Vol.2011-AL-136 No.7
2011/9/6

The Complexity of Free Flood Filling Games

Hiroyuki Fukui[†1] Akihiro Nakanishi[†1] Ryuhei Uehara[†1]
Takeaki Uno[†2] Yushi Uno[†3]

Abstract The flood filling games on a graph is a kind of graph coloring game. The one player version of the game is known as Flood-It, and the two player version is known as Honey-Bee Game. We consider the case that the player can color arbitrary vertex. This version is called free flood filling game. We concentrate at the one player version of the free flood filling game on a graph. In this paper, we show that the free Flood-It is NP-complete on trees with only three colors, and it is polynomial time solvable on paths and cycles with any number of colors.

Keywords Flood-It, Honey-Bee game, graph coloring.

1. Introduction

The *flood filling game* is played on a precolored board, and each player colors a cell on the board in a turn. When a cell is colored with the same color as its neighbor, they will be merged into one colored area. If a player changes the color of one of the cells belonging to a colored area of the same color, the color of all cells in the area are changed. The game finishes when all cells are colored with one color. The one player flood filling game is known as Flood-It. In Flood-It, each cell is a precolored square, the board consists of $n \times n$ cells, the player always changes the color of the top-left corner cell, and the goal is to minimize the number of turns[⋆1]. The two player flood filling game is known as Honey-Bee game played on a honeycomb board [⋆2]. The board is also precolored, and each player alternately colors his/her specified hexagon with any color except the color of the other players specified hexagon. The game finishes when all cells belong to one of two areas containing the specified hexagons of the players. Then the player who eventually occupies more areas wins.

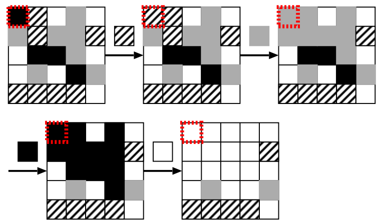
図 1 A sequence of five moves on a 5 × 5 Flood-It board.

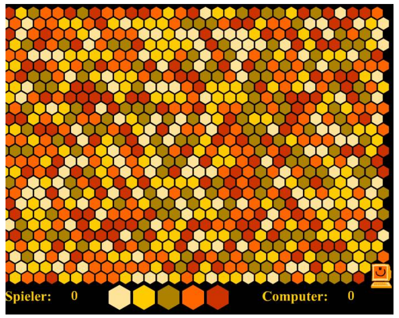

図 2 A snapshot of the Honey-Bee game. The human player changes the color of the top-left corner.

In the original flood filling games, each player colors a specified cell. However, it is natural to extend the game that the player can color any cell. We say that the original game is *fixed* and the extended game is *free*. The game board also can be generalized to a general graph; that is, the vertex set corresponds to the set of cells, and two cells are neighbors if and only if the corresponding vertices are adjacent in the graph. It is also natural to parameterize the number c of colors. Recently, the generalized flood filling game on a general graph is investigated from the viewpoint of computational complexity[1)–3)]. The known results can be summarized as follows:

- Two player, fixed, and general graphs: PSPACE-complete[2)].
- Two player, fixed, and series parallel graphs: NP-hard[2)].

†1 Japan Advanced Institute of Science and Technology
†2 National Institute of Informatics
†3 Osaka Prefecture University

⋆1 You can play Flood-It on the Web page at `http://floodit.appspot.com/`.
⋆2 You can play Honey-Bee game on the Web page at `http://www.ursulinen.asn-graz.ac.at/Bugs/htm/games/biene.htm`.

- One player, fixed, split graphs, and c is unbounded: NP-complete[2].
- One player, fixed, split graphs, and c is a constant: in P[2].
- One player, fixed, trees, and $c \geq 3$: NP-complete[2].
- One player, fixed, and co-comparability graphs: in P[2].
- One player, free/fixed, general graphs, and $c \geq 3$: NP-complete[1].
- One player, free/fixed, general graphs, and $c \leq 2$: in P[3].

In this paper, we add two new results as follows:

Theorem1 *The free flood filling game for one player is* NP*-complete even on trees with three colors.*

Theorem 1 improves one of the results in[2] from "fixed" to "free."

Theorem2 *The free flood filling game for one player on a path is in* P*. More precisely, given a colored path of length n with $|C|$ colors, the optimal way to flood fill the path can be found in $O(|C|n^3)$ time and $O(|C|n^2)$ space.*

Using the same resources, we can extend Theorem 2 to cycles.

