VERTEX NIM PLAYED ON GRAPHS

ERIC DUCHÊNE AND GABRIEL RENAULT

Abstract. Given a graph G with positive integer weights on the vertices, and a token placed on some current vertex u, two players alternately remove a positive integer weight from u and then move the token to a new current vertex adjacent to u. When the weight of a vertex is set to 0, it is removed and its neighborhood becomes a clique. The player making the last move wins. This adaptation of Nim on graphs is called Vertexnim, and slightly differs from the game Vertex NimG introduced by Stockman in 2004. Vertexnim can be played on both directed or undirected graphs. In this paper, we study the complexity of deciding whether a given game position of Vertexnim is winning for the first or second player. In particular, we show that for undirected graphs, this problem can be solved in quadratic time. Our algorithm is also available for the game Vertex NimG, thus improving Stockman's exptime algorithm. In the directed case, we are able to compute the w inning strategy in polynomial time for several instances, including circuits or digraphs with self loops.

Keywords: Combinatorial games; Nim; graph theory

1. Background and definitions

We assume that the reader has some knowledge in combinatorial game theory. Basic definitions can be found in [1]. We remind the reader that a $\mathcal{P}$ position denotes a position from which the second player has a winning strategy, while an $\mathcal{N}$ position means that the first player to move can win. Graph theoretical notions used in this paper will be standard and according to [2]. In particular, given a graph $G = (V, E)$ and a vertex v of V, we set $N(v) = \{w \in V : (v, w) \in E\}$.

The original idea of this work is the study of a variant of Nim, called Adjacent Nim, in which both players are forced to play on the heaps in a specific cyclic order: given N heaps of tokens of respective sizes $(n_1, \ldots, n_N)$, play the game of Nim under the constraint that if your opponent has moved on heap i, you must move on heap $i + 1$ (or on the smallest next non-empty heap, in a circular way). Actually, our investigations led us to consider Adjacent Nim as a particular instance of the game NimG (for "Nim on Graphs") introduced by Stockman in [20].

As a brief history of the game, we remind the reader that the game of Nim was introduced and solved by Bouton in 1904 [3]. Since then, lots of variations were considered in the literature, the most famous one being Wythoff's game [11, 21]. One can also mention [6, 12–14, 16] as a non-exhaustive list, or the newest Circular Nim [9], which deals with heaps of tokens arranged along a cycle. One of the most recent variants of Nim provides a topology to the heaps, which are organized as the edges of an undirected graph. This game was proposed by Fukuyama in 2003 [18, 19]. More precisely, an instance of this game is an undirected graph

$G = (V, E)$ with an integer weight function on E. A token is set on an arbitrary vertex. Then two players alternately move the token along an adjacent edge e with positive weight and decrease the label of e to any strictly smaller non-negative integer. The first player unable to move loses the game (this happens when the token has all its adjacent edges with a label equal to zero). In his papers, Fukuyama gives necessary and sufficient conditions for a position on a bipartite graph to be $\mathcal{P}$. He also computes the Grundy values of this game for some specific families of bipartite graphs, including trees, paths and cycles. In [10], a larger set of graphs is investigated (including complete graphs), but only for the weight function $f : E \mapsto \{1\}$.

In 2004, Stockman considered another generalization of Nim on graphs that she called Vertex NimG. The main difference with Fukuyama's work is that the Nim heaps are embedded into the vertices of a graph. This definition raises a natural question when playing the game: does the player first remove some weight from a vertex and then move to another one, or does he first move to a vertex and then remove weight from it?

- The variant *Move then remove* of Vertex NimG was recently investigated by Burke & George in [4]. They showed that in the case where each vertex of the input graph G has a self loop, then this game is PSPACE-hard. To the best of our knowledge, nothing was proved in the general case yet.
- The variant *Remove then move* of Vertex NimG is the one that was considerd by Stockman in [20]. In the case where the weight function is bounded by a constant, she gave a polynomial time algorithm to decide whether a given position is $\mathcal{P}$ or $\mathcal{N}$. The same algorithm can be applied in the general case, but becomes exponential according to the order of G.

In Fukuyama's or Stockman's definitions, the game ends when the player is blocked because of a null weight. This means that unlike the original game of Nim, their variants may end with remaining weight on the graph. To be closer to the original Nim, we have defined the rules of our variant of Vertex NimG in such a way that the game ends only when all the weight is removed from the graph. This variant was introduced on both directed and undirected graphs with possible loops, and under the *remove then move* convention. Multiple edges are not considered, since the weight is set on the vertices. We start by giving the definition of our game on undirected graphs, which is called Undirected vertexnim.

