Theoretical Computer Science 452 (2012) 75–87

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

An algorithmic analysis of the Honey-Bee game

Rudolf Fleischer [a,b,*], Gerhard J. Woeginger [c]

[a] SCS and IIPL, Fudan University, Shanghai, China

[b] GUtech, Muscat, Oman

[c] Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

ARTICLE INFO

Article history:
Received 6 December 2010
Received in revised form 19 January 2012
Accepted 14 May 2012
Communicated by T. Erlebach

Keywords:
Combinatorial game
Computational complexity
Graph problem

ABSTRACT

The Honey-Bee game is a two-player board game that is played on a connected hexagonal colored grid or (in a generalized setting) on a connected graph with colored nodes. In a single move, a player calls a color and thereby conquers all the nodes of that color that are adjacent to his own current territory. Both players want to conquer the majority of the nodes. We show that winning the game is PSPACE-hard in general, NP-hard on series-parallel graphs, but easy on outer-planar graphs.

In the solitaire version, the goal of the single player is to conquer the entire graph with the minimum number of moves. The solitaire version is NP-hard on trees and split graphs, but can be solved in polynomial time on co-comparability graphs.

1. Introduction

The Honey-Bee game is a popular two-player board game that shows up in many different variants and at many different places on the web (the game is best be played on a computer). For a playable version we refer the reader for instance to Axel Born's web-page [4]; see Fig. 1 for a screenshot. The playing field in Honey-Bee is a grid of hexagonal honey-comb cells that come in various colors; the coloring changes from game to game. The playing field may be arbitrarily shaped and may contain holes, but must always be connected. In the beginning of the game, each player controls a single cell in some corner of the playing field. Usually, the playing area is symmetric and the two players face each other from symmetrically opposing starting cells. In every move a player may call a color c, and thereby gains control over all connected regions of color c that have a common border with the area already under his control. The only restriction on c is that it cannot be one of the two colors used by the two players in their last move before the current move, respectively. A player wins when he controls the majority of all cells. On Born's web-page [4] one can play against a computer, choosing from four different layouts for the playing field. The computer uses a simple greedy strategy: "Always call the color c that maximizes the immediate gain". This strategy is short-sighted and not very strong, and an alert human player usually beats the computer after a few practice matches.

In this paper we perform a complexity study of the Honey-Bee game when played by two players on some arbitrary connected graph instead of the hex-grid of the original game. We will show in Section 4 that Honey-Bee-2-Players is NP-hard on series-parallel graphs but PSPACE-complete in general. On outer-planar graphs, however, it is easy to compute a winning strategy.

In the *solitaire* (single-player) version of Honey-Bee the goal for the single player is to conquer the entire playing field as quickly as possible. Intuitively, a good strategy for the solitaire game will be close to a strong heuristic for the two-player game. For the solitaire version, our results draw a sharp separation line between easy and difficult cases. In particular,

* Corresponding author. Tel.: +86 21 65654634; fax: +86 21 65654253.
E-mail addresses: rudolf@fudan.edu.cn (R. Fleischer), gwoegi@win.tue.nl (G.J. Woeginger).

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2012.05.032

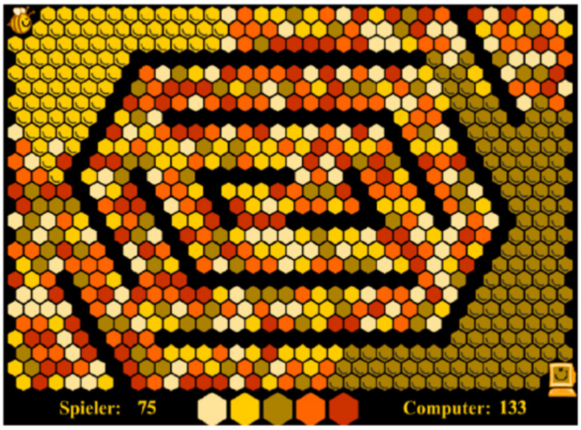

Fig. 1. Born's "Biene". The human player (starting from the top-left corner) is on the edge of losing against the computer (starting from the bottom-right corner).

we show in Section 3 that Honey-Bee-Solitaire is NP-hard for split graphs and for trees, but polynomial-time solvable on co-comparability graphs (which include interval graphs and permutation graphs). Thus, the complexity of the game is well-characterized for the class and subclasses of perfect graphs; see Fig. 2 for a summary of our results. Note that trees are planar graphs; hence our results also imply hardness for planar graphs.

Related Work. The popular game Flood It is a special case of Honey-Bee-Solitaire played on a square grid. Verbin showed NP-hardness of flooding a star with four colors and a grid with six colors [19]. Later, Arthur et al. improved this result by showing NP-hardness for flooding a grid with three colors from an arbitrary start cell [2]. They also studied approximability of optimal flooding. Similar to our hardness proofs, their proofs are also by reduction from a variant of SCS (however, we are using a different variant).

There are many other board games with a similar flavor as Honey-Bee-Solitaire. We just want to mention Clickomania [3] and Tetris [5] which have also been proven to be difficult. Actually, many two-player board games are PSPACE-hard, such as for example Go [11,20], Gobang [14], Hex [15], Othello [10], two-player Tetris [16], and Sabotage [12], while some games are even Exptime-hard, such as for example Checkers [17], Chess [7], and Shogi [1]. For a survey of board games and an extensive list of references we refer to the survey paper by Demaine and Hearn [6].

2. Definitions

We model Honey-Bee in the following graph-theoretic setting. The playing field is a connected, simple, loopless, undirected graph $G = (V, E)$. There is a set C of k colors, and every node $v \in V$ is colored by some color $col(v) \in C$; we stress that this coloring does not need to be *proper*, that is, there may be edges $[u, v] \in E$ with $col(u) = col(v)$. For a color $c \in C$, the subset $V_c \subseteq V$ contains the nodes of color c. For a node $v \in V$ and a color $c \in C$, we define the *color-c-neighborhood* $\Gamma(v, c)$ as the set of nodes in V_c either adjacent to v or connected to v by a path of nodes of color c. Similarly, we denote by $\Gamma(W, c) = \bigcup_{w\in W} \Gamma(w, c)$ the color-c-neighborhood of a subset $W \subseteq V$. For a subset $W \subseteq V$ and a sequence $\gamma = \langle \gamma_1, \dots, \gamma_b \rangle$ of colors in C, we define a corresponding sequence of node sets $W_1 = W$ and $W_{i+1} = W_i \cup \Gamma(W_i, \gamma_i)$, for $1 \le i \le b$. We say that sequence γ started on W *conquers* the final node set W_{b+1} in b moves, and we denote this situation by $W \to_\gamma W_{b+1}$. The nodes in $V - W_{b+1}$ are called *free* nodes.

In the *solitaire* version of Honey-Bee, the goal is to conquer the entire playing field with the smallest possible number of moves. Note that Honey-Bee-Solitaire is trivial in the case of only two colors. But as we will see in Section 3, the case of four colors can already be difficult.

Problem Honey-Bee-Solitaire

Input: A graph $G = (V, E)$; a set C of k colors and a coloring $col : V \to C$; a start node $v_0 \in V$; and a bound b.

Question: Does there exist a color sequence $\gamma = \langle \gamma_1, \dots, \gamma_b \rangle$ of length b such that $\{v_0\} \to_\gamma V$?