2. Preliminaries

We model the flood filling game in the following graph-theoretic manner. The game board is a connected, simple, loopless, undirected graph $G = (V, E)$. We denote by n and m the number of vertices and edges, respectively. There is a set C of colors, and every vertex $v \in V$ is precolored (as input) with some color $col(v) \in C$. Note that we may have an edge $\{u, v\} \in E$ with $col(u) = col(v)$. For a color $c \in C$, the subset V_c contains all vertices in V of color c. For a vertex $v \in V$ and color $c \in C$, we define the *color-c-neighborhood* $N_c(v)$ as the set of vertices in V_c either adjacent to v or connected to v by a path of vertices of color c. Similarly, we denote by $N_c(W) = \cup_{w \in W} N_c(w)$ the color-c-neighborhood of a subset $W \subseteq V$. For a given graph $G = (V, E)$ and the coloring $col()$, a coloring operation (v, c) for $v \in V$ and $c \in C$ is defined by, for each vertex $v' \in N_{c'}(v) \cup \{v\}$ with $c' = col(v)$, setting $col(v') = c$. For a given graph $G = (V, E)$ and a sequence $((v_1, c_1), (v_2, c_2), \ldots, (v_k, c_k))$ of coloring operations in $V \times C$, we let $G_0 = G$ and G_i is the graph obtained by the coloring operation (v_i, c_i) on G_{i-1} for each $i = 1, 2, \ldots, k$. In the case, we denote by $G_{i-1} \rightarrow_{(v_i, c_i)} G_i$ and $G_0 \rightarrow^i G_i$ for each $0 \leq i \leq k$. Then the problems in this paper are defined as follows:

Problem 1: Fixed flood filling game

Input : A graph $G = (V, E)$, a vertex $s \in V$, and an integer k such that each vertex in V is precolored with $col(v) \in C$;

Output: Determine if there is a sequence of operations $((s, c_1), (s, c_2), \ldots, (s, c_k))$ of length k such that all vertices in the resulting graph G' (i.e. $G \rightarrow^k G'$) have the same color;

Problem 2: Free flood filling game

Input : A graph $G = (V, E)$ and an integer k such that each vertex in V is precolored with $col(v) \in C$;

Output: Determine if there is a sequence of operations $((v_1, c_1), (v_2, c_2), \ldots, (v_k, c_k))$ of length k such that all vertices in the resulting graph G' (i.e. $G \rightarrow^k G'$) have the same color;

For these problems, if a sequence of operations of length k colors the graph, the sequence is called *solution* of length k.

3. NP-completeness on trees

In this section, we show NP-completeness of the free flood filling game on trees even with three colors. This is based on the NP-completeness of a similar problem; the fixed flood filling game. As mentioned in Introduction, the fixed flood filling game on trees is NP-complete even with three colors[2]. We reduce this fixed game to our free game. Let $T = (V, E)$, s, and k be the input of the fixed flood filling game. That is, T is a tree, s is the fixed vertex in V we can color, and k is the number of turns to color all vertices. The reduction is simple. We first make n copies $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$, $\ldots$, $T_n = (V_n, E_n)$ of the tree T, where $n = |V|$. All vertices in T_is are distinct except the copies of s; all the trees share the specified vertex s. Let $\mathcal{T}$ be the resulting graph. That is, the vertex set of $\mathcal{T}$ is $V_1 \cup V_2 \cup \cdots \cup V_n$ which is disjoint union of n copies of $V \setminus \{s\}$ and the unique vertex s. Hence the number of vertices in $\mathcal{T}$ is $n(n-1)+1 = n^2 - n + 1$.

Clearly, $\mathcal{T}$ is a tree. The reduction can be done in polynomial time. We now show the following lemma:

Lemma3 *The fixed flood filling game on T has a solution of length k if and only if the free flood filling game on $\mathcal{T}$ has a solution of length k.*

Proof. Without loss of generality, we assume that all vertices in T are colored by a sequence of coloring operations $((s,c_1),(s,c_2),\ldots,(s,c_k))$ for given k, and this is a shortest solution for the problem. Clearly, all vertices in $\mathcal{T}$ are also colored by this sequence since $\mathcal{T}$ consists of n copies of T that share the common vertex s. Hence, it is sufficient to show that $\mathcal{T}$ cannot have any solution of length $k' < k$.

We first observe that any connected graph $G = (V,E)$ has a solution of length at most $|V| - 1$; pick any edge $e = (u,v)$ with $col(u) \neq col(v)$, change the color to make $col(u) = col(v)$, and repeat this process until all the vertices have the same color. This greedy algorithm eventually makes all vertices in the same color after at most $|V| - 1$ coloring operations.

To derive a contradiction, we assume that all vertices in $\mathcal{T}$ are colored by a sequence of coloring operations $((v_1,c'_1),(v_2,c'_2),\ldots,(v_{k'},c'_{k'}))$ with $k' < k$. By the observation and assumption, we have $k' < k < n$. Then, since we have n copies of T, there exists a subtree $T_i = (V_i, E_i)$ in $\mathcal{T}$ such that V_i contains no vertex in the sequence. That is, all vertices in V_i are colored by changing the color of s. Hence this sequence is also the solution of T for the fixed flood filling game, which contradicts that any shortest solution of T is of length k. ∎

Theorem 1 immediately follows Lemma 3. We note that using the result in[3], we can show that the free flood filling game on trees is polynomial time solvable if the number of colors is at most 2. Hence the number three of colors in Theorem 3 cannot be improved unless P = NP.

