Theoretical Computer Science 614 (2016) 97–108

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Graph operations on parity games and polynomial-time algorithms

Christoph Dittmann [a,*], Stephan Kreutzer [a], Alexandru I. Tomescu [b,1]

[a] Chair for Logic and Semantics, Technical University Berlin, Germany
[b] Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland

ARTICLE INFO

Article history:
Received 18 November 2013
Received in revised form 24 August 2015
Accepted 21 November 2015
Available online 10 December 2015
Communicated by P. Aziz Abdulla

Keywords:
Parity games
Graph theory
Directed graphs
Tournaments
Graph operations
Graph algorithms

ABSTRACT

In this paper we establish polynomial-time algorithms for special classes of parity games. In particular we study various constructions for combining graphs that often arise in structural graph theory and show that polynomial-time solvability of parity games is preserved under these operations. This includes the join of two graphs, repeated pasting along vertices, and the addition of a vertex. As a consequence we obtain polynomial time algorithms for parity games whose underlying graph is an orientation of a complete graph (such as tournaments), a complete bipartite graph, a block graph, or a block-cactus graph. These are classes where the problem was not known to be efficiently solvable before.

1. Introduction

A parity game is a form of two player games on labelled directed graphs in which the players move a token along edges of the digraph. The vertex set of the digraph is partitioned into two sets $V_{\bigcirc}, V_{\Box}$ of positions for the two players $\bigcirc$ and $\Box$. At the beginning of the play, the token resides on some initial vertex and in each round Player $\bigcirc$ can push the token to a neighbour of the current position if this position is in $V_{\bigcirc}$ and otherwise Player $\Box$ can move the token along an edge. A player who cannot move loses immediately. For infinite plays the winner is determined as follows. Every vertex in the digraph is labelled with a natural number, its *priority*. Player $\bigcirc$ wins an infinite play if the maximal priority of a vertex that is repeated infinitely often is even, otherwise Player $\Box$ wins.

The computational problem of solving parity games is to determine for a given parity game and initial position v which of the two players has a winning strategy starting from that vertex. Usually, algorithms for solving parity games such as in [15,6,20,29,28] immediately compute the winning regions for the two players, i.e. the set of positions from which the players have a winning strategy. It is well-known that in any parity game from every vertex exactly one of the players has a

* Corresponding author.
E-mail addresses: christoph.dittmann@tu-berlin.de (C. Dittmann), stephan.kreutzer@tu-berlin.de (S. Kreutzer), tomescu@cs.helsinki.fi (A.I. Tomescu).

[1] This work was carried out while A.I. Tomescu was visiting the Chair for Logic and Semantics, Technical University Berlin. Support by the European Science Foundation, activity "Games for Design and Verification", exchange grant 3680, and by the Academy of Finland, grant 274977, is gratefully acknowledged.

http://dx.doi.org/10.1016/j.tcs.2015.11.044
0304-3975/

winning strategy (see [10]). Hence, the problem is symmetric in the two players and the union of the two winning regions is the entire graph.

Parity games have been studied intensively in the literature. A primary motivation for this comes from the area of formal verification of reactive systems by model checking (see [9,1]). Deciding the winner in a parity game is polynomial-time equivalent to the model checking problem of the modal μ-calculus (e.g., [15]), a prominent logic for the specification of temporal properties of processes combining high expressive power with good algorithmic properties. It is also equivalent to the non-emptiness problem of ω-tree automata [11].

Despite considerable efforts in the community, determining the exact computational complexity of solving parity games is a long-standing open problem. It is known that the problem is in the complexity class UP $\cap$ coUP [19]. The currently best explicit algorithm is subexponential [21]. Another promising approach is strategy improvement [28]. However, this algorithm as presented in [28] has been shown to require exponential time in the general case [13].

As polynomial-time algorithms for solving general parity games remain elusive, various papers have approached the problem by identifying restricted classes of parity games which are solvable in polynomial time. These include classes of bounded tree-width [23,12], bounded entanglement [5], bounded DAG-width [4], and bounded clique-width [24].

One motivation for this line of research is that efficient solutions to special classes of parity games may already be interesting from an application point of view. Furthermore, it may also pave the way for solving parity games in general using a graph structural approach: we first show that certain classes of "basic" parity games can be solved in polynomial time and then show that all parity games can be built up from these by operations that preserve polynomial-time solvability.

Such structural methods for establishing polynomial-time algorithms have been very successful in the past. Let us briefly review this approach as it provides part of the motivation for the results obtained in this paper. The most prominent example for a polynomial-time algorithm developed using such a structural approach is perhaps the disjoint paths problem which has been shown by Robertson and Seymour to be solvable in polynomial time [27]. Another good example is a polynomial-time algorithm for computing the crossing numbers of a graph given in [16]. In both cases, the algorithms work as follows: As a base case, it is first shown that the problem can easily be solved on graphs of bounded tree-width. Hence it remains to consider graphs of very high tree-width. But in this case the fact that the tree-width is high also yields information about the graph. In particular it either contains a large clique minor or a large flat grid. If there is a large clique minor, the problems can both be solved directly. Otherwise, as shown in [26], the graph can be decomposed into simpler graphs which are "almost" embeddable into a fixed surface. "Almost" in this context means that they are embeddable after removing a constant number of vertices which are called *apices* (furthermore, there are constantly many regions of non-planarity which are called *vortices* but are not so important for our results here). This decomposition into simpler graphs can then be used to solve the problem in the case that there is no large grid minor.

In this context, the motivation for studying parity games on restricted classes of graphs such as bounded tree-width can also be understood as providing polynomial-time algorithms for the base cases of a structural approach as described above. While many papers such as [23,12,5,4,24] study potential base cases, so far not much work has gone into developing methods that would allow to combine parity games from polynomial-time solvable classes of parity games to form larger and more general parity games preserving polynomial-time solvability. However, for any structural approach to parity games to work, such methods are an essential part.

In this paper we therefore take first steps towards closing this gap by studying some constructions that typically arise in structural graph theory: the join of two graphs, repeated pasting along vertices, and the addition of a vertex. See below for details of these constructions. More precisely, given a class $\mathcal{C}$ of digraphs on which we can solve parity games in polynomial time, we show that the same holds for the class obtained from $\mathcal{C}$ by applying once any of these three operations to its elements.

As explained above, *apices*, i.e. a constant number of vertices that have to be removed to obtain a graph embeddable into a fixed surface, play an important role in the design of algorithms using structural results developed in Robertson and Seymour's graph minor project [25]. Our methods here allow to deal with these apices. More precisely, if parity games embeddable into a fixed surface can be shown to be polynomial-time solvable, then our methods allow to lift this to parity games embeddable into a fixed surface up to a constant number of apices. In the same way, our results can be applied to any tractable class of parity games to obtain larger and still tractable classes of parity games.

The methods developed here can furthermore be used to obtain new classes of parity games which can be solved in polynomial time. These results provide, in particular, polynomial time algorithms for parity games whose underlying graph is an orientation of a complete graph (such as tournaments), a complete bipartite graph, a block graph, or a block-cactus graph. These are classes where the problem was not known before to be efficiently solvable.

It is worth noting that energy games, a natural generalisation of parity games, have been recently shown on orientations of complete bipartite graphs to be as hard as in the general case [8].

2. Preliminaries

2.1. Basic definitions and results

A *directed graph* (digraph) is a pair $G = (V, E)$ where V is the set of vertices and $E \subseteq V \times V$ is the set of arcs. For a vertex v, we write $N(v)$ for the set of neighbours of v, that is, $N(v) := \{u \in V \mid (u, v) \in E \text{ or } (v, u) \in E\}$. In this paper,

a *graph* is always a directed graph and we only consider finite graphs. An *undirected graph* is a graph where E is a symmetric irreflexive relation. For undirected graphs, we write $\{x, y\} \in E$ instead of $(x, y) \in E$.

