adding neighbors in S to r can create a p. a.$\}$ /* i ¡ 0 */

Once these cases have been defined, we can consider the composition of tree T from its subtrees. At each point we combine a subtree T_1, rooted at r_1, p. a. S_1, and difference i, with a subtree T_2, rooted at r_2, p. a. S_2, and difference j, to produce a new subtree $T = T_1 \diamond T_2$ with root r_1 and p.a. $S_1 \cup S_2$. We must consider all possible combinations of cases [a:i] and [b:j], for $1 \leq a,b \leq 7$. If a particular combination can never occur when producing a minimum cardinality powerful alliance, then this combination is marked with an X in 1.

From 1, we obtain a set of recurrence relations as follows:
[1:i+1] = [1:i]◇[1:j] ∪ [2:i]◇[1:j] ∪ [2:i]◇[2:j]
[1:i-1] = [1:i]◇[2:j] ∪ [1:i]◇[3:j] ∪ [1:i]◇[5:j] ∪ [1:i]◇[6:j]
[2:i+1] = [2:i]◇[1:j] ∪ [2:i]◇[2:j]
[2:i-1] = [1:i]◇[3:j] ∪ [1:i]◇[5:j] ∪ [1:i]◇[6:j] ∪ [2:i]◇[3:j] ∪ [2:i]◇[5:j] ∪ [2:i]◇[6:j]
[3:i+1] = [3:i]◇[1:j] ∪ [5:i]◇[1:j] ∪ [6:i]◇[1:j]
[3:i-1] = [3:i]◇[5:j]
[4:i-1] = [4:i]◇[5:j] ∪ [5:i]◇[3:j] ∪ [5:i]◇[4:j]
[5:i-1] = [5:i]◇[5:j]
[6:i+1] = [5:i]◇[1:j] ∪ [6:i]◇[1:j]
[6:i-1] = [3:i]◇[5:j] ∪ [6:i]◇[5:j]

The initial values for each case combine to form the initial vector for the algorithm. We use either the number of vertices in the set, or ∞ to represent a state which cannot logically exist. When combining subtrees, we want to minimize the number of vertices in the set. The only cases which logically can exist initially for an isolated vertex are case 1, with a value of 1, and case 6, with a value of 0. This means that the initial vector for all vertices is [1, ∞, ∞, ∞, ∞, 0, ∞]. As the last step to set up the tree for the algorithm, we create a parent array for the tree. This means that we have an array which stores, for each vertex, the parent of that vertex, with the roots parent being ∞. After execution of the algorithm, we

	[1:j]	[2:j]	[3:j]	[4:j]	[5:j]	[6:j]	[7:j]
[1:i]	[1:i+1]	if (j = -1) then [1:i+1] else X	[2:i-1]	[1:i-1]	X	if (j = -1) then { if (i > 0) then [1:i-1] else [2:i-1} else X	if (j = -1) then { if (i > 0) then [1:i-1] else [2:i-1} else X
[2:i]	if (i = -1) then [1:i+1] else [2:i+1]	if (j = -1) then {if (i = -1) then [1:i+1] else [2:i+1]} else X	[2:i-1]	2:i-1	X	[2:i-1]	if (j = -1) then [2:i-1] else X
[3:i]	if (j > 0) then [4:i+1] else X	X	X	X	X	[7:i-1]	X
[4:i]	if (j > 0) then [4:i+1] else X	X	X	X	X	if (i=1) then [3:i-1] else 4:i-1]	X
[5:i]	X	X	X	X	X	[5:i-1]	X
[6:i]	if (j > 0) then { if (i = -1) then [3:i+1] else [7:i+1]} else X	X	X	[5:i-1]	[5:i-1]	[6:i-1]	X
[7:i]	if (j > 0) then { if (i = -1) then [3:i+1] else [7:i+1]} else X	X	X	X	X	[7:i-1]	X

Table 1. Compositions for powerful alliances.

need to determine the answer. Only three cases can produce a minimum powerful alliance. These are [3:x], [4:x], or [5:x] where x can be any value, or [1:(x>0)]. The minimum value from the final vector in these 4 cases is the size of a minimum powerful alliance for the tree T.

