Spanning trees and the complexity of flood-filling games

Kitty Meeks and Alexander Scott
Mathematical Institute, University of Oxford, 24-29 St Giles, Oxford OX1 3LB, UK
{meeks,scott}@maths.ox.ac.uk

April 7, 2012

Abstract

We consider problems related to the combinatorial game (Free-) Flood-It, in which players aim to make a coloured graph monochromatic with the minimum possible number of flooding operations. We show that the minimum number of moves required to flood any given graph G is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T. This result is then applied to give two polynomial-time algorithms for flood-filling problems. Firstly, we can compute in polynomial time the minimum number of moves required to flood a graph with only a polynomial number of connected subgraphs. Secondly, given any coloured connected graph and a subset of the vertices of bounded size, the number of moves required to connect this subset can be computed in polynomial time.

1 Introduction

In this paper we consider several problems related to the one-player combinatorial game (Free-)Flood-It, introduced as a topic for theoretical research by Arthur, Clifford, Jalsenius, Montanaro and Sach at FUN 2010 [5]. The game is played on a coloured graph, and the goal is to make the entire graph monochromatic with as few moves as possible. A move involves picking a vertex v and a colour d, and giving all vertices in the same monochromatic component as v colour d.

When the game is played on a planar graph, it can be regarded as modelling repeated use of the flood-fill tool in Microsoft Paint. Implementations of the game, played on a square grid, are widely available online, and include a flash game [1] as well as popular smartphone apps [2, 3]. Mad Virus [4]

is a version of the same game played on a hexagonal grid, while the Honey Bee Game [6] is a two player variant played on a triangular grid, analysed by Fleischer and Woeginger at FUN 2010 [9].

For any coloured graph, we define the following problems.

- Free-Flood-It is the problem of determining the minimum number of moves required to flood the coloured graph. The number of colours may be unbounded.
- c-Free-Flood-It is the variant of Free-Flood-It in which only colours from some fixed set of size c are used.

A related problem which naturally arises when considering algorithms for Flood-It is to consider the number of moves required to connect a given set of vertices.

- k-Linking Flood-It is the problem, given a subset U of at most k vertices, of determining the minimum number of moves required to create a monochromatic component containing U. The number of colours may be unbounded.

The implementations of the game mentioned above are in fact of a variant in which all moves must be played at some fixed root vertex; we refer to the problem of determining the minimum number of moves required to flood the board in this case as Fixed-Flood-It.[1]

In [5], Arthur, Clifford, Jalsenius, Montanaro and Sach proved that c-Free-Flood-It is NP-hard in the case of an $n \times n$ grid, for every $c \geq 3$, and that this result also holds for the fixed variant. Lagoutte, Noual and Thierry [11, 12] showed that the same result holds when the game is played instead on a hexagonal or triangluar grid, as in Mad Virus or a one-player version of the Honey Bee Game respectively. Lagoutte et. al. [11, 12] and Fukui, Nakanishi, Uehara, Uno and Uno [10] also proved that c-Free-Flood-It remains NP-hard when restricted to trees, for every $c \geq 3$.

A few positive results are known, however. 2-Free-Flood-It is solvable in polynomial time on arbitrary graphs, a result shown independently by Clifford et. al. [7], Lagoutte [11] and Meeks and Scott [13]. It is also known that Free-Flood-It is solvable in polynomial time on paths [7, 13, 10] and cycles [10]. Although c-Free-Flood-It is NP-hard on rectangular $3 \times n$ boards for any $c \geq 4$ [13], c-Free-Flood-It is fixed parameter tractable

[1] Fixed Flood It is often referred to as simply Flood-It, but we use the longer name to avoid confusion with the free version.

with parameter c when restricted to $2 \times n$ boards (Meeks and Scott [14]), and the fixed variant can be solved in linear time in this situation [7]. Meeks and Scott [13] also show that 2-Linking Flood-It can be solved in polynomial time for arbitrary graphs, even when the number of colours is unbounded.

In this paper we give some more general tractability results, which do not require the number of colours to be bounded. Our first such result is that Free-Flood-It can be solved in polynomial time on the class of graphs which have only a polynomial number of connected subgraphs. This class includes a number of interesting families of graphs, and the result implies a conjecture from [13] that the problem can be solved in polynomial time on subdivisions of any fixed graph. This substantially extends a result of Fukui, Nakanishi, Uehara, Uno and Uno [10], that the problem is polynomially solvable on cycles.