It makes no difference whether we allow reflexive edges in directed graph or not because parity games with reflexive edges can readily be reduced to parity games without reflexive edges.

Complying with the terminology in [2], we say that an *orientation* of an undirected graph $G = (V, E)$ is a digraph obtained from G by replacing every edge $\{x, y\} \in E$ by one of the arcs (x, y) or (y, x), but not both. A *biorientation* of G is a digraph obtained from G by replacing every edge $\{x, y\} \in E$ with the arcs (x, y) or (y, x) or both.

A *parity game* $P = (V, V_\bigcirc, V_\square, E, \Omega)$ is a finite directed graph (V, E) with a partitioning of the vertices $V = V_\bigcirc \cup V_\square$ equipped with a *priority* map $\Omega : V \to \mathbb{N}$. We write $V(P)$ to denote the set of vertices of P, and similarly $V_\bigcirc(P)$, $V_\square(P)$, $E(P)$ and $\Omega(P)$ to denote the respective parts of P.

A *play* on P is a maximal sequence of vertices $v_0, v_1, v_2, \ldots \in V$ such that for all i there is an edge $(v_i, v_{i+1}) \in E$. If the play is finite and the last node is v, then Player $\bigcirc$ wins the play if and only if $v \in V_\square$. If the play is infinite, Player $\bigcirc$ wins the play if and only if the maximum priority that appears infinitely often is even.

A *positional strategy* for Player $\bigcirc$ is a map $\rho : V_\bigcirc \to V$ such that $\rho(v)$ is a successor of v for all v such that v has a successor. A play $v = v_0, v_1, v_2, \ldots$ *conforms* to ρ if $v_{i+1} = \rho(v_i)$ for all i such that $v_i \in V_\bigcirc$. A positional strategy ρ is a *positional winning strategy* for Player $\bigcirc$ from vertex v if every play that starts at v and conforms to ρ is winning for Player $\bigcirc$.

We call the set of vertices $W_\bigcirc(P) \subseteq V$ from which Player $\bigcirc$ has a positional winning strategy the *winning region* of Player $\bigcirc$, similar for $W_\square$ and Player $\square$. We will write $W_\bigcirc$, $W_\square$ if the game is clear from the context.

Parity games are *positionally determined* in the sense that $W_\bigcirc \cup W_\square = V$ and $W_\bigcirc \cap W_\square = \emptyset$ [15, Chapter 6]. For this reason, we only consider positional strategies from now on.

Given $A \subseteq V$, we denote by $P \cap A$ the parity game restricted to the vertices in A, that is, $(V \cap A, V_\bigcirc \cap A, V_\square \cap A, E \cap (A \times A), \Omega\restriction_A)$. Similarly, we denote the game $P \cap (V \setminus A)$ by $P \setminus A$. Given a class of parity games $\mathcal{C}$, we say that $\mathcal{C}$ is *hereditary* if for all $P \in \mathcal{C}$ and all subsets A of vertices of P, we have $P \cap A \in \mathcal{C}$. For a given game P, we call a game P' a *proper subgame of* P if $P' = P \cap V(P')$ and $P' \neq P$. For $d \in \mathbb{N}$, we denote by $\Omega^{-1}(d)$ the set of vertices having priority d.

We say that a class $\mathcal{C}$ of parity games is $O(f(n))$-*solvable* if there is a deterministic algorithm solving parity games from $\mathcal{C}$ in time $O(f(n))$, where n is the number of nodes. Similarly, we call a class $O(f(n))$-*decidable* if there is an $O(f(n))$-time algorithm that decides membership to $\mathcal{C}$.

Furthermore, we call a class *polynomial-time solvable (decidable)* if it is $O(f(n))$-solvable (decidable) where f is a polynomial.

The following two notions are well-known [15] and form the basis of the exponential-time algorithms of McNaughton [22] and Zielonka [29], and of the sub-exponential-time algorithm of Jurdziński, Paterson, Zwick [21].

For $i \in \{\bigcirc, \square\}$, we denote by $\overline{i}$ the element of $\{\bigcirc, \square\} \setminus \{i\}$. Given $i \in \{\bigcirc, \square\}$, a set $A \subseteq V$ is said to be i-*closed* if for every $u \in A$, the following two properties hold:

- If $u \in V_i$, then there is some $(u, v) \in E$ such that $v \in A$.
- If $u \in V_{\overline{i}}$, then for every $(u, v) \in E$ we have $v \in A$.

We denote by $\mathrm{attr}_i(A)$ the set of vertices in V from which Player i has a strategy to enter A at least once, and call it the i-*attractor set of* A. Clearly, the sets W_i, and $\mathrm{attr}_i(A)$, for every i-closed set $A \subseteq V$, are i-closed, and $\mathrm{attr}_i(W_i) = W_i$. Attractor sets are particularly useful since they allow a decomposition of a parity game, as Lemmas 1 and 2 testify. Working with attractor sets is also computationally efficient, since they can be computed in time linear in the number of edges.

The following two lemmas are well-known results about i-closed sets and i-attractor sets.

Lemma 1. *Let $P = (V, V_\bigcirc, V_\square, E, \Omega)$ be a parity game. For every $A \subseteq V$ and $i \in \{\bigcirc, \square\}$, the set $V \setminus \mathrm{attr}_i(A)$ is $\overline{i}$-closed. Moreover, $W_{\overline{i}}(P \setminus \mathrm{attr}_i(A)) \subseteq W_{\overline{i}}(P)$.*

Lemma 2. *Let $P = (V, V_\bigcirc, V_\square, E, \Omega)$ be a parity game and let $i \in \{\bigcirc, \square\}$. If $U \subseteq W_i(P)$, then*

1. *$W_i(P) = \mathrm{attr}_i(U) \cup W_i(P \setminus \mathrm{attr}_i(U))$ and*
2. *$W_{\overline{i}}(P) = W_{\overline{i}}(P \setminus \mathrm{attr}_i(U))$.*

Proof.

1. First we show "$\supseteq$". From $U \subseteq W_i(P)$ it follows that $\mathrm{attr}_i(U) \subseteq W_i(P)$. Let $v \in W_i(P \setminus \mathrm{attr}_i(U))$. Consider the plays starting from v in the game P and assume that Player i plays according to his winning strategy on the subgame $P \setminus \mathrm{attr}_i(U)$ as long as a play stays in this subgame. If a play never leaves the subgame, then Player i wins. If it leaves the subgame, then it enters $\mathrm{attr}_i(U)$ and we already know that from these vertices Player i wins in P, so we have $v \in W_i(P)$.
   For "$\subseteq$", let $v \in W_i(P)$ and assume $v \notin \mathrm{attr}_i(U)$. Because $V \setminus \mathrm{attr}_i(U)$ is $\overline{i}$-closed, v is also winning in $P \setminus \mathrm{attr}_i(U)$. If it were not, then Player $\overline{i}$ could force the play to stay in the Player $\overline{i}$-closed set $V \setminus \mathrm{attr}_i(U)$ and win in P.

Algorithm 1: Turn a partial solution into a full solution.

```
SOLVE(P = (V, V_○, V_□, E, Ω))
    R ← HALF-SOLVE(P)
    if R = (W_○, W_□) then
        return (W_○, W_□)
    if R = (P', W*_○, W*_□) then
        (W_○(P'), W_□(P')) ← SOLVE(P')
        return (W*_○ ∪ W_○(P'), W*_□ ∪ W_□(P'))
    if R = P then
        d ← MAXIMUM-PRIORITY(Ω)
        i ← ○ if d is even, □ otherwise
        (C_○, C_□) ← SOLVE(P \ attr_i(Ω^{-1}(d)))
        if C_ī ≠ ∅ then
            (W_i, W_ī) ← (∅, V)
        else
            (W_i, W_ī) ← (V, ∅)
        return (W_○, W_□)
```

2. By the determinacy of parity games, we have $W_{\bar{i}}(P') = V \setminus W_i(P')$ for all games P'. Together with the previous statement we have

$$\begin{aligned} W_{\bar{i}}(P) &= V \setminus W_i(P) \\ &= (V \setminus \mathrm{attr}_i(U)) \cap (V \setminus W_i(P \setminus \mathrm{attr}_i(U))) \\ &= (V \setminus \mathrm{attr}_i(U)) \cap W_{\bar{i}}(P \setminus \mathrm{attr}_i(U)) \\ &= W_{\bar{i}}(P \setminus \mathrm{attr}_i(U)). \qquad \square \end{aligned}$$

A *single-player game* is a parity game where all nodes belong to one of the players. If all the nodes of a single-player game P belong to Player i, we will say that P *belongs to* Player i.

