An algorithmic analysis of Flood-It and Free-Flood-It on graph powers

Uéverton dos Santos Souza, Fábio Protti, Maise Silva

Discrete Mathematics and Theoretical Computer Science DMTCS vol. **16:3**, 2014, 279–290

An algorithmic analysis of Flood-it and Free-Flood-it on graph powers

Uéverton dos Santos Souza[1 2*] Fábio Protti[2†] Maise Dantas da Silva[3‡]

[1] *Centro Federal de Educação Tecnológica Celso Suckow da Fonseca - Rio de Janeiro, Brasil*
[2] *Instituto de Computação - Universidade Federal Fluminense - Niterói, Brasil*
[3] *Instituto de Ciência e Tecnologia - Universidade Federal Fluminense - Rio das Ostras, Brazil*

received 27th June 2014, revised 15th Oct. 2014, 13th Nov. 2014, accepted 19th Nov. 2014.

Flood-it is a combinatorial game played on a colored graph G whose aim is to make the graph monochromatic using the minimum number of flooding moves, relatively to a fixed pivot. Free-Flood-it is a variant where the pivot can be freely chosen for each move of the game. The standard versions of Flood-it and Free-Flood-it are played on $m \times n$ grids. In this paper we analyze the behavior of these games when played on other classes of graphs, such as d-boards, powers of cycles and circular grids. We describe polynomial time algorithms to play Flood-it on C_n^2 (the second power of a cycle on n vertices), $2 \times n$ circular grids, and some types of d-boards (grids with a monochromatic column). We also show that Free-Flood-it is NP-hard on C_n^2 and $2 \times n$ circular grids.

Keywords: Combinatorial Games, Graph algorithms, Computational Complexity, Flood-it, Free-Flood-it

1 Introduction

Let G be a vertex-colored graph. A flooding move $m = (p, c)$ in G consists of changing to c the color of a pivot vertex $p \in V(G)$, and of all vertices in the monochromatic component containing p in G. The problem of determining the minimum number of flooding moves to make the graph monochromatic is called *Free-Flood-it*. On the other hand, the problem of determining the minimum number of flooding moves to make the graph monochromatic *using a fixed vertex as the pivot for all moves* is called *Fixed-Flood-it*, or just *Flood-it*.

The computational problems Flood-it and Free-Flood-it are generalizations of two combinatorial games named alike, which are originally played on a colored board consisting of an $m \times n$ grid, where each tile of the board has an initial color from a fixed color set. In the classic game, two tiles are *neighboring* if they lie in the same row (resp. column) and in consecutive columns (resp. rows). A sequence C of tiles is a *path* when every pair of consecutive tiles in C is formed by neighboring tiles. A *monochromatic path*

*Email: usouza@ic.uff.br.
†Email: fabio@ic.uff.br.
‡Email: maisedantas@id.uff.br.

is a path in which all the tiles have the same color. Two tiles a and b are *m-connected* when there is a monochromatic path between them. In Flood-It, a move consists of assigning a new color c to the top left tile p (the *pivot*) and also to all the tiles m-connected to p immediately before the move. Figure 1 shows a sequence of moves to flood a 3×3 grid.

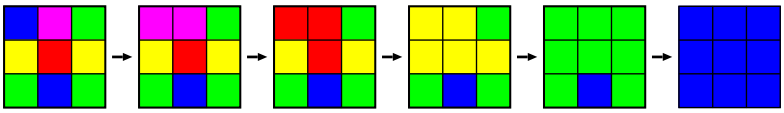

Fig. 1: An optimal sequence of moves to flood a 3×3 grid.

Many complexity issues on Flood-It and Free-Flood-It have recently been investigated. In [4], Clifford, Jalsenius, Montanaro, and Sach show that Flood-It and Free-Flood-It are NP-hard on $n \times n$ grids colored with at least three colors. Meeks and Scott [10] prove that Free-Flood-It is solvable in polynomial time on $1 \times n$ grids, and also that Flood-It and Free-Flood-It remain NP-hard on $3 \times n$ grids colored with at least four colors. Up to the authors' knowledge, the complexity of (Free-)Flood-It on $3 \times n$ grids colored with three colors remains as an open question. Clifford, Jalsenius, Montanaro, and Sach present in [4] a polynomial-time algorithm for Flood-It on $2 \times n$ grids. In [11], Meeks and Scott show that Free-Flood-It remains NP-hard on $2 \times n$ grids.

1.1 Flood-filling games on graphs

Flood-filling games played on graphs are a powerful model for several real world applications, mainly in Bioinformatics. In [15], for instance, Souza, Protti and Dantas da Silva show that Flood-it played on trees is analogous to an important subcase of the *Shortest Common Supersequence* problem, which is a classical problem from the realm of string analysis. Some problems and properties on strings are more easily observed when translated to a flood-filling dynamics, as done in [5], where a variant of the SCS problem is proved to be fixed-parameter tractable via an argument based on a translation of the problem in terms of the Flood-It game. Consequently, these games inherit many implications in bioinformatics, such as: microarray production [13], DNA sequence assembly [2], and a close relationship to multiple sequence alignment [14]. In addition, some disease spreading models described in [1] work in a similar way to flood-filling games on general graphs.

