Symmetric Strategy Improvement*

Sven Schewe[1], Ashutosh Trivedi[2], and Thomas Varghese[1]

[1] University of Liverpool
[2] Indian Institute of Technology Bombay

Abstract. Symmetry is inherent in the definition of most of the two-player zero-sum games, including parity, mean-payoff, and discounted-payoff games. It is therefore quite surprising that no symmetric analysis techniques for these games exist. We develop a novel symmetric strategy improvement algorithm where, in each iteration, the strategies of both players are improved simultaneously. We show that symmetric strategy improvement defies Friedmann's traps, which shook the belief in the potential of classic strategy improvement to be polynomial.

1 Introduction

We study turn-based graph games between two players—Player Min and Player Max—who take turns to move a token along the vertices of a coloured finite graph so as to optimise their adversarial objectives. Various classes of graph games are characterised by the objective of the players, for instance in *parity games* the objective is to optimise the parity of the dominating colour occurring infinitely often, while in *discounted and mean-payoff games* the objective is the discounted and limit-average sum of the colours. Solving graph games is the central and most expensive step in many model checking [17, 8, 32, 6, 1], satisfiability checking [32, 17, 30, 27], and synthesis [23, 28] algorithms. More efficient algorithms for solving graph games will therefore foster the development of performant model checkers and contribute to bringing synthesis techniques to practice.

Parity games enjoy a special status among graph games and the quest for performant algorithms [17, 9, 7, 21, 34, 4, 33, 15, 20, 24, 31, 3, 22, 18, 2, 16, 25, 26, 11] for solving them has therefore been an active field of research during the last decades. Traditional forward techniques ($\approx O(n^{\frac{1}{2}c})$ [15] for parity games with n positions and c colours), backward techniques ($\approx O(n^c)$ [21, 9, 33]), and their combination ($\approx O(n^{\frac{1}{3}c})$ [25]) provide good complexity bounds. These bounds are sharp, and techniques with good complexity bounds [25, 15] frequently display their worst case complexity on practical examples. On the other hand, strategy improvement algorithms [20, 24, 31, 3, 26, 11], a class of algorithms closely related to the Simplex for linear programming, are known to perform well in practice.

* The work has been done while the second author was visiting the University of Liverpool supported by a Liverpool India Fellowship.

The standard strategy improvement algorithms are built around the existence of optimal positional strategies for both players. They start with an arbitrary positional strategy for a player and iteratively compute a better positional strategy in every step until the strategy cannot be further improved. Since there are only finitely many positional strategies in a finite graph, termination is guaranteed. The crucial step in a strategy improvement algorithm is to compute a better strategy from the current strategy. Given a current strategy σ of a player (say, Player Max), this step is performed by first computing the globally optimal counter strategy τ_σ^c of the opponent (Player Min) and then computing the value of each vertex of the game restricted to the strategies σ and τ_σ^c. For the games under discussion (parity, discounted, and mean-payoff) both of these computations are simple and tractable. This value dictates potentially locally profitable changes or switches $\mathsf{Prof}(\sigma)$ that Player Max can make vis-à-vis his previous strategy σ. For the correctness of the strategy improvement algorithm it is required that such locally profitable changes imply a global improvement. The strategy of Player Max can then be updated according to a *switching rule* (akin to pivoting rule of the Simplex) in order to give an improved strategy. This has led to the following template for classic strategy improvement algorithms.

---

1 determine an optimal counter strategy τ_σ^c for σ
2 evaluate the game for σ and τ_σ^c and determine profitable changes $\mathsf{Prof}(\sigma)$ for σ
3 update σ by applying changes from $\mathsf{Prof}(\sigma)$ to σ

A number of switching rules, including the ones inspired by Simplex pivoting rules, have been suggested for strategy improvement algorithms. The most widespread ones are to select changes for all game states where this is possible, choosing a combination of those with an optimal update guarantee, or to choose uniformly at random. For some classes of games, it is also possible to select an optimal combination of updates [26]. There have also been suggestions to use more advanced randomisation techniques with sub-exponential – $2^{O(\sqrt{n})}$ – bounds [3] and snare memory [11]. Unfortunately, all of these techniques have been shown to be exponential in the size of the game [12–14].

Classic strategy improvement algorithms treat the two players involved quite differently where at each iteration one player computes a globally optimal counter strategy, while the other player performs local updates. In contrast, a *symmetric strategy improvement* algorithm symmetrically improves the strategies of both players at the same time, and uses the finding to guide the strategy improvement. This suggests the following naïve symmetric approach.

---

1 determine $\tau' = \tau_\sigma^c$	determine $\sigma' = \sigma_\tau^c$
2 update σ to σ'	update τ to τ'

This algorithm has earlier been suggested by Condon [5] where it was shown that a repeated application of this update can lead to cycles [5]. A problem with this naïve approach is that there is no guarantee that the primed strategies

are generally better than the unprimed ones. With hindsight this is maybe not very surprising, as in particular no improvement in the evaluation of running the game with σ', τ' can be expected over running the game with σ, τ, as an improvement for one player is on the expense of the other. This observation led to the approach being abandoned.

The key contribution of this paper is the following more careful symmetric strategy improvement algorithm that guarantees improvements in each iteration similar to classic strategy improvement.

