

Weighted Alliances in Graphs

Lindsay H. Jamieson and Brian C. Dean
School of Computing
Clemson University
Clemson, SC, USA

lch@cs.clemson.edu, bcdean@cs.clemson.edu

Abstract

The concept of an alliance in a graph was first presented in [9] in 2001 and subsequently studied by many others in the algorithmic and graph theory communities. In this paper, we discuss algorithmic and complexity results for several natural weighted generalizations of alliance problems.

1 Introduction

Let $G = (V, E)$ be a graph with n vertices and m edges and let $N[v]$ denote the closed neighborhood of vertex v . We denote by $N[S] = \cup_{v \in S} N[v]$ and $\partial(S) = N[S] - S$ the closed neighborhood and the boundary of a set $S \subseteq V$, and we let $\rho_S(v) = |N[v] \cap S|/|N[v]|$ denote the *local density* of S around v . An alliance in G is a non-empty subset of vertices $S \subseteq V$ that is sufficiently dense and numerous so that it is fortified to withstand a “military conflict” with $\partial(S)$. We say an alliance S is *defensive* if $\rho_S(v) \geq 1/2$ for all $v \in S$, *offensive* if $\rho_S(v) \geq 1/2$ for all $v \in \partial(S)$, and *powerful* if it is both defensive and offensive. An alliance S is *strong* if it satisfies these definitions with strict inequalities, and *global* if $N[S] = V$ (i.e., if S is also a dominating set).

Since alliances were first introduced in 2001 by Kristiansen, Hedetniemi and Hedetniemi [9], several authors have studied their mathematical properties as well as the complexity of computing minimum-cardinality alliances in various types of graphs [1, 2, 3, 5, 4, 6, 7]. In particular, it is NP-hard to compute a minimum-cardinality defensive or powerful alliance even in bipartite graphs (and the same is true if we seek a strong or global alliance), and one can compute a minimum-cardinality (strong or global, if desired) defensive, offensive, or powerful alliance in polynomial time on a tree using dynamic programming. The only missing piece in the complexity landscape is the minimum-cardinality non-global offensive alliance problem, whose complexity remains open even on a general graph (the global variant is NP hard [1]).

In this paper, we introduce several natural weighted generalizations of alliance problems and study their algorithmic complexity. Let $w : V \rightarrow \mathbb{N}$ be a non-negative integer weighting of the vertices in V , and let $w(S) = \sum_{v \in S} w(v)$. By setting $\rho_S(v) = w(N[v] \cap S)/w(N[v])$, we can define weighted (strong or global) defensive, offensive, and powerful alliances using exactly the same definitions as above. Our prior notion of an unweighted alliance now corresponds to the special case where $w(v) = 1$ for every $v \in V$.

We can broadly classify (weighted) alliance problems into two main groups depending on whether or not we seek a global alliance. In the global case, we typically seek a powerful alliance that can maintain control over the entire graph. In the “local” case, are interested in simply finding the smallest piece of a graph that one can possibly occupy in a defensive or offensive capacity. In both cases, we can ask for a *minimum-cardinality* weighted alliance that meets our desired needs, or we can associate non-negative costs $c : V \rightarrow \mathbb{N}$ with vertices and ask for a *minimum-cost* weighted alliance. Both problems are NP-hard in general graphs since they generalize the NP-hard unweighted alliance problem.

However, one of the main results of this paper is that the minimum-cost problem remains NP-hard even on a star, while the minimum-cardinality problem can be solved in polynomial time on a tree.

The remainder of this paper is structured as follows. Section 2 proves that the problem of finding a minimum-cost weighted alliance (of any type) is NP-hard even on a star. In Section 3 we discuss algorithmic results, focusing on special cases of our alliance problems that can be solved in polynomial time.

2 Complexity of Weighted Defensive and Powerful Alliances

All of the reductions in this section start with the NP-complete SUBSET-SUM problem:

INSTANCE: Finite set A , integer weight $w(a) > 0$ for each $a \in A$, integer $B > 0$.

QUESTION: Is there a subset $A' \subseteq A$ such that $w(A') = B$?

Note that SUBSET-SUM remains NP-complete even when $B > w(A)/2$, and also when $B = w(A)/2$ (this second case is known as the PARTITION problem).