Definition 1. Undirected vertexnim. *Let $G = (V, E)$ be an undirected connected graph, let $w : V \to \mathbb{N}_{>0}$ be a function which assigns to each vertex a positive integer. Let $u \in V$ be a starting current vertex. In this game, two players alternately decrease the value of the current vertex u and choose an adjacent vertex of u as the new current vertex. When the value $w(v)$ of a vertex v is set to 0, then v and its incident edges are removed from G, the subgraph $N(v)$ of G becomes a clique, and a loop is added on each vertex of $N(v)$. The game ends when G is empty. The player who makes the last move wins the game.*

In this definition, we make $N(v)$ a clique (i.e., every two vertices of $N(v)$ are connected) after v reaches zero to prevent the graph to be disconnected. In other words, we can say that in order to choose the next current vertex, it suffices to follow any path of zero vertices ending on a non zero vertex. We also add loops to

prevent a player to be blocked on a vertex. The example below shows an execution of the game, the current vertex being the one with the triangle.

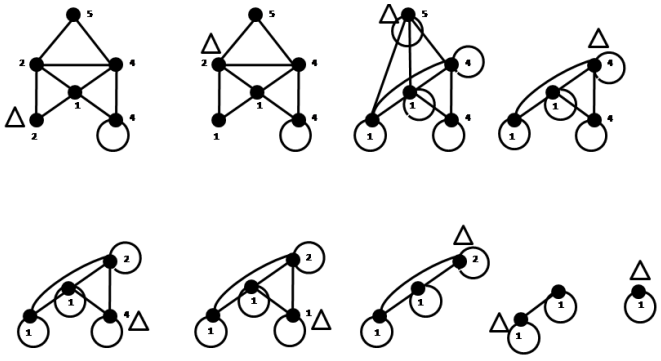

FIGURE 1. Playing UNDIRECTED VERTEXNIM

This game can naturally be extended to directed graphs, with some constraints to ensure that all the weight is removed in the end. In particular, arcs are added when the weight of a vertex goes to zero (by the same way that a clique is built in the undirected case). We also need to play on a strongly connected digraph, to avoid being blocked on a vertex having a null outdegree. Recall that in a *strongly connected digraph*, for every pair of vertices (u, v) there exists a path from u to v. This directed variant will be called DIRECTED VERTEXNIM.

Definition 2. DIRECTED VERTEXNIM. *Let $G = (V, E)$ be a strongly connected digraph, and let $w : V \to \mathbb{N}_{>0}$ be a function which assigns to each vertex a positive integer. Let $u \in V$ be the starting current vertex. In this game, two players alternately decrease the value of the current vertex u and choose an adjacent vertex of u as the new current vertex. When the value of a vertex v is set to 0, then v is removed from G and all the pairs of arcs (p, v) and (v, s) (with p and s not necessarily distinct) are replaced by an arc (p, s). The game ends when G is empty. The player who made the last move wins the game.*

With the above notation, instances of this game will be denoted (G, w, u) in the rest of the paper.

Note that in Definition 2, the strong connectivity of G is preserved when deleting a vertex. Hence it is always possible to play whenever G is not empty. Figure 1 illustrates a sequence of moves of DIRECTED VERTEXNIM, where both players remove all the weight of the current vertex at their turn.

The current paper deals with the complexity of both versions of VERTEXNIM, in the sense of Fraenkel [17]. In particular, we will prove the tractability of the game, which implies that the outcome ($\mathcal{P}$ or $\mathcal{N}$) of a game position can be computed in polynomial time. In Section 2, we will solve the game ADJACENT NIM, which is actually an instance of DIRECTED VERTEXNIM on circuits. Section 3 will be devoted to the resolution of DIRECTED VERTEXNIM for any strongly connected digraph having a loop on each vertex. Section 4 concerns UNDIRECTED VERTEXNIM, whose tractability is proved in the general case. As a corollary, we will show that

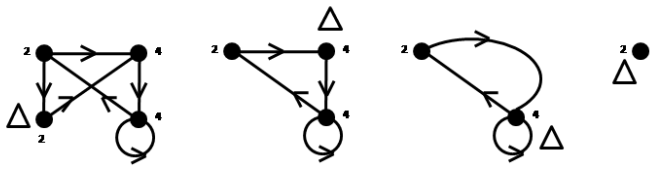

FIGURE 2. Playing DIRECTED VERTEXNIM

our algorithm also solves Stockman's VERTEX NIMG in quadratic time, improving the results presented in [20]. In Section 5, we finally mention how our results can be adapted to misère versions of VERTEXNIM.

2. ADJACENT NIM

As explained in the introduction, this game was the original motivation of our work. With the above formalism, it can be expressed as an instance of DIRECTED VERTEXIM on an elementary circuit $C_N = (v_1, v_2, \dots, v_N)$ with the orientation $(v_i, v_{i+1}) : 1 \leq i < N$ and (v_N, v_1) of the arcs. In Theorem 3, we fully solve ADJACENT NIM in the case where all the weights are strictly greater than 1. Without loss of generality, we will assume that the starting position is v_1.