Limitations of space preclude a full discussion of all 49 entries in 1. We will discuss three of the more interesting cases: [2:i]◇[2:j], [4:i]◇[6:j], and [6:i]◇[5:j].

To start, consider the combination of a subtree T_1 of case [2:i] with a subtree T_2 of case [2:j]. If either i or j = -1, then the root has 2 more children not in the set than in the set. If either i or j is less than -1, then root has 2 or more children not in the set than in the set. So, if we try to combine a tree with i ≤ -1 with a tree with j < -1, we do not get a valid tree because the tree with j < -1 will never be combined with anything else and its root is not defined. If j = -1 there are two choices:

1. i = -1 : This forms a powerful alliance because the addition of a child r2 in the set S creates a powerful alliance for the subtree T_1 ([2:i]) and the addition of a parent in the set S creates a powerful alliance for the subtree T_2 ([2:j]). Thus, the new tree T is [1:i+1]

2. i < -1: This does not form a powerful alliance in the subtree T_1 ([2:i]). However, the addition of a parent in the set S does create a powerful alliance in the subtree T_2 ([2:j]), so the new tree T is [2:i+1] since the composition adds one more defender for r_1.

Consider next the combination of a subtree T_1 of case [4:i] with a subtree T_2 of case [6:j]. Because T_2 has no vertices in the set S, the value of j has no bearing on the combination. This means that there are two possibilities for tree T_1:

1. i = 1: Combining these two trees takes i to 0. This means that currently we have a powerful alliance exactly in the subtree T_1. This is the definition of case 3, so the new tree is [3:i-1].

2. i > 1: Adding a child to r_1 which is not in S will still leave a powerful alliance which is not minimal in T_1, so this combination is [4:i-1].

Finally, consider the combination of a subtree T_1 of case [6:i] with a subtree T_2 of case [5:j]. Because a powerful alliance exists somewhere in T_2, a powerful alliance will still exist in the combined tree at a level below the children of the root, so the new tree is [5:i-1].

3 Global Powerful Alliances in Trees

When trying to create a set S which is a global powerful alliance, g.p.a., in a tree T, there are fewer cases to consider. Again, r is the root of the tree T. There are only 4 cases:

[1:i] = $\{r \in S, S \cap T$ is a g. p. a. in $T\}$ /* i $\geq$ 0 */
[2:i] = $\{r \in S, S \cap T$ is not a g. p. a. in T, but adding neighbors in S to r can create a g. p. a.$\}$ /* i < 0 */
[3:i] = $\{r \notin S, S \cap T$ is a g. p. a. in $T\}$ /* i $\geq$ 0 */
[4:i] = $\{r \notin S, S \cap T$ is not a g. p. a. in T, but adding neighbors in S to r can create a g. p. a.$\}$ /* i < 0 */

Once these cases have been defined, we can consider the composition of a tree T from these subtrees. At each point we combine a subtree T_1 rooted at r_1 and g.p.a. S_1 with a subtree T_2 rooted at r_2 and g.p.a. S_2 to produce a new tree T' = $T_1 \diamond T_2$ with root r_1 and g..a. $S_1 \cup S_2$. We must consider all possible combinations of cases [a:i] and [b:j] for 1≤ a,b ≤ 5. If a particular combination cannot occur when producing a minimum cardinality global powerful alliance, then this combination is marked with an 'X'.

	[1:j]	[2:j]	[3:j]	[4:j]
[1:i]	[1:i+1]	if (j = -1) then [1:i+1] else X	if (i > 0) then [1:i-1] else [2:i-1]	if (j = -1) then {if (i > 0) then [1:i-1] else [2:i-1} else X
[2:i]	if (j > 0) then { if (i = -1) then [1:i+1] else [2:i+1] } else X	if (j = -1) then { if (i = -1) then [1:i+1] else [2:i+1]} else X	[2:i-1]	if (j = -1) then [2:i-1] else X
[3:i]	if (j > 0) then [3:i+1] else X	X	if (j > 0) then { if (i > 0) then [3:i-1] else [4:i-1] } else X	X
[4:i]	if (j > 0) then { if (i = -1) then [3:i+1] else [4:i+1]} else X	X	if (j > 0) then [4:i-1] else X	X

Table 2. Compositions for global powerful alliances.