In the *two-player* version of Honey-Bee, the two players A and B start from two distinct nodes a_0 and b_0 and then extend their regions step by step by alternately calling colors. Player A makes the first move. One round of the game consists of a move of A followed by a move of B. Consider a round, where at the beginning the two players control node sets W_A and W_B, respectively. If player A calls color c, then he extends his region W_A to $W_A' = W_A \cup (\Gamma(W_A, c) - W_B)$. If afterwards player B calls color d, then he extends his region W_B to $W_B' = W_B \cup (\Gamma(W_B, c) - W_A')$. Note that once a player controls a node, he can never lose it again.

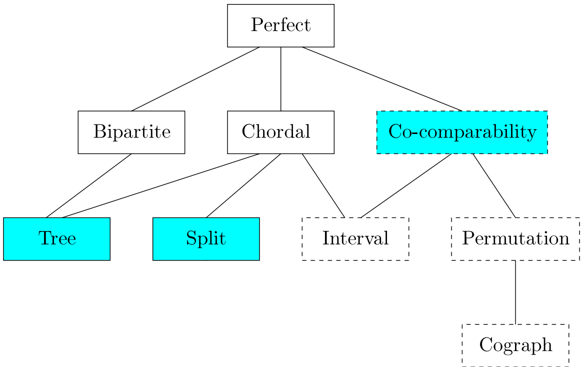

Fig. 2. Summary of the complexity results for Honey-Bee-Solitaire. NP-complete problems have a solid frame, polynomial-time solvable problems have a dashed frame. The results for the graph classes in the three colored boxes imply all other results.

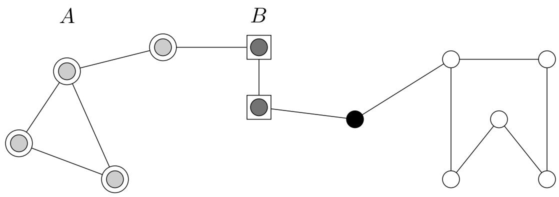

Fig. 3. Player A (circled nodes) is leading with four captured nodes over player B (squared nodes) with only two captured nodes. Player B would next like to play black to capture all the white nodes in the next move. Without rule R2, player A could prevent this by repeatedly playing black.

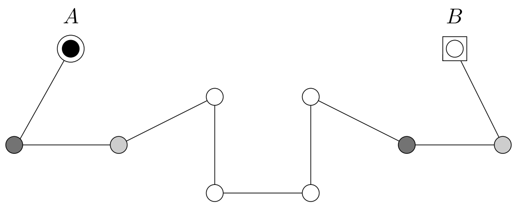

Fig. 4. Player A who controls the black node at the left end of the path loses if he calls dark-gray and hence prefers to call light-gray (white and black are not allowed by R1 and R2, respectively). Player B who controls the white node at the other end of the path loses if he calls light-gray and hence prefers to call dark-gray (black and white are not allowed by R1 and R2, respectively). However, Rule R3 forces the players to move into the unoccupied territory, thus the first player to move loses.

The game terminates as soon as one player controls more than half of all nodes. This player wins the game. To avoid draws, we require that the number of nodes is odd. There are three important rules that constrain the colors that a player is allowed to call.

R1. A player must never call the color that has just been called by the other player.
R2. A player must never call the color that he has called in his previous move.
R3. A player must always call a color that strictly enlarges his territory, unless rules R1 and R2 prevent him from doing so.

What is the motivation for these three rules? Rule R1 is a technical condition that arises from the graphical implementation [4] of the game: whenever a player calls a color c, his current territory is entirely recolored to color c. This makes it visually easier to recognize the territories controlled by both players. Furthermore, the regions controlled by the two players usually become adjacent after some time; rule R1 prevents that one player may take over the region of the other player. Rule R2 prevents the players from permanently blocking some color for the opponent. Fig. 3 shows a situation where rule R2 actually prevents the game from stalling. Rule R3 is quite delicate, and is justified by situations as depicted in Fig. 4. Rule R3 guarantees that every game must terminate with either a win for player A or a win for player B. Note that rule R2 is redundant except in the case when a player has no move to gain territory (see Fig. 3).

Problem Honey-Bee-**2**-Players

Input: A graph $G = (V, E)$ with an odd number of nodes; a set C of colors and a coloring $col : V \to C$; two start nodes $a_0, b_0 \in V$.

Question: Can player A enforce a win when the game is played according to the above rules?

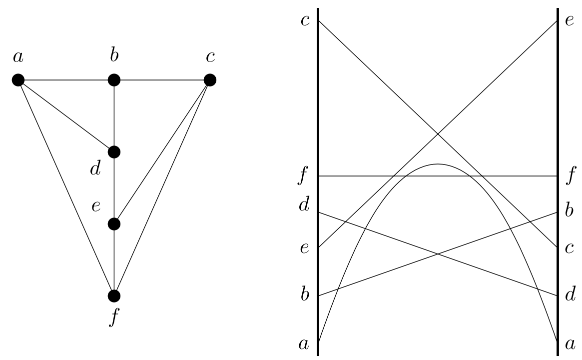

Fig. 5. A co-comparability graph corresponding to the order $a < c, a < e, b < e, b < f, d < c, d < f$ and its curve intersection representation. Without a, the graph would be a permutation graph with a straight line representation.

Note that Honey-Bee-2-Players is trivial in the case of only three colors: the players do not have the slightest freedom in choosing their next color, and always must call the unique color allowed by rules R1 and R2. However we will see in Section 4 that the case of four colors can already be difficult.

Finally we observe that calling a color c always conquers all connected components induced by V_c that are adjacent to the current territory. Hence an equivalent definition of the game could use a graph with node weights (that specify the size of the corresponding connected component) and a proper coloring of the nodes. Any instance under the original definition can be transformed into an equivalent instance under the new definition by contracting each connected component of V_c, for some c, into a single node of weight $|V_c|$. However, we are interested in restrictions of the game to particular graph classes, some of which are not closed under edge contractions (such as for instance the hex-grid graph of the original Honey-Bee game).

3. The solitaire game

In this section we study the complexity of finding optimally short color sequences for Honey-Bee-Solitaire. We will show that this is easy for co-comparability graphs (arbitrary number of colors), while it is NP-hard for split graphs (arbitrary number of colors), trees (four colors), and series-parallel graphs (three colors). Since the family of co-comparability graphs contains interval graphs, permutation graphs, and co-graphs as sub-families, our positive result for co-comparability graphs implies all other positive results in Fig. 2.

A first straightforward observation is that Honey-Bee-Solitaire lies in NP: any connected graph $G = (V, E)$ can be conquered in at most $|V|$ moves, and hence such a sequence of polynomially many moves can serve as an NP-certificate.

3.1. *The solitaire game on co-comparability graphs*

A *co-comparability graph* $G = (V, E)$ is an undirected graph whose nodes V correspond to the elements of some partial order $<$ and whose edges E connect any two elements that are incomparable in that partial order, i.e., $[u, v] \in E$ if neither $u < v$ nor $v < u$ holds. For simplicity, we identify the nodes with the elements of the partial order. Golumbic et al. [9] showed that co-comparability graphs are exactly the intersection graphs of continuous real-valued functions over some interval I. If two function curves intersect, the corresponding elements are incomparable in the partial order; otherwise, the curve that lies completely above the other one corresponds to the larger element in the partial order (see Fig. 5 for an example). If the curves are straight lines (i.e., linear functions), we have as a special case a permutation graph. More general than co-comparability graphs are *string graphs* [18] which are the intersection graphs of Jordan curves in the plane.

