Optimal Strategies in Weighted Limit Games[*]

Aniello Murano

Università degli Studi di Napoli "Federico II", Naples, Italy

murano@na.infn.it

Sasha Rubin

The University of Sydney, Sydney, Australia

sasha.rubin@sydney.edu.au

Martin Zimmermann

University of Liverpool, Liverpool, UK

martin.zimmermann@liverpool.ac.uk

We prove the existence and computability of optimal strategies in weighted limit games, zero-sum infinite-duration games with a Büchi-style winning condition requiring to produce infinitely many play prefixes that satisfy a given regular specification. Quality of plays is measured in the maximal weight of infixes between successive play prefixes that satisfy the specification.

1 Introduction

Reactive synthesis is an ambitious approach to the problem of producing correct controllers for reactive systems, e.g., systems continuously interacting with their environment over an infinite time horizon. Instead of an engineer coding the controller and then checking it for correctness against a formal specification, one automatically computes a correct-by-construction controller from the specification.

The basic case of the problem, formalized as Church's problem [16], has been solved by the seminal Büchi-Landweber Theorem [8]. Here, the problem is recast as a game-theoretic one: Given a finite graph describing the interaction between the desired controller and its environment, and a winning condition representing the controller's specification, determine whether the "controller player" has a winning strategy for this game. If yes, Büchi and Landweber proved that she has a finite-state winning strategy, e.g., one that can be implemented by a finite automaton with output. Such a strategy can be seen as a controller that satisfies the specification. We refer to these lecture notes [21] and the references therein for a contemporary overview of reactive synthesis.

Ever since the seminal work of Büchi and Landweber, their result has been extended in various directions, e.g., more expressive winning conditions, infinite state spaces, stochastic settings, settings with imperfect information, etc. All these are motivated by the quest to model ever more aspects of relevant application domains.

Recently, another aspect has received considerable attention: Oftentimes, specifications are qualitative but some controllers are more desirable than others. Consider, for example, a controller that has to bring a system into a desirable state. Then, it is often desired, although not formally specified, that the state is reached as quickly as possible or with the minimal amount of resource consumption. Much effort has been put into computing controllers that satisfy such "nonfunctional" requirements.

But not every specification is a reachability property. As another example, assume we need to generate an arbiter that controls access to some shared resource. A typical specification here is to require that every request to the resource is eventually granted [34]. Again, we typically prefer controllers that grant

---

requests as quickly as possible. Note that this specification is not a simple reachability property that requires to reach a certain set of states, but a recurrence property that requires to infinitely often reach a state in which no request is pending. The optimization criterion then asks to minimize the maximal time between visits to such states.

Formally, recurrence properties are captured by Büchi games (see, e.g., [23]), i.e., games whose underlying graphs come with a set of desirable vertices that need to be visited infinitely often for the controller player to win. In this work, we consider a slightly different approach: We equip the graph describing the interaction with labels on vertices (think of the set of atomic propositions holding true in this state) and the edges with nonnegative weights (capturing the cost or time it takes to make this transition). The winning condition is induced by a deterministic finite automaton processing finite label sequences and is satisfied by an infinite play if it has infinitely many prefixes whose label sequence is accepted by the automaton.[1] Now, the quality of a play is measured as the maximal weight of an infix between two successive prefixes whose label sequences are accepted by the automaton. Finally, the quality of a strategy is obtained by maximizing over the values of the plays that are consistent with it.

By separating the graph modeling the interaction and the specification automaton, we obtain a fine-grained analysis of the complexity of computing controllers and the complexity of implementing controllers (measured in their number of states). In detail, our contributions are as follows:

1. We show that every such game has an optimal strategy for the controller player. To prove the strategy optimal, we also show that the player representing the environment always has an optimal strategy as well, i.e., a strategy that maximizes the weight between prefixes that have a label sequence that is accepted by the automaton. Both strategies are obtained by a nested fixed-point characterization that generalizes the classical algorithm for solving Büchi games (see, e.g., [14]). The inner fixed point is a characterization of optimal strategies in reachability games, which we use as blackbox in the outer fixed point characterization for recurrence conditions.

2. The fixed point (and the optimal strategies) can be computed in time $\mathcal{O}(|V|^3 \cdot |E| \cdot |Q|^2 \cdot |F|^2)$, where (V, E) is the underlying graph and Q and F are the sets of states and accepting states of the automaton. Here, we use the unit-cost model for arithmetic operations.

3. The size of optimal strategies is bounded by $|V| \cdot |Q| \cdot |F|$ which is tight up to a factor of $|F|$.

4. The value of an optimal strategy is bounded by $(|V| \cdot |Q| + 1) \cdot W$, if it is finite at all, where W is the largest weight appearing in the graph. This upper bound is shown to be tight.

5. Finally, we briefly consider the case of infinite state systems. In finite graphs, if there is any controller, then there is also one with finite value. We give a very simple infinite graph in which this is no longer the case: There is a controller, but none of finite value.

Let us stress that the results for reachability games mentioned in Item 1) are not novel and follow from stronger results (see, e.g., [6, 25]). However, we were unable to locate a reference for all the properties we require of our blackbox. Hence, for the sake of completeness, we present the construction for reachability as well, which also serves as a gentle introduction to the machinery necessary for recurrence.

Proofs omitted due to space restrictions can be found in the full version [28].

2 Definitions

Let $\mathbb{N}$ denote the nonnegative integers and define $\overline{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$ with $n < \infty$ and $n + \infty = \infty$ for every $n \in \mathbb{N}$. Given a finite directed graph (V, E) and $v \in V$, let $vE = \{v' \in V \mid (v, v') \in E\}$ denote the set of

---

[1] It is not hard to reduce this setting to the one of classical Büchi games by taking the product of the graph and the automaton.

successors of a vertex v.

Finite Automata   A deterministic finite automaton (DFA) $\mathfrak{A} = (Q, C, q_I, \delta, F)$ consists of a finite set Q of states containing the initial state $q_I \in Q$ and the accepting states $F \subseteq Q$, a finite set C of colors which we use as input letters, and a transition function $\delta \colon Q \times C \to Q$. Let $\delta^*(w)$ denote the unique state that is reached by processing $w \in C^*$, i.e., $\delta^*(\varepsilon) = q_I$ for the empty word ε and $\delta^*(w_0 \cdots w_j w_{j+1}) = \delta(\delta^*(w_0 \cdots w_j), w_{j+1})$ for a nonempty word $w_0 \cdots w_j w_{j+1} \in C^+$. The language of $\mathfrak{A}$ is $L(\mathfrak{A}) = \{w \in C^* \mid \delta^*(w) \in F\}$. The size of $\mathfrak{A}$ is defined as $|\mathfrak{A}| = |Q|$.

Infinite Games   Let us fix a finite nonempty set C of colors. A (weighted and colored) arena $\mathscr{A} = (V, V_0, V_1, E, w, c)$ consists of a finite directed graph (V, E) whose vertices are partitioned into the vertices V_0 of Player 0 (drawn as circles) and the vertices V_1 of Player 1 (drawn as rectangles), a weight function $w \colon E \to \mathbb{N}$ (drawn as edge labels), and a coloring $c \colon V \to C$ (drawn as vertex labels). We require every vertex to have an outgoing edge. A game $\mathscr{G} = (\mathscr{A}, \mathrm{Win})$ consists of an arena $\mathscr{A}$ and a (qualitative) winning condition $\mathrm{Win} \subseteq C^\omega$.

A play in $\mathscr{G}$ is an infinite path $\rho = v_0 v_1 v_2 \cdots \in V^\omega$ through (V, E). We lift the weight function to plays and play prefixes by adding up the weights of the edges of the play (prefix). Similarly, we lift the coloring to plays and play prefixes by applying it vertex-wise. A play ρ is winning for Player 0 in $\mathscr{G}$, if $c(\rho) \in \mathrm{Win}$; otherwise, it is winning for Player 1.

A strategy for Player $i \in \{0, 1\}$ is a map $\sigma \colon V^* V_i \to V$ satisfying $(v_j, \sigma(v_0 \cdots v_j)) \in E$ for every $v_0 \cdots v_j \in V^* V_i$. A strategy σ for Player i is positional, if we have $\sigma(wv) = \sigma(v)$ for every $w \in V^*$ and every $v \in V_i$. We denote such strategies w.l.o.g. as mappings from V_i to V.

A play $v_0 v_1 v_2 \cdots$ is consistent with a strategy σ for Player i, if $v_{j+1} = \sigma(v_0 \cdots v_j)$ for every j with $v_j \in V_i$. A strategy for Player i is winning from a vertex v if every play that starts in v and is consistent with the strategy is winning for Player i.