We then go on to consider k-Linking-Flood-It. We prove that, for any fixed k, it is possible to solve k-Linking-Flood-It in polynomial time, without imposing any restrictions on the underlying graph or initial colouring.

The key tool we use to prove these tractability results is a result which allows us to consider only spanning trees of the graph G in order to determine the minimum number of moves required to flood it. Clearly this does not immediately allow us to solve Free-Flood-It, as the problem remains hard even on trees, and a graph will in general have an exponential number of subgraphs. However, the result does provide a very useful method for reasoning about the behaviour of sequences of flooding operations on arbitrary graphs.

We begin in Sect. 2 with some notation and definitions, then in Sect. 3 we outline the proof of our result about spanning trees, and give a number of corollaries. Section 4 is concerned with the complexity of Free-Flood-It and the fixed variant on graphs containing only a polynomial number of connected subgraphs, and in Sect. 5 we consider the complexity of k-Linking-Flood-It.

2 Notation and Definitions

Suppose the game is played on a graph $G = (V, E)$, with an initial colouring ω (not necessarily proper) using colours from the *colour-set* C. Each move $m = (v, d)$ then involves choosing some vertex $v \in V$ and a colour $d \in C$, and assigning colour d to all vertices in the same monochromatic component as v. The goal is to make every vertex in G the same colour, using as few moves

as possible. We define $m_G(G, \omega, d)$ to be the minimum number of moves required in the free variant to give all its vertices colour d, and $m_G(G, \omega)$ to be $\min_{d \in C} m(G, \omega, d)$.

Let A be any subset of V. We set $m_G(A, \omega, d)$ to be the minimum number of moves we must play in G (with initial colouring ω) to give every vertex in A colour d, and $m_G(A, \omega) = \min_{d \in C} m_G(A, \omega, d)$. We write $\omega|_A$ for the colouring ω restricted to the subset A (and use the same notation $\omega|_H$ for the restriction of ω to the vertices of a subgraph H). We say a move $m = (v, d)$ is *played in* A if $v \in A$, and that A is *linked* if it is contained in a single monochromatic component. Subsets $A, B \subseteq V$ are *adjacent* if there exists $ab \in E$ with $a \in A$ and $b \in B$.

For any vertex $v \in V$, we write $\mathrm{comp}_G(v, \omega)$ to denote the monochromatic component of G, with respect to ω, that contains v. Given any sequence of moves S on a graph G with initial colouring ω, we denote by $S(\omega, G)$ (or simply $S(\omega)$ if G is clear from the context) the new colouring obtained by playing S in G.

3 Spanning Trees

In this section we investigate the relationship between the number of moves required to flood a connected graph G and the number of moves required to flood spanning trees of G. For any connected graph G, let $\mathcal{T}(G)$ denote the set of all spanning trees of G. We prove the following result.

Theorem 3.1. *Let G be a connected graph with colouring ω from colour-set C. Then, for any $d \in C$,*

$$m_G(G, \omega, d) = \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d).$$

Since it remains hard to solve 3-Free-Flood-It on trees, this result does not imply that the number of moves required to flood a graph with only a polynomial number of spanning trees can be computed in polynomial time. However, this equality gives rise to a number of corollaries, proved later in this section, which are then applied to give polynomial-time algorithms for various flood-filling problems in Sects 4 and 5.

We now outline the proof of Theorem 3.1. Our strategy is to prove both the inequalities $\min_{T \in \mathcal{T}(G)} m_T(T, \omega, d) \leq m_G(G, \omega, d)$ and $m_G(G, \omega, d) \leq \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d)$, giving the result. Due to space constraints, details

of the proofs are omitted in this section and are left for the journal version [15].

The key step in proving that $\min_{T \in \mathcal{T}(G)} m_T(T, \omega, d) \leq m_G(G, \omega, d)$ is the following lemma, which allows us to consider independently optimal sequences to flood disjoint subtrees of a tree.

Lemma 3.2. *Let T be a tree, with colouring ω from colour-set C, and let A and B be disjoint subsets of $V(T)$ such that $V(T) = A \cup B$ and $T[A], T[B]$ are connected. Then, for any $d \in C$,*

$$m_T(T, \omega, d) \leq m_{T[A]}(A, \omega|_A, d) + m_{T[B]}(B, \omega|_B, d).$$

We use this result to argue that, given an optimal sequence of moves S to flood a graph G, we can construct inductively a spanning tree T for G such that playing S in T will flood the tree.