Lemma 3.1. *The class of co-comparability graphs is closed under edge contractions.*

Proof. Consider an edge (u, v) in a co-comparability graph G. Contracting the edge into a single node w adjacent to all nodes that were adjacent to u and v changes the corresponding order relation as follows. For all nodes $x \neq u, v, w < x$ if $u < x$ and $v < x$, $w > x$ if $u > x$ and $v > x$, and w is incomparable to x otherwise. Since u and v must have been incomparable in G, the cases $u < x < v$ and $u > x > v$ cannot happen, so the last of the three cases above can only occur if at least one of u and v was incomparable to x, i.e., connected to x by an edge in G which is preserved in the contracted graph. □

Therefore, we may w.l.o.g. restrict our analysis of Honey-Bee-Solitaire to co-comparability graphs with a proper node coloring where adjacent nodes have distinct colors (in the solitaire game the weight of a node after an edge contraction is not important). In this case, every color class is totally ordered because incomparable node pairs of the same color have been contracted.

Consider an instance of Honey-Bee-Solitaire with a minimal start node v_0 (in the partial order on V); a maximal start node could be handled similarly.

Lemma 3.2. *Conquering a node will simultaneously conquer all smaller nodes of the same color.*

Proof. If we start at a minimal node v_0 and at some time later conquer a node v of color c, then we must have conquered a path p from v_0 to v. In the function graph representation of the graph this means we have found a sequence of function graphs connecting v_0 to v, but this must intersect all nodes smaller than v, i.e., these nodes must be adjacent in the graph to some node on p. Thus, conquering v will at the same time conquer all smaller nodes of color c (if they had not been conquered before). □

For any color c, let $Max(c)$ denote the largest node of color c. Let M be the set of all these nodes, for all colors c, and let M_{max} be the set of maximal elements of the partial order. Note that $|M| = k$, the number of colors (all color classes are totally ordered and have therefore exactly one maximum element each), and $M_{max} \subseteq M$.

Lemma 3.3. *The nodes in M_{max} form a clique in G.*

Proof. Let $u, v \in M_{max}$. If u and v are not connected in G, then either $u < v$ or $v < u$, contradicting their maximality. Thus, all nodes in M_{max} form a clique. □

By Lemma 3.2, it suffices to find a shortest color sequence conquering the set M. We can do this by a single source shortest path computation. We assign every node $Max(c)$ weight 0, and all other nodes weight 1. Then we compute shortest paths (with respect to the node-weights) from v_0 to all the nodes in M_{max}. Let OPT denote the cost of a cheapest path to reach at least one node in M_{max}.

For a color sequence $\gamma = \langle \gamma_1, \ldots, \gamma_b \rangle$, we define the *length* of γ as $|\gamma| = b$. We also define the *essential length* $ess(\gamma)$ of γ as $|\gamma|$ minus the number of steps where γ conquers a node in M. Obviously, $|\gamma| = ess(\gamma) + k$ for any minimal color sequence γ conquering the entire graph (γ needs one step for each node in M). Note that OPT is the smallest essential cost of any color sequence conquering at least one node in M_{max}.

Lemma 3.4. *The optimal solution for* Honey-Bee-Solitaire *has cost $OPT + k$.*

Proof. Let γ be a shortest color sequence conquering the entire graph starting at v_0. After conquering the first node in M_{max}, γ only needs to conquer all remaining nodes in M to conquer the entire graph. This can be done with one step for each color by Lemmas 3.2 and 3.3. Thus, $|\gamma| = ess(\gamma) + k \geq OPT + k$, with equality if γ is an optimal color sequence. □

Theorem 3.5. Honey-Bee-Solitaire *starting at an extremal node v_0 can be solved in polynomial time on co-comparability graphs.*

Proof. Given the co-comparability graph G, we can compute the underlying partial order in polynomial time [9]. Assigning the weights and solving one single source shortest path problem starting at v_0 also takes polynomial time. □

We can also formulate this algorithm as a dynamic program. For any node v, let $D(v)$ denote the essential length of the shortest color sequence γ that can conquer v when starting at v_0. For any color c, let $min_v(c)$ denote the smallest node of color c connected to v, if such nodes exist. Then we can compute $D(v)$ recursively as follows:

$$D(v_0) = 0$$

and

$$D(v) = \min_c (D(min_v(c)) + \delta_v) \ ,$$

where $D(min_v(c)) = \infty$ if $min_v(c)$ is undefined, and $\delta_v = 0\,(1)$ if v is (not) in M.

This dynamic program simulates the shortest path computation of our first algorithm and we have $OPT = \min_v D(v)$, where we minimize over all maximal nodes $v \in M_{max}$. We now extend the dynamic program to the case that v_0 is not an extremal element. The problem is that we now must extend our territory in two directions. If we choose a move that makes good progress upwards it may make little progress downwards, or vice versa. In particular, the optimal strategy cannot be decomposed into two independent optimal strategies, one conquering upwards and one conquering downwards. Analogously to the algorithm above, for any color c and node v define $Min(c)$ as the smallest node of color c, and $max_v(c)$ as the largest node of color c connected to v.

Unfortunately, we must now redefine the essential length of a color sequence γ. In our original definition, we did not count coloring steps that conquered maximal elements of some color class. This is intuitively justified by the fact that these steps must be done by any color sequence conquering the entire graph at some time, therefore it is advantageous to do them as early as possible (which is guaranteed by giving these moves cost 0). But now we must also consider the minimal nodes of each color class. An optimal sequence conquering the entire graph will at some time have conquered a minimal node and a maximal node. Afterwards, it will only call extremal nodes for some color class. If both extremal nodes of a color class are still free, we only need *one* move to conquer both simultaneously. If one of them had been captured earlier, we still need to conquer the other one. This indicates that we should charge 1 for the first extremal node conquered while the second one should be charged 0, as before. If both nodes are conquered in the same move, we should also charge 0. Therefore, we now define the *essential length* $ess(\gamma)$ of γ as $|\gamma|$ minus the number of steps where γ conquers the second extremal node of some color class.

For a node v below v_0 or incomparable to v_0 and a node w above v_0 or incomparable to v_0 let $D(v, w)$ denote the essential length of the shortest color sequence γ that can conquer v and w when starting at v_0. Note that we do not need to keep track of which first extremal nodes of a color class have been conquered because we can deduce this from the two nodes v and w currently under consideration. In particular, we can compute $D(v, w)$ recursively as follows:

$$D(v_0, v_0) = 0$$

and

$$D(v, w) = \min_c (D(v, min_w(c)) + \delta_w(v), D(max_v(c), w) + \delta_v(w)) ,$$

where $\delta_v(w) = 0$ if and only if w is an extremal node of some color class c and the other extremal node of color class c is either between v and w, or incomparable to either v or w, or both (it was either conquered earlier, or it will be conquered in this step); otherwise, $\delta_v(w) = 1$. Obviously, $|\gamma| = ess(\gamma) + k$ for any minimal color sequence γ conquering the entire graph.

Lemma 3.6. *The optimal solution for* Honey-Bee-Solitaire *has cost* $\min_{v,w}(D(v, w) + k)$*, where we minimize over all minimal nodes v and all maximal nodes w.*

Proof. Let γ be a shortest color sequence conquering the entire graph starting at v_0. Let v be the first minimal node conquered by γ and w the first maximal node. After conquering v and w, γ only needs to conquer all remaining extremal nodes of each color class to conquer the entire graph. This can be done with one step for each color by Lemmas 3.2 and 3.3. Thus, $|\gamma| \geq D(v, w) + k$, with equality if γ is an optimal color sequence. □

Theorem 3.7. Honey-Bee-Solitaire *can be solved in polynomial time on co-comparability graphs.* □

3.2. The solitaire game on split graphs

A *split graph* is a graph whose node set can be partitioned into an induced clique and into an induced independent set. We will show that Honey-Bee-Solitaire is NP-hard on split graphs. Our reduction is from the NP-hard `Feedback Vertex Set` (FVS) problem in directed graphs; see for instance Garey and Johnson [8].

