SURIEM 2016 Final Report: Games on Graphs

Julie Anne Bowman, Arthur Diep-Nguyen,
Rashmika Goswami, Dylan King, Nicholas Lindell,
Emily Olson, Robert W. Bell

July 14, 2016

1 Introduction

The game of Cops and Robbers in the classical sense, first introduced in the 1980s, is a two-player, perfect information game played on the vertices of a combinatorial graph. The cop player, C, controls a squad of cop pawns, while the robber player, R, a single pawn. To start, C places her cops on several vertices of the graph, and R places his; the two then take turns by moving any number of their pawns along edges to neighboring vertices or by passing. The cop player wins if, at the end of any turn, a cop pawn and the robber pawn occupy the same vertex; if the robber can prevent this indefinitely, he wins. Of mathematical interest is the least number of cops required to guarantee victory for C on a given graph G, called the *cop number* $c(G)$.

An interesting extension of Cops and Robbers is to allow play on infinite graphs. Obviously, with an infinite escape route, the robber should always be able to start far away and keep avoiding capture by moving away whenever cops threaten his position; to eliminate this triviality, a different winning condition is necessary. Florian Lehner introduced a mathematically viable winning condition for infinite graphs in [reference]: the cops *weakly win* if the robber cannot visit any vertex infinitely often. Lehner showed that, under this definition, any infinite constructible graph G has a *weak cop number* $wc(G) = 1$. Here, we investigate the weak cop number of various tilings of the plane and extend some theorems about the cop number of finite graphs with particular properties to the weak cop number of infinite graphs. [I don't like how this sentence sounds–modifiers/prepositions kind of ambiguous. -N]

We also study a variation of Cops and Robbers called Seepage, first described by Clarke, et al. in [1]. This game is played on a directed acyclic graph with a single source by two players, Sludge and Green, who alternately claim vertices of the graph. Once a vertex has been claimed, it belongs to the player that claims it for the rest of the game, and cannot be claimed by the opponent. Sludge begins by claiming, or 'contaminating', the source. Afterwards, Green can claim, or 'protect', any vertex on the graph, while Sludge can contaminate any vertex adjacent to an already contaminated vertex. Sludge is said to win if any sink is contaminated; otherwise, Green wins. The generalized version of this game allows Green to claim multiple vertices each turn. Clarke, et al. defined the green number, $gr(G) = k$, to be the least number k such that Green is able to win by making at most k moves per turn. A graph is said to be green-win if $gr(G) = 1$, sludge-win if $gr(G) > 1$, and k-green-win if $gr(G) = k$. In their paper, Clarke, et al. characterized green-win and k-green-win rooted trees, providing a polynomial time algorithm for determining if $gr(T) = k$, for a rooted tree T. Additionally, they discussed the green numbers of graphs formed by taking cartesian products and truncating, and proposed a method of taking a tensor product while maintaining the acyclic and single-source conditions of the graphs. We introduce a more generalized algorithm that determines if $gr(G) = k$ for any directed acyclic graph, and introduce some methods to reduce the number of vertices and edges of a graph while preserving the green number. We also attempt to better characterize the green numbers of tensor products of graphs.

2 Tilings

2.1 Gluing Methods

2.2 Isometric Path Method

2.3 Theorem 5 expanded

2.4 Summary/Computations

3 Constructability

3.1 wc = 1

4 Seepage

4.1 Background/Notation

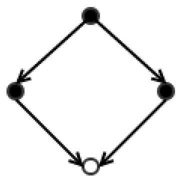
The game of Seepage is played only on a directed, acyclic graph with a single source. Therefore, for the remainder of this report, we will assume all graphs to be simple digraphs that are acyclic and only have one source. Similarly, when discussing rooted trees, we will assume that the source and root are the same. The "distance" in a graph G from vertex a to vertex b, $d(a, b)$ will be the length of the shortest directed path from a to b, and the distance of vertex v will simply refer to the distance from the source to v. Additionally, since all graphs are assumed to be directed, we will use "path" to refer to a directed path. "Parents" of vertex v are considered to be the incoming neighbors, or $N^-(v)$, while "children" refers to outgoing neighbors, or $N^+(v)$.