We also make use of Lemma 3.2 in the proof of the reverse inequality. First, we introduce some further notation. We call a spanning tree T of G *d-minimal* (with respect to ω) if $m_T(T, \omega, d) = \min_{T' \in \mathcal{T}(G)} m_{T'}(T', \omega, d)$, and say that a spanning tree T *preserves monochromatic components of G* (with respect to ω) if T and G have the same monochromatic components, i.e. $\mathrm{comp}_G(v, \omega) = \mathrm{comp}_T(v, \omega)$ for all $v \in V(G)$.

Our first step is to show that, given any tree T and an edge $e \notin E(T)$, we can replace T by another tree T' that contains e, without increasing the number of moves we need to flood the tree. The proof of this lemma relies heavily on Lemma 3.2.

Lemma 3.3. *Let T be a tree with colouring ω from colour-set C, and suppose $e = uv \notin E(T)$. Then, for any $d \in C$, there exists a spanning tree T' of $T \cup \{e\}$, with $e \in E(T')$, such that $m_{T'}(T', \omega, d) \leq m_T(T, \omega, d)$.*

We then proceed to show that every coloured graph has a d-minimal spanning tree that preserves monochromatic components.

Lemma 3.4. *Let $G = (V, E)$ be a connected graph with colouring ω from colour-set C. Then, for any $d \in C$, there exists a d-minimal spanning tree for G that preserves monochromatic components of G with respect to ω.*

To complete the proof that $m_G(G, \omega, d) \leq \min_{T \in \mathcal{T}(G)} m_T(T, \omega, d)$, we argue that, if T is a d-minimal spanning tree for G (with colouring ω), then $m_G(G, \omega, d) \leq m_T(T, \omega, d)$. This completes our proof of Theorem 3.1.

By exploiting Theorem 3.1, it is possible to generalise Lemma 3.2 very substantially: note that this extends Lemma 3.2 from trees to arbitrary graphs, and that we do not require A and B to be disjoint.

Corollary 3.5. *Let G be a connected graph, with colouring ω from colour-set C, and let A and B be subsets of $V(G)$ such that $V(G) = A \cup B$ and $G[A], G[B]$ are connected. Then, for any $d \in C$,*

$$m_G(G, \omega, d) \leq m_{G[A]}(A, \omega|_A, d) + m_{G[B]}(B, \omega|_B, d).$$

Theorem 3.1 also allows us show that, if H is a subgraph of G, the number of moves we must play in G to link the vertices of H is at most the number required to flood the isolated subgraph H.

Corollary 3.6. *Let G be a connected graph with colouring ω from colour-set C, and H a connected subgraph of G. Then, for any $d \in C$,*

$$m_G(V(H), \omega, d) \leq m_H(H, \omega|_H, d).$$

Finally, we consider the number of moves required to connect a given subset of the vertices of G. For any $U \subseteq V(G)$, let $\mathcal{T}(U, G)$ be the set of all subtrees T of G such that $U \subseteq V(T)$. We then characterise the number of moves required to link U in terms of the number of moves required to flood elements of $\mathcal{T}(U, G)$. The following result then follows easily from Theorem 3.1 and Corollary 3.6.

Lemma 3.7. *Let G be a connected graph with colouring ω from colour-set C, and let $U \subseteq V(G)$. Then, for any $d \in C$,*

$$m_G(U, \omega, d) = \min_{T \in \mathcal{T}(U,G)} m_T(T, \omega|_T, d).$$

4 Graphs with Polynomial Bounds on the Numbers of Connected Subgraphs

Given a vertex v in an arbitrary graph G, the number of possible values of $\mathrm{comp}_G(v, \omega)$, as ω ranges over all possible colourings of G, will in general

be exponential. However, it is clear that $\text{comp}_G(v, \omega)$ must be a connected subgraph of G containing v, and in some interesting classes of graphs the number of connected subgraphs containing any given vertex is bounded by a polynomial function of $|G|$. In this section we apply corollaries of Theorem 3.1 to show that Free-Flood-It can be solved in polynomial time in this situation. Fixed-Flood-It is also polynomially solvable on this class of graphs, a result proved directly in the journal version [15].

