OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

INF FAKULTÄT FÜR INFORMATIK

SwarmLab

Generalisation of Simulation-Based Search for Autonomous Gameplaying

Doktorandentag Presentation

Alexander Dockhorn

dockhorn@ovgu.de

Otto-von-Guericke University of Magdeburg

Faculty of Computer Science

Institute for Intelligent Cooperating Systems

Motivation

Computational Intelligence is about task automation, e.g.:

- developing self-driving cars

[1]

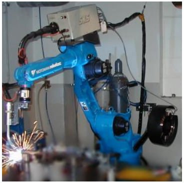
[1] https://commons.wikimedia.org/wiki/File:Tesla_Autopilot_Engaged_in_Model_X.jpg

Motivation

Computational Intelligence is about task automation, e.g.:

- developing self-driving cars
- automating a production pipeline

[1]



[2]

[1] https://commons.wikimedia.org/wiki/File:Tesla_Autopilot_Engaged_in_Model_X.jpg

Motivation

Computational Intelligence is about task automation, e.g.:

- developing self-driving cars
- automating a production pipeline

Two ways of approaching task automation:

1) manually develop controllers for each task
2) autonomously learn to solve tasks

Motivation

Computational Intelligence is about task automation, e.g.:

- developing self-driving cars
- automating a production pipeline

Two ways of approaching task automation:

1) manually develop controllers for each task
2) autonomously learn to solve tasks

Problems:

- real tasks are hard to setup and evaluate
- failure of the algorithm has considerable cost (e.g. car crash)

Motivation

Computational Intelligence is about task automation, e.g.:

- developing self-driving cars
- automating a production pipeline

Two ways of approaching task automation:

1) manually develop controllers for each task
2) autonomously learn to solve tasks

Problems:

- real tasks are hard to setup and evaluate
- failure of the algorithm has considerable cost (e.g. car crash)

Practical Solution:

- test algorithms in a simulation

Games and General Game Learning

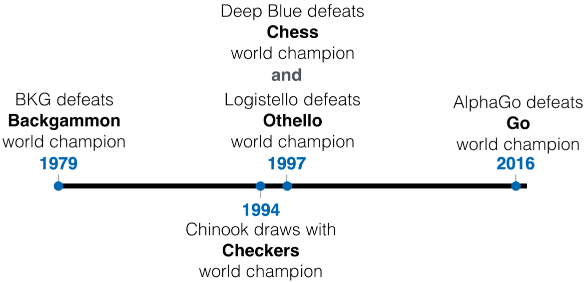
Games can be simulations of real world tasks

- quantifiable goal, varying difficulty, popular (large data sets)
- digital games are fully accessible to computers

Games and General Game Learning

Games can be simulations of real world tasks

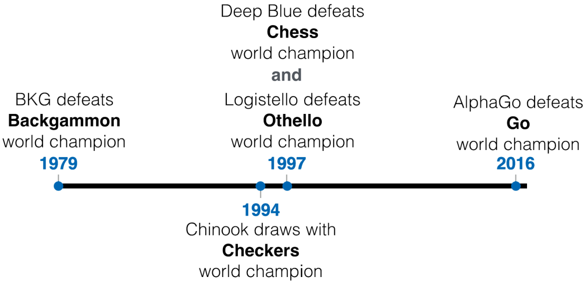
- quantifiable goal, varying difficulty, popular (large data sets)
- digital games are fully accessible to computers



Games and General Game Learning

Games can be simulations of real world tasks

- quantifiable goal, varying difficulty, popular (large data sets)
- digital games are fully accessible to computers



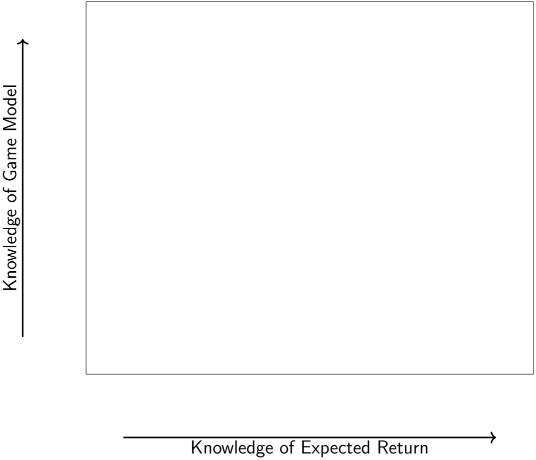
General Game Playing/Learning generalizes learning strategy across games but ignores learning the game's specific representation