> **Problem** FVS
>
> **Input:** A directed graph (X, A); a bound $t < |X|$.
>
> **Question:** Does there exist a subset $X' \subseteq X$ with $|X'| = t$ such that the directed graph induced by $X - X'$ is acyclic?

Theorem 3.8. Honey-Bee-Solitaire *on split graphs is NP-hard.*

Proof. Consider an instance (X, A, t) of FVS. To construct an instance (V, E, b) of Honey-Bee-Solitaire, we first build a clique on the nodes in X and a new node v_0 (which will be the start node of Honey-Bee-Solitaire). Each node $x \in X \cup \{v_0\}$ has a different color c_x. Next, we build the independent set. For every arc $(x, y) \in A$, we introduce a corresponding node $v(x, y)$ of color c_y which has degree one and which is only connected to node x in the clique. Finally, we set $b = |X| + t$. We claim that the constructed instance of Honey-Bee-Solitaire has answer YES, if and only if the considered instance of FVS has answer YES.

Assume that the FVS instance has answer YES. Let X' with $|X'| = t$ be a feedback set whose removal makes (X, A) acyclic. Let π be a topological order of the nodes in $X - X'$, and let τ be an arbitrary ordering of the nodes in X'. Consider the color sequence γ of length $|X| + t$ that starts with τ, followed by π, and followed by τ again. We claim that $\{v_0\} \rightarrow_\gamma V$. Indeed, γ first runs through τ and π and thereby conquers all clique nodes. Every independent set node $v(x, y)$ with $y \in X'$ is conquered during the first or second transversal of τ. Every independent set node $v(x, y)$ with $y \in X - X'$ is conquered during the transversal of π, since π first conquers x with color c_x, and afterwards $v(x, y)$ with color c_y.

Next assume that the instance of Honey-Bee-Solitaire has answer YES. Let γ be a color sequence of length at most $|X|+t$ conquering V. Define X' as the set of nodes x such that color c_x occurs at least twice in γ. As every color c_x with $x \in X$ must appear at least once in γ, we conclude $|X'| \leq t$. Consider an arc $(x, y) \in A$ with $x, y \in X - X'$. Since γ contains color c_y only once, it must conquer node $v(x, y)$ of color c_y after node x of color c_x. Hence, γ induces a topological order of $X - X'$. □

The construction in the proof above uses linearly many colors. What about the case of few colors? On split graphs, Honey-Bee-Solitaire can always be solved by traversing the color set C twice; the first traversal conquers all clique nodes, and the second traversal conquers all remaining free independent set nodes. Thus, every split graph can be completely conquered in at most $2|C|$ steps. If there are only few colors, we can simply check all color sequences of length at most $2|C|$.

Theorem 3.9. *If the number of colors is bounded by a fixed constant,* Honey-Bee-Solitaire *on split graphs is polynomial-time solvable. In other words,* Honey-Bee-Solitaire *is fixed parameter tractable when parameterized by the number of colors.* □

3.3. The solitaire game on trees

In this section we will show that Honey-Bee-Solitaire is NP-hard on trees, even if there are only four colors. We reduce Honey-Bee-Solitaire from a variant of the `Shortest Common Supersequence` (SCS) problem which is known to be NP-complete (see Middendorf [13]). Note that subsequences do not need to be contiguous.

> **Problem** SCS
>
> **Input:** A positive integer t; finite sequences $\sigma_1, \ldots, \sigma_s$ with elements from $\{0, 1\}$ with the following properties: (i) all sequences have the same length; (ii) every sequence contains exactly two 1s, and these two 1s are separated by at least one 0; and (iii) every sequence ends with a 0.
>
> **Question:** Does there exist a sequence σ of length t that contains $\sigma_1, \ldots, \sigma_s$ as subsequences?

Middendorf's hardness result also implies the hardness of the following variant of SCS:

> **Problem** `Modified SCS` (MSCS)
>
> **Input:** A positive integer t; finite sequences $\sigma, \ldots, \sigma_s$ with elements from $\{0, 1, 2, 3\}$ with the following property: in every sequence any two consecutive elements are distinct, and no sequence starts with 2 or 3.
>
> **Question:** Does there exist a sequence σ of length t that contains $\sigma_1, \ldots, \sigma_s$ as subsequences?

Theorem 3.10. MSCS *is NP-complete.*

Proof. Here is a reduction from SCS to MSCS. Consider an arbitrary sequence τ with elements from $\{0, 1\}$. We define $f(\tau)$ as the sequence obtained by replacing every occurrence of the element $0 \in \tau$ by two consecutive elements 0 and 2, and by replacing every occurrence of the element $1 \in \tau$ by two consecutive elements 1 and 3. Now consider an instance $(\sigma_1, \ldots, \sigma_s, t)$ of SCS. Construct an instance $(\sigma_1', \ldots, \sigma_s', t')$ of MSCS by setting $\sigma_i' = f(\sigma_i)$ for $1 \leq i \leq s$ and by defining $t' = 2t$. Then for any sequence σ with elements from $\{0, 1\}$, σ is a common supersequence of $\sigma_1, \ldots, \sigma_s$ if and only if $f(\sigma)$ is a common supersequence of $\sigma_1', \ldots, \sigma_s'$. This implies the NP-hardness of MSCS. $\square$

Theorem 3.11. Honey-Bee-Solitaire *is NP-hard on trees, even in the case of only four colors.*

Proof. We reduce MSCS to Honey-Bee-Solitaire on trees. Consider an instance $(\sigma_1, \ldots, \sigma_s, t)$ of MSCS. As color set $C = \{0, 1, 2, 3\}$ we use the four letters in the strings of the MSCS instance. We first construct a root v_0 of color 2, and then for each sequence σ_i attach a path of length $|\sigma_i|$ to v_0; the j-th node on this path is colored by color k if the j-th letter in σ_i is the letter k. Finally, we set $b = t$. It is straightforward to see that the constructed instance of Honey-Bee-Solitaire has answer YES if and only if the instance of MSCS has answer YES. $\square$

Note that the proof of the previous theorem actually shows NP-hardness for *stars* with four colors.

3.4. The solitaire game on series-parallel graphs

A graph is *series-parallel* if it does not contain K_4 as a minor. Equivalently, a series-parallel graph can be constructed from a single edge by repeatedly doubling edges, or removing edges, or replacing edges by a path of two edges with a new node in the middle of the path.

Theorem 3.12. Honey-Bee-Solitaire *is NP-hard on series-parallel graphs, even in the case of only three colors.*

Proof. The proof is by reduction from the supersequence problem SCS with binary sequences; see Section 3.3. Consider an instance $(\sigma_1, \ldots, \sigma_s, t)$ of SCS, and let n denote the common length of all sequences σ_i. Without loss of generality we assume our color set is $C = \{0, 1, 2\}$. We create a start node v_0 of color 2, and attach to it the following gadgets.

- For each sequence σ_i with $1 \leq i \leq s$, there is a path P_i that consists of $2n$ nodes and that is attached to v_0. The colors of the n nodes with odd numbers encode the sequence σ_i, while the n dummy nodes with even numbers along the path all receive color 2.
- There is also a control-gadget that consists of $3n$ nodes a_j, b_j, v_j with $1 \leq j \leq n$ and a special node w. For $1 \leq j \leq n$, node a_j has color 0 and is connected to v_{j-1} and v_j (where node v_0 is the starting node). For $1 \leq j \leq n$, node b_j has color 1 and is connected to v_{j-1} and v_j. All nodes v_j have color 2. Node w is only adjacent to node v_n and has color 1.