From 2, we obtain the following recurrence relations:
[1:i+1] = [1:i]◇[1:j] ∪ [1:i]◇[2:j] ∪ [2:i]◇[1:j] ∪ [2:i]◇[2:j]
[1:i-1] = [1:i]◇[3:j] ∪ [1:i]◇[4:j]
[2:i+1] = [2:i]◇[1:j] ∪ [2:i]◇[2:j]
[2:i-1] = [2:i]◇[3:j] ∪ [2:i]◇[4:j] ∪ [1:i]◇[3:j] ∪ [1:i]◇[4:j]
[3:i+1] = [3:i]◇[1:j] ∪ [4:i]◇[1:j]
[3:i-1] = [3:i]◇[3:j]
[4:i+1] = [4:i]◇[1:j]
[4:i-1] = [3:i]◇[3:j] ∪ [4:i]◇[3:j]
As with non-global powerful alliances, the initial values for each case in global powerful alliances combine to form the initial vector for the algorithm. We use either the number of vertices in the set or ∞ to represent a state which cannot logically exist. When combining subtrees, we want to minimize the number of vertices in the set. The only cases which logically can exist initially are case 1, with a value of 1, and case 4, with a value of 0. This means that the initial vector is [1, ∞, ∞, 0]. As the last step to set up the tree for the algorithm, we create a parent array for the tree. This means that we have an array which stores, for each vertex, the parent of that vertex, with the roots parent being ∞. After execution of the algorithm, we need to determine the answer. Two states can determine a global powerful alliance. These are [3:x] where x ≥ 0, or [1:(x>0)]. The minimum value from the final vector in these two cases is the size of the minimum global

powerful alliance for that tree.
As before, we will only discus a couple of entries in 2. We will examine two entries which are different from those examined in the powerful alliances section: [3:i]◇[2:j] and [4:i]◇[1:j].
First, consider the combination of a subtree T_1 of case [3:i] with a subtree of case [2:j]. A subtree of case [3:i] will have i $\geq$ 0 more children of the root in the set S than not in S. A subtree of type [2:j] will have the absolute value of j +1 more children of the root not in the set S than are in the set S. This combination is invalid because this is the last composition involving the [2:j] tree T_2, and it doesnt have a global powerful alliance. Adding another vertex not in S adjacent to the root r_2 of T_2 is not going to make a global powerful alliance. This is therefore, an invalid combination, which is denoted with an X.
Next, consider the composition of a subtree T_1 of case [4:i] with a subtree T_2 of case [1:j]. A subtree of case [4:i] will have i $\geq$ 0 more children of the root not in the set S than in the set S. A subtree of case [1: j] will have j-1 more children of the root in the set S than not in the set S. For this combination, if j = 0, then combining with a subtree of type [4:i] will add a parent of [1:j] which is not in S, which means that a global powerful alliance will no longer exist. So we consider j > 0. There are 2 options:

1. i = -1: [1:j] will still be a global powerful alliance with [4:i] and [4:i] will now be a global powerful alliance, so it becomes a [3:i+1].
2. i < -1: [1:j] will still be a global powerful alliance when connected to [4:i]. However, [4:i] will not be a global powerful alliance, so it becomes [4:i+1].

4 Complexity of Alliances

In this section we establish the NP-completeness of the following two decision problems:

DEFENSIVE ALLIANCE
INSTANCE: Graph G = (V, E), positive integer k < $|V|$.
QUESTION: Does G have a defensive alliance of size at most k?

POWERFUL ALLIANCE
INSTANCE: Graph G = (V, E), positive integer k < $|V|$.
QUESTION: Does G have a powerful alliance of size at most k?

4.1 Defensive Alliance Complexity

THEOREM: DEFENSIVE ALLIANCE is NP- complete, even when restricted to split, chordal, or bipartite graphs. PROOF: DEFENSIVE ALLIANCE is clearly in NP. A set S of size at most k, could be given as a witness to a yes instance and verified in O(E) time to be a defensive alliance. In order to show that DEFENSIVE ALLIANCE is NP-complete we construct a transformation from the following well-known NP-complete problem.

