Theoretical Computer Science 110 (1993) 215–245
Elsevier

Mathematical Games

Complexity of path-forming games*

Hans L. Bodlaender

Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Communicated by J.H. van Lint
Received January 1991
Revised February 1992

Abstract

Bodlaender, H.L., Complexity of path-forming games, Theoretical Computer Science 110 (1993) 215–245.

For a number of two-player games where players alternately choose the next vertex of a simple or an elementary path in a graph, we consider the problem to determine whether, for a given game instance, there is a winning strategy for the first player. We show several of these problems to be PSPACE-complete. In some special cases, we obtain polynomial-time algorithms, based on graph rewriting or an intricate form of dynamic programming, e.g. we show GENERALIZED GEOGRAPHY and some other PSPACE-complete problems to be linear-time-solvable on graphs with constant bounded treewidth.

1. Introduction

Games are not only a popular pastime, but they can also serve as a model for several different phenomena, e.g.

- conflicts between parties with different interests (e.g. different companies that operate on the same market);
- fault tolerance. Here the erroneous behavior of a system is modeled by assuming that the system uses an intelligent strategy to prevent us from reaching our goal. If

Correspondence to: H.L. Bodlaender, Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands. Email: hansb@cs.ruu.nl.

* This research was partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075 (project ALCOM).

we are able to deal with this type of error, we are also able to deal with all weaker types of errors;

- worst-case complexity of algorithms (see e.g. [37]);
- complexity theory. For instance, the definition of the alternating Turing machine (which is one of the standard models for parallel computation) can be stated in terms of a game (see [29]).

In this paper we restrict ourselves to games with 2 players that have full information. We concentrate on the question: Given a certain game, how hard is it to determine whether there is a winning strategy for the first (or second) player. With the notion of "game", we usually mean a class of actual games (instances), where each game (instance) is distinguished from others in the class only by its starting position and playing area, but uses the same type of moves. With the complexity of a game (class), we denote the complexity of the following problem: Given a game (instance) from this class, does player 1 have a winning strategy in this game (instance)? Throughout this paper, we use the name of a game to denote this problem for that game.

For several problems of this type, PSPACE-completeness and EXPTIME-completeness results have been obtained, as well as for well-known and often played games, like chess, go, and checkers [24, 26, 35], and for more abstract games [1, 16, 22, 25, 40, 43, 45]. For an overview, see e.g. [28, 29].

In this paper we consider several abstract games on graphs that have "forming paths in a graph" as a common theme. We consider two already studied games, both known under the name GENERALIZED GEOGRAPHY, and several new games: the HAMILTONIAN CIRCUIT CONSTRUCTION GAME, the HAMILTONIAN PATH CONSTRUCTION GAME, the SIMPLE PATH CONSTRUCTION GAME, the ELEMENTARY PATH CONSTRUCTION GAME, and variants where the starting vertices are not specified.

In each of these games, the players are forming together one path in the graph. Some related games, called PARTIZAN GEOGRAPHY or TRON, where both players form their own path in the graph have also been proposed and studied by different authors [18, 27].

For most of the games considered in this paper, we prove the corresponding decision problem to be PSPACE-complete (under logarithmic-space reductions). For a few of these cases we have also a PSPACE-completeness result for the variant with undirected graphs. These results are presented in Section 3.

A huge amount of research has been done on the complexity of NP-complete graph problems when restricted to special classes of graphs (see e.g. [30]). Less is known on the complexity of PSPACE-complete graph problems when restricted to special classes of graphs. In Section 4, we give some new results in this area. We use the following as the main techniques: graph rewriting and an intricate form of dynamic programming. Among others, we show that (VERTEX) GENERALIZED GEOGRAPHY and some other PSPACE-complete problems are linear-time-solvable on graphs with bounded treewidth. This is the first known example of a PSPACE-complete problem solvable in polynomial time when restricted to graphs with bounded treewidth, where

previously such results were known only for problems known to be NP-complete or NP-hard [6, 8, 11, 15, 20, 21, 32, 44, 46]. In a recent paper, Arnborg [3] showed that some PSPACE-complete logic problems can also be solved in linear time when a graph with bounded treewidth can be associated with the input formula. Some final comments are made in Section 5.

2. Definitions

In this section we give most definitions that are needed in this paper. In Section 2.1 we give the definitions of graph-theoretic notions; in Section 2.2 we give the definitions of the considered games, and of a PSPACE-complete logic problem, used in our PSPACE-completeness proofs.

2.1. Graph-theoretic definitions

A path in a graph $G=(V, E)$ is called *simple* if no vertex appears more than once on the path. It is called *elementary* if no edge $e\in E$ is used more than once by the path. The subgraph of $G=(V, E)$ induced by $W\subseteq V$ is denoted by $G[W]=(W, \{(v, w)\in E \mid v, w\in W\})$.

Next we give the definitions of two special classes of graphs: the cacti, and the graphs with treewidth $\leqslant k$.

Definition. An undirected graph $G=(V, E)$ is a cactus (graph) if and only if every edge $e\in E$ belongs to at most one simple cycle in G.

In other words, a graph $G=(V, E)$ is a cactus if and only if each biconnected component of G is either a single edge or a cycle without chords.

Definition. Let $G=(V, E)$ be a directed or undirected graph. A tree decomposition of G is a pair $(\{X_i \mid i\in I\}, T=(I, F))$, with $\{X_i \mid i\in I\}$ a family of subsets of V, and T a tree, such that

- $\bigcup_{i\in I} X_i = V$.
- $\forall (v, w)\in E: \exists i\in I: v\in X_i \wedge w\in X_i$.
- $\forall i, j, k\in I$: if j is on the path from i to k in T, then $X_i\cap X_k\subseteq X_j$.

The treewidth of a tree decomposition $(\{X_i \mid i\in I\}, T=(I, F))$ is $\max_{i\in I} |X_i|-1$. The treewidth of G is the minimum treewidth over all possible tree decompositions of G.

An example of a graph with treewidth 2, and a tree decomposition of this graph can be found in Fig. 1.

The problem to determine the treewidth of a graph is NP-complete [4]. However, for fixed k, one can determine in $\mathcal{O}(n)$ time whether the treewidth of a given graph is $\leqslant k$ [5], and, if so, find a tree decomposition with treewidth $\leqslant k$ in $\mathcal{O}(n \log n)$ time

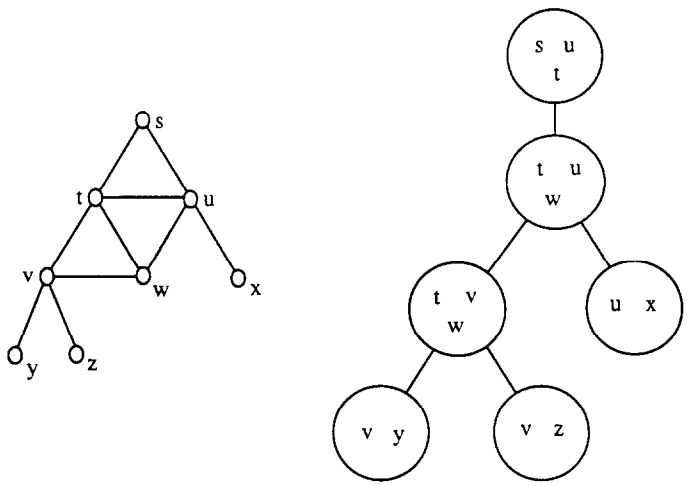

Fig. 1.

[18, 38]. (An $O(n \log^2 n)$ algorithm can be found by combining the results in [31] and [18].)

Several well-studied classes of graphs have the property that there is a constant upper bound for the treewidth of graphs in the class. For instance, cacti and series–parallel graphs have treewidth $\leqslant 2$, Halin graphs have treewidth 3, almost-trees with parameter k have treewidth $\leqslant k+1$, k-outerplanar graphs have treewidth $\leqslant 3k-1$. Also, if there is a fixed upper bound on the bandwidth, cutwidth, search number, vertex separation number of graphs in a class, or if the graphs in the class are interval graphs or chordal graphs with a fixed upper bound on the maximum clique size, then there is also a fixed upper bound on the treewidth of the graphs in the class (see [10, 13, 44, 46]).

There are several equivalent characterizations of the class of graphs with treewidth $\leqslant k$. For instance, a graph is partial k-tree if and only if its treewidth is at most k (see [2, 44]). Note also that each class of graphs that can be defined recursively in terms of rules of compositions with the method of Bern et al. [9] has a constant upper bound on the treewidth of the graphs in the class.

So, a polynomial algorithm for a certain problem for graphs with bounded treewidth directly implies polynomial algorithms for that problem for a large number of other classes of graphs.

2.2. *Definitions of games and a logical problem*

In this section we give definitions of some logical problems, used in our PSPACE-completeness proofs, and of the games we consider in this paper. First we give the definition of a well-known logical problem.

QUANTIFIED 3-SATISFIABILITY

Instance: Set $U=\{u_1, u_2, \ldots, u_n\}$ of variables, well-formed quantified boolean formula $F=(Q_1 u_1)(Q_2 u_2)\ldots(Q_n u_n)E$, where E is a boolean expression in conjunctive normal form with three literals per clause, and each Q_i is either $\forall$ or $\exists$.

Question: Is F true?

QUANTIFIED 3-SATISFIABILITY is PSPACE-complete. One may also assume that the quantifiers alternate, i.e., that $Q_i=\forall$ if i is even, and $Q_i=\exists$ if i is odd, or that $Q_i=\exists$ if i is even, and $Q_i=\forall$ if i is odd (see [28, 42]).

Next we introduce several games on graphs. We start with two slightly different games, which are both known under the name GENERALIZED GEOGRAPHY. To distinguish the variants, we call them here VERTEX GENERALIZED GEOGRAPHY and EDGE GENERALIZED GEOGRAPHY. Both games are played on a directed graph $G=(V, E)$, given with a starting vertex s. In the VERTEX GENERALIZED GEOGRAPHY game, players alternately choose a vertex. The first chosen vertex must be s, and each subsequently chosen vertex must have an incoming edge with the last chosen vertex as its other endpoint. Players may not choose a vertex that has been chosen before. The first player that is unable to move loses the game. In the EDGE GENERALIZED GEOGRAPHY game, players alternate choosing an edge that has not been chosen before, starting with an edge that has its tail at s; subsequent edges must have their tail at the vertex that was the head of the previous chosen edge. Again, the first player unable to move loses the game.

Thus, in other words, in VERTEX GENERALIZED GEOGRAPHY, players alternately choose the next vertex of a simple path in G, and in EDGE GENERALIZED GEOGRAPHY, players alternately choose the next edge of an elementary path.

Both games are PSPACE-complete. EDGE GENERALIZED GEOGRAPHY was proven to be PSPACE-complete by Schaefer [43]. In [35] it was proved that VERTEX GENERALIZED GEOGRAPHY is PSPACE-complete for planar, bipartite graphs that have no vertices with in/outdegree exceeding 2 or with degree exceeding 3.

In this paper we consider also the following variants of these games:

- SIMPLE PATH CONSTRUCTION GAME. This game is played as VERTEX GENERALIZED GEOGRAPHY, but with the following difference: the instance also contains an integer $k \leqslant |V|$. Player 1 wins if and only if the game ends with a path containing at least k vertices.
- ELEMENTARY PATH CONSTRUCTION GAME. This game is played as EDGE GENERALIZED GEOGRAPHY, but now the instance contains an integer $k \leqslant |E|$, and player 1 wins if and only if the game ends with a path containing at least k edges.
- HAMILTONIAN PATH CONSTRUCTION GAME is the special case of SIMPLE PATH CONSTRUCTION GAME with $k=|V|$. In other words, player 1 wins if and only if the game ends when all vertices have been visited.
- HAMILTONIAN CIRCUIT CONSTRUCTION GAME. This is similar to the HAMILTONIAN PATH CONSTRUCTION GAME, but now player 1 wins if and only if the game ends when all vertices have been visited and there is an edge from the last visited vertex to the first visited vertex.