Finally, we set $b = 2t + 1$. We claim that the constructed instance of Honey-Bee-Solitaire has answer YES if and only if the instance of SCS has answer YES.

Indeed, consider a super-sequence σ of length t for the SCS instance. Since every sequence σ_i ends with 0, we may assume without loss of generality that σ also ends with 0. We construct a color sequence by inserting the color 2 between every two consecutive elements of σ, and we close the sequence by adding color 1. This color sequence conquers the entire graph, and the final color 1 is needed to conquer node w. Vice versa, if there exists a color sequence of length $b = 2t + 1$ that conquers the entire graph, then one easily constructs a super-sequence of length t for the SCS instance. $\square$

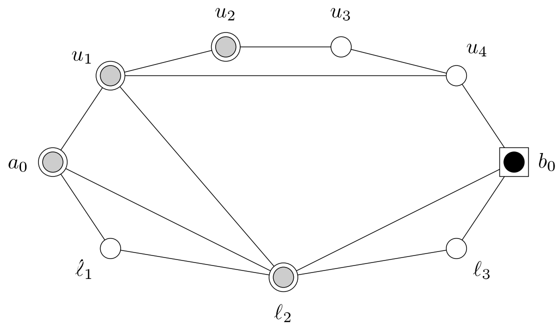

Fig. 6. An outer-planar graph with start nodes a_0 and b_0. Player A (circled nodes) has conquered the light-gray colored nodes, i.e., $U = 2$ and $L = 2$. Eventually, A will also conquer ℓ_1, since player B cannot reach it.

4. The two-player game

In this section we study the complexity of the two-player game. While on outer-planar graphs the players can compute their winning strategies in polynomial time for an arbitrary number of colors, this problem is NP-hard for series-parallel graphs with four colors, and PSPACE-complete on arbitrary graphs with four colors.

We stress that our negative results hold for four colors, which is the strongest possible type of result (recall that instances with three colors are trivial to solve). We stress furthermore that the borderline between easy (outer-planar) and hard (series-parallel) is very sharp and precise, if one consider the forbidden minors: series-parallel graphs do not contain K_4 as a minor, and outer-planar graphs contain neither K_4 nor $K_{2,3}$ as a minor.

4.1. The two-player game on outer-planar graphs

A graph is *outer-planar* if it contains neither K_4 nor $K_{2,3}$ as a minor. Outer-planar graphs have a planar embedding in which every node lies on the boundary of the so-called *outer face* (which is the unique infinite face in the embedding). For example, every tree is an outer-planar graph. Our approach crucially hinges on the ordering of the vertices along the outer face, and both players essentially follow this ordering while extending their regions. This also is the main difference to series-parallel graphs, which do not have such an ordering and on which the game is hard.

Consider an outer-planar graph $G = (V, E)$ as an instance of Honey-Bee-2-Players with starting nodes a_0 and b_0 in V, respectively. The starting nodes divide the nodes on the boundary of the outer face F into an upper chain $u_1, \ldots, u_s$ and a lower chain $\ell_1, \ldots, \ell_t$, where u_1 and ℓ_1 are the two neighbors of a_0 on F, while u_s and ℓ_t are the two neighbors of b_0 on F. We stress that these two chains are not necessarily disjoint (for instance, articulation nodes will occur in both chains). In particular, it might happen that $u_1 = \ell_1$ or $u_s = \ell_t$.

Now consider an arbitrary situation in the middle of the game. Let U (respectively L) denote the largest index k such that player A has conquered node u_k (respectively node ℓ_k). See Fig. 6 to illustrate these definitions and the following lemma.

Lemma 4.1. *Let X denote the set of nodes among $u_1, \ldots, u_U$ and $\ell_1, \ldots, \ell_L$ that currently do neither belong to A nor to B. Then no node in X can have a neighbor among $u_{U+1}, \ldots, u_s, b_0, \ell_t, \ldots, \ell_{L+1}$.*

Proof. The existence of such a node in X would lead to a K_4-minor in the outer-planar graph. This is true because X cannot articulation nodes (there cannot be a shortcut to bypass such nodes). Therefore any node in X is cut off by an edge where player A jumped ahead, see Fig. 6 for an example. □

Theorem 4.2. Honey-Bee-*2*-Players *on outer-planar graphs is polynomial-time solvable.*

Proof. The two indices U and L encode all necessary information on the future behavior of player A. Eventually, he will own all nodes $u_1, \ldots, u_U$ and $\ell_1, \ldots, \ell_L$, and the possible future expansions of his area beyond u_U and ℓ_L only depend on U and L. Symmetric observations hold true for player B.

As every game situation can be concisely described by just four indices, there is only a polynomial number $O(|V|^4)$ of relevant game situations. The rest is routine work in combinatorial game theory: we first determine the winner for every end-situation, and then by working backwards in time we can determine the winners for the remaining game situations. □

4.2. The two-player game on series-parallel graphs

Recall from Section 3.4 that a graph is *series-parallel* if it does not contain K_4 as a minor. Note that series-parallel graphs are planar, which yields that our hardness result proved in this section also holds for the class of planar graphs. We stress that we do not know whether the two-player game on series-parallel graphs is contained in the class NP (and we actually see no reason why it should lie in NP); therefore the following theorem only states NP-hardness.

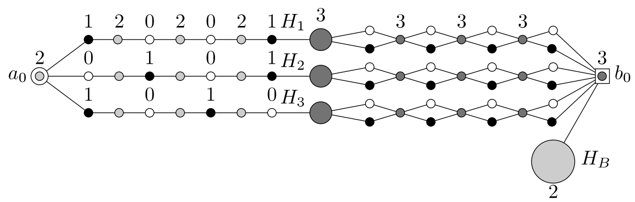

Fig. 7. The graph constructed in the proof of Theorem 4.3 for the sequences $\sigma_1 = 1001$, $\sigma_2 = 0101$, $\sigma_3 = 1010$, and $t = 4$. The optimal SCS solution is 10101. Thus, B can win this game.

Theorem 4.3. *For four (or more) colors, problem* Honey-Bee-2-Players *on series-parallel graphs is NP-hard.*

Proof. We use the color set $C = \{0, 1, 2, 3\}$. A central feature of our construction is that player B will have no real decision power, but will only follow the moves of player A: if player A starts a round by calling color 0 or 1, then player B must follow by calling the other color in $\{0, 1\}$ (or waste his move). And if player A starts a round by calling color 2 or 3, then player B must call the other color in $\{2, 3\}$ (or waste his move). In the even rounds the players will call the colors in $\{0, 1\}$ and in the odd rounds they will call the colors in $\{2, 3\}$. Both players are competing for a set of honey pots in the middle of the battlefield, and need to get there as quickly as possible. If a player deviates from the even–odd pattern indicated above, he might perhaps waste his move and delay the game by one round (in which neither player comes closer to the honey pots), but this remains without further impact on the outcome of the game.