4.2 Reduction Methods

This section contains 6 propositions on ways to reduce a directed graph in Seepage. Here are a few definitions that will be used throughout this section.

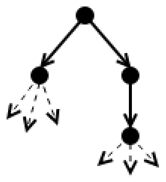
- **Reducing**–A way to reduce the number of vertices in a graph such that the winner of the graph remains unchanged.
- **Sink**–A vertex that has no outgoing edges.
- **Loose Sink**–A sink that has only one edge going into it.
- **Shortest Sink**–A sink that has the shortest distance from the root.

Proposition 4.1 *If the number of sinks is less than or equal to the distance of the shortest sink then the graph is g-win.*



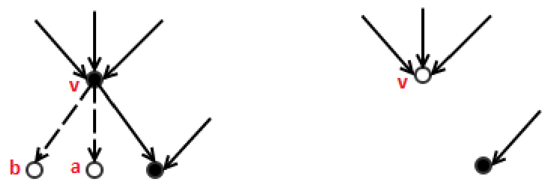
Proof. Let d equal to the shortest path to the closest sink. Green will have at least d turns before r contaminates a sink. Therefore when the number of sinks is less than or equal to d, C can protect all the sinks before S gets to them.

Proposition 4.2 *If a vertex contains only one child, then removing the vertex and all edges going in and out of the vertex will properly reduce the graph.*



Proof. Let a be a vertex and b is the only child of a. If s contaminates a then g can protect b thus cutting off s' path. This means that s would never want to contaminate a, so removing a would not affect s or g's strategy.

Proposition 4.3 *If a vertex has two or more loose sinks, then removing all edges leading out of the vertex and making it a sink does not change whether it is g-win or s-win.*



Proof. Let v be a vertex with 2 sinks a and b. If v becomes contaminated then g will have to protect either a or b. However, s can contaminate the one not protected the next turn and thus win. That means that if s can ever contaminate v, s will win. We can see that g's strategy also does not change. If g needs to protect a and b to win, the move of protecting v also protects a and b. This means that g only cares about v in this instance. So removing a and b and making v a sink will not change the outcome of the game.

Proposition 4.4 *If the graph contains a sub-tree, check to see whether it is s-win or g-win:*

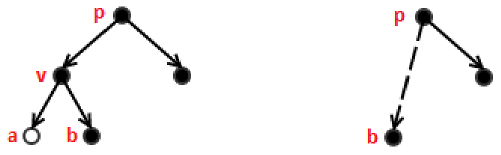
- **Subproposition 4.4.1** *If it is g-win, removing the entire sub-tree is a proper reduction of the graph.*

  Proof. Since the sub-tree is g-win, there is no way for s to win by reaching any of the sinks in the sub-tree. Therefore, it would not be in s' interest to contaminate the root of the tree. So removing the sub-tree and its root does not change the outcome of the game. The entire tree is gone from the graph after the root is taken since the only way to the vertices of a tree is through the root.

- **Subproposition 4.4.2** *If s-win, removing all of the sub-tree except the root will make a proper reduction. Make the root a sink.*

  Proof. We know this sub-tree is a s-win so if s ever gets to the root then he has won. This means if we make the root a sink it wouldn't make a difference in the whether the graph was s-win or g-win.

Proposition 4.5 *If a vertex has only two children and one is a sink. Then deleting the vertex and drawing a line from the parents of the vertex to the child that is not the sink will not change whether a g-win or s-win.*

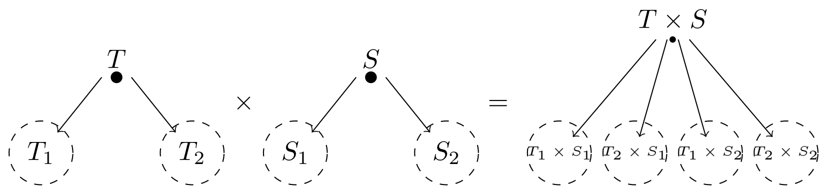


Proof. Let v be a vertex that has only 2 children, a and b, and a is a sink. If s contaminates v then g will be forced to protect a and s will contaminate b the next turn. This means that any parent of v would be able to get to b. So if we delete v and a and draw lines from v's parents to b, the graph would not change being s-win or g-win.