Theorem 1. *The problem of computing a (global) minimum-cost weighted powerful alliance is NP-hard even restricted to stars.*

Proof. Consider an n -element instance of SUBSET-SUM, and assume $B > w(A)/2$. Create an $(n + 2)$ -vertex star G with the center vertex having weight $w(A)$, one leaf having weight $w(A)$ and the remaining n leaves having weights $w(a)$ for each $a \in A$. Let the cost of each vertex be equal to its weight.

We claim that G has a minimum-cost weighted powerful alliance of cost $w(A) + B$ if and only if we started with a “yes” instance of SUBSET-SUM. Suppose there exists a subset $A' \subseteq A$ with $w(A') = B$. In this case, we can form a powerful alliance S in G by taking the center vertex along with the leaves corresponding to A' . We know S is defensive since for the center vertex v we have $\rho_S(v) = (w(A) + B)/(3w(A)) \geq 1/2$, and it is offensive since clearly $\rho_S(v) \geq 1/2$ for every leaf $v \in \partial(S)$. On the other hand, let S be a minimum-cost powerful alliance of G with cost $w(A) + B$. Note that S must include the center vertex or it cannot be powerful. Consequently, S cannot include the leaf of weight $w(A)$, or else its cost would be too high. Hence, S includes a set of leaves of weight exactly equal to B , giving us a solution to our original SUBSET-SUM instance. Since S is a global alliance, this argument applies to both the global and non-global minimum-cost weighted powerful alliance problems. \square

Slight variations of this construction can be used to establish the complexity of weighted defensive and offensive alliances in stars.

Theorem 2. *The problem of computing a (global) minimum-cost weighted defensive alliance is NP-hard even restricted to stars.*

Proof. Given an n -element instance of SUBSET-SUM with $B > w(A)/2$, construct an $(n + 2)$ -vertex star as before, only set the weight of the exceptional leaf to $2(w(A) - B) < w(A)$. We claim that G has a minimum-cost weighted powerful alliance of cost $2w(A) - B$ if and only if we started with a “yes” instance of SUBSET-SUM. Suppose there exists a subset $A' \subseteq A$ with $w(A') = B$. In this case, we can form a defensive alliance S of cost $2w(A) - B$ in G by taking the center vertex, and the leaves corresponding to $A - A'$. We know S is defensive since $\rho_S(v) = (2w(A) - B)/(4w(A) - 2B) = 1/2$ for the center vertex v . On the other hand, let S be a minimum-cost defensive alliance of G with cost $2w(A) - B$. Note that any defensive alliance must include the center vertex, since the weight of the center vertex is larger than the weight of any leaf. Consequently, S cannot include the leaf of weight $2(w(A) - B)$, or else its cost would be too high. Hence, S includes a set of leaves of weight exactly equal to $w(A) - B$, giving us the complement of a solution to our original SUBSET-SUM instance. \square

Theorem 3. *The problem of computing a minimum-cost weighted offensive alliance is NP-hard even restricted to stars.*

Proof. Consider an n -element instance of the PARTITION problem. Construct an $(n + 1)$ -vertex star graph G with zero weight in the center and weights $w(a)$ for $a \in A$ on each of the n leaves. We claim that G has a minimum-cost

weighted powerful alliance of cost $w(A)/2$ if and only if we started with a “yes” instance of PARTITION. Suppose there exists a subset $A' \subseteq A$ with $w(A') = w(A)/2$. By taking the subset S corresponding to the leaves in A (and not the center), we have an offensive alliance of cost $w(A)/2$. On the other hand, let S be a minimum-cost offensive alliance of G with cost $w(A)/2$. Note that since S is offensive it must *not* include the center vertex; otherwise, it would necessarily contain every vertex in the star, and its cost would be too high. Therefore, the leaves in S correspond to a subset A' of elements with $w(A') = w(A)/2$. \square

Interestingly, the problem of computing a *global* minimum-cost offensive alliance is trivially solvable in polynomial time on a star. Here, an optimal solution must either consist of all the leaves and not the center, or the center plus every leaf having larger weight than the center. However, we can show the following.