Regarding the complexity of Flood-it on non-grid graphs, Fleischer and Woeginger [6] proved that Flood-it (denoted by Honey-Bee-Solitaire) remains NP-hard even restricted to trees or split graphs, but it is polynomial-time solvable on co-comparability graphs. When Flood-it is played on paths ($1 \times n$ grids) the problem is trivially solvable if the pivot has degree one; however, when allowing the pivot to be any vertex of the path, the problem is analogous to the Shortest Common Supersequence problem (SCS) for two sequences [15], which is a very well-studied problem for which no linear-time algorithm is known. Consequently, it is easy to see that Flood-it on cycles can be solved in polynomial time. Free-Flood-It is solvable in polynomial time on 2-colored graphs [4, 9, 10]. In [10], Meeks and Scott show that Free-Flood-it on paths can be solved in $O(n^6)$ time. Free-Flood-it on cycles can also be solved in polynomial time [7, 12].

The main goal of this paper is to analyze the complexity of flood-filling games played on simple structures. We study the complexity of Flood-it and Free-Flood-it on others classes of boards, such as power of paths, powers of cycles, and circular grids. We describe polynomial-time algorithms to play Flood-it

on C_n^2 (the second power of a cycle on n vertices), $2 \times n$ circular grids ($2 \times n$ grids where the first and last tiles in a same row are neighboring tiles), $2 \times n$ d-boards ($2 \times n$ grids where the dth column is monochromatic), and $2 \times n$ circular d-boards. We also show that Free-Flood-it is NP-hard on C_n^2 and $2 \times n$ circular grids.

1.2 Additional definitions and notation

- Neighboring tiles naturally correspond to neighboring vertices of a graph G representing the board; therefore, from now on, we use the term *vertex* instead of *tile*. A subgraph H of G is *adjacent* to a vertex $v \in V(G)$ if v has a neighbor in $V(H)$.
- A *flood move*, or just *move*, is a pair $m = (p, c)$ where p is the *pivot* of m (the vertex chosen to have its color changed by m), and c is the new color assigned to p; in this case, we also say that color c is *played in move* m. In Flood-It all moves have the same pivot.
- A subgraph H is said to be *flooded* when H becomes monochromatic. A vertex v is *flooded by a move* m if the color of v is played in m and v becomes m-connected to new vertices after playing m. We say that a move m *floods a vertex* v *by a vertex* w if v and w are neighbors and move m changes the color of w to flood v.
- A (free-)flooding is a sequence of moves in (Free-)Flood-It which floods G (the entire board). An optimal (free-)flooding is a flooding with minimum number of moves.
- A move $m = (p, c)$ *is played on subgraph* H if $p \in V(H)$.
- A monochromatic subgraph H' of a subgraph H is abbreviated an *mcs of* H.
- An *island* is a vertex v colored with a color c such that no neighbor of v is colored with c.
- Let G_n be a graph with n vertices, the k-th power of G_n, denoted by G_n^k, is the graph formed by G_n plus edges between vertices at a distance at most k. Thus, P_n^k and C_n^k is the k-th power of a path P_n and a cycle C_n, respectively.
- A *circular grid* is an $m \times n$ grid with the additional property that the first and the last tiles in a same row are neighboring tiles.

Following, we present the formal definitions of the flood-filling games.

Flood-it (decision version)
Instance: A colored graph G with a pivot vertex p, an integer λ.
Question: Does there exist a sequence of λ flood moves which makes the graph monochromatic, and uses p as pivot in all moves?

Free-Flood-it (decision version)
Instance: A colored graph G, an integer λ.
Question: Does there exist a sequence of λ flood moves which makes the graph monochromatic?

---

2 Flood-it on Circular Boards

A d-board is an $m \times n$ grid where the dth column is monochromatic. Flood-it on a d-board consists of playing the game using column d as the pivot. Observe that Flood-it on paths (where any vertex can be the fixed pivot) is equivalent to Flood-it on $1 \times n$ d-boards.

In the literature it has been standard that Flood-It played on $m \times n$ grids uses the top left tile as the pivot. However, the situation for $m \times n$ grids being flooded from a fixed arbitrary pivot that is not necessarily the top left tile, as far as the authors know, has not previously been studied, and will follow as an easy corollary from the reduction shown below.

Let B be an $m \times n$ grid with an arbitrary pivot vertex p. Let (i, j) be the coordinate of p in B. We can construct an equivalent d-board B' by replacing column j by $2\max\{i-1, m-i\}+1$ columns. Figure 2 illustrates such a reduction for an 8×7 grid.