Theorem 3. *Let $(C_N, w, v_1) : N \geq 3$ be an instance of* ADJACENT NIM *with $w : V \to \mathbb{N}_{>1}$.*

1. *If N is odd, then (C_N, w, v_1) is an $\mathcal{N}$ position.*
2. *If N is even, then (C_N, w, v_1) is an $\mathcal{N}$ position iff $\min\{\operatorname{argmin}_{1\leq i\leq N} w(v_i)\}$ is even.*

Note that when N is even, the above Theorem implies that the first player who must play on a vertex of minimum weight will lose the game.

Proof. (1) If N is odd, then the first player can apply the following strategy to win: first play $w(v_1) \to 1$. Then for all $1 \leq i < (N-1)/2$: if the second player empties v_{2i}, then the first player also empties the following vertex v_{2i+1}. Otherwise play $w(v_{2i+1}) \to 1$. The strategy is different for the last two vertices of C_N: if the second player empties v_{N-1}, then play $w(v_N) \to 1$, otherwise play $w(v_N) \to 0$. As $w(v_1) = 1$, the second player is now forced to empty v_1. Since an even number of vertices have been deleted at this point, we have an odd circuit to play on. It now suffices for the first player to empty all the vertices on the second run. Indeed, the second player is also forced to set each weight to 0 since he has to play on vertices satisfying $w = 1$. Since the circuit is odd, the first player is guaranteed to make the last move on v_N or v_{N-1}.
(2) If N is even, we claim that who must play the first vertex of minimum weight will lose the game. The winning strategy of the other player consists of decreasing by 1 the weight of each vertex at his turn. First assume that $\min\{\operatorname{argmin}_{1\leq i\leq N} w(v_i)\}$ is odd. If the strategy of the second player always consists of playing $w(v_i) \to w(v_i) - 1$, then the first player will be the first to set the weight of a vertex, say v_k, to 0 or 1. If he sets v_k to 0, then the second player now faces an instance (C'_{N-1}, w', v_{k+1}) with $w' : V' \to \mathbb{N}_{>1}$, which is winning according to statement (1). If he sets v_k to

1, then the second player will empty the following vertex v_{k+1}, leaving to the first player a position $(C'_{N-1} = (v'_1, v'_2, \ldots, v'_{N-1}), w', v'_1 = v_{k+2})$ with $w' : V' \to \mathbb{N}_{>1}$ except on $w'(v'_{N-1}) = w(v_k) = 1$. This position corresponds to the one of (1) after the first move, and is thus losing. In the case where $\min\{\underset{1\leq i\leq N}{\operatorname{argmin}}\ w(v_i)\}$ is even, then the first player, by applying the same strategy, can force the other player to be the first to set a vertex to 0 or 1, which makes him lose the game. □

Open Problem 4. *The question of deciding whether a given position is $\mathcal{P}$ or $\mathcal{N}$ remains open in the cases where some vertices have a weight equal to 1. Indeed the previous strategy cannot be applied anymore, and we did not manage to get satisfying results when the $1's$ are "owned" by different players (player i owns a vertex v if he is the first to play on it; in other words, player 1 owns vertices v_k with k odd, and player 2 owns vertices v_k with k even).*

Remark 5. *What if we adapt Stockman's* VERTEX NIMG *to directed graphs? Recall that it means that vertices of null weight are never removed, and a player who must play from a 0 loses. In the case of circuits, it is easy to see that Theorem 3 remains true, even if there are vertices of weight 1. On a general graph, we conjecture that this game should be at least as hard as the game* GEOGRAPHY *[15] (nevertheless a proof needs to be done).*

3. Directed graphs with all loops

Dealing with DIRECTED VERTEXNIM on any strongly connected digraph is much harder. We managed to decide whether a position is $\mathcal{P}$ or $\mathcal{N}$ only in the case where there is a loop on each vertex. This can be seen as a way to consider Stockman's NIMG by allowing moves in the extended neighborhood of v (i.e., v or its neighbors) after having removed weight on it. Note that this is also the variant considered in [4], but for the other rule convention (*Move then remove*).

Theorem 6. *Let (G, w, u) be an instance of* DIRECTED VERTEXNIM *where G is strongly connected with a loop on each vertex. Deciding whether (G,w,u) is $\mathcal{P}$ or $\mathcal{N}$ can be done in time $O(|V(G)||E(G)|)$.*

The proof of this theorem requires several definitions that we present here.

Definition 7. *Let $G = (V, E)$ be a directed graph. Let $S \subseteq V(G)$ be a non-empty set of vertices such that the graph induced by S is strongly connected and $\forall u \in S, \forall v \in (V(G)\backslash S)$, $(u, v) \notin E(G)$. Let $T = \{v \in V(G)\backslash S \mid \exists u \in S, (v, u) \in E(G)\}$. Let G_e be the graph induced by $V(G)\backslash S$ and G_o the graph induced by $V(G)\backslash(S \cup T)$. We define a labeling $ld_G : V(G) \to \{\mathcal{P}, \mathcal{N}\}$ as follows:*
If $|S|$ is even, we label $\mathcal{N}$ all elements of $|S|$ and we label elements of $V \setminus S$ as we would have labeled them in the graph G_e.
If $|S|$ is odd, we label $\mathcal{P}$ all elements of $|S|$, we label $\mathcal{N}$ all elements of T, and we label elements of $V \setminus (S \cup T)$ as we would have labeled them in the graph G_o.