Memory Structures and Finite-state Strategies   A memory structure $\mathscr{M} = (M, \mathrm{init}, \mathrm{upd})$ for an arena (V, V_0, V_1, E, w, c) consists of a finite set M of memory states, an initialization function $\mathrm{init} \colon V \to M$, and an update function $\mathrm{upd} \colon M \times V \to M$. The update function can be extended to finite play prefixes in the usual way: $\mathrm{upd}^*(v) = \mathrm{init}(v)$ and $\mathrm{upd}^*(wv) = \mathrm{upd}(\mathrm{upd}^*(w), v)$ for $w \in V^*$ and $v \in V$. A next-move function $\mathrm{Nxt} \colon V_i \times M \to V$ for Player i has to satisfy $(v, \mathrm{Nxt}(v, m)) \in E$ for all $v \in V_i$ and $m \in M$. It induces a strategy σ for Player i with memory $\mathscr{M}$ via $\sigma(v_0 \cdots v_j) = \mathrm{Nxt}(v_j, \mathrm{upd}^*(v_0 \cdots v_j))$. A strategy is called finite-state if it can be implemented by a memory structure. We define $|\mathscr{M}| = |M|$. Slightly abusively, we say that the size of a finite-state strategy is the size of a memory structure implementing it.

An arena $\mathscr{A} = (V, V_0, V_1, E, w, c)$ and a memory structure $\mathscr{M} = (M, \mathrm{init}, \mathrm{upd})$ for $\mathscr{A}$ induce the expanded arena $\mathscr{A} \times \mathscr{M} = (V \times M, V_0 \times M, V_1 \times M, E', w', c')$ where E' is defined via $((v, m), (v', m')) \in E'$ if and only if $(v, v') \in E$ and $\mathrm{upd}(m, v') = m'$. Furthermore, $w'((v, m), (v', m')) = w(v, v')$ and $c'(v, m) = c(v)$. Every play $\rho = v_0 v_1 v_2 \cdots$ in $\mathscr{A}$ has a unique extended play $\mathrm{ext}(\rho) = (v_0, m_0)(v_1, m_1)(v_2, m_2) \cdots$ in $\mathscr{A} \times \mathscr{M}$ defined by $m_0 = \mathrm{init}(v_0)$ and $m_{j+1} = \mathrm{upd}(m_j, v_{j+1})$, i.e., $m_j = \mathrm{upd}^*(v_0 \cdots v_j)$. The extended play of a finite play prefix in $\mathscr{A}$ is defined analogously. Note that a play (prefix) and its extension have the same weight and the same color sequence.

Given a positional strategy σ' for Player i in $\mathscr{A} \times \mathscr{M}$, define the finite-state strategy σ for Player i in $\mathscr{A}$ by specifying the next-move function $\mathrm{Nxt}_{\sigma'}$ with $\mathrm{Nxt}(v, m) = v'$, where $v' \in V$ is the unique vertex with $\sigma'(v, m) = (v', m')$ for some $m' \in M$.

Remark 1. *Let σ and σ' be as above and let ρ a play in $\mathscr{A}$. Then, ρ is consistent with σ if and only if* $\text{ext}(\rho)$ *is consistent with σ'.*

Now let $\mathscr{M} = (M, \text{init}, \text{upd})$ be a memory structure for the arena $\mathscr{A} = (V, V_0, V_1, E, w, c)$ and let σ' be a finite-state strategy for Player i in $\mathscr{A} \times \mathscr{M} = (V', V_0, V_1', E', w', c')$ implemented by $\mathscr{M}' = (M', \text{init}', \text{upd}')$ and Nxt'. We define the product of $\mathscr{M}$ and $\mathscr{M}'$ as $\mathscr{M} \times \mathscr{M}' = (M \times M', \text{init}'', \text{upd}'')$ where $\text{init}''(v) = (\text{init}(v), \text{init}'(v, \text{init}(v)))$ and

$$\text{upd}''((m, m'), v) = (\text{upd}(m, v), \text{upd}'(m', (v, \text{upd}(m, v)))),$$

which is a memory structure for $\mathscr{A}$. Further, we obtain a finite-state strategy σ for Player i in $\mathscr{A}$ implemented by $\mathscr{M} \times \mathscr{M}'$ and Nxt, which is defined as $\text{Nxt}(v, (m, m')) = \text{Nxt}'((v, m), m')$.

Remark 2. *Let σ and σ' be as above and let ρ a play in $\mathscr{A}$. Then, ρ is consistent with σ if and only if* $\text{ext}(\rho)$ *is consistent with σ', where $\text{ext}(\rho)$ is defined with respect to $\mathscr{M}$.*

Let $\mathscr{A}$ be an arena with vertex set V and coloring $c \colon V \to C$, and let $\mathfrak{A} = (Q, C, q_I, \delta, F)$ be a DFA over C. Then, we define $\mathscr{M}_{\mathfrak{A}} = (Q, \text{init}_{\mathfrak{A}}, \text{upd}_{\mathfrak{A}})$ with $\text{init}_{\mathfrak{A}}(v) = \delta(q_I, c(v))$ and $\text{upd}_{\mathfrak{A}}(q, v) = \delta(q, c(v))$, which is a memory structure for $\mathscr{A}$. By construction, we have $\text{upd}^*(v_0 \cdots v_j) = \delta^*(c(v_0 \cdots v_j))$. In particular, $c(v_0 \cdots v_j) \in L(\mathfrak{A})$ if and only if $\text{upd}^*(v_0 \cdots v_j) \in F$.

3 Weighted Limit Games

Recall that C is the finite set of colors used to define winning conditions. The limit of a language $K \subseteq C^*$ of finite words is

$$\lim(K) = \{\alpha_0 \alpha_1 \alpha_2 \cdots \in C^\omega \mid \alpha_0 \cdots \alpha_j \in K \text{ for infinitely many } j\}$$

containing all infinite words that have infinitely many prefixes in K. For technical reasons, we require in the following $\varepsilon \notin K$.

We call a game of the form $\mathscr{G} = (\mathscr{A}, \lim(K))$ a weighted limit game and define the value of a play $\rho = v_0 v_1 v_2 \cdots$ as

$$\text{val}_{\mathscr{G}}(\rho) = \sup_{j \in \mathbb{N}} \min_{\substack{j' > j \\ c(v_0 \cdots v_{j'}) \in K}} w(v_j \cdots v_{j'}),$$

where $\min \emptyset = \infty$. Intuitively, we measure the quality of a winning play by the maximal weight of an infix between two consecutive prefixes whose color sequences are in K. Note that this value might be ∞, even for plays in $\lim(K)$ (see Example 1) and that it is necessarily ∞ if the play is not in $\lim(K)$. Also, let us remark that this definition depends on K, not only on $\lim(K)$: It is straightforward to construct languages K and K' with $\lim(K) = \lim(K')$, but the value functions induced by K and K' differ. Hence, we always make sure that the language K inducing the value function is clear from context.

Remark 3. *Let $\mathscr{G} = (\mathscr{A}, \text{Win})$ be a weighted limit game. Then, $\text{val}_{\mathscr{G}}(\rho) < \infty$ implies $c(\rho) \in \text{Win}$.*

Note that the other direction does not hold, as shown in the next example.

Example 1. *For the sake of simplicity, we identify vertices and their color in this example. Hence, let $K = \{v_0, v_1\}^* v_1$. Then, $\lim(K)$ is the set of words having infinitely many occurrences of v_1. Now, in a game $\mathscr{G}$ with winning condition $\lim(K)$ and a weight function mapping every edge to 1, $\text{val}_{\mathscr{G}}(\rho)$ is equal to the supremum over the length of infixes of the form $v_1 v_0^*$ in ρ. This may be ∞, even if the play ρ is in $\lim(K)$, e.g., in the play*

$$\rho = v_0 v_1 \, v_0 v_0 \, v_1 \, v_0 v_0 v_0 \, v_1 \, v_0 v_0 v_0 v_0 \, v_1 \, v_0 v_0 v_0 v_0 v_0 \, v_1 \cdots.$$

Given a strategy σ for Player 0 and a vertex v, define $\mathrm{val}_{\mathscr{G}}(\sigma, v) = \sup_{\rho} \mathrm{val}_{\mathscr{G}}(\rho)$ with the supremum ranging over all plays ρ that start in v and are consistent with σ. Remark 3 can be lifted from plays to strategies.

Remark 4. *Let $\mathscr{G} = (\mathscr{A}, \mathrm{Win})$ be a weighted limit game and let σ be a strategy for Player 0. Then, $\mathrm{val}_{\mathscr{G}}(\sigma, v) < \infty$ implies that σ is a winning strategy for Player 0 from v in $\mathscr{G}$.*

Again, the other direction of the implication does not hold, which can be seen by constructing a one-player game where Player 0 produces the play from Example 1.

We say that a strategy σ for Player 0 in a weighted limit game $\mathscr{G}$ is optimal, if it satisfies $\mathrm{val}_{\mathscr{G}}(\sigma, v) \leq \mathrm{val}_{\mathscr{G}}(\sigma', v)$ for every strategy σ' for Player 0 and every vertex v. Note that this definition is a global one, i.e., the strategy has to be better than any other strategy from *every* vertex.

Further, a weighted limit game with winning condition $\mathrm{Win} \subseteq C^{\omega}$ is regular, if $\mathrm{Win} = \lim(L(\mathfrak{A}))$ for some DFA $\mathfrak{A}$ over C. Note that every such language is ω-regular, (in fact it is recognized by $\mathfrak{A}$ when seen as Büchi automaton). In contrast, not every ω-regular language is a regular limit language, e.g., the ω-regular language $(a+b)^*b^{\omega}$ of words with finitely many a is not a regular limit language. In fact, Landweber showed that the regular limit languages are exactly the languages recognized by deterministic Büchi automata [27].