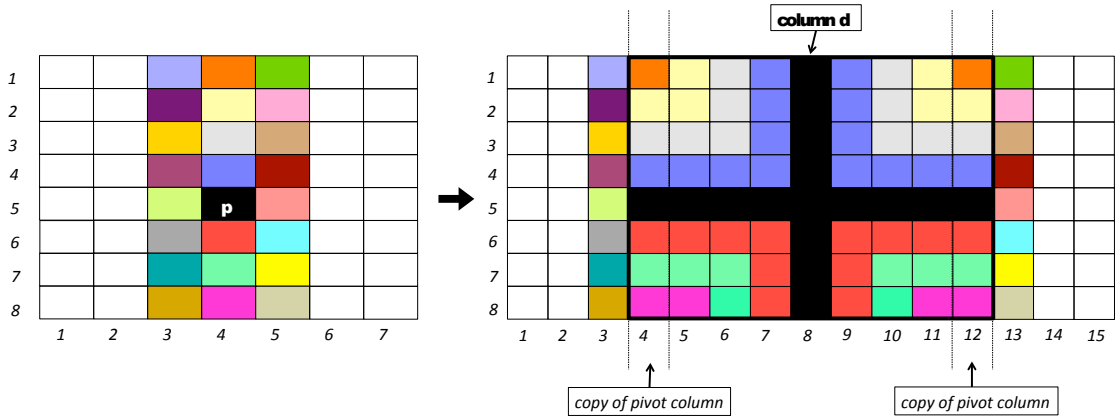

Fig. 2: Transformation of an 8×7 grid into an equivalent d-board.

Now we show a framework based on d-boards to solve Flood-it on (circular) $2 \times n$ grids with a fixed arbitrary pivot. The following lemma is a special case of Theorem 4.8 in [10], which establishes that the minimum number of moves to connect two tiles in an arbitrary colored graph G is $O(|V(G)|^3|E(G)|k^2)$, where k is the number of colors used in G. Our lemma says that this complexity can be expressively decreased to $O(n^2)$ when G is a $2 \times n$ d-board. We denote by B_l (resp. B_r) of a d-board B the board composed by d and all tiles to the left (resp. right) of d.

Lemma 1 *Given a $2 \times n$ d-board B, and vertices $v_l \in B_l$ and $v_r \in B_r$, the minimum number of moves to connect v_l and v_r can be found in $O(n^2)$ time.*

Proof: Let L, R be maximum mcs's of B_l and B_r containing d, respectively. We can think of L and R as "dynamic subgraphs" in the sense that they modify after each move. Observe that v_l will be flooded by a vertex which is either in the same column as v_l or in a column to the right of v_l. Therefore, columns to the left of v_l do not need to be analyzed. An analogous reasoning applies to v_r. Thus in order to choose which color must be played in each move, we only need to know the leftmost (resp. rightmost) vertex (or pair of vertices) of L (resp. R). This set of one or two vertices defines a *boundary* of L (or R).

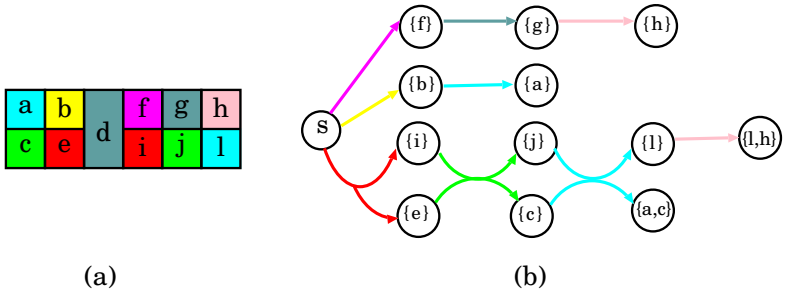

Fig. 3: (a) A $2 \times n$ d-board B. (b) A subgraph of the hypergraph H obtained from B.

At this point, we construct a directed acyclic hypergraph H as follows:

- create a vertex for each possible boundary of L or R;
- given boundaries S_1, T_1 and S_2, T_2 of mcs's L and R, respectively, add a directed hyperedge of the form $(\{S_1, S_2\}, \{T_1, T_2\})$ labeled with color c if by playing color c it is possible to simultaneously change the boundary of L from S_1 to T_1 and the boundary of R from S_2 to T_2;
- given boundaries S_1, T_1 of L (or R), add a directed hyperedge $(\{S_1\}, \{T_1\})$ labeled with color c if by playing color c it is possible to change the boundary of L (or R) from S_1 to T_1;
- collapse the vertices representing the initial boundaries of L and R into a single vertex s.

Each hyperedge of H represents a possible move of the game. A hyperedge $(\{S_1, S_2\}, \{T_1, T_2\})$ represents a move connecting vertices of L and R. For example, the hyperedge $(\{\{e\}, \{i\}\}, \{\{c\}, \{j\}\})$ in Figure 3 shows that by playing color green we connect tiles c and j via e and i. Finding the minimum number of moves to connect v_l and v_r amounts to finding the minimum number of hyperedges needed to construct paths between s and vertices representing boundaries containing v_l and v_r. The number of possible boundaries of L (or R) is clearly $O(n)$, and this implies that the number of hyperedges of H is $O(n^2)$. Finding minimum paths connecting s to all other vertices can be done in time linear in the size of H via a breadth-first search rooted at s. Hence, we can find the minimum number of moves to connect v_l and v_r in $O(n^2)$ time. □

Figure 3 shows a $2 \times n$ d-board B and a subhypergraph of H obtained from B which contains the optimal sequence of moves to connect a and h.