HITTING SET
INSTANCE: Set $X = \{x_1, x_2, \dots x_n\}$, Collection $C = \{C_1, C_2, \dots C_m\}$ of subsets of X, positive integer $k < |X|$.
QUESTION: Does there exist $Y \subset X$ with $|Y| \geq k$ such that Y contains at least one element from each subset in C?

It is known that HITTING SET remains NP-complete even when $|C_i| = 2$, $\forall C_i \in C$. We will use this restricted form. Since we want each element to appear in at least two subsets, we can repeat any subset that contains an element that appears in only one subset, and if some element appears in no subset, then w.l.o.g. we can remove it from X.
Transformation: Let X, C, k be an arbitrary instance of the restricted HITTING SET problem.
Create an instance DA(G, k) of DEFENSIVE ALLIANCE in this way:

1. For each $x_i \in X$, create a singleton vertex x_i.
2. For |C| = m, create 2m vertices labeled $c_1, c_2, \dots, c_m, c_{m+1}, \dots, c_{2m}$ and form a clique among these 2m vertices.
3. Add 2k+1 independent vertices: $z_1, z_2, \dots, z_{2k}, z_{2k+1}$. Add all possible edges between the z vertices and the second half of the c vertices $c_{m+1}, c_{m+2}, \dots, c_{2m}$.
4. For each vertex ci, 1 i m, if Ci = xr, xs then add edges cixr and cixs.
5. Set k = m+k (recall that |C| = m).

Example
$X = \{x_1, x_2, x_3, x_4, x_5\}$
$C = \{\{x_1, x_4\}, \{x_1, x_3\}, \{x_2, x_5\}, \{x_4, x_5\}, \{x_2, x_1\}, \{x_3, x_5\}\}$
$k = 2$

Note that there is a clique defined on the set of c vertices; all of these edges are not shown.
Claim: Given a hitting set Y of size at most k, the set $S = Y \cup \{c_1, c_2, \dots c_m\}$ is a defensive alliance of cardinality $\leq k$.
The x-vertices in S will have N[x] $\cap$ S = N[x], so they satisfy the definition of a defensive alliance. Each of the vertices $c_1, c_2, \dots, c_m$ has degree 2m-1+2 = 2m+1, so | N[c_i]| = 2m+2. Since all of the vertices $c_1, \dots, c_m$ are in S and at least one of

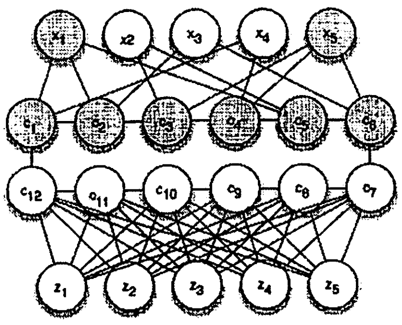

Figure 1. Defensive Alliance Example

the two x-vertices adjacent to each c_i is in S (from the hitting set) $|\,N[c_i]| \cap S\,| \geq$ m+1 for each c_i, $1 \leq i \leq$ m. Thus, S is a defensive alliance of cardinality $\leq$ k = m+k.

Conversely, suppose we have a defensive alliance S of size at most m+k.

1. The c-vertices $c_{m+1}, \ldots, c_{2m}$ have degree 2m-1+2k+1. Therefore $|N[c_i]|$ = 2m+2k+1, for m+1 $\leq$ i *leq* 2m. Therefore, their degree is too big for any of these vertices to be in S.

2. If a z vertex is in S, then some of its neighbors must be in S, but its only neighbors come from $c_{m+1}, \ldots, c_{2m}$, and we just ruled this out.

3. It follows that only the x-vertices and the vertices $c_1, \ldots, c_m$ can be elements of S. If an x-vertex is in S, then some c vertex must also be in S as well. But since the degree of each c_i vertex is 2m+1 at least m of the neighbors of a c-vertex in S must also be in S. Therefore either all of the vertices $\{c_1, c_2, \ldots, c_m\} \subseteq S$, or all but one of these vertices are in S.