4.3 Graph Products

In this section we examine the products of different graphs and relate the types of the graphs to the types of the product. The tensor product in this case is defined to be the component of the tensor product that contains the product of the two sources of the original graphs. This maintains the connected, acyclic, and single-source conditions on the new graph. Clarke, et al. posed the following question concerning this type of graph product: "Characterize those rooted trees where $T \times S$ is green-win." We attempt to answer this qestion by first exploring those rooted trees where $T \times S$ is sludge-win. In addition, we begin to generalize these results for graphs.

First, it is important to note a special property of the tensor product of two rooted trees. We see that for rooted trees, the children of the root are themselves roots of distinct subtrees. Therefore, given rooted trees T with source v_t and S with source v_s, the children of the product $(v_t, v_s) \in V(T \times S)$ are the products of the children of v_t and v_s. To illustrate:



Furthermore, we know from [1] that a rooted tree is sludge-win if the source has at least two children who are roots of sludge-win rooted trees. This leads us to the following results.

Lemma 4.1 *Given a sludge-win rooted tree S and any rooted tree T, $T \times S$ is always sludge-win.*

Proof. We proceed by strong induction on the depth of S. Let S be the sludge-win rooted tree of depth 0, and T be any rooted tree of depth $d > 0$. The categorical product of $T \times S$ is simply S and therefore also sludge-win. Now suppose that S is a sludge-win tree of depth $n > 0$, and that the result holds for all trees of depth $k < n$. Since S is sludge-win and has depth $n > 0$, its root must have at least two children, S_1 and S_2, that are the roots of sludge-win subtrees of depth $n - 1$. Similarly, since T has depth $d > 0$, the root of T must have at least one child, T_1. In this case, the root of $T \times S$ will have at least two children that are the roots of products of the subtrees of S rooted at S_1 and S_2 with the subtree of T rooted at T_1. That is, the root of $T \times S$ will at least have as children $T_1 \times S_1$ and $T_1 \times S_2$. Since these two children are roots of products of sludge-win rooted trees with depth $n-1$, by the inductive assumption, they are roots of sludge-win subtrees of $T \times S$. Therefore, $T \times S$ must also be sludge-win.

Lemma 4.2 *For any green-win rooted tree S that is not a path, there exists a green win rooted tree T such that the categorical product, $T \times S$, is sludge-win.*

Proof. We construct a green-win rooted tree T that fits the criteria. If S is not a path, then there exists a vertex, v_0 with multiple children. Let the depth of v_0 be n. Construct T so that all vertices of depth $k \neq n$ have at least two children, and all vertices of depth n have exactly 1 child. It is clear that T is a green win tree since if a vertex of depth n is ever contaminated, its single child can be protected and thus the contamination will never spread below a depth of $n-1$. However, every vertex of $T \times S$ will have multiple children, because any vertex of depth k will be a product of vertices of depth k in T and S, at least one of which will have multiple children, thus causing the product vertex to have multiple children. Therefore, $T \times S$ will be sludge-win.

Lemma 4.3 *If a sludge-win graph, S, is truncated at distance d to create a new graph, S', such that only vertices of distance $\leq d$ are kept in the S', then S' will also be sludge win.*

Proof. Assume S' is green-win. In this case, it is possible to prevent Sludge from contaminating any sink at distance $\leq d$, or any vertex at distance d, which means Sludge cannot get to any vertex below distance d. Therefore, by playing the same strategy on S, it is possible to prevent Sludge from contaminating any sink at or above distance d or any sink below distance d, which means S is green-win. Since this is not the case, S' cannot be green-win.

Theorem 4.1 *Given a sludge-win graph S and any graph T, $T \times S$ is always sludge-win.*

Proof. Let P be any path in T from a source to a sink. We will consider just the product of $P \times S$. Since the tensor product of a graph S and a path of length d is simply S truncated at depth d, we see from Lemma 5.3 that $P \times S$ must be sludge-win. Furthermore, since $P \times S$ is a subgraph of $T \times S$ such that all of its sinks are also sinks in $T \times S$, this means that $T \times S$ must be sludge-win, by the theorem in [1].

