AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **43** (2009), Pages 91–102

Seepage in directed acyclic graphs

N. E. Clarke

Department of Mathematics and Statistics
Acadia University
Wolfville, Nova Scotia B4P 2R6
Canada
nancy.clarke@acadiau.ca

S. Finbow

Department of Mathematics, Statistics, and Computer Science
St. Francis Xavier University
Antigonish, Nova Scotia B3G 2W5
Canada
sfinbow@stfx.ca

S. L. Fitzpatrick*

Department of Mathematics and Statistics
University of Prince Edward Island
Charlottetown, Prince Edward Island C1A 4P3
Canada
sfitzpatrick@upei.ca

M. E. Messenger* R. J. Nowakowski*

Department of Mathematics and Statistics
Dalhousie University
Halifax, Nova Scotia B3H 3J5
Canada
mmessing@dal.ca rjn@mathstat.dal.ca

Abstract

In the firefighting and the graph searching problems, a contaminate spreads relatively quickly. We introduce a new model, on directed acyclic graphs, in which the contamination spreads slowly. The model was inspired by the efforts to stem the lava flow from the Eldfell volcano in

* Partially supported by grants from NSERC.

Iceland. The contamination starts at a source, only one vertex at a time is contaminated and for some fixed k, k vertices are protected. The slowness is indicated by the name 'seepage'. The object is to protect the sinks of the graph. We show that if a sink of the graph can be contaminated then at most one directed path need be contaminated. We also investigate the Cartesian product of directed paths. We show that for the product of 3 directed paths that is truncated to only vertices up to a distance of d from the source, if $d \geq 9$, then only one vertex need be protected on each turn to protect the sinks. We also present bounds for the Cartesian product of more than 3 paths.

1 Introduction

There have been many models and papers about how to deal with intruders or contamination on a graph. For example, searching for a potential intruder who is infinitely fast (equivalently, decontaminating a network) [1], cleaning a graph [10], pursuit evasion [12] and firefighting [9]. This paper considers a variant of firefighting inspired by actual events in Iceland, where the Eldfell volcano erupted and lava poured downhill toward a town and port, see [11, 13]. The town was defended by pouring water on the lava to solidify the leading edge and redirecting the flow.

For this paper, there is a directed acyclic graph with one *source* (in-degree 0) and many *sinks* (out-degree 0). At time $t = 0$, the pollution is detected at the source and then an unpolluted vertex is protected, i.e. made safe from being polluted, for all time. At time step $t > 0$, the pollution spreads from some polluted vertex to (exactly) one adjacent non-polluted vertex then another unpolluted vertex is protected. The pollution is said to win if it reaches a sink. In contrast to node searching and firefighting, the rate of contamination is very slow, hence we refer to this version as *seepage* and the situation as the *seepage game*.

To personify the situation, we consider this a game between two players, Sludge and Green. The sinks are on the edge of a lake and the aim of Sludge is to pollute the lake while Green wants to protect the lake. The number of vertices that are polluted is irrelevant.

It is interesting to note, that if Sludge can win then he can do so by having polluted only the vertices of a path; see Theorem 2.4. This fact simplifies the proofs of our subsequent results.

In Section 3, we give two simple characterizations of those trees in which Green can win. This is in contrast to firefighting. In firefighting, the fire spreads from all burning vertices to all unprotected vertices. On a rooted tree (so effectively, the tree is a directed tree) the fire breaks out at the root. Given a rooted tree T and a subset of vertices S, the S-FIRE problem [5] is to determine whether all the vertices of S can be protected from the fire. In [5], it is shown show that S-FIRE is NP-complete even when restricted to S being the set of leaves of a full rooted tree of maximum degree three. In contrast, the determination of those trees that Green can win and indeed, how many vertices Green needs to protect on each move, is polynomial. The

characterizations are in terms of substructures of the individual trees, see Corollary 3.4 and Theorem 3.7 respectively.