   (a) $\{c_1, c_2, \ldots, c_m\} \subseteq S$. Then at most k x-vertices are in S since $|S| \leq k + m$. Each c_i in S must be adjacent to at least one x vertex, since at least m + 1 of the vertices in c_is closed neighborhood must be in S. Therefore the, at most k, x-vertices in S form a hitting set.

   (b) m-1 of the vertices $c_1, c_2, \ldots, c_m$ are in S. Each of the c-vertices in S, must have m+1 vertices chosen from their closed neighborhood. So both of the x-vertices they are adjacent to must be in S. There are at most k+1 x-vertices in S ($|S|$ - (m-1) = k+1) and these x-vertices hit each subset twice. Because each of these x-vertices appears in at least two subsets, selecting any k of these x-vertices produces a hitting set because each of the $c_1, c_2, \ldots, c_m$ will have at least one x-vertex in S.

Note that the graph G is a split graph, since the vertices can be partitioned into an independent set (the x and z vertices) and a clique (the c-vertices). Split graphs are also chordal. Thus DEFENSIVE ALLIANCE remains NP-complete even when restricted to split or chordal graphs. The NP-completeness of DEFENSIVE ALLIANCE when restricted to bipartite graphs can be shown by making a complete bipartite graph from two independent sets $c_1, \ldots, c_m$ and $c_{m+1}, \ldots c_{2m}$, and making a few other minor changes (details omitted).

4.2 Powerful Alliance Complexity

THEOREM: POWERFUL ALLIANCE is NP-complete, even when restricted to bipartite graphs. PROOF: POWERFUL ALLIANCE is in NP. A set S of size at most k can be given as a witness to a yes instance and verified in $O(E)$ time to be a powerful alliance (both offensive and defensive). We construct a polynomial-time transformation from the following, well known NP-complete problem.
DOMINATING SET
INSTANCE: Graph $G = (V, E)$, positive integer $k \leq |V|$.
QUESTION: Does G have a dominating set of cardinality $\leq k$?
Given an instance $G = (V, E)$ of DOMINATING SET, we construct the following graph H. We first construct what is called the VV-graph, with vertices labeled $a_1, \ldots, a_n$ (A) and $v_1, \ldots, v_n$ (V) where v_i is adjacent to a_j and v_j is adjacent to a_i if i and j are adjacent in E. To this we add another set of n independent vertices labeled $b_1, \ldots, b_n$ (B) and form a VV+-graph with the A vertices. We then add an additional n vertices $c_1, \ldots, c_n$ (C) that form a VV-graph with both the V and B vertices. We add one additional vertex d_1, which is connected to each of the V vertices and add an additional 3n independent vertices, $e_1, \ldots, e_{3n}$ (E), that are only connected to vertex d_1. Let H denote the graph so constructed.

Example: n = 5 k = 2

Claim: Graph G of order n has a dominating set of size at most k if and only if there is a powerful alliance of size at most k+2n in H. Highlighted is a dominating set of size k = 2 for the original graph G and a powerful alliance of size 2+2*5 for the transformed graph H. Let deg(x) be the degree of vertex x in the original graph G. If G has a dominating set of size k, then there is a powerful alliance S of size k +2n as follows:

1. Choose all the B and C vertices added in steps one and two of the transformation. This gives us 2n vertices.

2. Choose the vertices corresponding to the dominating set from the A vertices. This is an additional k vertices.

This is a defensive alliance because the chosen vertices in S fall into one of three categories.

