IPSJ SIG Technical Report

Vol.2012-AL-139 No.7
2012/3/14

On Complexity of Flood Filling Games on Interval Graph Classes

Hiroyuki Fukui,[†1] Ryuhei Uehara,[†1] Takeaki Uno[†2] and Yushi Uno[†3]

The flooding games on a graph are a kind of graph coloring game, which are called Flood-It, Mad Virus, or HoneyBee played online. These games are supposed that each player can color one fixed vertex. It is natural to consider these games that allow the player to color arbitrary vertex. This version is called free flooding game. Recently, computational complexities of these games on some graph classes are studied. For example, one player version is NP-complete on a tree with maximum degree 3. In this paper, we investigate these games on some graph classes characterized by interval representations. Our results state that the number of colors is a key parameter to determine the difficulty of these games. When the number of colors is a fixed constant, these games can be solved in polynomial time on an interval graph. On the other hand, if the number of colors is not bounded, the free flooding game is NP-complete on a proper interval graph, which has an interval representation with intervals of unit length.

Keywords: Computational complexity, fixed parameter tractable, flooding game, graph coloring, interval graph.

1. Introduction

The *flooding 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, and the objective of the game is to minimize the number of turns (or

†1 北陸先端科学技術大学院大学 情報科学研究科 School of Information Science, Japan Advanced Institute of Science and Technology (JAIST)
†2 国立情報学研究所 National Institute of Informatics (NII)
†3 大阪府立大学理学系研究科 School of Science, Osaka Prefecture University

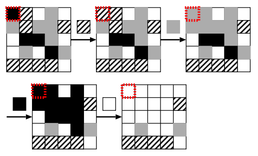
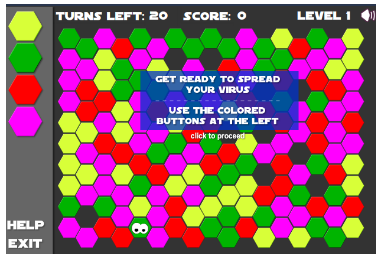


Fig. 1 A sequence of five moves on a 5 × 5 Flood-It board.


TURNS LEFT: 20
SCORE: 0
LEVEL 1
GET READY TO SPREAD YOUR VIRUS
USE THE COLORED BUTTONS AT THE LEFT
click to proceed
HELP
EXIT
Fig. 2 The initial screen of the Mad Virus (`http://www.bubblebox.com/play/puzzle/539.htm`). The player changes the cell having eyes.

to finish the game within a given number of turns). The one player flooding game is known as Flood-It (Figure 1). 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. This game is also called Mad Virus played on a honeycomb board (Figure 2). You can play all the games online: Flood-It (`http://floodit.appspot.com/`) and Mad Virus (`http://www.bubblebox.com/play/puzzle/539.htm`).

In the original flooding 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 one 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 k of colors. Recently, the generalized flooding game on a general graph is well investigated from the viewpoint of computational complexity. We summarize resent results in Table 1.

In this paper, we first investigate the computational complexities of the game on a graph that has an interval representation. The following results imply that the game is fixed parameter tractable with respect to the number of colors:

Theorem 1 The free flooding game is NP-complete even on a proper interval graph.

Theorem 2 (1) The free flooding game on a proper interval graph can be

Graph classes	fixed	fixed, k is bounded
general graphs	NP-C	NP-C if $k \geq 3$ 1) P if $k \leq 2$ (trivial)
(□/△/hex.) grids	NP-C	NP-C if $k \geq 3$ 2)
paths/cycles	$O(n^2)$ 2)	$O(n^2)$ 2)
co-comparability graphs	P 3)	P 3)
split graphs	NP-C 3)	P 3)
proper interval graphs	P[⋆1]	$O(8^k k^2 n^3)$ (This)
interval graphs	P[⋆1]	$O(8^k k^2 n^3)$ (This)

Graph classes	free	free, k is bounded
general graphs	NP-C	NP-C if $k \geq 3$ 1) P if $k \leq 2$ 2), 4)
(□/△/hex.) grids	NP-C	NP-C if $k \geq 3$ 2)
paths/cycles	$O(n^3)$ 5)[⋆2]	$O(n^3)$ 5)[⋆2]
split graphs	NP-C (This)	$O((k!)^2 + n)$ (This)
proper interval graphs	NP-C (This)	$O(8^k k^2 n^3)$ (This)
interval graphs	NP-C (This)	$O(8^k k^2 n^3)$ (This)

Table 1 Computational complexities of the flooding games on some graph classes.

solved in $O(8^k k^2 n^3)$ time. (2) The free flooding game on an interval graph can be solved in $O(8^k k^2 n^3)$ time. That is, the free flooding game in these graph classes is polynomial time solvable if the number k of colors is fixed.