Survey of Related Methods

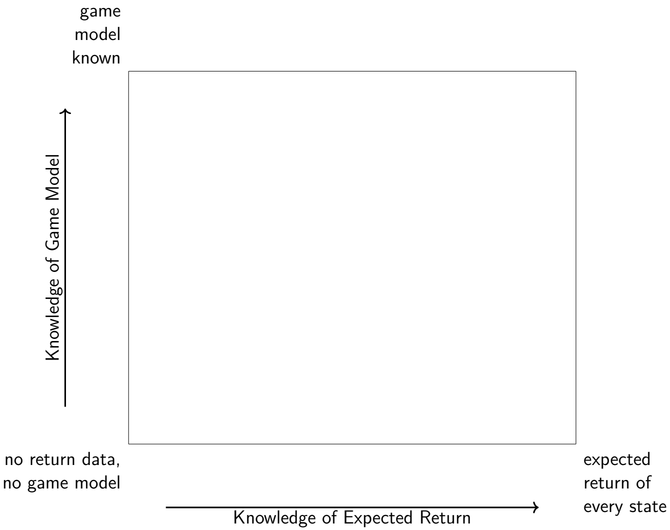
Survey of Related Methods

Knowledge of Expected Return

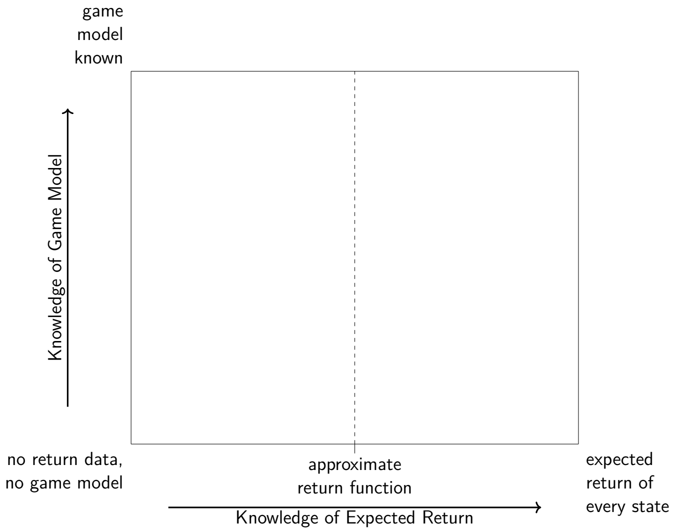
Survey of Related Methods



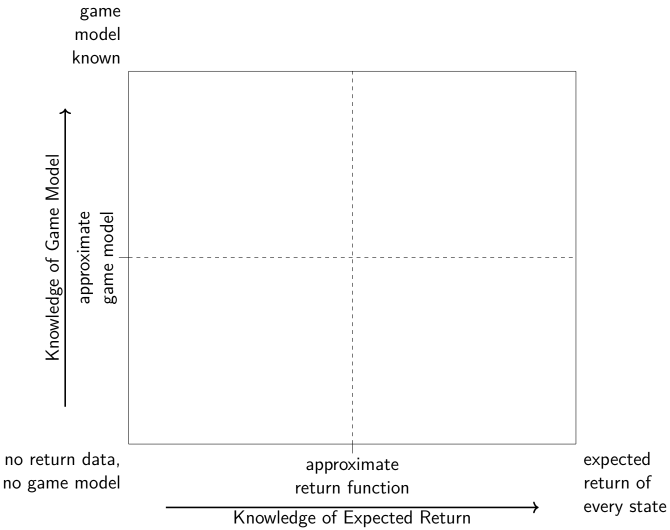
Survey of Related Methods



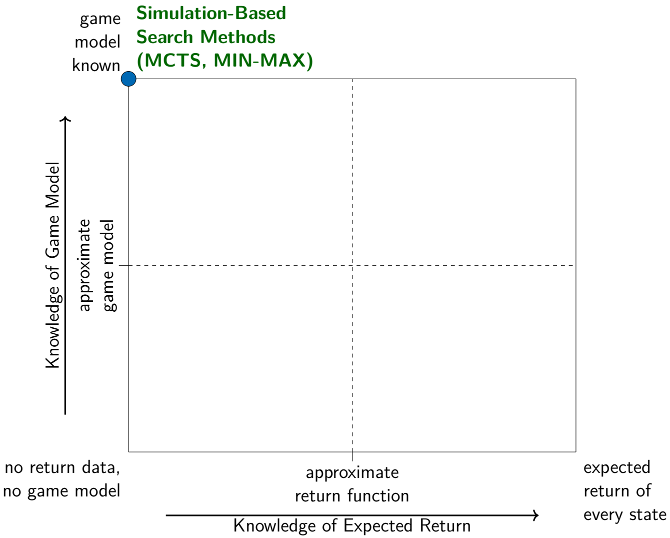
Survey of Related Methods



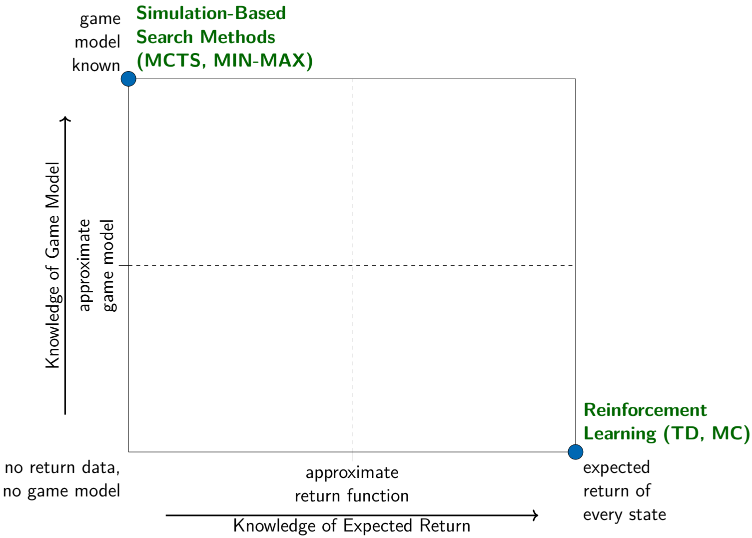
Survey of Related Methods



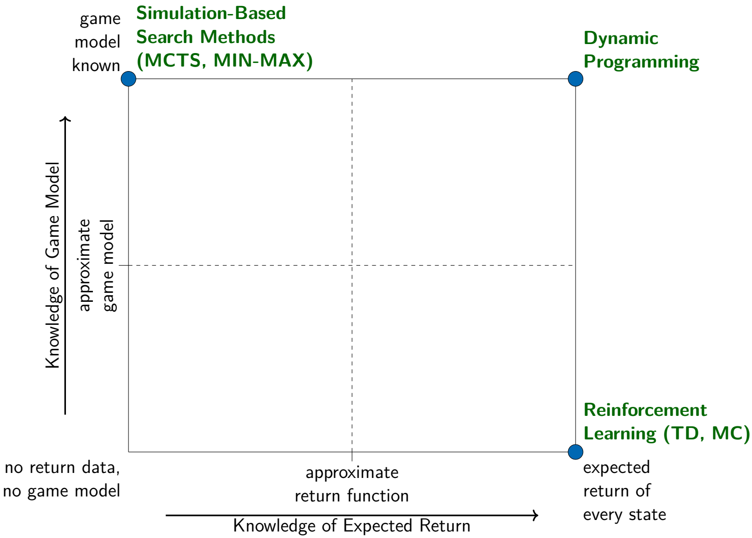
Survey of Related Methods



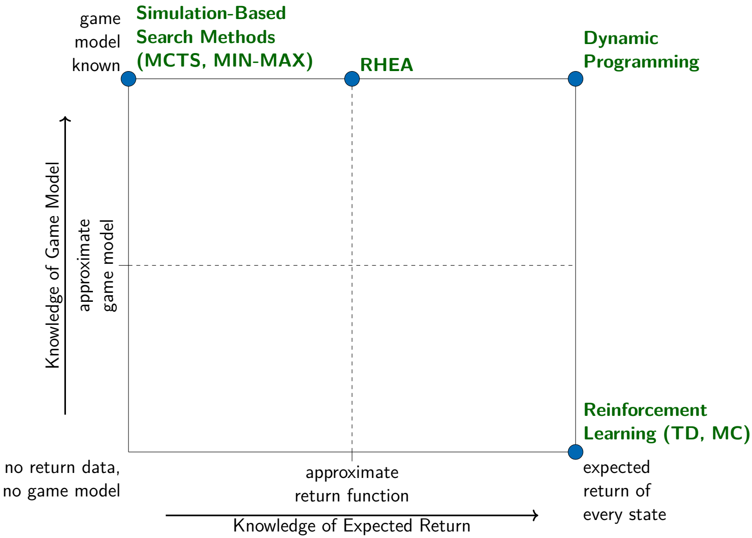
Survey of Related Methods



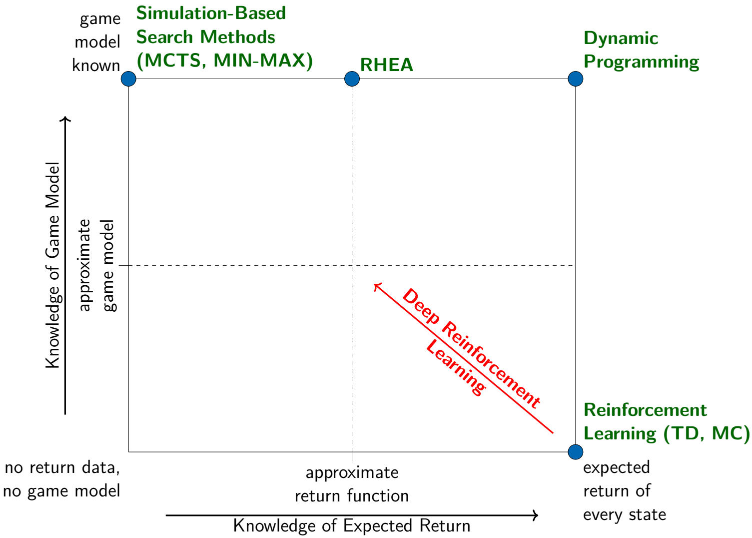
Survey of Related Methods



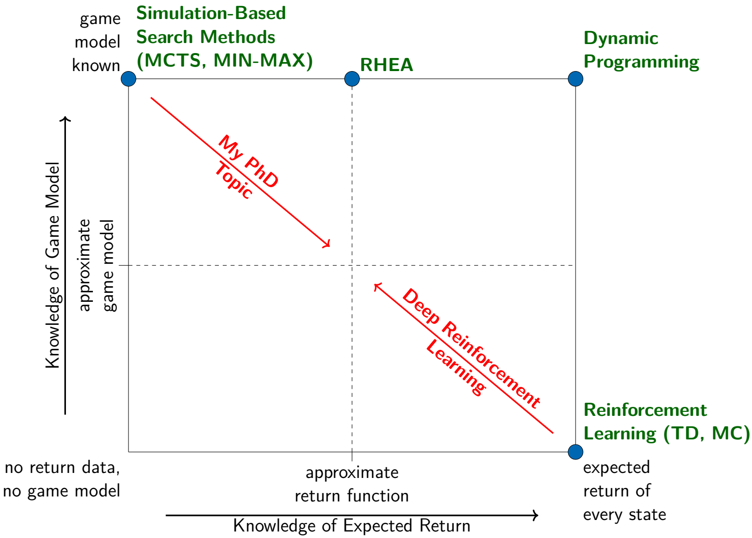
Survey of Related Methods



Survey of Related Methods



Survey of Related Methods



Simulation-Based Search Algorithms

Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms

current state

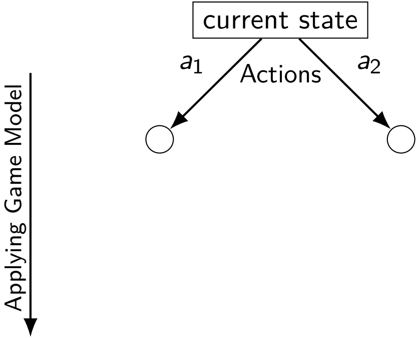
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



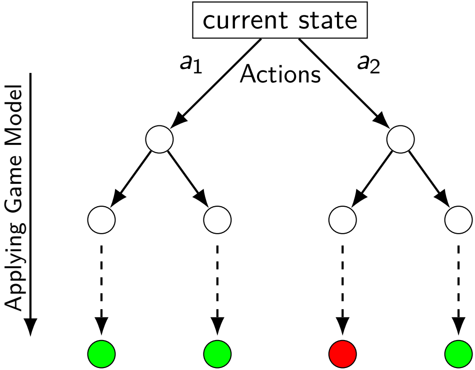
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