Theorem 2 *Flood-it can be solved in polynomial time on $2 \times n$ d-boards. More precisely, in $O(kn^2)$ time, where k is the number of colors.*

Proof: Suppose that B is a $2 \times n$ d-board and k is the number of colors. We say that a tile t on L (resp. R) is *marked* if it has color c_t and no other tile in the columns strictly to the left (resp. right) of t has the color c_t . A column is marked if it contains a marked tile. As in [4], the key property which holds on $2 \times n$ d-boards is that if the marked tiles are flooded then so is the whole board B. To see this, note that when a marked tile t of color c_t in L (resp. R) is flooded, all other tiles of color c_t in L (resp. R) that have not yet

been flooded are to the right (resp. left) of t and therefore adjacent to the flooded region. Hence they will be flooded when t is flooded. Thus, we iteratively ask for the shortest sequence of moves to connect the rightmost non-flooded marked tile of L with the leftmost non-flooded marked tile of R until all marked tiles of L and R are connected. Hence, the minimum number of moves needed to flood B can be obtained in time $O(kn^2)$. □

Theorem 3 *Flood-it can be solved in $O(kn^4)$ time on $2 \times n$ circular d-boards, where k is the number of colors.*

Proof: Let $d, c_1, c_2, \ldots, c_{n-1}$ be the columns of a $2 \times n$ circular d-board B. Let S be an optimal sequence of moves to flood B. Let us say that $v \in c_i$ is *right-flooded by* S if either $i = 1$ or $i \leq n-2$, and before playing the move $m \in S$ which floods v there is a path from d to v such that every internal vertex of such a path belongs to the maximum mcs containing the pivot and is right-flooded. On the other hand, a vertex v is *left-flooded by* S if it is not right-flooded. A column c_i is said to be right-flooded (left-flooded) by S if its two vertices are right-flooded (left-flooded) by S. Note that even by removing all edges connecting right-flooded vertices to left-flooded vertices, the sequence S still floods B.

Let c_i be the right-flooded column with maximum index i, and v_j be a right-flooded vertex lying in a column j such that j is maximum ($j \geq i$). Let H_r (resp., H_ℓ) be the subgraph induced by the vertices in d plus the right-flooded (resp. left-flooded) vertices. By removing all edges between H_r and H_ℓ, we can naturally define a $2 \times n$ d-board B' (if after removing the edges some column d' is left with only one vertex w, replace d' by a monochromatic column with the same color as w). Note that an optimal flooding of B' corresponds to an optimal flooding of B. Since c_i and v_j are not previously known, we must execute this process for each possibility (there are $O(n^2)$ many of them). Hence, we can solve Flood-it on $2 \times n$ circular d-boards in $O(kn^4)$ time.

We remark that the technique used in this proof (deleting certain edges without changing the effect of the flooding operations) is studied in [12], where the authors analyze the relationships between floodings of graphs and their spanning trees. □

Corollary 4 *Flood-it is solvable in polynomial time on $2 \times n$ circular grids.*

Proof: Follows from Theorem 3, by replacing the column c containing the pivot by three consecutive columns c_1, d, c_2, where c_1 and c_2 are copies of c, and d is a monochromatic column with the same color of the pivot of c. □

Lemma 5 *Flood-it on C_n^2 is a particular case of Flood-it on circular grids.*

Proof: Let $v_1, v_2, \ldots, v_n$ be the vertices of a graph C_n^2. Then, by taking circular indices, the neighborhood of v_i is $\{v_{i-2}, v_{i-1}, v_{i+1}, v_{i+2}\}$. We can create a $2 \times n$ circular grid T equivalent to C_n^2 as follows:

- For n even:
  (i) define $q_{a_1}, q_{b_1}, q_{a_3}, q_{b_3}, \ldots, q_{a_{n-1}}, q_{b_{n-1}}$ as the first row;
  (ii) define $q_{b_n}, q_{a_2}, q_{b_2}, q_{a_4}, q_{b_4}, \ldots, q_{a_{n-2}}, q_{b_{n-2}}, q_{a_n}$ as the second row.
- For n odd:
  (i) define $q_{a_1}, q_{a_3}, q_{b_3}, q_{a_5}, q_{b_5}, \ldots, q_{a_{n-2}}, q_{b_{n-2}}, q_{a_n}$ as the first row;
  (ii) define $q_{b_1}, q_{a_2}, q_{a_4}, q_{b_4}, \ldots, q_{a_{n-1}}, q_{b_{n-1}}$ as the second row.

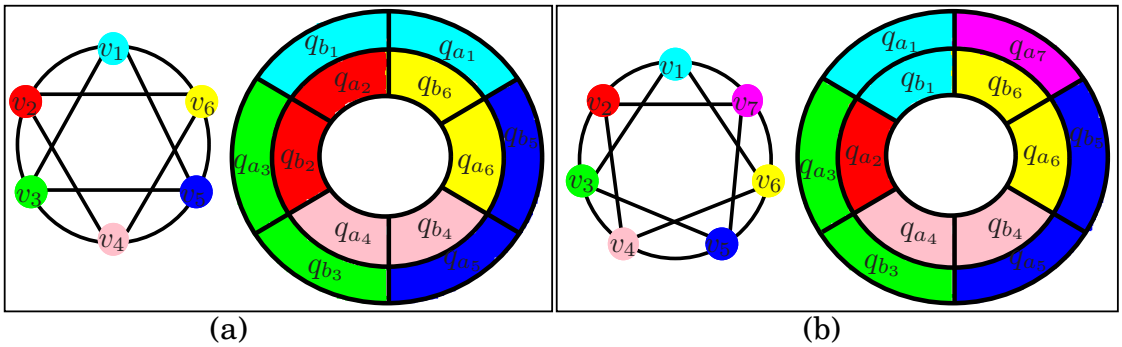

Fig. 4: (a) $2 \times n$ circular grid for even n; (b) $2 \times n$ circular grid for odd n.