- Variants *without specified starting vertex*. These are similar to the original games, but now player 1 is free to choose whatever vertex or edge he/she wants as starting vertex or edge.

In this paper, we will also consider variants where the games are played on undirected graphs.

The game VERTEX GENERALIZED GEOGRAPHY without specified starting vertex, on undirected graphs, is solvable in polynomial time: there is a winning strategy for player 1 if and only if the input graph G contains no perfect matching [19].

There is a close connection between VERTEX and EDGE GENERALIZED GEOGRAPHY and games with rules, forbidding positions on moves to appear more than once. A game can be modeled by a directed graph, where the vertices correspond to positions in the game. Edges correspond to possible moves from a position. (For most games, this graph has a very large size.) If it is forbidden to move to a position that has appeared already earlier in the game, then this corresponds to the condition that the players alternately choose a vertex on a simple path; a rule that forbids the same move from the same position corresponds to an elementary path (compare [41]).

3. PSPACE-completeness results

In this section we give a number of new PSPACE-completeness results. We establish PSPACE-completeness for the following games/problems:

- VERTEX GENERALIZED GEOGRAPHY without specified starting vertex for directed graphs.
- HAMILTONIAN PATH CONSTRUCTION GAME with and without specified starting vertex for directed graphs.
- HAMILTONIAN CIRCUIT CONSTRUCTION GAME with and without specified starting vertex for directed graphs.
- SIMPLE PATH CONSTRUCTION GAME with and without specified starting vertices for directed and for undirected graphs.
- ELEMENTARY PATH CONSTRUCTION GAME with and without specified starting vertex for directed and for undirected graphs.

In each of our proofs, we use either a transformation from QUANTIFIED 3-SATISFIABILITY similar to the proof for (VERTEX or EDGE) GENERALIZED GEOGRAPHY in [35, 43], or a transformation from a closely related game.

Theorem 3.1. VERTEX GENERALIZED GEOGRAPHY *without specified starting vertex is* PSPACE-*complete.*

Proof. Clearly, the problem is in PSPACE. To prove PSPACE-hardness, we use a transformation from the standard VERTEX GENERALIZED GEOGRAPHY problem (i.e., with specified starting vertex).

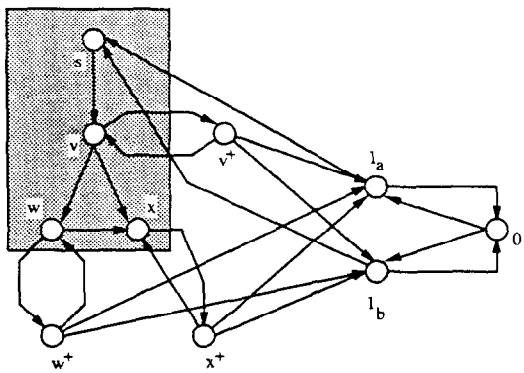

Fig. 2.

Let an instance $G=(V, E)$, $s\in V$ of VERTEX GENERALIZED GEOGRAPHY be given. We may suppose that the indegree of s is 0 (it is never possible to traverse an edge to the starting vertex, so these edges may as well be deleted from G).

Now, let $G'=(V', E')$ be defined as follows (see Fig. 2 for an example):

$$V' = V \cup \{0, 1_a, 1_b\}$$
$$\cup \{v^+ \mid v\in V \text{ and } v\neq s\},$$
$$E' = E \cup \{(v, v^+) \mid v\in V \text{ and } v\neq s\}$$
$$\cup \{(v^+, v) \mid v\in V \text{ and } v\neq s\}$$
$$\cup \{(v^+, 1_a) \mid v\in V \text{ and } v\neq s\}$$
$$\cup \{(v^+, 1_b) \mid v\in V \text{ and } v\neq s\}$$
$$\cup \{(1_a, s), (1_b, s), (0, 1_a), (0, 1_b), (1_a, 0), (1_b, 0)\}.$$

As the construction of G' can be carried out in logarithmic work space, the theorem follows with the help of the following claim. □

Claim 3.2. *There is a winning strategy for player* 1 *for* VERTEX GENERALIZED GEOGRAPHY *with starting vertex* s *on* G *if and only if there is a winning strategy for player* 1 *for* VERTEX GENERALIZED GEOGRAPHY *without specified starting vertex on* G'.

Proof. Suppose player 1 has a winning strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G. Then he can use the following strategy for VERTEX GENERALIZED GEOGRAPHY without specified starting vertex on G': start in 0. Player 2 will move to 1_a or 1_b. Move to s. Now, as long as player 2 moves to vertices in V, also move to vertices in V, using the original strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G. After a number of moves, player 2 will be unable to move

to a vertex in V. So, player 2 will move eventually to a vertex v^+. Now player 1 moves to the unused vertex in $\{1_a, 1_b\}$ and wins the game.

Suppose, player 2 has a winning strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G. Player 2 now also has a winning strategy for VERTEX GENERALIZED GEOGRAPHY without specified starting vertex on G'. We consider a number of cases:

Case 1: If player 1 starts at 0, then player 2 wins, using a strategy similar to the argument above.

Case 2: Suppose player 1 starts at 1_a or 1_b. Without loss of generality, suppose player 1 starts at 1_a. Player 2 moves to s. Player 1 must move to a $v \in V$, $v \neq s$. Player 2 moves to v^+. Player 1 must move to 1_b. Player 2 moves to 0 and wins.

Case 3: Suppose player 1 starts at $v \in V$, $v \neq s$. Then player 2 moves to v^+. Player 1 must move to 1_a or 1_b. Player 2 moves to 0. Player 1 must move to the unused vertex in $\{1_a, 1_b\}$. Player 2 moves to s. Player 1 must move to a vertex $v \in V$, $v \neq s$. Player 2 moves to v^+ and wins the game.

Case 4: Player 1 starts at a vertex v^+, $v \in V$. Player 2 moves to v. Player 1 must move to a vertex $w \in V$. $w \neq s$, because indegree$(s) = 0$. Player 2 moves to w^+. Player 1 must move to 1_a or 1_b. Player 2 moves to 0. Player 1 must move to the unused vertex in $\{1_a, 1_b\}$. Player 2 moves to s. Player 1 must move to a vertex $x \in V$, $x \neq v$, $x \neq w$. (If x does not exist, player 1 loses directly). Now player 2 moves to x^+ and wins the game.

Case 5: Player 1 starts at vertex s. Player 2 now follows the strategy for VERTEX GENERALIZED GEOGRAPHY with starting vertex s on G, as long as player 1 moves to vertices $v \in V$. Eventually, player 1 must move to a vertex v^+. Now player 2 moves to 1_a, player 1 to 0, and player 2 wins by moving to 1_b. □

Corollary 3.3. VERTEX GENERALIZED GEOGRAPHY *without specified starting vertex is* PSPACE-*complete for graphs with thickness* $\leqslant 2$.

Proof. VERTEX GENERALIZED GEOGRAPHY with specified starting vertex is PSPACE-complete for planar graphs [35]. If the construction in the proof of Theorem 3.1 is applied to a planar graph, one obtains a graph with thickness $\leqslant 2$. □

Theorem 3.4. HAMILTONIAN PATH CONSTRUCTION GAME *with specified starting vertex is* PSPACE-*complete.*

Proof. Clearly, the problem is solvable in polynomial space. To prove PSPACE-hardness, we use a transformation from QUANTIFIED 3-SATISFIABILITY. Let an instance of this problem be given. Without loss of generality, we may suppose that it is of the form

$$F = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n F_0,$$

with F_0 a boolean expression in conjunctive normal form. Let $C = \{c_1, \ldots, c_m\}$ be the set of clauses in F_0. Without loss of generality, we may suppose that $m \geqslant 4$.

Let $G=(V,E)$ be the directed graph defined by

$$V=\{s,t\}\cup\{x_i \mid 1\leqslant i\leqslant n\}\cup\{\overline{x_i} \mid 1\leqslant i\leqslant n\}\cup\{r_1,r_2\}\cup\{c_i \mid 1\leqslant i\leqslant m\},$$

$$E=\{(s,x_1),(s,\overline{x_1}),(t,x_1),(t,\overline{x_1}),(x_n,r_1),(\overline{x_n},r_2)\}\cup\{(c_i,t) \mid 1\leqslant i\leqslant m\}$$

$$\cup\{(r_i,c_j) \mid i=1,2,\ 1\leqslant j\leqslant m\}$$

$$\cup\{(c_i,c_j) \mid i\neq j,\ 1\leqslant i,j\leqslant m\}$$

$$\cup\{(x_i,x_{i+1}) \mid 1\leqslant i<m\}\cup\{(x_i,\overline{x_{i+1}}) \mid 1\leqslant i<m\}$$

$$\cup\{(\overline{x_i},x_{i+1}) \mid 1\leqslant i<m\}\cup\{(\overline{x_i},\overline{x_{i+1}}) \mid 1\leqslant i<m\}$$

$$\cup\{(c,l) \mid l \text{ is a literal, appearing in clause } c\}$$

(see Fig. 3). □

Claim 3.5. *There is a winning strategy for player* 1 *in the* HAMILTONIAN CIRCUIT CONSTRUCTION GAME *on G with starting vertex s if and only if F is false.*

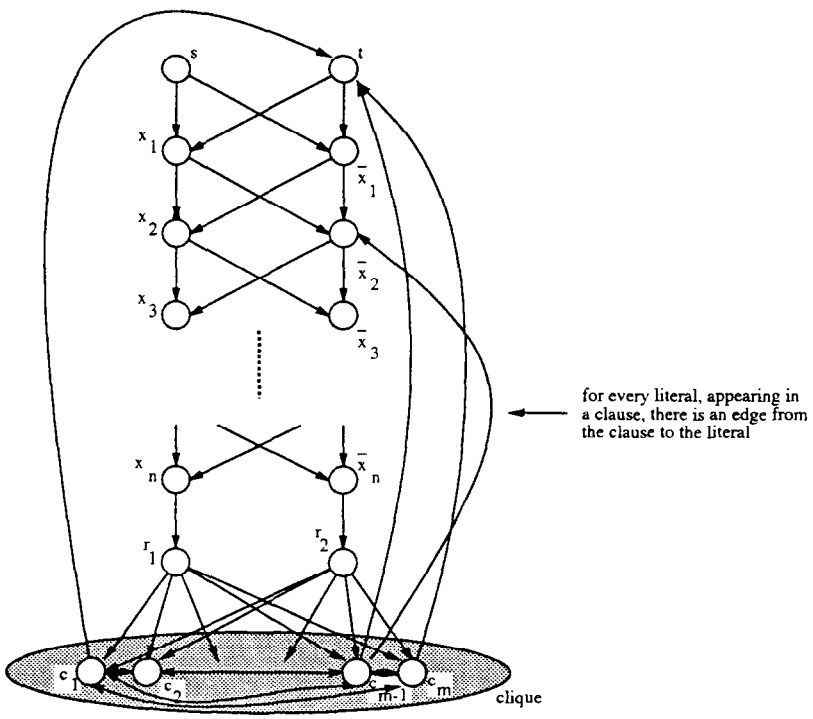

Fig. 3.