We also extend the results for the fixed flooding game on a split graph mentioned in 3) to the free flooding game on a split graph. Precisely, the free flooding game is NP-complete even on a split graph, and it can be solved in $O((k!)^2 + n)$ time.

Although we only consider one player game in this paper, it is also natural to consider two player flooding game. This is known as HoneyBee, which is available online at `http://www.ursulinen.asn-graz.ac.at/Bugs/htm/games/biene.htm`. Fleischer and Woeginger also investigate this game from the viewpoint of computational complexity. See 3) for further details.

⋆1 The class of co-comparability graphs properly contains interval graphs and hence proper interval graphs. Since this game is polynomial time solvable on a co-comparability graph, so they follow.

⋆2 In 5), the authors gave an $O(kn^3)$ algorithm. However, it can be improved to $O(n^3)$ easily in the same way in 2).

2. Preliminaries

We model the flooding 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 $C = \{1, 2, \ldots, k\}$ 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 precoloring $col()$, 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 $(v_1, c_1), (v_2, c_2), \ldots, (v_t, c_t)$ 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 $i = 1, 2, \ldots, t$. 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 $0 \leq i \leq t$. Then the problem in this paper are defined as follows[⋆3]:

Problem 1: Free flooding game

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

Output: Determine if there is a sequence of coloring operations $((v_1, c_1), (v_2, c_2), \ldots, (v_t, c_t))$ of length t such that all vertices in the resulting graph G' (i.e. $G \rightarrow^k G'$) have the same color;

For the problem, if a sequence of operations of length t colors the graph, the sequence is called a *solution* of length t.

A graph (V, E) with $V = \{v_1, v_2, \cdots, v_n\}$ is an *interval graph* if there is a set of intervals $\mathcal{I} = \{I_{v_1}, I_{v_2}, \cdots, I_{v_n}\}$ such that $\{v_i, v_j\} \in E$ if and only if $I_{v_i} \cap I_{v_j} \neq \emptyset$ for each i and j with $1 \leq i, j \leq n$. We call the set $\mathcal{I}$ of intervals an *interval*

⋆3 In the fixed flooding game, $v_1 = v_2 = \cdots = v_t$ is also required.

representation of the graph. For each interval I, we denote by $L(I)$ and $R(I)$ the left and right endpoints of the interval, respectively (hence we have $L(I) \leq R(I)$ and $I = [L(I), R(I)]$). For a point p, let $N[p]$ denote the set of intervals containing the point p.

An interval representation is *proper* if no two distinct intervals I and J exist such that I properly contains J or vice versa. That is, either $I \prec J$ or $J \prec I$ holds for every pair of intervals I and J. An interval graph is *proper* if it has a proper interval representation. If an interval graph G has an interval representation $\mathcal{I}$ such that every interval in $\mathcal{I}$ has the same length, G is said to be a *unit interval graph*. Such interval representation is called a *unit interval representation*. It is well known that proper interval graphs coincide with unit interval graphs[6)]. That is, given a proper interval representation, we can transform it to a unit interval representation. A simple constructive way of the transformation can be found in 7). With perturbations if necessary, we can assume without loss of generality that $L(I) \neq L(J)$ (and hence $R(I) \neq R(J)$), and $R(I) \neq L(J)$ for any two distinct intervals I and J in a unit interval representation $\mathcal{I}$.

A graph $G = (V, E)$ is a *split graph* if V can be partitioned into C and I such that $G[C]$ induces a clique and $G[I]$ induces an independent set.

3. Interval Graphs

Let $G = (V, E)$ be an interval graph precolored with at most k colors. We first show the NP-completeness of the flooding game on G even if G is a proper interval graph. Next we show an algorithm that solves the flooding game in $O(8^k k^2 n^3)$. That is, the flooding game is fixed parameter tractable on an interval graph with respect to the number of colors.