Our main results on regular weighted limit games show that Player 0 has an optimal strategy in every such game and how to compute an optimal strategy.

Theorem 1.

1. *Player 0 has an optimal finite-state strategy in every regular weighted limit game.*

2. *The problem "Given an arena $\mathscr{A}$ and a DFA $\mathfrak{A}$, compute an optimal strategy for Player 0 in $(\mathscr{A}, \lim(L(\mathfrak{A})))$" is solvable in time $\mathscr{O}(|V|^3 \cdot |E| \cdot |Q|^2 \cdot |F|^2)$, where (V, E) is the graph underlying $\mathscr{A}$ and Q and F are the sets of states and accepting states of $\mathfrak{A}$ (using the unit-cost model).*

Before we prove this result, let us comment on one restriction of our model: We only allow nonnegative edge weights. The reason is that it is straightforward to construct a game witnessing that optimal finite-state strategies do not necessarily exist in arenas with negative weights.

Example 2. *Consider the game depicted in Figure 1. As Player 0 moves at every vertex, we can identify plays and strategies. Also, for the sake of simplicity, we identify vertex names and colors and consider $K = (v_0 v_1^* v_2)^*$, i.e., the winning plays are of the form $(v_0 v_1^+ v_2)^{\omega}$. For every $j > 0$, Player 0 has a finite-state strategy to produce the play $\rho_j = (v_0 v_1^j v_2)^{\omega}$ with $\mathrm{val}_{\mathscr{G}}(\rho_j) = -j$, which is also the value of the strategy from v_0. Hence, she can enforce arbitrarily small values. Furthermore, straightforward pumping arguments show that every finite-state strategy has a bounded value, as it has to leave v_1 after a bounded number of steps.*

Altogether, there is no optimal finite-state strategy.

To prove Theorem 1, we first consider the simpler setting of weighted reachability games, i.e., games where a prefix in K has to be reached at least once. This problem is a special case of more general problems that have been considered before (see, e.g., [6, 25]). However, these works do not prove all the results we require here. Hence, we discuss in Subsection 3.1 a fixed point algorithm computing optimal strategies in reachability games. Then, we use this algorithm as a black box to build another fixed point algorithm computing optimal strategies in weighted limit games (Subsection 3.2).
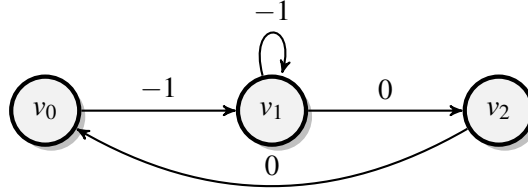
Figure 1: The arena for Example 2.

3.1 Computing Optimal Strategies in Weighted Reachability Games

Given a DFA $\mathfrak{A}$ over C with $\varepsilon \notin L(\mathfrak{A})$, define for a play $\rho = v_0 v_1 v_2 \cdots$

$$\text{val}_{\mathscr{G}}^R(\rho) = \min_{j \in \mathbb{N}} \{ w(v_0 \cdots v_j) \mid c(v_0 \cdots v_j) \in L(\mathfrak{A}) \},$$

where $\min \emptyset = \infty$. So, $\text{val}_{\mathscr{G}}^R(\rho)$ is the weight of the shortest nonempty prefix of ρ whose label sequence is accepted by $\mathfrak{A}$. This also minimizes the accumulated weight, as we only consider nonnegative weights on edges. This definition for plays is lifted to strategies σ for Player 0 as for limit games: $\text{val}_{\mathscr{G}}^R(\sigma, v) = \sup_\rho \text{val}_{\mathscr{G}}^R(\rho)$ where ρ ranges over all plays starting in the vertex v that are consistent with σ. Similarly, optimality of strategies is defined as for limit games.

In the remainder of this section, we show how to compute optimal strategies with respect to $\text{val}_{\mathscr{G}}^R$, given an arena $\mathscr{A}$ and a DFA $\mathfrak{A}$. First, let $\mathscr{A} \times \mathscr{M}_{\mathfrak{A}} = (V, V_0, V_1, E, w, c)$ be the product of $\mathscr{A}$ and the memory structure induced by $\mathfrak{A}$ (see Page 116). Furthermore, let F be the set of vertices of the form (v, q) where q is an accepting state of $\mathfrak{A}$, i.e., F is a set of vertices of the product arena, *not* the set of accepting states of $\mathfrak{A}$. However, reaching a state in F from a vertex of the form $(v, \text{init}(v))$ signifies that the label sequence induced by the play is accepted by $\mathfrak{A}$ (see Page 117).

A ranking for $\mathscr{A} \times \mathscr{M}_{\mathfrak{A}}$ is a mapping $r \colon V \to \overline{\mathbb{N}}$. Let $\mathscr{R}$ denote the set of all rankings. We order rankings by defining $r \sqsubseteq r'$ if $r(v) \geq r'(v)$ for all $v \in V$, i.e., r' is "better" than r if r' assigns ranks that are pointwise no larger than those of r. Hence, the least (and thus the worst) ranking is the one mapping every vertex to ∞. Furthermore, there are no infinite strictly ascending chains of rankings, as the ranks only decrease in such a chain, but are always nonnegative.

Next, we define the map $\ell \colon \mathscr{R} \to \mathscr{R}$ via

$$\ell(r)(v) = \begin{cases} 0 & \text{if } v \in F, \\ \min\{r(v), \min_{v' \in vE} w(v, v') + r(v')\} & \text{if } v \in V_0 \setminus F, \\ \min\{r(v), \max_{v' \in vE} w(v, v') + r(v')\} & \text{if } v \in V_1 \setminus F. \end{cases}$$

We will use ℓ to compute the value of an optimal strategy: At vertices in F, Player 0 has already achieved her goal, i.e., they are assigned a rank of 0. Now, if it is Player 0's turn at a vertex $v \notin F$, then she has to move to a successor. As she aims to minimize the accumulated weight, she prefers a successor v' that minimizes the sum of the weight $w(v, v')$ of the edge leading to v' and the rank of v'. The reasoning for Player 1 is dual: he tries to maximize the accumulated weight. Finally, for technical reasons, we ensure that ℓ does never increase a rank via taking the minimum with the old rank of v (which ensures that ℓ is monotone).

Remark 5. *We have $r \sqsubseteq \ell(r)$ for every ranking r.*

Let r_0 be the least element of $\mathscr{R}$, i.e., the ranking mapping every vertex to ∞, and let $r_{j+1} = \ell(r_j)$ for every j. Then, we define $r^* = r_n$ for the minimal n with $r_n = r_{n+1}$. Note that such a (least) fixed point r^n exists due to Remark 5 and as $\sqsubseteq$ has no infinite strictly ascending chain. From r^* one can derive an optimal strategy for Player 0 and the values of such a strategy.

Example 3. *Consider the arena depicted in Figure 2, where we mark vertices in F by doubly-lined vertices. We illustrate the computation of the rankings r_j below the arena, which reaches a fixed point after four applications of ℓ, i.e., $r_4 = r_5$. Note that the rank of vertex v_4 is updated twice.*

Let us sketch how to extract a strategy for Player 0 from the fixed point r_4. Consider, e.g., the vertex $v_2 \in V_0$. It has rank 4 and an edge of weight 4 leading to a vertex of rank $4 - 4 = 0$, which is the optimal move. In general, every vertex v of Player 0 with finite rank $r(v)$ has an edge to a successor v' such that $r(v') = r(v) - w(v, v')$. Dually, consider the vertex $v_1 \in V_1$: It has rank 5 and every edge leaving v_1 goes to a vertex v' of rank at most $5 - w(v, v')$. Again, this property is satisfied for every vertex with finite rank.

Hence, using these two properties inductively shows that Player 0 has a strategy so that every move from a vertex that is not in F decreases the rank by the weight of the edge taken. Thus, as ranks are nonnegative, a visit to F is guaranteed unless from some point onwards only edges of weight 0 are used. However, we will rule this out by ensuring that the target of the edge of weight 0 has reached its final rank before the source of the edge, e.g., the successors v_2 and v_3 of vertex v_3 have rank 4 and the corresponding edge has weight 0. However, v_2 has reached its final rank one step before v_3 has. Ultimately, we show that either the rank or this so-called settling time strictly decreases along every edge taken from a vertex that is not in F. As there is no infinite descending chain in this product order, F has to be reached eventually. Using dual arguments, one can define a strategy for Player 1 and then show these strategies to be optimal.

In the example, Player 0 moves from v_4 to v_3, from where she moves to v_2 and then to v_0. This strategy is optimal from every vertex and realizes the value $r_4(v)$ from every vertex v. For example, the unique play consistent with this strategy starting in v_4 has value 11.