Proof. Suppose F is false. We give a winning strategy for player 1. Player 2 will choose x_1 or $\bar{x}_1$, then player 1 chooses x_2 or $\bar{x}_2$, etc. Call vertices which are chosen (before the negation is chosen) *false*. As F is false, player 1 can choose the x_i's (i even) in such a way that, whatever strategy player 2 will use, there will be at least one clause with only false variables, say clause c_{i_0}. Now, after the moment that one vertex of each pair $x_i, \bar{x}_i$ has been visited, player 2 will visit r_1 or r_2. Then player 1 goes to c_{i_0}. Player 2 must go either to t, or to another c_i. In the latter case, player 1 moves to t. Now from t all vertices $x_i, \bar{x}_i$ that have not yet been visited will be visited, then the vertex in $\{r_1, r_2\}$ that has not yet been visited, and then all unvisited vertices c_i. So, player 1 wins the game, as all vertices will eventually be visited.

Suppose F is true. Now player 2 can force that at the first moment a vertex c_{i_0} is visited, each clause contains at least one literal, corresponding to a vertex that is not already visited. Player 1 has moved to c_{i_0}, and then player 2 moves to such a vertex x_i or $\bar{x}_i$. Now all yet unvisited vertices $x_j, \bar{x}_j$ with $j > i$ are visited, then the remaining vertex in $\{r_1, r_2\}$, and then a player moves to a vertex c_i. If player 2 now may move, he moves to t. If player 1 now may move, and moves to another c_i, then player 2 moves from this c_i to t. Now the game will stop before all vertices c_i have been visited. (At most 3 c_i's are visited and $m \geqslant 4$.) So, player 2 wins the game. □

As the construction of G can be done in logarithmic working space, Theorem 3.6 follows.

Theorem 3.6. HAMILTONIAN CIRCUIT CONSTRUCTION GAME *with specified starting vertex is* PSPACE-*complete.*

Proof. Use the construction of Theorem 3.4, but add an edge from each c_i to s. □

Theorem 3.7. HAMILTONIAN PATH CONSTRUCTION GAME *without specified starting vertex is* PSPACE-*complete.*

Proof. Look at the proof of Theorem 3.4. Note that player 1 must start in s, because indegree(s) = 0. □

Theorem 3.8. HAMILTONIAN CIRCUIT CONSTRUCTION GAME *without specified starting vertex is* PSPACE-*complete.*

Proof. Use the construction of Theorem 3.6 (i.e., with an edge from each c_i to s). If player 1 starts at s or t, then the game is as with specified starting vertex s. If player 1 starts at a vertex $x_i, \bar{x}_i$, or r_i, the player 2 can win if $m \geqslant 5$: he always moves to s or t when possible. At least one vertex c_i now will be unvisited at the end of the game. If player 1 starts at a vertex c_j, then player 2 directly moves to a vertex x_i or $\bar{x}_i$.

Hereafter, he moves always to s or t when possible. Now, if $m \geqslant 6$, then at least one vertex c_i will be unvisited at the end of the game. □

Corollary 3.9. SIMPLE PATH CONSTRUCTION GAME *with or without specified starting vertex is* PSPACE-*complete.*

Theorem 3.10. SIMPLE PATH CONSTRUCTION GAME *with specified starting vertex is* PSPACE-*complete for undirected graphs.*

Proof. Clearly, the problem is in PSPACE. We use again a transformation of QUANTIFIED 3-SATISFIABILITY to prove PSPACE-hardness. Let an instance of Q-3-SAT,

$$F = \exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n F_0,$$

be given; F_0 is a boolean formula in conjunctive normal form. Without loss of generality, assume that n is even. Let $C = \{c_1, \ldots, c_m\}$ be the set of clauses in F_0. Assume that $\forall x_i$: $\exists c \in C$: $x_i \in c$; $\exists c \in C$: $\bar{x}_i \in C$. Let $G = (V, E)$ be the following graph:

$$\begin{aligned}
V = {} & \{v_i \mid 1 \leqslant i \leqslant n\} \\
& \cup \{x_i \mid 1 \leqslant i \leqslant n\} \cup \{\bar{x}_i \mid 1 \leqslant i \leqslant n\} \\
& \cup \{y_i \mid 1 \leqslant i \leqslant n\} \cup \{\bar{y}_i \mid 1 \leqslant i \leqslant n\} \\
& \cup \{w_i \mid 1 \leqslant i \leqslant n\} \cup \{z_i \mid 1 \leqslant i \leqslant n\} \\
& \cup \{c_i \mid 1 \leqslant i \leqslant m\} \\
& \cup \{d_{ijk} \mid 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant 3,\ 1 \leqslant k \leqslant K\}, \\
E = {} & \{(v_i, x_i), (v_i, \bar{x}_i), (x_i, y_i), (\bar{x}_i, \bar{y}_i), (y_i, w_i), (\bar{y}_i, w_i) \mid 1 \leqslant i \leqslant n,\ i \text{ odd}\} \\
& \cup \{(v_i, y_i), (v_i, \bar{y}_i), (y_i, x_i), (\bar{y}_i, \bar{x}_i), (x_i, w_i), (y_i, w_i) \mid 1 \leqslant i \leqslant n,\ i \text{ even}\} \\
& \cup \{(w_i, z_i) \mid 1 \leqslant i \leqslant n\} \\
& \cup \{(z_i, v_{i+1}) \mid 1 \leqslant i < n\} \\
& \cup \{(z_n, c_i) \mid 1 \leqslant i \leqslant m\} \\
& \cup \{(c_i, d_{ij1}) \mid 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant 3\} \\
& \cup \{(d_{ijk}, d_{ij(k+1)}) \mid 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant 3,\ 1 \leqslant k \leqslant K\} \\
& \cup \{(d_{ij1}, l) \mid l \text{ is the } j\text{th literal appearing in clause } c_i\},
\end{aligned}$$

where $K = 5n + 8$ (see Fig. 4). The starting vertex is v_1.

Claim 3.11. *F is true if and only if player 1 can force a path with length $\geqslant K$.*

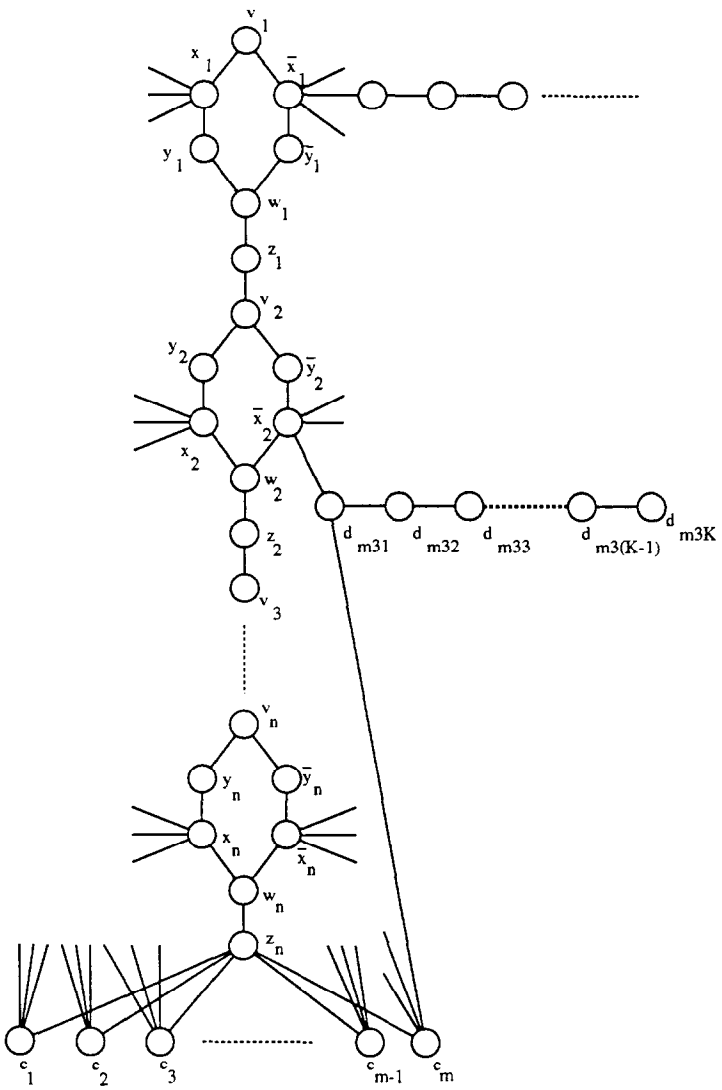

Fig. 4.

Proof. Note that the resulting path will have length $\geqslant K$ if and only if a "long branch" $d_{ij1}d_{ij2}\ldots d_{ijK}$ is used. (Every other path in G has length $<K$.) Next note that that player who must move from a vertex w_i must move to z_i or lose the game: if player 1 must move from w_i, then i is even. If he moves to x_i or $\bar{x}_i$, then player 2 moves to y_2 or $\bar{y}_2$, and the game ends with a path shorter than K. If player 2 must move from w_i, then i is odd. If he moves to y_i or $\bar{y}_i$, then player 1 moves to x_i or $\bar{x}_i$; player 2 then must move to a vertex d_{ij1}, and then player 1 moves to d_{ij2} and forces a path with length $\geqslant K$.

The proof proceeds with arguments which are similar to arguments used before. Let a *used* x_i or $\bar{x}_i$ correspond with *true*. Player 1 tries to have in each clause a literal l, with the corresponding vertex l is being visited when the path reaches z_n. He succeeds if and only if the formula is true. Player 2 must move from z_n. He will move, if possible, to a unsatisfied clause, i.e., each literal in the clause is unvisited. Player 1 will move to a vertex d_{ij1}. If the corresponding literal in the clause is true (visited), then player 2 must move to d_{ij2}, and the resulting path has length $\geqslant K$. Otherwise, player 2 can move to that literal vertex, and the resulting path has length $< K$. □

We have obtained a log-space transformation from QUANTIFIED 3-SATISFIABILITY to SIMPLE PATH CONSTRUCTION GAME for undirected graphs. Hence, the latter is PSPACE-complete.

Theorem 3.12. SIMPLE PATH CONSTRUCTION GAME *without specified starting vertex is* PSPACE-*complete for undirected graphs.*

Proof. To prove PSPACE-hardness, we use a transformation from the case with specified starting vertex. Let an instance $G=(V,E)$, $s\in V$, $K\in\mathbb{N}^+$ of the latter problem be given. Let $G'=(V',E')$ be defined as follows (see Fig. 5):

$$V' = V\cup\{r_i \mid 1\leqslant i\leqslant 2\cdot|V|+2\}$$
$$\cup\{q_i \mid 1\leqslant i\leqslant K+2\}\cup\{y\},$$
$$E' = E\cup\{(r_i, r_{i+1}) \mid 1\leqslant i\leqslant 2|V|\}$$
$$\cup\{(q_i, q_{i+1}) \mid 1\leqslant i\leqslant K+1\}$$
$$\cup\{(r_{2|V|+2}, s), (r_{2|V|+1}, q_1), (r_{2|V|}, y)\}.$$

Claim 3.13. *Player* 1 *can force a path with length* $\geqslant K+2|V|+2$ *on* G' *with no specified starting vertex if and only if player* 1 *can force a path with length* $\geqslant K$ *on* G *with starting vertex* s.

Proof. $\Leftarrow$: Player 1 starts at r_1. When player 2 must move eventually from $r_{2|V|+1}$, he can go to q_1 (in which case the resulting path has the required length), or go to $r_{2|V|+2}$. In the latter case, player 1 now can use the strategy for the game on G with starting vertex s.

$\Rightarrow$: We consider several cases for the start of player 1 on G'.

