






# Connectivity in the Presence of an Opponent

Zihui Liang<sup>1</sup>  



University of Electronic Science and Technology of China, Chengdu, China

Bakh Khoussainov<sup>1</sup> 

University of Electronic Science and Technology of China, Chengdu, China

Toru Takisaka  

University of Electronic Science and Technology of China, Chengdu, China

Mingyu Xiao  

University of Electronic Science and Technology of China, Chengdu, China

---

## Abstract

The paper introduces two player connectivity games played on finite bipartite graphs. Algorithms that solve these connectivity games can be used as subroutines for solving Müller games. Müller games constitute a well established class of games in model checking and verification. In connectivity games, the objective of one of the players is to visit every node of the game graph infinitely often. The first contribution of this paper is our proof that solving connectivity games can be reduced to the incremental strongly connected component maintenance (ISCCM) problem, an important problem in graph algorithms and data structures. The second contribution is that we non-trivially adapt two known algorithms for the ISCCM problem to provide two efficient algorithms that solve the connectivity games problem. Finally, based on the techniques developed, we recast Horn's polynomial time algorithm that solves explicitly given Müller games and provide the first correctness proof of the algorithm. Our algorithms are more efficient than that of Horn's algorithm. Our solution for connectivity games is used as a subroutine in the algorithm.

**2012 ACM Subject Classification** Theory of computation → Logic and verification; Theory of computation → Complexity theory and logic; Theory of computation → Dynamic graph algorithms

**Keywords and phrases** Explicit Müller games, games played on finite graphs, winning strategies, synthesis and analysis of games

**Digital Object Identifier** 10.4230/LIPIcs.ESA.2023.79

**Funding** *Bakh Khoussainov*: Bakh Khoussainov acknowledges the National Science Foundation of China under Grant No.62172077.

## 1 Introduction

### 1.1 Müller games given explicitly

In the area of logic, model checking, and verification of reactive systems, studying games played on graphs is a key research topic [10]. This is mostly motivated through modelling reactive systems and reductions of model checking problems to games on graphs. Understanding the algorithmic content of determinacy results is also at the core of this research. Müller games constitute a well-established class of games for verification. Recall that a *Müller game*  $\mathcal{G}$  is a tuple  $(V_0, V_1, E, \Omega)$ , where

- The tuple  $G = (V_0 \cup V_1, E)$  is a finite directed bipartite graph so that  $V_0$  and  $V_1$  partition the set  $V = V_0 \cup V_1$ . Usually  $G$  is called the arena of  $\mathcal{G}$ .
- The set  $E \subseteq (V_0 \times V_1) \cup (V_1 \times V_0)$  of edges.

---

<sup>1</sup> Corresponding authors.



- $V_0$  and  $V_1$  are sets from which player 0 and player 1, respectively, move. Positions in  $V_\sigma$  are called player  $\sigma$  positions,  $\sigma \in \{0, 1\}$ .
- $\Omega \subseteq 2^V$  is a collection of winning sets.

Say that the game  $\mathcal{G} = (V_0, V_1, E, \Omega)$  is *explicitly given* if  $V$ ,  $E$ , and all sets in  $\Omega$  are fully presented as input. The (input) size of explicitly given Müller game is thus bounded by  $|V| + |E| + 2^{|V|} \cdot |V|$ . Finally, the *game graph* of the Müller game  $\mathcal{G}$  is the underlying bipartite graph  $G = (V_0 \cup V_1, E)$ .

For each  $v \in V$ , let  $E(v) = \{u \mid E(v, u)\}$  be the set of successors of  $v$ . Let  $X \subseteq V$ . Call the set  $E(X) = \bigcup_{v \in X} E(v)$  the successor of  $X$ . Similarly, for a  $v \in V$ , the predecessor of  $v$  is the set  $E^{-1}(v) = \{u \mid (u, v) \in E\}$ . Call the set  $E^{-1}(X) = \bigcup_{v \in X} E^{-1}(v)$  the predecessor of  $X$ .

Let  $\mathcal{G} = (V_0, V_1, E, \Omega)$  be a Müller game. The players play the game by moving a given token along the edges of the graph. The token is initially placed on a node  $v_0 \in V$ . The play proceeds in rounds. At any round of the play, if the token is placed on a player  $\sigma$ 's node  $v$ , then player  $\sigma$  chooses  $u \in E(v)$ , moves the token to  $u$  and the play continues on to the next round. Formally, a play (starting from  $v_0$ ) is a sequence  $\rho = v_0, v_1, \dots$  such that  $v_{i+1} \in E(v_i)$  for all  $i \in \mathbb{N}$ . If a play reaches a position  $v$  such that  $E(v) = \emptyset$ , then player 1 wins the play. For an infinite play  $\rho$ , set  $\text{Inf}(\rho) = \{v \in V \mid \exists^\omega i (v_i = v)\}$ . We say player 0 wins the play  $\rho$  if  $\text{Inf}(\rho) \in \Omega$ ; otherwise, player 1 wins the play.

A strategy for player  $\sigma$  is a function that takes as inputs initial segments of plays  $v_0, v_1, \dots, v_k$  where  $v_k \in V_\sigma$  and output some  $v_{k+1} \in E(v_k)$ . A strategy for player  $\sigma$  is winning from  $v_0$  if, assuming player  $\sigma$  follows the strategy, all plays starting from  $v_0$  generated by the players are winning for player  $\sigma$ . The game  $\mathcal{G}$  is determined if one of the players has a winning strategy. Müller games are Borel games, and hence, by the result of Martin [18], they are determined. Since Müller games are determined we can partition the set  $V$  onto two sets  $\text{Win}_0$  and  $\text{Win}_1$ , where  $v \in \text{Win}_\sigma$  iff player  $\sigma$  wins the game starting at  $v$ ,  $\sigma \in \{0, 1\}$ . To solve a given Müller game  $\mathcal{G} = (V_0, V_1, E, \Omega)$  means to find the sets  $\text{Win}_0$  and  $\text{Win}_1$ . There are several known algorithms that solve Müller games. These algorithms provide the basis for analysis and synthesis of Müller games. In particular, these algorithms extract finite state winning strategies for the players [8, 12, 13, 19, 20, 22]. Also, efficiency of algorithms depend on the underlying structure of graphs [17] [9]. We stress that the algorithms that solve Müller games depend on the presentations of the games. The problem of solving Müller games is typically in PSPACE for many reasonable representations [19, 20]. However, if the winning condition is represented as a Zielonka tree [22] or as the well-known parity condition, then solving the games turns into a  $NP \cap co-NP$  problem [5]. P. Hunter and A. Dawar [13] investigate five other representations: win-set, Muller, Zielonka DAGs, Emerson-Lei, and explicit Muller. They show that the problem of the winner is PSPACE-hard for the first four representations. F. Horn [12] provides a polynomial time algorithm that solves explicit Müller games. However, his proof of correctness has non-trivial flaws. So, we provide an alternative correctness proof based on ideas totally independent of Horn's. Designing new algorithms, improving and analysing the state of the art techniques in this area is a key research direction. This paper contributes to this.

## 1.2 Connectivity games

One motivation for defining connectivity games comes from solving Müller games. Many algorithms that solve Müller games or its variants are recursive. Given a Müller game  $\mathcal{G}$ , one constructs a set of smaller Müller games. The solution of the games  $\mathcal{G}'$  from this set is then used to solve  $\mathcal{G}$ . Through an iteration process, these reductions produce sequences of the

form  $\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_r$ , where  $\mathcal{G}_{i+1} = \mathcal{G}'_i$  such that  $\mathcal{G}'_r = \mathcal{G}_{r+1}$ . The key point is that solving the game  $\mathcal{G}_r$  at the base of this iteration boils down to investigating connectivity of the graph  $G_r$  in the game-theoretic setting. Namely, to win the game  $\mathcal{G}_r$ , one of the players has to visit all the nodes of  $\mathcal{G}_r$  infinitely often. This observation calls for deeper and refined analysis of those Müller games  $\mathcal{G} = (V_0, V_1, E, \Omega)$  where the objective of player 0 is to visit all the nodes of the underlying graph  $G$ , that is,  $\Omega = \{V\}$ . We single out these games:

► **Definition 1.** A Müller game  $\mathcal{G} = (V_0, V_1, E, \Omega)$  is called a **connectivity game** if  $\Omega$  is a singleton that consists of  $V$ .