It is instructive to compare the computation of the rankings to the attractor computation for reachability games (see, e.g., [23]): a straightforward induction shows that the j-th level of the attractor computation is equal to $\{v \mid r_{j+1}(v) \neq \infty\}$. However, the attractor yields a strategy that minimizes the number of moves necessary to reach F while the rankings minimize the accumulated weight. This difference is witnessed by vertex v_4: the attractor strategy takes the direct edge to v_0 of weight 99 while the rankings induce the strategy described above, which realizes a smaller value by taking a longer path through the arena.

We sketch how to obtain an optimal strategy σ for Player 0 from the fixed point r^*, and how r^* and σ can be computed in polynomial time. To this end, we need to introduce some additional notation. Consider the sequence $r_0, r_1, \ldots, r_n = r^*$ as above. Due to Remark 5, we have $r_j(v) \geq r_{j+1}(v)$ for every j and every v. The settling time of a vertex v is defined as $t_s(v) = \min\{j \mid r_j(v) = r^*(v)\}$, i.e., as the first time v is assigned its final rank $r^*(v)$. The construction of an optimal strategy is based on the following results about ranks and settling times, which formalize the intuition given in Example 3.

Lemma 1. *Let $v \in V$.*

1. $r^*(v) = \infty$ *if and only if* $t_s(v) = 0$.

2. $v \in F$ *implies* $r^*(v) = 0$ *and* $t_s(v) = 1$.

3. *If* $v \in V_0 \setminus F$ *then* $r^*(v) \leq w(v, v') + r^*(v')$ *for all successors* $v' \in vE$. *Furthermore, there is some successor* $\bar{v} \in vE$ *with* $r^*(v) = w(v, \bar{v}) + r^*(\bar{v})$. *Finally, if* $r^*(v) < \infty$, *then* $\bar{v}$ *can be chosen such that it additionally satisfies* $t_s(v) = t_s(\bar{v}) + 1$.
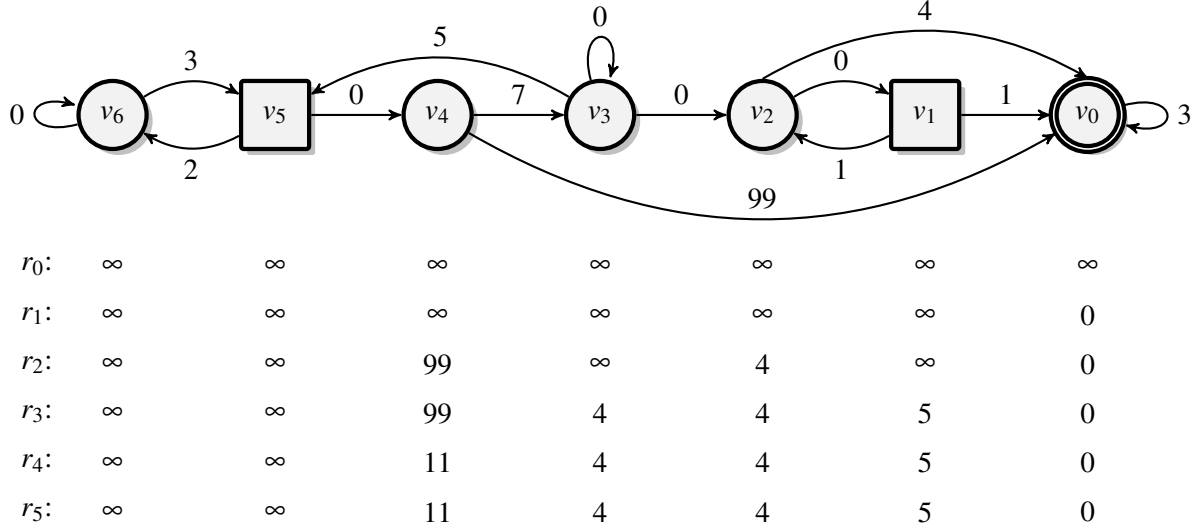
r_0:	∞	∞	∞	∞	∞	∞	∞
r_1:	∞	∞	∞	∞	∞	∞	0
r_2:	∞	∞	99	∞	4	∞	0
r_3:	∞	∞	99	4	4	5	0
r_4:	∞	∞	11	4	4	5	0
r_5:	∞	∞	11	4	4	5	0

Figure 2: The arena for Example 3 and the evolution of the corresponding rankings.

4. If $v \in V_1 \setminus F$ then $r^*(v) \geq w(v,v') + r^*(v')$ for all successors $v' \in vE$. Furthermore, there is some successor $\bar{v} \in vE$ with $r^*(v) = w(v,\bar{v}) + r^*(\bar{v})$.

5. If $v \in V_1 \setminus F$ and $\bar{v} \in vE$ with $r^*(v) = r^*(\bar{v}) < \infty$, then $t_s(v) > t_s(\bar{v})$.

We call successors $\bar{v}$ as in Items 3 and 4 optimal. If Player 0 uses an optimal successor, then the rank decreases by the weight of the edge. If this weight is 0, i.e., the rank stays constant, then the settling time decreases. Similarly, along all edges available to Player 1, the rank decreases at least by the weight of the edge. Again, if that value is 0, i.e., the rank stays constant, then the settling time decreases.

Using these properties, we define a strategy for Player 0 in $\mathscr{A}$. To this end, we first define a positional strategy σ' for her on $\mathscr{A} \times \mathscr{M}_{\mathfrak{A}}$ as follows: at a vertex $v \in V_0 \setminus F$ move to some optimal successor of v. From every vertex $v \in F \cap V_0$ move to an arbitrary successor. Now, let σ be the unique finite-state strategy in $\mathscr{A}$ implemented by $\mathscr{M}_{\mathfrak{A}}$ and $\mathrm{Nxt}_{\sigma'}$, the next-move function induced by σ'.

Lemma 2. σ as defined above is an optimal strategy for Player 0 in $\mathscr{G}$.

This result is proven in two steps. First, one shows $\mathrm{val}_{\mathscr{G}}^R(\sigma,v) \leq r^*(v,\mathrm{init}(v))$ for every vertex v of $\mathscr{A}$, applying the properties posited in Lemma 1 inductively. Secondly, analogously to the construction of σ, one constructs a strategy τ for Player 1 satisfying $\mathrm{val}_{\mathscr{G}}^R(\rho) \geq r^*(v,\mathrm{init}(v))$ for every vertex v of $\mathscr{A}$ and every play ρ starting in v and consistent with τ, which is again proven by applying Lemma 1 inductively.

Furthermore, by bounding the settling times of vertices one can show that the fixed point r^* is reached after a linear number of applications of ℓ.

Lemma 3. We have $r^* = r_{|\mathscr{A}| \cdot |\mathfrak{A}| + 1}$.

A simple corollary of the previous lemma yields an upper bound on $\mathrm{val}_{\mathscr{G}}^R$, which follows from the fact that each application of ℓ increases the ranks by no more than the maximal weight of an edge.

Corollary 1. If $\mathrm{val}_{\mathscr{G}}^R(v) < \infty$ then $\mathrm{val}_{\mathscr{G}}^R(v) \leq |\mathscr{A}| \cdot |\mathfrak{A}| \cdot W$, where W is the largest weight in $\mathscr{A}$.

One can show that the upper bound on the value is tight, e.g., using a game similar to the one presented in .

3.2   Computing Optimal Strategies in Weighted Limit Games

Now, we use the fixed point algorithm of the previous subsection to achieve the main goal of this work: solving regular weighted limit games optimally. Thus, fix a weighted arena $\mathscr{A}$ and a DFA $\mathfrak{A}$ over C inducing the winning condition $\lim(K)$ and let $\mathscr{A} \times \mathscr{M}_{\mathfrak{A}} = (V, V_0, V_1, E, w, c)$ be the product of $\mathscr{A}$ and the memory structure induced by $\mathfrak{A}$. Furthermore, let F be the set of vertices of the form (v, q) where q is an accepting state of $\mathfrak{A}$, i.e., F is again a set of vertices of the product arena, *not* the set of accepting states of $\mathfrak{A}$.

Recall that $\mathscr{R}$ is the set of rankings $r : V \to \overline{\mathbb{N}}$, which is ordered by $\sqsubseteq$ with $r \sqsubseteq r'$ if and only if $r(v) \geq r'(v)$ for all $v \in V$. Hence, the largest (i.e., best) element of $\mathscr{R}$ is the ranking mapping every vertex to 0. We use the operator ℓ defined in Subsection 3.1 to solve limit games. Recall that ℓ allows to compute, for a given set of goal vertices, an optimal strategy that ensures a visit to a goal vertex. However, here we have to treat the set of goal vertices as a parameter because we need to compute optimal strategies for subsets of F. Hence, we write $\ell_{F'}$ for $F' \subseteq V$ for the operator

$$
\ell_{F'}(r)(v) = \begin{cases} 0 & \text{if } v \in F', \\ \min\{r(v), \min_{v' \in vE} w(v, v') + r(v')\} & \text{if } v \in V_0 \setminus F', \\ \min\{r(v), \max_{v' \in vE} w(v, v') + r(v')\} & \text{if } v \in V_1 \setminus F'. \end{cases}
$$

All results proven about ℓ in Subsection 3.1 also hold true for $\ell_{F'}$. In particular, we can compute an optimal strategy for Player 0 to reach F' and for Player 1 to avoid F' whenever possible, and to maximize the weight, if it is not possible.