Theorem 4. *The problem of computing a minimum-cost global weighted offensive alliance is NP-hard even restricted to trees.*

Proof. We use the same construction as in the preceding proof, except we attach n additional vertices of weight $w(A)$, one to each leaf. These extra vertices must be selected as part of any global offensive alliance, so now we claim (according to the same argument as before) that our tree has a minimum-cost global weighted powerful alliance of cost $w(A)/2 + nw(A)$ if and only if we started with a “yes” instance of SUBSET-SUM. \square

3 Algorithmic Results

Since minimum-cost alliance problems are NP-hard even on stars, we can only hope to solve them in polynomial time for very simple special cases. For example, on a path we can easily solve any of our weighted alliance problems in linear time using a straightforward dynamic program. In fact, for the case of a weighted defensive alliance on a path, it suffices to consider only each vertex and each pair of consecutive vertices, and to take whichever of these is a defensive alliance of minimum cost:

Lemma 1. *One can always find a minimum-cost weighted defensive alliance on a path consisting of either a single vertex or two adjacent vertices.*

Proof. Let S be a minimum-cost weighted defensive alliance of minimum cardinality. We can assume S is connected, since otherwise we can retain only one connected component in S and obtain a solution of strictly smaller cardinality and no larger cost. Therefore, suppose that S is a subpath starting with three consecutive vertices a , b , and c . Since $S - \{a\}$ is not defensive, we have $w(a) > w(b) + w(c)$, and since $\{a, b\}$ is not defensive, we know that $w(c) > w(a) + w(b)$. Adding these together, we have $0 > 2w(b)$, contradicting the assumption that $w(b) \geq 0$. \square

Using straightforward dynamic programming techniques, we can also construct a polynomial-time (in fact, linear-time) algorithm for computing a minimum-cost (global) weighted alliance on a cycle, a tree of bounded degree, or a graph of bounded pathwidth. On trees, since a straightforward dynamic programming algorithm runs in pseudo-polynomial time, we can obtain a polynomial running time for the special case where vertex weights are bounded by a polynomial function of n ; this includes unweighted alliance problems as a special case, for which polynomial-time algorithms on trees are discussed in [6].

We now turn our attention to the minimum-cardinality (global) weighted defensive, offensive, and powerful alliance problems on a tree, and we show how to solve these in polynomial time. Specifically, we describe here a $O(n^3)$ algorithm for computing a minimum-cardinality defensive weighted alliance on a tree. With similar approaches, one can also compute minimum-cardinality weighted global defensive, (global) offensive, or (global) powerful alliances.

Consider a rooted tree $T = (V, E)$. Let T_v denote the subtree rooted at vertex v , and let T'_v denote T_v with v 's parent attached. Our dynamic programming formulation is as follows: let $A^+_+(v, k)$ be true if and only if there exists a subset $S \subseteq V(T'_v)$ for which $|S \cap V(T_v)| = k$, $\rho_S(v) \geq 1/2$ for all $v \in V(T_v) \cap S$, and where both v and v 's parent belong to S . We define $A^-_+(v, k)$ the same way, except here v belongs to S but not v 's parent. Similarly, we define $A_- (v, k)$ such that v does not belong to S (so the superscript and subscript indicate whether v 's parent and/or v belong to S). We denote by $A^+_+(v)$ the $(n + 1)$ -component vector $\langle A^+_+(v, 0), \dots, A^+_+(v, n) \rangle$; the vectors $A^-_+(v)$ and $A_- (v)$ are

similarly defined. Our dynamic programming algorithm performs a postorder scan through T and computes $A_+^+(v)$, $A_+^-(v)$, and $A_-(v)$ for each vertex v we encounter in sequence.

Consider now the computation of $A_-(v)$ for a particular vertex v , where v has r children $u_1 \dots u_r$. We claim that $A_-(v, k)$ should be true if and only if we can find non-negative integers $k_1 \dots k_r$ summing to k for which

$$\bigwedge_{i=1}^r A_-(u_i, k_i) \vee A_+^-(u_i, k_i)$$

is true. To determine this efficiently, let $B(x, y, k)$ be true if we can find non-negative integers $k_x \dots k_y$ summing to k for which

$$\bigwedge_{i=x}^y A_-(u_i, k_i) \vee A_+^-(u_i, k_i)$$

is true, and let $B(x, y)$ denote the vector $\langle B(x, y, 0), \dots, B(x, y, n) \rangle$. To compute $A_-(v) = B(1, r)$, we first recursively compute $B(1, r-1)$, and then we have

$$\begin{aligned} B(1, r, k) &= \bigvee_{k'=0}^k B(1, r-1, k') \wedge B(r, r, k-k') \\ &= \bigvee_{k'=0}^k B(1, r-1, k') \wedge (A_-(u_r, k-k') \vee A_+^-(u_r, k-k')). \end{aligned}$$

A direct application of this formula to compute of $A_-(v)$ takes $O(rn^2)$ time (one can actually speed up the running time to $O(rn \log n)$ using the Fast Fourier Transform, but this trick seems not to apply to the next part of our algorithm, and therefore saves us nothing in the long run). Since we spend $O(rn^2)$ time on a vertex with r children, we spend $O(n^3)$ time on the entire tree.