The second motivation to investigate the connectivity games comes from the concept of connectivity itself. The notion of (vertex) connectivity is fundamental in graph theory and its applications. There is a large amount of work ranging from complexity theoretic issues to designing efficient data structures and algorithms that aim to analyse connectivity in graphs. Connectivity in graphs and graph like structures is well-studied in almost all areas of computer science in various settings and motivations. For undirected graphs, connectivity of a graph  $G$  is defined through existence of paths between all vertices of  $G$ . For directed graphs  $G$  connectivity is defined through strong connectivity. The digraph  $G$  is strongly connected if for any two vertices  $x$  and  $y$  there exist paths from  $x$  to  $y$  and from  $y$  to  $x$ . One can extend these notions of (vertex) connectivity into a game-theoretic setting as follows. There are two players: player 0 and player 1. A token is placed on a vertex  $v_0$  of a bipartite graph  $G = (V_0 \cup V_1, E)$ . Player 0 starts the play by moving the token along an outgoing edge  $(v_0, v_1)$ . Player 1 responds by moving the token along an outgoing edge from the vertex  $v_1$ , say  $(v_1, v_2)$ . This continues on and the players produce a path  $v_0, v_1, \dots, v_k$  called a play starting at  $v_0$ . Say that player 0 wins the play  $v_0, v_1, \dots, v_k$  if the play visits every node in  $V$ . Call thus described game *forced-connectivity game*. A possible scenario for this situation is that player 0 wants to pass a message through all the nodes of a given network in the presence of an adversary. If player 0 has a winning strategy, then we say that the player wins the game starting at  $v_0$ . Winning this forced-connectivity game from  $v_0$  does not always guarantee that the player wins the game starting at any other vertex. Therefore we can define game-theoretic connectivity as follows. A directed bipartite graph  $G$  is *forced-connected* if player 0 wins the forced-connectivity game in  $G$  starting at any vertex of  $G$ . Thus, finding out if  $G$  is a forced-connected is equivalent to solving connectivity games as in Definition 1.

► **Definition 2.** Let  $\mathcal{G} = (V_0, V_1, E, \Omega)$  be a connectivity game. Call the bipartite graph  $G = (V_0, V_1, E)$  **forced-connected** if player 0 wins the game  $\mathcal{G}$ .

The third motivation is related to generalised Büchi winning condition. The generalised Büchi winning condition is given by subsets  $F_1, \dots, F_k$  of the game graph  $G$ . Player 0 wins a play if the play meets each of these winning sets  $F_1, \dots, F_k$  infinitely often. Our connectivity games winning condition can be viewed as a specific generalised Büchi winning condition where the accepting sets are all singletons.

### 1.3 Our contributions

The focus of this paper is two-fold. On the one hand, we study connectivity games and provide the state-of-the-art algorithms for solving them. H. Bodlaender, M. Dinneen, and B. Khossainov [3, 4] call connectivity games *update games*. Their motivation comes from modelling the scenario where messages should be passed to all the nodes of the network in the presence of adversary. On the other hand, using the connectivity game solution process as a subroutine, we recast Horn's polynomial time algorithm that solves explicitly given Müller games and provide a proof of its correctness. We detail these below.

1. Our first contribution is that given a connectivity game  $\mathcal{G}$ , we construct a sequence of directed graphs  $\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_s$  such that player 0 wins  $\mathcal{G}$  if and only if  $\mathcal{G}_s$  is strongly connected [See Theorem 8]. Due to this result, we reduce solving connectivity game problem to the incremental strongly connected component maintenance (ISCCM) problem, one of the key problems in graph algorithms and data structure analysis [1, 11].
2. A standard brute-force algorithm that solves the connectivity game  $\mathcal{G}$  runs in time  $\mathbf{O}(|V|^2(|V| + |E|))$ . H. Bodlaender, M. Dinneen, and B. Khoussainov in [3, 4, 7, 16] provided algorithms that solve the connectivity games in  $\mathbf{O}(|V||E|)$ . Due to Theorem 8, we solve the connectivity game problem by adapting two known algorithms that solve the ISCCM problem. The first algorithm is by Haeupler et al. [11] who designed the *soft-threshold search algorithm* that handles sparse graphs. Their algorithm runs in time  $\mathbf{O}(\sqrt{mm})$ , where  $m$  is the number of edges. The second is the solution by Bender et al. [1, 2]. Their algorithm is best suited for the class of dense graphs and runs in time of  $\mathbf{O}(n^2 \log n)$ , where  $n$  is the number of vertices. By adapting these algorithms, we design new algorithms to solve the connectivity games. The first algorithm, given a connectivity game  $\mathcal{G}$ , runs in time  $\mathbf{O}((\sqrt{|V_1|} + 1)|E| + |V_1|^2)$  [See Theorem 9]. The first feature of this algorithm is that the algorithm solves the problem in linear time in  $|V_0|$  if  $|V_1|$  is considered as a parameter. The parameter constant in this case is  $|V_1|^{3/2}$ . The second feature is that the algorithm runs in linear time if the underlying game graph is sparse. Our second algorithm solves the connectivity game in time  $\mathbf{O}((|V_1| + |V_0|) \cdot |V_0| \log |V_0|)$  [See Theorem 10]. In contrast to the previous algorithm, this algorithm solves the connectivity game problem in linear time in  $|V_1|$  if  $|V_0|$  is considered as a parameter. The parameterised constant is  $|V_0| \log |V_0|$ . Furthermore, the second algorithm is more efficient than the first one on dense graphs. These two algorithms outperform the standard bound  $O(|V||E|)$ , mentioned above, for solving the connectivity games. As a framework, this is similar to the work of K. Chatterjee and M. Henzinger [6] who improved the standard  $O(|V||E|)$  time bound for solving Büchi games to  $O(|V|^2)$  bound through analysis of maximal end-component decomposition algorithms.
3. In [12] Horn provided a polynomial time algorithm that solves explicitly given Müller games. In his algorithm, Horn uses the standard procedure of solving connectivity games as a subroutine. Directly using our algorithms above, as a subroutine to Horn's algorithm, we obviously improve Horn's algorithm in an order of magnitude. Horn's proof of correctness uses three lemmas (see Lemmas 5, 6, and 7 in [12]). However, his Lemmas 6 and 7 contain non-trivial flaws. We provide our independent proof of correctness. In terms of ideas, our proof is completely different from Horn's proof ideas. We discuss these in Section 6. To the best of our knowledge, this is the first work that correctly and fully recasts Horn's polynomial time algorithm with the efficient sub-routine for solving the connectivity games. Furthermore, in terms of running time, our algorithms perform better than that of Horn's algorithm [See Theorem 20 and Theorem 21]. For instance, one of our algorithms decreases the degree of  $|\Omega|$  from  $|\Omega|^3$  in Horn's algorithm to  $|\Omega|^2$  [See Theorem 21]. Since  $|\Omega|$  is bounded by  $2^{|V|}$ , the improvement is significant.