1 determine τ^c_σ	determine σ^c_τ
2 determine $\mathsf{Prof}(\sigma)$ for σ	determine $\mathsf{Prof}(\tau)$ for τ
3 update σ using $\mathsf{Prof}(\sigma) \cap \sigma^c_\tau$	update τ using $\mathsf{Prof}(\tau) \cap \tau^c_\sigma$

Observe that the main difference to classic strategy improvement approaches is that we exploit the strategy of the other player to inform the search for a good improvement step. In this algorithm we select only such updates to the two strategies that agree with the optimal counter strategy to the respective other's strategy. We believe that this will provide a gradually improving advice function that will lead to few iterations. We support this assumption by showing that this algorithm suffices to escape the traps Friedmann has laid to establish lower bounds for different types of strategy improvement algorithms [12–14].

2 Preliminaries

We focus on turn-based zero-sum games played between two players—Player Max and Player Min—over finite graphs. A game arena $\mathcal{A}$ is a tuple $(V_{\text{Max}}, V_{\text{Min}}, E, C, \phi)$ where $(V = V_{\text{Max}} \cup V_{\text{Min}}, E)$ is a finite directed graph with the set of vertices V partitioned into a set V_{Max} of vertices controlled by Player Max and a set V_{Min} of vertices controlled by Player Min, $E \subseteq V \times V$ is the set of edges, C is a set of colours, $\phi : V \to C$ is the colour mapping. We require that every vertex has at least one outgoing edge.

A turn-based game over $\mathcal{A}$ is played between players by moving a token along the edges of the arena. A play of such a game starts by placing a token on some initial vertex $v_0 \in V$. The player controlling this vertex then chooses a successor vertex v_1 such that $(v_0, v_1) \in E$ and the token is moved to this successor vertex. In the next turn the player controlling the vertex v_1 chooses the successor vertex v_2 with $(v_1, v_2) \in E$ and the token is moved accordingly. Both players move the token over the arena in this manner and thus form a play of the game. Formally, a play of a game over $\mathcal{A}$ is an infinite sequence of vertices $\langle v_0, v_1, \ldots \rangle \in V^\omega$ such that, for all $i \geq 0$, we have that $(v_i, v_{i+1}) \in E$. We write $\mathsf{Plays}_{\mathcal{A}}(v)$ for the set of plays over $\mathcal{A}$ starting from vertex $v \in V$ and $\mathsf{Plays}_{\mathcal{A}}$ for the set of plays of the game. We omit the subscript when the arena is clear from the context. We extend the colour mapping $\phi : V \to C$ from vertices to plays by defining the mapping $\phi : \mathsf{Plays} \to C^\omega$ as $\langle v_0, v_1, \ldots \rangle \mapsto \langle \phi(v_0), \phi(v_1), \ldots \rangle$.

A graph game $\mathcal{G}$ is a tuple $(\mathcal{A}, \eta, \prec)$, where $\mathcal{A}$ is an *arena*, $\eta \colon C^\omega \to \mathbb{D}$ is an evaluation function, and $\mathbb{D}$ is equipped with a preference order $\prec$. Parity,

mean-payoff and discounted payoff games are graph games $(\mathcal{A}, \eta, \prec)$ played on game arenas $\mathcal{A} = (V_{\text{Max}}, V_{\text{Min}}, E, \mathbb{R}, \phi)$. For mean payoff games the evaluation function is $\eta : \langle c_0, c_1, \ldots \rangle \mapsto \liminf_{i\to\infty} \frac{1}{i} \sum_{j=0}^{i-1} c_j$, while for discounted payoff games with discount factor $\lambda \in [0, 1)$ it is $\eta : \langle c_0, c_1, \ldots \rangle \mapsto \sum_{i=0}^{\infty} \lambda^i c_i$ with $\prec$ as the natural order over the reals. For (max) parity games the evaluation function is $\eta : \langle c_0, c_1, \ldots \rangle \mapsto \limsup_{i\to\infty} c_i$ often used with a preference order $\prec_{\text{parity}}$ where higher even colours are preferred over smaller even ones, even colours are preferred over odd ones, and smaller odd colours are preferred over higher ones.

In the remainder, we will use parity games where every colour is unique, i.e., where ϕ is injective. The reason for this assumption is that we extended [19], which implements variants of [31], and the lower bounds from [13, 12, 14] refer to such games. All parity games can be translated into such games as discussed in [31]. For these games, we use a valuation function based on their progress measure. We define η as $\langle c_0, c_1, \ldots \rangle \mapsto (c, C, d)$, where $c = \limsup_{i\to\infty} c_i$ is the dominant colour of the colour sequence, $d = \min\{i \in \omega \mid c_i = c\}$ is the index of the first occurrence of c, and $C = \{c_i \mid i < d, c_i > c\}$ is the set of colours that occur before the first occurrence of c. The preference order is such that $(c', C', d') \prec (c, C, d)$ if either $c' \prec_{\text{parity}} c$, or $c = c'$ and following holds:

- $C \neq C'$ and $h = \max((C \setminus C') \cup (C' \setminus C))$ is even and belongs to C,
- $C \neq C'$ and $h = \max((C \setminus C') \cup (C' \setminus C))$ is odd and belongs to C',
- $C = C'$, c is even and $d < d'$, or $C = C'$, c is odd and $d > d'$.