The fixed point of $\ell_{F'}$ induces an optimal strategy for Player 0 to reach F'. However, on vertices in F', from which she reaches F' trivially (i.e., in zero steps), the fixed point does not yield any information on how to reach F' *again*. However, this information can easily be generated from the fixed point. Given an arbitrary ranking r and a set $F' \subseteq V$ of vertices, define the completion $\mathrm{cmplt}_{F'}(r)$ of r (with respect to F') via

$$
\mathrm{cmplt}_{F'}(r)(v) = \begin{cases} r(v) & \text{if } v \notin F', \\ \min_{v' \in vE} w(v, v') + r(v') & \text{if } v \in F' \cap V_0, \\ \max_{v' \in vE} w(v, v') + r(v') & \text{if } v \in F' \cap V_1. \end{cases}
$$

If r is the least fixed point of $\ell_{F'}$, then $\mathrm{cmplt}_{F'}(r)$ is obtained from r by assigning to each vertex in F' the minimal weight it takes Player 0 to reach F' once more. This is necessary, as we need to reach F infinitely often to win a limit game. The values for all $v \notin F'$ coincide in r and $\mathrm{cmplt}_{F'}(r)$.

Recall that the definition of optimal successors in Subsection 3.1 with respect to the least fixed point r of $\ell_{F'}$ is only defined for vertices in $V \setminus F'$. For $r' = \mathrm{cmplt}_{F'}(r)$, we can extend this notion to F' as well as follows: a successor $\bar{v}$ of v in F' is optimal, if $r'(v) = w(v, \bar{v}) + r(\bar{v})$.

Now, we again define an operator ℓ_L updating rankings and show that determining a fixed point of the operator induces optimal strategies for both players. Intuitively, the operator tries to reach F with minimal weight, but also has to account for the fact that F has to be reached repeatedly, i.e., the ranks of the vertices reached in F should be as small as possible.

Formally, given a ranking r, let $r(F) = \{\mathfrak{r}_1 < \mathfrak{r}_2 < \cdots < \mathfrak{r}_k\}$, i.e., the $\mathfrak{r}_h$ are the different ranks assigned by r to vertices in F. Now, define $F_h = \{v \in F \mid r(v) \leq \mathfrak{r}_h\}$ for $1 \leq h \leq k$, i.e., we order the vertices in F into a hierarchy $F_1 \subseteq F_2 \subseteq \cdots \subseteq F_k$ according to their rank with the intuition that smaller ranks are preferable for Player 0. Let r'_h be the least fixed point of ℓ_{F_h} for $1 \leq h \leq k$ and $r''_h = \mathrm{cmplt}(r'_h)$. Then, we

	v_4		v_3		v_2		v_1		v_0	

r_0:   0   0   0   0   0
r_1:   ∞   ∞   3   2   4
r_2:   ∞   ∞   ∞   3   4
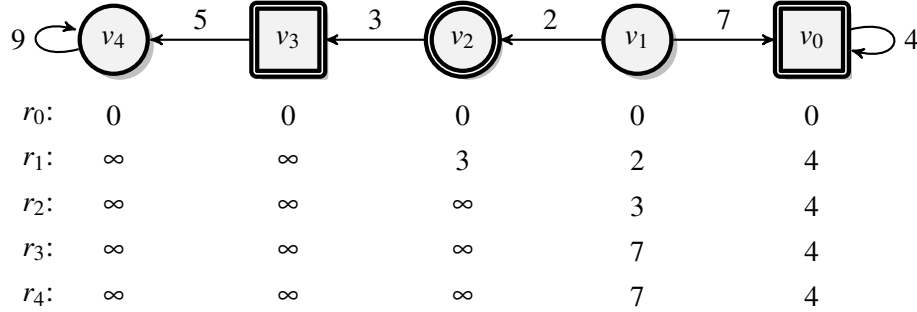r_3:   ∞   ∞   ∞   7   4
r_4:   ∞   ∞   ∞   7   4

Figure 3: The arena for Example 4 and the evolution of the corresponding rankings.

define the ranking $\ell_L(r)$ via

$$\ell_L(r)(v) = \min_{1 \le h \le k} (\max\{r(v), r_h''(v), \mathfrak{r}_h\}),$$

i.e., to compute the new rank of v we take into account the old rank and then minimize over the maximum of the weight to reach some F_h and the maximal old rank of the vertices in F_h, which indicates (in the fixed point) how costly it is to reach F repeatedly from this vertex.

Remark 6. *We have $r \sqsupseteq \ell_L(r)$ for every ranking r.*

Now, let r_0 be the ranking mapping every vertex to 0, i.e., the $\sqsubseteq$-largest ranking, and define $r_{j+1} = \ell_L(r_j)$ for every $j > 0$.

Example 4. *Consider the game in Figure 3 and focus on vertex v_1. Its rank is updated from its initial value of 0 to 2 (because the vertex v_2 in F can be reached with weight 2) and then 3 (because reaching F once more from v_2 incurs weight $3 = \max\{2,3\}$) and then to 7 (as F is no longer reachable from v_3, but from v_0 which incurs weight $\max\{4,7\}$).*

To begin our proof of correctness, we show that the ranks assigned by the r_j are bounded by some polynomial that only depends on $\mathscr{A}$ and $\mathfrak{A}$ (but is exponential if weights are encoded in binary). In particular, this implies that there is some n such that $r_n = r_{n+1}$. Again, we denote r_n for the smallest such n as r^* (which is the greatest fixed point of ℓ_L).

Lemma 4. *Let $v \in V$ and $j \ge 0$. If $r_j(v) < \infty$ then $r_j(v) \le (|\mathscr{A}| \cdot |\mathfrak{A}| + 1) \cdot W$, where W is the largest weight in $\mathscr{A}$.*

In the following, consider the application of ℓ_L to r^*: let the $\mathfrak{r}_h$, F_h, r_h', and r_h'' be computed with respect to r^* as described above. For every $v \in V$, let $h(v)$ be such that

$$r^*(v) = \min_{1 \le h \le k} (\max\{r^*(v), r_h''(v), \mathfrak{r}_h\}) = \max\{r^*(v), r_{h(v)}''(v), \mathfrak{r}_{h(v)}\}.$$

If there are several possible values for $h(v)$, we pick the smallest one with this property (although this is inconsequential).

Next, we define a finite-state strategy σ' for Player 0 in $\mathscr{A} \times \mathscr{M}_{\mathfrak{A}}$ implemented by a memory structure $\mathscr{M}' = (M', \mathrm{init}', \mathrm{upd}')$ with $M' = \{1, \cdots, k\}$, $\mathrm{init}'(v) = h(v)$, and $\mathrm{upd}'(h, v) = h$, if $v \notin F_h$, and $\mathrm{upd}'(h, v) = \mathrm{init}'(v)$, if $v \in F_h$. Thus, the memory is initalized to $h(v)$ when starting at v and stays constant until a vertex $v' \in F_{h(v)}$ is visited. While moving to v', the memory is again initialized to $h(v')$ and stays constant until $F_{h(v')}$ is visited. This procedure is repeated ad infinitum. It remains to define the next-move

function: $\mathrm{Nxt}'(v, h)$ is an optimal successor of v with respect to r'_h, if $v \notin F_h$, and an optimal successor of v with respect to r''_h, if $v \in F_h$. Let σ' be the strategy implemented by $\mathcal{M}'$ and Nxt' in $\mathcal{A} \times \mathcal{M}$ and let σ be the strategy induced by $\mathcal{M}$ and σ' in $\mathcal{A}$.

Lemma 5. *We have* $val_{\mathcal{G}}(\sigma, v) \leq r^*(v, init(v))$ *for every* v *in* $\mathcal{A}$.

Recall that we have a sequence $r_0 \sqsupseteq r_1 \sqsupseteq \cdots \sqsupseteq r_n = r_{n+1} = r^*$ of rankings with $r_{j+1} = \ell_L(r_j)$ for every $j \leq n$. Here, we define the settling time $t_s(v)$ of a vertex $v \in V$ as the minimal j with $r_j(v) = r^*(v)$.

Remark 7. $r^*(v) > 0$ *implies* $t_s(v) > 0$ *and* $r_{t_s(v)-1}(v) < r_{t_s(v)}(v)$.

Next, we define a finite-state strategy τ' for Player 1 in $\mathcal{A} \times \mathcal{M}_{\mathfrak{A}}$ implemented by a memory structure $\mathcal{M}' = (M', init' upd')$ with $M' = V$, $init'(v) = v$, and $upd'(v, v') = v$, if $v' \notin F$, and $upd'(v, v') = init'(v') = v'$, if $v' \in F$ (recall that the first argument of an update function is the current memory state and the second one a vertex). To define the next-move function, we distinguish three types of vertices $v \in V$.