## 2 A characterization theorem

A Müller game  $\mathcal{G} = (V_0, V_1, E, \Omega)$  is a connectivity game if  $\Omega = \{V\}$ . In this section we focus on connectivity games  $\mathcal{G}$ . In the study of Müller games, often it is required that for each  $v$  the successor set  $E(v) = \{u \mid (v, u) \in E\}$  is not empty. We do not put this condition as it will be convenient for our analysis of connectivity games to consider cases

when  $E(v) = \emptyset$ . Recall that a strongly connected component of a directed graph is a maximal set  $X$  such that there exists a path between any two vertices of  $X$ . Denote the collection of all strongly connected components of the game graph  $G$  of game  $\mathcal{G}$  by  $SCC(\mathcal{G})$ . For all distinct components  $X, Y \in SCC(\mathcal{G})$ , we have  $X \cap Y = \emptyset$  and  $\bigcup_{X \in SCC(\mathcal{G})} X = V$ .

► **Definition 3.** Let  $\mathcal{G}$  be a connectivity game. Consider two sets  $U \subseteq V_1$  and  $S \subseteq V$ . Define

$$Force(U, S) = \{v \mid v \in (E^{-1}(S) \setminus S) \cap U \text{ and } E(v) \subseteq S\}.$$

► **Definition 4.** We say that a set  $X \subseteq V$  in game  $\mathcal{G}$  is a **forced trap (FT)** if either  $|X| = 1$  or if  $|X| > 1$  then  $E(X \cap V_1) \subseteq X$  and  $X$  is strongly connected. Further  $X$  is **forced-connected component (FCC)** if either  $|X| = 1$  or if  $|X| > 1$  then the sub-game  $\mathcal{G}(X)$  of the game  $\mathcal{G}$  played in  $X$  is forced-connected.

► **Lemma 5.** Let  $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ , where  $k > 1$ , be a collection of FTs that partition the game graph  $G$ . If  $\mathcal{G}$  is forced-connected then for every  $X \in \mathcal{C}$  there is a  $Y \in \mathcal{C}$  distinct from  $X$  such that either  $Y$  is a singleton consisting of a player 1's node and  $E(Y) \subseteq X$ , or  $Y$  has player 0's node  $y$  with  $E(y) \cap X \neq \emptyset$ .

We now define the sequence  $\{\mathcal{G}_k\}_{k \geq 0}$  of graphs. Initially, in  $G_0$ , the edges are only those that start in the nodes of player 0. Then the inductive construction increases the set of edges as follows. Intuitive explanation of the process is the following. Throughout the sequence, the invariant that FTs coincide with strongly connected components is maintained. By Lemma 7, these FTs are FCCs. This is ensured at each iteration by only adding the edges of those Player 1 vertices from where Player 1 is forced onto an existing SCC in the graph constructed so far. When this iteration terminates, the FCCs of the original graph coincide with the FCCs of the resulting graph, which in turn coincides with the SCCs by the invariant.

Here is now a formal process. We will call each  $\mathcal{G}_k$  the  $k^{th}$ -derivative of  $\mathcal{G}$ . We will also view each  $\mathcal{G}_k$  as a connectivity game. Our construction is the following.

- Initially, for  $k = 0$ , set  $F_0 = \emptyset$ ,  $U_0 = V_1$  and  $\mathcal{G}_0 = (V_0, V_1, E_0)$ , where  $E_0$  consists of all outgoing edges in  $E$  of player 0.
- For  $k > 0$ , consider the set  $F_k = \bigcup_{S \in SCC(\mathcal{G}_{k-1})} Force(U_{k-1}, S)$ , and define  $U_k = U_{k-1} \setminus F_k$ ,  $\mathcal{G}_k = (V_0, V_1, E_k)$ , where  $E_k = E_{k-1} \cup \{(v, u) \mid v \in F_k \text{ and } (v, u) \in E\}$ .

Note that the SCCs of  $\mathcal{G}_0$  are all singletons. For  $k = 1$  we have the following. The set  $F_1$  consists of all player 1 nodes of out-degree 1. The set  $E_1$  contains  $E_0$  and all outgoing edges from the set  $F_1$ . We note that each SCC of  $\mathcal{G}_1$  is also a FT in  $\mathcal{G}_1$ . Therefore each SCC  $X$  in  $\mathcal{G}_1$  is also a maximal FT. Observe that each  $F_k \subseteq U_{k-1}$  consists of player 1's nodes  $v$  such that all moves of player 1 from  $v$  go to the same SCC in  $\mathcal{G}_{k-1}$ . Moreover,  $U_k$  is the set of player 1's nodes whose outgoing edges aren't in  $E_k$ . Now we list some properties of the sequence  $\{\mathcal{G}_k\}_{k \geq 0}$ . A verification of these properties follows from the construction:

- For every player 1's node  $v$  and  $k > 0$ , the outgoing edges of  $v$  are in  $E_k \setminus E_{k-1}$  iff all the outgoing edges of  $v$  point to the same SCC in  $\mathcal{G}_{k-1}$  and in  $\mathcal{G}_{k-1}$  the out-degree of  $v$  is 0.
- For each  $k \geq 0$ , every SCC in  $\mathcal{G}_k$  is a FT in  $\mathcal{G}_k$ .
- For all  $k \geq 0$  we have  $F_{k+1} \subseteq U_k \subseteq U_{k-1} \subseteq \dots \subseteq U_0 = V_1$ .
- For all  $k_1 \neq k_2$  we have  $F_{k_1} \cap F_{k_2} = \emptyset$ .
- If  $F_k = \emptyset$  with  $k > 0$  then for all  $i \geq k$ ,  $\mathcal{G}_i = \mathcal{G}_{k-1}$ . We call the minimal such  $k$  the **stabilization point** and denote it by  $s$ . Note that  $s \leq |V_1|$ .
- If for all  $X \in SCC(\mathcal{G}_k)$ , either  $|X| > 1$  or  $X$  is a singleton consisting of player 0's node only then  $\mathcal{G}_k = \mathcal{G}$ .
- For each  $k \geq 0$  and player 1's node  $v$ , if  $v$  is in a nontrivial SCC in  $\mathcal{G}_k$  then all  $v$ 's outgoing edges from  $v$  are in  $E_k$ .

► **Lemma 6.** *If  $\mathcal{G}$  is forced-connected and  $|SCC(\mathcal{G}_k)| > 1$ , then  $\mathcal{G}_k \neq \mathcal{G}_{k+1}$ .*

Given a connectivity game  $\mathcal{G}$ , we now construct a sequence of forests  $\{\Gamma_k(\mathcal{G})\}_{k \geq 0}$  by induction. The idea is to represent the interactions of the SCCs of the graphs  $\mathcal{G}_k$  with SCCs of the  $\mathcal{G}_{k-1}$ , for  $k = 1, 2, \dots$ . The sequence of forests  $\Gamma_k(\mathcal{G}) = (N_k, Son_k)$ ,  $k = 0, 1, \dots$ , is defined as follows:

- For  $k = 0$ , set  $\Gamma_0(\mathcal{G}) = (N_0, Son_0)$ , where  $N_0 = \{\{v\} \mid v \in V\}$  and  $Son_0(\{v\}) = \emptyset$  for all  $v \in V$ .
- For  $k > 0$ , let  $C = SCC(\mathcal{G}_k) \setminus N_{k-1}$  be the set of new SCCs in  $\mathcal{G}_k$ . Define the forest  $\Gamma_k(\mathcal{G}) = (N_k, Son_k)$ , where
  1.  $N_k = N_{k-1} \cup C$ , and
  2.  $Son_k = Son_{k-1} \cup \{(X, Y) \mid X \in C, Y \in SCC(\mathcal{G}_{k-1}) \text{ and } Y \subset X\}$ .

Thus the new SCCs  $X$  that belong to  $C$  have become the roots of the trees in the forest  $\Gamma_k(\mathcal{G})$ . The children of  $X$  are now SCCs in  $\mathcal{G}_{k-1}$  that are contained in  $X$ .