The Cartesian product gives rise to many familiar graphs, including hypercubes (Theorem 4.2). One of the earliest problems considered, and one that still brings insight, is whether Green can win on the following graphs: take the Cartesian product of three directed paths of length d and take H_d as the subgraph of vertices at distance at most d from the source, see Figure 1 (the sinks are indicated by white vertices). In Section 4, we identify all the 'truncated' Cartesian cubes, except for a handful which we leave for the reader. We also give bounds for the Cartesian products of more than 3 paths. In the last section, we present some problems.

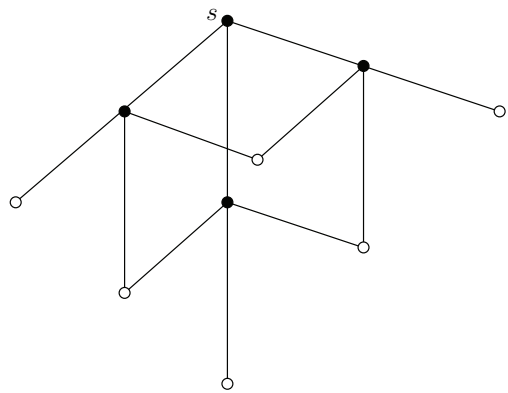

Figure 1: The subgraph of $P_3 \Box P_3 \Box P_3$ which contains vertices at most distance 2 from the source s.

This problem is reminiscent of the *Angel-&-Devil* problem [3]. Originally played on the infinite grid, the Devil blocks one square whilst the Angel of power k, who starts at the origin, can move to any unblocked square at distance no more than k away. This problem has been solved in 3-dimensions [4, 7] and now in 2-dimensions [2, 6, 8]. The graph for the Seepage problem then could be interpreted as the graph of possible moves for the Angel. However, Theorem 2.4 shows that if the Sludge (= Angel!) can win then he can win by polluting just a path. For the Angel this would mean always moving away from the origin. Conway [3] [page 8] calls such an Angel an *out-and-out fool* and he gives *Theorem 6.1. The Devil can catch an Out-and-Out Fool [on the square grid].*

2 General Results

Definition 2.1 *Let G be a directed acyclic graph and, for $x \in V(G)$, let G_x be the induced subgraph of G on $\{v \in V(G) \ : \$ there is a directed path from x to $v\}$.*

Definition 2.2 *A directed acyclic graph G (with one source) is called green-win if*

Green, protecting only one unpolluted vertex each time step can win; otherwise it is called sludge-win.

Definition 2.3 *Let G be a directed acyclic graph with one source. Let $gr(G)$ be the least number k, so that if Green may protect k vertices of G on each move, the Green has a winning strategy in the seepage game. A directed acyclic graph G (with one source) is called k-green-win if $gr(G) = k$.*

After a few moments thought, it is intuitively obvious that if Sludge can win then he need only pollute the vertices of a path and this is our next Theorem. Obviously, the path cannot be given in advance since it depends on the vertices that Green protects. In a sludge-win graph, we call a Sludge strategy that pollutes only the vertices of a path, against any strategy of Green, *efficient* and we say Sludge wins *efficiently.*

Theorem 2.4 (Single Directed Path.) *If G is a sludge-win graph then Sludge has an efficient strategy.*

Proof: The following Claim helps simplify the situation.

Claim 1: Let v be a polluted vertex with unpolluted children a and b where b is also a child of a. If Sludge has an efficient strategy starting with a, b then he has an efficient strategy starting with b.

Proof of Claim: If Sludge pollutes b before a then G_b has one fewer protected vertex than if Sludge pollutes a and then b and hence the same winning strategy suffices.

Now suppose that for every directed acyclic graph of size less than k in which Sludge can win, Sludge has an efficient strategy.