We say v is of type zero, if $r^*(v) = 0$. If this is not the case, i.e., if $r^*(v) > 0$, then we have

$$r^*(v) = r_{t_s(v)}(v) = \min_h(\max\{r''_h(v), \mathfrak{r}_h\}) \tag{1}$$

due to Remark 7, where the r''_h and $\mathfrak{r}_h$ are computed with respect to $r_{t_s(v)-1}$. Now, we say v is of type one, if there is an h such that $r^*(v) = r''_h(v)$. Then, we define $h(v)$ to be the maximal h with this property.

Finally, if there is no h with $r^*(v) = r''_h(v)$, then we must have $r^*(v) = \mathfrak{r}_h$ for some h. Due to the $\mathfrak{r}_h$ being strictly increasing, there is a unique $h = h(v)$ with this property. In this case, we say v is of type two.

Now, if v is of type zero, then we define $\mathrm{Nxt}'(v', v)$ to be an arbitrary successor of v' (recall that the first argument of a next-move function is the current vertex and the second one the current memory state). If v is of type one, then we define $\mathrm{Nxt}'(v'v)$ to be an optimal successor of v' with respect to $r''_{h(v)}$. Finally, if v is of type two and we have $h(v) = 1$, then let $\mathrm{Nxt}'(v'v)$ be an arbitrary successor of v'. On the other hand, if v is of type two and we have $h(v) > 1$, then let $\mathrm{Nxt}'(v'v)$ be an optimal successor of v' with respect to $r''_{h(v)-1}$. Let τ' be the strategy implemented by $\mathcal{M}'$ and Nxt' in $\mathcal{A} \times \mathcal{M}$ and let τ be the strategy induced by $\mathcal{M}$ and τ' in $\mathcal{A}$.

Lemma 6. *We have* $val_{\mathcal{G}}(\tau, v) \geq r^*(v, init(v))$ *for every* v *in* $\mathcal{A}$.

Lemmata 5 and 6 imply that σ and τ are optimal strategies (where optimality of Player 1 strategies is defined as expected), i.e., the first part of our main theorem is proven.

The construction of τ also yields an upper bound on the number of iterations of ℓ_L that are necessary to reach the fixed point.

Lemma 7. *We have* $r^* = r_{|F|+1}$.

It remains to determine the overall running time of our algorithm. Recall that we have defined F to be the product of the set of vertices of the arena $\mathcal{A}$ and the accepting states of $\mathfrak{A}$. Untangling the construction above shows that the fixed point of ℓ_L can be computed in time $\mathcal{O}(n^3 e s^2 f^2)$, where n and e are the number of vertices and edges of $\mathcal{A}$ and s and f are the number of states and accepting states of $\mathfrak{A}$: Due to Lemma 7, it takes at most $|F| + 1 = n \cdot f + 1$ applications of ℓ_L to reach the fixed point, each taking at most $|F|$ computations of a fixed point of $\ell_{F'}$. Each of these takes at most $n \cdot s + 1$ applications of ℓ, which each takes time $e \cdot s$ in the unit-cost model.

Note that optimal strategies for Player 0 in $\mathcal{A}$ are implemented by memory structures that do not need to keep track of weights of play prefixes, only pairs of vertices and states. The following corollary gives an upper bound on the size and quality of optimal strategies.
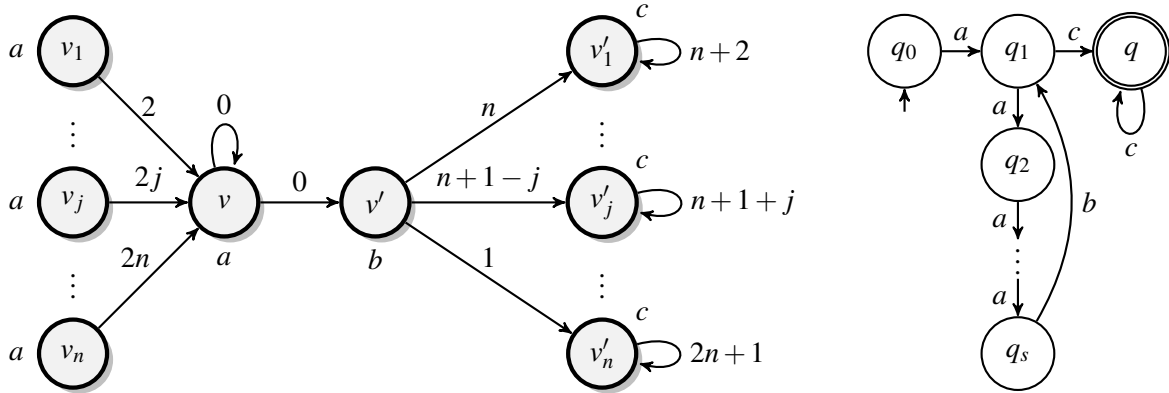
Figure 4: The arena $\mathscr{A}_n$ (left) and the automaton $\mathfrak{A}_s$ (right) for the lower bounds in Example 5.1. Here, a, b, and c are the colors of the vertices. Furthermore, all missing transitions of the automaton lead to a rejecting sink state that is not drawn for the sake of readability.

Lemma 8. *Let $\mathscr{G} = (\mathscr{A}, \lim(L(\mathfrak{A})))$ be a weighted reachability game with n vertices and largest weight W, and let s and f be the number of states and accepting states of $\mathfrak{A}$. Then, Player 0 has an optimal strategy for $\mathscr{G}$ of size nsf with $val_{\mathscr{G}}(v) \leq (ns+1) \cdot W$ for all vertices v with $val_{\mathscr{G}}(v) < \infty$.*

The following example shows that both the upper bound on the memory size and the upper bound on the value of an optimal strategy are (almost) tight.

Example 5.

1. *We begin with the lower bound on the memory. Consider the arena $\mathscr{A}_n$ and the automaton $\mathfrak{A}_s$ (for $n > 0$ and $s > 1$) depicted in Figure 4 inducing the game $\mathscr{G}_{n,s}$. The automaton accepts the language $a(a^{s-1}b)^*c$. Note that we can identify (winning) strategies for Player 0 with (winning) plays, as all vertices are controlled by Player 0. Also, from every vertex v_j there is a unique play (strategy) $\rho_j = v_j v^{s-1} v'(v'_j)^\omega$ with $val_{\mathscr{G}_{n,s}}(\rho_j) = n + 1 + j$. Every other play starting in v_j has a larger value. Hence, there is a unique optimal strategy for Player 0, which, for every j, yields the play ρ_j when starting in v_j.*

   Furthermore, standard pumping arguments show that every strategy for Player 0 yielding, for every j, the play ρ_j when starting at v_j has at least $n(s-1)$ states, which are required to reach v'_j when starting at v_j and to be able to traverse the self-loop at the vertex v exactly $n-2$ times, as required by the winning condition. Note that this lower bound does not take the number of accepting states into account, i.e., it is not completely tight.

2. *Next, we consider the lower bound on the value of an optimal strategy for Player 0. Figure 5 depicts an arena $\mathscr{A}_m$ and a DFA $\mathfrak{A}_n$ (for $m > 1$ and $n > 1$), which accepts the language $((a^{n-1}b)^*c)^*$. Note that we can identify (winning) strategies for Player 0 with (winning) plays, as all vertices are controlled by Player 0. Actually, there is a unique winning play (i.e., winning strategy) for Player 0 starting in v_1, i.e., the play*

$$((v_1)^{s-1} v'_1 (v_2)^{s-1} v'_2 \cdots (v_n)^{s-1} v'_n v)^\omega$$

   with value mnW. Hence, the value of an optimal strategy from v_1 is mnW.

The lower bound on the value presented above is tight while the lower bound on the memory is off by a factor of f, where f is the number of accepting states of the automaton. We expect that the upper bound
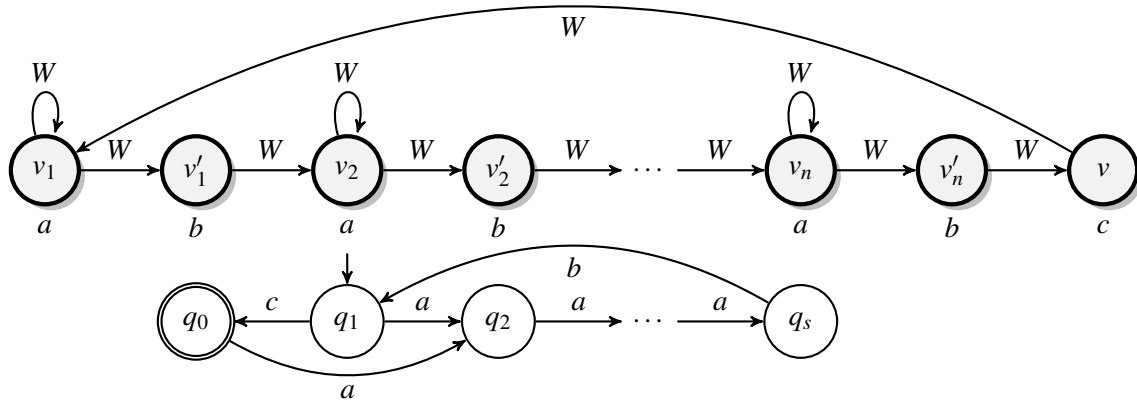
Figure 5: The arena $\mathscr{A}_n$ (top) and the automaton $\mathfrak{A}_s$ (bottom) for the lower bounds in Example 5.2. Here, W is an arbitrary nonnegative integer and a, b, and c are the colors of the vertices. Furthermore, all missing transitions of the automaton lead to a rejecting sink state that is not drawn for the sake of readability.

can be improved by removing the factor f by exploiting some monotonicity properties. In particular, this should be true in the case where we are not constructing a uniform optimal strategy, i.e., one that is optimal from every vertex. Recall the game presented in Example 5.1: here, the factor n in the memory requirement is due to the fact that the strategy intuitively has to memorize the vertex v_j the play starts in in order to move to the corresponding v'_j to achieve the optimal value. On the other hand, a strategy that is only optimal from some fixed v_j does not have to store the initial vertex but can instead always move to v'_j and thus only needs $v - 1$ memory states. Whether the upper bound can be improved in this setting is left open for further work.