Note that if  $s$  is the stabilization point of the sequence  $\{\mathcal{G}_k\}_{k \geq 0}$ , then for all  $k \geq s$  we have  $\Gamma_k(\mathcal{G}) = \Gamma_s(\mathcal{G})$ . Therefore, we set  $\Gamma(\mathcal{G}) = \Gamma_s(\mathcal{G})$ . Thus, for the forest  $\Gamma(\mathcal{G})$  we have  $N = N_s$  and  $Son = Son_s$ . The following properties of the forest  $\Gamma(\mathcal{G})$  can easily be verified:

- For all nodes  $X \in N$ ,  $X$ 's sons partition  $X = \bigcup_{Y \in Son(X)} Y$ .
- For all nodes  $X \in N$  with  $|X| > 1$ ,  $E(X \cap V_1) \subseteq X$ .
- The roots of  $\Gamma(\mathcal{G})$  are strongly connected components of  $\mathcal{G}_s$ .

► **Lemma 7.** *Consider  $\Gamma(\mathcal{G}) = (N, Son)$ . Let  $X \in N$  be such that  $|X| > 1$ . Then the sub-game  $\mathcal{G}(X)$  of the game  $\mathcal{G}$  played in  $X$  is forced-connected.*

Given the results above, we now relate solving forced connectivity problem to strong connectedness in directed graphs:

► **Theorem 8 (Characterization Theorem).** *The connectivity game  $\mathcal{G}$  is forced-connected if and only if the directed graph  $\mathcal{G}_s$  is strongly connected.*

### 3 Solving connectivity games efficiently

In a dynamic setting the incremental strongly connected maintenance (ISCCM) problem is stated as follows. Initially, we are given  $n$  vertices and the empty edge set. A sequence of edges  $e_1, \dots, e_m$  are added. No multiple edges and loops are allowed. The goal is to design a data structure that maintains the SCCs of the graphs after each addition of edges. By Theorem 8 the connectivity games problem is reduced to the incremental strongly connected component maintenance (ISCCM) problem. Note that Tarjan's algorithm solves the static version of the strongly connected component maintenance problem in time  $\mathbf{O}(m)$  [21].

We mention two algorithms that solve the ISCCM problem. The first is the *soft-threshold search algorithm* by Haeupler et al. [11] that handles sparse graphs. Their algorithm runs in time  $\mathbf{O}(\sqrt{mm})$ . The second is by Bender et al. [1, 2]. Their algorithm is best suited for the class of dense graphs and runs in time of  $\mathbf{O}(n^2 \log n)$ . We adapt these algorithms carefully in the proofs of our Theorems 9 and 10 below.

► **Theorem 9.** *The connectivity game  $\mathcal{G}$  can be solved in time  $\mathbf{O}((\sqrt{|V_1|} + 1)|E| + |V_1|^2)$ .*

We point out two features of this theorem. The first is that if the cardinality  $|V_1|$  is considered as a parameter, then we can solve the problem in linear time in  $|V_0|$ . The parameter constant in this case is  $|V_1|^{3/2}$ . The second feature is that the algorithm runs in linear time if the underlying game graph is sparse. Our second theorem is the following:

► **Theorem 10.** *The connectivity game  $\mathcal{G}$  can be solved in time  $\mathbf{O}((|V_1| + |V_0|) \cdot |V_0| \log |V_0|)$ .*

In comparison to the theorem above, this theorem implies that we can solve the problem in linear time in  $|V_1|$ . The parameterised constant is  $|V_0| \log |V_0|$ . Furthermore, the algorithm is more efficient than the first one on dense graphs.

Finally, both of the algorithms outperform the standard known bound  $\mathbf{O}(|V||E|)$  that solves the connectivity games.

#### 4 Solving explicitly given Müller games

We start with standard notions about games on graphs. Let  $\mathcal{G}$  be a Müller game. A set  $S \subseteq V$  determines a subgame in  $\mathcal{G}$  if for all  $v \in S$  we have  $E(v) \cap S \neq \emptyset$ . We call  $\mathcal{G}(S)$ , the subgame of  $\mathcal{G}$  determined by  $S$ . The set  $S \subseteq V$  is a  $\sigma$ -trap in  $G$  if  $S$  determines a subgame in  $\mathcal{G}$  and  $E(S \cap V_\sigma) \subseteq S$ .

Let  $Win_\sigma(\mathcal{G})$  be the set of all  $v$  in  $\mathcal{G}$  such that player  $\sigma$  wins  $\mathcal{G}$  starting from  $v$ . If  $Win_\sigma(\mathcal{G}) = V$ , we say that player  $\sigma$  wins  $\mathcal{G}$ . Otherwise, we say that player  $\sigma$  cannot win  $\mathcal{G}$ .

Let  $Attr_\sigma(X, G(Y))$  be the set of all  $v$  in  $Y$  such that player  $\sigma$  can force the token from  $v$  to  $X$  in game  $\mathcal{G}(Y)$ .

Let  $\Omega$  be the set of all winning sets of the Müller game  $\mathcal{G}$ . We *topologically order*  $<$  the set  $\Omega$ , that is, for all distinct  $X, Y \in \Omega$ , if  $X \subsetneq Y$  then  $X < Y$ . Thus, if  $W_1 < W_2 < \dots < W_s$  is a topological linear order on  $\Omega$  then we have this. If  $i < j$  then  $W_i \not\supseteq W_j$ .

Below we provide several results that are interesting on their own. We will also use them in our analysis of Müller games.

► **Lemma 11.** *Let  $\mathcal{G}$  be a game,  $F_0 = \Omega$  and  $F_1 = 2^V \setminus \Omega$ . If  $V \in F_\sigma$  and for all  $v \in V$ , either  $Attr_\sigma(\{v\}, G) = V$  or player  $\sigma$  wins  $\mathcal{G}(V \setminus Attr_\sigma(\{v\}, G))$  then player  $\sigma$  wins  $\mathcal{G}$ .*

The proof of the next lemma uses the lemma above:

► **Lemma 12.** *Let  $\mathcal{S} = \{S_1, S_2, \dots, S_k\} \subseteq 2^V \setminus \{V\}$  be the collection of all 0-traps in  $\mathcal{G}$  and assume that  $V \in \Omega$ . If for all  $S_i \in \mathcal{S}$ , player 1 can't win  $\mathcal{G}(S_i)$  then player 0 wins  $\mathcal{G}$ .*

The next lemma shows that we can reduce the size of the winning condition set  $\Omega'$  if one of the sets  $W \in \Omega'$  is minimal (with respect to  $\subseteq$ ) and not forced-connected. The proof uses Lemmas 11 and 12.

► **Lemma 13.** *Let  $W \subseteq V$  be a subgame. If  $\mathcal{G}(W)$  isn't forced-connected and no winning set in  $\Omega$  is contained in  $W$ , then  $Win_1(\mathcal{G}) = Win_1(\mathcal{G}')$ , where  $\mathcal{G}'$  is the same as  $\mathcal{G}$  but has the additional winning set:  $\Omega' = \Omega \cup \{W\}$ .*

Let  $\mathcal{G}$  be Müller game with  $\Omega = \{W_1, W_2, \dots, W_s\}$ . For the next two lemmas and the follow-up theorem we assume that there exists a  $W \in \Omega$  such that  $\mathcal{G}(W)$  is forced-connected and  $W$  isn't a 1-trap. The following is a construction that occurs naturally if one wants to analyse Müller games. We attribute this to Horn [12]:

► **Definition 14 (Horn's construction).** *Let  $W \in \Omega$  such that  $\mathcal{G}(W)$  is forced-connected and is not a 1-trap. The game  $\mathcal{G}_W = (G_W, \Omega_W)$  determined by  $W$  is defined as follows:*

1.  $V_W = V_0 \cup V_1 \cup \{\mathbf{W}\}$ , where  $\mathbf{W}$  is a player 1's new vertex.
2.  $E_W = E \cup (V_0 \cap W) \times \{\mathbf{W}\} \cup \{\mathbf{W}\} \times (E(V_1 \cap W) \setminus W)$ .
3.  $\Omega_W = (\Omega \cup \{W' \cup \{\mathbf{W}\} \mid W' \in R\}) \setminus (R \cup \{W\})$ , where the set  $R$  is the following  $R = \{W' \mid W' \in \Omega \text{ and } W \subset W'\}$ .