Let G be a graph in which the Sludge can win where $|V(G)| = k$. Let v be the source and $N(v) = \{v_1, v_2, \ldots, v_n\}$. Sludge pollutes the source v, Green protects some vertex x and now Sludge can win by polluting, say, v_1.

Let G' be the graph induced by $V(G') = V(G) - \{x, v\}$ together with the directed edges $\{(v_1, v_i) \ : \ i = 2, 3, \ldots, n\}$. By induction, Sludge can win on G' using an efficient strategy, call it S. If, regardless of which vertex Green next protects, S has Sludge winning by polluting some vertex not in $\{v_2, v_3, \ldots, v_n\}$ then Sludge has won in G by polluting only the edges of a path, i.e. efficiently. So suppose that Green can protect a vertex, y such that, under S, Sludge pollutes say v_2. Note that by Claim 1, Sludge will not pollute any other v_i. Therefore, Sludge can win in $G_{v_2} - \{y\}$ efficiently. Now by Claim 1, Sludge can win in G efficiently. ∎

Theorem 2.5 *Let G be a directed acyclic graph with a single source and let H be an induced subgraph of G that includes the source of G and every sink of H is a sink of G. If H is sludge-win then G is sludge-win.*

Proof: If Sludge wins on H then any move by Green on $V(G) - V(H)$ is regarded by Sludge as a pass in H. ∎

3 Trees

Let T denote a rooted tree; we assume the root to be the source and the leaves to be the sinks.

Lemma 3.1 *If T is green-win then Green can win by always protecting at a child of the previously polluted vertex.*

Proof: Suppose Sludge moves to v_i and Green can win by protecting x, where $dist(x, v_i) \geq 2$. As T is a rooted tree, Green can win by protecting v_i', where v_i' adjacent to v_i and v_i' is on the path from v_i to x. ∎

Figure 2 shows that Lemma 3.1 does not hold if the graph is not a tree. The only winning strategy for Green is to protect a first (the source is s and the sinks are indicated by white vertices).

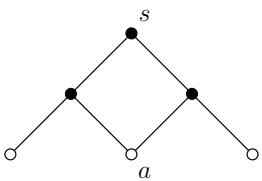

Figure 2: $P_3 \square P_3$ truncated at distance $d = 2$.

Lemma 3.2 *If T is a rooted tree with no vertex v where $d^+(v) = 1$, then T is sludge-win.*

Proof: If Sludge has polluted vertex v which is not a sink, then by Lemma 3.1, Green can subsequently only protect one of the children of v. Therefore, if Sludge has not polluted a sink, he always has a move. ∎

Give a rooted tree T, let T_x be the subtree rooted at x.

Theorem 3.3 *If $x \in V(T)$ and $d^+(x) = 1$, then $T - T_x$ is green-win if and only if T is green-win.*

Proof: Suppose there is vertex $x \in V(T)$ such that $d^+(x) = 1$.

Suppose $T - T_x$ is green-win. When Sludge moves on vertices of $T - T_x$, Green follows his winning strategy. If Sludge moves out of $T - T_x$, it must first move to x. Green then responds by protecting the child of x. Sludge is forced to continue its play on $T - T_x$ for the remainder of the game. The game continues on $T - T_x$ and Green wins.

Suppose $T - T_x$ is sludge-win. Then when playing on T, Sludge restricts its moves to vertices of $T - T_x$. If Green plays outside of $T - T_x$, Sludge treats that like a pass on $T - T_x$, continues its strategy on $T - T_x$, and wins. ∎

The previous two theorems give a characterization based on a decomposition procedure reminiscent of the characterization of cop-win graphs [12]. Let x be a vertex of a rooted tree T. T is said to be *green-reduced* to $T - T_x$ if x has out-degree 1 in T and no ancestor of x has out-degree 1. Then $T - T_x$ is a *green-reduction* of T.

Corollary 3.4 *A rooted tree T is green-win if and only if T can be reduced to the empty graph by a sequence of green-reductions.*

It is easy to see that any valid series of green-reductions will terminate in the same graph, so Sludge can apply this procedure to find a subtree that he wishes to play on.