3.1 NP-completeness on proper interval graphs

To prove Theorem 1, we reduce the following well-known NP-complete problem to our problem (See 8) [GT1]):

Problem 2: Vertex Cover

Input : A graph $G = (V, E)$ and an integer k;

Output: Determine if there is a subset S of V such that for each edge $e = \{u, v\} \in E$, $e \cap S \neq \emptyset$ and $|S| = k$;

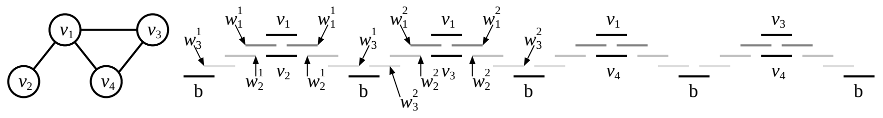


Fig. 3 Reduction from Vertex Cover to Flooding game.

Let $G = (V, E)$ and k be an instance of the vertex cover problem. Let $n = |V|$, $m = |E|$. We construct a proper interval representation $\mathcal{I}$ and coloring of the graph[⋆1] as follows (see also Figure 3).

(1) Let C be the color set $V \cup \{w_i^j \mid 1 \leq i \leq m-1, 1 \leq j \leq m\} \cup \{b\}$ of $n + m(m-1) + 1$ different colors.

(2) For each $0 \leq i \leq m$, we put an interval $I_i = [4i, 4i+1]$ with precolor $col(I) = b$. We call these $m+1$ intervals *backbones*.

(3) For each $e_i = \{u, v\} \in E$ with $0 \leq i < m$, we add two identical intervals $J_i = [4i+2, 4i+3]$ and $J'_i = [4i+2, 4i+3]$ with precolor $col(J_i) = u$ and $col(J'_i) = v$. (Note that the ordering of the edges is arbitrary.)

(4) We join backbones and the pair intervals by a path of length m; precisely, the intervals $I = [4i, 4i+1]$ and $J_i = [4i+2, 4i+3]$ are joined by a path $(v_m, v_{m-1}, \ldots, v_1, v_0)$ such that $I_{v_m} = [4i, 4i+1]$ and $I_{v_0} = [4i+2, 4i+3]$. (Note that v_1 has three neighbors v_2, v_0 and another vertex v'_0 with $I_{v_0} = I_{v'_0} = [4i+2, 4i+3]$.) Then each vertex v_j with $1 \leq j \leq m-1$ is precolored by w_j^i. The intervals $I_i = [4i+2, 4i+3]$ and $I = [4i+4, 4i+5]$ are joined in a symmetric way. That is, they are joined by a path $(v_0, v_1, \ldots, v_{m-1}, v_m)$ such that $I_{v_0} = [4i+2, 4i+3]$, $I_{v_m} = [4i+4, 4i+5]$, and $col(v_j) = w_j^i$ with $1 \leq j \leq n$.

Now we show a lemma that immediately implies Theorem 1.

Lemma 3 In the reduction above, the original graph G has a vertex cover of size k if and only if there is a sequence of coloring operations of length $m^2 + k$ to make the resulting interval representation in monochrome.

Proof. We first suppose that the graph G has a vertex cover S of size k. Then we can construct a sequence of coloring operations of length $m^2 + k$ as follows. First step is joining the backbones. For an edge $\{u, v\} \in E$ with $u \in S$, pick up

⋆1 We sometimes identify an interval graph and its interval representation.

v (we do not mind if v is in S). Then we color v by $w_1, w_2, \ldots, w_{m-1}$, and b. Repeat this process for every edge. Then all the backbones are connected and colored by b after m^2 colorings. We then still have $|E|$ intervals corresponding to the vertices in S. Thus we pick up each vertex v in S and color the backbone by $col(v)$. After $|S|$ colorings, all vertices become monochrome.

Next we suppose that we have a sequence of coloring operations of length m^2+k that makes the representation in monochrome. We extract a vertex cover of size k from this operations. In the representation, for each i with $1 \le i \le m$, we have 2 distinct paths $(w^i_1, w^i_2, \ldots, w^i_{m-1})$. Hence we have $2m$ distinct paths in total, and each of them requires m coloring operations. Since k is the (potential) size of a vertex cover, we can assume that $k < m$ without loss of generality. First, we observe that the sequence of coloring operation includes (v, w^i_1) or (u, w^i_1) for each edge $e_i = (u, v)$. Otherwise, we need $2m$ coloring operations to connect the neighboring backbones (colored b). The operations never help to connect other backbones. Thus the length of any sequence is no less than $m^2 + m$. Therefore, we can see either (v, w^i_1) or (u, w^i_1) appears in the sequence. We say that v is *selected* if (v, w^i_1) appears before (u, w^i_1) (or (u, w^i_1) may not appear).