## 79:8 Connectivity in the Presence of an Opponent

Note that  $|\Omega_W| + 1 = |\Omega|$ , and  $\mathcal{G}_W(W)$  is forced-connected. Thus, similar to the lemma above, Horn's construction also reduces the size of  $\Omega$ . Now our goal is to show that Horn's construction preserves the winners of the original game. This is shown in the next two lemmas. Here we note that Horn's original proof of his correctness followed a different line of proof; this will be explained later.

► **Lemma 15.** *We have  $Win_0(\mathcal{G}_W) \setminus \{\mathbf{W}\} \subseteq Win_0(\mathcal{G})$ .*

**Proof.** Let  $\sigma_W$  be a winning strategy for player 0 in game  $\mathcal{G}_W$  starting at  $s \in V$ . We now describe a winning strategy for player 0 in  $\mathcal{G}$  starting from  $s$ . Player 0 plays the game  $\mathcal{G}$  by simulating plays  $\rho$  consistent with  $\sigma_W$  in  $\mathcal{G}_W$ . If a play  $\rho$  stays out of  $\mathbf{W}$ , then the player 0 copies  $\rho$  in  $\mathcal{G}$ . Once  $\rho$  moves to  $\mathbf{W}$ , then player 0 in  $\mathcal{G}$  moves to any node in  $W \cap V_1$ . Then player 0 stays in  $W$  and uses its strategy to visit every node in  $W$ . If player 1 moves out of  $W$  to a node  $u$  in  $\mathcal{G}$ , this will correspond to a move by player 1 from  $\mathbf{W}$  to  $u$  in  $\mathcal{G}_W$ . Player 0 continues on simulating  $\rho$ .

Let  $\rho'$  be the play in  $\mathcal{G}$  consistent with the strategy. If  $\rho$  meets  $\mathbf{W}$  finitely often then  $\text{Inf}(\rho) = \text{Inf}(\rho')$  and  $\text{Inf}(\rho') \in \Omega$ . If  $\rho$  never moves out of  $\mathbf{W}$  from some point on, then  $\text{Inf}(\rho') = W$ . In both cases player 0 wins. If the simulation leaves  $\mathbf{W}$  infinitely often, then  $\text{Inf}(\rho) \in \{W' \cup \{\mathbf{W}\} \mid W' \in R\}$  and  $W \subseteq \text{Inf}(\rho)$ . Therefore

$$\text{Inf}(\rho') \subseteq \text{Inf}(\rho) \setminus \{\mathbf{W}\} \cup W = \text{Inf}(\rho) \setminus \{\mathbf{W}\} \subseteq \text{Inf}(\rho'),$$

and hence  $\text{Inf}(\rho') = \text{Inf}(\rho) \setminus \{\mathbf{W}\} \in R$ , and player 0 wins. ◀

The next lemma is more involved. Assume that the set  $W' \subseteq V$  determines a subgame in  $\mathcal{G}$ . Then  $W'$  also determines a subgame of  $\mathcal{G}_W$ . We call the set  $W'$  *extendible* if  $W' \cup \{\mathbf{W}\}$  is a subgame of  $\mathcal{G}_W$ . Note that there could exist non-extendible  $W'$ . In particular, some winning sets in  $\Omega$  could become non-extendible in  $\mathcal{G}_W$ . In the analysis of  $Win_1(\mathcal{G}_W)$  extendible and non-extendible sets must be taken into account. The lemma below does exactly that.

► **Lemma 16.** *We have  $Win_1(\mathcal{G}_W) \setminus \{\mathbf{W}\} \subseteq Win_1(\mathcal{G})$ .*

**Proof.** We define the following two sets of subgames of the game  $\mathcal{G}$ . The first set  $\mathcal{A}$  is the following set of subgames of  $\mathcal{G}$ :

$$\{W' \mid W' \text{ is extendible \& player 1 wins } \mathcal{G}_W(W' \cup \{\mathbf{W}\})\}.$$

Note that if  $W' \in \mathcal{A}$  then player 1 wins the subgame  $\mathcal{G}_W(W')$ . The second set  $\mathcal{B}$  is the following set of subgames of  $\mathcal{G}$ :

$$\{W' \mid W \not\subseteq W' \text{ and player 1 wins } \mathcal{G}_W(W')\}.$$

Now we define the set  $\mathcal{S} = \mathcal{A} \cup \mathcal{B}$ . The sets  $\mathcal{A}$  and  $\mathcal{B}$  are disjoint. The set  $W$  does not belong to  $\mathcal{S}$  because  $W \cup \{\mathbf{W}\}$  is not a subgame in  $\mathcal{G}_W$  and  $W \notin \mathcal{B}$  by definition of  $\mathcal{B}$ . Note that the set  $\mathcal{B}$  can contain sets  $W'$  that are subsets of non-extendible (winning) sets.

Since player 1 wins  $\mathcal{G}_W(Win_1(\mathcal{G}_W))$ ,  $Win_1(\mathcal{G}_W)$  is a 0-trap in  $\mathcal{G}_W$  and it's easy to see that  $Win_1(\mathcal{G}_W) \setminus \{\mathbf{W}\}$  is also a 0-trap in  $\mathcal{G}$ . Then if  $\mathbf{W} \in Win_1(\mathcal{G}_W)$  then  $Win_1(\mathcal{G}_W) \setminus \{\mathbf{W}\} \in \mathcal{A}$ , otherwise  $Win_1(\mathcal{G}_W) \setminus \{\mathbf{W}\} \in \mathcal{B}$ . Since  $Win_1(\mathcal{G}_W) \setminus \{\mathbf{W}\} \in \mathcal{S}$ , to prove the lemma it suffices to show that player 1 wins  $\mathcal{G}(S)$  for all  $S \in \mathcal{S}$ .

Topologically order  $\mathcal{S}$ :  $S_1 < S_2 < \dots < S_s$ . For each  $\ell = 1, 2, \dots, s$ , we want to show that player 1 wins  $\mathcal{G}(S_\ell)$ . As player 1 wins  $\mathcal{G}_W(S_\ell)$ , for all 1-traps  $S' \subset S_\ell$  player 1 wins  $\mathcal{G}_W(S')$ . Let  $\mathcal{T} = \{T_1, T_2, \dots, T_t\} \subseteq 2^{S_\ell} \setminus \{S_\ell\}$  be all 1-traps in the game  $\mathcal{G}(S_\ell)$ . For each  $T_i \in \mathcal{T}$  we reason as follows.



*Case 1:*  $W \subseteq T_i$ . Then  $E_W(\mathbf{W}) \cap T_i = (E_W(V_1 \cap W) \setminus W) \cap T_i = (E_W(V_1 \cap W) \setminus W) \cap S_\ell = E_W(\mathbf{W}) \cap S_\ell$ . Since  $W \subseteq S_\ell$  implies player 1 wins  $\mathcal{G}_W(S_\ell \cup \{\mathbf{W}\})$  and  $E_W(\mathbf{W}) \cap S_\ell \neq \emptyset$ ,  $T_i \cup \{\mathbf{W}\}$  is also a 1-trap in the game  $\mathcal{G}_W(S_\ell \cup \{\mathbf{W}\})$  and player 1 wins  $\mathcal{G}_W(T_i \cup \{\mathbf{W}\})$ . Hence  $T_i$  belongs to  $\mathcal{A}$ .

*Case 2:*  $W \not\subseteq T_i$ . Note that  $T_i$  is also a 1-trap in the game  $\mathcal{G}_W(S_\ell)$  and player 1 wins  $\mathcal{G}_W(T_i)$ . Hence  $T_i$  belongs to  $\mathcal{B}$ .