We also give an algorithm that determines if the rooted tree is green-win based on labeling the vertices from the sinks back toward the source:

Tree Labeling Algorithm

1. Initially, all the vertices are unlabeled.
2. Label the leaves s.
3. All vertices that have two children or more labeled s are also labeled s. Any vertex that has all but one of its children labeled g is labeled g.
4. Repeat Step 3 until all vertices are labeled.

Theorem 3.5 *Let T be a rooted tree. The Tree Labeling Algorithm labels the root g if and only if T is green-win.*

Proof: The proof is a straightforward application of Lemma 3.2 ∎

From the Tree Labeling Algorithm, we can determine if a rooted tree T is green-win in polynomial time. More specifically, we visit each vertex v and look at the children of it to determine the label on v. Thus, the Tree Labeling Algorithm returns whether T is green-win in at most n^2 steps, where $n = |V(T)|$.

We now give analogous characterization of Corollary 3.4 for k-green-win trees. The proofs are similar to those for Theorem 3.3 and Corollary 3.4.

Lemma 3.6 *If T is k-green-win then there is a vertex v with $d^+(v) \leq k$.*

Let x be a vertex of a rooted tree T. T is said to be *k-green-reduced* to $T - T_x$ if x has out-degree at most k in T and every ancestor of x has out-degree greater than k.

Theorem 3.7 *A rooted tree T is k-green-win if and only if T can be reduced to the empty graph by a sequence of k-green-reductions.*

4 Cartesian Product of Paths, Truncated

Let P_k be the directed path with vertices $\{0, 1, \ldots, k\}$. Let $(P_{p_1} \Box P_{p_2} \Box \ldots \Box P_{p_n}, d)$ represent the game where the graph is the product $P_{p_1} \Box P_{p_2} \Box \ldots \Box P_{p_n}$ but restricted to the vertices that are within distance d of the source, which is located at coordinate $(0, 0, \ldots, 0)$. Every vertex at distance d from $(0, 0, \ldots, 0)$ is a sink.

The proof of the first result is found by observing the graph in Figure 2 is green-win.

Theorem 4.1 *For positive integers p_1, p_2, $(P_{p_1} \Box P_{p_2}, d)$ is green-win for $d \geq 2$.*

Theorem 4.2 *Let Q_n denote the n-dimensional hypercube. (Q_n, d) is sludge-win for $1 \leq d < n$ and is green-win for $d = n$.*

Proof: (Q_n, n) is clearly green-win since there is only one sink.

(Q_2, d) is sludge-win for $d = 0, 1$. In Q_n, wherever Green protects, Sludge can move to the 'source' in a copy of $(Q_{n-1}, d-1)$ that has no protected vertices. The result follows by induction. ∎

Theorem 4.3 *Two Greens suffice to protect $(P_n \Box P_n \Box P_n, d)$ for $d \geq 3$; that is,*

$$gr((P_n \Box P_n \Box P_n, d)) \leq 2 \text{ for } d \geq 3.$$

Proof: By construction: the Greens protect $(1, 1, 1)$ and $(0, 0, 1)$. If Sludge pollutes $(1, 0, 0)$, the Greens protect $(2, 0, 0)$ and $(1, 1, 0)$; from here Sludge can only pollute $(1, 0, 1)$, but the Greens protects $(2, 0, 1)$ and $(1, 0, 2)$. If Sludge at any time pollutes $(0, 1, 0)$, the Greens protects $(1, 1, 0)$ (which may already be protected) and $(0, 1, 1)$. This forms a cut set with polluted vertices on one side and the sinks on the other. ∎

The situation for the products of more than 3 paths is less clear. The following result gives an upper bound.