In both constructions tiles q_{a_i} and q_{b_i} (if they exist) receive the same color as v_i. See Figure 4.

By assuming that the "component" $\{q_{a_i}, q_{b_i}\}$ (or $\{q_{a_i}\}$, for n odd and $i = 2$ or $i = n$) represent vertex v_i, we observe that: (i) for n even, a tile of color c is adjacent to $\{q_{a_i}, q_{b_i}\}$ if and only if v_i has a neighbor of color c; (ii) for n odd, the same property above is valid, except for $\{q_{a_2}\}$ and $\{q_{a_n}\}$; however, both are adjacent to component $\{q_{a_1}, q_{b_1}\}$ that represents the pivot vertex v_1. Thus, in Flood-it, C_2^n can be represented as a $2 \times n$ circular grid. □

Corollary 6 *Flood-it can be solved in polynomial time on C_n^2.*

Proof: Follows from Lemma 5 and Corollary 4. □

Corollary 7 *Flood-it can be solved in polynomial time on P_n^2.*

Proof: Follows using a similar construction as in Lemma 5 and Corollary 3. □

3 Free-Flood-it on Powers of Cycles

Flood-it on paths can be easily solved in $O(n^2)$ time by a dynamic programming [15], and as show in this paper, the problem remains polynomially solvable when played on circular grids, C_n^2 and P_n^2. Although Free-Flood-it can be solved in polynomial time when played on paths and cycles, in this section, we show that Free-Flood-it is NP-hard when played on C_n^2, P_n^2 or circular grids.

Theorem 8 *Free-Flood-it remains NP-hard on C_n^2.*

Proof: Our proof uses similar ideas as in Theorem 4.1 and Lemma 4.3 in [11], although they do not immediately apply to our case.

We use a reduction from the Vertex Cover Problem. Let Q be a graph formed by vertices x_1, x_2, x_3, x_4 and edges $e_a = (x_1, x_2), e_b = (x_1, x_3), e_c = (x_2, x_4)$. Note that Q contains a minimum cover formed by x_1 and x_2; this cover contains the two endpoints of e_a. It is clear that the Vertex Cover Problem remains NP-hard for all graphs containing Q as an isolated component. Thus let $G_Q = G \cup Q$ be such a graph,

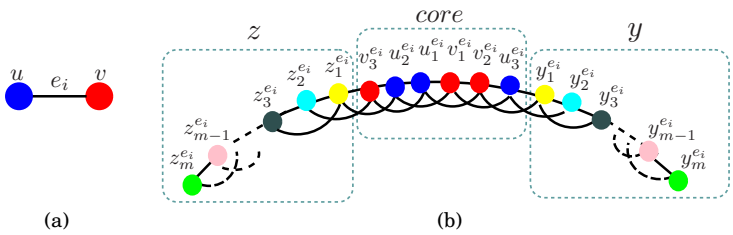

Fig. 5: (a) an edge e_i; (b) gadget corresponding to e_i.

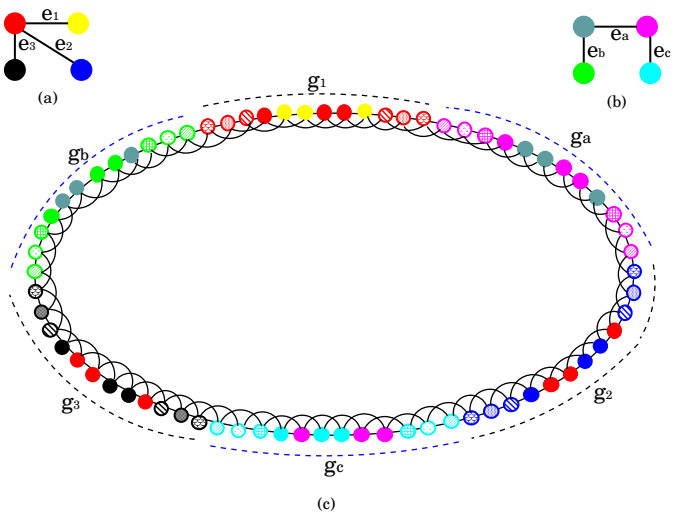

Fig. 6: (a) a graph G; (b) the graph Q; (c) the graph C_n^2 obtained from G_Q

where G is a graph with n vertices, m edges and a vertex cover of size k. Clearly, G_Q has a vertex cover of size $k+2$.

From G_Q we will construct a graph H isomorphic to a 2-power of a cycle.