Thus,  $\mathcal{T} \subset \mathcal{S}$  and by hypothesis, player 1 wins all  $\mathcal{G}(T_i)$ .

If  $S_\ell \in \mathcal{B}$  then player 1 wins  $\mathcal{G}(S_\ell) = \mathcal{G}_W(S_\ell)$ . Otherwise  $S_\ell \in \mathcal{A}$  and player 1 wins  $\mathcal{G}_W(S_\ell \cup \{\mathbf{W}\})$ .

- If  $S_\ell \notin \Omega$ , then for all  $v \in S_\ell$ ,  $\text{Attr}_1(\{v\}, G(S_\ell)) = S_\ell$  or player 1 wins  $\mathcal{G}(S_\ell \setminus \text{Attr}_1(\{v\}, G(S_\ell)))$  since  $S_\ell \setminus \text{Attr}_1(\{v\}, G(S_\ell))$  is a 1-trap in the game  $\mathcal{G}(S_\ell)$ . By Lemma 11, player 1 wins  $\mathcal{G}(S_\ell)$ .
- Otherwise by Lemma 12, there is a 0-trap  $Q \subset S_\ell \cup \{\mathbf{W}\}$  in  $\mathcal{G}_W(S_\ell \cup \{\mathbf{W}\})$  such that player 1 wins  $\mathcal{G}_W(Q)$ .
  - If  $\mathbf{W} \notin Q$  then  $W \cap V_0 \cap Q = \emptyset$  and  $Q$  also determines a 0-trap in the game  $\mathcal{G}(S_\ell)$ . Since  $W \not\subseteq Q$ , player 1 wins  $\mathcal{G}(Q) = \mathcal{G}_W(Q)$  and let  $Y = Q$ .
  - If  $\mathbf{W} \in Q$  then let  $Y = Q \setminus \{\mathbf{W}\}$ . Note that for all  $v \in V_0 \cap W \cap Q$ ,  $E_W(v) \cap Q = E_W(v) \cap (S_\ell \cup \{\mathbf{W}\})$  and  $|E_W(v) \cap (S_\ell \cup \{\mathbf{W}\})| > 1$ . Hence  $Y$  determines a 0-trap in the game  $\mathcal{G}(S_\ell)$ . Since player 1 wins  $\mathcal{G}_W(Y \cup \{\mathbf{W}\})$ ,  $Y$  belongs to  $\mathcal{A}$  and by hypothesis player 1 wins  $\mathcal{G}(Y)$ .

Therefore there exists a 0-trap  $Y$  in the game  $\mathcal{G}(S_\ell)$  such that player 1 wins  $\mathcal{G}(Y)$ . Also  $\text{Attr}_1(Y, G(S_\ell)) = S_\ell$  or player 1 wins  $\mathcal{G}(S_\ell \setminus \text{Attr}_1(Y, G(S_\ell)))$  since  $S_\ell \setminus \text{Attr}_1(Y, G(S_\ell))$  is a 1-trap in the game  $\mathcal{G}(S_\ell)$ . Then we construct a winning strategy for player 1 in the game  $\mathcal{G}(S_\ell)$  as follows.

- If the token is in  $Y$  then player 1 forces the token in  $Y$  forever and follows the winning strategy in  $\mathcal{G}(Y)$ .
- If the token is in  $\text{Attr}_1(Y, G(S_\ell))$  then player 1 forces the token to  $Y$ .
- Otherwise, player 1 follows the winning strategy in  $\mathcal{G}(S_\ell \setminus \text{Attr}_1(Y, G(S_\ell)))$ .

By hypothesis, for all  $S_i \in \mathcal{S}$ , player 1 wins  $\mathcal{G}(S_i)$ . ◀

By Lemmas 15 and 16, we have the following theorem.

► **Theorem 17.**  $\text{Win}_0(\mathcal{G}) = \text{Win}_0(\mathcal{G}_W) \setminus \{\mathbf{W}\}$  and  $\text{Win}_1(\mathcal{G}) = \text{Win}_1(\mathcal{G}_W) \setminus \{\mathbf{W}\}$ . ◻

```

Input: An explicit Müller game  $\mathcal{G} = (G, \Omega)$ 
Output: The winning regions of player 0 and player 1
topologically order  $\Omega$ ;
 $G' = (V_0', V_1', E') \leftarrow G = (V_0, V_1, E)$ ;
 $\Omega' \leftarrow \Omega$ ;
 $\text{Win}_0 \leftarrow \emptyset$ ;
while  $\Omega' \neq \emptyset$  do
   $W'_i \leftarrow \text{pop}(\Omega')$ 
  if  $\mathcal{G}'(W'_i)$  is forced-connected then
    if  $W'_i$  is a 1-trap in  $\mathcal{G}'$  then
      remove  $\text{Attr}_0(W'_i, G')$  from  $\mathcal{G}'$  and add it to  $\text{Win}_0$ ;
    else
       $\mathcal{G}' \leftarrow \mathcal{G}'_{W'_i}$ ;
    end
  end
end
return  $\text{Win}_0 \cap V$  and  $V \setminus \text{Win}_0$ 

```

■ **Figure 1** Algorithm for explicit Müller games.

## 79:10 Connectivity in the Presence of an Opponent

We briefly explain the algorithm, presented in Figure 1, that takes as input an explicit Müller game  $\mathcal{G}$  and returns the winning regions of the players. Initially, the algorithm orders  $\Omega$  topologically:  $W_1 < W_2 < \dots < W_s$ . At each iteration, the algorithm modifies the arena and the winning conditions:

- If  $W'_i$  doesn't determine a subgame in game  $\mathcal{G}'$  or  $\mathcal{G}'(W'_i)$  isn't forced-connected,  $W'_i$  is removed from  $\Omega'$ .
- Otherwise,  $\mathcal{G}'(W'_i)$  is forced-connected, then:
  - If  $W'_i$  is a 1-trap in  $\mathcal{G}'$  then  $Attr_0(W'_i, \mathcal{G}')$  is removed from  $\mathcal{G}'$  and added to the winning region of player 0. Note that all  $W' \in \Omega'$  with  $W' \cap Attr_0(W'_i, \mathcal{G}') \neq \emptyset$  are removed.
  - Otherwise, apply Horn's construction to  $\mathcal{G}'$  by setting  $\mathcal{G}' = \mathcal{G}'_{W'_i}$ . In this construction, a new player 1's node  $\mathbf{W}'_i$  is added to  $\mathcal{G}'$ ,  $\mathbf{W}'_i$  is added to all supersets of  $W'_i$  in  $\Omega'$  and  $W'_i$  itself is removed from  $\Omega'$ , which maintains the topological order of  $\Omega'$ .

Now our goal is to show that Horn's algorithm preserves the winner of the original games at each iteration so that the algorithm can compute the winning regions correctly. This is shown in the next lemma and theorem.