4 Limit Games in Infinite Arenas

The (qualitative) winning region $W_i(\mathscr{G})$ of Player i in a regular weighted limit game $\mathscr{G}$ contains all vertices v from which Player i has a winning strategy. In the previous section, we have considered a quantitative notion of winning by measuring the quality of strategies. For finite arenas, it turns out that our quantitative notion is a refinement of the qualitative one.

Lemma 9. *Let $\mathscr{G} = (\mathscr{A}, \lim(L(\mathfrak{A})))$ be a regular weighted limit game and let σ be an optimal strategy for Player 0 in $\mathscr{G}$. Then, $W_0(\mathscr{G}) = \{v \mid val_{\mathscr{G}}(\sigma, v) < \infty\}$ and $W_1(\mathscr{G}) = \{v \mid val_{\mathscr{G}}(\sigma, v) = \infty\}$.*

The previous refinement result relies on the finiteness of the arena. In fact, it is no longer valid in infinite arenas, even in very simple ones with unit weights.

Example 6. *Consider the infinite arena presented in Figure 6 and $K = (ab^+c^+)^*ab^*$, i.e., Player 0 wins every play starting in the vertex colored by a. Furthermore, the value of a play is equal to the length of the longest infix with label sequence in c^*a.*

Now consider the play ρ with coloring

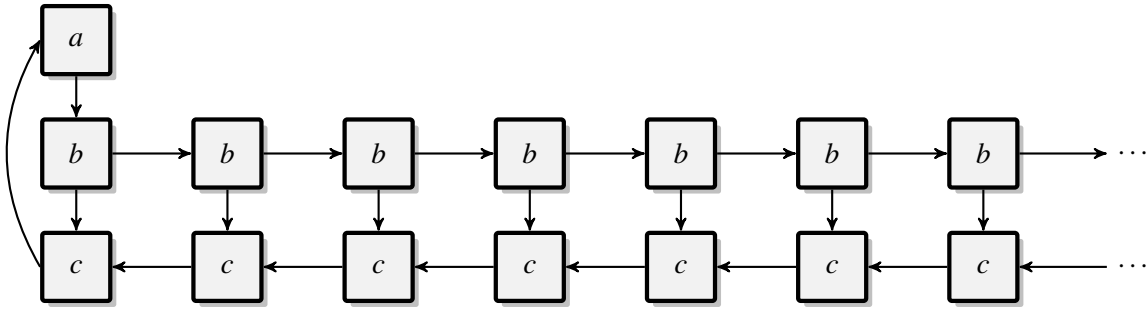$$abc\,abbcc\,abbbccc\,abbbbcccc\,abbbbbccccc\cdots.$$

Figure 6: The arena for Example 6. Vertices are labeled by their colors and every edge has weight 1.

It is winning for Player 0*, has value* ∞ *(as the length of c-blocks is unbounded), and consistent with every strategy for Player* 0*, as Player* 1 *moves at every vertex.*

Hence, although Player 0 *wins from the vertex with label a, she does not have a strategy with finite value from this vertex.*

Note that the graph underlying the arena in Example 6 is a configuration graph of a one-counter machine, a particularly simple class of infinite graphs with many desirable decidability properties (see, e.g., [32] for games on such graphs). Nevertheless, quantitative winning no longer refines qualitative winning.

As mentioned above, the proof of the refinement lemma relies crucially on the finiteness of the arena, which yields the upper bound on the values of an optimal strategy. Hence, on infinite arenas, there are three classes of vertices: those from which Player 0 can win with a bounded value, those from which she can win, but not with a bounded value, and those from which she cannot win at all. Thus, the landscape for infinite arenas is, in a sense, much more interesting than for finite arenas and being able to win even with a finite value is more useful than just being able to win.

5   Related Work

Quantitative infinite-duration games have received considerable attention, e.g., in the form of games with mean-payoff conditions [2, 18, 30, 38] and other payoff conditions [6, 22, 38], energy conditions [4, 15, 24, 33], quantitative logics for specifying winning conditions [1, 19, 26, 36, 37], variations of the classical parity condition [11, 12, 13, 20, 31], and other models [3, 5, 7]. Weighted limit games are related to some of these models.

In particular, the problem of determining the value of an optimal strategy in a weighted limit game is related to the optimal cover problem for one-dimensional consumption games [5]. Such a game is also played in a weighted arena and while an edge with weight w is traversed, a battery is discharged by w units. Furthermore, there are special edges that allow to recharge the battery to an arbitrary amount. Now, the optimal cover problem asks to compute the minimal battery capacity that allows Player 0 to play indefinitely without ever completely depleting the battery.

As long as the arena does not contain any cycles consisting only of edges with weight 0, one can turn a weighted limit game into a consumption game: After every visit to a vertex in F, the battery is recharged and then drained by the weight along the edges until F is visited again. Now, one can show that the minimal sufficient capacity for the battery corresponds to the value of an optimal strategy. However, in the presence of cycles of weight 0, this correspondence no longer holds, as such a cycle is sufficient for

Player 0 to not drain the battery, while this is not sufficient in a weighted limit game if the cycle does not contain a vertex from F. Formulated differently: consumption games have a safety winning condition while a limit game has a liveness condition.[2]

On the other hand, synthesis of optimal strategies in weighted limit games can be seen as a special case of the optimization problem for Prompt-LTL with costs [37].[3] This is an extension of classical LTL [29] by the prompt-eventually $\mathbf{F}_P$ [26]: The formula $\mathbf{F}_P\varphi$ holds with respect to a bound k on some weighted trace π, if π can be decomposed into $\pi = \pi_0\pi_1$ such that the weight of π_0 is at most k and π_1 satisfies φ with respect to k. Intuitively, φ has to be satisfied within a prefix of weight at most k. Now, the formula $\mathbf{GF}_Pa$ with respect to a bound k expresses that the atomic proposition a holds infinitely often and that the weight between consecutive occurrences is bounded by k. So, computing the minimal k for which Player 0 has a winning strategy for the game with winning condition $\mathbf{GF}_Pa$, where a holds exactly at the vertices in F, yields the value of an optimal strategy. Furthermore, a witnessing winning strategy can be computed [37].

Finally, weighted limit games can be seen as a special case of two-color parity games with costs [20] (with binary encoding [35]), a variant of parity games where Player 0 aims to minimize the weight between the occurrences of odd colors and the next larger even color. An optimal strategy for the parity game with costs [35] is also optimal for the weighted limit game.

However, all three approaches do not yield the fine-grained complexity analysis presented here, e.g., tight upper and lower bounds on the memory requirements and values of optimal strategies.

6   Conclusion

In this work, we have considered the problem of computing optimal strategies in regular weighted limit games. Such strategies always exist in finite arenas, and are efficiently computable by a fixed point algorithm. Furthermore, we have shown that allowing negative weights leads to games without optimal strategies and how the relation between qualitative and quantitative winning is affected by considering infinite arenas.

The case of infinite arenas is also a promising direction for further work. We conjecture that our fixed point characterization can be lifted to limit games in infinite arenas as well, with some minor adaptions to account for infinite branching and using transfinite induction to obtain the fixed points. However, these are no longer effective, due to the infiniteness of the arena. Instead, it seems promising to consider saturation-based methods [9, 10].

Another direction for further work is concerned with more general definitions for the value of a play. Here, we have accumulated the weight of certain infixes. Instead, one could, e.g., consider the average weight of these infixes.

Finally, another promising direction for further work concerns quantitative winning conditions, e.g., limit conditions, in games with imperfect information [17].

---

[2]Note that a visit to F could be enforced by having a second dimension that implements a countdown timer that is decremented along each edge.

[3]Prompt-LTL with costs is the fragment of Parametric LTL with costs allowing only one parameter, which is introduced in [37] without a name.

References

[1] Rajeev Alur, Kousha Etessami, Salvatore La Torre & Doron A. Peled (2001): *Parametric temporal logic for "model measuring"*. *ACM Trans. Comput. Log.* 2(3), pp. 388–407, doi:10.1145/377978.377990.

[2] Henrik Björklund & Sergei G. Vorobyov (2007): *A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games*. *Discret. Appl. Math.* 155(2), pp. 210–229, doi:10.1016/j.dam.2006.04.029.