Let R be the set of vertices v such that it is not selected in some edge $e_i = (v, u)$. Then R is a vertex cover. Since the sequence makes all the vertices in monochrome, the sequence includes either (v, w_1) or $(*, v)$ for each unselected vertex v and edge $e_i = (v, u)$. We call such operations *cover operations*. Thus, the number of cover operations is no less than $|R|$. Remind that m^2 operations are needed to connect the selected intervals and paths, and these operations are either of form $(*, w^i_j)$ or $(u, *)$ for selected u. This implies that the length of the sequence is no less than $m^2 + |R|$, and thus $|R| = k$. ∎

The reduction can be done in polynomial time, and the flooding game is clearly in NP. Hence, by Lemma 3, Theorem 1 immediately follows.

3.2 Polynomial time algorithm on interval graphs

We first assume that $G = (V, E)$ is a proper interval graph, and later, we extend the algorithm for an interval graph.

3.3 Algorithm for a proper interval graph

Let $\mathcal{I}(G)$ be an interval representation of the proper interval graph $G = (V, E)$. The interval representation is given in a compact form (see 9) for the details). Precisely, each endpoint is a positive integer, $N[p] \ne N[p+1]$ for each integer p, and there are no indices $N[p] \subset N[p+1]$ or vice versa for each integer p with $N[p] \ne \emptyset$ (otherwise we can shrink it). Intuitively, each integer point corresponds to a set of different endpoints since the representation has no redundancy. Then, it is known that $\mathcal{I}(G)$ is unique up to isomorphism (see 10)), and it is clear that $\mathcal{I}(G)$ can be placed in $[0..P]$ for some $P \le 2n-1$. Moving a point p from 0 to P on the representation, the color set $N[p]$ differs according to p. More precisely, we obtain $2P+1$ different color sets for each $p = 0, 0.5, 1, 1.5, 2, 2.5, \ldots, P-0.5, P$. Let S_i be the color set obtained by the ith p (to simplify, we use from S_0 to S_{2P}). Since the color set C has size k, each S_i consists of at most k colors. That is, the possible number of color sets is $2^k - 1$ (since $S_i \ne \emptyset$).

Now we can regard the interval representation as a path $\mathcal{P} = (\hat{S_0}, \hat{S_1}, \ldots, \hat{S_{2P}})$, where each vertex $\hat{S_i}$ is precolored by the color set S_i. Then we can use a dynamic programming technique, which is similar to the algorithms for the flooding game on a path[2),5)]. However, we have to take care of the influence of changing a color of a vertex. In the algorithms for an ordinary path[2),5)], when the color of v is changed, it has an influence to two neighbors of it. In our case, when we change a color c in S_i to c', all reachable color sets joined by c from $\hat{S_i}$ are changed. Thus we have to remove c from S_j and add c' to S_j for each j with $i' \le j \le i''$, where i' and i'' are the leftmost and the rightmost vertices reachable from $\hat{S_i}$ joined by the color c. This can be handled by the dynamic programming table. But by this coloring operation, some colors may be left independent on the backbone of color c'. To deal with these color sets, we maintain a table $f(\ell, r, c, S)$ that is the minimum number of coloring operations to satisfy the following conditions: (1) $c \in S_i$ for each i with $\ell \le i \le r$, and (2) $\cup_{\ell \le i \le r} S_i \subseteq (S \cup \{c\})$. That is, $f(\ell, r, c, S)$ gives the minimum number of coloring operations to make this interval connected by the color c, and the remaining colors in this interval is contained in S. Once we obtain $f(0, 2P, c, S)$ for all c and S on $\mathcal{P}$, that implies the solution by taking

$\min_{c,S}(f(0,2P,c,S)+|S|)$. This function satisfies the following recursive relation.

$$\begin{array}{lll} f(\ell,r,c,S) = \min\{ & & \\ \min_{\ell<i\le r} f(\ell,i-1,c,S') + f(i,r,c',S'') + 1 & \text{such that} & S',S'' \subseteq S\cup\{c\} \\ \min_{\ell<i\le r} f(\ell,i-1,c',S') + 1 + f(i,r,c,S'') & \text{such that} & S',S'' \subseteq S\cup\{c\} \\ \min_{\ell<i\le r} f(\ell,i-1,c,S') + f(i,r,c,S'') & \text{such that} & S',S'' \subseteq S \\ \} & & \end{array}$$

Hence we have the following lemma.