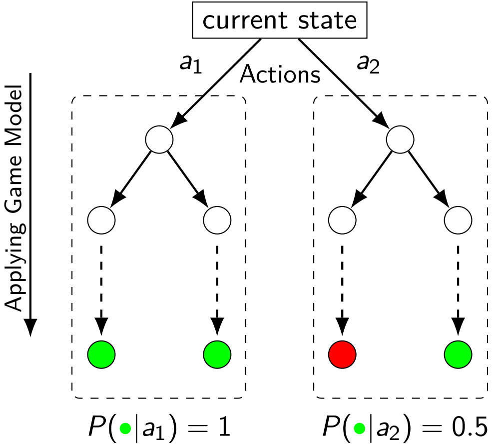
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



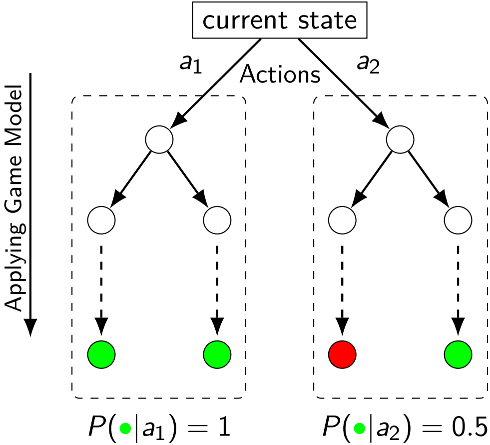
Input:

- current state
- game-model

Output:

- action with highest win-rate

Simulation-Based Search Algorithms



Input:

- current state
- game-model

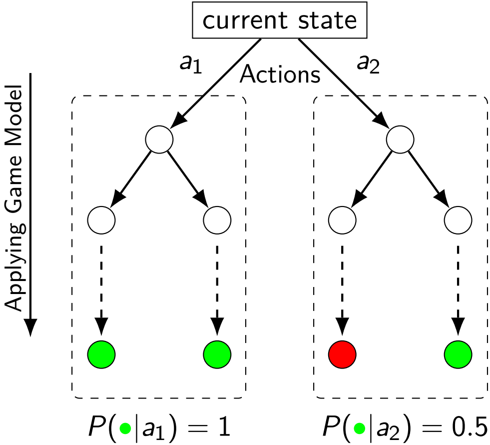
Output:

- action with highest win-rate

Advantage:

- no evaluation function needed

Simulation-Based Search Algorithms



Input:

- current state
- game-model

Output:

- action with highest win-rate

Advantage:

- no evaluation function needed

Problem:

- game model unknown

Goals of the Thesis

Identify, develop and evaluate different methods for general game learning that enable simulation-based search in scenarios without knowledge of the game's model.

Goals of the Thesis

Identify, develop and evaluate different methods for general game learning that enable simulation-based search in scenarios without knowledge of the game's model.

Prerequisites:

- What does characterize a game model?
- How can we approximate it?

Goals of the Thesis

Identify, develop and evaluate different methods for general game learning that enable simulation-based search in scenarios without knowledge of the game's model.

Prerequisites:

- What does characterize a game model?
- How can we approximate it?

Algorithm:

- How can we use this approximation in simulation-based search?
- Can we measure the risk of using this approximation?

Goals of the Thesis

Identify, develop and evaluate different methods for general game learning that enable simulation-based search in scenarios without knowledge of the game's model.

Prerequisites:

- What does characterize a game model?
- How can we approximate it?

Algorithm:

- How can we use this approximation in simulation-based search?
- Can we measure the risk of using this approximation?

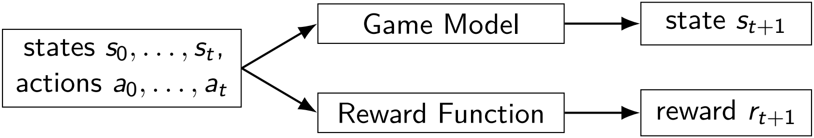
Evaluation:

- How do the developed models compare with state-of-the-art methods in terms of performance and learning speed?
- Which properties influence their applicability?

Forward Model Approximation

Developing techniques for approximating a game's model from previous interactions.

Components of a Game



state s_t perceived through multiple sensors $(s_t^{(1)}, s_t^{(2)}, \ldots, s_t^{(n)})$

- state may not be fully accessible (partial information game)

reward r_t is a performance signal

- how good do we perform in solving the task

game description as a probability distribution over possible outcomes

$$P(r_{t+1}, s_{t+1} \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t)$$

The Markov Property

Markov Property:

- the environments response at time $t+1$ only depends on the state s_t and the agent's action a_t

$$P(r_{t+1}, s_{t+1} \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = P(r_{t+1}, s_{t+1} \mid s_t, a_t)$$

The Markov Property

Markov Property:

- the environments response at time $t+1$ only depends on the state s_t and the agent's action a_t

$$P(r_{t+1}, s_{t+1} \mid s_0, a_0, s_1, a_1, \ldots, s_t, a_t) = P(r_{t+1}, s_{t+1} \mid s_t, a_t)$$

If the markov property holds, the environment dynamics can be defined by specifying:

$$P(s_{t+1} \mid s_t, a_t) \quad \text{and} \quad P(r_{t+1} \mid s_t, a_t)$$

Learning Target

Learning Target

Reinforcement Learning

predict upcoming reward signals to maximize expected reward over time

$$\mathbb{E}[G \mid s_t, a_t] \qquad G = \sum_{k=0}^{T} R_{t+k+1}$$

Learning Target

Reinforcement Learning

predict upcoming reward signals to maximize expected reward over time

$$\mathbb{E}[G \mid s_t, a_t] \qquad G = \sum_{k=0}^{T} R_{t+k+1}$$

Simulation-based search

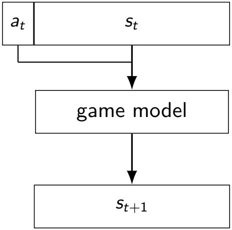
predict next game state for searching promising actions in the game tree

$$\mathbb{E}[s_{t+1} \mid s_t, a_t]$$

Analysis of the Model Space

$$P(s_{t+1} \mid s_t, a_t)$$

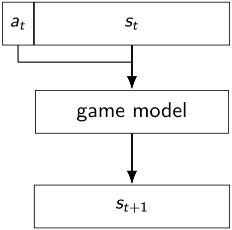
$$f : (s_t, a_t) \mapsto s_{t+1}$$

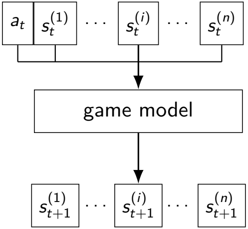


Analysis of the Model Space

$$P(s_{t+1} \mid s_t, a_t) \qquad P(s_{t+1}^{(i)} \mid s_t, a_t)$$

$$f : (s_t, a_t) \mapsto s_{t+1} \qquad f_i : (s_t, a_t) \mapsto s_{t+1}^{(i)}$$

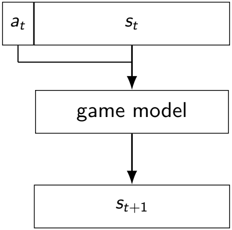


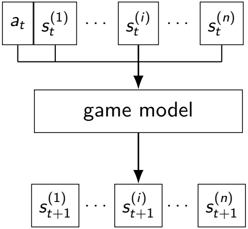


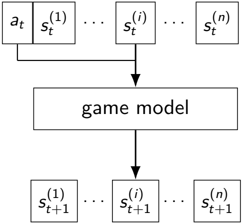
Analysis of the Model Space

$$P(s_{t+1} \mid s_t, a_t) \qquad P(s_{t+1}^{(i)} \mid s_t, a_t) \qquad P(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t)$$

$$f : (s_t, a_t) \mapsto s_{t+1} \qquad f_i : (s_t, a_t) \mapsto s_{t+1}^{(i)} \qquad f_i : (s_t^{(i)}, a_t) \mapsto s_{t+1}^{(i)}$$



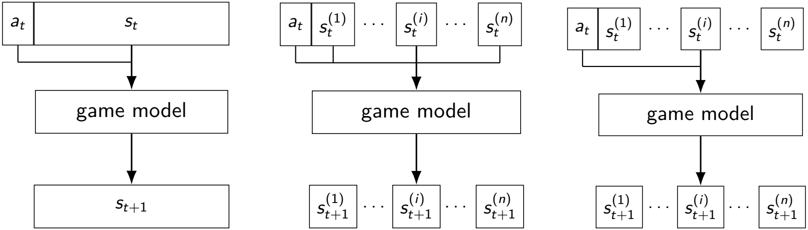




Analysis of the Model Space

$$P(s_{t+1} \mid s_t, a_t) \qquad P(s_{t+1}^{(i)} \mid s_t, a_t) \qquad P(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t)$$

$$f : (s_t, a_t) \mapsto s_{t+1} \qquad f_i : (s_t, a_t) \mapsto s_{t+1}^{(i)} \qquad f_i : (s_t^{(i)}, a_t) \mapsto s_{t+1}^{(i)}$$



Prerequisites	Game Model Characteristics	✓
	Approximation of a Game Model	?