► **Lemma 18.** *At the end of each iteration, we have*

$$Win_0(\mathcal{G}) = (Win_0(\mathcal{G}') \cup Win_0) \cap V \text{ and } Win_1(\mathcal{G}) = Win_1(\mathcal{G}') \cap V.$$

**Proof.** Initially,  $\mathcal{G}' = \mathcal{G}$  and  $Win_0 = \emptyset$ , which holds the lemma. Then for  $i = 1, 2, \dots, s$ , we want to show that at the end of  $i$ th iteration,  $Win_0(\mathcal{G}) = (Win_0(\mathcal{G}') \cup Win_0) \cap V$  and  $Win_1(\mathcal{G}) = Win_1(\mathcal{G}') \cap V$ . Let  $\mathcal{G}''$  be  $\mathcal{G}'$  and  $Win'_0$  be  $Win_0$  at the beginning of  $i$ th iteration. Let  $\mathcal{G}'''$  be  $\mathcal{G}'$  and  $Win''_0$  be  $Win_0$  at the end of  $i$ th iteration. By hypothesis,  $Win_0(\mathcal{G}) = (Win_0(\mathcal{G}'') \cup Win'_0) \cap V$  and  $Win_1(\mathcal{G}) = Win_1(\mathcal{G}'') \cap V$ . If  $W'_i$  doesn't determine a subgame in game  $\mathcal{G}'$  or  $W'_i$  isn't forced-connected then by Lemma 13,  $W'_i$  can be removed without affecting the winning regions of the players of the game. Otherwise, if  $W'_i$  is a 1-trap in  $\mathcal{G}'$  then player 0 wins  $\mathcal{G}''(Attr_0(W'_i, \mathcal{G}''))$  by forcing the token to  $W'_i$  and then to go through  $W'_i$ . Since  $Attr_0(W'_i, \mathcal{G}'')$  is a 1-trap in  $\mathcal{G}''$ ,  $Attr_0(W'_i, \mathcal{G}'') \subseteq Win_0(\mathcal{G}'')$ . Since  $\mathcal{G}''' = \mathcal{G}''(V \setminus Attr_0(W'_i, \mathcal{G}''))$ ,  $Win_0(\mathcal{G}''') = Win_0(\mathcal{G}'') \cup Attr_0(W'_i, \mathcal{G}'')$  and  $Win_1(\mathcal{G}''') = Win_1(\mathcal{G}''')$ . Therefore,  $Win_0(\mathcal{G}) = (Win_0(\mathcal{G}''') \cup Attr_0(W'_i, \mathcal{G}'') \cup Win'_0) \cap V = (Win_0(\mathcal{G}''') \cup Win''_0) \cap V$  and  $Win_1(\mathcal{G}) = Win_1(\mathcal{G}''') \cap V$ . If  $W'_i$  isn't a 1-trap in  $\mathcal{G}'$  then by Theorem 17,  $Win_0(\mathcal{G}'') = Win_0(\mathcal{G}''') \setminus \{\mathbf{W}'_i\}$  and  $Win_1(\mathcal{G}'') = Win_1(\mathcal{G}''') \setminus \{\mathbf{W}'_i\}$ . Therefore,  $Win_0(\mathcal{G}) = (Win_0(\mathcal{G}''') \cup Win''_0) \cap V$  and  $Win_1(\mathcal{G}) = Win_1(\mathcal{G}''') \cap V$ . By hypothesis, at the end of each iteration,  $\mathcal{G}'$  and  $Win_0$  hold the lemma. ◀

By Lemma 18, we have the following theorem.

► **Theorem 19.** *At the end of the algorithm, we have*

$$Win_0(\mathcal{G}) = Win_0 \cap V \text{ and } Win_1(\mathcal{G}) = V \setminus Win_0. \quad \lrcorner$$

At each iteration, at most one player 1's vertex is added and at most  $|V'_0|$  edges are added. Therefore,  $|V'_0| = |V_0|$ ,  $|V'_1|$  is bounded by  $|V_1| + |\Omega|$  and  $|E'|$  is bounded by  $|E| + |V_0||\Omega|$ . For time complexity of the algorithm, there are at most  $|\Omega|$  iterations in a run and the most time-consuming operation is to determine if  $\mathcal{G}'(W'_i)$  is forced-connected. By Theorem 9 and Theorem 10, we have the following theorems.

► **Theorem 20.** *There exists an algorithm that solves the explicit Müller game  $\mathcal{G}$  in time  $\mathcal{O}(|\Omega| \cdot ((\sqrt{|V_1| + |\Omega|} + 1)(|E| + |V_0||\Omega|) + (|V_1| + |\Omega|)^2))$ .* ◻

► **Theorem 21.** *There exists an algorithm that solves the explicit Müller game  $\mathcal{G}$  in time  $\mathbf{O}(|\Omega| \cdot (|V_0| + |V_1| + |\Omega|) \cdot |V_0| \log |V_0|)$ .* ◻

Both of these algorithms beat the bound of Horn's algorithm. Importantly, Theorem 21 decreases the degree of  $|\Omega|$  from  $|\Omega|^3$  in Horn's algorithm to  $|\Omega|^2$ . Since  $|\Omega|$  is bounded by  $2^{|V|}$ , the improvement is significant.

## 5 Applications

We now apply the results above to specific classes of games. Here we give three examples of such classes. The first such class is the class of fully separated Müller games. A Müller game  $\mathcal{G}$  is *fully separated* if for each  $W \in \Omega$  there is a  $s_W$ , called separator, such that all  $s_W \in W$  but  $s_W \notin W'$  for all  $W' \in \Omega$  distinct from  $W$ . The second class of games is the class of linear games. A Müller game  $\mathcal{G}$  is a *linear game* if the set  $\Omega$  forms a linear order  $W_1 \subset W_2 \subset \dots \subset W_s$ . These classes of games were studied in [15]. As the games are fully separated, when one constructs  $\mathcal{G}'_{W'_i}$  there is no need to add a new vertex. Then applying Theorems 9 and 10 to Horn's algorithm, we get the following result:

► **Theorem 22.** *Each of the following is true:*

1. *Any fully separated Müller game  $\mathcal{G}$  can be solved in time  $\mathbf{O}(|V| \cdot ((\sqrt{|V_1|} + 1)|E| + |V_1|^2))$ .*
2. *Any fully separated Müller game  $\mathcal{G}$  can be solved in time  $\mathbf{O}(|V|^2 \cdot |V_0| \log |V_0|)$ .* ◻

Both of these algorithms beat the bound of [15]  $\mathbf{O}(|V|^2|E|)$  that solves fully separated Müller game. Applying Theorem 20 and Theorem 21, we have the following theorems.

► **Theorem 23.** *Each of the following is true:*

1. *Any linear Müller game  $\mathcal{G}$  can be solved in time  $\mathbf{O}(|V| \cdot ((\sqrt{|V|} + 1) \cdot |V_0||V| + |V|^2))$ .*
2. *Any linear Müller game  $\mathcal{G}$  can be solved in time  $\mathbf{O}(|V|^2 \cdot |V_0| \log |V_0|)$ .* ◻

Both of these algorithms beat the bound  $\mathbf{O}(|V|^{2 \cdot |V| - 1}|E|)$  from of [15] and the bound  $\mathbf{O}(|V|^3 \cdot |V_0|)$  implied from Horn's algorithm.

The third class of Müller games was introduced by A. Dawar and P. Hunter in [14]. They investigated games with anti-chain winning condition. A winning condition  $\Omega$  is an *anti-chain* if  $X \not\subseteq Y$  for all  $X, Y \in \Omega$ . Applying Theorem 20 and Theorem 21, we have the following theorems. Note that, since the winning condition is an anti-chain,  $|V'_1|$  is bounded by  $|V_1|$ ,  $|E'|$  is bounded by  $|E|$  and no new player 1's vertex is added to  $\Omega'$ .

► **Theorem 24.** *Each of the following is true:*

1. *Any Müller game  $\mathcal{G}$  with anti-chain winning condition can be solved in time  $\mathbf{O}(|\Omega| \cdot ((\sqrt{|V_1|} + 1)|E| + |V_1|^2))$ .*
2. *Any Müller game  $\mathcal{G}$  with anti-chain winning condition can be solved in time  $\mathbf{O}(|\Omega||V| \cdot |V_0| \log |V_0|)$ .* ◻

Just as above, both of the algorithms beat the bound  $\mathbf{O}(|\Omega||V|^2|E|)$  from [14] and the bound of Horn's algorithm  $\mathbf{O}(|\Omega||V||E|)$  that solves the explicit Müller games with anti-chain winning conditions.

## 6 A note on Horn's proof

In [12], Horn considers sensible sets. A winning set  $W \in \Omega$  is *sensible* if it determines a subgame. Initially, all non-sensible sets are removed (this is fine). Then Horn's assumption is that through the iteration process (in the algorithm in figure 1) sensibility is preserved.