A strategy of Player Max is a function $\sigma : V^* V_{\text{Max}} \to V$ such that $\big(v, \sigma(\pi v)\big) \in E$ for all $\pi \in V^*$ and $v \in V_{\text{Max}}$. Similarly, a strategy of Player Min is a function $\tau : V^* V_{\text{Min}} \to V$ such that $\big(v, \sigma(\pi v)\big) \in E$ for all $\pi \in V^*$ and $v \in V_{\text{Min}}$. We write Σ^∞ and T^∞ for the set of strategies of Player Max and Player Min, respectively. For a strategy pair $(\sigma, \tau) \in \Sigma^\infty \times T^\infty$ and an initial vertex $v \in V$ we denote the unique play starting from the vertex v by $\pi(v, \sigma, \tau)$ and we write $\mathsf{val}_{\mathcal{G}}(v, \sigma, \tau)$ for the value of the vertex v under the strategy pair (σ, τ) defined as $\mathsf{val}_{\mathcal{G}}(v, \sigma, \tau) \stackrel{\text{def}}{=} \eta\big(\phi(\pi(v, \sigma, \tau))\big)$. We also define the concept of the value of a strategy $\sigma \in \Sigma^\infty$ and $\tau \in T^\infty$ as $\mathsf{val}_{\mathcal{G}}(v, \sigma) \stackrel{\text{def}}{=} \inf_{\tau \in T^\infty} \mathsf{val}_{\mathcal{G}}(v, \sigma, \tau)$ and $\mathsf{val}_{\mathcal{G}}(v, \tau) \stackrel{\text{def}}{=} \sup_{\sigma \in \Sigma^\infty} \mathsf{val}_{\mathcal{G}}(v, \sigma, \tau)$. We also extend the valuation for vertices to a valuation for the whole game by defining $|V|$-dimensional vectors $\mathsf{val}_{\mathcal{G}}(\sigma) : v \mapsto \mathsf{val}_{\mathcal{G}}(v, \sigma)$ with the usual $|V|$-dimensional partial order $\sqsubseteq$, where $\mathsf{val} \sqsubseteq \mathsf{val}'$ iff $\mathsf{val}(v) \preceq \mathsf{val}'(v)$ for all $v \in V$.

We say that a strategy $\sigma \in \Sigma^\infty$ is memoryless or *positional* if for all $\pi, \pi' \in V^*$ and $v \in V_{\text{Max}}$ we have that $\sigma(\pi v) = \sigma(\pi' v)$. Thus, a positional strategy can be viewed as a function $\sigma : V_{\text{Max}} \to V$ such that for all $v \in V_{\text{Max}}$ we have that $(v, \sigma(v)) \in E$. The concept of positional strategies of Player Min is defined in an analogous manner. We write Σ and T for the set of positional strategies of Players Max and Min, respectively. We say that a game is positionally determined if:

- $\mathsf{val}_{\mathcal{G}}(v, \sigma) = \min_{\tau \in T} \mathsf{val}_{\mathcal{G}}(v, \sigma, \tau)$ holds for all $\sigma \in \Sigma$,
- $\mathsf{val}_{\mathcal{G}}(v, \tau) = \max_{\sigma \in \Sigma} \mathsf{val}_{\mathcal{G}}(v, \sigma, \tau)$ holds for all $\tau \in T$,
- **Existence of value**: for all $v \in V$ $\max_{\sigma \in \Sigma} \mathsf{val}_{\mathcal{G}}(v, \sigma) = \min_{\tau \in T} \mathsf{val}_{\mathcal{G}}(v, \tau)$ holds, and we use $\mathsf{val}_{\mathcal{G}}(v)$ to denote this value, and

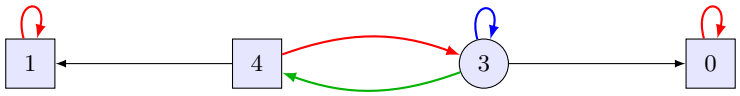

Fig. 1. Parity game arena with four vertices and unique colours.

– **Existence of globally positional optimal strategies**: there is a pair $\tau_{\min}, \sigma_{\max}$ of strategies such that, for all $v \in V$, $\mathsf{val}_{\mathcal{G}}(v) = \mathsf{val}_{\mathcal{G}}(v, \sigma_{\max}) = \mathsf{val}_{\mathcal{G}}(v, \tau_{\min})$ holds. Observe that for all $\sigma \in \Sigma$ and $\tau \in T$ we have that $\mathsf{val}_{\mathcal{G}}(\sigma_{\max}) \sqsupseteq \mathsf{val}_{\mathcal{G}}(\sigma)$ and $\mathsf{val}_{\mathcal{G}}(\tau_{\min}) \sqsubseteq \mathsf{val}_{\mathcal{G}}(\tau)$.

Observe that the classes of games with positional strategies guarantee a positional optimal counter strategy for Player Min to all strategies $\sigma \in \Sigma$ of Player Max. We denote these strategies by τ^c_σ. Similarly, we denote the positional optimal counter strategy for Player Max to a strategy $\tau \in T$ by σ^c_τ of Player Min. While this counter strategy is not necessarily unique, we use the notational convention in all proofs that τ^c_σ is always the same counter strategy for $\sigma \in \Sigma$, and σ^c_τ is always the same counter strategy for $\tau \in T$.