Forward Model Approximation Implementations

Association Rule Learning[1]

- learning an understandandable ruleset of an unknown game
- special handling of termination rules

[1] Alexander Dockhorn, Chris Saxton, and Rudolf Kruse; *Association Rule Mining for Unknown Video Games*, A fuzzy dictionary of fuzzy modelling. Common concepts and perspectives (tentative title), accepted, 2019

Forward Model Approximation Implementations

Association Rule Learning[1]

- learning an understandandable ruleset of an unknown game
- special handling of termination rules

Hierarchical Knowledge Bases[2]

- rule learning through reinforcement learning
- repair mechanism for existing rules that were wrong

[1] Alexander Dockhorn, Chris Saxton, and Rudolf Kruse; *Association Rule Mining for Unknown Video Games*, A fuzzy dictionary of fuzzy modelling. Common concepts and perspectives (tentative title), accepted, 2019

[2] Alexander Dockhorn and Daan Apeldoorn; *Forward Model Approximation for General Video Game Learning*, 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2018, pp. 425-432

Forward Model Approximation Implementations

Association Rule Learning[1]

- learning an understandandable ruleset of an unknown game
- special handling of termination rules

Hierarchical Knowledge Bases[2]

- rule learning through reinforcement learning
- repair mechanism for existing rules that were wrong

Composite Model of Decision Trees[3]

- modelling sensor values individually
- speeds up model learning and increases model accuracy

---

[1] Alexander Dockhorn, Chris Saxton, and Rudolf Kruse; *Association Rule Mining for Unknown Video Games*, A fuzzy dictionary of fuzzy modelling. Common concepts and perspectives (tentative title), accepted, 2019

[2] Alexander Dockhorn and Daan Apeldoorn; *Forward Model Approximation for General Video Game Learning*, 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2018, pp. 425-432

[3] Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse; *Model Decomposition for Forward Model Approximation*, IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, November 2018, pp. 1751–1757

Evaluation of Prediction Accuracy[1]

General Video Game AI Framework:

- similar state representation for ∼100 games with 5 levels each
- object based sensor values (e.g. positions)

[1] Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse; *Model Decomposition for Forward Model Approximation*, IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, November 2018, pp. 1751–1757

Evaluation of Prediction Accuracy[1]

General Video Game AI Framework:

- similar state representation for ∼100 games with 5 levels each
- object based sensor values (e.g. positions)

Procedure:

- Play the first three levels
  - store each interaction as: $s_t, a_t \to s_{t+1}$

[1] Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse; *Model Decomposition for Forward Model Approximation*, IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, November 2018, pp. 1751–1757

Evaluation of Prediction Accuracy[1]

General Video Game AI Framework:

- similar state representation for ~100 games with 5 levels each
- object based sensor values (e.g. positions)

Procedure:

- Play the first three levels
  - store each interaction as: $s_t, a_t \rightarrow s_{t+1}$
- Generate a composite model using all previous interactions
  - train a decision tree for each sensor value

[1] Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse; *Model Decomposition for Forward Model Approximation*, IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, November 2018, pp. 1751–1757

Evaluation of Prediction Accuracy[1]

General Video Game AI Framework:

- similar state representation for ~100 games with 5 levels each
- object based sensor values (e.g. positions)

Procedure:

- Play the first three levels
  - store each interaction as: $s_t, a_t \rightarrow s_{t+1}$
- Generate a composite model using all previous interactions
  - train a decision tree for each sensor value
- Evaluate the composite model on unseen levels
  - model accuracy ranging from 60%-95%

[1] Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse; *Model Decomposition for Forward Model Approximation*, IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, November 2018, pp. 1751–1757

Evaluation of Playing Performance[1]

General Video Game AI Competition:

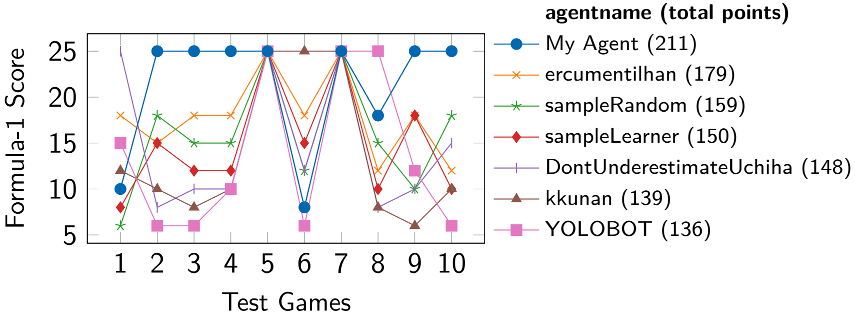
- 6 agents entered in 2017/2018
- evaluation is based on a set of 10 games
- Formula-1 scoring system

[1] Alexander Dockhorn and Daan Apeldoorn; *Forward Model Approximation for General Video Game Learning*, 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2018, pp. 425-432

Evaluation of Playing Performance[1]

General Video Game AI Competition:

- 6 agents entered in 2017/2018
- evaluation is based on a set of 10 games
- Formula-1 scoring system

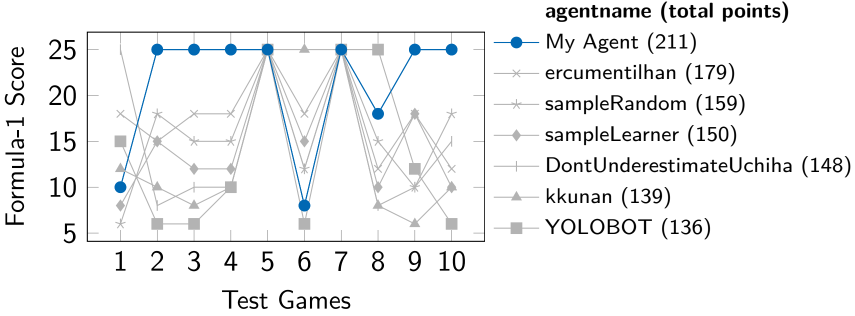


[1] Alexander Dockhorn and Daan Apeldoorn; *Forward Model Approximation for General Video Game Learning*, 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2018, pp. 425-432

Evaluation of Playing Performance[1]

General Video Game AI Competition:

- 6 agents entered in 2017/2018
- evaluation is based on a set of 10 games
- Formula-1 scoring system



[1] Alexander Dockhorn and Daan Apeldoorn; *Forward Model Approximation for General Video Game Learning*, 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2018, pp. 425-432

Summary

Prerequisites	Game Model Characteristics Approximation of a Game Model
Algorithm	Applying Model to Search Risk Measures
Evaluation	Performance Comparison Applicability Analysis

Summary

$\rightarrow$ Categorization of state-of-the-art methods

Prerequisites	Game Model Characteristics Approximation of a Game Model
Algorithm	Applying Model to Search Risk Measures
Evaluation	Performance Comparison Applicability Analysis

Summary

→ Categorization of state-of-the-art methods

→ Analysis of game model characteristics

Prerequisites	Game Model Characteristics	✓
	Approximation of a Game Model	
Algorithm	Applying Model to Search	
	Risk Measures	
Evaluation	Performance Comparison	
	Applicability Analysis	

Summary

→ Categorization of state-of-the-art methods

→ Analysis of game model characteristics

→ Game models can be learned using supervised learning

Prerequisites	Game Model Characteristics	✓
	Approximation of a Game Model	✓
Algorithm	Applying Model to Search	
	Risk Measures	
Evaluation	Performance Comparison	
	Applicability Analysis	

Summary

→ Categorization of state-of-the-art methods

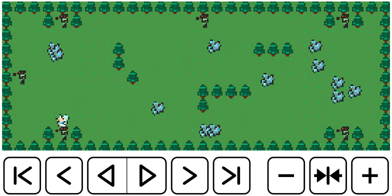
→ Analysis of game model characteristics

→ Game models can be learned using supervised learning

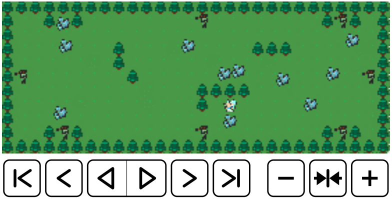
→ The proposed model is able to outperform state-of-the-art methods

Prerequisites	Game Model Characteristics	✓
	Approximation of a Game Model	✓
Algorithm	Applying Model to Search	✓
	Risk Measures	
Evaluation	Performance Comparison	(✓)
	Applicability Analysis	

State-of-the-Art



Proposed Solution



Thank you for your attention!