When decomposing the graph into strongly connected components, S is one of those with no out-arc. The choice of S is not unique, unlike the ld_G labeling: if S_1 and S_2 are both strongly connected components without out-arcs, the one which is not chosen as the first set S will remain a strongly connected component after the removal of the other, and as it has no out-arc, none of its vertices will be in the T set.

Proof. Let G' be the induced subgraph of G such that $V(G') = \{v \in V(G) \mid w(v) = 1\}$.
If $G = G'$, then (G, w, u) is an $\mathcal{N}$ position if and only if $|V(G)|$ is odd since the problem reduces to "She loves me, she loves me not". We will now assume that $G \neq G'$, and consider two cases for $w(u)$:
• Assume $w(u) \geqslant 2$. If there is a winning move which reduces u to 0, then we can play it and win. Otherwise, reducing u to 1 and staying on u is a winning move. Hence (G, w, u) is an $\mathcal{N}$ position.
• Assume $w(u) = 1$, i.e., $u \in G'$. According to Definition 7, computing $ld_{G'}$ yields a sequence of couples of sets (S_i, T_i) (which is not unique). Note that we will not consider T_i when S_i has an even size, according to Definition 7. Thus the following assertions hold: if $u \in S_i$ for some i, then any direct successor v of u is either in the component S_i (as there are no out-arcs) or has been previously labeled (is in $\cup_{j<i}(S_j \cup T_j)$), and if $u \in T_i \neq \emptyset$ for some i, then there exists a direct successor v of u in the set S_i, with $ld_{G'}(v) = \mathcal{P}$.
Our goal is to show that (G, w, u) is an $\mathcal{N}$ position if and only if $ld_{G'}(u) = \mathcal{N}$ by induction on $|V(G')|$. If $|V(G')| = 1$, then $V(G') = \{u\}$ and $ld_{G'}(u) = \mathcal{P}$. Since $w(u) = 1$, we are forced to reduce u to 0 and go to a vertex v such that $w(v) \geqslant 2$, which we previously proved to be a losing move. Now assume $|V(G')| \geqslant 2$. First, note that when one reduces the weight of a vertex v to 0, the replacement of the arcs does not change the strongly connected components (except for the component containing v of course, which loses one vertex). Consequently, if $u \in S_i$ for some i, then for any vertex $v \in \cup_{l=1}^{i-1}(T_l \cup S_l)$, $ld_{G'\setminus\{u\}}(v) = ld_{G'}(v)$ and for any vertex $w \in S_i\setminus\{u\}$, $ld_{G'\setminus\{u\}}(w) \neq ld_{G'}(w)$ since parity of S_i has changed. If $u \in T_i$ for some i, then for any vertex $v \in (\cup_{l=1}^{i-1}(T_l \cup S_l)) \cup S_i$, $ld_{G'\setminus\{u\}}(v) = ld_{G'}(v)$.
We now consider two cases for u: first assume that $ld_{G'}(u) = \mathcal{P}$, with $u \in S_i$ for some i. We reduce u to 0 and we are forced to move to a direct successor v. If $w(v) \geqslant 2$, we previously proved this is a losing move. If $v \in \cup_{l=1}^{i-1}(T_l \cup S_l)$, then $ld_{G'\setminus\{u\}}(v) = ld_{G'}(v) = \mathcal{N}$ (if $ld_{G'}(v) = \mathcal{P}$, then $v \in S_l$ with $|S_l|$ odd, and thus $u \in T_l$!) and the move to v is a losing move by induction hypothesis. If $v \in S_i$, then $ld_{G'\setminus\{u\}}(v) \neq ld_{G'}(v) = \mathcal{P}$ and it is a losing move by induction hypothesis.
Now assume that $ld_{G'}(u) = \mathcal{N}$. If $u \in T_i$ for some i, we can reduce u to 0 and move to a vertex $v \in S_i$, which is a winning move by induction hypothesis. If $u \in S_i$ for some i, it means that $|S_i|$ is even, we can reduce u to 0 and move to a vertex $v \in S_i$, with $ld_{G'\setminus\{u\}}(v) \neq ld_{G'}(v) = \mathcal{N}$. This is a winning move by induction hypothesis. Hence, (G, w, u) is an $\mathcal{N}$ position if and only if $ld_{G'}(u) = \mathcal{N}$. Figure 3 illustrates the computation of the ld labeling on the graph G' induced by the circled vertices (i.e., those having a weight equal to 1). In this example, the first strongly connected component chosen is the triangle S_1, which has no in-arc in G'. Hence $T_1 = \emptyset$. Then we choose S_2 as a unique vertex on the top of the figure. Its unique predecessor in G' constitute T_2. The last two vertices make the two sets S_3 and S_4 (with $T_3 = T_4 = \emptyset$).