Example 1. Consider the parity game arena shown in the Figure 1. We use circles for the vertices of Player Max and squares for Player Min. We label each vertex with its colour. Notice that a positional strategy can be depicted just by specifying an outgoing edge for all the vertices of a player. The positional strategies σ of Player Max is depicted in blue and the positional strategy τ of Player Min is depicted in red.

Classic Strategy Improvement Algorithm. For a strategy σ, an edge $(v, v') \in E$ with $v \in V_{\text{Max}}$ is a *profitable update* if $\sigma' \in \Sigma$ with $\sigma' : v \mapsto v'$ and $\sigma' : v'' \mapsto \sigma(v'')$ for all $v'' \neq v$ has a strictly greater evaluation than σ, i.e. $\mathsf{val}_{\mathcal{G}}(\sigma') \sqsupset \mathsf{val}_{\mathcal{G}}(\sigma)$. Let $\mathsf{Prof}(\sigma)$ be the set of profitable updates.

Example 2. Consider the strategies σ from the Example 1. Notice that strategy $\tau = \tau^c_\sigma$ is the optimal counter strategy to σ, i.e. $\mathsf{val}(\sigma) = \mathsf{val}(\sigma, \tau)$. It follows that $\mathsf{Prof}(\sigma) = \{(3, 4), (3, 0)\}$, because both the successor to the left and the successor to the right have a better valuation, $(3, \{4\}, 1)$ and $(0, \emptyset, 0)$, resp., than the successor on the selected self-loop, $(3, \emptyset, 0)$.

For a strategy σ and a functional (right-unique) subset $P \subseteq \mathsf{Prof}(\sigma)$ we define the strategy σ^P with $\sigma^P : v \mapsto v'$ if $(v, v') \in P$ and $\sigma^P : v \mapsto \sigma(v)$ if there is no $v' \in V$ with $(v, v') \in P$.

Algorithm 3 provides a generic template for strategy improvement algorithms. As we discussed in the introduction, the classic strategy improvement algorithms work well for classes of games that are positionally determined and have evaluation function are such that the set $\mathsf{Prof}(\sigma)$ of profitable updates is easy to identify, and reach an optimum exactly where there are no profitable updates. We next formalise these prerequisites for a class of games to be good for strategy improvement algorithm.

Algorithm 3: Classic strategy improvement algorithm

1 Let σ_0 be an arbitrary positional strategy. **Set** $i := 0$.
2 If $\mathsf{Prof}(\sigma_i) = \emptyset$ **return** σ_i
3 $\sigma_{i+1} := {\sigma_i}^P$ for some functional subset $P \subseteq \mathsf{Prof}(\sigma_i)$ s.t. $P \neq \emptyset$ if $\mathsf{Prof}(\sigma_i) \neq \emptyset$. **Set** $i := i + 1$. **go to** 2.

For a class of graph games, profitable updates are *combinable* if, for all strategies σ and all functional (right-unique) subsets $P \subseteq \mathsf{Prof}(\sigma)$ we have that $\mathsf{val}_{\mathcal{G}}(\sigma^P) \sqsupset \mathsf{val}_{\mathcal{G}}(\sigma)$. Moreover, we say that a class of graph games is *maximum identifying* if $\mathsf{Prof}(\sigma) = \emptyset \Leftrightarrow \mathsf{val}_{\mathcal{G}}(\sigma) = \mathsf{val}_{\mathcal{G}}$. We say that a class of games is *good for* max *strategy improvement* if they are positionally determined and have combinable and maximum identifying improvements.

Theorem 1. *Algorithm 3 returns an optimal strategy* σ $(\mathsf{val}_{\mathcal{G}}(\sigma) = \mathsf{val}_{\mathcal{G}})$ *of Player Max for all games that are good for* max *strategy improvement.*

As a remark, we can drop the combinability requirement while maintaining correctness when we restrict the updates to a single position, that is, when we require P to be singleton for every update. We call such strategy improvement algorithms *slow*, and a class of games *good for slow* max *strategy improvement* if it is maximum identifying and positionally determined.

Theorem 2. *Slow variants of Algorithm 3 returns an optimal strategy* σ $(\mathsf{val}_{\mathcal{G}}(\sigma) = \mathsf{val}_{\mathcal{G}})$ *of Player Max for all games that are positionally determined with maximum identifying improvement.*

Proof (of Theorems 1 and 2). The proof for both theorems is the same. The strategy improvement algorithm will produce a sequence $\sigma_0, \sigma_1, \sigma_2 \ldots$ of positional strategies with increasing quality $\mathsf{val}_{\mathcal{G}}(\sigma_0) \sqsubset \mathsf{val}_{\mathcal{G}}(\sigma_1) \sqsubset \mathsf{val}_{\mathcal{G}}(\sigma_2) \sqsubset \ldots$. As the set of positional strategies is finite, this chain must be finite. As the game is maximum identifying, the stopping condition provides optimality. □

Various concepts and results extend naturally for analogous claims about Player Min. We call a class of game *good for strategy improvement* if it is good for max strategy improvement and good for min strategy improvement. Parity games, mean payoff games, and discounted payoff games are all good for strategy improvement (for both players). Moreover, the calculation of $\mathsf{Prof}(\sigma)$ is cheap in all of these instances, which makes them well suited for strategy improvement.