Alexander Dockhorn
dockhorn@ovgu.de
Otto-von-Guericke University of Magdeburg
Faculty of Computer Science
Institute for Intelligent Cooperating Systems

My Related Publications I/II

Book Chapters

Alexander Dockhorn, Chris Saxton, and Rudolf Kruse; *Association Rule Mining for Unknown Video Games*, A fuzzy dictionary of fuzzy modelling. Common concepts and perspectives (tentative title), accepted, 2019

Conference Papers

Alexander Dockhorn, Tim Tippelt, and Rudolf Kruse; *Model Decomposition for Forward Model Approximation*, IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, November 2018, pp. 1751–1757

Alexander Dockhorn and Daan Apeldoorn; *Forward Model Approximation for General Video Game Learning*, 2018 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2018, pp. 425-432

Alexander Dockhorn, Max Frick, Ünal Akkaya, and Rudolf Kruse; *Predicting Opponent Moves for Improving Hearthstone AI*, 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU), Springer International Publishing, May 2018, pp. 621-632

My Related Publications II/II

Conference Papers

Alexander Dockhorn, Christoph Doell, Matthias Hewelt, and Rudolf Kruse; *A decision heuristic for Monte Carlo tree search doppelkopf agents*, IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, November 2017, pp. 51-58

Alexander Dockhorn and Rudolf Kruse; *Combining cooperative and adversarial coevolution in the context of pac-man*, 2017 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2017, pp. 60-67

Workshop Papers

Alexander Dockhorn and Rudolf Kruse; *Detecting Sensor Dependencies for Building Complementary Model Ensembles*, 28. Workshop Computational Intelligence, KIT Publishing, November 2018, pp. 217-233

Further References

Yannakakis, Georgios N., and Julian Togelius; *Artificial Intelligence and Games*, Springer, 2018, http://gameaibook.org/

Ilhan, E., & Etaner-Uyar, A. S.; *Monte Carlo tree search with temporal-difference learning for general video game playing*, 2017 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, August 2017, pp. 317–324

Perez-Liebana, D., Liu, J., Khalifa, A., Gaina, R. D., Togelius, J., & Lucas, S. M.; *General Video Game AI: a Multi-Track Framework for Evaluating Agents*, Games and Content Generation Algorithms, 2018, http://arxiv.org/abs/1802.10363

Gaina, R. D., Lucas, S. M., & Perez-Liebana, D.; *Rolling horizon evolution enhancements in general video game playing*, In 2017 IEEE Conference on Computational Intelligence and Games, CIG 2017, pp. 88–95

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., . . . Colton, S. (2012); *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, 4(1), 2012, pp 1–43

Appendix (Appendix)

Alexander Dockhorn
dockhorn@ovgu.de
Otto-von-Guericke University of Magdeburg
Faculty of Computer Science
Institute for Intelligent Cooperating Systems

Planning

Planning

November - January 2019: planning missing experiments

January 2019: Doktorandentag

February - April 2019: theoretic generalisation of simulation-based search without partial information, finishing the remaining experiments

May - August 2019: finished first draft

September - November 2019: necessary changes and adjustments

December 2019: Submission of the thesis

State	Nov 18	Dez 18	Jan 19	Feb 19	Mar 19	Apr 19	May 19	Jun 19	Jul 19	Aug 19	Sep 19	Oct 19	Nov 19	Dec 19
Thesis Proposal														
Writing Contents														
Doktorandentag														
Remaining Experiments														
First Complete Draft														
Minor Changes														
Submission														

Risks

Exhaustive Experiments take a long time

- many agent configurations need to be tested
- many scenarios need to be evaluated

Risks

Exhaustive Experiments take a long time

- many agent configurations need to be tested
- many scenarios need to be evaluated

No objective target value available

- the developed framework can only be compared against other agents, which were published by their authors
- availability of comparable agents is limited

Risks

Exhaustive Experiments take a long time

- many agent configurations need to be tested
- many scenarios need to be evaluated

No objective target value available

- the developed framework can only be compared against other agents, which were published by their authors
- availability of comparable agents is limited

The framework consists of multiple exchangable components

- the influence on the performance needs to be evaluated on each of them, no single best component (no free lunch theorem)

General Game Playing/Learning

Why is Game-AI Research so Popular?

Games let us engage in a wide range of tasks.

- studying developed agents in multiple scenarios

Why is Game-AI Research so Popular?

Games let us engage in a wide range of tasks.

- studying developed agents in multiple scenarios

Games are (hopefully) well balanced by design.

- making the comparison of concurring algorithms feasible

Why is Game-AI Research so Popular?

Games let us engage in a wide range of tasks.

- studying developed agents in multiple scenarios

Games are (hopefully) well balanced by design.

- making the comparison of concurring algorithms feasible

Games are hard because if their often vast finite state spaces.

- breadth and height of the search tree is often huge

Why is Game-AI Research so Popular?

Games let us engage in a wide range of tasks.

- studying developed agents in multiple scenarios

Games are (hopefully) well balanced by design.

- making the comparison of concurring algorithms feasible

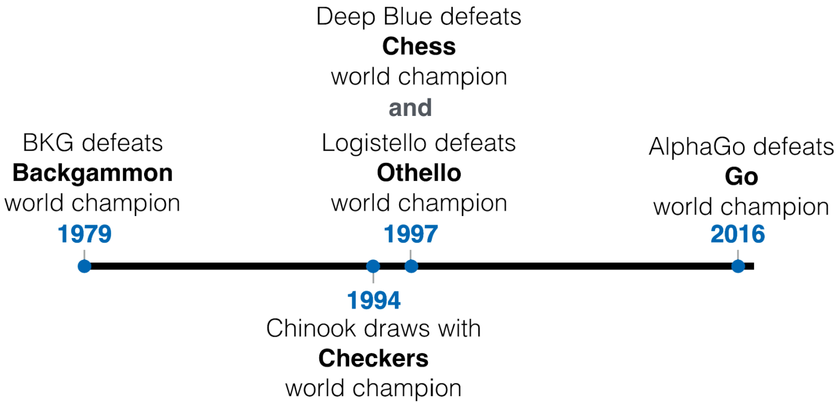
Games are hard because if their often vast finite state spaces.

- breadth and height of the search tree is often huge

Games are popular!

- more players, more games, more data

What did AI in Games Achieve?



What did AI in Games Achieve?

BKG defeats **Backgammon** world champion **1979**

1994 Chinook draws with **Checkers** world champion

Deep Blue defeats **Chess** world champion **and** Logistello defeats **Othello** world champion **1997**

AlphaGo defeats **Go** world champion **2016**

All these solutions are based on search algorithms.

Why doing Research on Game-AI?

Games let us engage in a wide range of tasks.

- developing agents in multiple scenarios
- often modelling real life scenarios
- solutions can be applied to all sorts of areas: robotics, finance, etc.

Why doing Research on Game-AI?

Games let us engage in a wide range of tasks.

- developing agents in multiple scenarios
- often modelling real life scenarios
- solutions can be applied to all sorts of areas: robotics, finance, etc.

Games are (hopefully) well balanced by design.

- making the comparison of concurring algorithms feasible

Why doing Research on Game-AI?

Games let us engage in a wide range of tasks.

- developing agents in multiple scenarios
- often modelling real life scenarios
- solutions can be applied to all sorts of areas: robotics, finance, etc.

Games are (hopefully) well balanced by design.

- making the comparison of concurring algorithms feasible

Games are hard because of their often vast finite state spaces.

- breadth and height of the search tree is often huge

Why doing Research on Game-AI?

Games let us engage in a wide range of tasks.

- developing agents in multiple scenarios
- often modelling real life scenarios
- solutions can be applied to all sorts of areas: robotics, finance, etc.

Games are (hopefully) well balanced by design.

- making the comparison of concurring algorithms feasible

Games are hard because of their often vast finite state spaces.

- breadth and height of the search tree is often huge

Games are popular!

- more players, more games, more data

Human vs. Computer Game Playing

- humans can learn to play multiple games
- most AI agents focus on playing a single game

Human vs. Computer Game Playing

- humans can learn to play multiple games
- most AI agents focus on playing a single game

General Game Playing: ability to play multiple games

- Requirements: unified state representation, unified forward model
- game definition languages offer both

Human vs. Computer Game Playing

- humans can learn to play multiple games
- most AI agents focus on playing a single game

General Game Playing: ability to play multiple games

- Requirements: unified state representation, unified forward model
- game definition languages offer both

But is this the same task?

- humans can learn to play games by playing them
- Can computers do the same?