Lemma 4 The value of $\min_{c,S}(f(0,2P,c,S)+|S|)$ can be computed in $O(8^k k^2 n^3)$ time.

Proof. This can be done in a standard dynamic programming technique. Initialization step is that, for each i, $f(i,i,c,S)=0$ if $c\in S$ and $f(i,i,c,S)=1$ if $c\notin S$. This step requires $(2P+1)\times k\times 2^k = O(2^k kn)$.

In general step, the algorithm computes $f(\ell,r,c,S)$ for each pair ℓ and r with $\ell<r$. The algorithm computes all pairs ℓ and r in the order $r-\ell=1$, $r-\ell=2$, $r-\ell=3$, ..., $r-\ell=2P$. For a pair ℓ,r with $\ell<r$, the algorithm next fix the color c. Then the algorithm generates all possible subsets S of C. Using the above recursive relation, a value of $f(\ell,r,c,S)$ can be computed in $O(k(\ell-r)2^{|S|+1}\cdot 2^{|S|+1})$ time. Therefore, in total, $f(0,2P,c,S)$ can be computed in $O(n^2\cdot k\cdot 2^k\cdot k\cdot n\cdot 2^k\cdot 2^k) = O(8^k k^2 n^3)$ time. This completes the proof. ∎

We note that we assume that whether a color c is in a color set S or not can be determined in $O(1)$ time. If $|S|$ is large, say $O(n)$, the running time increases to $O(8^k k^2(\log k)n^3)$.

By Lemma 4, Theorem 2(1) is obtained.

3.4 Extension to interval graphs

A proper interval graph has a simple interval representation. Its interval representation is essentially unique up to isomorphism. On the other hand, an interval graph has exponentially many different interval representations. However, it has essentially unique tree representation called $\mathcal{MPQ}$-tree. The $\mathcal{MPQ}$-tree is stands for modified $\mathcal{PQ}$-tree, introduced by Korte and Möhring to solve the graph isomorphism problem for interval graphs[11]. In this paper, we omit the details of definitions of $\mathcal{MPQ}$-tree (see Appendix A). A parent-child relationship on the $\mathcal{MPQ}$-tree of an interval graph $G=(V,E)$ represents inclusion relationship. Thus, if a vertex v is an ancestor of another vertex u in the $\mathcal{MPQ}$-tree, I_v always contains I_u. Thus, for a connected interval graph $G=(V,E)$, the set of intervals corresponding to the vertices belonging to the root node of the $\mathcal{MPQ}$-tree contains all other intervals. Hence the algorithm for the free flooding game on an interval graph consists of two phases; (1) it connects the intervals and makes a monochrome backbone only using the intervals in the root node of the $\mathcal{MPQ}$-tree, and (2) join the remaining vertices by changing the color of the backbone. It is easy to see that the induced subgraph by the set of intervals belonging to the root node has a unique interval representation. (If it is a $\mathcal{P}$-node, the induced subgraph is a clique, and if it is a $\mathcal{Q}$-node, the representation is essentially unique up to isomorphism by its definition.) Therefore, the first phase is essentially the same as the algorithm for proper interval graphs. The algorithm makes the sequence of color sets, and regards it as a path, and proceeds. In that time, all other vertices (or small intervals) belonging to the other nodes are put in the set S of uncolored vertices. This implies Theorem 2(2).

We can also extend it to circular arc graphs:

Corollary 5 The free flooding game for one player on a circular arc graph can be solved in $O(8^k k^2 n^3)$ time.

4. Split Graphs

In 3), the fixed flooding game on a split graph is investigated. Using a similar idea in 5), we can extend the results for the fixed flooding game to the free flooding game.

Theorem 6 (1) The free flooding game is NP-complete even on a split graph. (2) The free flooding game on a split graph can be solved in $O(k!+n)$ time.