Concerning the complexity of the computation, note that when $w(u) \geqslant 2$, the algorithm answers in constant time. The computation of $ld_{G'}(u)$ when $w(u) = 1$ needs to be analyzed more carefully. Decomposing a directed graph H into strongly connected components to find the sets S and T can be done in time $O(|V(H)| + |E(H)|)$, and both $|V(H)|$ and $|E(H)|$ are less than or equal to $|E(G)|$ in our case

since H is a subgraph of G and G is strongly connected. Morevover, the number of times we compute S and T is clearly bounded by $|V(G)|$. These remarks lead to a global algorithm running in $O(|V(G)||E(G)|)$ time. □

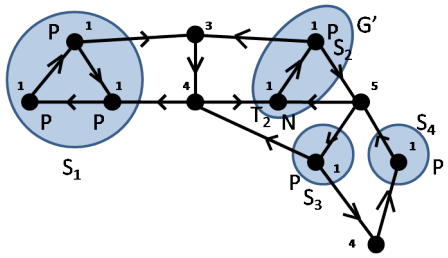

FIGURE 3. Example of *ld* labeling for subgraph G'

Open Problem 8. *Can one provide a characterization of the $\mathcal{P}$ and $\mathcal{N}$ positions in the general case where self loops are optional?*
Note that one of the reasons for which we have slightly changed Stockman's rules is that we thought it would make the game more affordable than VERTEX NIMG *or* GEOGRAPHY *[15] on directed graphs. The previous theorem shows that our assumption was somehow true, since we remind the reader that* VERTEX NIMG *was proved to be PSPACE-hard with all loops [4].*

4. UNDIRECTED GRAPHS

In the undirected case, it is easy to show that if each vertex has a self loop, deciding whether a position is $\mathcal{P}$ or $\mathcal{N}$ only depends on the size of the subset $\{v \in V \mid w(v) = 1\}$. This game can be solved by Theorem 6, by replacing each edge (u, v) by two arcs (u, v) and (v, u). Yet, the following proposition improves the complexity of the method, which becomes linear.

Proposition 9. *Let (G, w, u) be an instance of* UNDIRECTED VERTEXNIM *such that there is a loop on each vertex of G. Deciding whether (G, w, u) is $\mathcal{P}$ or $\mathcal{N}$ can be done in time $O(|V(G)|)$.*

Proof. Let G' be the induced subgraph of G such that $V(G') = \{v \in V(G) \mid w(v) = 1\}$.
If $G = G'$, then (G, w, u) is an $\mathcal{N}$ position if and only if $|V(G)|$ is odd since the problem reduces to "She loves move, she loves me not."In the rest of the proof, assume $G \neq G'$.
• We first consider the case where $w(u) \geqslant 2$. If there is a winning move which reduces u to 0, then we play it and win. Otherwise, reducing u to 1 and staying on u is a winning move. Hence (G, w, u) is an $\mathcal{N}$ position.
• Assume $w(u) = 1$. Let n_u be the number of vertices of the connected component of G' which contains u. We show that (G, w, u) is an $\mathcal{N}$ position if and only if n_u is even by induction on n_u. If $n_u = 1$, then we are forced to reduce u to 0 and move to another vertex v having $w(v) \geqslant 2$, which we previously proved to be a losing move. Now assume $n_u \geqslant 2$. If n_u is even, we reduce u to 0 and move to an adjacent vertex v with $w(v) = 1$, which is a winning move by induction hypothesis. If n_u is

odd, then we reduce u to 0 and we are forced to move to an adjacent vertex v. If $w(v) \geqslant 2$, then we previously proved it is a losing move. If $w(v) = 1$, this is also a losing move by induction hypothesis. Therefore in that case, (G, w, u) is an $\mathcal{N}$ position if and only if n_u is even.

Concerning the complexity of the computation, note that when $w(u) \geqslant 2$, the algorithm answers in constant time. When $w(u) = 1$, we only need to find connected component of G' containing u and the number of its vertices, which can be done in $O(|V(G)|)$ time. Thus, the algorithm runs in $O(|V(G)|)$ time. □

In the general case where the loops are optional, the tractability of the game is still guaranteed, even though the previous linear time algorithm is no longer available.

Theorem 10. *Let (G, w, u) be an instance of* UNDIRECTED VERTEXNIM. *Deciding whether (G, w, u) is $\mathcal{P}$ or $\mathcal{N}$ can be done in $O(|V(G)||E(G)|)$ time.*

The proof of this theorem requires several definitions that we present here.

Definition 11. *Let $G = (V, E)$ be an undirected graph with a weight function $w : V \to \mathbb{N}_{>0}$ defined on its vertices. Let $S = \{u \in V(G) \mid \forall v \in v(G), w(u) \leqslant w(v)\}$. Let $T = \{v \in V(G) \backslash S \mid \exists u \in S, (v, u) \in E(G)\}$. Let $\widetilde{G}$ be the graph induced by $G \backslash (S \cup T)$.*
We define a labeling $lu_{G,w}$ of its vertices as follows :
$\forall u \in S$, $lu_{G,w}(u) = \mathcal{P}$, $\forall v \in T$, $lu_{G,w}(v) = \mathcal{N}$ and $\forall t \in G \backslash (S \cup T)$, $lu_{G,w}(t) = lu_{\widetilde{G},w}(t)$.