Human vs. Computer Game Playing

- humans can learn to play multiple games
- most AI agents focus on playing a single game

General Game Playing: ability to play multiple games

- Requirements: unified state representation, unified forward model
- game definition languages offer both

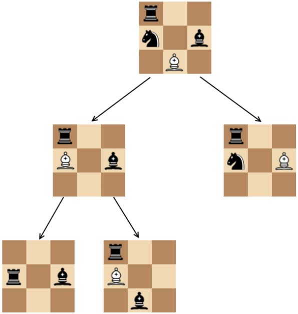
But is this the same task?

- humans can learn to play games by playing them
- Can computers do the same?

General Game Learning: ability to learn to play any game

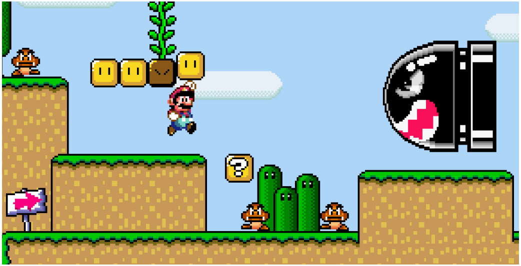
Action-Selection in Games

Action-Selection: search and select the best action among several alternative possibilities



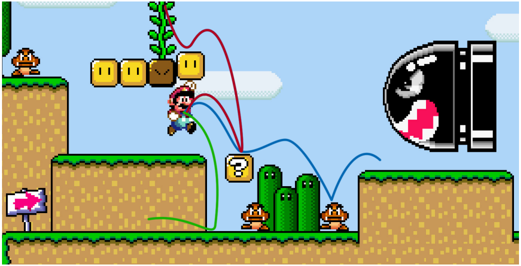
Action-Selection in Games

Action-Selection: search and select the best action among several alternative possibilities



Action-Selection in Games

Action-Selection: search and select the best action among several alternative possibilities



The Agent-Environment Interface

A general framework for studying games includes the following elements:

1. **Agent:** the learner and decision-maker

The Agent-Environment Interface

A general framework for studying games includes the following elements:

1. **Agent:** the learner and decision-maker
2. **Environment:** Anything the agent interacts with-everything outside the agent.

The Agent-Environment Interface

A general framework for studying games includes the following elements:

1. **Agent:** the learner and decision-maker
2. **Environment:** Anything the agent interacts with-everything outside the agent.
3. **Actions:** Agent and environment interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent.

The Agent-Environment Interface

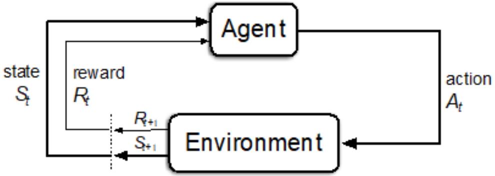
A general framework for studying games includes the following elements:

1. **Agent:** the learner and decision-maker
2. **Environment:** Anything the agent interacts with-everything outside the agent.
3. **Actions:** Agent and environment interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent.
4. **Reward:** Special numerical values provided by the environment that the agent tries to maximize over time.

The Agent-Environment Interface

A general framework for studying games includes the following elements:

1. **Agent:** the learner and decision-maker
2. **Environment:** Anything the agent interacts with-everything outside the agent.
3. **Actions:** Agent and environment interact continually, the agent selecting actions and the environment responding to those actions and presenting new situations to the agent.
4. **Reward:** Special numerical values provided by the environment that the agent tries to maximize over time.



General Video Game AI

General Video Game AI (GVGAI) Competition

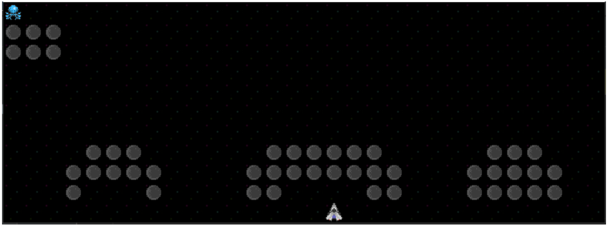
Competition framework including more than 100 games using Video Game Definition Language (VGDL) providing:

- general rules of a game and its forward model
- multiple levels per game
- visual representation

Example game: Butterflies

Game 1 - Aliens

- Destroy all aliens by shooting them
- Don't get shot or destroyed by touching an alien
- Aliens move from left to right and change direction at the end



Game 2 - Boulderdash

- Dig your way to the diamonds
- Don't get crushed by a boulder or touched by an enemy
- Leave through the door

Game 3 - Butterflies

- **Fairy:** the agent's representation in game, moved via actions
- **Butterfly:** moves randomly, removed when touched by the fairy
- **Cocoon:** static, becomes a butterfly when touched by a butterfly
- **Tree:** static, obstacle hindering movement other objects

Player actions: 4 way movement, moves the fairy in a direction
Win-Condition: no butterflies left, **Lose-Condition:** no cocoons left

Game 4 - Chase

- White birds fly away from you
- Catch all birds by cornering them

Game 5 - Frogs

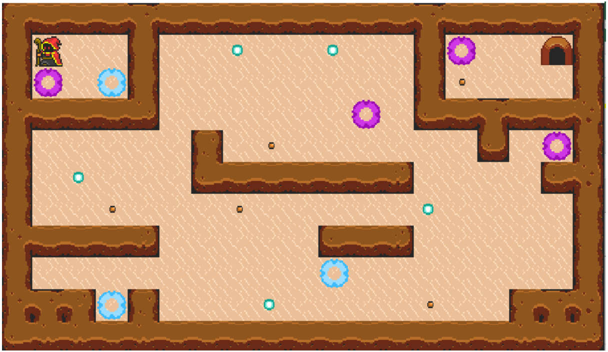
- Get to the goal post at the top
- . . . without getting hit by a car
- . . . and don't jump into the water

Game 6 - Missile Command

- Destroy all fireballs (comets) before they hit the city

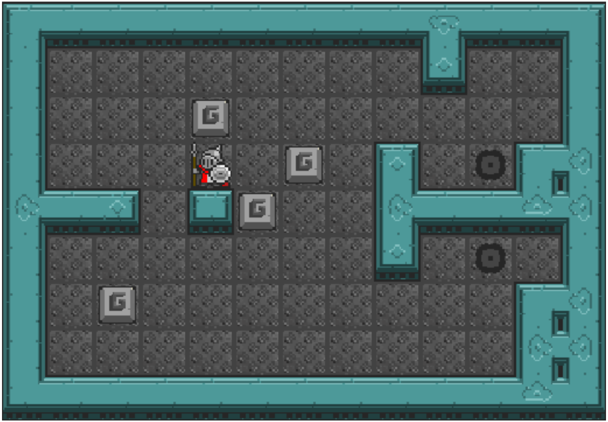
Game 7 - Portals

- Find the door to win the level



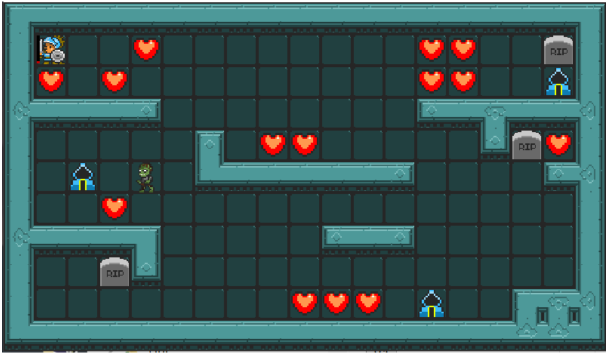
Game 8 - Sokoban

- Move the boulders to a hole in the ground



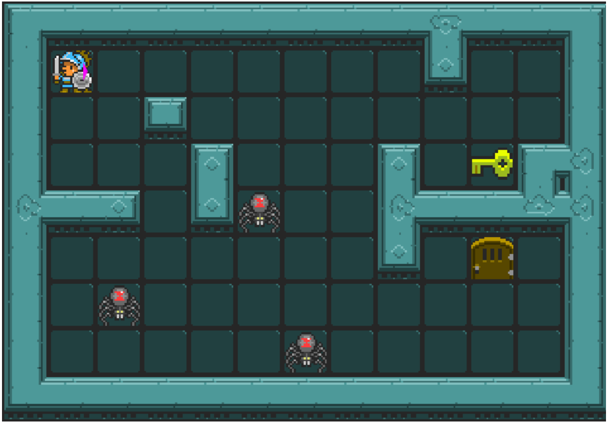
Game 9 - Survive Zombies

- Kill zombies and survive as long as possible
- Get healed by collecting hearts
- Stop magicians from getting killed



Game 10 - Zelda

- Kill the enemies
- Collect the key
- Flee through the door



Forward Model Learning

Forward Models

Using a forward model we can predict the outcome of an action.