Case 1. Player 1 *starts at a vertex* q_i: Then he will lose: either the path ends at q_{K+2} (with length $\leqslant K+2$), or a player moves to $r_{2|V|+1}$. If player 1 moves from $r_{2|V|+1}$ to $r_{2|V|}$, player 2 moves to y and wins. If any player moves from $r_{2|V|+1}$ to $r_{2|V|+2}$, the resulting path will have length $\leqslant K+4+|V|$.

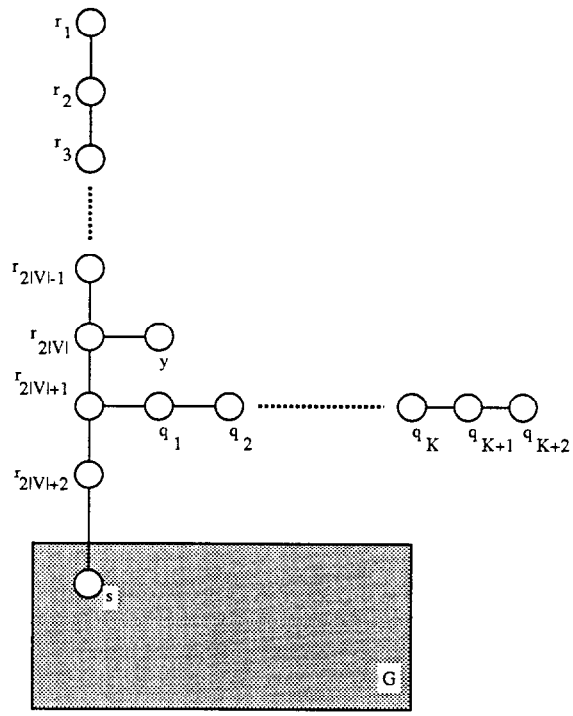

Fig. 5.

Case 2. Player 1 *starts at a vertex* r_i, $i \neq 1$: If i is odd, player 2 will eventually move to q_1; if i is even, then player 2 will eventually move to y. In both cases, the resulting path has length $\leqslant K + 2|V|$.

Case 3. Player 1 *starts at* y: No matter what strategy is used by either player, the resulting path will have length $\leqslant 2|V| + 1$.

Case 4. Player 1 *starts at a vertex* $v \in V$: After at most $|V|$ moves, a player will move to $r_{2|V|+2}$, or the resulting path has length $\leqslant |V|$. Player 2 uses the strategy to move to q_1 or y if possible. The resulting path will have length $\leqslant |V| + 2 + K + 2$.

Case 5. Player 1 *starts at* r_1: If player 2 moves from $r_{2|V|+1}$ to q_1, then player 1 succeeds. If player 2 moves from $r_{2|V|+1}$ to $r_{2|V|+2}$, then player 1 succeeds exactly if he can force a path with length K in G with starting vertex s. □

Again, this transformation can be done in logarithmic working space.

Theorem 3.14. ELEMENTARY PATH CONSTRUCTION GAME (*with specified starting vertex*) *is* PSPACE-*complete.*

Proof. The proof is similar to the proof of Theorem 3.10. A variable x_i with i odd is replaced by the construction of Fig. 6a, and a variable with i even will be replaced by

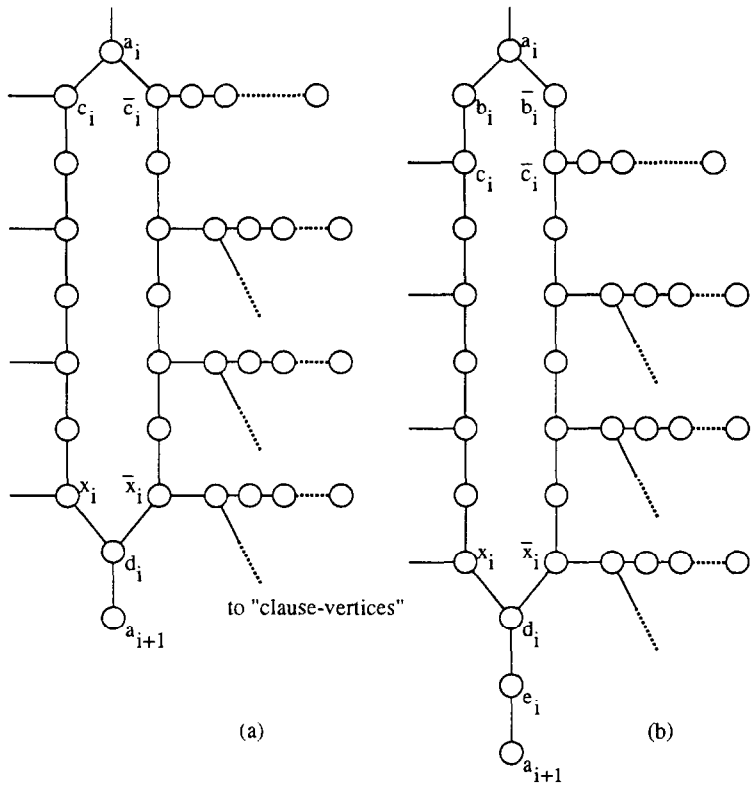

Fig. 6.

the construction of Fig. 6b. In the "first pass", players will move from a_i to d_i and from d_i to a_{i+1}. Player 2 will never take a "side-branch", because then he loses the game. Player 1 will never move from d_i to x_i or $\bar{x}_i$, because then player 2 will move such that the game stops after 8 or 9 moves in a_i. An unvisited x_i corresponds to true. Player 1 can move to a vertex on a branch before an unvisited x_i if and only if the formula is true. If x_i is visited, then player 2 moves from the vertex on the branch to x_i, and the game stops. Otherwise, player 1 can move such that he can go from c_i to the branch with length K, attached to c_i. We omit the details. □

Theorem 3.15. ELEMENTARY PATH CONSTRUCTION GAME *without specified starting vertex is* PSPACE-*complete.*

Proof. This follows with a construction similar to the construction in the proof of Theorem 3.12. □

We end this section with a small comment on the standard VERTEX and EDGE GENERALIZED GEOGRAPHY GAMES. Clearly, when we restrict ourselves to *acyclic graphs*,

then the problems are easy to resolve in $\mathcal{O}(n+e)$ time. However, the proof in [35, 43] for the PSPACE-completeness of VERTEX OR EDGE GENERALIZED GEOGRAPHY can easily be modified, such that we have PSPACE-completeness for the problems on graphs obtained by *adding one edge to an acyclic graph.*

4. Polynomial-time algorithms for path-forming games on special classes of graphs

In this section we give polynomial-time algorithms for several of the considered games on special classes of graphs. In Section 4.1 we give linear algorithms for some of the games on graphs with bounded treewidth, based on an intricate characterization of subgraphs, and dynamic programming. In Section 4.2 we show how graph rewriting can be employed to solve some problems on cacti.

Note that each of the four problems VERTEX GENERALIZED GEOGRAPHY, EDGE GENERALIZED GEOGRAPHY, HAMILTONIAN PATH CONSTRUCTION GAME, and HAMILTONIAN CIRCUIT CONSTRUCTION GAME can be easily solved in $\mathcal{O}(n+e)$ time when restricted to directed acyclic graphs. See also [27].

4.1. Linear-time algorithms for some games on graphs with bounded treewidth

In this section we show how VERTEX GENERALIZED GEOGRAPHY, HAMILTONIAN PATH CONSTRUCTION GAME, and HAMILTONIAN CIRCUIT CONSTRUCTION GAME can be solved in linear time on graphs with a fixed upper bound k on the treewidth. We first consider VERTEX GENERALIZED GEOGRAPHY.

Let in the remainder of this section k be a constant. We will assume that input graphs $G=(V,E)$ are given with a tree decomposition $(\{X_i \mid i\in I\}, T=(I,F))$ of G with treewidth $\leqslant k$. If not, then such a tree decomposition can be found (if it exists) in $\mathcal{O}(n \log n)$ time [17, 38]. For $k=1,2,3$, the tree decomposition can be found in linear time [7, 36]. Other algorithms for this problem can be found in [4, 14, 31, 39].

It is not difficult to see that one may assume that the tree T in the tree decomposition is binary (e.g. use the transformation used in [15]). In the remainder we assume that we have a tree decomposition $(\{X_i \mid i\in I\}, T=(I,F))$ of $G=(V,E)$ with treewidth $\leqslant k$, and T a binary tree. We also suppose that there exists an $i_0\in I$, with $X_{i_0}=\{s\}$. (Take an arbitrary $i\in I$, with $s\in X_i$. Add a branch (i,i_0) to T, with $X_{i_0}=\{s\}$. A correct tree decomposition results. Now apply the technique to make T a binary tree.) i_0 is taken as root of T.

Our algorithm is based on dynamic programming. The idea is the following: for each $i\in I$, the algorithm will compute a "table" containing the information charac(X_i, Y_i), where $Y_i=\{v\in X_j \mid j$ is a descendant of $i\}-X_i$. These tables can be computed from the tables of the children in $\mathcal{O}(1)$ time each. From charac(X_{i_0}, Y_{i_0})=charac$(\{s\}, V-\{s\})$ the answer to the problem can be determined quickly.

The characteristic of a pair (X, W) is in terms of "subgames": games that start in some vertex $x \in X$, and are played as VERTEX GENERALIZED GEOGRAPHY on the graph $G[X \cup W]$, but with this difference: the game not only ends when a player cannot make a move, but also ends when a player moves to a vertex $y \in X - \{x\}$. Depending upon this y and the characteristic of the subgraph induced by the unvisited vertices in $X \cup W$, this player wins or loses this subgame. The characteristic of (X, W) basically denotes, for each possible subgame, which player has a winning strategy.

We now give a more precise, inductive definition of charac(X, W), for $X, W \subseteq V$, $X \cap W = \emptyset$. charac(X, W) is defined inductively on $|X|$.

For this inductive definition, we need the following notation: for sets X, $C(X)$ denotes the set of all possible values of charac(X, W) over all graphs $G = (V, E)$, $X, W \subseteq V$, $X \cap W = \emptyset$.

If $X = \emptyset$, then charac(X, W) is the empty string.

If $|X| = 1$, then charac(X, W) is a boolean $\in \{\text{true}, \text{false}\}$, that denotes whether there is a winning strategy for player 2 for VERTEX GENERALIZED GEOGRAPHY played on $G[X \cup W]$, the subgraph of G induced by $X \cup W$, with starting vertex the unique vertex $x \in X$. The first move of player 1 is to move to x, i.e. player 2 is the first player who actually can make a choice in the game (when the degree of x is larger than one.)

Now suppose $|X| \geq 2$. We first must introduce some other notions. Let $x \in X$. Consider the following type of variant of VERTEX GENERALIZED GEOGRAPHY: the game ends when a player moves to a vertex $y \in X$, or when a player cannot make a move. In the former case, let $W' \subseteq W$ be the set of vertices in W not yet visited, and consider charac$(X - \{x, y\}, W') \in C(X - \{x, y\})$. Let $P(X, W, x, y)$ be the set of all pairs (p, c), with $p \in \{1, 2\}$ denoting a player and $c \in C(X - \{x, y\})$, such that there is a possible play by players 1 and 2 in the above type of game, starting at x, where player p moves to y, and $c = \text{charac}(X - \{x, y\}, W')$, with W' the set of vertices in W that are not visited in the game. Let $P(X, W, x)$ be the set of all triples (p, c, y), with $y \in X - \{x\}$ and $(p, c) \in P(X, W, x, y)$. We call such a triple (p, c, y) an *outcome*. A set $R \subseteq P(X, W, x)$ is called a *set of outcomes*. For each set of outcomes $R \subseteq P(X, W, x)$, we now consider the game VGG(X, W, x, R). This is a variant of VERTEX GENERALIZED GEOGRAPHY described above, with the following properties: player 1 starts with a move from x to a vertex in $W \cup X$. The game ends when a player cannot make a move from a vertex in W – then this player loses the game – or when a player j moves to a vertex $y \in X$, where W' is the set of vertices in W that are not visited. In this case, player 1 wins the game if and only if $(j, \text{charac}(X - \{x, y\}, W'), y) \in R$. We say that the game has outcome $(j, \text{charac}(X - \{x, y\}, W'), y)$. In other words, player 1 wins if player 2 loses by not being able to move, or when the game ends with an outcome in the set of outcomes R.