It should be noted, however, that this condition is not *necessary* for a graph to admit a polynomial-time algorithm to solve Free-Flood-It. K_n has $\Theta(2^n)$ connected induced subgraphs, but the number of moves required to flood the graph in either version of the game is always one fewer than the number of colours used in the initial colouring. Graphs corresponding to rectangular $2 \times n$ boards give another such example for the fixed case, as there are $\Omega(2^n)$ connected subgraphs containing any given vertex but Fixed-Flood-It can be solved in linear time in this setting [7].

4.1 The FREE-FLOOD-IT Case

In this section we prove the following theorem.

Theorem 4.1. *Let p be a polynomial, and let $\mathcal{G}_p$ be the class of graphs such that, for any $G \in \mathcal{G}_p$, the number of connected subgraphs of G is at most $p(|G|)$. Suppose $G \in \mathcal{G}_p$ has colouring ω from colour-set C. Then, for any $d \in C$, we can compute $m(G, \omega, d)$ in polynomial time, and hence we can also compute $m(G, \omega)$ in polynomial time.*

It is easy to check that, if G is a subdivision of some fixed graph H, the number of connected subgraphs of G is bounded by a polynomial function of $|G|$, and so Theorem 4.1 implies a conjecture of Meeks and Scott [13].

Corollary 4.2. Free-Flood-It *is solvable in polynomial time on subdivisions of any fixed graph H.*

In the next theorem, we give an explicit bound on the time taken to solve Free-Flood-It in terms of the number of connected subgraphs in the graph we are considering. The proof relies on Corollary 3.5, which allows us to consider optimal sequences in distinct components of the graph independently. Theorem 4.1 follows immediately from this result.

Theorem 4.3. *Let G be a connected graph with colouring ω from colour-set C, and suppose G has at most N connected subgraphs. Then we can compute $m_G(G, \omega, d)$ for each $d \in C$, and hence $m_G(G, \omega)$, in time $O(|C|^3 \cdot N^3)$.*

Proof. Note that we may assume without loss of generality that ω is a proper colouring of G, otherwise we can contract monochromatic components to obtain an equivalent coloured graph. Let $\mathcal{H}$ be the set of connected subgraphs of G. We compute $m_H(H, \omega|_H, d_1)$ recursively, for each $H \in \mathcal{H}$ and $d_1 \in C$. For any $H \in \mathcal{H}$ we write $(A, B) \in \text{split}(H)$ if A and B are connected proper subgraphs of H such that $V(A) \cup V(B) = V(H)$, and $V(A) \cap V(B) = \emptyset$.

We define a function $m^*(H, \omega|_H, d_1)$, and claim that for any $H \in \mathcal{H}$ and $d_1 \in C$, we have $m_H(H, \omega|_H, d_1) = m^*(H, \omega|_H, d_1)$. We first define

$$m^*(\{v\}, \omega|_v, d_1) = \begin{cases} 0 & \text{if } \omega(v) = d_1 \\ 1 & \text{otherwise.} \end{cases}$$

and observe that this gives $m_H(H, \omega|_H, d_1) = m^*(H, \omega|_H, d_1)$ whenever $|H| = 1$. Further values of m^* are defined recursively using as follows:

$$\begin{aligned} m^*(H, \omega|_H, d_1) = \\ \min\{ & \min_{(A,B)\in\text{split}(H)} \{m_A(A, \omega|_A, d_1) + m_B(B, \omega|_B, d_1)\}, \\ & 1 + \min_{\substack{(A,B)\in\text{split}(H) \\ d_2 \in C}} \{m_A(A, \omega|_A, d_2) + m_B(B, \omega_B, d_2)\}\}, \quad (1) \end{aligned}$$