We will use a function SOLVE-SINGLE-PLAYER-GAME(P) which solves single-player games in time cubic in the number of vertices, by returning the two winning regions $(W_\bigcirc(P), W_\square(P))$. A description of such an algorithm is given in [14].

2.2. Half-solving parity games

In the sequel we will repeatedly use the following lemma, whose proof is based on McNaughton's algorithm [22].

Definition 1 *(Half-solving parity games).* Let $\mathcal{C}$ be a class of parity games. An algorithm *half-solving parity games in* $\mathcal{C}$ is an algorithm which takes a game $P \in \mathcal{C}$ as input and returns one of the following three results.

1. $W_\bigcirc(P)$ and $W_\square(P)$.
2. A proper subgame P' of P and sets $W^*_\bigcirc, W^*_\square \subseteq V(P) \setminus V(P')$ such that $W_\bigcirc(P) = W^*_\bigcirc \cup W_\bigcirc(P')$ and $W_\square(P) = W^*_\square \cup W_\square(P')$.
3. The game P itself, but only if either $W_\bigcirc(P) = \emptyset$ or $W_\square(P) = \emptyset$.

Lemma 3. *Let $\mathcal{C}$ be a hereditary class of parity games. If there is a $c \geq 1$ and an $O(n^c)$-time algorithm half-solving parity games in $\mathcal{C}$, then $\mathcal{C}$ is $O(n^{c+1})$-solvable.*

Proof. Let HALF-SOLVE be the given algorithm. Consider Algorithm 1.

Let $T(n)$ be the running time of SOLVE and kn^c be the running time of HALF-SOLVE with $k \in \mathbb{N}$. Then we have

$$T(n) \leq O(n) + kn^c + f(n) + T(n-1)$$

where $f(n)$ is the time to compute the attractor set $\mathrm{attr}_i(\Omega^{-1}(d))$.

An attractor set can be computed in time $O(n^2)$, but this would only give the bound $T(n) \in O(n^{c+2})$. However, we observe that in total each arc is only considered once because $P \setminus \mathrm{attr}_i(\Omega^{-1}(d))$ does not contain any arcs with their start or end point in $\mathrm{attr}_i(\Omega^{-1}(d))$. So the algorithm only needs $O(n^2)$ time to compute all the attractor sets in total over all recursive steps. So we have $T(n) \in O(n^{c+1} + n^2)$, which gives the desired bound of $T(n) \in O(n^{c+1})$.

To prove the correctness, it is enough to consider the case where HALF-SOLVE returns P unmodified because the other cases are trivially correct.

Let $D \subseteq V$ be the set of vertices of highest priority in P. Assume that their priority is good for Player $\bigcirc$ (the case of Player $\square$ is similar). Then the algorithm solves the game $P' := P \setminus \mathrm{attr}_\bigcirc(D)$.

If $W_\square(P') \neq \emptyset$, then $W_\square(P) \neq \emptyset$ because every vertex winning for Player $\square$ in P' is also winning for Player $\square$ in P, because we only removed a $\bigcirc$-attractor (Lemma 1). Since we are in the case where one of the winning regions of P is empty, this implies $W_\bigcirc(P) = \emptyset$.

If $W_\square(P') = \emptyset$, then it is easy to see that Player $\bigcirc$ wins everywhere in P. Indeed, if a play eventually stays in P', then Player $\bigcirc$ wins because $W_\square(P') = \emptyset$. If a play visits $\mathrm{attr}_\bigcirc(D)$ an infinite number of times, then Player $\bigcirc$ can force to visit D an infinite number of times, and hence wins the play. $\square$

In the previous lemma, we showed a method to decide the global winner under the restriction that there is one definite global winner and under the assumption that we can solve subgames recursively. A natural question is if this can be generalised to recognise winning regions without this restriction. Unfortunately, the answer is no, as shown in [3, Proposition 2].

3. The join of two parity games

In this section we will show that if $\mathcal{C}$ and $\mathcal{C}'$ are hereditary classes of parity games that can be solved in polynomial time, then the class of parity games obtained by *joining* games from $\mathcal{C}$ and $\mathcal{C}'$ can also be solved in polynomial time.

Definition 2 *(Join of parity games).* Given parity games $P' = (V', V'_\bigcirc, V'_\square, E', \Omega')$ and $P'' = (V'', V''_\bigcirc, V''_\square, E'', \Omega'')$ with $V' \cap V'' = \emptyset$, we say that the game $P = (V, V_\bigcirc, V_\square, E, \Omega)$ is a *join of P' and P''* if the following conditions hold:

- $V = V' \cup V''$, $V_\bigcirc = V'_\bigcirc \cup V''_\bigcirc$, $V_\square = V'_\square \cup V''_\square$,
- $E = E' \cup E'' \cup E^*$, where for all $x \in V'_\square$, $y \in V''_\bigcirc$ and for all $x \in V'_\bigcirc$, $y \in V''_\square$, the set $E^* \subseteq (V' \times V'') \cup (V'' \times V')$ contains at least one arc (x, y) or (y, x).
- The vertices of P have the same priorities as they have in P' and P''.

Given two classes of parity games $\mathcal{C}$ and $\mathcal{C}'$, we define

$$\mathsf{HalfJoin}(\mathcal{C}) := \left\{ P \mid P \text{ is a join of a single-player game } P' \text{ and a game } P'' \in \mathcal{C} \right\},$$

$$\mathsf{Join}(\mathcal{C}, \mathcal{C}') := \left\{ P \mid P \text{ is a join of } P' \in \mathcal{C} \text{ and } P'' \in \mathcal{C}' \right\}.$$

Remark 1. Recall that the *join* of two undirected graphs G_1 and G_2 is the operation which adds an edge between every vertex of G_1 and every vertex of G_2 (see e.g., [17]). Observe that a parity game whose underlying undirected graph is a biorientation of a join of two undirected graphs is a particular case of a join of two parity games. The second bullet of Definition 2 requires only that the vertices of opposite players in the two graphs be joined by an edge. It places no constraint on the other pairs of vertices.

Remark 2. Observe that if $\mathcal{C}$ and $\mathcal{C}'$ are hereditary, then so are $\mathsf{HalfJoin}(\mathcal{C})$ and $\mathsf{Join}(\mathcal{C}, \mathcal{C}')$.

We will first show how to solve parity games obtained by joining a polynomial time solvable parity game with a single-player game and then extend this construction to the general case of joining arbitrary parity games.

As an immediate corollary we get that parity games whose underlying undirected graph is a complete graph, so-called *tournaments*, can be solved in polynomial time. As corollary from the more general case of arbitrary joins we get that parity games whose underlying undirected graph is a complete bipartite graph can be solved in polynomial time. Note that the result for tournaments is not a special case of Obdržálek's polynomial time algorithm [24] for parity games of bounded directed clique-width because biorientations of complete graphs and of complete bipartite graphs do not have bounded directed clique-width, although their underlying undirected graphs have bounded undirected clique-width.

3.1. Adjoining vertices belonging to one player

We now define the construction that joins a parity game P and a parity game P' whose vertices belong to only one player. This construction will also be used later in the proof of the main theorem on joins, given in Section 3.2.