The proof is by reduction from the supersequence problem SCS with binary sequences; see Section 3.3. Consider an instance $(\sigma_1, \ldots, \sigma_s, t)$ of SCS, and let n denote the common length of all sequences σ_i. We first construct two start nodes a_0 and b_0 of colors 2 and 3, respectively. For each sequence σ_i with $1 \le i \le s$ we do the following:

- We construct a path P_i that consists of $2n - 1$ nodes and that is attached to a_0: the n nodes with odd numbers mimic sequence σ_i, while the $n - 1$ nodes with even numbers along the path all receive color 2. The first node of P_i is adjacent to a_0, and its last node is connected to a so-called honey pot H_i.
- The *honey pot* H_i is a long path consisting of $4st$ nodes of color 3. Intuitively, we may think of a honey pot as a single node of large weight, because conquering one of the nodes will simultaneously conquer the entire path.
- Every honey pot H_i can also be reached from b_0 by another path Q_i that consists of $2t - 1$ nodes. Nodes with odd numbers get color 0, and nodes with even numbers get color 3. The first node of Q_i is adjacent to b_0, and its last node is connected to H_i. Furthermore, we create for each odd-numbered node (of color 0) a new twin node of color 1 that has the same two neighbors as the color 0 node. Note that for every path Q_i there are t twin pairs.

Finally we create a private honey pot H_B for player B that is connected to node b_0 and that consists of $4s(s-1)t + (2n-1)s$ nodes of color 2. This completes the construction; see Fig. 7 for an example.

Assume that the SCS instance has answer YES. During his first $2t - 1$ steps, player B can only conquer the paths Q_i and his private honey pot H_B. At the same time, player A can conquer all paths P_i by calling color 2 in his even moves and by following a shortest 0–1 supersequence in his odd moves. Then, in round $2t$ player A will simultaneously conquer all the honey pots H_i with $1 \le i \le s$. This gives A a territory of at least $1 + (2n - 1)s + 4s^2t$ nodes, and B a smaller territory of at most $1 + (3t - 1)s + 4s(s - 1)t + (2n - 1)s$ nodes. Hence A can enforce a win.

Next assume that player A has a winning strategy. Player B can always conquer his starting node b_0 and his private honey pot H_B. If B also manages to conquer one of the pots H_i, then he gets a territory of at least $1 + 4s(s - 1)t + (2n - 1)s + 4st$ nodes and surely wins the game. Hence player A can only win if he conquers all s honey pots H_i. To reach them before player B does, player A must conquer them within his first $2t$ moves. In every odd round, player A will call a color 0 or 1 and player B will call the other color in $\{0, 1\}$. Hence, in the even rounds, colors 0 and 1 are forbidden for player A, and the only reasonable move is to call color 2. Note that the slightest deviation of these forced moves would give player B a deadly advantage. In order to win, the odd moves of player A must induce a supersequence of length at most t for all sequences σ_i. Therefore, the SCS instance has answer YES. □

4.3. The two-player game on arbitrary graphs

In this section we will show that Honey-Bee-2-Players is PSPACE-complete on arbitrary graphs. Our reduction is from the PSPACE-complete `Quantified Boolean Formula` (QBF) problem; see for instance Garey and Johnson [8].

Problem QBF

Input: A quantified Boolean formula with $2n$ variables in conjunctive normal form: $\exists x_1 \forall x_2 \cdots \exists x_{2n-1} \forall x_{2n} \bigwedge_j C_j$, where the C_j are clauses of the form $\bigvee_k l_{jk}$, where the l_{jk} are literals.

Question: Is the formula true?

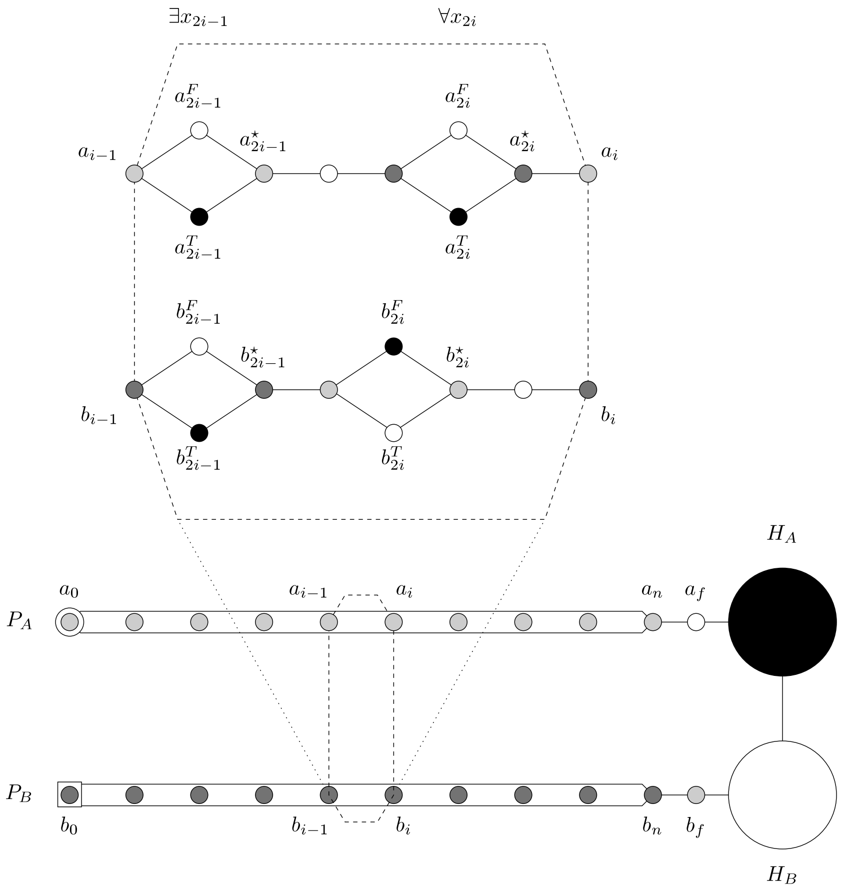

Fig. 8. The variable gadget in the proof of Theorem 4.4.

Theorem 4.4. *For four (or more) colors, problem* Honey-Bee-2-Players *on arbitrary graphs is PSPACE-complete.*

Proof. We reduce from QBF. Let $F = \exists x_1 \forall x_2 \cdots \exists x_{2n-1} \forall x_{2n} \bigwedge_j C_j$ be an instance of QBF. We construct a bee graph $G_F = (V, E)$ with four colors (white, light-gray, dark-gray, and black) such that player A has a winning strategy if and only if F is true. Let a_0 (colored light-gray) and b_0 (colored dark-gray) denote the start nodes of players A and B, respectively.

Each player controls a *pseudo-path*, that is, a path where some nodes may be duplicated as parallel nodes in a diamond-shaped structure; see Fig. 8. Player A controls path P_A and player B controls path P_B. A so-called *choice pair* consists of a node on a pseudo-path together with some duplicated node in parallel. The start nodes are at one end of the respective pseudo-paths, and the players can conquer the nodes on their own path without interference from the other player. However, they must do so in a timely manner because either path ends at a humongous *honey pot*, denoted respectively by H_A and H_B. A honey pot is a large clique of identically-colored nodes (we may think of it as a single node of large weight, because conquering one node will simultaneously conquer the entire clique). Both honey pots have the same size but different colors, namely black (H_A) and white (H_B), and they are connected to each other by an edge. Consequently, both players must rush along their pseudo-paths as quickly as possible to reach their honey pot before the opponent can reach it and to prevent the opponent from winning by conquering both honey pots. The last nodes before the honey pots are denoted by a_f and b_f, respectively. They separate the last variable gadgets (described below) from the honey pots. While rushing to the big honey pots, the players also try to conquer smaller honey pots associated with each clause; player A must win all of them to win, while player B tries to prevent this from happening.