The fact that $m_H(H, \omega|_H, d_1) \leq m^*(H, \omega|_H, d_1)$ follows from Corollary 3.5. To see the reverse inequality in the case that $|H| > 1$ (and so by assumption H is not monochromatic under ω), we consider the final move α in an optimal sequence to flood H with colour d_1: either α changes the colour of some monochromatic area X, linking it to monochromatic areas $Y_1, \ldots, Y_r$ which already have colour d_1, or else H is already monochromatic in some colour d_2 before the final move, and α simply changes its colour to d_1. In the first case, we set $A = Y_1$ and $B = X \cup Y_2 \cup \ldots \cup Y_r$, and note that the disjoint subsequences of S consisting of moves played in A and B respectively flood the relevant subgraphs with colour d_1. Hence $|S| \geq m_A(A, \omega|_A, d_1) + m_B(B, \omega|_B, d_1)$. In the case that H is monochromatic before α, we observe that H cannot be monochromatic before the penultimate move of S (otherwise S would not be optimal) and apply the reasoning above to the initial segment S' of S in which the final move is omitted, a sequence which floods H with colour d_2: there exists $(A, B) \in \text{split}(H)$ such that $|S'| \geq m_A(A, \omega|_A, d_2) + m_B(B, \omega|_B, d_2)$, and hence $|S| \geq 1 + m_A(A, \omega|_A, d_2) + m_B(B, \omega|_B, d_2)$. Thus in either case we have $m^*(H, \omega|_H, d_1) \leq m_H(H, \omega|_H, d_1)$.

Observe that every subgraph on the right hand side of (1) is strictly smaller than H, and so a recursion based on this relationship will terminate.

Thus it remains to show that we can calculate $m^*(H, \omega|_H, d_1)$ for all $H \in \mathcal{H}$ and $d_1 \in C$ in time $O(|C|^3 \cdot N^3)$.

First we need to construct a list of all connected subgraphs of G. Clearly each vertex in the graph is a connected subgraph of order one, and given all connected subgraphs of order k we can construct all connected subgraphs of order $k+1$ by considering all possible ways of adding a vertex. Thus, if N_i denotes the number of connected subgraphs of order i in G, we can construct the list in time

$$n + \sum_{i=1}^{n-1} N_i(n-i) \leq n \cdot N = O(N^2).$$

To compute m^*, we begin by initialising the table in time $O(|G|)$, then all further values of m^* are then calculated as the minimum over combinations of two other entries. As our table has $N \cdot |C|$ entries, there are at most $N^2 \cdot |C|^2$ combinations we need to consider, and so we can compute all entries in time at most $O(N^3 \cdot |C|^3)$. This immediately gives $m_G(G, \omega, d_1)$ for each $d_1 \in C$, and to compute $m_G(G, \omega)$ we simply take the minimum over $|C|$ entries. Thus we can compute both $m_G(G, \omega, d)$ and $m_G(G, \omega)$ in time $O(N^3 \cdot |C|^3)$. □

5 The Complexity of k-LINKING FLOOD IT

In this section we use results from Sect. 3 to show that k-Linking-Flood-It, the problem of determining the minimum number of moves required to link some given set of k points (when moves can be played at any vertex), is solvable in polynomial time for any fixed k. Some details of the proof are omitted here due to space constraints; a full proof can be found in [15].

We begin with some additional notation. Let U be a subset of $V(G)$. We will say $(U_1, U_2) \in \text{part}(U)$ if U_1 and U_2 are disjoint nonempty subsets of U such that $U = U_1 \cup U_2$. Recall that $\mathcal{T}(U, G)$ is the set of all subtrees T of G such that $U \subseteq V(T)$. For $1 \leq i \leq |G|$, set $\mathcal{T}_i(U, G) = \{T \in \mathcal{T}(U, G) : |T| \leq i\}$.

Theorem 5.1. *Let $G = (V, E)$ be a connected graph of order n, with proper colouring ω from colour-set C, and let $U \subseteq V$ with $|U| = k$. Then, for any $d \in C$, we can compute $m_G(U, \omega, d)$ in time $O(n^{k+3} \cdot |E| \cdot |C|^2 \cdot 2^k)$.*

Proof. We demonstrate a dynamic programming algorithm to compute values of a function f, taking as arguments a nonempty subset $W \subset V$ of at

most k vertices, the initial colouring ω of the graph, a colour $d_1 \in C$, and an index $i \in \{1, \ldots, n\}$. We show that, for any values of these arguments, we have

$$f(W, \omega, d_1, i) = \begin{cases} \min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega|_T, d_1) & \text{if } \mathcal{T}_i(W, G) \neq \emptyset \\ \infty & \text{otherwise.} \end{cases}$$