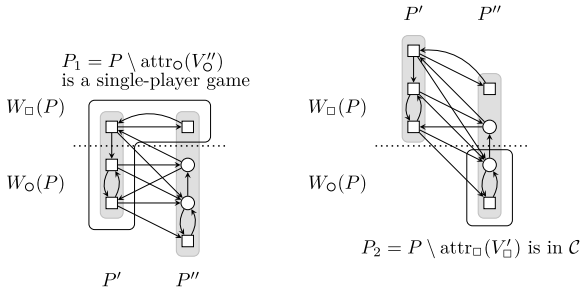
Lemma 4. *If $\mathcal{C}$ is a polynomial-time solvable hereditary class of parity games, then all games $P \in \mathsf{HalfJoin}(\mathcal{C})$ can be solved in polynomial time, provided that a decomposition of P as a join of a game in $\mathcal{C}$ with a single-player parity game is given.*

Proof. Let $P = (V, V_\bigcirc, V_\square, E, \Omega) \in \mathsf{HalfJoin}(\mathcal{C})$ be a join of the single-player game $P' = (V', V'_\bigcirc, V'_\square, E', \Omega')$ and the game $P'' = (V'', V''_\bigcirc, V''_\square, E'', \Omega'') \in \mathcal{C}$. We may assume without loss of generality that $V'_\bigcirc$ is empty, so that $V' = V'_\square$ and $V_\bigcirc = V''_\bigcirc$.

By Lemma 3 it suffices to establish an algorithm half-solving the parity games in $\mathsf{HalfJoin}(\mathcal{C})$. See Algorithm 2 for a detailed description of our algorithm. Note that the algorithm cannot assume that $V'_\bigcirc$ is empty because it must work on all valid inputs. Only in this correctness proof we assume that $V'_\bigcirc$ is empty because the other case follows analogously.

Algorithm 2: A polynomial-time algorithm for solving parity games on half joins where P is a join of the $\bigcirc$- or $\square$-player game $P' \in \mathcal{C}$.

```
HALF-SOLVE-HALF-JOIN(P = (V, V_○, V_□, E, Ω), P', P'')
    if V = ∅ then
        return (∅, ∅)
    i ← ○ if V_○(P') = ∅, □ otherwise
    (A_i, A_ī) ← SOLVE-SINGLE-PLAYER-GAME(P \ attr_i(V_i(P'')))
    if A_ī ≠ ∅ then
        W*_i ← ∅
        W*_ī ← attr_ī(A_ī)
        return (P \ W*_ī, W*_○, W*_□)
    (B_i, B_ī) ← SOLVE-C-GAME(P \ attr_ī(V_ī(P')))
    if B_i ≠ ∅ then
        W*_i ← attr_i(B_i)
        W*_ī ← ∅
        return (P \ W*_i, W*_○, W*_□)
    return P
```



(a) The first subgame (b) The second subgame

Fig. 1. An illustration of the subgames solved by Algorithm 2, where P is a join of the single-player parity game P' and the game $P'' \in \mathcal{C}$.

We assume that there is an algorithm SOLVE-$\mathcal{C}$-GAME that solves games in $\mathcal{C}$ in time $O(n^c)$ where n is the number of vertices of the game. Recall from Section 2.1 that single-player parity games can be solved in cubic time by the algorithm SOLVE-SINGLE-PLAYER-GAME(P).

The algorithm first solves the single-player game $P_1 := P \setminus \mathrm{attr}_i(V_i(P''))$. Under our assumption $V'_{\bigcirc} = \emptyset$ this means it solves the game $P_1 = P \setminus \mathrm{attr}_{\bigcirc}(V''_{\bigcirc})$. This takes time $O(n^3)$.

See Fig. 1(a) for an illustration of the situation, where nodes lie above the dotted line if and only if they are winning for Player $\square$. By Lemma 1 we know that $W_{\square}(P_1) \subseteq W_{\square}(P)$ and by Lemma 2 we get that

$$W_{\square}(P) = \mathrm{attr}_{\square}(W_{\square}(P_1)) \cup W_{\square}(P \setminus \mathrm{attr}_{\square}(W_{\square}(P_1))).$$

Therefore, if $W_{\square}(P_1)$ is not empty, then we can return $(P \setminus \mathrm{attr}_{\square}(W_{\square}(P_1)), \emptyset, \mathrm{attr}_{\square}(W_{\square}(P_1)))$, which is the second outcome of the algorithm according to Definition 1.

If $W_{\square}(P_1)$ is empty, then we solve the game $P_2 := P \setminus \mathrm{attr}_{\square}(V'_{\square})$, in time $O(n^c)$, and proceed as before (see Fig. 1(b)). By Lemma 1 we know that $W_{\bigcirc}(P_2) \subseteq W_{\bigcirc}(P)$. Thus, if $W_{\bigcirc}(P_2)$ is not empty, then we can return $(P \setminus \mathrm{attr}_{\bigcirc}(W_{\bigcirc}(P_2)), \mathrm{attr}_{\bigcirc}(W_{\bigcirc}(P_2)), \emptyset)$, which again is the second possible outcome according to Definition 1.

Finally, suppose both $W_{\square}(P_1)$ and $W_{\bigcirc}(P_2)$ are empty. We claim that either $W_{\square}(P) = \emptyset$ or $W_{\bigcirc}(P) = \emptyset$. In both cases we return P unmodified as this is the third possible outcome according to Definition 1. To establish the claim we distinguish two cases, depending on whether $V'_{\square} \cap W_{\bigcirc}(P) = \emptyset$ or not.

Case 1. $V'_{\square} \cap W_{\bigcirc}(P) \neq \emptyset$ (see Fig. 2(a)).

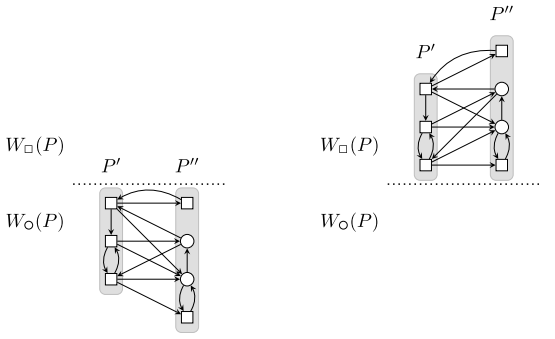
Observe that

$$V''_{\bigcirc} \subseteq W_{\bigcirc}(P).$$

Assume to the contrary that there is $v \in V''_{\bigcirc} \cap W_{\square}(P)$ and let $w \in V'_{\square} \cap W_{\bigcirc}(P)$. But then there can be no arc (v, w) in P because if there were such an arc, then Player $\bigcirc$ would have a winning strategy from v by choosing w as v's successor. Similarly, there cannot be an arc (w, v). This contradicts the definition of P. By Lemma 2, we have that

$$W_{\square}(P) = W_{\square}(P \setminus \mathrm{attr}_{\bigcirc}(V''_{\bigcirc})) = W_{\square}(P_1).$$

As $W_{\square}(P_1) = \emptyset$, this implies that $W_{\bigcirc}(P) = V$.



(a) Case 1 (b) Case 2

Fig. 2. An illustration of the case distinction of the proof of Lemma 4, where P is a join of the single-player parity game P' and the game $P'' \in \mathcal{C}$.

Case 2. $V'_\square \cap W_\bigcirc(P) = \emptyset$, equivalently $V'_\square \subseteq W_\square(P)$ (see Fig. 2(b)). Again by Lemma 2, we have that

$$W_\bigcirc(P) = W_\bigcirc(P \setminus \mathrm{attr}_\square(V'_\square)) = W_\bigcirc(P_2).$$

As $W_\bigcirc(P_2) = \emptyset$, this implies that $W_\square(P) = V$.