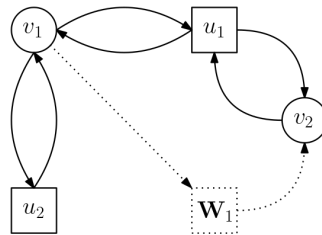
## 79:12 Connectivity in the Presence of an Opponent

When  $\mathcal{G}_W$  is built (during iterations), a winning condition  $W'$  that contains  $W$  might become non-sensible. Hence, sensibility is not preserved during iterations. Horn's analysis doesn't take non-sensible sets into account. This is important. In Horn's defence, assume we remove all non-sensible winning sets in the current game  $\mathcal{G}_W$  (Note that the algorithm removes non-sensible winning conditions). However, Horn does not prove that this is a correct action. Horn's proof does not analyse an intricate interplay between sensibility and maintenance of the winning sets at each iteration. Neglecting non-sensible sets makes the proofs of Lemmas 6 and 7 (in [12]) incorrect. Here is an example.

Let  $\mathcal{G} = (G, \Omega)$  be a game where  $G$  is shown in figure 2 and  $\Omega = \{W_1, W_2, W_3\}$ , where  $W_1 = \{v_1, u_1\}$ ,  $W_2 = \{v_1, u_1, u_2\}$ , and  $W_3 = \{v_1, v_2, u_1, u_2\}$ . During the algorithm, player 0 wins the subgame determined by  $W_1$ . The set  $W_1$  isn't a 1-trap. So the new player 1's node  $\mathbf{W}_1$  is added. The sensible set  $W_2$  now becomes non-sensible in  $\mathcal{G}_{W_1}$ . Let us assume that  $W_2 \cup \{\mathbf{W}_1\}$  is removed from the winning set in  $\mathcal{G}_{W_1}$ . Then player 0 wins the subgame determined by  $W_3 \cup \{\mathbf{W}_1\}$ . Horn's Lemma 7 applied to the game in  $\mathcal{G}_{W_1}$  that occurs on  $W_3 \cup \{\mathbf{W}_1\}$  states the following.

*Player 1 wins the original game played on  $W_3$ , where  $W_3$  is removed from  $\Omega$ .*

The proof uses induction that has the following important (in our view unrecoverable) flaw. The proof considers the maximal winning sets inside  $W_3$ . Clearly, the maximal winning set inside  $W_3$  is  $W_3$  itself. Horn refers to player 1 winning strategy on  $W_3$ . Such strategy does not exist as it needs to be built. This is a self-loop argument. Trying to save Horn's proof, let's assume that  $W_2$  was considered by Horn to be the maximal set. Since  $W_2 \cup \{\mathbf{W}_1\}$  is non-sensible in  $\mathcal{G}_{W_1}$ , it is removed during the algorithm. Horn claims that player 1 has a winning strategy by playing inside  $W_2$  and uses it to build a winning strategy in  $W_3$ . However, player 0 wins  $\mathcal{G}(W_1)$  and  $W_1$  is a 1-trap in  $\mathcal{G}(W_2)$ . As a result, player 1 has no winning strategy in  $\mathcal{G}(W_2)$  and Horn's proof fails. Since Horn reuses the proof of Lemma 7 in the proof of Lemma 6, Horn fails on the proofs of Lemmas 6 and 7. These show that Horn's inductive arguments fail.



■ **Figure 2** The counter case to Horn's Lemmas 6 and 7.

Our section 4 develops new ideas and methods that are not present in Horn's arguments. As an example, we consider extendible and non-extendible sets in the proof of Lemma 16. The lemma takes winning regions that are subsets of non-sensible sets into account. These are placed in set  $\mathcal{B}$  in the lemma. We also show that the players' winning nodes stay invariant with each iteration. This completely differs from Horn's arguments. Horn's Lemmas 5, 6, and 7 are aimed at proving that the original game restricted to  $W$ , given by the iteration, is won by one of the players. Our approach is obviously different.

---

**References**


---

- 1 Michael A Bender, Jeremy T Fineman, and Seth Gilbert. A new approach to incremental topological ordering. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 1108–1115. SIAM, 2009.
- 2 Michael A Bender, Jeremy T Fineman, Seth Gilbert, and Robert E Tarjan. A new approach to incremental cycle detection and related problems. *arXiv preprint*, 2011. [arXiv:1112.0784](https://arxiv.org/abs/1112.0784).
- 3 Hans L Bodlaender, Michael J Dinneen, and Bakhadyr Khoussainov. On game-theoretic models of networks. In *Algorithms and Computation: 12th International Symposium, ISAAC 2001 Christchurch, New Zealand, December 19–21, 2001 Proceedings 12*, pages 550–561. Springer, 2001.
- 4 Hans L Bodlaender, Michael J Dinneen, and Bakhadyr Khoussainov. Relaxed update and partition network games. *Fundamenta Informaticae*, 49(4):301–312, 2002.
- 5 Cristian S Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 252–263, 2017.
- 6 Krishnendu Chatterjee and Monika Henzinger. Efficient and dynamic algorithms for alternating büchi games and maximal end-component decomposition. *Journal of the ACM (JACM)*, 61(3):1–40, 2014.
- 7 Michael J Dinneen and Bakhadyr Khoussainov. Update networks and their routing strategies. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 127–136. Springer, 2000.
- 8 Stefan Dziembowski, Marcin Jurdzinski, and Igor Walukiewicz. How much memory is needed to win infinite games? In *Proceedings of Twelfth Annual IEEE Symposium on Logic in Computer Science*, pages 99–110. IEEE, 1997.
- 9 Aniruddh Gandhi, Bakhadyr Khoussainov, and Jiamou Liu. Efficient algorithms for games played on trees with back-edges. *Fundamenta Informaticae*, 111(4):391–412, 2011.
- 10 Erich Grädel, Wolfgang Thomas, and Thomas Wilke. Automata, logics, and infinite games. *lncs*, vol. 2500, 2002.
- 11 Bernhard Haeupler, Telikeyalli Kavitha, Rogers Mathew, Siddhartha Sen, and Robert E Tarjan. Incremental cycle detection, topological ordering, and strong component maintenance. *ACM Transactions on Algorithms (TALG)*, 8(1):1–33, 2012.
- 12 Florian Horn. Explicit muller games are ptime. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.
- 13 Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In *International Symposium on Mathematical Foundations of Computer Science*, pages 495–506. Springer, 2005.
- 14 Paul Hunter and Anuj Dawar. Complexity bounds for muller games. *Theoretical Computer Science (TCS)*, 2008.
- 15 Hajime Ishihara and Bakhadyr Khoussainov. Complexity of some infinite games played on finite graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 270–281. Springer, 2002.
- 16 Imran Khaliq, Bakhadyr Khoussainov, and Jiamou Liu. Extracting winning strategies in update games. In *Models of Computation in Context: 7th Conference on Computability in Europe, CiE 2011, Sofia, Bulgaria, June 27–July 2, 2011. Proceedings 7*, pages 142–151. Springer, 2011.
- 17 Bakhadyr Khoussainov, Jiamou Liu, and Imran Khaliq. A dynamic algorithm for reachability games played on trees. In *Mathematical Foundations of Computer Science 2009: 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24–28, 2009. Proceedings 34*, pages 477–488. Springer, 2009.
- 18 Donald A Martin. Borel determinacy. *Annals of Mathematics*, 102(2):363–371, 1975.
- 19 Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

## 79:14 Connectivity in the Presence of an Opponent

- 20 Anil Nerode, Jeffrey B Rummel, and Alexander Yakhnis. Mcaughton games and extracting strategies for concurrent programs. *Annals of Pure and Applied Logic*, 78(1-3):203–242, 1996.
- 21 Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.
- 22 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.