- for each edge $e_i = (u, v)$ in G_Q create a gadget g_i in H as follows:
  - create vertices $u_1^{e_i}, u_2^{e_i}, u_3^{e_i}, v_1^{e_i}, v_2^{e_i}, v_3^{e_i}$ and edges $(v_3^{e_i}, u_2^{e_i})$, $(u_2^{e_i}, u_1^{e_i})$, $(u_1^{e_i}, v_1^{e_i})$, $(v_1^{e_i}, v_2^{e_i})$, $(v_2^{e_i}, u_3^{e_i})$;
  - create vertices $z_1^{e_i}, z_2^{e_i}, z_3^{e_i}, \ldots, z_m^{e_i}$, edges $(z_j^{e_i}, z_{j+1}^{e_i})$, $1 \leq j \leq m-1$, and edge $(z_1^{e_i}, v_3^{e_i})$;
  - create vertices $y_1^{e_i}, y_2^{e_i}, y_3^{e_i}, \ldots, y_m^{e_i}$, edges $(y_j^{e_i}, y_{j+1}^{e_i})$, $1 \leq j \leq m-1$, and edge $(y_1^{e_i}, u_3^{e_i})$;
  - add an edge between vertices x, y of g_i whenever they are at distance 2;
  - color $u_1^{e_i}, u_2^{e_i}, u_3^{e_i}$ with color c_u; $v_1^{e_i}, v_2^{e_i}, v_3^{e_i}$ with color c_v; and for every $1 \leq j \leq m$, vertices $z_j^{e_i}$ and $y_j^{e_i}$ with color $c_j^{e_i}$.

- after constructing the gadgets, for all $1 \leq i \leq m+2$, add $(z_m^{e_i}, y_m^{e_{i+1}})$;
- add edge $(z_m^{e_{m+3}}, y_m^{e_1})$;
- add an edge between x and y in H whenever they are at distance 2.

Each gadget g_i of H is divided in three parts: the core, consisting of vertices $u_1^{e_i}, u_2^{e_i}, u_3^{e_i}, v_1^{e_i}, v_2^{e_i}, v_3^{e_i}$; the z-arm, consisting of $z_1^{e_i}, \ldots, z_m^{e_i}$; and the y-arm, consisting of $y_1^{e_i}, \ldots, y_m^{e_i}$. We denote by g_a, g_b, g_c the gadgets associated with edges e_a, e_b, e_c, respectively. Figure 5 shows a gadget according to the construction above. Its core and arms are shown in detail. Figure 6 shows in (a) a graph G, in (b) the graph Q, and in (c) the power of cycle obtained from G_Q. Each detailed gadget, g_i, in (c) is equivalent to an edge e_i in G_Q, $i \in \{1, 2, 3, a, b, c\}$.

First, we will prove that if G_Q contains a vertex cover V' of size $k+2$ (i.e., G contains a vertex cover of size k) then the constructed graph H has a free-flooding with $m^2 + 5m + k + 5$ moves.

By construction, in every gadget g_i of H vertices $z_j^{e_i}$ and $y_j^{e_i}$ have the same color. Thus to make the arms of each gadget $g_i \neq g_a$ monochromatic in only m moves, we play one move in its core such that five vertices become colored with the same color. The remaining vertex (with another color) will be associated with a vertex of the cover in G_Q. After this move, we play m moves, starting from the core, to flood the arms. Figure 7 illustrates, as described above, a sequence of $m+1$ moves to flood the gadget presented in Figure $5(b)$.

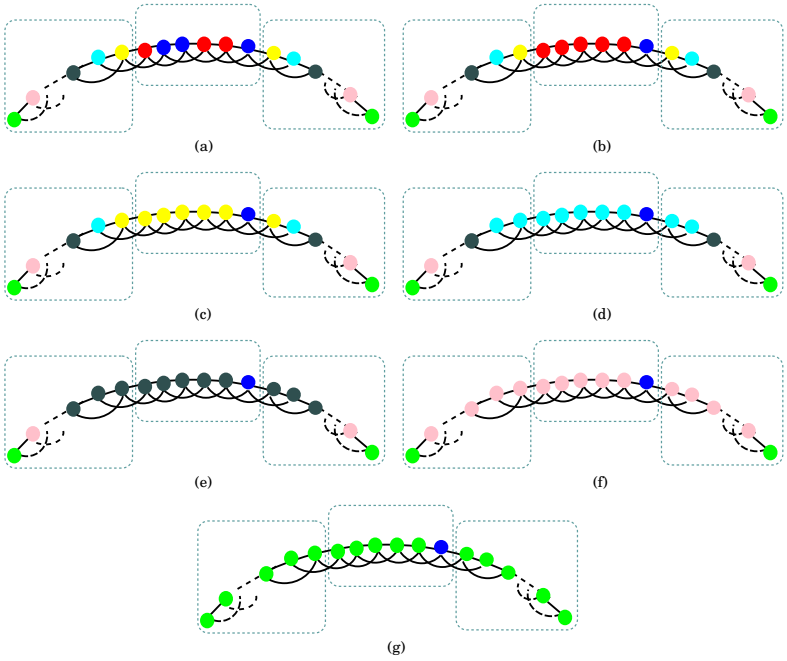

Fig. 7: A sequence of moves to flood the arms of the gadget presented in Figure $4(b)$.