We can now describe charac(X, W) for $|X| \geq 2$. charac(X, W) is a pair (f_1, f_2), where

- f_1 maps each pair (x, R), with $R \subseteq \bigcup_{y \in X - \{x\}} \{(p, c, y) \mid p \in \{1, 2\}, c \in C(X - \{x, y\})\}$, to a boolean that is true if and only if $R \subseteq P(X, W, x)$ and there is a winning strategy for player 1 in the game VGG(X, W, x, R),
- f_2 maps each $x \in X$ to charac$(X - \{x\}, W)$.

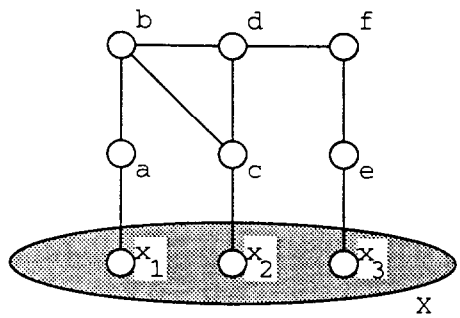

Fig. 7.

(The case with $|X|=1$ can be seen as a special case of the above definition.)

As an example, consider the graph $G=(W\cup X, E)$, shown in Fig. 7. $X=\{x_1, x_2, x_3\}$, $W=\{a, b, c, d, e, f\}$. Here $P(X, W, x_1, x_2)=\{(1, \text{true}), (2, \text{false})\}$, because (x_1, a, b, c, x_2) is a play, where player 1 moves to x_2, and the resulting graph (with vertices d, f, e, x_3) gives a winning strategy for player 2, and (x_1, a, b, d, c, x_2) is a play where player 2 moves to x_2, and the resulting graph gives a winning strategy for player 1. Similarly, $P(X, W, x_1, x_3)=\{(1, \text{true}), (2, \text{false})\}$. So, $P(X, W, x_1)=\{(1, \text{true}, x_2), (2, \text{false}, x_2), (1, \text{true}, x_3), (2, \text{false}, x_3)\}$. For each subset R of $P(X, W, x_1)$, we can consider the game $\text{VGG}(X, W, x_1, R)$. Consider $R=\{(1, \text{true}, x_2), (2, \text{false}, x_2)\}$. Player 1 will win the game $\text{VGG}(X, W, x_1, R)$: player 2 must move from b. If he moves to d, player 1 moves to c, and the game ends with situation $(2, \text{false}, x_2)$. Otherwise, player 1 moves from c to x_2 and the game ends with situation $(1, \text{true}, x_2)$. As these are in R, player 1 wins. So, f_1 in $\text{charac}(X, W)$ has $f_1(x_1, R)=\text{true}$. In total, f_1 can be specified here with $3\cdot 2^4$ bits plus the space to write down all three sets $P(W, X, x_i)$. f_2 contains the information $\text{charac}(X-\{x_1\}, W)$, $\text{charac}(X-\{x_2\}, W)$ and $\text{charac}(X-\{x_3\}, W)$.

Note that if $|X|$ is bounded by some constant c, then the number of bits needed to denote $\text{charac}(X, W)$ is also bounded by some constant c'. If S_l denotes this number for $|X|=l$, then $S_l \leqslant l\cdot 2^{(l-1)S_{l-2}}+l\cdot S_{l-1}$.

For $i\in I$, let $Y_i=\{v\in X_j \mid j \text{ is a descendant of } i \text{ in } T\}-X_i$. Our algorithm is based on computing $\text{charac}(X_i, Y_i)$ for all $i\in I$.

Lemma 4.1. *Let $i\in I$ be a leaf of T. Then $Y_i=\emptyset$, and $\text{charac}(X_i, Y_i)$ can be computed in $\mathcal{O}(1)$ time.*

Proof. Clearly, $Y_i=\emptyset$. Note that $|X_i|\leqslant k+1=\mathcal{O}(1)$. □

Lemma 4.2. *There is a winning strategy for player 1 for* VERTEX GENERALIZED GEOGRAPHY *if and only if $\text{charac}(X_{i_0}, Y_{i_0})=\text{false}$.*

Proof. $\text{charac}(X_{i_0}, Y_{i_0})=\text{charac}(\{s\}, V-\{s\})$ denotes whether there is a winning strategy for player 2 for VERTEX GENERALIZED GEOGRAPHY played on $G[\{s\}\cup(V-\{s\})]=G$ with starting vertex s. □

Lemma 4.3. Let $i \in I$ be an internal node of T, and let j_1 and j_2 be the two children of i. Let charac(X_{j_1}, Y_{j_1}), charac(X_{j_2}, Y_{j_2}) be given. Then charac(X_i, Y_i) can be computed in $\mathcal{O}(1)$ time.

Proof. We will show how, for all $x \in X_i$ and all $R \subseteq P(X_i, Y_i, x)$, we can determine which player has a winning strategy in the game VGG(X_i, Y_i, x, R). We model all possible (optimal) plays in this game by a rooted labeled tree, as described below.

Nodes in this tree are of two types. The nodes of the first type are labeled with 6-tuples of the form $(v, p, Z_1, Z_2, c_1, c_2)$, where v is a vertex in $X_{j_1} \cup X_{j_2} \cup X_i$, p denotes a player ($p \in \{1, 2\}$), c_1 is a characteristic in $\bigcup_{Y \subseteq X_{j_1}} C(Y)$, and c_2 is a characteristic in $\bigcup_{Y \subseteq X_{j_2}} C(Y)$.

The tuple $(v, p, Z_1, Z_2, c_1, c_2)$ represents an (equivalence) class of positions during the game. Each of these positions is of the following type: The player $3-p$ just has moved to a vertex v; hence, player p must now move to an unvisited vertex adjacent to v. Z_1 is the set of unvisited vertices in X_{j_1}. If W_1 is the set of unvisited vertices in Y_{j_1}, then $c_1 = \text{charac}(Z_1, W_1)$. Z_2 and c_2 are defined similarly using X_{j_2} and Y_{j_2}.

Nodes of the second type are labeled with 6-tuples of the form $(\alpha, p', R', Z_1', Z_2', c)$, with $\alpha \in \{1, 2\}$ denoting either the left or the right child of i in T, $p' \in \{1, 2\}$ denoting a player. The father of a node with such a label will be a node of type one. If this father has the label $(v, p, Z_1, Z_2, c_1, c_2)$, then $v \in Z_\alpha$, $R' \subseteq P(Z_\alpha, W_\alpha, v)$ and there is a winning strategy for player 2 in the game VGG$(Z_\alpha, W_\alpha, v, R')$ for W_α, with charac$(Z_\alpha, W_\alpha) = c_\alpha$. (Alternatively, one can take W_α to be the set of unvisited vertices in Y_{i_α}.) Further, $Z_1' = Z_1 - \{v\}$ is the set of unvisited vertices in X_{j_1} and $Z_2' = Z_2 - \{v\}$ is the set of unvisited vertices in X_{j_2}. $c = \text{charac}(Z_{3-\alpha}', W_{3-\alpha})$, with $W_{3-\alpha}$ the set of unvisited vertices in $Y_{j_{3-\alpha}}$. Nodes of this second type represent a certain subgame, described in more detail below.

Call a node labeled $(v, p, \ldots)$ or $(\alpha, p, \ldots)$ a *player p node*. The children of a player p node denote the situations where player p can move to from the situation represented by the player p node. These situations can either be a position in the game, or player p can decide to play a "subgame".

From a position in the game, represented by a node of type one labeled with $(v, p, Z_1, Z_2, c_1, c_2)$, player p has the following possibilities: he can move either to a vertex $w \in X_i - \{x\} - X_{j_1} - X_{j_2}$, with $(v, w) \in E$ (this vertex w cannot be visited earlier), or to an unvisited vertex in $Z_1 - \{v\} \cup Y_{j_1}$, or to an unvisited vertex in $Z_2 - \{v\} \cup Y_{j_2}$. In the second and third cases, players will visit zero or more vertices in Y_{j_1} (Y_{j_2}), and then a vertex in Z_1 (Z_2), or the game will halt because a player cannot move anymore. These situations can be represented by subgames of the VGG-type. Let W_α denote the unvisited vertices in Y_{i_α} ($\alpha = 1, 2$). The game that represents the situation is a game VGG$(Z_\alpha, W_\alpha, v, R')$, with $R' \subseteq P(Z_\alpha, Y_{i_\alpha}, v)$. Note that player p acts as player 2 in such a subgame. If VGG$(Z_\alpha, W_\alpha, v, P(Z_\alpha, Y_{i_\alpha}))$ is winning for player 2, then in this subgame, player 2 has a strategy that wins without reaching an outcome, i.e., player 1 loses by being unable to move. In this case, player p, who acts as player 2 in the subgame, has a winning strategy. If VGG$(Z_\alpha, W_\alpha, v, \emptyset))$ is winning for player 1, then player 1 can win

the subgame without reaching an outcome. In this case, player p will lose when he moves to a vertex in $Z_\alpha \cup Y_{i_\alpha}$. In general, player p can decide to play a subgame $\mathrm{VGG}(Z_\alpha, W_\alpha, v, R')$ with R' chosen such that this subgame is winning for player 2. In this way, he can force that this subgame ends with an outcome in R'. When player p decides to play this subgame, player $3-p$ can choose an outcome from R' (if $R' \neq \emptyset$), and force that this outcome is the actual outcome of the subgame. (If $p=1$, the roles of players 1 and 2 get switched.)

We remark that it is not necessary to know W_1 and W_2: at all times, all relevant information of these sets is contained in the characteristics.

Now we describe how the tree T is built. The root of this tree is a node of type one, with label $(x, 2, X_{j_1}, X_{j_2}, \mathrm{charac}(X_{j_1}, Y_{j_1}), \mathrm{charac}(X_{j_2}, Y_{j_2}))$. This tuple indeed represents the situation as it is in the beginning of the game $\mathrm{VGG}(X_i, Y_i, x, R)$.

A node of type one, labeled with $(v, p, Z_1, Z_2, c_1, c_2)$, with $v \in X_i - \{x\}$, has no children. This is so because, when such a vertex v is reached, the game has ended.

A node of type one, labeled with $(v, p, Z_1, Z_2, c_1, c_2)$, with $v \notin X_i - \{x\}$, has the following children. (Note that the three cases represent the three different possibilities for player p, described above.)

(1) For all $w \in X_i - X_{j_1} - X_{j_2} - \{x\}$, with $(v, w) \in E$, take a child node of type one, labeled with $(w, 3-p, Z_1 - \{v\}, Z_2 - \{v\}, c_1', c_2')$. If $v \in Z_1$, then suppose $c_1 = (f_1, f_2)$, and take $c_1' = f_2(v)$. If $v \notin Z_1$, then take $c_1' = c_1$. We have that if $c_1 = \mathrm{charac}(Z_1, W_1)$, then $c_1' = \mathrm{charac}(Z_1 - \{v\}, W_1)$. Define c_2' in the same way.