4. Polynomial time algorithm on paths

In this section, we assume that $G = (V,E)$ is a path P_n of length $n-1$; that is, $V = \{v_1,\ldots,v_n\}$ and $E = \{\{v_i,v_{i+1}\} \mid 1 \leq i \leq n-1\}$. We denote by $P[i,j]$ the subpath induced by $\{v_i,v_{i+1},\ldots,v_j\}$ (e.g. $P_n = P[1,n]$). We first employ a standard dynamic programming technique with the following table:

$T[i,j,c]$: the minimum number of coloring operations to make all the vertices $v_i, v_{i+1}, \ldots, v_j$ in the color c.

We note that we do not take care of the colors of the vertices v_{i-1} and v_{j+1}. In other words, we do not mind if $col(v_{i-1}) = col(v_i)$ or $col(v_{i-1}) \neq col(v_i)$. For each $i = 1,2,\ldots,n$, we initialize as follows:

$$T[i,i,c] = \begin{cases} 0 & \text{if } col(v_i) = c \\ 1 & \text{otherwise.} \end{cases}$$

Then, with careful case analysis, for each i and j with $i < j$ and each color $c \in C$, we obtain the following relationship for all possible i' and j' with $i < i' < j' < j$ and color $c' \neq c$:

$$\begin{aligned} T[i,j,c] = \min_{i<i'<j'<j, c'\neq c} \{ & \\ & T[i,i',c'] + 1 + T[i'+1,j,c], \\ & \quad \text{(color the left part of color } c' \text{ with color } c) && (1) \\ & T[i,i',c] + T[i'+1,j,c'] + 1, \\ & \quad \text{(color the right part of color } c' \text{ with color } c) && (2) \\ & T[i,i',c] + T[i'+1,j',c'] + 1 + T[j'+1,j,c], \\ & \quad \text{(color the vertices in } P[i'+1,j'] \text{ of color } c') && (3) \\ & T[i,j,c'] + 1 \qquad \text{(color all vertices of color } c' \text{ with color } c) && (4) \\ \} \end{aligned}$$

Then, the optimal solution is obtained by evaluation of $\min_c T[1,n,c]$. Now we consider an efficient computation of the table.

Lemma4 *The table $T[i,j,c]$ can be computed in $O(|C|n^3)$ time and $O(|C|n^2)$ space.*

Proof. By a straightforward implementation, the table $T[i,j,c]$ can be computed in $O(|C|^2 n^4)$ time and $O(|C|n^2)$ space as follows: First, the algorithm initializes $T[i,i,c]$ for each i defined above in $O(cn)$ time. We let $\ell = j - i$. The algorithm next fills the table $T[i,j,c]$ for each $\ell = 1,2,\ldots,n-1$ and $c \in C$. For each ℓ, there are $n - \ell + 1$ pairs of (i,j) with $i < j$. We fix $\ell = j - i$ and i and j. Then the computation of $T[i,i',c'] + 1 + T[i'+1,j,c]$ in (1) and $T[i,i',c] + T[i'+1,j,c'] + 1$ in (2) takes $O(|C|\ell)$

time since i' takes $\ell - 1$ different values and c' takes $|C| - 1$ different values. To compute $T[i,i',c] + T[j',j,c] + T[i',j',c] + 1$ in (3), the number of possible combinations of i' and j' with $i < i' < j' < j$ is $\binom{j-i-2}{2}$. Thus the computation of (3) takes $O(|C|\ell^2)$ time. The computation of (4) for a fixed ℓ is postponed until all computations of (1), (2), and (3) for the ℓ are done. After that, the algorithm first finds the minimum value of $T[i,j,c]$ for each $c \in C$. Then, it updates $T[i,j,c]$ properly. Thus, for each fixed pair (i,j) the computation of (4) requires $O(|C|)$ time.

Therefore, for each $\ell = 1, 2, \ldots, n-1$ and $c \in C$, the computation of (1), (2), (3) takes $(n - \ell + 1)O(|C|\ell^2)$ time, and the computation of (4) requires $(n - \ell + 1)O(|C|)$ time. Since the choices of each ℓ are $n - \ell + 1$ and that of c are $|C|$, the total computation time is $O(|C|^2 n^4)$.

Now we turn to an efficient implementation.