Thus, as $\mathcal{T}_n(U, G) \neq \emptyset$, we see by Lemma 3.7 that

$$\begin{aligned} m_G(U, \omega, d) &= \min_{T \in \mathcal{T}(U,G)} m_T(T, \omega|_T, d) \\ &= \min_{T \in \mathcal{T}_n(W,G)} m_T(T, \omega|_T, d) \\ &= f(U, \omega, d, n). \end{aligned}$$

We initialise our table by setting

$$f(W, \omega, d_1, 1) = \begin{cases} \infty & \text{if } |W| \geq 2 \\ 1 & \text{if } W = \{w\} \text{ and } \omega(w) \neq d_1 \\ 0 & \text{if } W = \{w\} \text{ and } \omega(w) = d_1, \end{cases}$$

and observe that this gives the desired value of $f(W, \omega, d_1, 1)$ for all choices of W and d_1.

We define further values of f inductively. First, for any W, ω, d_1 and i, we set

$$\begin{aligned} \text{poss}(W, \omega, d_1, i) = \{&((W_1 \cup \{x_1\}, \omega, d_1, j_1), (W_2 \cup \{x_2\}, \omega, d_1, j_2)) : \\ &(W_1, W_2) \in \text{part}(W), x_1x_2 \in E, x_1 \notin W_2, x_2 \notin W_1, \\ &j_1 + j_2 = i, j_1, j_2 > 0\}, \end{aligned}$$

so there is an element of $\text{poss}(W, \omega, d_1, i)$ corresponding to each way of partitioning W into two non-empty subsets, and each pair of positive integers summing to i. We then define

$$f_1(W,\omega, d_1, i) = \begin{cases} \min_{(\mathbf{z_1}, \mathbf{z_2}) \in \text{poss}(W,\omega,d_1,i)} \{f(\mathbf{z_1}) + f(\mathbf{z_2})\} & \text{if } \text{poss}(W, \omega, d_1, i) \neq \emptyset \\ \infty & \text{otherwise,} \end{cases}$$

and

$$f_2(W, \omega, d_1, i) = 1 + \min_{d_2 \in C} \{f_1(W, \omega, d_2, i)\}.$$

Finally we set

$$f(W, \omega, d_1, i) = \min\{f_1(W, \omega, d_1, i), f_2(W, \omega, d_1, i), f(W, \omega, d_1, i-1)\}. \quad (2)$$

To show that f has the required properties, we first prove by induction on i that we have $f(W, \omega, d_1, i) \leq \min_{T \in \mathcal{T}_i(W,G)} m_T(T, \omega|_T, d_1)$ for each choice of W and d_1, if $\mathcal{T}_i(W, G) \neq \emptyset$. The base case follows from the definitions above, and the inductive step can be proved by fixing $T \in \mathcal{T}_i(W, G)$ such that $m_T(T, \omega|_T, d_1) = \min_{T' \in \mathcal{T}_i(W,G)} m_{T'}(T', \omega|_{T'}, d_1)$ and $|T|$ is minimal, and applying case analysis to the final move of an optimal sequence to flood T with colour d_1.

The reverse inequality is also proved by induction on i, and again the base case follows from the definition of f. The proof of the inductive step in this case relies on Theorem 3.1 and Corollary 3.5.

It remains only to bound the time taken to compute $f(U, \omega, d, n)$. Note that each value $f(W, \omega, d_1, 1)$ (for any $W \subset V$ of size at most k and $d_1 \in C$) can be computed in constant time.

Suppose we have computed the value of $f(W, \omega, d_1, i)$ for each $W \subset V$ of size at most k and $d_1 \in C$. To compute $f_1(W', \omega, d_2, i+1)$ for any W' and d_2, we take the minimum over at most 2^k ways to partition a set of up to k points, the $|E|$ edges in the graph, the $|C|$ colours in the initial colouring, and the $2(i-1)$ ordered pairs of positive integers that sum to i. Thus we take the minimum over a set of $O(2^k \cdot |E| \cdot |C| \cdot n)$ values, each of which can be computed in time $O(n)$ by adding a pair of existing values in the table, and so compute $f_1(W', \omega, d_2, i+1)$ in time $O(2^k \cdot |E| \cdot |C| \cdot i \cdot n) = O(2^k \cdot |E| \cdot |C| \cdot n^2)$.

Once we have computed the value of f_1 for all entries with index $i+1$, we can compute f_2 for each such entry in time $O(|C|)$. Given the values of f_1 and f_2 for each entry with index $i+1$, and the values of f for entries with index i, we can compute f for any entry with index $i+1$ in constant time.

