Time-optimal Winning Strategies in Infinite Games

Martin Zimmermann
RWTH Aachen University
zimmermann@automata.rwth-aachen.de

Gasics meeting
Brussels, March 5-6, 2009
Two-player games of infinite duration on graphs

- Solution to the *synthesis problem* for reactive systems.
- Well-developed theory with nice results.
- Classical quality measure: *memory size of a winning strategy*.
Two-player games of infinite duration on graphs

- Solution to the *synthesis problem* for reactive systems.
- Well-developed theory with nice results.
- Classical quality measure: *memory size of a winning strategy*.

**But:** many winning conditions allow other quality measures.

- “From qualitative to quantitative games.”
- “Optimal controller synthesis.”
1. Definitions & Related Work
An \textit{(initialized) Arena} \( G = (V, V_0, V_1, E, s_0) \) consists of

\begin{itemize}
  \item a finite directed graph \((V, E)\),
  \item a partition \(\{V_0, V_1\}\) of \(V\) denoting the positions of Player 0 and 1,
  \item an \textit{initial vertex} \(s_0 \in V\).
\end{itemize}

A \textit{play} \(\rho_0 \rho_1 \rho_2 \ldots\) in \(G\) is an infinite path starting in \(s_0\).

A \textit{strategy} for Player \(i\) is a \textit{(partial)} mapping \(\sigma : V^*V_i \rightarrow V\) such that \((s, \sigma(ws)) \in E\) for all \(w \in V^*\) and all \(s \in V_i\).

\(\rho_0 \rho_1 \rho_2 \ldots\) is consistent with \(\sigma\) if \(\rho_{n+1} = \sigma(\rho_0 \ldots \rho_n)\) for all \(\rho_n \in V_i\).

Martin Zimmermann RWTH Aachen

Time-optimal Winning Strategies in Infinite Games
The outcome of a play can be

- *qualitative*: win or lose
  - one player wins a play, the other loses it.
  - Büchi, Co-Büchi, Rabin, Streett, Parity, Muller,...
- $\sigma$ winning strategy for Player $i$: every play that is consistent with $\sigma$ is won by Player $i$. 
The outcome of a play can be

- **qualitative**: win or lose
  - one player wins a play, the other loses it.
  - Büchi, Co-Büchi, Rabin, Streett, Parity, Muller,…
  - $\sigma$ winning strategy for Player $i$: every play that is consistent with $\sigma$ is won by Player $i$.

- **quantitative**: a payoff for each player
  - each player tries to maximize her payoff.
  - Mean-Payoff, Discounted Payoff,…
  - Value of $\sigma$: payoff of the worst play consistent with $\sigma$. 
Optimal Strategies

Idea:

- The outcome of a play is still binary: win or lose.
- But the quality of the (winning) plays and strategies is measured:
  - determine optimal (w.r.t. given quality measure) winning strategies for Player 0.
An Example

Request-Response Game $\mathcal{G} = (G, (Q_j, P_j)_{j=1,\ldots,k})$ where $Q_j, P_j \subseteq V$.

- Player 0 wins a play if every visit to a $Q_j$ vertex is *responded* by a later visit to $P_j$.

- Waiting times: start a clock for every request that is stopped as soon as it is responded (and ignore subsequent requests).

- Accumulated waiting time: sum up the clock values up to that position (quadratic influence).

- Value of a play: limit superior of the average accumulated waiting time; corresponding notion of *optimal* strategies.
**Theorem:** (Horn, Thomas, Wallmeier)

If Player 0 has a winning strategy for an RR Game, then she also has an optimal winning strategy, which is finite-state and effectively computable.
Other Winning Conditions

Many other winning conditions have a natural notion of waiting times.

- Reachability Games: the number of steps to the target vertices.
- Büchi Games: the periods between visits of the target vertices.
- Co-Büchi Games: the number of steps until the target vertices are reached for good.
- Parity Games: the periods between visits of vertices colored with a maximal even color (which can be optimized as well).

Some classical algorithms compute optimal winning strategies.
2. Poset Games
Motivation
Motivation

- green_0, green_1
- raise_0, raise_1
- crossing free
- train go
- lower_0, lower_1
- red_0, red_1
Request: still a singular event.
Response: partially ordered set of events.
Definition

Poset Game $\mathcal{G} = (G, (q_j, \mathcal{P}_j)_{j=1,\ldots,k}), \ P$ set of atomic propositions

- $G$ arena (labeled with $l_G : V \rightarrow 2^P$)
- $q_j \in P$ request
- $\mathcal{P}_j = (D_j, \preceq_j)$ labeled poset where $D_j \subseteq P$
**Definition cont’d**

*Embedding* of $\mathcal{P}_j$ in $\rho_0 \rho_1 \rho_2 \ldots$: function $f : D_j \rightarrow \mathbb{N}$ such that

- $d \in l_G(\rho_{f(d)})$ for all $d \in D_j$
- $d \preceq_{j} d'$ implies $f(d) \leq f(d')$ for all $d, d' \in D_j$

Player 0 wins $\rho_0 \rho_1 \rho_2 \ldots$ if

$$\forall j \forall n \left( q_j \in l_G(\rho_n) \rightarrow \rho_n \rho_{n+1} \ldots \text{ allows embedding of } \mathcal{P}_j \right)$$