4.4 Algorithm

Definition. A *wall* of a graph G is a set $w(G) \subset V(G)$ such that:

- the connected component of $G - w(G)$ containing the source and only counting as sinks those vertices which were already sinks of G is green-win

- for any $v \in w(G)$, the connected component of $G - w(G) + v$ containing the source and only counting as sinks those vertices which were already sinks of G is sludge-win.

Definition. Let g_0 be the vertex, if any, Green protects after Sludge's first move when playing a winning strategy on $G - w(G)$. A *complete wall* of a graph G, $c(G) = w(G) \cup \{g_0\}$.

Definition. Let a $w(G)$ be a wall and $c(G)$ be a complete wall such that $c(G) = w(G) \cup \{g_0\}$. We say $w(G)$ is a related wall of $c(G)$.

The following definition was originally given in [1].

Definition. Let G_v be the induced subgraph of G with the vertex set $V(G_v) = \{u \in V(G) \mid \text{there is a path from } v \text{ to } u\}$. [1]

Theorem 4.2 *If $v \in G$ has children $v_0, v_1, \ldots, v_n$, any complete wall $c(G_v)$ is equivalent to $\bigcup_{i=0}^{n} \chi_i$, where χ_i is* $\{v_i\}$ or $\{w(G_{v_i})\}$.

Proof. Consider a contaminated vertex v with only its complete wall protected. This means that Green is able to win no matter which vertex Sludge chooses to contaminate next. In other words, for all unprotected children $b_0, b_1, \ldots, b_n$ of v, G_{b_i} is green-win, so each $w(G_{b_i})$ must be protected. Therefore, for all children, v_i, of v, $v_i \notin c(G_v)$ and so [I removed the arrow] $w(G_{v_i}) \subseteq c(G_v)$. This ~~which~~ means that $\bigcup_{i=0}^{n} \chi_i \subseteq c(G_v)$.

Furthermore, we see that if $\exists x \in c(G_v)$ s.t. $x \notin \bigcup_{i=0}^{n} \chi_i$, then if $x = g_0$, $c(G_v) = w(G_v) \cup \{g_0\} = \bigcup_{i=0}^{n} \chi_i \cup \{x\}$, which means that $w(G_v) = \bigcup_{i=0}^{n} \chi_i$ is a related wall of the complete wall. But this cannot be a wall, since the connected component of $G_v - w(G_v) + u, u \in w(G_v)$ containing the source and only counting as sinks those vertices which were already sinks of G_v is green-win. We see that in the turn after Sludge contaminates v, Green can protect u, and then for every unprotected child b_i that Sludge contaminates, Green has already protected $w(G_{b_i})$, making all G_{b_i} green-win, and therefore making $G_v - w(G_v) + u$ green-win.

Similarly, if $x \neq g_0$, and WLOG $g_0 = v_0$ or $u \in w(G_{v_0})$, then the related wall is $w(G_v) = \bigcup_{i=1}^{n} \chi_i \cup \{x\}$. However, again, this cannot be a wall, since $G_v - w(G_v) + x$ would be green-win. In the turn after Sludge contaminates v, Green can protect g_0, and then for every unprotected child b_i that Sludge contaminates, Green has already protected $w(G_{b_i})$, making all G_{b_i} green-win, and therefore making $G_v - w(G_v) + x$ green-win. Therefore, $\nexists x \in c(G_v)$ s.t. $x \notin \bigcup_{i=0}^{n} \chi_i$, so $c(G_v) \subseteq \bigcup_{i=0}^{n} \chi_i$, which means that $c(G_v) = \bigcup_{i=0}^{n} \chi_i$.

Corollary 4.1 *A graph G is green-win if and only if it has the empty set as a wall.*

Proof. We see from above that G is green-win if and only if it has a wall protected. Since Green is unable to protect a wall before the game begins, G is green-win if and only if the empty set is a wall.

Using this theorem, we now present an algorithm to determine whether a directed acyclic graph is green-win or sludge-win. $W(G)$ denotes the set, $\{w(G)\}$, of walls of G, while $C(G)$ denotes the set, $\{c(G)\}$, of complete walls of G.