Theorem 4.4 *Let (P_n^k, d) be the game played on the Cartesian product of k paths of length n truncated at level d. Then*

$$gr((P_n^k, d)) \leq \min_{1 \leq j \leq k} \{\max\{j, gr((P_n^j, d - (k-j)n))\}\}.$$

Proof: Fix j. After each Sludge move, the Greens protect so as to prevent Sludge from changing the last j coordinates, i.e. if Sludge moves to $(a_1, a_2, \ldots, a_{k-j}, 0, 0, \ldots, 0)$, the Greens protect $(a_1, a_2, \ldots, a_{k-j}, 1, 0, \ldots, 0)$, $(a_1, a_2, \ldots, a_{k-j}, 0, 1, \ldots, 0)$, etc. If d is large enough, Sludge eventually pollutes $(n, n, \ldots, n, 0, 0, \ldots, 0)$. This leaves the seepage game to be played out on $(P_n^j, d - (k-j)n)$. Consequently, for this fixed j, the Greens need to protect $\max\{j, gr((P_n^j, d - (k-j)n))\}$ vertices on each move. Clearly, with this approach, the Greens could choose the j that gave the least number of vertices that need to be protected. ∎

From the proof of Theorem 4.4, if we fix j and have the Greens prevent Sludge from moving in the last j coordinates then the game is eventually played on $(P_n^j, d - (k-j)n)$. The source of this subgraph has exactly j children and so the Greens can win on the next turn. This proves the next corollary.

Corollary 4.5 *$gr((P_n^k, d)) \leq j$ if $d \geq (k-j)n+1$.*

By collecting together the results from this section (some of which follow), if we allow the Greens to protect j vertices at each step, Table 1 gives an upper bound on d, the truncation depth, needed for the Greens to win. See Problem 5.3.

$k=$	1	2	3	4	5
$j=1$	1	2	≤ 9	$\leq 3n+1$	$\leq 4n+1$
$j=2$	1	1	3	$\leq 2n+1$	$\leq 3n+1$
$j=3$	1	1	1	$\leq n+1$	$\leq 2n+1$

Table 1: Minimum truncation when the Greens protect j vertices and wins.

Although Theorem 4.3 showed that for $d \geq 3$, 2 Greens can win on any truncated graph (P_m^3, d), Theorem 4.6 now shows that if $d \geq 9$, then 1 Green will suffice.

Theorem 4.6 *If $n > 8$ and $d > 8$ then $(P_n \Box P_n \Box P_n, d)$ is green-win.*

Proof: It will be sufficient to show this in the case $d = 9$ and by Theorem 2.4 we may assume the Sludge will play a single directed path. For the proof we use a representation which is a view from the source vertically downwards. We orient the graph so that the source, $(0,0,0)$ (at level 0) is the highest point on the graph and $(1,1,1)$ (at level 3) lies directly below the source. Drawing the vertices at level 3, and joining them by an edge if they are adjacent to a common vertex at level 2, we get the (partial) hexagonal grid (also called the triangular grid) as seen in Figure 3.

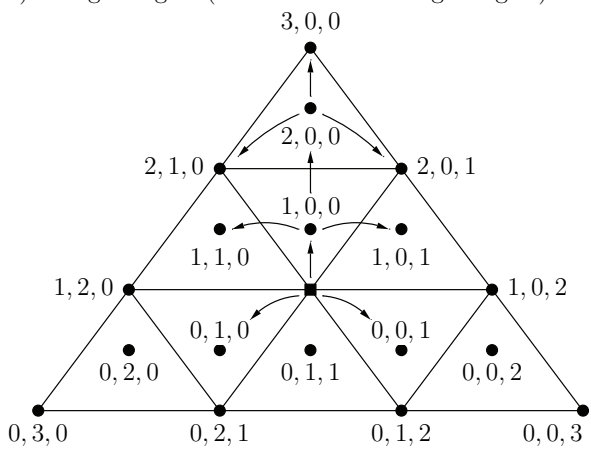

Figure 3: Part of the hexagonal grid: the square vertex represents $0,0,0$ (and also $1,1,1$).