(2) If $v \in Z_1$ then, for any R', with $c_1 = (f_1, f_2)$ and $f_1(v, R') = \mathit{false}$ (i.e., there is a winning strategy for player 2 in the subgame $\mathrm{VGG}(Z_1, W_1, v, R)$), take a child node of type two, labeled with $(1, 3-p, R', Z_1 - \{v\}, Z_2 - \{v\}, c)$. Suppose $c_2 = (g_1, g_2)$. Take if $v \in Z_2$, $c = g_2(v)$, and if $v \notin Z_2$, take $c = c_2$. This case represents the decision of player p to play the subgame $\mathrm{VGG}(Z_1, W_1, v, R')$.

(3) If $v \in Z_2$, we take in a similar way as in the previous case child nodes of type two, labeled with $(2, 3-p, R', Z_1 - \{v\}, Z_2 - \{v\}, c)$, where c is derived from c_1 similar as above.

Next we describe the children of nodes of type two, labeled with $(\alpha, p, R', Z_1, Z_2, c)$. We suppose that $\alpha = 1$. The case $\alpha = 2$ is similar. Take, for every $(q, c', z) \in R'$, a child node of type one, labeled with $(z, 3-q, Z_1, Z_2, c', c)$ if $p=1$, and labeled with (z, q, Z_1, Z_2, c', c) if $p=2$. When the outcome of the subgame $\mathrm{VGG}(Z_1, W_1, v, R')$ is (q, c', z), then in this subgame, player q moves to z, leaving a graph with characteristic c. As player p has the role of player 1 in this subgame, it follows that if $p=1$, then in the game $\mathrm{VGG}(X_i, Y_i, v, R)$ player q moves to z and, hence, player $3-q$ must move from z. If $p=2$, then in the game $\mathrm{VGG}(X_i, Y_i, v, R)$, player $3-q$ moves to z; hence, player q must move from z. If W_α' is the set of unvisited vertices in Y_{j_α} after the move to vertex z, then we have that $c = \mathrm{charac}(Z_2, W_2')$ and $c' = \mathrm{charac}(Z_1, W_1')$, as $W_2 = W_2'$ and by choice of outcome.

We now show how to compute for which player there is a winning strategy in $\mathrm{VGG}(X_i, Y_i, x, R)$. For each tree node, we determine whether it is *winning for player* 1, or *winning for player* 2. This is done bottom up in the tree T, starting at leaf nodes. The

game VGG(X_i, Y_i, x, R) is winning for player 1 if and only if the root node of T is winning for player 1.

For leaf nodes, there are two cases:

(1) A player p node that is a leaf of T and of type two, or of type one of the form $(v, p, \ldots)$, with $v \notin X_i - \{x\}$, is losing for player p and, hence, winning for player $3-p$. (Player p must move, as the game has not ended yet, but has no winning move available.)

(2) For player p nodes that are a leaf of T and of type one, of the form $(v, p, Z_1, Z_2, c_1, c_2)$, with $v \in X_i$, one can determine in the following way whether this node is winning for player 1 or player 2. The characteristic charac($X_i - \{x, v\}, U$), with U the set of unvisited vertices in $Y_{j_1} \cup Y_{j_2}$, is uniquely determined. It is an element of $C(X_i, \{x, v\})$, and can be determined recursively with the procedure described in this proof. Now we have instead of X_i the set $X_i - \{x, v\}$, which is of smaller size. Hence, recursion depth is $O(k)$. Instead of X_{j_1} and X_{j_2}, we have sets Z_1 and Z_2, and instead of charac(X_{j_1}, Y_{j_1}) and charac(X_{j_2}) and charac(Y_{j_2}), we have c_1 and c_2 here. Suppose the computed characteristic is $c = \text{charac}(X_i - \{u, v\}, U)$. It follows that the outcome corresponding to the leaf node is $(v, 3-p, c)$. (As player p must move from v, player $3-p$ has moved to v.) Hence, the node is winning for player one if this outcome is in the set of outcomes R, i.e., $(v, 3-p, c) \in R$.

An internal player p node is winning for player p if and only if at least one child of the node is winning for player p. (p player p can choose a situation, represented by one of the children of the node. He can win if he can choose such a situation that is winning for him.)

In this way, one can determine whether the root of the tree is winning for player 1, i.e., whether there is a winning strategy for player 1 in VGG(X_i, Y_i, x, R).

All information needed for charac(X_i, Y_i) can be determined in this way. As only charac(X_{j_1}), charac(X_{j_2}), and the structure of X_i, X_{j_1}, X_{j_2} and edges between these vertices are consulted, the procedure uses constant time. □

Now we are able to derive the main result of this subsection. Note that in the $\mathcal{O}$-notation, a large constant factor depending on k is hidden.

Theorem 4.4. *For every constant $k \geqslant 1$,* VERTEX GENERALIZED GEOGRAPHY *(with specified starting vertex) can be solved in $\mathcal{O}(n)$ time for graphs $G = (V, E)$ with treewidth $\leqslant k$, that are given together with a tree decomposition with treewidth $\leqslant k$.*

Proof. Compute, for every $i \in I$, charac(X_i, Y_i). This is done by starting at leaf nodes (Lemma 4.1), and then repeatedly computing charac(X_i, Y_i) when this has been computed for both children of i (Lemma 4.3). When charac(X_{i_0}, Y_{i_0}) has been determined, the answer to the problem can be given (Lemma 4.2). As per node $i \in I$, only constant time is used; this takes in total $\mathcal{O}(|I|) = \mathcal{O}(n)$ time. □

Note that the constant factor in the algorithm grows very fast with k. The following bound is known on the number of bits S_l needed to denote charac(X, W), with $|W|=l$: $S_l \leqslant l \cdot 2^{(l-1)S_{l-2}} + l \cdot S_{l-1}$. One can observe that the time to compute one value of charac(X_i, W_i), when these values are known for the children of node $i \in I$, is polynomial in the number of bits in these charac-strings and, hence, polynomial in S_{k+1}. So, adding 2 to k gives one extra level of exponentiation in the constant factor hidden in the $\mathcal{O}$-notation, i.e., the constant factor grows faster than exponentially in k. Thus, our algorithm will only be practical for very small values of k, probably only for $k=1,2,3$. However, a technique is known to optimize algorithms that work on tree decompositions. This technique, which basically is a tree variant of the well known Myhill–Nerode result on finite-state automata, is advocated, among others by Fellows and Langston [23] (see also [9]). It would carry too far to explain here more about this method. We expect that with the help of this Myhill–Nerode technique, it is possible to construct practical algorithms for several small values of k. On the other hand, it may be true that a constant factor that is superexponential in k is unavoidable. Algorithms that solve in linear time problems which are NP-complete for arbitrary graphs on tree decompositions with treewidth at most a constant k have a constant factor that is exponential in k. It is not surprising when similar algorithms for PSPACE-complete problems are more complicated and much slower.

The algorithm can be modified in order to obtain similar results for related games. Note that, for each of the algorithms mentioned below in this subsection, the same comments about the constant factor hidden in the '$\mathcal{O}$' notation apply as for the algorithm of Theorem 4.4.

Theorem 4.5. *For every constant $k \geqslant 1$,* VERTEX GENERALIZED GEOGRAPHY *without specified starting vertex can be solved in $\mathcal{O}(n)$ time for graphs $G=(V,E)$ with treewidth $\leqslant k$, that are given together with a tree decomposition with treewidth $\leqslant k$.*

Proof. VERTEX-GENERALIZED GEOGRAPHY without starting vertex on $G=(V,E)$ is equivalent to VERTEX GENERALIZED GEOGRAPHY on $G'=(V \cup \{v^+, v^{++}\}, E \cup \{(v^+, v^{++})\} \cup \{(v^{++}, w) \mid w \in V\})$ with starting vertex v^+. It is easy to make a tree decomposition of G' with treewidth $\leqslant k+1$, given the tree decomposition of G. □

Theorem 4.6. *For every constant $k \geqslant 1$,* HAMILTONIAN CIRCUIT CONSTRUCTION GAME *and* HAMILTONIAN PATH CONSTRUCTION GAME *can be solved in $\mathcal{O}(n)$ time for graphs $G=(V,E)$ with treewidth $\leqslant k$, that are given together with a tree decomposition with treewidth $\leqslant k$.*

Proof. This is done with a method similar to VERTEX GENERALIZED GEOGRAPHY. Basically, one must change the charac-functions a little, and incorporate in functions charac(X_i, W_i) whether $W_i = \emptyset$. We omit the details. □

It is not clear whether the other problems that are considered in this paper can be solved in polynomial time on graphs with bounded treewidth. For the SIMPLE (ELEMENTARY) PATH CONSTRUCTION GAME, it seems that incorporating the length of the paths in

the characteristics will give rise to characteristics of non-polynomial size. For EDGE GENERALIZED GEOMETRY, the size of the characteristics in our type of scheme becomes exponential, because a vertex v can be visited $\mathcal{O}(\text{degree}(v))$ times, which can be linear. This problem disappears when we assume a fixed upper bound on the degree of the vertices. We use the following lemma.

Lemma 4.7. *Let $G=(V,E)$ be a graph with treewidth $\leqslant k$ and maximum vertex degree $\leqslant d$. Then the treewidth of the edge graph of G is at most $(k+1)d-1$.*

Proof. Consider a tree decomposition $(\{X_i \mid i\in I\}, T=(I,F))$ of G with treewidth $\leqslant k$. Take $Y_i=\{(v,w)\in E \mid v\in X_i \vee w\in X_i\}$. For every pair of edges $(v,w),(w,x)\in E$, note that $\exists i$: $(v,w),(w,x)\in Y_i$, namely, take $i\in I$, with $w\in X_i$. Also note that the set of nodes $i\in I$, with $(v,w)\in Y_i$, is the union of the subtree of T $\{i\in I \mid v\in X_i\}$ and the subtree of T $\{i\in I \mid w\in X_i\}$. As $\exists i$: $v,w\in X_i$, these subtrees are not disjoint; hence, their union is a connected subtree of T. It follows that $(\{Y_i \mid i\in I\}, T=(I,F))$ is a tree decomposition of the edge graph of G. Clearly, $\forall i\in I$: $|Y_i|\leqslant d\cdot|X_i|\leqslant(k+1)d$. □

Theorem 4.8. *For every constant $k\geqslant 1$, $d\geqslant 1$:* EDGE GENERALIZED GEOGRAPHY *can be solved in $\mathcal{O}(n)$ time for graphs $G=(V,E)$ with treewidth $\leqslant k$ and maximum vertex degree $\leqslant d$, that are given together with a tree decomposition with treewidth $\leqslant k$.*

Proof. EDGE GENERALIZED GEOGRAPHY on G with starting vertex v is equivalent to VERTEX GENERALIZED GEOGRAPHY on the edge graph of G, with starting vertex one of the $\leqslant d$ vertices that correspond to an edge with head v. Now use Lemma 4.7 and Theorem 4.4. □

Clearly, Theorems 4.6 and 4.8 hold also for the case without specified starting vertex. We now consider an application to QUANTIFIED SATISFIABILITY.

Definition. Let $F=Q_1x_1Q_2x_2\ldots Q_nx_nE$ be a well-formed quantified boolean expression, where each Q_i is either $\forall$ or $\exists$, and E is a boolean expression in conjunctive normal form. The graph G_F is defined as follows: $G_F=(\{x_1,\ldots,x_n\},E_F)$, with $E_F=\{(x_i,x_{i+1}) \mid 1\leqslant i\leqslant n\}\cup\{(x_i,x_j) \mid$ there exists a clause c in expression E that contains a literal x_i or $\bar{x}_i$, and that contains a literal x_j or $\overline{x_j}\}$.