- defined by the rules of the game being played

Search algorithms use the forward model to traverse the state-space

- Actions (edges of the game tree) are chosen based on their outcome

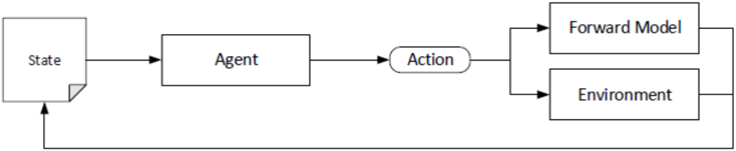
Forward Models

Using a forward model we can predict the outcome of an action.

- defined by the rules of the game being played

Search algorithms use the forward model to traverse the state-space

- Actions (edges of the game tree) are chosen based on their outcome



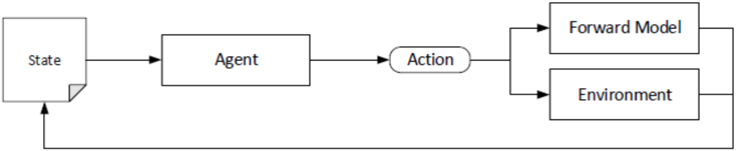
Forward Models

Using a forward model we can predict the outcome of an action.

- defined by the rules of the game being played

Search algorithms use the forward model to traverse the state-space

- Actions (edges of the game tree) are chosen based on their outcome



Examples

- **Robotics:** activating a motor to adjust the leg
- **Gameplaying Chess:** taking the king wins the game

FMA vs RL: Concept

Reinforcement Learning

- estimates the value of an action based on previous data
- determines the best action (sequence) using its expected value

FMA vs RL: Concept

Reinforcement Learning

- estimates the value of an action based on previous data
- determines the best action (sequence) using its expected value

Forward Model Approximation

- estimates the outcome of an action based on previous data
- determines an action (sequence) and their expected future states
- searches for the win-condition

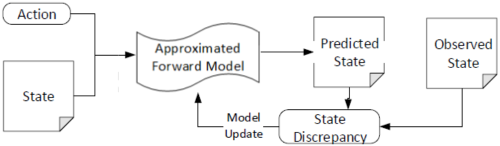
FMA vs RL: Concept

Reinforcement Learning

- estimates the value of an action based on previous data
- determines the best action (sequence) using its expected value

Forward Model Approximation

- estimates the outcome of an action based on previous data
- determines an action (sequence) and their expected future states
- searches for the win-condition



FMA vs RL: Learning Target

Reinforcement Learning

- learning target: $\mathbb{E}_\pi[G_t | S_t = s]$

Forward Model Approximation

- learning target: $\mathbb{E}_\pi[S_{t+1} | S_t = s]$ or $\mathbb{E}_\pi \delta S_t, S_{t+1}$

FMA vs RL: Learning Target

Reinforcement Learning

- learning target: $\mathbb{E}_\pi[G_t|S_t = s]$
- simulation-based search is widely applied

Forward Model Approximation

- learning target: $\mathbb{E}_\pi[S_{t+1}|S_t = s]$ or $\mathbb{E}_\pi \delta S_t, S_{t+1}$
- reinforcement learning and heuristics are widely applied

FMA vs RL: Learning Target

Reinforcement Learning

- learning target: $\mathbb{E}_\pi[G_t | S_t = s]$
- simulation-based search is widely applied
- algorithm performance close to or better than human

Forward Model Approximation

- learning target: $\mathbb{E}_\pi[S_{t+1} | S_t = s]$ or $\mathbb{E}_\pi \delta S_t, S_{t+1}$
- reinforcement learning and heuristics are widely applied

FMA vs RL: Learning Target

Reinforcement Learning

- learning target: $\mathbb{E}_\pi[G_t|S_t = s]$
- simulation-based search is widely applied
- algorithm performance close to or better than human

Forward Model Approximation

- learning target: $\mathbb{E}_\pi[S_{t+1}|S_t = s]$ or $\mathbb{E}_\pi\delta S_t, S_{t+1}$
- reinforcement learning and heuristics are widely applied
- algorithm performance on par with random decision-making

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search

-
-
-

Reinforcement Learning

-
-
-

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search	Reinforcement Learning
▶ searching at run-time	▶
▶	▶
▶	▶

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search

- searching at run-time
- estimates the value of an action based on simulations
-

Reinforcement Learning

-
-
-

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search	Reinforcement Learning
▶ searching at run-time	▶
▶ estimates the value of an action based on simulations	▶
▶ needs a model of the game and its current state	▶

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search	Reinforcement Learning
searching at run-time	training prior evaluation
estimates the value of an action based on simulations	
needs a model of the game and its current state	

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search	Reinforcement Learning
searching at run-time	training prior evaluation
estimates the value of an action based on simulations	estimates the value of an action based on previous data
needs a model of the game and its current state	

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search

- searching at run-time
- estimates the value of an action based on simulations
- needs a model of the game and its current state

Reinforcement Learning

- training prior evaluation
- estimates the value of an action based on previous data
- needs huge amounts of data (e.g. expert replays)

Comparison SBS vs. RL

Two popular algorithm classes for action-selection (in general games):

Simulation-Based Search

- searching at run-time
- estimates the value of an action based on simulations
- needs a model of the game and its current state

Reinforcement Learning

- training prior evaluation
- estimates the value of an action based on previous data
- needs huge amounts of data (e.g. expert replays)

Simulation-Based Search cannot be applied when the game's model is unknown

Problem Scenario 1) Unknown Forward Model

Differences of human and computer gameplaying:

- humans can learn to play multiple games
- most AI agents focus on playing a single game

Problem Scenario 1) Unknown Forward Model

Differences of human and computer gameplaying:

- humans can learn to play multiple games
- most AI agents focus on playing a single game

Next Step: General Game Playing

- Requirements: unified state representation, unified forward model
- game definition languages offer both

Problem Scenario 1) Unknown Forward Model

Differences of human and computer gameplaying:

- humans can learn to play multiple games
- most AI agents focus on playing a single game

Next Step: General Game Playing

- Requirements: unified state representation, unified forward model
- game definition languages offer both

But is this the same task?

- humans can learn to play games by playing them
- Can computers do the same?

Problem Scenario 1) Unknown Forward Model

Differences of human and computer gameplaying:

- humans can learn to play multiple games
- most AI agents focus on playing a single game

Next Step: General Game Playing

- Requirements: unified state representation, unified forward model
- game definition languages offer both

But is this the same task?

- humans can learn to play games by playing them
- Can computers do the same?

Next Step: General Game Learning

Proposal 1) Forward Model Approximation

Instead of learning a value function we try to learn a forward model.

- apply simulation-based search using the learned model

Proposal 1) Forward Model Approximation

Instead of learning a value function we try to learn a forward model.

▶ apply simulation-based search using the learned model

Idea:

In case the game fulfills the Markov Property the next state is only dependent on the observation of the current state and our action.

Classification task:

- map the current state and an action to the upcoming state

Proposal 1) Forward Model Approximation

Instead of learning a value function we try to learn a forward model.

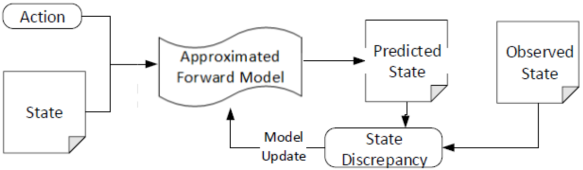
▶ apply simulation-based search using the learned model

Idea:

In case the game fulfills the Markov Property the next state is only dependent on the observation of the current state and our action.

Classification task:

- map the current state and an action to the upcoming state



Characteristics of the Forward Model

Model the state change as a single transition, considering the history of:

- previous game states,
- player actions,
- and rewards

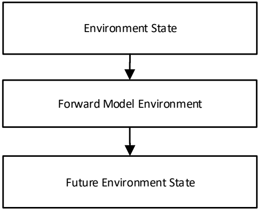
Characteristics of the Forward Model

Model the state change as a single transition, considering the history of:

- previous game states,
- player actions,
- and rewards

Markov property (order 1):

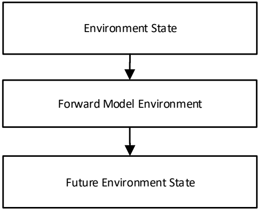
- only consider the last interaction



Characteristics of the Forward Model

Model the state change as a single transition, considering the history of:

- previous game states,
- player actions,
- and rewards



Markov property (order 1):

- only consider the last interaction

Problem:

- the state can be arbitrarily complex

Can we further reduce the number of possible models?