We see that the running time of the algorithm is $O(n^3 + n^c)$ because solving the single-player game and the game in $\mathcal{C}$ are the most expensive operations. With Lemma 3, we get an algorithm that solves all games in $\mathsf{HalfJoin}(\mathcal{C})$ in time $O(n^{1+\max\{3,c\}})$. □

Definition 3. We say that a digraph $D = (V, E)$ with a partition of its vertices $V = V_\bigcirc \cup V_\square$ is a *weak tournament* if between every two vertices $v \in V_\bigcirc$, $w \in V_\square$ we have that $(v, w) \in E$ or $(w, v) \in E$ (or both).

We denote by wTournaments the class of parity games on a weak tournament, that is,

$$\mathsf{wTournaments} := \{P = (V, V_\bigcirc, V_\square, E, \Omega) \mid (V, E) \text{ with the partition } V = V_\bigcirc \cup V_\square \text{ is a weak tournament}\}.$$

An easy corollary of Lemma 4 is the following.

Corollary 1. *There is an algorithm that solves all parity games* $P = (V, V_\bigcirc, V_\square, E, \Omega) \in \mathsf{wTournaments}$ *and runs in time* $O(|V|^4)$.

Proof. Let $\mathcal{C}$ be the class of single-player games. Then we have $\mathsf{HalfJoin}(\mathcal{C}) = \mathsf{wTournaments}$. The required decompositions can be easily computed. □

For $i \in \{\bigcirc, \square\}$, we define $\mathsf{HalfJoin}_i(\mathcal{C})$ to be the class of joins of single-player games that belong to player i with games from $\mathcal{C}$. Note that $\mathsf{HalfJoin}(\mathcal{C}) = \mathsf{HalfJoin}_\bigcirc(\mathcal{C}) \cup \mathsf{HalfJoin}_\square(\mathcal{C})$.

We show that in some cases we do not need to provide a decomposition as required by Lemma 4.

Corollary 2. *Let* $\mathcal{C}$ *be a hereditary class of parity games and* $i \in \{\bigcirc, \square\}$.

1. *If* $\mathcal{C}$ *is polynomial-time solvable, then* $\mathsf{HalfJoin}_i(\mathcal{C})$ *is polynomial-time solvable.*
2. *If* $\mathcal{C}$ *is polynomial-time solvable and also polynomial-time decidable, then* $\mathsf{HalfJoin}(\mathcal{C})$ *is polynomial-time solvable.*

Proof. Given $P = (V, V_\bigcirc, V_\square, E, \Omega) \in \mathsf{HalfJoin}_i(\mathcal{C})$, we can compute a decomposition of it in time linear in the number of edges. Indeed, we can take

- $P'_i := P \cap \{v \in V_i \mid N(v) \supseteq V_{\bar{i}}\}$ as the single-player game,
- $P''_i := P \setminus \{v \in V_i \mid N(v) \supseteq V_{\bar{i}}\}$ as the game from the hereditary class $\mathcal{C}$.

This proves the first statement.

If we do not know the player of the single-player game, then we have to check for both $i \in \{\bigcirc, \square\}$ which game P''_i considered above belongs to $\mathcal{C}$ in order to find a decomposition. This proves the second statement. □

Note that if we could also compute the winning strategies for the graphs in $\mathsf{HalfJoin}_i(\mathcal{C})$, then we could solve games in $\mathsf{HalfJoin}(\mathcal{C})$ without having to test membership to $\mathcal{C}$. Indeed, one could construct a game which assumes that Player $\bigcirc$ owns the single player game, and a game which assumes that Player $\square$ owns the single player game, run the algorithm

from Lemma 4 on these two instances in parallel, and then check the validity of the returned winning strategies. However, computing the winning strategies for games in $\mathsf{HalfJoin}_i(\mathcal{C})$ is not trivial, as in our current proof we reduce the game to one in which we are only guaranteed that one player wins from all vertices. Moreover, since the HalfJoin operation is not closed under edge deletions, we cannot pass from winning regions to winning strategies by the simple policy of removing edges one at a time and checking in which resulting game the winning regions change.

Even though not needed in the next section, we introduce here another kind of adjoining operation between a single-player parity game P' and an arbitrary parity game P''. In this case, assuming the vertices of P' belong to Player $\overline{i}$, we fix a subset M of vertices of Player i of P'' and connect every vertex of P' with every vertex in M. These are the only arcs that may exist between P' and P''. So if $M = \emptyset$, we have the disjoint union of P' and P''.

Definition 4. Given $i \in \{\Box, \bigcirc\}$, a single-player game $P' = (V', V'_{\bigcirc}, V'_{\Box}, E', \Omega')$ with $V'_{\overline{i}} = \emptyset$ and a parity game $P'' = (V'', V''_{\bigcirc}, V''_{\Box}, E'', \Omega'')$ with $V' \cap V'' = \emptyset$, we say that a game $P = (V, V_{\bigcirc}, V_{\Box}, E, \Omega)$ is a *generalised single-player join (G-join) of P' and P''* if P is the join of P' and P'' except that the set of edges is defined differently: There exists a set $M \subseteq V''_i$ such that $E = E' \cup E'' \cup E^*$ and $E^* \subseteq (M \times V'_{\overline{i}}) \cup (V'_{\overline{i}} \times M)$ and there is at least one arc (x, y) or (y, x) for all $x \in M$, $y \in V'_{\overline{i}}$.

If $\mathcal{C}$ is a class of parity games, then we denote by $\mathsf{HalfJoin}_G(\mathcal{C})$ the class

$$\mathsf{HalfJoin}_G(\mathcal{C}) := \left\{ P \mid P \text{ is a G-join of a single-player game } P' \text{ and a game } P'' \in \mathcal{C} \right\}.$$

Theorem 1. *If $\mathcal{C}$ is a polynomial-time solvable hereditary class of parity games, then all games $P \in \mathsf{HalfJoin}_G(\mathcal{C})$ can be solved in polynomial time, provided that a decomposition of P as a G-join of a single-player game and a game in $\mathcal{C}$ is given.*

Proof. The algorithm and proof are very similar to Lemma 4, the only difference being the subgame the algorithm solves in the first case. We briefly sketch the difference here, under the same notations and assumptions as before. Let M be as given in the definition of the G-join.

In the original algorithm, Algorithm 2, the first subgame considered by the algorithm is $P \setminus \mathrm{attr}_i(V_i(P''))$, which is then solved in polynomial time because it is a single-player game. The difference for $\mathsf{HalfJoin}_G(\mathcal{C})$ is that for the first subgame we instead consider $P_1 := P \setminus \mathrm{attr}_i(M)$. Now P_1 will not be single-player game in general. However, P_1 is the disjoint union of the single player game $P_1 \cap P'$ and the game $P_1 \cap P'' \in \mathcal{C}$, so we can solve P_1 in polynomial time.

What remains is to prove that this algorithm is correct. The proof proceeds along the lines of Lemma 4, the only difference being in Case 1 because the first subgame is different.

If the set $V'_{\Box} \cap W_{\bigcirc}(P)$ is not empty, then let v be an arbitrary vertex in this set. Note that $M = N(v) \cap V''_{\bigcirc}$. By the same argument given in the proof of Lemma 4, we get $M \subseteq W_{\bigcirc}(P)$. Therefore, $W_{\Box}(P) = W_{\Box}(P_1)$, where $P_1 := P \setminus \mathrm{attr}_{\bigcirc}(M)$, which is what we needed to show.

The complementary case $V'_{\Box} \cap W_{\bigcirc}(P) = \emptyset$ is identical to Case 2 in the proof of Lemma 4, and the result immediately follows. □

3.2. Joining two parity games

Now we can state our main theorem on the join of two parity games.