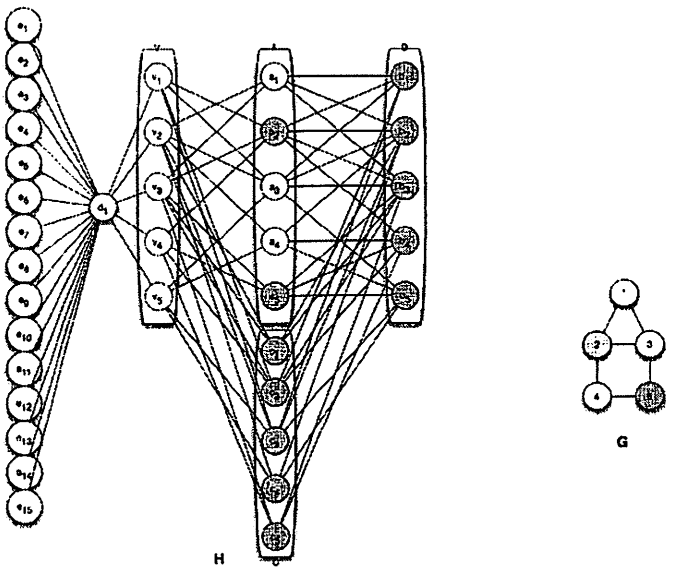

Figure 2. Powerful Alliance Example

1. The B vertices have as neighbors the C vertices corresponding to those they were connected in the original graph and the A vertices to which they were connected in the original graph, plus its corresponding vertex in the original graph because a_i is adjacent to b_i for $1 \leq i \leq n$. This means that, including itself, each of the vertices has 2deg(i)+1 neighbors in H and at least deg(i) +1 neighbors in S.
2. The C vertices c_i have deg(i) neighbors in B and deg(i) neighbors in V. This means that, including itself, each of the C vertices has deg(i) neighbors in S and at most deg(i) neighbors not in S.
3. The A vertices in the dominating set S have as neighbors the B vertices to which they were connected in G plus its corresponding vertex in the original graph. This means they have deg(i)+1 neighbors that are in S and deg(i) neighbors not in S. Thus, including itself, each of the A vertices in S has deg(i)+2 neighbors in S and deg(i) neighbors not in S.

Each of these cases combine to form a defensive alliance. The set S is also an offensive alliance because the vertices fall into one of 2 categories.

1. The A vertices which are not part of the dominating set in the original graph have as neighbors the B vertices to which they were connected in the original graph, plus its corresponding vertex in the original graph, which means

deg(i)+1 vertices that are in S, and the V vertices to which they were connected in the original graph, that is, deg(i) neighbors not in S. This means that, including itself, each of the vertices has deg(i)+1 neighbors in S and deg(i)+1 neighbors not in S.

2. The V vertices have as neighbors the C vertices to which they were connected in the original graph, which means deg(i) vertices that are in S, and the A vertices to which they were connected in the original graph. Since the dominating set of the original graph is in S, at least one of the A- neighbors must be in S and at most deg(i)-1 neighbors are not in S. Including itself, each of the vertices has at least deg(i)+1 neighbors in S and at most deg(i) neighbors not in S.

Because S is both an offensive and a defensive alliance, by definition, it is a powerful alliance of size at most k+2n. Conversely, if graph H has a powerful alliance S of size at most k+2n, then we must show that G has a dominating set D of size at most k:

1. If S contains any of the E or V vertices or vertex d_1, then S must contain at least half of them because the size of the closed neighborhood of d is 4n+1 and, in order to have a powerful alliance, if any of the vertices in N[d_1] are in S, at least half of these vertices must be in S. However, half of the neighbors of d1 would be $\lceil(4n+1)/2\rceil$ which is more than k+2n. Thus S has no vertices in E or V or d.

2. If S is a powerful alliance of H and $S \cap$ E = $\oslash$, S $\cap$ {d} = $\oslash$ and $S \cap$ V = $\oslash$, then it is easy to see that:

   (a) $S \subseteq$ A is not possible, (S does not dominate half of N[v_i] and is not a defensive alliance)

   (b) $S \subseteq$ B is not possible, (S does not dominate half of N[c_j] and is not a defensive alliance)

   (c) $S \subseteq$ C is not possible (S does not dominate half of N[v_i] or N[b_j] and is not a definsive alliance)

   (d) $S \subseteq$ A $\cup$ B is not possible (S does not dominate half of N[c_i] or N[v_j])

   (e) $S \subseteq$ A $\cup$ C is not possible (S does not dominate half of N[a_i] for $a_i \notin S$, and is not a defensive alliance)

   (f) $S \subseteq$ B $\cup$ C is not possible (S does not dominate half of N[v_i])

   Thus, $S \cap$ A $\neq \oslash$, $S \cap$ B $\neq \oslash$ and $S \cap$ C $\neq \oslash$.