Algorithm 1: Seepage Algorithm

```
Data: $G = (V, E)$
Result: $True$ if $G$ is green win, $False$ otherwise
begin
    $\forall$ sinks $v, W(G_v) \leftarrow \emptyset, C(G_v) \leftarrow \emptyset$
    // Find $W(G_v), C(G_v)$ for all vertices
    while $\exists v$ s.t. $W(G_v), C(G_v)$ haven't yet been found do
        for $v \in V(G)$ s.t. $W(G_v), C(G_v)$ haven't yet been found, but
        $W(G_x), C(G_x)$ have been found $\forall x \in N^+(v)$ do
            // Build possible complete walls by unioning walls
               of children
            $C(G_v) \leftarrow \{\emptyset\}$
            for $x \in N^+(v)$ do
                for $S \in C(G_v)$ do
                    $C(G_v) \leftarrow C(G_v) - S$
                    $C(G_v) \leftarrow C(G_v) \cup \{S \cup \{x\}\}$
                    for $T \in W(G_x)$ do
                        $C(G_v) \leftarrow C(G_v) \cup \{S \cup T\}$
                    end
                end
            end
            // Find all possible walls using complete walls
            for $S \in C(G_v)$ do
                for $x \in S$ do
                    $W(G_v) \leftarrow W(G_v) \cup \{S - x\}$
                end
            end
            // Make sure $S \in W(G_v)$ are walls
            for $S \in W(G_v)$ do
                for $T \in W(G_v)$ do
                    if $(T \neq S) \wedge (T \subseteq S)$ then
                        $W(G_v) \leftarrow W(G_v) - S$
                    end
                end
            end
        end
    end
    // Check if source has empty set as wall
    if $\emptyset \in W(G)$ then
        return True
    else
        return False
    end
end
```

This can easily be extended to determine if a graph is k-green-win. Define a *k-wall* of vertex v to be a set $w_k(G) \subset V(G)$ such that:

- the connected component of $G - w_k(G)$ containing the source and only counting as sinks those vertices which were already sinks of G has green number k
- for any $v \in w_k(G)$, the connected component of $G - w_k(G) + v$ containing the source and only counting as sinks those vertices which were already sinks of G has green number $> k$.

Similarly, let $g_1, ..., g_k$ be the first k vertices Green protects after Sludge's first move when playing a winning strategy on $G - w_k(G)$. A *k-complete wall* of a graph G is $c_k(G) = w_k(G) \cup \{g_1, ..., g_k\}$. The analogous results for the k move game are as follows. The proofs are parallel to those for the original game.

Theorem 4.3 *If $v \in G$ has children $v_0, v_1, \ldots, v_n$, any k-complete wall $c_k(G_v)$ is equivalent to $\bigcup_{i=0}^{n} \chi_i$, where χ_i is either $\{v_i\}$* or *$\{w_k(G_{v_i})\}$.*

Corollary 4.2 *A graph G is k-green-win if and only if the empty set is a k-wall, but not a $(k-1)$-wall of the source.*

5 Future Directions

5.1 Seepage

It is clear that our algorithm, while accurate, must become much more efficient in order to be practical. To this end, we'd like to continue devising graph reductions in order to minimize actual runtime as much as possible. Furthermore, we'd like to reduce the complexity of the algorithm, if possible, and explore the bounds on the complexity of the algorithm. A closely related game, Firefighter, was shown to have an NP-complete determination problem for certain graphs, and it seems possible that this game might also [2]. On another note, we'd like to improve our characterization of green-win graph products, possibly enabling us to determine the outcome of larger graphs more efficiently by determining the outcomes of their factors.

References

[1] N. E. Clarke, S. Finbow, S. L. Fitzpatrick, M. E. Messenger, and R. J. Nowakowski. Seepage in directed acyclic graphs. *Australas. J. Combin.*, 43:91–102, 2009.

[2] Andrew King and Gary MacGillivray. The firefighter problem for cubic graphs. *Discrete Math.*, 310(3):614–621, 2010.