Theorem 2. *If $\mathcal{C}$ and $\mathcal{C}'$ are polynomial-time solvable hereditary classes of parity games, then there is a polynomial-time algorithm for solving any $P \in \mathsf{Join}(\mathcal{C}, \mathcal{C}')$, assuming a decomposition of P as a join of a game in $\mathcal{C}$ and a game in $\mathcal{C}'$ is given.*

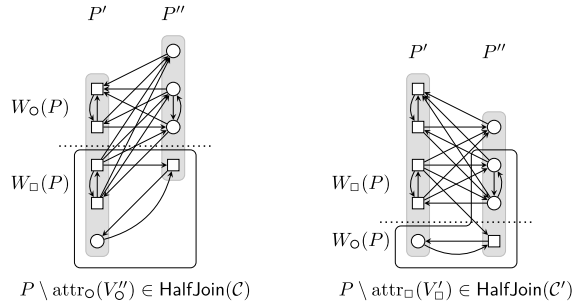
Proof. Let $P = (V, V_{\bigcirc}, V_{\Box}, E, \Omega) \in \mathsf{Join}(\mathcal{C}, \mathcal{C}')$ be a join of a game $P' = (V', V'_{\bigcirc}, V'_{\Box}, E', \Omega') \in \mathcal{C}$ with a game $P'' = (V'', V''_{\bigcirc}, V''_{\Box}, E'', \Omega'') \in \mathcal{C}'$. Let $O(n^{c_1})$ and $O(n^{c_2})$ be the time complexities for solving games in $\mathcal{C}$ and in $\mathcal{C}'$, respectively, where n is the number of nodes.

We slightly adapt Algorithm 2 to the problem at hand. Instead of calling Solve-Single-Player-Game, we solve $P \setminus \mathrm{attr}_i(V_i(P''))$ by applying Lemma 4 to the class $\mathsf{HalfJoin}(\mathcal{C})$. Similarly, instead of calling Solve-$\mathcal{C}$-Game, we solve $P \setminus \mathrm{attr}_{\overline{i}}(V_{\overline{i}}(P'))$ by applying Lemma 4 to $\mathsf{HalfJoin}(\mathcal{C}')$.

In both cases we can easily satisfy the requirement of Lemma 4 that the decomposition needs to be given: By assumption, we have a decomposition of P into $P' \in \mathcal{C}$ and $P'' \in \mathcal{C}'$. Then $P \setminus \mathrm{attr}_i(V_i(P''))$ is the game P' joined to the single-player game $(P \setminus \mathrm{attr}_i(V_i(P''))) \setminus P'$, and similar for the other game.

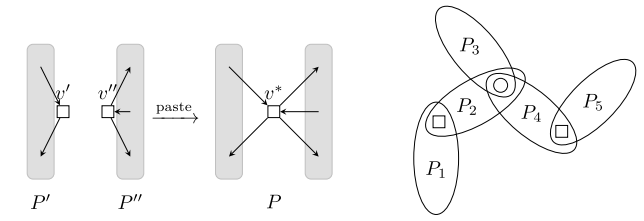
Now, for the correctness proof we follow the lines of the proof of Lemma 4. The case distinctions and their conclusions are exactly the same. The only difference is that in the proof of Lemma 4, the two subgames $P \setminus \mathrm{attr}_{\bigcirc}(V''_{\bigcirc})$ and $P \setminus \mathrm{attr}_{\Box}(V'_{\Box})$ were solvable in polynomial time because they were single-player games or in $\mathcal{C}$, respectively. Here these games can be solved in time $O(n^{1+\max\{3,c_1,c_2\}})$ by Lemma 4 because both games are in $\mathsf{HalfJoin}(\mathcal{C})$ and $\mathsf{HalfJoin}(\mathcal{C}')$, respectively. See Fig. 3 and note the similarity to Figs. 1(a) and 1(b). Note that in contrast to the other figures, in Fig. 3(a) the winning region for Player $\bigcirc$ is at the top.

The remaining part of the algorithm is exactly the same as in the proof of Lemma 4: If neither of these subgames provides a usable winning region, then we have $W_{\Box}(P) = \emptyset$ or $W_{\bigcirc}(P) = \emptyset$ and we return the original game unmodified.



(a) Case 1a (b) Case 2a

Fig. 3. An illustration of the proof of Theorem 2, where P is a join of $P' \in \mathcal{C}$ and $P'' \in \mathcal{C}'$.



(a) The paste of P' and P'' at v', v'' (b) A game from $\mathsf{RepeatedPasting}(\mathcal{C})$

Fig. 4. The paste operation.

In total this gives a running time of $O(n^{1+\max\{3,c_1,c_2\}})$ for the preliminary algorithm. Lemma 3 turns this into an algorithm that solves all parity games in $\mathsf{Join}(\mathcal{C},\mathcal{C}')$ in time $O(n^{2+\max\{3,c_1,c_2\}})$. □

We remark that if $\mathcal{C}$ is the class of parity games without arcs, then the class $\mathsf{Join}(\mathcal{C},\mathcal{C})$ contains the class of parity games whose underlying graph is a biorientation of a complete bipartite graph.

It is a little unfortunate that for the above results, the decomposition needs to be given. It seems that $\mathcal{C}$ being hereditary and polynomial-time decidable does not provide sufficient structure to compute the decompositions. However, we have no proof for this. As it stands, this restriction makes the results most applicable to cases where the decomposition is part of the construction and readily available.

4. Pasting of parity games

Definition 5. Let $P' = (V', V'_{\bigcirc}, V'_{\square}, E', \Omega')$ and $P'' = (V'', V''_{\bigcirc}, V''_{\square}, E'', \Omega'')$ be two parity games with $V' \cap V'' = \emptyset$, and let $v' \in V'$ and $v'' \in V''$. Assume that v', v'' have the same priority and belong to the same player in P', and in P'', respectively. We say that the game $P = (V, V_{\bigcirc}, V_{\square}, E, \Omega)$ is obtained by *pasting* P', P'' *at* v', v'', if P is the disjoint copy of P' and P'' with v', v'' identified (see Fig. 4(a)). Formally, assuming that v', v'' belong to Player i,

- $V = (V' \cup V'' \cup \{v^*\}) \setminus \{v', v''\}$, $V_i = (V'_i \cup V''_i \cup \{v^*\}) \setminus \{v', v''\}$, $V_{\bar{i}} = V'_{\bar{i}} \cup V''_{\bar{i}}$,
- $E = \{(u,w) \in E' \cup E'' \mid \{v', v''\} \cap \{u,w\} = \emptyset\} \cup$
  $\{(u, v^*) \mid (u, v') \in E' \vee (u, v'') \in E''\} \cup$
  $\{(v^*, u) \mid (v', u) \in E' \vee (v'', u) \in E''\}$,
- the vertices of P have the same priorities as they have in P' and P'', where $\Omega(v^*) = \Omega(v') = \Omega(v'')$.

Given a class of parity games $\mathcal{C}$, we denote by $\mathsf{RepeatedPasting}(\mathcal{C})$ the class of games obtained by repeated pasting of a finite number of games from $\mathcal{C}$ and then closing this class under disjoint unions.

We observe that if $\mathcal{C}$ is hereditary and closed under disjoint unions, then $\mathsf{RepeatedPasting}(\mathcal{C})$ is hereditary. Moreover, every $P \in \mathsf{RepeatedPasting}(\mathcal{C})$ has a decomposition into components $P_1, P_2, \ldots, P_k$ with $P_i \in \mathcal{C}$, such that every distinct P_i and P_j are either disjoint or share exactly one vertex. This means that the games $P_1, \ldots, P_k$ form a tree-like structure, in the sense that the graph T_P obtained by adding a vertex i for every P_i and an edge (i,j) if P_i and P_j share a vertex is a tree (see Fig. 4(b)).

Note that if a hereditary class $\mathcal{C}$ is polynomial-time solvable, its closure under disjoint union is also polynomial-time solvable because disjoint games can be solved separately.