At this point, $m^2 + 3m + 2$ moves were played and each gadget $g_i \neq g_a$ has only two colors, the color of its maximum mcs (denote by h_i this mcs) and the color of an island representing a vertex of V'. Figure $8(a)$ illustrates the graph presented in Figure $6(c)$ after these $m^2 + 3m + 2$. Note that none move was played in the gadget g_a. By playing $m + 1$ additional moves we create a big mcs H' from the h_i's, and by playing more m moves we obtain an mcs H'' which contains H' and both arms of g_a. Figure $8(b)$ illustrates the graph presented in Figure $6(c)$ after created H', and Figure $8(b)$ shows the graph after created H''.

Now vertices that do not belong to H'' are vertices of the core of g_a or islands of other gadgets. Since both endpoints of edge e_a lie in a minimum cover of Q, and assuming that the islands in g_b and g_c represent vertices x_1 and x_2, more two moves suffice to flood g_a, g_b and g_c.

As each gadget in H represent an edge in G_Q, and each color in a core represent a vertex in a edge, by construction, the coloring of the remaining islands represent a vertex cover of G. Considering that these remaining islands represent the minimum vertex cover of G, H can be flooded in at most k final moves. This gives a free-flooding of H in at most $m^2 + 5m + k + 5$ moves.

Conversely, assume that H has an optimal free-flooding S with $m^2 + 5m + k + 5$ moves. For two vertices a, b belonging, respectively, to the z-arm and the y-arm of a gadget g_i, note that a and b can be flooded by the same move m if and only if a and b have the same color c and, immediately before move m, there exists an mcs H' adjacent to a and b with color $c' \neq c$. When such an mcs H' exists we have two cases: (i) H' contains vertices of the core of g_i (H' is of *type 1*); (ii) H' is not of type 1, but contains vertices of the arms and the core of all the other gadgets (H' is of *type 2*).

It is easy to see that during the flooding only one gadget of H, say g_f, can contain vertices a, b such that: (1) a, b lie in distinct arms of g_f; (2) a, b are flooded by the same move, m, played on an mcs of type 2.

Hence, at least $m + 2$ gadgets forming a collection U contain no pair a, b of vertices as described. In each gadget g_j in U, the minimum number of moves required to create an mcs containing all vertices of its arms is $m + 1$, where one of the moves exclusively floods vertices in the core of g_j. Since S is an optimal free-flooding, without loss of generality we can assume that these subsequences of $m + 1$ moves for each gadget in U correspond to the first $(m + 2)(m + 1)$ moves in S. Figure $8(a)$ illustrates the graph presented in Figure $6(c)$ after a possible sequence of $m^2 + 3m + 2$ moves, as described.

After playing these moves, each gadget in U contains an mcs and an island. Now at least $m + 1$ moves are necessary to join all the $m + 2$ mcs's created so far into a single one. Thus there still remain $m + k + 2$ moves to be analyzed. Observe that joining the $m+2$ mcs's, we can use only m moves to flood all vertices in the arms of the remaining gadget, g_f, (gadget whith no flooded vertex). Note that g_f must exist to the free-flooding to be optimal.

At this point, let W' be the subset of vertices of H not yet flooded. They either lie in g_f's core or are islands in the core of other gadgets. As these vertices are flooded in $k + 2$ moves, since each gadget in H represents an edge of G_Q and vertices lying in the core of a gadget are associated with vertices of G_Q, by construction, the colors of the vertices in W' correspond to a vertex cover of G_Q of size $k + 2$. This gives a vertex cover of G of size at most k. □

Corollary 9 *Free-Flood-it remains NP-hard on P_n^2.*

Proof: First of all, construct a second power of a path, P_n^2, using a similar construction of Theorem 8 without the component Q, and desconsidering the first and the last gadgets as neighbors. After that,