Proof.(Sketch) (1) In 3), the feedback vertex set problem is reduced to the fixed flooding game on a split graph $G=(V,E)$. The resulting graph G consists of a clique K and an independent set I. Each vertex in I has degree one except one universal vertex u incident to all vertices in K. It is easy to see that this universal vertex u can be one of the clique K. Now we add $|K|$ vertices to I and join them to u, and each of them is colored by $|K|$ colors that are same to the colors of vertices in K. Then, the resultant graph is still split graph. We consider the free flooding game on this new split graph. Then, using the similar argument in 5), this graph has a solution if and only if there is a sequence of operations that always colors the universal vertex u. Thus the feedback vertex set problem has a solution if and only if the free flooding game has a solution.

(2) We can observe that there is a solution of length at most $2k$ that first makes all vertices in K having the same color, and changes the color of the clique to join the vertices in I. We can also see that there is an optimum solution of this form. This means that we always change the color of a clique vertex. Since the vertices in K of the same color are always connected, the number of possibilities of each operation is at most $k'(k'-1)$, where k' is the current number of colors used in K. Thus, we can find an optimum solution in $O((k!)^2 + n)$ time. ∎

5. Concluding remarks

In this paper, we investigate the free flooding game on graphs that have interval representations. We show that this game is fixed parameter tractable with respect to the number of colors. We also show the similar results for split graphs. In 3), it is shown that the fixed flooding game on a co-comparability graph can be solved in polynomial time based on a dynamic programming technique. In this case, computing a shortest path on a co-comparability graph is better idea than using the dynamic programming. In the case, the idea can be extended to the free flooding game on a co-comparability graph, and we can obtain a polynomial time algorithm.

Acknowledgment

The authors thank Eric Theirry for sending 2).

References

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) A. Lagoutte, M. Naual, and E. Thierry. Flooding games on graphs. In *Latin-American Algorithms, Graphs and Optimization Symposium (LAGOS 2011)*, 2011.
3) 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.
4) Aurélie Lagoutte. 2-Free-Flood-It is polynomial. Technical report, arXiv:1008.3091v1, 2010.
5) H. Fukui, A. Nakanishi, R. Uehara, T. Uno, and Y. Uno. The Complexity of Free Flood Filling Game. In *WAAC 2011*, pages 51–56, 2011.
6) F.S. Roberts. Indifference graphs. In F.Harary, editor, *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, 1969.
7) K.P. Bogart and D.B. West. A short proof that 'proper=unit'. *Discrete Mathematics*, 201:21–23, 1999.
8) M.R. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* Freeman, 1979.
9) R. Uehara and Y. Uno. On Computing Longest Paths in Small Graph Classes. *International Journal of Foundations of Computer Science*, 18(5):911–930, 2007.
10) Toshiki Saitoh, Katsuhisa Yamanaka, Masashi Kiyomi, and Ryuhei Uehara. Random Generation and Enumeration of Proper Interval Graphs. *IEICE Transactions on Information and Systems*, E93-D(7):1816–1823, 2010.
11) N.Korte and R.H. Möhring. An Incremental Linear-Time Algorithm for Recognizing Interval Graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989.
12) K.S. Booth and G.S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity Using *PQ*-Tree Algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976.
13) G.S. Lueker and K.S. Booth. A Linear Time Algorithm for Deciding Interval Graph Isomorphism. *Journal of the ACM*, 26(2):183–195, 1979.
14) C.J. Colbourn and K.S. Booth. Linear Time Automorphism Algorithms for Trees, Interval Graphs, and Planar Graphs. *SIAM Journal on Computing*, 10(1):203–225, 1981.

Appendix

A.1 Definitions and Notations for MPQ-trees

The notion of $\mathcal{PQ}$-tree was introduced by Booth and Lueker[12], and that can be used to recognize interval graphs as follows. A *$\mathcal{PQ}$-tree* is a rooted tree T with two types of internal nodes: $\mathcal{P}$ and $\mathcal{Q}$, which will be represented by circles and rectangles, respectively. The leaves of T are labeled 1-1 with the maximal cliques of the interval graph G. The *frontier* of a $\mathcal{PQ}$-tree T is the permutation of the maximal cliques obtained by the ordering of the leaves of T from left to right. $\mathcal{PQ}$-tree T and T' are *equivalent*, if one can be obtained from the other by applying the following rules a finite number of times;

(1) arbitrarily permute the successor nodes of a $\mathcal{P}$-node, or

(2) reverse the order of the successor nodes of a $\mathcal{Q}$-node.