Corollary 4.9. *One can decide in $\mathcal{O}(n)$ time whether a formula F of the form described above is true, when the treewidth of G_F is bounded by a constant k, and E is given with a tree decomposition of G_F with treewidth $\leqslant k$.*

Proof. Consider the transformation from QUANTIFIED 3-SATISFIABILITY TO VERTEX GENERALIZED GEOGRAPHY, given in [35]. It is not hard to see that if G_F has treewidth $\leqslant k$, then the treewidth of the graph resulting from this transformation has treewidth $\mathcal{O}(k)$, and

that the corresponding tree decomposition can be constructed from the tree decomposition of G_F in $\mathcal{O}(n)$ time. Then apply Theorem 4.4. □

In all cases, if the required tree decomposition is not given, it can be found (if it exists) in $\mathcal{O}(n \log n)$ time [38]. It is also possible to find parallel algorithms that use polylogarithmic time for the considered problems on graphs with bounded treewidth.

Theorem 4.10. *For every constant* $k \geqslant 1$, VERTEX GENERALIZED GEOGRAPHY, EDGE GENERALIZED GEOGRAPHY *restricted to graphs with maximum degree* $\geqslant d$ *(d constant)*, HAMILTONIAN CIRCUIT CONSTRUCTION GAME, HAMILTONIAN PATH CONSTRUCTION GAME *when restricted to graphs with treewidth* $\leqslant k$ *belong to the class* NC.

Proof. This follows directly from the algorithms and the fact that a suitable tree decomposition with T a tree of logarithmic depth can be found in polylogarithmic time on a (CRCW or EREW) PRAM [12, 31]. □

A similar type of result holds for QUANTIFIED SATISFIABILITY. Lengauer [33, 34] introduced a method for hierarchical description of graphs. With this method, it is possible to specify graphs that have a size exponential in the size of the specification.

Theorem 4.11. *For each constant* $k \geqslant 1$, VERTEX GENERALIZED GEOGRAPHY, HAMILTONIAN PATH CONSTRUCTION GAME and HAMILTONIAN CIRCUIT CONSTRUCTION GAME *for hierarchical graphs, where each cell contains at most k vertices, can be solved in time linear in the size of the specification.*

Proof. Use a method similar to the method for graphs with bounded treewidth. For each cell G_i, we compute charac(V_i, X_i), where V_i are the vertices in cell G_i and X_i is the set of all other vertices in the expansion of G_i. We omit the details. □

Note the huge savings in running time over the straightforward algorithm to first expand the graph, and then use backtracking: the latter algorithm can use time double exponential in the size of the specification.

4.2. EDGE GENERALIZED GEOGRAPHY *on cacti*

In this section we give a linear-time algorithm for EDGE GENERALIZED GEOGRAPHY on cacti, based on graph rewriting. The resulting algorithm is easier and more practical than the algorithm for EDGE GENERALIZED GEOGRAPHY on graphs with bounded treewidth and degree. Also, we do not need to restrict the degree of the graphs here, but, on the other hand, the class of cacti is much more limited than that of the graphs with treewidth $\leqslant 2$.

We will use the following notations: (G, s) denotes the game (instance), where the EDGE GENERALIZED GEOGRAPHY game is played on the graph G with starting vertex s.

We will allow for parallel edges and self-loops in the undirected graphs that we are dealing with. Write $(G, s) \equiv (H, t)$ if and only if there is a winning strategy for player 1 in $(G, s) \Leftrightarrow$ there is a winning strategy for player 1 in (H, t).

Lemma 4.12. *Let $G=(V, E)$ be an undirected graph. Let $s, v \in V$, degree$(v)=1$, $(v, w) \in E$, $v \neq s$, $w \neq s$. Let $G-\{v, w\}$ denote the graph $(V-\{v, w\}, E-\{(x, y) \in E \mid x=v \vee x=w\})$. Then $(G, s) \equiv (G-\{v, w\}, s)$.*

Proof. We show the construction in Fig. 8. Suppose player $j \in \{1, 2\}$ has a winning strategy in $(G-\{v, w\}, s)$. Then he has a winning strategy in (G, s): as long as player $3-j$ does not move to w, make the same moves as in $(G-\{v, w\}, s)$. When player $3-j$ moves to w, then move to v and win the game. Now the lemma follows. □

Lemma 4.13. *Let $G=(V, E)$ be an undirected graph. Let $(v, w) \in E$; degree$(v)=$ degree$(w)=2$; and $s \notin \{v, w\}$. Let the neighbors of v be w and x, and the neighbors of w be v and y. Let $G'=(V-\{v, w\}, E-\{(x, v), (v, w), (w, y)\} \cup \{(x, y)\})$. Then $(G, s) \equiv (G', s)$.*

Proof. The construction is shown in Fig. 9. The game on both graphs is similar: e.g., when player i moves from x to y in G', this corresponds to the situation that player i moves from x to v, then player $3-i$ moves from v to w, and then player i moves from w to y in G. □

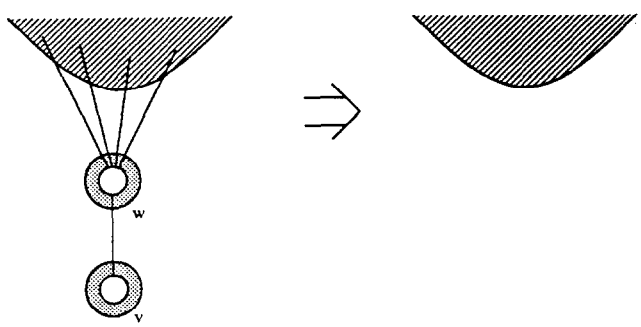

Fig. 8.

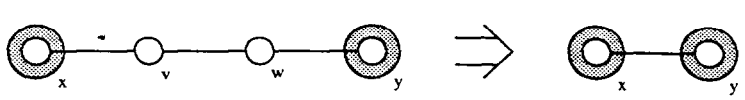

Fig. 9.

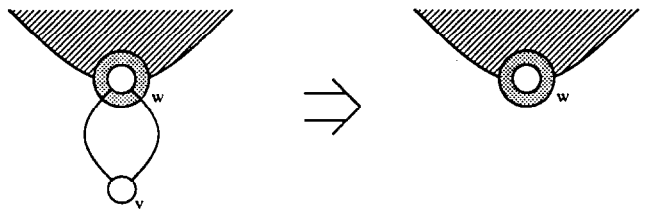

Fig. 10.

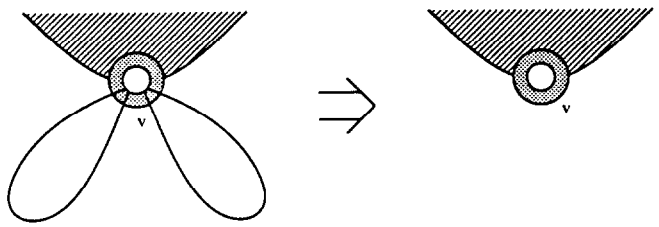

Fig. 11.

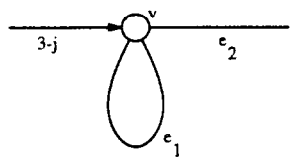

Fig. 12.

Proof. The construction is shown in Fig. 10. If there exists a winning strategy in $(G-\{v\}, s)$ for player $j \in \{1,2\}$, then there exists one in (G,s): move as in $(G-\{v\},s)$, except when player $3-j$ moves to v, then move back to w. □

Lemma 4.15. *Let $G=(V,E)$ be an undirected graph. Let $v \in V$ be adjacent to two self-loops $e_1=(v,v)$ and $e_2=(v,v)$ $(e_1 \neq e_2)$. Then $(G,s) \equiv (G',s)$ with, $G'=(V,E-\{e_1,e_2\})$.*

Proof. Similar as before. When player $3-j$ moves over e_1 or e_2, then player j moves over the other edge in $\{e_1,e_2\}$ (see Fig. 11 for the construction). □

Lemm 4.16. *Let $G=(V,E)$ be an undirected graph. Let $v \in V$. Suppose v is adjacent to exactly 3 edges, where exactly one of these is a self-loop, and suppose $v \neq s$. Let $G-\{v\}$ be as above. Then $(G,s) \equiv (G-\{v\},s)$.*

Proof. Suppose there is a winning strategy for player $j \in \{1,2\}$ in $(G-\{v\},s)$. Then there is a winning strategy for player j in (G,s). As long as player $3-j$ does not move to v, player j moves as in $(G-\{v\},s)$. Suppose player $3-j$ moves to v. Let e_1 be the self-loop (v,v), and let e_2 be the other unused edge, adjacent to v (see Fig. 12). Moving

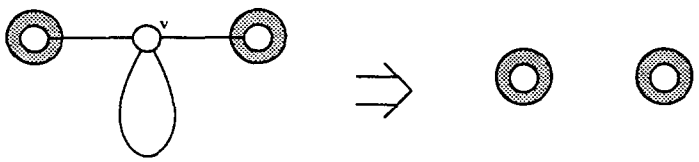

Fig. 13.

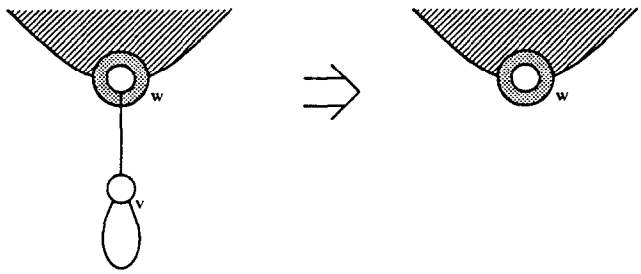

Fig. 14.

over edge e_2 is either a winning or a losing move, regardless of what player makes the move. So, if it is a winning move, player j moves over e_2, and if it is a losing move, then player j moves over e_1, and player $3-j$ must move over e_2 and loses the game (see Fig. 13 for the construction). □

Lemma 4.17. *Let $G=(V,E)$ be an undirected graph. Let $v\in V$; suppose v is adjacent to exactly 2 edges, one of which is a self-loop. Let $(v,w)\in E$, $v\neq w$ be the other edge. Suppose $v\neq s$. Then $(G,s)\equiv(G-\{v\},s)$.*

Proof. Similar as before. Player j plays in G as in $G-\{v\}$, but when player $3-j$ moves to v from w, then player j moves over the self-loop and wins the game (see Fig. 14 for the construction). □

Lemma 4.18. *After applying the rules of Lemmas* 4.12–4.17 *as often as possible, starting with (G,s), with G a connected cactus, a game (H,s) will result, with $H=(\{s\},\emptyset)$, $H=(\{s\},\{(s,s)\})$ or $H=(\{s,v\},\{(s,v)\})$ for some v.*

Proof. (See Fig. 15 for the possibilities for H.) Each application of one of Lemmas 4.12–4.17 will result in another, smaller cactus. Note that the biconnected components of G form a tree. Every leaf node in this tree that corresponds to a single edge or a cycle with even length will disappear with Lemma 4.12, 4.13 and 4.14. Every leaf node corresponding to a cycle with odd length will reduce to a self-loop. Suppose no application of one of Lemmas 4.12–4.16 is possible. The resulting graph H cannot have more than one biconnected component. Suppose not. Look at a biconnected

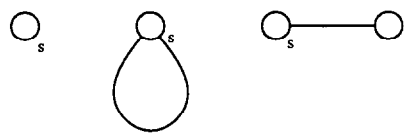

Fig. 15.