We note that vertices at level 1 and level 2 can be identified with the triangles of the Figure. Each vertex at level 2 is adjacent to a set of vertices which form an upward pointing triangle. Henceforth, a level 2 vertex is identified with its corresponding upward pointing triangle. We call these *utriangles*. Vertices at level 1 are called *dtriangles* because they can be identified with the downward pointing triangles: these vertices are adjacent to the vertices at level 2 which are identified with the three surrounding triangles. This pattern 'repeats' with period three. That is, we will identify the vertices at levels 0, 3 and 6 with the vertex at level 9 directly below it. Similarly, we will identify dtriangles at levels 1 and 4 with the corresponding dtriangles at level 7 and we will identify dtriangles at levels 2 and 5 with the corresponding utriangles at level 8. In our proofs, we take the hexagonal grid as that given by the vertices on the bottom level. Green's moves will always be on that bottom level, but we will refer to Sludge as moving to a triangle or vertex regardless of the actual level.

We require three small results to break the main proof into manageable pieces. However, we need coordinates not tied to the original cube. Instead, we label the vertices at level 9 (somewhat imprecisely) with Cartesian coordinates. The source is labeled with $(0,0)$. Now label the remaining vertices with the following process. If a vertex is labeled (a,b), label all of its neighbours counterclockwise, starting from the vertex on its left are $(a-2,b)$, $(a-1,b-1)$, $(a+1,b-1)$, $(a+2,b)$, $(a+1,b+1)$ and $(a-1,b+1)$. See Figure 4. Because of the projective nature of the representation, a vertex can be simultaneously polluted and protected because of the difference in levels. We will refer to Sludge *moving to* rather than polluting, a vertex or triangle.

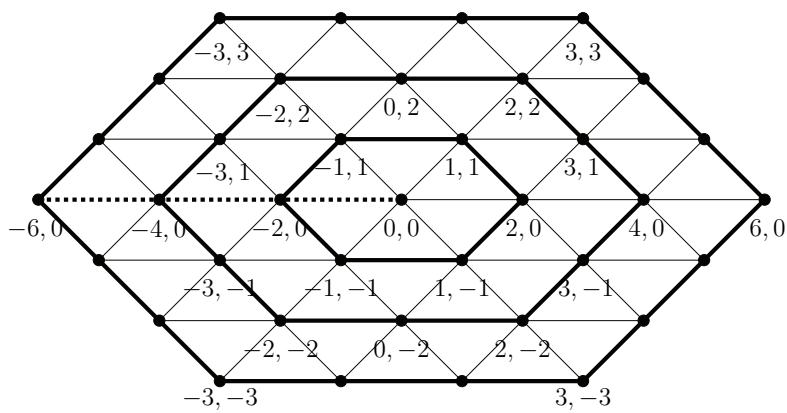

Figure 4: Part of the hexagonal grid with new coordinate representation.

A triangle is *safe* if the vertices which form the triangle (on the bottom level) are all protected. The next two Claims show that once Sludge is on a safe triangle, Green can ensure that any utriangle Sludge moves to will be safe by the end of that

time step. This also ensures that Green will win because when Sludge reaches the next-to-last level (Sludge will be on a utriangle), Green will be able to ensure that all of the adjacent vertices at the bottom level are protected.

Claim 1: If Sludge moves to a dtriangle at time t, which Green can make safe by the end of time step t, then at the end of $t+1$, Green can make safe the utriangle to which Sludge has just moved.

Proof: We may suppose that Sludge has just moved to the triangle $\langle(0,0),(2,0),(1,-1)\rangle$ and after Green's move all of $(0,0),(2,0),(1,-1)$ (at the bottom level) are protected. All three of the adjacent utriangles have two vertices in common with $\langle(0,0),(2,0),(1,-1)\rangle$ and so Green can protect the third vertex of whichever of these three utriangles Sludge moves to.