The most costly computation is (4), which requires to compute $T[i,i',c] + T[i'+1,j',c'] + 1 + T[j'+1,j,c]$. This is the case that the paths $P[i,i']$ and $P[j'+1,j]$ are colored with c, and $P[i'+1,j']$ is colored with c'. Then we pick any vertex in $P[i'+1,j']$ and color it with color c, and obtain the path $P[i,j]$ colored with c. This situation can be regarded as follows. We have the paths $P[i'+1,j']$ of color c' and $P[j'+1,j]$ of color c. Then we pick up any vertex in $P[i'+1,j']$ and color it with color c, and obtain the path $P[i'+1,j]$ colored with c. After that, we join the path $P[i,i']$ of color c and the other path $P[i'+1,j]$ of color c with no coloring operation. From this viewpoint, this cost is exactly given by $T[i,i',c] + T[i'+1,j,c]$. That is, we obtain $T[i,i',c] + T[i'+1,j',c'] + 1 + T[j'+1,j,c] = T[i,i',c] + T[i'+1,j,c]$.

For a faster implementation, we introduce a new table $\bar{T}[i,j,c]$ defined as follows:

$$\bar{T}[i,j,c] = \min_{c' \neq c} T[i,j,c']$$

That is, the table $\bar{T}[i,j,c]$ gives the minimum cost to color the path $P[i,j]$ with any color but c. Instead of "taking the minimum value for all $c' \in C \setminus \{c\}$" to compute (1) to (4), we can obtain the value from this new table. Furthermore, since $T[i,j,c]$ cannot be better than $\bar{T}[i,j,c]$, we can define

$$T[i,j] = \min_{c \in C} T[i,j,c]$$

and use it instead of $\bar{T}[i,j,c]$ defined above. This observation simplifies the algorithm description furthermore.

Using these tricks, we can implement the algorithm in Algorithm 3. Since we can assume that $|C| \leq n$, it is easy to see that this modified algorithm runs in $O(|C|n^3)$ time with $O(|C|n^2)$ space. ∎

Algorithm 3: Naive implementation for computing $T[i,j,c]$

```
Input  : A path $P_n = (V,E)$ of length $n-1$ such that each vertex in $V$ is
         precolored with $col(v) \in C$;
Output: The minimum number of coloring operations to color $P_n$ with a color
         $c \in C$;
foreach $i = 1, 2, \ldots, n$ do
    foreach $c \in C$ do
        if $col(v_i) = c$ then $T[i,i,c] = 0$ else $T[i,i,c] = 1$;
    $T[i,i] = 0$;
foreach $c \in C$ do
    foreach $\ell = 1, 2, \ldots, n-1$ do
        foreach $i = 1, 2, \ldots, n-\ell$ do
            $j = i + \ell$;
            $T[i,j,c] = n$ ;                                  /* Trivial upper bound */
            foreach $i' = i+1, i+2, \ldots, j-2$ do
                $T[i,j,c] = \min\{T[i,j,c], T[i,i'] + 1 + T[i'+1,j,c], T[i,i',c] + T[i'+1,j] + 1, T[i,i',c] + T[i'+1,j,c]\}$;
            $T[i,j] = n$ ;                                    /* Trivial upper bound */
            foreach $c' \in C$ do
                $T[i,j] = \min\{T[i,j], T[i,j,c']\}$;
            $T[i,j,c] = \min\{T[i,j,c], T[i,j] + 1\}$;
output $T[1,n]$;
```

Vol.2011-AL-136 No.7
2011/9/6

Corollary5 *The free flood filling game for one player on a cycle is in* P*. The running time is as the same as Theorem 2.*

Proof. In order to deal with a cycle, we have to modify the definition of the table $T[i,j,c]$. In Lemma 4, $T[i,j,c]$ is the table for the interval $[i,j]$ with $i<j$. For a cycle, we extend it to the case $j>i$ that means the interval $[j,j+1,\ldots,n-1,n,1,2,\ldots,i]$. The modification is straightforward, and the running time is the same up to constant factor. ∎

5. Concluding remarks

In this paper, we show that the free flood filling game is intractable even on trees, and tractable on paths and cycles. Intuitively, to solve the game efficiently, it seems that we need some kind of linear structure. Hence the investigation of the complexity of the game on interval graphs and their subclasses is nice future work.

参 考 文 献

1) David Arthur, Raphaël Clifford, Markus Jalsenius, Ashley Montanaro, and Benjamin Sach. The Complexity of Flood Filling Games. In *FUN 2010*, pages 307–318. Lecture Notes in Computer Science Vol.6099, Springer-Verlag, 2010.
2) Rudolf Fleischer and GerhardJ. Woeginger. An Algorithmic Analysis of the Honey-Bee Game. In *FUN 2010*, pages 178–189. Lecture Notes in Computer Science Vol.6099, Springer-Verlag, 2010.
3) Aurélie Lagoutte. 2-Free-Flood-It is polynomial. Technical report, arXiv:1008.3091v1, 2010.