Theorem 3. *If $\mathcal{C}$ is a hereditary class of parity games that is $O(n^c)$-solvable, then* RepeatedPasting($\mathcal{C}$) *is $O(n^{1+\max\{2,c\}})$-solvable.*

Proof. Let $P \in \mathsf{RepeatedPasting}(\mathcal{C})$ and let $T(n)$ be the running time of the algorithm we are going to construct for RepeatedPasting($\mathcal{C}$), where n is the number of vertices of P. We assume that P is connected because we can solve disjoint games separately.

In order to find a decomposition of P into games from $\mathcal{C}$, we compute the biconnected components (that is, maximal 2-connected subgraphs) of the underlying undirected graph of P, say $P_1, \ldots, P_k$, in time linear in the number of edges of P, for example by the algorithm presented in [18]. Since $P \in \mathsf{RepeatedPasting}(\mathcal{C})$ and $\mathcal{C}$ is hereditary, each such biconnected component belongs to $\mathcal{C}$.

Let $L \in \mathcal{C}$ be a leaf-component of the tree T_P associated with P. This means that L shares at most one vertex with all other components of P. If there is no such vertex, we are done because the graph is disconnected and we can easily solve different components separately. Otherwise, let v be this vertex. Without loss of generality we assume that v is a vertex that belongs to Player $\bigcirc$. Since all paths between L and $P' := P \setminus (L \setminus \{v\})$ use vertex v, and thus there are no cycles without repeated vertices spanning both L and P', we have that $v \in W_\bigcirc(P)$ if and only if $v \in W_\bigcirc(L)$ or $v \in W_\bigcirc(P')$.

First we solve the parity game L, which can be done in time $O(n^c)$ since $L \in \mathcal{C}$. If Player $\bigcirc$ wins on v in L, then we solve $P \setminus (L \cup \mathrm{attr}_\bigcirc(v))$ recursively, in time $T(n-1)$. Since all paths between L and $P \setminus (L \cup \mathrm{attr}_\bigcirc(v))$ use vertex v, from which Player $\bigcirc$ has a winning strategy inside L, we have, thanks to Lemma 2:

- $W_\bigcirc(P) = W_\bigcirc(L) \cup \mathrm{attr}_\bigcirc(v) \cup W_\bigcirc(P \setminus (L \cup \mathrm{attr}_\bigcirc(v)))$,
- $W_\square(P) = W_\square(L) \cup W_\square(P \setminus (L \cup \mathrm{attr}_\bigcirc(v)))$.

However, if Player $\square$ wins on v in L, then we solve P' recursively, in time $T(n-1)$. Then we have two cases. If Player $\square$ wins on v in P', then we merge the winning regions of L and P', that is:

- $W_\bigcirc(P) = W_\bigcirc(L) \cup W_\bigcirc(P')$,
- $W_\square(P) = W_\square(L) \cup W_\square(P')$.

This is correct because no matter which successor Player $\bigcirc$ chooses on v, the game will either continue in L or in P', and $v \in W_\square(L)$ and $v \in W_\square(P')$. It is sufficient to consider a fixed choice of successor because parity games are positionally determined.

Otherwise, if Player $\bigcirc$ wins on v in P', then we have to re-compute the winning regions of L, from which we remove $\mathrm{attr}_\bigcirc(v)$. Since $\mathcal{C}$ is hereditary, we have $L \setminus \mathrm{attr}_\bigcirc(v) \in \mathcal{C}$, which we can solve in time $O(n^c)$. Thus, again by Lemma 2 and by the fact that all paths between $L \setminus \mathrm{attr}_\bigcirc(v)$ and P' use vertex v, from which Player $\bigcirc$ has a winning strategy inside P', we put:

- $W_\bigcirc(P) = W_\bigcirc(L \setminus \mathrm{attr}_\bigcirc(v)) \cup \mathrm{attr}_\bigcirc(v) \cup W_\bigcirc(P')$,
- $W_\square(P) = W_\square(L \setminus \mathrm{attr}_\bigcirc(v)) \cup W_\square(P')$.

Finally, the running time of the algorithm satisfies

$$T(n) \leq n^2 + tn^c + T(n-1),$$

for some fixed $t > 1$, where n^2 accounts for the computation of the attractor sets. Thus, $T(n) \in O(n^{1+\max\{2,c\}})$. □

As a corollary of Corollary 1 and Theorem 3, we can solve parity games in polynomial time on any biorientation of a *block-cactus graph* [7], that is, a graph whose biconnected components are cliques or cycles.

On the other hand, it is unlikely that we can easily extend the above method by pasting along 2 vertices since this immediately leads to a polynomial-time algorithm for the class of all parity games by pasting along complete graphs with 2 vertices.

5. Adding a single vertex

Definition 6. If $\mathcal{C}$ is a class of parity games, we let AddVertex($\mathcal{C}$) denote the class of parity games obtained by adding a single vertex to every game in $\mathcal{C}$ in any possible way. Formally,

$$\mathsf{AddVertex}(\mathcal{C}) := \{P \mid P \text{ is a parity game and there exists a vertex } v \text{ such that } P \setminus \{v\} \in \mathcal{C}\}.$$

Theorem 4. *If $\mathcal{C}$ is a hereditary class of parity games that is $O(n^c)$-solvable and $O(n^d)$-decidable with $c \geq 2$, then* AddVertex($\mathcal{C}$) *is $O(n^{\max\{c,d\}+1})$-solvable and $O(n^{d+1})$-decidable.*

Proof. Let $P = (V, V_\bigcirc, V_\Box, E, \Omega)$. In order to find a vertex v such that $P \setminus \{v\} \in \mathcal{C}$, and at the same time test whether $P \in \mathsf{AddVertex}(\mathcal{C})$, we can iterate over all $v \in V$ and test the membership of $P \setminus \{v\}$ to $\mathcal{C}$, with an overall complexity of $O(n^{d+1})$.

First, we solve the two subgames $P_1 := P \setminus \mathrm{attr}_\bigcirc(v) \in \mathcal{C}$ and $P_2 := P \setminus \mathrm{attr}_\Box(v) \in \mathcal{C}$ in time $O(n^c)$. If $W_\Box(P_1)$ or $W_\bigcirc(P_2)$ is not empty, we return the corresponding subgame $P \setminus P_1$ or $P \setminus P_2$.

If $W_\Box(P_1) = W_\bigcirc(P_2) = \emptyset$, we claim that $W_\Box(P) = \emptyset$ or $W_\bigcirc(P) = \emptyset$. We can then return the original game unchanged. To establish the claim, assume to the contrary that $W_\Box(P) \neq \emptyset$ and $W_\bigcirc(P) \neq \emptyset$. Because of positional determinacy, we have two cases: $v \in W_\Box(P)$ or $v \in W_\bigcirc(P)$.

The two cases are analogous, so let us assume $v \in W_\Box(P)$. Then we have

$$W_\bigcirc(P) = W_\bigcirc(P \setminus \mathrm{attr}_\Box(v)) = W_\bigcirc(P_2) = \emptyset$$

by Lemma 2. This contradicts the assumption $W_\bigcirc(P) \neq \emptyset$.

The running time of this part is $O(n^c)$ because computing the attractor sets is in $O(n^2) \subseteq O(n^c)$. Lemma 3 then yields an algorithm that solves all games in $\mathsf{AddVertex}(\mathcal{C})$ in time $O(n^{c+1})$. Finding the vertex v that we have to remove takes time $O(n^{d+1})$, so the total running time is $O(n^{\max\{c,d\}+1})$. □

This theorem implies, for example, that if parity games can be solved in polynomial time on orientations of planar graphs, then they can also be solved in polynomial time on orientations of apex graphs, which are planar graphs with one additional vertex.