Proof. Let G_u be the induced subgraph of G such that $V(G_u) = \{v \in V(G) \mid w(v) = 1$ *or* $v = u\}$, and G' be the induced subgraph of G such that $V(G') = \{v \in V(G) \mid w(v) \geqslant 2$ *and* $(v, v) \notin E(G)$ *and* $\forall t \in V(G), (v, t) \in E(G) \Rightarrow w(t) \geqslant 2\}$.
If $G = G_u$ and $w(u) = 1$, then (G, w, u) is an $\mathcal{N}$ position if and only if $|V(G)|$ is odd since it reduces to "She loves me, she loves me not."
If $G = G_u$ and $w(u) \geqslant 2$, we reduce u to 0 and move to any vertex if $|V(G)|$ is odd, and we reduce u to 1 and move to any vertex if $|V(G)|$ is even; both are winning moves, hence (G, w, u) is an $\mathcal{N}$ position.
In the rest of the proof we will assume that $G \neq G_u$. In the first three cases, we assume $u \notin G'$.
• *Case (1)* Assume $w(u) \geqslant 2$ and there is a loop on u. If there is a winning move which reduces u to 0, then we can play it and win. Otherwise, reducing u to 1 and staying on u is a winning move. Therefore (G, w, u) is an $\mathcal{N}$ position.
• *Case (2)* Assume $w(u) = 1$.
Let n_u be the number of vertices of the connected component of G_u which contains u. We will show that (G, w, u) is an $\mathcal{N}$ position if and only if n_u is even by induction on n_u. If $n_u = 1$, then we are forced to reduce u to 0 and move to another vertex v, with $w(v) \geqslant 2$, which was proved to be a losing move since it creates a loop on v. Now assume $n_u \geqslant 2$. If n_u is even, we reduce u to 0 and move to a vertex v satisfying $w(v) = 1$, which is a winning move by induction hypothesis (the connected component of G_u containing u being unchanged, except the removal of u). If n_u is odd, we reduce u to 0 and move to some vertex v, creating a loop on it. If $w(v) \geqslant 2$, we already proved this is a losing move. If $w(v) = 1$, it is a losing move by induction hypothesis. We can therefore conclude that (G, w, u) is an $\mathcal{N}$ position if and only if n is even. Figure 4 illustrates this case.

• *Case (3)* Assume $w(u) \geqslant 2$ and that there is a vertex v such that $(u, v) \in E(G)$ and $w(v) = 1$. Let n be the number of vertices of the connected component of G_u which contains u. If n is odd, we reduce u to 1 and we move to v, which we proved to be a winning move. If n is even, we reduce u to 0 and we move to v, which we also proved to be winning. Hence (G, w, u) is an $\mathcal{N}$ position in that case. Figure 4 illustrates this case.

• *Case (4)* Assume $u \in G'$. We will show that (G, w, u) is $\mathcal{N}$ if and only if

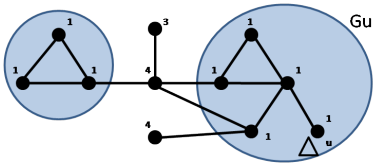

FIGURE 4. Case 2: $w(u) = 1$ and the connected component containing u has an odd size: this is a $\mathcal{P}$ position.

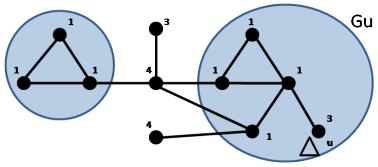

FIGURE 5. Case 3: an $\mathcal{N}$ position since u of weight $w(u) > 1$ has a neighbor of weight 1.

$lu_{G',w}(u) = \mathcal{N}$ by induction on $\sum_{v \in V(G')} w(v)$. If $\sum_{v \in V(G')} w(v) = 2$, we get $G' = \{u\}$ and we are forced to play to a vertex v such that $w(v) \geqslant 2$ and $v \notin V(G')$, which we proved to be a losing move. Assume $\sum_{v \in V(G')} w(v) \geqslant 2$. If $lu_{G',w}(u) = \mathcal{N}$, we reduce u to $w(u) - 1$ and move to a vertex v of G' such that $w(v) < w(u)$ and $lu_{G',w}(v) = \mathcal{P}$. Such a vertex exists by definition of lu. Let (G_1, w_1, v) be the resulting position after such a move. Hence $lu_{G'_1,w_1}(v) = lu_{G',w}(v) = \mathcal{P}$ since the only weight that has been reduced remains greater or equal to the one of v. And (G_1, w_1, v) is a $\mathcal{P}$ position by induction hypothesis. If $lu_{G',w}(u) = \mathcal{P}$, the first player is forced to reduce u and to move to some vertex v. Let (G_1, w_1, v) be the resulting position. First remark that $w_1(v) \geqslant 2$ since $u \in G'$. If he reduces u to 0, he will lose since v now has a self loop. If he reduces u to 1, he will also lose since $(u, v) \in E(G_1)$ and $w_1(u) = 1$ (according to case (3)).