In 12), Booth and Lueker showed that a graph G is an interval graph if and only if there is a $\mathcal{PQ}$-tree T whose frontier represents a consecutive arrangement of the maximal cliques of G. They also developed a linear time algorithm that either constructs a

$\mathcal{PQ}$-tree for G, or states that G is not an interval graph. The algorithm by Booth and Lueker contains an update procedure that constructs, from a given $\mathcal{PQ}$-tree T for a system M, a $\mathcal{PQ}$-tree T' representing M plus one additional constraint set. This is done in a bottom-up way along the tree T by comparing parts of the tree with a fixed number of *patterns* that induce certain local *replacements* in T. If G is an interval graph, then all consecutive arrangements of the maximal cliques of G are obtained by taking equivalent $\mathcal{PQ}$-trees. The $\mathcal{PQ}$-tree with appropriate label defined by the maximal cliques is *canonical*; that is, given interval graphs G_1 and G_2 are isomorphic if and only if corresponding labeled $\mathcal{PQ}$-trees T_1 and T_2 are isomorphic. Since we can determine if two labeled $\mathcal{PQ}$-trees T_1 and T_2 are isomorphic, the isomorphism of interval graphs can be determined in linear time (see 12)–14) for further details).

The $\mathcal{MPQ}$-tree model, which stands for *modified $\mathcal{PQ}$-tree*, is developed by Korte and Möhring to simplify the $\mathcal{PQ}$-tree[11)].The $\mathcal{MPQ}$-tree T^* assigns sets of vertices (possibly empty) to the nodes of a $\mathcal{PQ}$-tree T representing an interval graph $G = (V, E)$. A $\mathcal{P}$-node is assigned only one set, while a $\mathcal{Q}$-node has a set for each of its sons (ordered from left to right according to the ordering of the sons).

For a $\mathcal{P}$-node P, this set consists of those vertices of G contained in all maximal cliques represented by the subtree of P in T, but in no other cliques.

For a $\mathcal{Q}$-node Q, the definition is more involved. Let $Q_1, \cdots, Q_m$ be the set of the sons (in consecutive order) of Q, and let T_i be the subtree of T with root Q_i (note that $m \geq 3$). We then assign a set S_i, called *section*, to Q for each Q_i. Section S_i contains all vertices that are contained in all maximal cliques of T_i and some other T_j, but not in any clique belonging to some other subtree of T that is not below Q. The key property of $\mathcal{MPQ}$-trees is summarized as follows:

Theorem 7 (11) [Theorem 2.1]) Let T be a $\mathcal{PQ}$-tree for an interval graph $G = (V, E)$ and let T^* be the associated $\mathcal{MPQ}$-tree. Then we have the following:
(a) T^* can be obtained from T in $O(|V| + |E|)$ time and represents G in $O(|V|)$ space.
(b) Each maximal clique of G corresponds to a path in T^* from the root to a leaf, where each vertex $v \in V$ is as close as possible to the root.
(c) In T^*, each vertex v appears in either one leaf, one $\mathcal{P}$-node, or consecutive sections $S_i, S_{i+1}, \cdots, S_{i+j}$ for some $\mathcal{Q}$-node with $j > 0$.

Property (b) is the essential property of $\mathcal{MPQ}$-trees. For example, the root of T^* contains all vertices belonging to all maximal cliques, and the leaves contain the simplicial vertices of G. In 11), they did not state Theorem 7(c) explicitly. Theorem 7(c) is immediately obtained from the fact that the maximal cliques containing a fixed vertex occur consecutively in T.

In order to solve the graph isomorphism problem, a $\mathcal{PQ}$-tree has additional information which is called *characteristic node* in 13), 14). This is the unique node which roots the subtree whose leaves are exactly the cliques to which the vertex belongs. As noted in 14) [p.212], the term characteristic node to mean the leaf, $\mathcal{P}$-node, or portion of a $\mathcal{Q}$-node which contains those cliques. It is easy to see that each vertex v in $\mathcal{MPQ}$-tree directly corresponds to the characteristic node in the $\mathcal{PQ}$-tree. In 11), they did not discuss the uniqueness of $\mathcal{MPQ}$-tree. However, their algorithm certainly constructs the unique $\mathcal{MPQ}$-tree for a given interval graph up to isomorphism.