Claim 2: If Sludge moves to a utriangle which Green makes safe by the end of that time step, then by the end of three time steps later, Green can make safe the utriangle to which Sludge has just moved.

Proof: We may suppose that Sludge has moved to utriangle $\langle(0,0),(2,0),(1,1)\rangle$ and after Green's move, all of $(0,0),(2,0),(1,1)$ are protected. Without loss of generality, suppose Sludge now moves to $(2,0)$. Then Green protects $(4,0)$. Sludge now has three options: the dtriangles $\langle(2,0),(1,1),(3,1)\rangle$, $\langle(2,0),(4,0),(3,-1)\rangle$, $\langle(2,0),(0,0),(1,-1)\rangle$. But each has at least two protected vertices, so Green can protect the third vertex (if necessary) on his next move and the result follows from Claim 1.

The conclusion of the first two Claims is that *Sludge will never move to a triangle with two protected vertices* $(*)$. For brevity, in the following arguments, we will never consider such a move.

Claim 3: If Sludge is to move at time step t, where Sludge is on unprotected vertex v, but by the end of t, v is adjacent to three protected vertices, x, y, z where $\{x, y, z\}$ is an independent set, then after at most four time steps, Green can ensure that Sludge is on a protected triangle.

Proof: We may assume Sludge is on $(0,0)$ and that $(2,0),(-1,1),(-1,-1)$ are protected. By symmetry, we may assume Sludge moves to $\langle(0,0),(2,0),(1,-1)\rangle$. Then Green protects $(1,-1)$ which by $(*)$ forces Sludge to move to $\langle(0,0),(2,0),(1,1)\rangle$. Green responds with $(1,1)$.

If Sludge moves to $(0,0)$, then Green protects $(0,0)$. If Sludge moves to $(2,0)$, then Green protects $(4,0)$. If Sludge moves to $(1,1)$, then Green protects $(2,2)$. In each of the three possible Sludge moves, Green's indicated response forces Sludge's next move to be onto a dtriangle with two protected vertices and the result follows from Claim 1.

We may assume that Sludge starts at (0,0) and Green protects (0,0). Sludge then moves to $\langle(0,0),(2,0),(1,-1)\rangle$ and Green protects (2,0). By $(*)$ and symmetry, we may now assume that Sludge moves to $\langle(2,0),(1,-1),(3,-1)\rangle$ and Green protects $(3,-1)$. By Claim 3, Sludge moves to $(3,-1)$ and then by (*) Green can force the following sequence of moves.

$$\stackrel{G}{\to}(4,-2)\stackrel{S}{\to}\langle(1,-1),(0,-2),(2,-2)\rangle\stackrel{G}{\to}(2,-2)\stackrel{S}{\to}\langle(1,-1),(0,-2),(2,-2)\rangle\stackrel{G}{\to}(0,-2);$$

Sludge now has three choices after which Green, using the strategy listed below can force Sludge to only one move per turn by (*).

$$(a): \quad \xrightarrow{S} (1,-1) \xrightarrow{G} (1,-1);$$
$$(b): \quad \xrightarrow{S} (2,-2) \xrightarrow{G} (3,-3);$$
$$(c): \quad \xrightarrow{S} (0,-2) \xrightarrow{G} (-1,-3) \xrightarrow{S} \langle (0,-2),(1,-1),(-1,-1)\rangle \xrightarrow{G} (-1,-1);$$

Note that, in all cases, Sludge is forced onto a safe triangle by the time he had move to a vertex on the eighth level, hence Green will prevent Sludge from moving past the ninth level. ∎

If we truncate at $d < 9$ then more than one vertex may have to be protected on each turn. We leave this to the reader to evaluate exactly, but we do know from Theorem 4.3, that 2 vertices are enough.