3 Symmetric Strategy Improvement Algorithm

We first extend the termination argument for classic strategy improvement techniques (Theorems 1 and 2) to symmetric strategy improvement given as Algorithm 4. In this section, we show the correctness of Algorithm 4.

Algorithm 4: Symmetric strategy improvement algorithm

1 Let σ_0 and τ_0 be arbitrary positional strategies. **set** $i := 0$.
2 Determine $\sigma^c_{\tau_i}$ and $\tau^c_{\sigma_i}$
3 $\sigma'_i := {\sigma_i}^P$ for $P \subseteq \mathsf{Prof}(\sigma_i) \cap \sigma^c_{\tau_i}$, s.t. $P \neq \emptyset$ if $\mathsf{Prof}(\sigma_i) \cap \sigma^c_{\tau_i} \neq \emptyset$.
4 $\tau'_i := {\tau_i}^P$ for $P \subseteq \mathsf{Prof}(\tau_i) \cap \tau^c_{\sigma_i}$, s.t. $P \neq \emptyset$ if $\mathsf{Prof}(\tau_i) \cap \tau^c_{\sigma_i} \neq \emptyset$.
5 **if** $\sigma'_i = \sigma_i$ and $\tau'_i = \tau_i$ **return** (σ_i, τ_i).
6 **set** $\sigma_{i+1} = \sigma'_i$; $\tau_{i+1} = \tau'_i$; $i := i + 1$. **go to** 2.

Lemma 1. *The symmetric strategy improvement algorithm terminates for all classes of games that are good for strategy improvement.*

Proof. We first observe that the algorithm yields a sequence $\sigma_0, \sigma_1, \sigma_2, \ldots$ of Player Max strategies for $\mathcal{G}$ with improving values $\mathsf{val}_\mathcal{G}(\sigma_0) \sqsubseteq \mathsf{val}_\mathcal{G}(\sigma_1) \sqsubseteq \mathsf{val}_\mathcal{G}(\sigma_2) \sqsubseteq \ldots$, where equality, $\mathsf{val}_\mathcal{G}(\sigma_i) \equiv \mathsf{val}_\mathcal{G}(\sigma_{i+i})$, implies $\sigma_i = \sigma_{i+1}$. Similarly, for the sequence $\tau_0, \tau_1, \tau_2, \ldots$ of Player Min strategies for $\mathcal{G}$, the values $\mathsf{val}_\mathcal{G}(\tau_0) \sqsupseteq \mathsf{val}_\mathcal{G}(\tau_1) \sqsupseteq \mathsf{val}_\mathcal{G}(\tau_2) \sqsupseteq \ldots$, improve (for Player Min), such that equality, $\mathsf{val}_\mathcal{G}(\tau_i) \equiv \mathsf{val}_\mathcal{G}(\tau_{i+i})$, implies $\tau_i = \tau_{i+1}$. As the number of values that can be taken is finite, eventually both values stabilise and the algorithm terminates. □

What remains to be shown is that the symmetric strategy improvement algorithm cannot terminate with an incorrect result. In order to show this, we first prove the weaker claim that it is optimal in $\mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma) = (V_{\max}, V_{\min}, E', \mathsf{val})$ such that $E' = \big\{\big(v, \sigma(v)\big) \mid v \in V_{\max}\big\} \cup \big\{\big(v, \tau(v)\big) \mid v \in V_{\min}\big\} \cup \big\{\big(v, \sigma^c_\tau(v)\big) \mid v \in V_{\max}\big\} \cup \big\{\big(v, \tau^c_\sigma(v)\big) \mid v \in V_{\min}\big\}$ is the subgame of $\mathcal{G}$ whose edges are those defined by the four positional strategies, when it terminates with the pair σ, τ.

Lemma 2. *When the symmetric strategy improvement algorithm terminates with the strategy pair σ, τ on games that are good for strategy improvement, then σ and τ are the optimal strategies for the respective players in $\mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma)$.*

Proof. For $\mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma)$, both update steps are not restricted: the changes Player Max can potentially select his updates from are the edges defined by σ^c_τ at the vertices $v \in V_{\max}$ where σ and σ^c_τ differ ($\sigma(v) \neq \sigma^c_\tau(v)$). Consequently, $\mathsf{Prof}(\sigma) = \mathsf{Prof}(\sigma) \cap \sigma^c_\tau$. Thus, $\sigma = \sigma'$ holds if, and only if, σ is the result of an update step when using classic strategy improvement in $\mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma)$ when starting in σ. As game is maximum identifying, σ is the optimal Player Max strategy for $\mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma)$. Likewise, the Player Min can potentially select every updates from τ^c_σ at vertices $v \in V_{\min}$, and we first get $\mathsf{Prof}(\tau) = \mathsf{Prof}(\tau) \cap \tau^c_\sigma$ with the same argument. As the game is minimum identifying, τ is the optimal Player Min strategy for $\mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma)$. □

We can now expand the optimality in the subgame $\mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma)$ from Lemma 2 to global optimality the valuation of these strategies for $\mathcal{G}$.