6. Conclusions

We presented some graph operations that preserve solvability of parity games in polynomial time. In Section 3.2, we saw that the join of two classes of parity games is as easy to solve as the individual classes up to a polynomial factor provided a decomposition of the join is available. In Section 4 we considered the case of pasting many games together along vertices to form a larger game and in Section 5 we analysed the problem of adding a single vertex to a parity game. In both cases we showed that the resulting classes can be solved in time only a small polynomial factor slower than the original classes.

It is an open problem whether our approach can be adapted to further graph operations. One graph operations that comes to mind is the operation of *substitution*. Particular instances of this operation are obtained by starting from a tree T whose nodes are coloured with either ○ or □ such that no two adjacent nodes have the same colour. Then replace every i-coloured node with a single-player game consisting of nodes of player i and connect games that correspond to adjacent nodes in T analogous to the Join-operation defined in Section 3. The resulting game can be solved with dynamic programming and the help of Theorem 2 in time $O(n^c)$, but c depends on the depth of the tree T. These games seem to be simpler than general parity games, so it could be reasonable to expect a polynomial time algorithm.

On the other hand, it is interesting to study whether the graph operations considered here preserve the polynomial time solvability of other games. A starting point could be parity games whose *edges* are assigned priorities; more generally, one could consider some of our operations for mean payoff games, energy games, or simple stochastic games.

Acknowledgements

We thank the anonymous reviewers for their helpful comments.

References

[1] C. Baier, J.P. Katoen, Principles of Model Checking, MIT Press, 2008.
[2] J. Bang-Jensen, G. Gutin, Digraphs: Theory, Algorithms and Applications, second ed., Springer Monographs in Mathematics, Springer-Verlag, London, 2008.
[3] D. Berwanger, K. Chatterjee, M.D. Wulf, L. Doyen, T.A. Henzinger, Strategy construction for parity games with imperfect information, Inform. and Comput. 208 (2010) 1206–1220, http://dx.doi.org/10.1016/j.ic.2009.09.006.
[4] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, DAG-width and parity games, in: B. Durand, W. Thomas (Eds.), Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science, STACS, Marseille, France, February 23–25, 2006, Springer, 2006, pp. 524–536.
[5] D. Berwanger, E. Grädel, Entanglement – a measure for the complexity of directed graphs with applications to logic and games, in: F. Baader, A. Voronkov (Eds.), Proceedings of 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR, Montevideo, Uruguay, March 14–18, 2005, Springer, 2005, pp. 209–223.
[6] H. Björklund, S. Sandberg, S.G. Vorobyov, A discrete subexponential algorithm for parity games, in: H. Alt, M. Habib (Eds.), Proceedings of 20th Annual Symposium on Theoretical Aspects of Computer Science, STACS, Berlin, Germany, February 27–March 1, 2003, Springer, 2003, pp. 663–674.
[7] A. Brandstädt, V.B. Le, J.P. Spinrad, Graph Classes: A Survey, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.
[8] K. Chatterjee, M. Henzinger, S. Krinninger, D. Nanongkai, Polynomial-time algorithms for energy games with special weight structures, in: L. Epstein, P. Ferragina (Eds.), Proceedings of the 20th Annual European Symposium on Algorithms, ESA, Ljubljana, Slovenia, September 10–12, 2012, Springer, 2012, pp. 301–312.
[9] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, 1999.
[10] E.A. Emerson, C.S. Jutla, Tree automata, mu-calculus and determinacy, in: 32nd Annual Symposium on Foundations of Computer Science, FOCS, San Juan, Puerto Rico, 1–4 October 1991, IEEE Computer Society, 1991, pp. 368–377.
[11] E.A. Emerson, C.S. Jutla, A.P. Sistla, On model-checking for fragments of μ-calculus, in: C. Courcoubetis (Ed.), Proceedings of the 5th International Conference on Computer Aided Verification, CAV, Elounda, Greece, June 28–July 1, 1993, Springer, 1993, pp. 385–396.

[12] J. Fearnley, S. Schewe, Time and parallelizability results for parity games with bounded treewidth, in: A. Czumaj, K. Mehlhorn, A.M. Pitts, R. Wattenhofer (Eds.), Proceedings of the 39th International Colloquium on Automata, Languages, and Programming, Part II, ICALP, Warwick, UK, July 9–13, 2012, Springer, 2012, pp. 189–200.
[13] O. Friedmann, An exponential lower bound for the parity game strategy improvement algorithm as we know it, in: Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science, LICS, Los Angeles, CA, USA, August 11–14, 2009, IEEE Computer Society, 2009, pp. 145–156.
[14] O. Friedmann, M. Lange, Solving parity games in practice, in: Z. Liu, A. Ravn (Eds.), Automated Technology for Verification and Analysis, in: Lecture Notes in Computer Science, vol. 5799, Springer, Berlin/Heidelberg, 2009, pp. 182–196.
[15] E. Grädel, W. Thomas, T. Wilke (Eds.), Automata, Logics, and Infinite Games: A Guide to Current Research, Lecture Notes in Computer Science, vol. 2500, Springer, 2002.
[16] M. Grohe, Computing crossing numbers in quadratic time, J. Comput. System Sci. 68 (2004) 285–302, http://dx.doi.org/10.1016/j.jcss.2003.07.008.
[17] F. Harary, Graph Theory, Addison–Wesley Series in Mathematics, Perseus Books, 1994.
[18] J. Hopcroft, R. Tarjan, Algorithm 447: efficient algorithms for graph manipulation, Commun. ACM 16 (1973) 372–378, http://dx.doi.org/10.1145/362248.362272.
[19] M. Jurdziński, Deciding the winner in parity games is in UP ∩ co-UP, Inform. Process. Lett. 68 (1998) 119–124, http://dx.doi.org/10.1016/S0020-0190(98)00150-1.
[20] M. Jurdziński, Small progress measures for solving parity games, in: H. Reichel, S. Tison (Eds.), Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science, STACS, Lille, France, February 2000, Springer, 2000, pp. 290–301.
[21] M. Jurdziński, M. Paterson, U. Zwick, A deterministic subexponential algorithm for solving parity games, in: Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22–26, 2006, ACM Press, 2006, pp. 117–123.
[22] R. McNaughton, Infinite games played on finite graphs, Ann. Pure Appl. Logic 65 (1993) 149–184, http://dx.doi.org/10.1016/0168-0072(93)90036-D.
[23] J. Obdržálek, Fast mu-calculus model checking when tree-width is bounded, in: W.A.H. Jr., F. Somenzi (Eds.), Proceedings of the 15th International Conference on Computer Aided Verification, CAV, Boulder, CO, USA, July 8–12, 2003, Springer, 2003, pp. 80–92.
[24] J. Obdržálek, Clique-width and parity games, in: J. Duparc, T.A. Henzinger (Eds.), Proceedings of the 21st International Workshop on Computer Science Logic, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11–15, 2007, Springer, 2007, pp. 54–68.
[25] N. Robertson, P. Seymour, Graph minors I–XXIII, J. Combin. Theory Ser. B (1982–2011).
[26] N. Robertson, P. Seymour, Graph minors XVI. Excluding a non-planar graph, J. Combin. Theory Ser. B 77 (1999) 1–27, http://dx.doi.org/10.1016/S0095-8956(03)00042-X.
[27] N. Robertson, P.D. Seymour, Graph minors XIII. the disjoint paths problem, J. Combin. Theory Ser. B 63 (1995) 65–110, http://dx.doi.org/10.1006/jctb.1995.1006.
[28] J. Vöge, M. Jurdziński, A discrete strategy improvement algorithm for solving parity games, in: E.A. Emerson, A.P. Sistla (Eds.), Proceedings of the 12th International Conference on Computer Aided Verification, CAV, Chicago, IL, USA, July 15–19, 2000, Springer, 2000, pp. 202–215.
[29] W. Zielonka, Infinite games on finitely coloured graphs with applications to automata on infinite trees, Theoret. Comput. Sci. 200 (1998) 135–183, http://dx.doi.org/10.1016/S0304-3975(98)00009-7.