Fig. 8 shows an overview of the pseudo-paths and one *variable gadget* in detail. A variable gadget is a part of the two pseudo-paths corresponding to a pair of variables $\exists x_{2i-1} \forall x_{2i}$, for some $i \geq 1$. For player A, the gadget starts at node a_{i-1} with a choice pair a^F_{2i-1} and a^T_{2i-1}, colored white and black, respectively. The first node conquered by A will determine the truth value for variable x_{2i-1}. In the same round, player B has a choice on his pseudo-path P_B between nodes b^F_{2i-1} and b^T_{2i-1}. Since these nodes have the same color as A's choices in the same round, B actually does not have a choice but must select the other color not chosen by A.

Three rounds later, player B has a choice pair b^F_{2i} and b^T_{2i}, assigning a truth value to variable x_{2i}. In the next step (which is in the next round), player A has a choice pair a^F_{2i} and a^T_{2i} with the same colors as B's choice pair for x_{2i}. Again, this means that A does not really have a choice but must select the color not chosen by B in the previous step. Since we want A to conquer

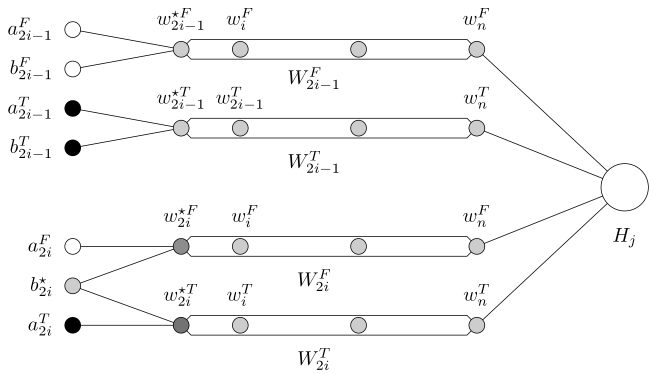

Fig. 9. The waiting gadgets for existential variables (W^F_{2i-1} and W^T_{2i-1}, the two top paths) and universal variables (W^F_{2i} and W^T_{2i}, the two bottom paths) in the proof of Theorem 4.4. Note that usually only one of the two waiting paths W^F_k or W^T_k would be connected to H_j because we may assume that a clause does not contain x_k and $\overline{x_k}$ at the same time.

those clauses containing a literal set to true by player B, the colors in B's choice pair have been switched, i.e., b^F_{2i} is black and b^T_{2i} is white.

Note that all the nodes $a_0, \ldots, a_n$ are light-gray and all the nodes $b_0, \ldots, b_n$ are dark-gray. This allows us to concatenate as many variable gadgets as needed. Further note that a_f is white, while b_f is light-gray.

The *clause* gadgets are very simple. Each clause C_j corresponds to a small honey pot H_j of color white. The size of the small honey pots is smaller than the size of the large honey pots H_A and H_B, but large enough such that player A loses if he misses one of them. Player A should conquer H_j if and only if C_j is true in the assignment chosen by the players while conquering their respective pseudo-paths. We could connect a^T_{2i-1} directly with H_j if C_j contains literal x_{2i-1}, however then player A could in subsequent rounds shortcut his pseudo-path by entering variable gadgets for the other variables in C_j from H_j. To prevent this from happening, we place waiting gadgets between the variable gadgets and the clauses. They are basically a copy of the remaining part of the pseudo-path.

Let $a^\star_k$ denote the node on P_A right after the choice pair a^F_k and a^T_k, for $k = 1, \ldots, 2n$; similarly, $b^\star_k$ are the nodes on P_B right after B's choice pairs. A *waiting gadget* W_k consists of two copies W^F_k and W^T_k of the sub-path of P_A starting at $a^\star_k$ and ending at a_n, see Fig. 9. If clause C_j contains literal x_k, H_j is connected to the node w^T_n corresponding to a_n in W^T_k; if C_j contains literal $\overline{x_k}$, H_j is connected to the node w^F_n corresponding to a_n in W^F_k. If $k = 2i - 1$ (i.e., we have an existential variable x_{2i-1} whose value is assigned by player A), then a^F_{2i-1} and b^F_{2i-1} are connected to $w^{\star F}_{2i-1}$, and a^T_{2i-1} and b^T_{2i-1} are connected to $w^{\star T}_{2i-1}$. If $k = 2i$ (i.e., we have a universal variable x_{2i} whose value is assigned by player B), then a^F_{2i} and $b^\star_{2i}$ are connected to $w^{\star F}_{2i}$, and a^T_{2i} and $b^\star_{2i}$ are connected to $w^{\star T}_{2i}$.

Finally, we connect b_f with all clause honey pots H_j to give player B the opportunity to conquer all those clauses that contain no true literal. This completes the construction of G_F. Fig. 10 shows the complete graph G_F for a small example formula F.

We claim that player A has a winning strategy on G_F if and only if formula F is true. It is easy to verify that player A can indeed win if F is true. All he has to do is to conquer those nodes in his existential choice pairs corresponding to the variable values in a satisfying assignment for F. For the existential variables, he has full control to select any value, and for the universal variables he must pick the opposite color as selected by player B in the previous step, which corresponds to setting the variable to exactly the value that player B has selected. Hence player B can block a move of player A by appropriately selecting a value for a universal variable. Note that no other blocking moves of player B are advantageous: if B blocks A's next move by choosing a color that does not make progress on his own pseudo-path, then A will simply make an arbitrary waiting move and then in the next round B cannot block A again. When player A conquers node a_n, he will simultaneously conquer the last nodes in all waiting gadgets corresponding to true literals. Since every clause contains a true literal for a satisfying assignment, player A can then in the next round conquer a_f together with all clause honey pots (which all have color white). Player B will respond by conquering b_f, and the game ends with both players conquering their own large honey pots H_A and H_B, respectively. Since player A got all clause honey pots, he wins.

To make this argument work, we must carefully choose the sizes of the honey pots. Each pseudo-path contains $9n + 1$ nodes, of which at most n can be conquered by the other player. The waiting gadgets contain two paths of length $9k + 6$ for existential variables and $9k + 1$ for universal variables, for $k = n - 1, n - 2, \ldots, 1, 0$, respectively (see Fig. 10 for an example). At the end, player A will have conquered one of the two paths completely and maybe some parts of the sibling path, that is, we do not know exactly the final owner of less than n^2 nodes. The clause honey pots should be large enough to absorb this fuzziness, which means it is sufficient to give them $2n^2$ nodes. The honey pots H_A and H_B should be large enough to punish any foul play by the players, that is, when they do not strictly follow their pseudo-paths. It is sufficient to give them $2n^3$ nodes.

To see that F is true if player A has a winning strategy note that player A must strictly follow his pseudo-path, as otherwise player B could beat him by reaching the large honey pots first. Thus player A's strategy induces a truth assignment for the