Lemma 3. *When the symmetric strategy improvement algorithm terminates with the strategy pair σ, τ on a game $\mathcal{G}$ that is good for strategy improvement, then σ is an optimal Player Max strategy and τ an optimal Player Min strategy.*

Proof. Let σ, τ be the strategies returned by the symmetric strategy improvement algorithm for a game $\mathcal{G}$, and let $\mathcal{L} = \mathcal{G}(\sigma, \tau, \sigma^c_\tau, \tau^c_\sigma)$ denote the local game from Lemma 2 defined by them. Lemma 2 has established optimality in $\mathcal{L}$. Observing that the optimal responses in $\mathcal{G}$ to σ and τ, τ^c_σ and σ^c_τ, respectively, are available in $\mathcal{L}$, we first see that they are also optimal in $\mathcal{L}$. Thus, we have

- $\mathsf{val}_{\mathcal{L}}(\sigma) \equiv \mathsf{val}_{\mathcal{L}}(\sigma, \tau^c_\sigma) \equiv \mathsf{val}_{\mathcal{G}}(\sigma, \tau^c_\sigma)$ and
- $\mathsf{val}_{\mathcal{L}}(\tau) \equiv \mathsf{val}_{\mathcal{L}}(\sigma^c_\tau, \tau) \equiv \mathsf{val}_{\mathcal{G}}(\sigma^c_\tau, \tau)$.

Optimality in $\mathcal{L}$ then provides $\mathsf{val}_{\mathcal{L}}(\sigma) = \mathsf{val}_{\mathcal{L}}(\tau)$. Putting these three equations together, we get $\mathsf{val}_{\mathcal{G}}(\sigma, \tau^c_\sigma) \equiv \mathsf{val}_{\mathcal{G}}(\sigma^c_\tau, \tau)$.

Taking into account that τ^c_σ and σ^c_τ are the optimal responses to σ and τ, respectively, in $\mathcal{G}$, we expand this to $\mathsf{val}_{\mathcal{G}} \sqsupseteq \mathsf{val}_{\mathcal{G}}(\sigma) \equiv \mathsf{val}_{\mathcal{G}}(\sigma, \tau^c_\sigma) \equiv \mathsf{val}_{\mathcal{G}}(\sigma^c_\tau, \tau) \equiv \mathsf{val}_{\mathcal{G}}(\tau) \sqsupseteq \mathsf{val}_{\mathcal{G}}$ and get $\mathsf{val}_{\mathcal{G}} \equiv \mathsf{val}_{\mathcal{G}}(\sigma) \equiv \mathsf{val}_{\mathcal{G}}(\tau) \equiv \mathsf{val}_{\mathcal{G}}(\sigma, \tau)$. □

The lemmas in this subsection yield the following results.

Theorem 3. *The symmetric strategy improvement algorithm is correct for games that are good for strategy improvement.*

Theorem 4. *The slow symmetric strategy improvement algorithm is correct for positionally determined games that are maximum and minimum identifying.*

We implemented our symmetric strategy improvement algorithm based on the progress measures introduced by Vöge and Jurdziński [31]. The first step is to determine the valuation for the optimal counter strategies to and the valuations for σ and τ.

Example 3. In our running example from Figure 1, we have discussed in the previous section that τ is the optimal counter strategy τ^c_σ and that $\mathsf{Prof}(\sigma) = \{(3,4),(3,0)\}$. In the optimal counter strategy σ^c_τ to τ, Player Max moves from 3 to 4, and we get $\mathsf{val}(1,\tau) = (1,\emptyset,0)$, $\mathsf{val}(4,\tau) = (4,\emptyset,0)$, $\mathsf{val}(3,\tau) = (4,\emptyset,1)$, and $\mathsf{val}(0,\tau) = (0,\emptyset,0)$. Consequently, $\mathsf{Prof}(\tau) = \{(4,1)\}$. For the update of σ, we select the intersection of $\mathsf{Prof}(\sigma)$ and σ^c_τ. In our example, this is the edge from 3 to 4 (depicted in green). To update τ, we select the intersection of $\mathsf{Prof}(\tau)$ and τ^c_σ. In our example, this intersection is empty, as the current strategy τ agrees with τ^c_σ.

A minor improvement on stopping criteria. We look at a minor but natural improvement over Algorithm 4. In Algorithm 4, we use termination on both sides as a condition to terminate the algorithm. We could alternatively check if *either* player has reached an optimum. Once this is the case, we can return the optimal strategy and an optimal counter strategy to it.

The correctness of this stopping condition is provided by Theorems 1 and 2. Checking this stopping condition is cheap: it suffices to check if $\mathsf{Prof}(\sigma_i)$ is empty—and to return $(\sigma_i, \tau^c_{\sigma_i})$ in this case—and to check $\mathsf{Prof}(\tau_i)$ is empty—and then return $(\sigma^c_{\tau_i}, \tau_i)$.

Theorem 5. *The difference in the number of iterations of Algorithm 4 and the improved algorithm is at most linear in the number of states of $\mathcal{G}$.*

This holds because one simply converges against the optimal strategy: every change replcase a decision that deviates from it by a decision that concurs with it. While the improvement is minor, it implies that, for single player games, the number of updates required is at most linear in the number of states of $\mathcal{G}$. Consequently, exponential lower bounds for one player cases, e.g., for MDPs [10], do not apply for symmetric strategy improvement.

Friedmann's traps. A thorough discussion on how symmetric strategy improvement defies Friedmann's traps is provided in [29]. Broadly speaking, Friedmann's traps have two main ingredients. The most important one is a binary counter, which is counted up (hence providing an exponential bound) and a deceleration lane (a technical trick to orchestrate the timely counting).