3. If we take any of the C vertices then we need to take all of the C vertices and all the B vertices because each c_i has a closed neighborhood of size 2*deg(i)+1. So, if we take c_i, we need to take every b_j vertex to which

it is connected because we cant take any of the v-vertices to which it is connected. Also, if we take any b_j vertex which has a closed neighborhood of size 2*deg(j)+2, we need to take every c_i vertex to which it is connected. Because of these connections, we end up taking all of the C and B vertices.

4. Although S cannot contain any of the V vertices, they affect the powerful alliance because they are in the neighborhood of the C vertices that are in S. Therefore, there must be an offensive alliance which can dominate the V vertices. These vertices have closed neighborhoods of size 2*deg(i)+2. Because S contains all of the C vertices, there are deg(i) vertices in S adjacent to each of the v_i vertices. Therefore we must take at least one vertex from the A vertices that are connected to each of V vertices. If we take a dominating set of the graph G and associate that set with the A vertices, then at least one of the vertices connected to each of the V vertices will be in S. This means each v_i will have at least deg(i)+1 vertices in S and at most deg(i)+1 vertices not in S from its closed neighborhood.

Note that this graph is bipartite, which means that POWERFUL ALLIANCE is NP-complete even when restricted to bipartite graphs.

5 Open Questions

Having proved that DEFENSIVE ALLIANCE and POWERFUL ALLIANCE are NP-complete and having introduced linear time algorithms for powerful alliances and global powerful alliances in trees, it will be interesting to see if one can construct a polynomial time algorithm for circular-arc graphs or interval graphs.

6 Acknowledgments

The authors would like to thank Dr. Goddard at Clemson University for pointing out a flaw in the original powerful alliance algorithm which we have now fixed.

References

[1] R. C. Brigham, R. D. Dutton, T. W. Haynes, and S. T. Hedetniemi. Powerful alliances in graphs. submitted.

[2] R. C. Brigham, R. D. Dutton, and S. T. Hedetniemi. A sharp lower bound on the powerful alliance number of c_m x c_n. *Congr. Numer.*, 167:57–63, 2004.

[3] O. Favaron, G. Fricke, W. Goddard, S. M. Hedetniemi, S. T. Hedetniemi, P. Kristiansen, R. C. Laskar, and D. Skaggs. Offensive alliances in graphs. In I. Cicekli, N. K. Ciceklie, and E. Gelenbe, editors, *Proc. 17th Internat. Symp. Comput. Inform. Sci*, number ISCIS XVII, pages 298–302, Orlando, FL, USA, October 2002. CRC Press.

[4] O. Favaron, G. Fricke, W. Goddard, S. M. Hedetniemi, S. T. Hedetniemi, P. Kristiansen, R. C. Laskar, and D. Skaggs. Offensive alliances in graphs. *Discussiones Math. Graph Theory*, 24:263–275, 2002.

[5] G. H. Fricke, L. M. Lawson, T. W. Haynes, S. M. Hedetniemi, and S. T. Hedetniemi. A note on defensive alliances in graphs. *Bull. ICA*, 38:37–41, 2003.

[6] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. Global defensive allliances in graphs. In I. Cicekli, N. K. Cicekli, and E. Gelenbe, editors, *Proc. 17th Internat. Symp. Comput. Inform. Sci*, number ISCIS XVII, pages 303–307, Orlando, FL, USA, October 2002. CRC Press.

[7] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. Global defensive allliances in graphs. *Electr. J. Comb.*, 10(1):R47, December 2003. Global defensive allliances in graphs.

[8] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. A characterization of trees with eqal domination and global strong alliance number. *Util. Math.*, 66:105–119, 2004.

[9] P. Kristiansen, S. M. Hedetniemi, and S. T. Hedetniemi. Introduction to alliances in graphs. In I. Cicekli, N. K. Cicekli, and E. Gelenbe, editors, *Proc. 17th Internat. Symp. Comput. Inform. Sci*, volume ICIS XVII, pages 308–312, Orlando, FL, USA, October 2002. CRC Press.

[10] K. H. Shafique and R. D. Dutton. On satisfactory partitioning of graphs. *Congr. Numer.*, 154:183–194, 2002.

[11] T. V. Wimer, S. T. Hedetniemi, and R. C. Laskar. A methodology for constructing linear graph algorithms. *Congr. Numer.*, 50:43–60, 1985.