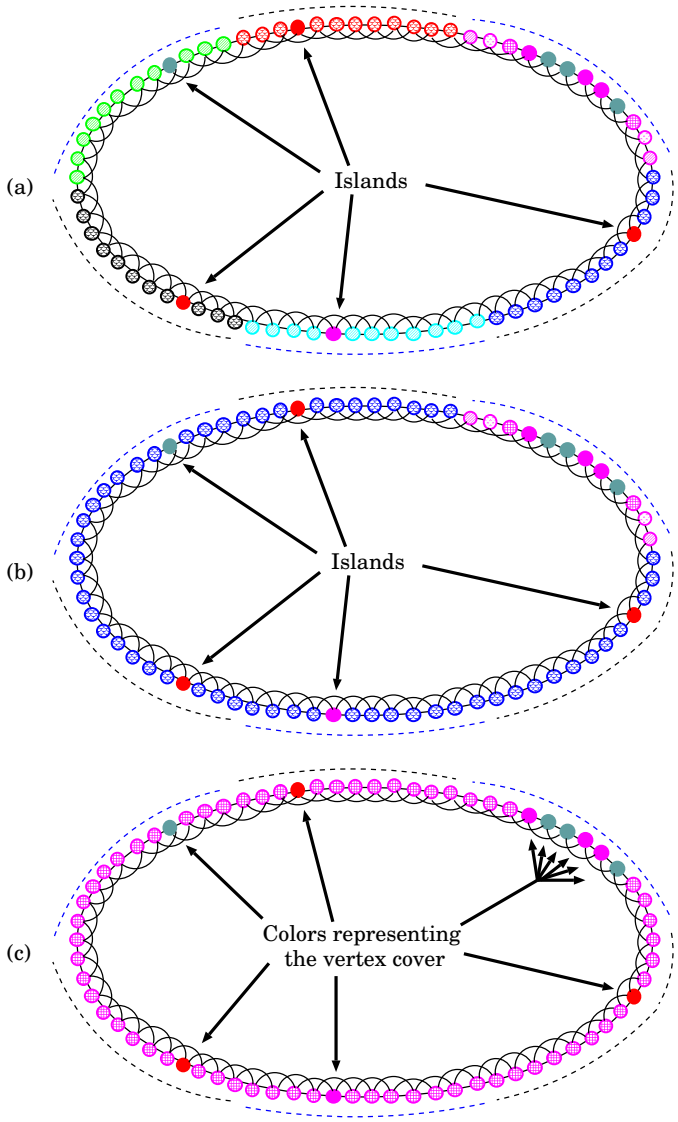

Fig. 8: States of the graph in Figure $5(c)$ during an optimal flooding.

we similarly can show that P_n^2 has a free-flooding of size $m^2 + 2m + k - 1$ if and only if G has a vertex cover of size k. □

Corollary 10 *Free-Flood-it remains NP-hard on* $2 \times n$ *circular grids.*

Proof: Use the construction in Lemma 5 (for n even) and Theorem 8. □

4 Conclusions

In this paper we have analyzed complexity aspects of flood-filling games, Flood-It and Free-Flood-it, which are very popular one-player combinatorial games. Many complexity issues on Flood-It and Free-Flood-It have recently been investigated, and these games have presented interesting behavior when are played on non-grid graphs such as trees or split graphs. Analyzing the computational complexity of these games on classes of graphs shuch as power of paths, power of cycles, circular grids and graphs with bounded vertex cover, we conclude that: (i) Flood-it can be solved in polynomial time on P_n^2, C_n^2, $2 \times n$ circular grids; (ii) Free-Flood-it is NP-hard on P_n^2, C_n^2 and $2 \times n$ circular grids.

References

[1] C. Aschwanden, Spatial Simulation Model for Infectious Viral Disease with Focus on SARS and the Common Flu, *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, IEEE Computer Society, Washington, DC, USA, 2004.

[2] P. Barone, P. Bonizzoni, G. D. Vedova, and G. Mauri, An Approximation Algorithm for the Shortest Common Supersequence Problem: An Experimental Analysis, In: *ACM Symposium on Applied Computing*, pp. 56–60, 2001.

[3] H. L. Boadlaender, G. Rodney, R. Downey, M. R. Fellows, D. Hermelin, On Problems without Polynomial Kernels, *Journal of Computer and System Sciences* 75 (2009) 423–434.

[4] R. Clifford, M. Jalsenius, A. Montanaro, B. Sach, The Complexity of Flood-Filling Games, *Theory of Computing Systems* 50:1 (2012) 72-92.

[5] M. R. Fellows, M. T. Hallett, U. Stege, Analogs & Duals of the MAST Problem for Sequences & Trees, Journal of Algorithms 49:1 (2003) 192–216.

[6] R. Fleischer, G. J. Woeginger, An Algorithmic Analysis of the Honey-Bee Game, *Theoretical Computer Science* 452 (2012) 75–87.

[7] H. Fukui, A. Nakanishi, R. Uehara, T. Uno, Y. Uno, The Complexity of Free Flooding Games, *Information Processing Society of Japan (IPSG) SIG Notes* 2011, 1-5.

[8] J. Guo, R. Niedermeier, Invitation to Data Reduction and Problem Kernelization, *ACM SIGACT News* 38 (2007) 31–45.

[9] A. Lagoutte, M. Noual, E. Thierry, Flooding games on graphs, *Discrete Applied Mathematics* 164:2 (2014) 532–538.

[10] K. Meeks, A. Scott, The Complexity of Flood-Filling Games on Graphs, *Discrete Applied Mathematics* 160 (2012) 959–969.

[11] K. Meeks, A. Scott, The Complexity of Free-Flood-It on $2 \times n$ Boards, *Theoretical Computer Science* 500 (2013) 25–43.

[12] K. Meeks, A. Scott, Spanning Trees and the Complexity of Flood Filling Games, Theory of Computing Systems 54:4 (2014) 731–753.

[13] S. Rahmann, The Shortest Common Supersequence Problem in a Microarray Production Setting, *Bioinformatics* 19, Suppl. 2 (2003) ii156–ii161.

[14] J. Sim, K. Park, The Consensus String Problem for a Metric is NP-complete, *Journal of Discrete Algorithms* 1:1 (2003) 111–117.

[15] U. S. Souza, F. Protti, M. Dantas da Silva, Parameterized Complexity of Flood-Filling Games on Trees, *Proceedings of COCOON*, *Lecture Notes in Computer Science* 7936 (2013) 531–542, Springer.