Assume we reduced u to a number $w_1(u) \geqslant 2$. Thus $lu_{G'_1,w_1}(u)$ still equals $\mathcal{P}$ since the only weight we modified is the one of u and it has been decreased. If $v \notin G'$, i.e., v has a loop or there exists $t \in V(G_1)$ such that $(v, t) \in E(G_1)$ and $w_1(t) = 1$, then the second player wins according to cases (1) and (3). If $v \in G'$ and $lu_{G',w}(v) = \mathcal{N}$, then $lu_{G'_1,w_1}(v)$ is still $\mathcal{N}$ since the only weight we modified is the one of a vertex labeled $\mathcal{P}$. Consequently the resulting position makes the second player win by induction hypothesis. If $v \in G'$ and $lu_{G',w}(v) = \mathcal{P}$, then we necessarily have $w(v) = w(u)$ in G'. As $lu_{G'_1,w_1}(u) = \mathcal{P}$ and $(u, v) \in E(G_1)$, then $lu_{G'_1,w_1}(v)$ becomes $\mathcal{N}$, implying that the second player wins by induction hypothesis. Hence (G, w, u) is $\mathcal{N}$ if and only if $lu_{G',w}(u) = \mathcal{N}$. Figure 4 shows an example of the lu labeling.

Concerning the complexity of the computation, note that all the cases except (4) can be executed in $O(|E(G)|)$ operations. Hence the computation of $lu_{G',w}(u)$ to solve case (4) becomes crucial. It is rather straightforward to see that in the worst case, the computation of S and T can be done in $O(|E(G)|)$ time. And

the number of times where S and T are computed in the recursive definition of lu is clearly bounded by $|V(G)|$. All of this leads to a global algorithm running in $O(|V(G)||E(G)|)$ time. □

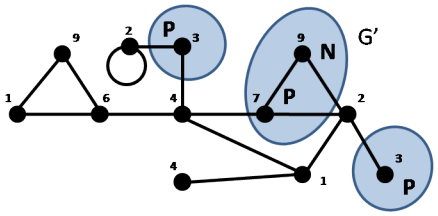

FIGURE 6. Case 4: lu-labeling of the subgraph G'

The technique described above can also be applied to Stockman's version of the game VERTEX NIMG. In [20], an exptime algorithm is given to decide the outcome of a given position. We here show that the complexity can be decreased to $O(|V||E|)$.

Corollary 12. *Let (G, w, u) be an instance of* VERTEX NIMG *with $w : V \to \mathbb{N}_{>0}$. Deciding whether (G, w, u) is $\mathcal{P}$ or $\mathcal{N}$ can be done in $O(|V(G)||E(G)|)$ time.*

Proof. The proof works similarly to the previous one, except that the subgraph G_u is no longer useful. Hence we have four cases:

- If $w(u) = 1$ and u has no self loop, then the position is $\mathcal{P}$.
- If $w(u) \geq 1$ and there is a loop on u, then it is $\mathcal{N}$.
- If $w(u) \geq 2$ and there is a vertex v such that $(u, v) \in E$ and $w(v) = 1$, then it is an $\mathcal{N}$ position.
- If $u \in G'$, then compute $lu_{G',w}(u)$ as in Theorem 10.

□

Remark 13. *Note that the proof still works if there exist vertices of null weight at the beginning. It suffices to consider the two following properties: if $w(u) = 0$ then this is $\mathcal{P}$, and if u is adjacent to some v with $w(v) = 0$, then it is $\mathcal{N}$.*

Open Problem 14. *We showed that* VERTEX NIMG *and* UNDIRECTED VERTEXNIM *can both be solved in polynomial time. Does this remain true when considering the Move then remove convention?*

5. MISÈRE VERSIONS

The misère version of a game is a game with the same rules except that the winning condition is reversed, i.e., the last player to move loses the game. The following results shows that in almost all cases, misère and normal versions of VERTEXNIM have the same outcomes.