This structure of Friedmann's traps has proven to be quite resistant against different update strategies. The traps for different update strategies differ mainly in the details of how the counter is incremented.

As we discuss in [29], symmetric strategy improvement defies the central counting strategy by setting the bits successively, starting with the most significant bit of the counter. Additionally, it also causes a malfunction of the deceleration lane. The deceleration lane ceases to work because, for all strategies of the opponent player, the optimal counter strategy is to take the same (the longest) path through the deceleration lane. Thus, the strategy for the deceleration lane will quickly converge to taking this path, and will never be reset in parts or full.

4 Experimental Results

We have implemented the symmetric strategy improvement algorithm for parity games and compared it with the standard strategy improvement algorithm with the popular locally optimising and other switching rules. To generate various examples we used the tools `steadygame` and `stratimprgen` that comes as a part of the parity game solver collection PGSolver [19]. We have compared the performance of our algorithm on parity games with 100 positions (see [29]) and found that the locally optimising policy outperforms other switching rules. We therefore compare our symmetric strategy improvement algorithm with the locally optimising strategy improvement below.

Since every iteration of both algorithms is rather similar—one iteration of our symmetric strategy improvement algorithm essentially runs two copies of an iteration of a classical strategy improvement algorithm—and tractable, the key data to compare these algorithms is the number of iterations taken.

Symmetric strategy improvement will often rule out improvements at individual positions: it disregards profitable changes of Player Max and Min if they do not comply with σ^c_τ and τ^c_σ, respectively. It is well known that considering fewer updates can lead to a significant increase in the number of updates on

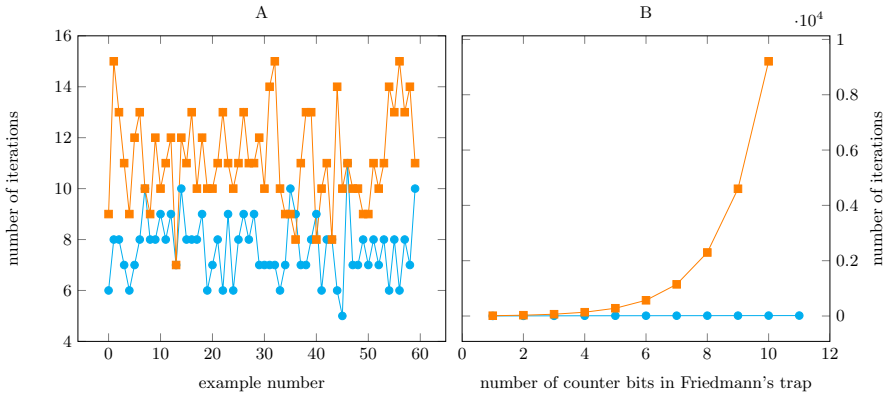

Fig. 2. The plots compare the performance of symmetric strategy improvement (data points in cyan circles) with classic strategy improvement using the locally optimising policy rule (data points in orange squares). The left plot refers to random examples generated using the `steadygame 1000 2 4 3 5 6` command. The right plot refers to Friedmann's trap [12] generated by the command `stratimprgen -pg switchallsubexp i`.

random examples and benchmarks. An algorithm based on the random-facet method [20, 3], e.g., needs around a hundred iterations on the random examples with 100 positions we have drawn, simply because it updates only a single position at a time. The same holds for a random-edge policy where only a single position is updated. The figures for these two methods are given in [29].

It is therefore good news that symmetric strategy improvement does not display a similar weakness. It even uses less updates when compared to classic strategy improvement with the popular locally optimising and locally random policy rules. Note also that having less updates can lead to a faster evaluation of the update, because unchanged parts do not need to be re-evaluated [3].

Switch Rule	1	2	3	4	5	6	7	8	9	10
Cunningham	2	6	9	12	15	18	21	24	27	30
CunninghamSubexp	1	1	1	1	1	1	1	1	1	1
FearnleySubexp	4	7	11	13	17	21	25	29	33	37
FriedmannSubexp	4	9	13	15	19	23	27	31	35	39
RandomEdgeExpTest	1	2	2	2	2	2	2	2	2	2
RandomFacetSubexp	1	2	7	9	11	13	15	17	19	21
SwitchAllBestExp	4	5	8	11	12	13	15	17	18	19
SwitchAllBestSubExp	5	7	9	11	13	15	17	19	21	23
SwitchAllSubExp	3	5	7	9	10	11	12	13	14	15
SwitchAllExp	3	4	6	8	10	11	12	14	16	18
ZadehExp	-	6	10	14	18	21	25	28	32	35
ZadehSubexp	5	9	13	16	20	23	27	30	34	37

As shown in Figure 2, the symmetric strategy improvement algorithm performs better (on average) in comparison with the traditional strategy improvement algorithm with the locally optimising policy rule. It also avoids Friedmann's traps for the strategy improvement algorithm: the table above shows the performance of the symmetric strategy improvement algorithm for Friedmann's traps for other common switching rules. It is clear that our algorithm is not exponential for these classes of examples.

5 Discussion

We have introduced symmetric approaches to strategy improvement, where the players take inspiration from the respective other's strategy when improving theirs. This creates a rather moderate overhead, where each step is at most twice as expensive as a normal improvement step. For this moderate price, we have shown that we can break the traps Friedmann has introduced to establish exponential bounds for the different update policies in classic strategy improvement [12–14].