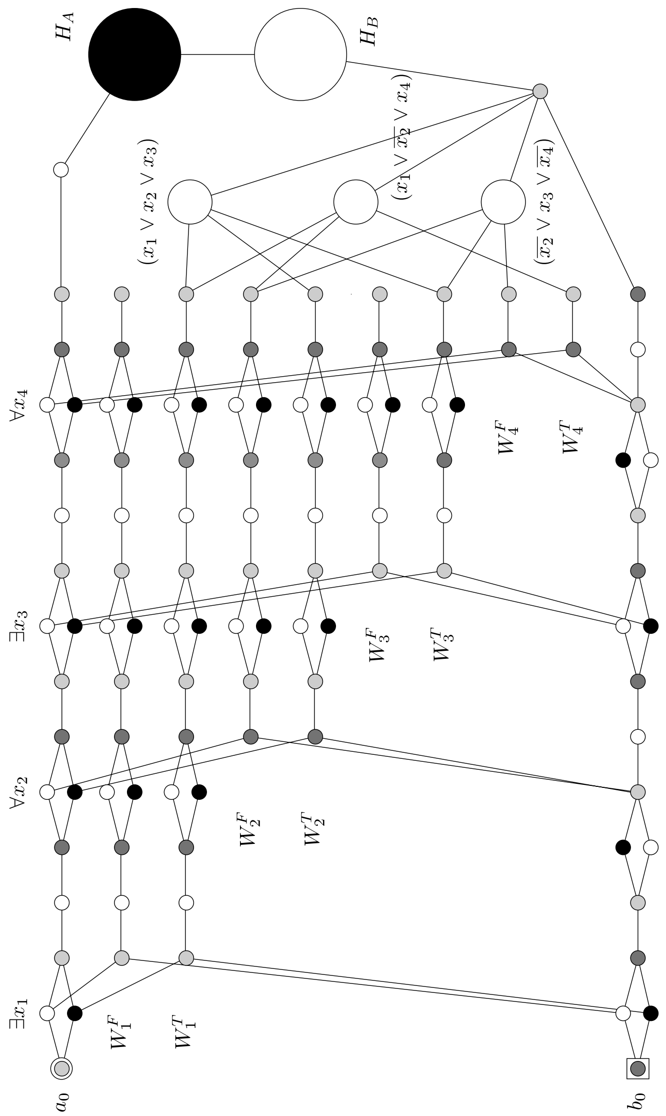

Fig. 10. The reduction in the proof of Theorem 4.4 would produce this graph for the formula $F = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4})$.

existential variables. Similarly, player *B*'s strategy induces a truth assignment for the universal variables. Player *A* can only win if he also conquers all clause honey pots, and hence the players must have chosen truth values that make at least one literal per clause true. This means that formula *F* is satisfiable. □

5. Conclusions

We have modeled the well-known Honey-Bee puzzle game as a combinatorial game on colored graphs. For the solitaire version, we have analyzed the complexity on many classes of perfect graphs. For the two-player version, we have shown that even the highly restricted case of series-parallel graphs is hard to tackle. Our results draw a clear separating line between easy and hard variants of these problems.

In particular, we showed that both the single player and the two-player versions are NP-hard on planar graphs. This implies that the original game played on a hex-grid *with holes* is also NP-hard for both variants. Furthermore, it is straightforward to adapt the solitaire hardness proof to the case of a complete hex-grid *without holes* using a construction similar to the proof of Lemma 2 in [2]. However, this approach does not seem to work for the two-player game (strings are represented by concentric rings of different colors; the second player would have to work from inside out but simultaneously for all these rings). Hence we pose the open problem of deciding the complexity of the two-player game on a hex-grid *without holes*.

Another open question is whether the two-player game is PSPACE-complete on series-parallel graphs and/or planar graphs. Finally, we conjecture that the solitaire version is NP-hard on trees even for *three* colors, whereas we only managed to establish NP-hardness for the case of four colors.

Acknowledgments

Part of this research was done while GJW visited Fudan University in spring 2009.

RF acknowledges support by the National Natural Science Foundation of China (No. 60973026), the Shanghai Leading Academic Discipline Project (project number B114), the Shanghai Committee of Science and Technology of China (09DZ2272800), and the Robert Bosch Foundation (Science Bridge China 32.5.8003.0040.0).

GJW acknowledges support by the Netherlands Organisation for Scientific Research (NWO grant 639.033.403), and by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

References

[1] H. Adachi, H. Kamekawa, S. Iwata, Shogi on $n \times n$ board is complete in exponential time (in Japanese), Transactions of the IEICE J70-D (1987) 843–1852.
[2] D. Arthur, R. Clifford, M. Jalsenius, A. Montanaro, B. Sach, The complexity of flood filling games, in: Proceedings of the 5th International Conference FUN with Algorithms, FUN'10, in: Springer Lecture Notes in Computer Science, 6099, 2010, pp. 307–318.
[3] T. C. Biedl, E. D. Demaine, R. Fleischer, L. Jacobsen, The complexity of Clickomania, in: R.J. Nowakowski (Ed.), More Games of No Chance — Combinatorial Games at MSRI, 2000, in: Mathematical Sciences Research Institute Publications, vol. 42, Cambridge University Press, Cambridge, England, 2002, pp. 389–404. Invited submission to a book on selected papers of the 2000 MSRI Workshop on Combinatorial Games.
[4] A. Born, Flash application for the computer game Biene (Honey-Bee), 2009. http://www.ursulinen.asn-graz.ac.at/Bugs/htm/games/biene.htm.
[5] R. Breukelaar, E. D. Demaine, S. Hohenberger, H. J. Hoogeboom, W. A. Kosters, D. Liben-Nowell, Tetris is hard, even to approximate, International Journal of Computational Geometry and Applications 14 (1–2) (2004) 41–68.
[6] E. D. Demaine, R. A. Hearn, Playing games with algorithms: algorithmic combinatorial game theory, in: M. H. Albert, R. J. Nowakowski (Eds.), Games of No Chance 3, in: Mathematical Sciences Research Institute Publications, vol. 56, Cambridge University Press, Cambridge, England, 2009, pp. 3–56.
[7] A. Fraenkel, D. Lichtenstein, Computing a perfect strategy for $n \times n$ Chess requires time exponential in n, Journal of Combinatorial Theory, Series A 31 (1981) 199–214.
[8] M. R. Garey, D. S. Johnson, Computers and Intractability — A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, New York, 1979.
[9] M. C. Golumbic, D. Rotem, J. Urrutia, Comparability graphs and intersection graphs, Discrete Mathematics 43 (1) (1983) 37–46.
[10] S. Iwata, T. Kasai, The Othello game on an $n \times n$ board is PSPACE-complete, Theoretical Computer Science 123 (1994) 329–340.
[11] D. Lichtenstein, M. Sipser, GO is polynomial-space hard, Journal of the ACM 27 (2) (1980) 393–401.
[12] C. Löding, P. Rohde, Solving the Sabotage game is PSPACE-hard, in: Proceedings of the 28th International Symposium on the Mathematical Foundations of Computer Science (MFCS'03), in: Springer Lecture Notes in Computer Science, vol. 2747, 2003, pp. 531–540.
[13] M. Middendorf, More on the complexity of common superstring and supersequence problems, Theoretical Computer Science 125 (1994) 205–228.
[14] S. Reisch, Gobang ist PSPACE-vollständig (Gomoku is PSPACE-complete), Acta Informatica 13 (1980) 59–66.
[15] S. Reisch, Hex ist PSPACE-vollständig (Hex is PSPACE-complete), Acta Informatica 15 (1981) 167–191.
[16] Lev Reyzin, 2 player Tetris is PSPACE hard, in: Proceedings of the 16th Fall Workshop on Computational and Combinatorial Geometry, FWCG'06, 2006. Electronic Proceedings, http://maven.smith.edu/~streinu/FwCG/proceedings.html.
[17] J. M. Robson, N by N Checkers is Exptime complete, SIAM Journal on Computing 13 (2) (1984) 252–267.
[18] String graph, 2011. http://en.wikipedia.org/w/index.php?title=String_graph&oldid=418952019.
[19] E. Verbin, Response to ''Is this game NP-hard?'', 2009. http://valis.cs.uiuc.edu/blog/?p=2005.
[20] D. Wolfe, Go endgames are PSPACE-hard, in: Proceedings of the 2000 MSRI Workshop on Combinatorial Games — More Games of No Chance, in: Mathematical Sciences Research Institute Publications, vol. 42, Cambridge University Press, Cambridge, England, 2002, pp. 125–136.