Theorem 15. *Let (G, w, u) be an instance of* UNDIRECTED VERTEXNIM *under the misère convention. Deciding whether (G, w, u) is $\mathcal{P}$ or $\mathcal{N}$ can be done in $O(|V(G)||E(G)|)$ time.*

Proof. If all vertices have weight 1, then (G, w, u) is an $\mathcal{N}$ position if and only if $|V(G)|$ is even since it reduces to the misère version of "She loves me, she loves me not". Otherwise, we can use the same proof as the one of Theorem 10 to see that (G, w, u) is $\mathcal{N}$ in the misère version if and only if it is $\mathcal{N}$ in the normal version. □

Theorem 16. *Let (G, w, u) be an instance of* DIRECTED VERTEXNIM *in the misère version, where G is strongly connected, with a loop on each vertex. Deciding whether (G,w,u) is $\mathcal{P}$ or $\mathcal{N}$ can be done in time $O(|V(G)||E(G)|)$.*

Proof. If all vertices have weight 1, then (G, w, u) is an $\mathcal{N}$ position if and only if $|V(G)|$ is even since it reduces to the misère version of "She loves me, she loves me not."Otherwise, we can use the same proof as the one of Theorem 6 to see that (G, w, u) is $\mathcal{N}$ in the misère version if and only if it is $\mathcal{N}$ in the normal version. □

Remark 17. *Though the algorithms we give for both* UNDIRECTED VERTEXNIM *and* DIRECTED VERTEXNIM *can easily be adapted for the misère version, it does not seem to be the case with the algorithm we give for* VERTEX NIMG.

CONCLUSION

When dealing with an undirected graph, we proved that both versions of VERTEX NIMG (Stockman's version where the game ends whenever a player is blocked on a 0, and our version which allows play until all the weight is removed from the graph) are tractable. We even proved that deciding whether a given position is $\mathcal{P}$ or $\mathcal{N}$ can be done in quadratic time, which is a real improvement compared to the exptime algorithm presented in [20]. Unfortunately, the directed case turns out to be more tricky, even for simple graphs such as circuits. Yet, it seems that our variant of Nim on graphs is more accessible than VERTEX NIMG or GEOGRAPHY, as the results obtained in Theorem 6 allow us to be optimistic for graphs where loops become optional.

REFERENCES

[1] E. Berlekamp, J. H. Conway and R. K. Guy, *Winning Ways for your Mathematical Plays*, Vol. 1, Second edition. A K Peters, Ltd., Natick, MA, (2001).

[2] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, McMillan, London, Elsevier, New York, (1976).

[3] C. L. Bouton, Nim, a game with a complete mathematical theory, *Annals of Mathematics* **3** (1905), 35–39.

[4] K. Burke and O. George, A PSPACE-complete Graph Nim, to appear in Games of No Chance 5, arXiv:1101.1507v2 [cs.GT].

[5] E. Duchêne, A. S. Fraenkel, R. Nowakowski and M. Rigo, Extensions and restrictions of Wythoff's game preserving Wythoff's sequence as set of P positions, *Journal of Combinatorial Theory series A*. **117** (2010), 545–567.

[6] E. Duchêne and S. Gravier, Geometrical extensions of Wythoff's game, *Discrete Mathematics* **309** (2009), 3595–3608.

[7] E. Duchêne and M. Rigo, A morphic approach to combinatorial games : the Tribonacci case, *Theoretical Information and Applications* **42** (2008), 375–393.

[8] E. Duchêne and M. Rigo, Cubic Pisot Unit Combinatorial Games, *Monatshefte für Mathematik* **155** (2008), 217–249.

[9] M. Dufour and S. Heubach, Circular Nim Games, *Electronic Journal of Combinatorics* **20(2)** (2013).

[10] L. Erickson, Nim on the complete graph, arXiv:1010.1455v1 [math.CO] (2010).

[11] A. S. Fraenkel, How to beat your Wythoff games' opponent on three fronts, *American Mathematical Monthly* **89** (1982), 353–361.

[12] A. S. Fraenkel, The Raleigh game, *Combinatorial number theory*, de Gruyter, Berlin (2007), 199–208.
[13] A. S. Fraenkel, The Rat and the Mouse game, preprint.
[14] A. S. Fraenkel and I. Borosh, A generalization of Wythoff's game, *Journal of Combinatorial Theory series A*. **15** (1973), 175–191.
[15] A.S. Fraenkel and S. Simonson, Geography, *Theoretical Computer Science* **110** (1993), 197–214.
[16] A. S. Fraenkel and D. Zusman, A new heap game, *Theoretical Computer Science* **252** (2001), 5–12.
[17] A.S. Fraenkel, Complexity, Appeal and Challenges of combinatorial games, *Theoretical Computer Science* **313** (2004), 393–415.
[18] M. Fukuyama, A Nim game played on graphs, *Theoretical Computer Science* **304** (2003), 387–399.
[19] M. Fukuyama, A Nim game played on graphs II, *Theoretical Computer Science* **304** (2003), 401–419.
[20] G. Stockman. Presentation: The game of nim on graphs: NimG (2004). Available at `http://www.aladdin.cs.cmu.edu/reu/mini_probes/papers/final_stockman.ppt.`
[21] W. A. Wythoff, A modification of the game of Nim, *Nieuw Archief voor Wiskunde* **7** (1907), 199–202.

(E. Duchêne)
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France
`eric.duchene@univ-lyon1.fr`

(G. Renault)
Université Bordeaux 1, LaBRI, France
`gabriel.renault@labri.fr`