“Every request $q_j$ is responded by a later embedding of $\mathcal{P}_j$ in $\rho$. “
An Example

\[
\begin{array}{c}
\text{train go} \\
\downarrow \\
\text{lower}_0 & \text{lower}_1 \\
\uparrow & \uparrow \\
\text{red}_0 & \text{red}_1 \\
\end{array}
\]
An Example

```
train go

lower_0  lower_1

red_0  red_1

{req}  {red_0}
```
An Example

\[ \text{train go} \]

\[ \text{lower}_0 \quad \text{lower}_1 \]

\[ \text{red}_0 \quad \text{red}_1 \]

\[ \{\text{req}\} \quad \{\text{red}_0\} \quad \{\text{lower}_0\} \]
An Example

\[
\text{train go} \quad \rightarrow \\
\text{lower}_0 \quad \rightarrow \\
\text{red}_0 \\
\text{lower}_1 \\
\text{red}_1
\]

\[
\begin{align*}
\{\text{req}\} & \quad \{\text{red}_0\} & \quad \{\text{lower}_0\} & \quad \{\text{red}_1\}
\end{align*}
\]
An Example

train go

lower_0

red_0

red_1

lower_1

{req}  {red_0}  {lower_0}  {red_1}  {lower_1}
An Example

\begin{figure}
\centering
\begin{tikzpicture}
    \node (train) at (0,0) {train go};
    \node (lower_0) at (-2,-2) {lower_0};
    \node (red_0) at (-4,-4) {red_0};
    \node (lower_1) at (2,-2) {lower_1};
    \node (red_1) at (4,-4) {red_1};
    \node (req) at (0,-6) \{req\};
    \node (red_0) at (0,-8) \{red_0\};
    \node (lower_0) at (0,-10) \{lower_0\};
    \node (red_1) at (0,-12) \{red_1\};
    \node (lower_1) at (0,-14) \{lower_1\};
    \node (train) at (0,-16) \{train go\};

    \draw[->, dashed] (train) to (lower_0);
    \draw[->, dashed] (lower_0) to (red_0);
    \draw[->, dashed] (red_0) to (req);
    \draw[->, dashed] (lower_0) to (lower_1);
    \draw[->, dashed] (lower_1) to (red_1);
    \draw[->, dashed] (red_1) to (train);
\end{tikzpicture}
\end{figure}
Overlapping Embeddings

\[
\text{train go} \\
\text{lower}_0 \\
\text{red}_0
\]
Overlapping Embeddings

\[
\text{req} \quad \{\text{red}_0\} \quad \text{req} \quad \{\text{red}_0\} \quad \{\text{lower}_0\} \quad \{\text{train go}\}
\]
Theorem:
Poset Games are reducible to Büchi Games.
Theorem:

Poset Games are reducible to Büchi Games.

Proof:

Use memory to

- store elements of the posets that still have to be embedded,
- deal with overlapping embeddings, and
- implement a cyclic counter to ensure that every request is responded by an embedding.
3. Time-optimal Winning Strategies for Poset Games
As desired, there is a natural definition of *waiting times*

- Start a clock if a request is encountered...
- ... that is stopped as soon as the embedding is completed.
Waiting Times

As desired, there is a natural definition of *waiting times*

- Start a clock if a request is encountered...
- … that is stopped as soon as the embedding is completed.
- Need a clock for every revisit of a request (while the request is already active).
Waiting Times

As desired, there is a natural definition of *waiting times*

- Start a clock if a request is encountered...
- ... that is stopped as soon as the embedding is completed.
- Need a clock for every revisit of a request (while the request is already active).
- The value of a play is the limit superior of the average accumulated waiting time.
- The value of a strategy is the value of the worst play consistent with that strategy; corresponding notion of *optimal* strategies.
**Theorem:**

If Player 0 has a winning strategy for a Poset Game \( G \), then she also has an optimal winning strategy, which is finite-state and effectively computable.
The Main Theorem

Proof:

- If Player 0 has a winning strategy, then she also has one of value less than a certain constant (from reduction). This bounds the value of the optimal strategy, too.

- For every strategy there is another strategy of smaller or equal value, that also bounds all waiting times.

- If the waiting times are bounded, then $G$ can be reduced to a finite Mean-Payoff Game such that the values coincide.
Step 1: Bounding Waiting Times
Skip loops, but pay attention to other embeddings!
Step 1: Bounding Waiting Times

Repeating this leads to bounded waiting times.
Step 2: Reduction to Mean-Payoff Games

Mean-Payoff Game:

- edges labeled by $l : E \rightarrow \mathbb{N}$.

- goal for Player 0: maximize limit inferior of the average accumulated edge labels.

- goal for Player 1: minimize limit superior of the average accumulated edge labels.

**Theorem:** (Ehrenfeucht, Mycielski / Zwick, Paterson)
In a Mean-Payoff Game, both players have optimal strategies, which are positional and effectively computable.
Step 2: Reduction to Mean-Payoff Games

From a Poset Game $G$ with bounded waiting times, construct a Mean-Payoff Game $G'$ such that

- the memory keeps track of the waiting times, and
- the value of a play in $G$ and the payoff for Player 1 of the corresponding play in $G'$ are equal.

Then: an optimal strategy for Player 1 in $G'$ induces an optimal strategy for Player 0 in $G$.

Complexity analysis: size of the Mean-Payoff Game is super-exponential (holds already for RR Games).
4. Conclusion & Further Research
Conclusion

We have introduced a novel winning condition for Infinite Games that

- extends the Request-Response condition,
- is well-suited to model Planning Problems,
- but retains a natural definition of waiting times and optimal strategies.

We have proven the existence of optimal strategies for Poset Games, which are finite-state and effectively computable.
Conclusion

Further Research

- Avoid the detour via Mean-Payoff Games and directly compute (approximatively) optimal strategies.
- Understand the trade-off between the size and value of a strategy.
- Define and determine optimal strategies for other winning conditions.