component that is a leaf in the tree of biconnected components if we do not look to self-loops. If it is a cycle, and some vertices ($\neq$ the unique vertex adjacent to other biconnected components) have self-loops, then Lemma 4.15 or 4.16 can be applied. If it is a cycle without such self-loops, it can be reduced to nothing or a self-loop. If it is a single edge, then Lemma 4.12, 4.15 or 4.17 can be applied. With a similar argument, H cannot have a single biconnected component with three or more vertices. So, H has at most 2 vertices. Simple case analysis gives the theorem. □

Theorem 4.19. EDGE GENERALIZED GEOGRAPHY *can be solved in* $\mathcal{O}(n)$ *time on cacti.*

Proof. First we remark that we may restrict ourselves to connected graphs. Cacti have $\mathcal{O}(n)$ edges. It remains to show that by proper choice of data structures, we can dynamically determine where one of Lemmas 4.12–4.17 can be applied, in $\mathcal{O}(1)$ amortized time per operation. Hereto, we use for each vertex an integer variable, that denotes the current degree of the veretex, and a boolean variable, that denotes whether it has an adjacent self-loop. (We may assume, by Lemma 4.15, that each vertex has 0 or 1 adjacent self-loops.) In a queue Q we put each vertex, where one of Lemmas 4.12–4.14, 4.16 and 4.17 can be applied. Repeatedly, a vertex is taken from Q; if the vertex has not been deleted already by an earlier operation, the operation corresponding to v is applied. For each removed edge, its still existing endpoints have their degree updated, and are possibly put in Q, and some other checks are made (depending upon the particular operation), possibly resulting in the setting of a "self-loop boolean", or putting one or more vertices in Q. We omit the easy, but tedious, details. Finally, if the resulting graph $H=(\{s\},\emptyset)$, then player 2 has a winning strategy; otherwise, player 1 has a winning strategy. □

It is possible to prove other lemmas of a similar flavor as Lemmas 4.12–4.17. With similar techniques one can show the following result.

Theorem 4.20. EDGE GENERALIZED GEOGRAPHY *can be solved in* $\mathcal{O}(n)$ *times for directed graphs* $G=(V,E)$ *with the property that the undirected graph* $G'=(V,\{(v,w)\mid(v,w)\in E \vee (w,v)\in E\})$ *is a cactus.*

Also, similar algorithms can be designed for the VERTEX GENERALIZED GEOGRAPHY game on cacti. The advantages of this approach over the algorithms in Section 4.1 are the simplicity and better running time (also in constant factor) of the algorithm.

5. Final comments

This research leaves several directions for further research. On the one hand, there are still several interesting variants that have not yet been shown to be PSPACE-complete, like EDGE GENERALIZED GEOGRAPHY without specified starting vertex, and most of the games considered in this paper on undirected graphs. One of the more promising of these seems to be VERTEX GENERALIZED GEOGRAPHY on undirected graphs with specified starting vertex. It may well be that this problem is solvable in polynomial time using graph matching. (Recall that this problem without specified starting vertex is equivalent to the problem whether the input graph has no perfect matching [19, p. 71].) On the other hand, much work can still be done on the complexity of the problems when restricted to special classes of graphs. There are many interesting problems in this area that are worth being studied.

We close with a mention, without proof, of some other special cases of the problems considered in this paper:

- VERTEX GENERALIZED GEOGRAPHY, EDGE GENERALIZED GEOGRAPHY, SIMPLE PATH CONSTRUCTION GAME and ELEMENTARY PATH CONSTRUCTION GAME are solvable in $\mathcal{O}(n+e)$ time on acyclic graphs, but become PSPACE-complete if restricted to graphs obtained by adding one edge to an acyclic graph.
- All games that are considered in this paper are linear-time-solvable on (undirected graphs that are) trees.
- SIMPLE PATH CONSTRUCTION GAME is solvable in $\mathcal{O}(n)$ time on cacti.
- ELEMENTARY PATH CONSTRUCTION GAME is solvable in $\mathcal{O}(n^3 + 2^{d/2}n)$ time for cacti with maximum vertex degree d.

References

[1] A. Adachi, S. Iwata and T. Kasai, Some combinatorial game problems require $\Omega(n^k)$ time, *J. ACM* **31** (1984) 361–376.

[2] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability – a survey, *BIT* **25** (1985) 2–23.

[3] S. Arnborg, Some PSPACE-complete logic decision problems that become linear-time-solvable on formula graphs of bounded treewidth, manuscript, 1991.

[4] S. Arnborg, D.G. Corneil and A. Proskurowski, Complexity of finding embeddings in a k-tree, *SIAM J. Algebraic Discrete Methods* **8** (1987) 277–284.

[5] S. Arnborg, B. Courcelle, A. Proskurowski and D. Seese, An algebraic theory of graph reduction, in: *Proc. 4th Workshop on Graph Grammars and Their Applications to Computer Science*, Lecture Notes in Computer Science, Vol. 532 (Springer, Berlin, 1991) 70–83.

[6] S. Arnborg, J. Lagergren and D. Seese, Easy problems for tree decomposable graphs, *J. Algorithms*, **12** (1991) 308–340.

[7] S. Arnborg and A. Proskurowski, Characterization and recognition of partial 3-trees, *SIAM J. Algebraic Discrete Methods* **7** (1986) 305–314.

[8] S. Arnborg and A. Proskurowski, Linear-time algorithms for NP-hard problems restricted to partial k-trees, *Discrete Appl. Math.* **23** (1989) 11–24.

[9] M.W. Bern, E.L. Lawler and A.L. Wong, Linear-time computation of optimal subgraphs of decomposable graphs, *J. Algorithms* **8** (1987) 216–235.

[10] H.L. Bodlaender, Classes of graphs with bounded treewidth, Tech. Report RUU-CS-86-22, Department of Computer Science, Utrecht University, Utrecht, 1986.
[11] H.L. Bodlaender, Dynamic programming algorithms on graphs with bounded tree-width, in: *Proc. 15th Internat. Colloq. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988) 105–119.
[12] H.L. Bodlaender, NC-algorithms for graphs with small treewidth, in: J. van Leeuwen, ed., *Proc. Workshop on Graph-Theoretic Concepts in Computer Science WG'88*, Lecture Notes in Computer Science, Vol. 344 (Springer, Berlin, 1988) 1–10.
[13] H.L. Bodlaender, Planar graphs with bounded treewidth, Tech. Report RUU-CS-88-14, Department of Computer Science, Utrecht University, Utrecht, 1988.
[14] H.L. Bodlaender, Improved self-reduction algorithms for graphs with bounded treewidth, in: *Proc. 15th Internat. Workshop on Graph-Theoretic Concepts in Computer Science, WG'89*, Lecture Notes in Computer Science, Vol. 411 (Springer, Berlin, 1990) 232–244; *Ann. Discrete Math.*, to appear.
[15] H.L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, *J. Algorithms* **11** (1990) 631–643.
[16] H.L. Bodlaender, On the complexity of some coloring games, *Internat. J. Found. Comput. Sci.* **2** (1991) 133–147.
[17] H.L. Bodlaender and T. Kloks, Fast algorithms for the TRON game on trees, Tech. Report RUU-CS-90-11, Department of Computer Science, University of Utrecht, 1990.
[18] H.L. Bodlaender and T. Kloks, Better algorithms for the pathwidth and treewidth of graphs, in: *Proc. 18th Internat. Colloq. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 510 (Springer, Berlin, 1991) 544–555.
[19] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications* (Elsevier/MacMillan, New York, 1976).
[20] R.B. Borie, R.G. Parker and C.A. Tovey, Automatic generation of linear algorithms from predicate calculus descriptions of problems on recursively constructed graph families, *J. Algorithms* **7** (1992) 555–582.
[21] B. Courcelle, The monadic second-order logic of graphs I: Recognizable sets of finite graphs, *Inform. and Comput.* **85** (1990) 12–75.
[22] S. Even and R. Tarjan, A combinatorial problem which is complete in polynomial space, *J. ACM* **23** (1976) 710–719.
[23] M.R. Fellows and M.A. Langston, An analogue of the Myhill–Nerode theorem and its use in computing finite-basis characterizations, in: *Proc. 30th Ann. Symp. on Foundations of Computer Science* (1989) 520–525.
[24] A. Fraenkel, M. Garey, D. Johnson, T. Schaefer and Y. Yesha, The complexity of checkers on an $n \times n$ board – preliminary version, in: *Proc. 19th Ann. Symp. on Foundations of Computer Science* (1978) 55–64.
[25] A. Fraenkel and E. Goldschmidt, Pspace-hardness of some combinatorial games, *J. Combin. Theory Ser. A*, **46** (1987) 21–38.
[26] A. Fraenkel and D. Lichtenstein, Computing a perfect strategy for n by n chess requires time exponential in n, *J. Combin. Theory Ser. A*, **31** (1981) 199–214.
[27] A.S. Fraenkel and S. Simonson, Geography, *Theoret. Comput. Sci.* **110** (1993) 197–214.
[28] M.R. Garey and D.S. Johnson, *Computers and Intractability, A Guide to the Theory of* NP-*Completeness* (Freeman, New York, 1979).
[29] D. Johnson, The NP-completeness column: an ongoing guide, *J. Algorithms* **4** (1983) 397–411.
[30] D.S. Johnson, The NP-completeness column: an ongoing guide, *J. Algorithms* **6** (1985) 434–451.
[31] J. Lagergren, Efficient parallel algorithms for tree decomposition and related problems, in: *Proc. 31st Ann. Symp. on Foundations of Computer Science* (1990) 173–182.
[32] C. Lautemann, Efficient algorithms on context-free graph languages, in: *Proc. 15th Internat. Colloq. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988) 362–378.
[33] T. Lengauer, Efficient algorithms for finding minimum spanning forests in hierarchically defined graphs, *J. Algorithms* **8** (1987) 260–284.
[34] T. Lengauer and E. Wanke, Efficient solutions of connectivity problems on hierarchically defined graphs, *SIAM J. Comput.* **17** (1988) 1063–1080.
[35] D. Lichtenstein and M. Sipser, Go is polynomial-space hard, *J. ACM* **27** (1980) 393–401.

[36] J. Matoušek and R. Thomas, Algorithms finding tree decompositions of graphs, *J. Algorithms* **12** (1991) 1–22.
[37] K. Mehlhorn, S. Näher and M. Rauch, On the complexity of a game related to the dictionary problem, in: *Proc. 30th Ann. Symp. on Foundations of Computer Science* (1989) 546–548.
[38] B. Reed, Finding approximate separators and computing treewidth quickly, in: *Proc. Symp. on Theory of Computation* (1992) 221–228.
[39] N. Robertson and P.D. Seymour, Graph minors. XIII. The disjoint paths problem, manuscript, 1986.
[40] J. Robson, N by N checkers is exptime complete, *SIAM J. Comput.* **13** (1984) 252–267.
[41] J. Robson, Alternation with restrictions on looping, *Inform. and Control* **67** (1985) 2–11.
[42] T.J. Schaefer, The complexity of satisfiability problems, in: *Proc. 10th Symp. on Theory of Computation* (1978) 216–226.
[43] T.J. Schaefer, On the complexity of some two-person perfect-information games, *J. Comput. System Sci.*, **16** (1978) 185–225.
[44] P. Scheffler, Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme, Ph.D Thesis, Akademie der Wissenschaften der DDR, Berlin, 1989.
[45] L.J. Stockmeyer and A.K. Chandra, Provably difficult combinatorial games, *SIAM J. Comput.* **8** (1979) 151–174.
[46] T.V. Wimer, Linear algorithms on k-terminal graphs. Ph.D Thesis, Department of Computer Science, Clemson University, 1987.