To compute $A_+^-(v)$ and $A_+^+(v)$, we use a similar approach. Let us now define $B(x, y, k)$ as the maximum weight $w(\{u_i : i \in S\})$ of a subset $S \subseteq \{x, \dots, y\}$ for which we can find non-negative integers $k_x \dots k_y$ summing to k such that

$$\left(\bigwedge_{i \in S} A_+^+(u_i, k_i) \right) \wedge \left(\bigwedge_{i \in \{x, \dots, y\} - S} A_-(u_i, k_i) \right)$$

is true. We then set $A_+^-(v, k)$ to true if

$$\frac{B(1, r, k-1) + w(v)}{w(N[v])} \geq 1/2,$$

and we set $A_+^+(v, k)$ to true if

$$\frac{B(1, r, k-1) + w(v) + w(\text{parent}(v))}{w(N[v])} \geq 1/2.$$

Using essentially the same approach as above to compute $B(1, r)$ in $O(rn^2)$ time, we obtain a total running time of $O(n^3)$ for the entire algorithm. After termination, the minimum cardinality of a weighted defensive alliance can be determined by finding the minimum value of k such that $A_-(r, k) \vee A_+^-(r, k)$ is true, where r denotes the root of the tree.

4 Concluding Remarks

There are several other natural alliance questions one might want to consider. For example, as a middle ground between the ‘‘global’’ and ‘‘local’’ alliance problems, we can associate a fortification requirement r_v with each vertex v and ask for a minimum-cardinality or minimum-cost subset of vertices S such that $w(N[v] \cap S) \geq r_v$ for each $v \in V$. Here, our goal is generally to provide protection for a collection of strategic locations in a graph, some of

which may require more protection than others. This problem is already somewhat well-studied, since it is easily shown to be equivalent to the NP-hard column-restricted covering integer programming (CCIP) problem; see [8] for further information. Since it generalizes the NP-hard knapsack cover problem, this problem is NP-hard even on a star. Its approximability on trees in particular may be an interesting problem to consider for future research.

We also wish to introduce the slightly more general notion of an α -alliance: we say S is a defensive α -alliance if $\rho_S(v) \geq \alpha$ for all $v \in V$; offensive, powerful, strong, and global α -alliances are defined similarly. Based on this notion, we propose the following natural problem variant that we call the *maximum-control* (weighted) alliance problem. Here, we are given a budget B and asked to compute a global α -alliance having cost or cardinality at most B , with a goal of maximizing α . This problem inherits the complexity from the minimum cardinality/cost global weighted alliance problem, since we can solve that problem using an algorithm for the maximum-control problem by binary searching on B . To the best of our knowledge, nothing algorithmic is yet known about this problem, even in the unweighted case.

References

- [1] A. Cami, H. Balakrishnan, N. Deo, and R. D. Dutton. On the complexity of finding optimal global alliances. *J. Combin. Math. Combin. Comput.*, 58:23–31, 2006.
- [2] 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.
- [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. Cicekli, and E. Gelenbe, editors, *Proc. 17th Internat. Symp. Comput. Inform. Sci.*, volume ISCS XVII, pages 298–302, Orlando, FL, USA, October 2002. CRC Press.
- [4] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. Global defensive alliances in graphs. In I. Cicekli, N. K. Cicekli, and E. Gelenbe, editors, *Proc. 17th Internat. Symp. Comput. Inform. Sci.*, pages 303–307, Orlando, FL, USA, October 2002. CRC Press.
- [5] T. W. Haynes, S. T. Hedetniemi, and M. A. Henning. Global defensive alliances in graphs. *Electr. J. Comb.*, 10(1):R47, December 2003.
- [6] L. H. Jamieson. *Algorithms and Complexity for Alliances and Weighted Alliances of Different Types*. PhD thesis, Clemson University, 2007.
- [7] L. H. Jamieson, S. T. Hedetniemi, and A. A. McRae. The algorithmic complexity of alliances in graphs. *J. Combin. Math. Combin. Comput.*, to appear, 2007.
- [8] S. Kolliopoulos. Approximating covering integer programs with multiset constraints, 2000. Submitted for journal publication.
- [9] P. Kristiansen, S. M. Hedetniemi, and S. T. Hedetniemi. Introduction to alliances in graphs. In I. Cicekli, N. 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.