In hindsight, attacking a symmetric problem with a symmetric approach seems so natural, that it is quite surprising that it has not been attempted immediately. There are, however, good reasons for this, but one should also consent that the claim is not entirely true: the concurrent update to the respective optimal counter strategy has been considered quite early [12–14], but was dismissed, because it can lead to cycles [5].

The first reason is therefore that it was folklore that symmetric strategy improvement does not work. The second reason is that the argument for the techniques that we have developed in this paper would have been restricted to beauty until some of the appeal of classic strategy improvement was caught in Friedmann's traps. Friedmann himself, however, remained optimistic:

> We think that the strategy iteration still is a promising candidate for a polynomial time algorithm, however it may be necessary to alter more of it than just the improvement policy.

This is precisely, what the introduction of symmetry and co-improvement tries to do.

References

1. R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.
2. D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer. Dag-width and parity games. In *Proc. of STACS*, pages 524–436. Springer-Verlag, 2006.
3. H. Björklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Appl. Math.*, 155(2):210–229, 2007.
4. A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *TCS*, 178(1–2):237–255, 1997.
5. A. Condon. On algorithms for simple stochastic games. In *Advances in Computational Complexity Theory*, pages 51–73. American Mathematical Society, 1993.
6. L. de Alfaro, T. A. Henzinger, and R. Majumdar. From verification to control: Dynamic programs for omega-regular objectives. In *Proc. of LICS*, pages 279–290, 2001.
7. E. A. Emerson and C. S. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. of FOCS*, pages 368–377. IEEE Computer Society Press, October 1991.
8. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In *Proc. of CAV*, pages 385–396, 1993.

9. E. A. Emerson and C. Lei. Efcient model checking in fragments of the propositional μ-calculus. In *Proc. of LICS*, pages 267–278. IEEE Computer Society Press, 1986.
10. J. Fearnley. Exponential lower bounds for policy iteration. In *Proc. of ICALP*, volume LNCS 6199, pages 551–562, 2010.
11. J. Fearnley. Non-oblivious strategy improvement. In *Proc. of LPAR*, pages 212–230, 2010.
12. O. Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. *LMCS*, 7(3), 2011.
13. O. Friedmann. A subexponential lower bound for zadeh's pivoting rule for solving linear programs and games. In *Proc. of IPCO*, LNCS, pages 192–206, 2011.
14. O. Friedmann. A superpolynomial lower bound for strategy iteration based on snare memorization. *Discrete Applied Mathematics*, 161(10-11):1317–1337, 2013.
15. M. Jurdziński. Small progress measures for solving parity games. In *Proc. of STACS*, pages 290–301. Springer-Verlag, 2000.
16. M. Jurdzinski, M. Paterson, and U. Zwick. A deterministic subexponential algorithm for solving parity games. *SIAM J. Comput.*, 38(4):1519–1532, 2008.
17. D. Kozen. Results on the propositional μ-calculus. *TCS*, 27:333–354, 1983.
18. M. Lange. Solving parity games by a reduction to SAT. In *Proc. of Int. Workshop on Games in Design and Verification*, 2005.
19. M. Lange and O. Friedmann. The PGSolver collection of parity game solvers. Technical report, Institut für Informatik Ludwig-Maximilians-Universität, 2010.
20. W. Ludwig. A subexponential randomized algorithm for the simple stochastic game problem. *Inf. Comput.*, 117(1):151–155, 1995.
21. R. McNaughton. Infinite games played on finite graphs. *Ann. Pure Appl. Logic*, 65(2):149–184, 1993.
22. J. Obdržálek. Fast mu-calculus model checking when tree-width is bounded. In *Proc. of CAV*, pages 80–92. Springer-Verlag, 2003.
23. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. In *Proc. of LICS*, pages 255–264. IEEE Computer Society, 2006.
24. A. Puri. *Theory of hybrid systems and discrete event systems.* PhD thesis, Computer Science Department, University of California, Berkeley, 1995.
25. S. Schewe. Solving parity games in big steps. In *Proc. of FSTTCS*, pages 449–460, 2007.
26. S. Schewe. An optimal strategy improvement algorithm for solving parity and payoff games. In *Proc. of CSL 2008*, pages 368–383. Springer-Verlag, 2008.
27. S. Schewe and B. Finkbeiner. Satisfiability and finite model property for the alternating-time *mu*-calculus. In *Proc. of CSL*, pages 591–605, 2006.
28. S. Schewe and B. Finkbeiner. Synthesis of asynchronous systems. In *Proc. of LOPSTR*, pages 127–142. Springer-Verlag, 2006.
29. S. Schewe, A. Trivedi, and T. Varghese. Symmetric strategy improvement. *CoRR*, abs/1501.06484, 2015.
30. M. Y. Vardi. Reasoning about the past with two-way automata. In *Proc. of ICALP*, pages 628–641. Springer-Verlag, 1998.
31. J. Vöge and M. Jurdziński. A discrete strategy improvement algorithm for solving parity games. In *Proceedings of the CAV*, pages 202–215. Springer-Verlag, 2000.
32. T. Wilke. Alternating tree automata, parity games, and modal μ-calculus. *Bull. Soc. Math. Belg.*, 8(2), May 2001.
33. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998.
34. U. Zwick and M. S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2):343–359, 1996.