5 Problems

If we consider Sludge playing on several different graphs at once, (i.e. he has to make a move on each of them) then as coordinates, he moves from $(a_1, a_2, \ldots, a_n)$ to $(b_1, b_2, \ldots, b_n)$ where a_i is adjacent to b_i for each i. But this is the definition of an edge in the categorical product. Specifically, let T and S be directed trees rooted at t and s respectively. We will consider the seepage game on only the connected component (which happens to be a tree) that contains the vertex ts of the categorical product $T \times S$. If Green can win on both T and S, it is not the case that Green can automatically win in $T \times S$. Indeed, $gr(T \times S)$ can be arbitrarily high even both T and S are green-win, see Figure 5.

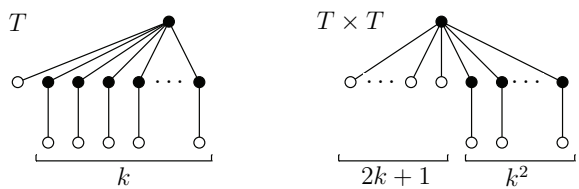

Figure 5: An example where $gr(T) = 1$, but $gr(T \times T) = 2k + 1$.

Problem 5.1 *Characterize those rooted trees T and S where $T \times S$ is green-win.*

We suspect that Theorem 4.4 is far from the best possible. For the Cartesian product of 4 paths, the Theorem gives $gr((P_n^4, d)) \leq 2$ for $d \geq 2n + 2$.

Problem 5.2 *Is it true that $gr((P_n^4, d)) = 2$ for d larger than some constant independent of n?*

Problem 5.3 *Find the exact value of* $gr((P_n^k, d))$.

The results and techniques used in Section 4 can be used to determine $gr((G, d))$ where G is the Cartesian product of three different length paths. In general, the Cartesian product of paths, oriented as we have in this paper, forms a distributive lattice. Let L be a finite lattice, and (L, d) the lattice truncated at level d.

Problem 5.4 *Let L be a finite distributive lattice. If the top element of L has k children, is $gr((L, d)) = gr((P_n^k, d))$?*

Modular lattices also have a nice structure.

Problem 5.5 *Let L be a finite modular lattice. Find $gr((L, d))$.*

References

[1] B. Alspach, Searching and Sweeping Graphs: A Brief Survey, *Le Matematiche* **59** (2004), 5–37.

[2] B. H. Bowditch, The Angel Game in the Plane, *Combinatorics Probability and Computing* **16** (2007), 345–362.

[3] J. H. Conway, The angel problem, *Games of No Chance*, Cambridge University Press, Cambridge, **29** (1996), 3–12.

[4] B. Bollobás and I. Leader, The devil and the angel in three dimensions, *J. Combin. Theory Ser. A* **113** (2006), 176–184.

[5] A. King and G. MacGillivray, The Firefighter Problem For Cubic Graphs, *Discrete Math.* (to appear).

[6] O. Kloster, A solution to the Angel Problem, *Theoretical Computer Science* **389** (2007), 152–161.

[7] M. Kutz, Conway's angel in three dimensions, *Theoretical Computer Science* **349** (2005), 443–451.

[8] A. Mathe, The Angel of Power 2 Wins, *Combinatorics Probability and Computing* **16** (2007), 363–374.

[9] M. E. Messinger, Firefighting on the Triangular Grid, *J. Combin. Math. Combin. Comput.* (to appear).

[10] M. E. Messinger, R. J. Nowakowski and P. Pralat, Cleaning a Network with Brushes, *Theoretical Computer Science* (to appear).

[11] J. McPhee, The Control of Nature: Cooling the Lava–II, *The New Yorker*, Feb 29, 1988, Vol. 64, Iss. 2, p. 64.

[12] R. Nowakowski and P. Winkler, Vertex-to-vertex pursuit in a graph, *Discrete Math.* **43** (1983), 235–239.

[13] `http://en.wikipedia.org/wiki/Eldfell`

(Received 30 Oct 2007; revised 1 Sep 2008)