Thus in total we require time at most $O(2^k \cdot |E| \cdot |C| \cdot n^2)$ to compute the value of f for each entry in the table. In total, the table contains $O(n^{k+1} \cdot |C|)$ entries (as there are $O(n^k)$ subsets of size at most k, a choice of $|C|$ colours, and i takes integer values in the range $[1, n]$), so we can compute all entries, and hence determine $f(U, \omega, d, n)$, in time $O(n^{k+3} \cdot |E| \cdot |C|^2 \cdot 2^k)$. □

6 Conclusions and Open Problems

We have shown that, for any connected graph G, the minimum number of moves required in the free variant of Flood-It to make G monochromatic in

colour d is equal to the minimum, taken over all spanning trees T of G, of the number of moves required to flood T with colour d.

Using this result, we saw that Free-Flood-It, and the fixed variant, are solvable in polynomial time on graphs with only a polynomial number of connected subgraphs. This proves a conjecture of Meeks and Scott [13]: Free-Flood-It is solvable in polynomial time on subdivisions of any fixed graph. This in turn implies that both Free-Flood-It is polynomially solvable on trees with bounded degree and a bounded number of vertices of degree at least three, although the problem is known to be NP-hard on arbitrary trees. It would be interesting to investigate other minor-closed classes of trees on which the problem can be solved in polynomial time.

Finally, we applied the result on spanning trees to the problem of k-Linking-Flood-It, demonstrating an algorithm to solve the problem in time $n^{O(k)}$. There is potential for further investigation of the parameterised complexity of this problem, with parameter k: can k-Linking-Flood-It be shown to be W[1]-hard, or is there another approach to the problem which might yield a fixed-parameter algorithm? Such an investigation could also consider a "fixed" variant of k-Linking-Flood-It, in which all moves must be played at some fixed vertex.

References

[1] *Flood It Game*, http://floodit.appspot.com.

[2] *Flood It! 2*, available at http://itunes.apple.com.

[3] *Flood It!*, available at https://market.android.com.

[4] *Mad Virus*, http://www.bubblebox.com/play/puzzle/539.htm.

[5] David Arthur, Raphaël Clifford, Markus Jalsenius, Ashley Montanaro, and Benjamin Sach, The Complexity of Flood Filling Games, in Paolo Boldi and Luisa Gargano, editors, *FUN*, volume 6099 of *Lecture Notes in Computer Science*, Springer, ISBN 978-3-642-13121-9, 2010, pages 307-318.

[6] A. Born, Flash application for the computer game *Biene (Honey-Bee)*, 2009. http://www.ursulinen.asn-graz.ac.at/Bugs/htm/games/biene.htm.

[7] Raphaël Clifford, Markus Jalsenius, Ashley Montanaro, and Benjamin Sach, The Complexity of Flood Filling Games, arXiv.1001.4420v2 [cs.DS], August 2010.

[8] Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1990.

[9] Rudolf Fleischer and Gerard J. Woeginger, An Algorithmic Analysis of the Honey-Bee Game, in Paolo Boldi and Luisa Gargano, editors, *FUN*, volume 6099 of *Lecture Notes in Computer Science*, Springer, ISBN 978-3-642-13121-9, 2010, pages 178-189.

[10] H. Fukui, A. Nakanishi, R. Uehara, T. Uno, Y. Uno, The complexity of free flooding games, Information Processing Society of Jamap (IPSG) SIG Notes 2011 (August 2011), 1-5.

[11] Aurélie Lagoutte, Jeux d'inondation dans les graphes, Technical report, ENS Lyon, HAL: hal-00509488, August 2010.

[12] A. Lagoutte, M. Noual, E. Thierry, Flooding games on graphs, HAL: hal-00653714, December 2011.

[13] Kitty Meeks and Alexander Scott, The complexity of flood-filling games on graphs, *Discrete Applied Mathematics* (2011), doi:10.1016/j.dam.2011.09.001.

[14] Kitty Meeks and Alexander Scott, The complexity of Free-Flood-It on $2 \times n$ boards, arxiv.1101.5518v1 [cs.DS], January 2011.

[15] Kitty Meeks and Alexander Scott, Spanning trees and the complexity of flood-filling games, arXiv:1203.2538v1 [cs.DS], March 2012.