[3] Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger & Barbara Jobstmann (2009): *Better Quality in Synthesis through Quantitative Objectives*. In: *CAV 2009*, *LNCS* 5643, Springer, pp. 140–156, doi:10.1007/978-3-642-02658-4_14.

[4] Patricia Bouyer, Nicolas Markey, Mickael Randour, Kim G. Larsen & Simon Laursen (2018): *Average-energy games*. *Acta Informatica* 55(2), pp. 91–127, doi:10.1007/s00236-016-0274-1.

[5] Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera & Petr Novotný (2012): *Efficient Controller Synthesis for Consumption Games with Multiple Resource Types*. In P. Madhusudan & Sanjit A. Seshia, editors: *CAV 2012*, *LNCS* 7358, Springer, pp. 23–38, doi:10.1007/978-3-642-31424-7_8.

[6] Thomas Brihaye, Gilles Geeraerts, Axel Haddad & Benjamin Monmege (2015): *To Reach or not to Reach? Efficient Algorithms for Total-Payoff Games*. In Luca Aceto & David de Frutos-Escrig, editors: *CONCUR 2015*, *LIPIcs* 42, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 297–310, doi:10.4230/LIPIcs.CONCUR.2015.297.

[7] Véronique Bruyère, Emmanuel Filiot, Mickael Randour & Jean-François Raskin (2017): *Meet your expectations with guarantees: Beyond worst-case synthesis in quantitative games*. *Inf. Comput.* 254, pp. 259–295, doi:10.1016/j.ic.2016.10.011.

[8] J. Richard Büchi & Lawrence H. Landweber (1969): *Solving Sequential Conditions by Finite-State Strategies*. *Trans. Amer. Math. Soc.* 138, pp. pp. 295–311, doi:10.2307/1994916.

[9] Thierry Cachat (2002): *Symbolic Strategy Synthesis for Games on Pushdown Graphs*. In: *ICALP 2002*, *LNCS* 2380, Springer, pp. 704–715, doi:10.1007/3-540-45465-9_60.

[10] Arnaud Carayol & Matthew Hague (2018): *Optimal Strategies in Pushdown Reachability Games*. In: *MFCS 2018*, *LIPIcs* 117, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 42:1–42:14, doi:10.4230/LIPIcs.MFCS.2018.42.

[11] Krishnendu Chatterjee & Laurent Doyen (2012): *Energy parity games*. *Theor. Comput. Sci.* 458, pp. 49–60, doi:10.1016/j.tcs.2012.07.038.

[12] Krishnendu Chatterjee, Thomas A. Henzinger & Florian Horn (2009): *Finitary winning in omega-regular games*. *ACM Trans. Comput. Log.* 11(1), pp. 1:1–1:27, doi:10.1145/1614431.1614432.

[13] Krishnendu Chatterjee, Thomas A. Henzinger & Marcin Jurdzinski (2005): *Mean-Payoff Parity Games*. In: *LICS 2005*, IEEE Computer Society, pp. 178–187, doi:10.1109/LICS.2005.26.

[14] Krishnendu Chatterjee, Thomas A. Henzinger & Nir Piterman (2008): *Algorithms for Büchi Games*. arXiv 0805.2620. Available at `http://arxiv.org/abs/0805.2620`.

[15] Krishnendu Chatterjee, Mickael Randour & Jean-François Raskin (2014): *Strategy synthesis for multi-dimensional quantitative objectives*. *Acta Informatica* 51(3-4), pp. 129–163, doi:10.1007/s00236-013-0182-6.

[16] Alonzo Church (1963): *Logic, arithmetic, and automata*. In: *Proc. Int. Congr. Math. 1962*, Inst. Mittag-Lefler, Djursholm, Sweden, pp. 23–35.

[17] L. Doyen & J.-F. Raskin (2011): *Games with Imperfect Information: Theory and Algorithms*. In: *Lectures in Game Theory for Computer Scientists*, Cambridge University Press, pp. 185–212, doi:10.1017/CBO9780511973468.007.

[18] Andrzej Ehrenfeucht & Jan Mycielski (1979): *Positional strategies for mean payoff games*. *Int. J. Game Theory* 8, pp. 109–113, doi:10.1007/BF01768705.

[19] Peter Faymonville & Martin Zimmermann (2017): *Parametric Linear Dynamic Logic*. *Inf. Comput.* 253, pp. 237–256, doi:10.1016/j.ic.2016.07.009.

[20] Nathanaël Fijalkow & Martin Zimmermann (2014): *Parity and Streett Games with Costs*. *Log. Methods Comput. Sci.* 10(2), doi:10.2168/LMCS-10(2:14)2014.

[21] Bernd Finkbeiner (2016): *Synthesis of Reactive Systems*. In Javier Esparza, Orna Grumberg & Salomon Sickert, editors: *Dependable Software Systems Engineering*, *NATO Science for Peace and Security Series - D: Information and Communication Security* 45, IOS Press, pp. 72–98, doi:10.3233/978-1-61499-627-9-72.

[22] Hugo Gimbert & Wieslaw Zielonka (2004): *When Can You Play Positionally?* In Jirí Fiala, Václav Koubek & Jan Kratochvíl, editors: *MFCS 2004*, *LNCS* 3153, Springer, pp. 686–697, doi:10.1007/978-3-540-28629-5_53.

[23] Erich Grädel, Wolfgang Thomas & Thomas Wilke, editors (2002): *Automata, Logics, and Infinite Games: A Guide to Current Research*. *LNCS* 2500, Springer, doi:10.1007/3-540-36387-4.

[24] Line Juhl, Kim Guldstrand Larsen & Jean-François Raskin (2013): *Optimal Bounds for Multiweighted and Parametrised Energy Games*. In Zhiming Liu, Jim Woodcock & Huibiao Zhu, editors: *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, *LNCS* 8051, Springer, pp. 244–255, doi:10.1007/978-3-642-39698-4_15.

[25] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, Vladimir Gurvich, Gábor Rudolf & Jihui Zhao (2008): *On Short Paths Interdiction Problems: Total and Node-Wise Limited Interdiction*. *Theory Comput. Syst.* 43(2), pp. 204–233, doi:10.1007/s00224-007-9025-6.

[26] Orna Kupferman, Nir Piterman & Moshe Y. Vardi (2009): *From liveness to promptness*. *Formal Methods Syst. Des.* 34(2), pp. 83–103, doi:10.1007/s10703-009-0067-z.

[27] Lawrence H. Landweber (1969): *Decision Problems for omega-Automata*. *Math. Syst. Theory* 3(4), pp. 376–384, doi:10.1007/BF01691063.

[28] Aniello Murano, Sasha Rubin & Martin Zimmermann (2020): *Optimal Strategies in Weighted Limit Games (full version)*. *arXiv* 2008.11562. Available at `https://arxiv.org/abs/2008.11562`.

[29] Amir Pnueli (1977): *The Temporal Logic of Programs*. In: *FOCS 1977*, IEEE Computer Society, pp. 46–57, doi:10.1109/SFCS.1977.32.

[30] Anuj Puri (1995): *Theory of hybrid systems and discrete event systems*. Ph.D. thesis, Computer Science Department, University of California, Berkeley.

[31] Sven Schewe, Alexander Weinert & Martin Zimmermann (2019): *Parity Games with Weights*. *Log. Methods Comput. Sci.* 15(3), doi:10.23638/LMCS-15(3:20)2019.

[32] Olivier Serre (2006): *Parity Games Played on Transition Graphs of One-Counter Processes*. In: *FOSSACS 2006*, *LNCS* 3921, Springer, pp. 337–351, doi:10.1007/11690634_23.

[33] Frank Thuijsman & Otto J. Vrieze (1987): *The bad match; A total reward stochastic game*. *OR Spektrum* 9(2), pp. 93–99, doi:10.1007/BF01732644.

[34] Nico Wallmeier, Patrick Hütten & Wolfgang Thomas (2003): *Symbolic Synthesis of Finite-State Controllers for Request-Response Specifications*. In Oscar H. Ibarra & Zhe Dang, editors: *CIAA 2003*, *LNCS* 2759, Springer, pp. 11–22, doi:10.1007/3-540-45089-0_3.

[35] Alexander Weinert & Martin Zimmermann (2017): *Easy to Win, Hard to Master: Optimal Strategies in Parity Games with Costs*. *Log. Methods Comput. Sci.* 13(3), doi:10.23638/LMCS-13(3:29)2017.

[36] Martin Zimmermann (2013): *Optimal bounds in parametric LTL games*. *Theor. Comput. Sci.* 493, pp. 30–45, doi:10.1016/j.tcs.2012.07.039.

[37] Martin Zimmermann (2018): *Parameterized linear temporal logics meet costs: still not costlier than LTL*. *Acta Informatica* 55(2), pp. 129–152, doi:10.1007/s00236-016-0279-9.

[38] Uri Zwick & Mike Paterson (1996): *The Complexity of Mean Payoff Games on Graphs*. *Theor. Comput. Sci.* 158(1&2), pp. 343–359, doi:10.1016/0304-3975(95)00188-3.