Composite Model using Background Knowledge

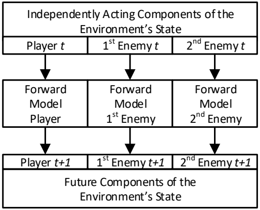
Inherent constraint of the VGDL:

- model components individually and combine the result model
- simple component models are combined to a complex game model

Composite Model using Background Knowledge

Inherent constraint of the VGDL:

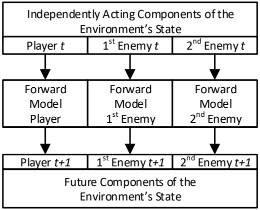
- model components individually and combine the result model
- simple component models are combined to a complex game model



Composite Model using Background Knowledge

Inherent constraint of the VGDL:

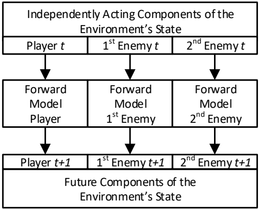
- model components individually and combine the result model
- simple component models are combined to a complex game model
- merge similar models

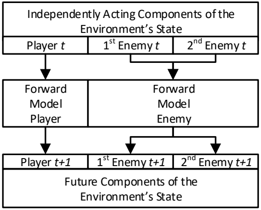


Composite Model using Background Knowledge

Inherent constraint of the VGDL:

- model components individually and combine the result model
- simple component models are combined to a complex game model
- merge similar models

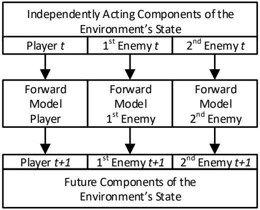




Composite Model using Background Knowledge

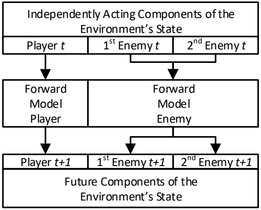
Inherent constraint of the VGDL:

- model components individually and combine the result model
- simple component models are combined to a complex game model
- merge similar models



Idea:

- reduce the number of considered sensors to an interesting subset
- predict the change of all components/sensors individually



Composite Models

Benefits

- reduces size of generated models

Composite Models

Benefits

- reduces size of generated models
- speeds up induction process

Composite Models

Benefits

- reduces size of generated models
- speeds up induction process
- reduces training time and amount of necessary training data

Composite Models

Benefits

- reduces size of generated models
- speeds up induction process
- reduces training time and amount of necessary training data

- drastically improves performance of Forward Model Approximation

Composite Models

Benefits

- reduces size of generated models
- speeds up induction process
- reduces training time and amount of necessary training data

- drastically improves performance of Forward Model Approximation

Problem

Composite Models

Benefits

- reduces size of generated models
- speeds up induction process
- reduces training time and amount of necessary training data

- drastically improves performance of Forward Model Approximation

Problem

- **needs background knowledge**

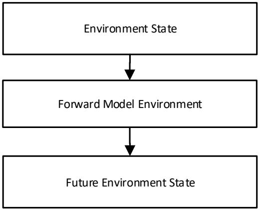
Finding a Composite Model

Can we somehow build composite models?

Finding a Composite Model

Can we somehow build composite models?

Modelling the complete state transition is very complex for a lot of applications

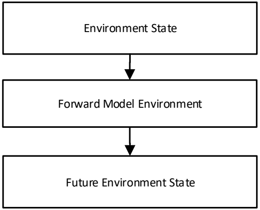


Finding a Composite Model

Can we somehow build composite models?

Modelling the complete state transition is very complex for a lot of applications

- ... because of a lot of variables
- ... that are dependent on each other

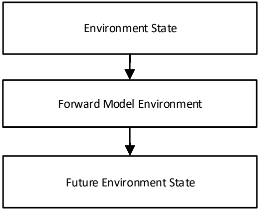


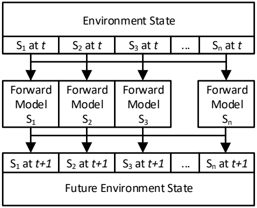
Finding a Composite Model

Can we somehow build composite models?

Modelling the complete state transition is very complex for a lot of applications

- … because of a lot of variables
- … that are dependent on each other





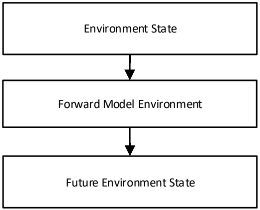
Finding a Composite Model

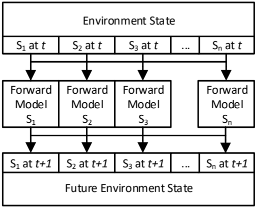
Can we somehow build composite models?

Modelling the complete state transition is very complex for a lot of applications

- ... because of a lot of variables
- ... that are dependent on each other

Use dependency analysis to find groups of dependent variables.





Finding a Composite Model

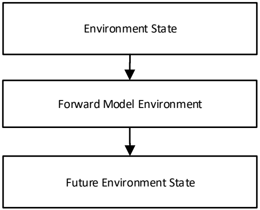
Can we somehow build composite models?

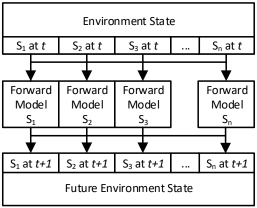
Modelling the complete state transition is very complex for a lot of applications

- … because of a lot of variables
- … that are dependent on each other

Use dependency analysis to find groups of dependent variables.

- filter unrelevant variables
- speeds up model building
- reduces noise in each submodel





Dependency Analysis

Dependency Analysis

Goal:

- predict the change of each sensor variable
- using a simple model

Dependency Analysis

Goal:

- predict the change of each sensor variable
- using a simple model

To build a simple model for a single variable

- we want to find all non-independent variables
- to describe its change

Dependency Analysis

Goal:

- predict the change of each sensor variable
- using a simple model

To build a simple model for a single variable

- we want to find all non-independent variables
- to describe its change

We can do this for all variables at once by learning a belief net structure.

- it encodes the necessary independencies for all involved nodes
- only variable that is known in advance is the players action

Dependency Analysis Algorithms

Scoring-based Structure Learning Algorithms

- rates candidate structures based on a goodness of fit measure
- greedy algorithms iteratively modify an existing structure

Dependency Analysis Algorithms

Scoring-based Structure Learning Algorithms

- rates candidate structures based on a goodness of fit measure
- greedy algorithms iteratively modify an existing structure

Constraint-based Structure Learning Algorithms

- use conditional independence tests to learn dependence structure
- learn local causal and markov blanket relationships

Dependency Analysis Algorithms

Scoring-based Structure Learning Algorithms

- rates candidate structures based on a goodness of fit measure
- greedy algorithms iteratively modify an existing structure

Constraint-based Structure Learning Algorithms

- use conditional independence tests to learn dependence structure
- learn local causal and markov blanket relationships

Hybrid Structure Learning Algorithms

- 2-phase algorithms combining restriction and maximation phase
- restriction phase: find undirected candidate structure
- maximation phase: find best configuration for all edge directions

Results - Model Decomposition I/II

Found substructures contain:

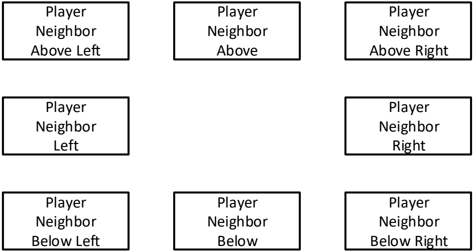
- position changes

- explanations for an component's death

Results - Model Decomposition II/II

Some patterns evolve over a lot of timesteps

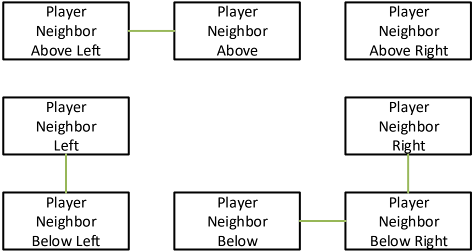
- e.g. neighboring relations
- starting without any connection



Results - Model Decomposition II/II

Some patterns evolve over a lot of timesteps

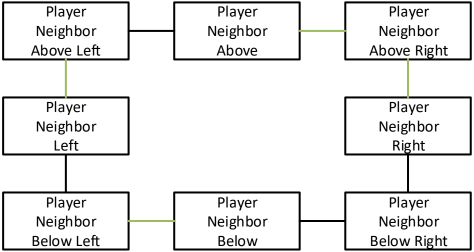
- e.g. neighboring relations
- starting without any connection
- adding edges dependencies over time



Results - Model Decomposition II/II

Some patterns evolve over a lot of timesteps

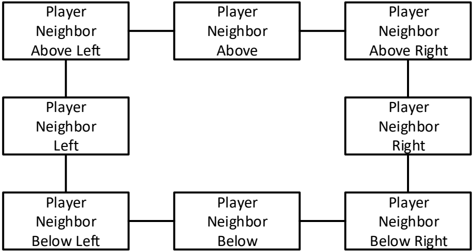
- e.g. neighboring relations
- starting without any connection
- adding edges dependencies over time



Results - Model Decomposition II/II

Some patterns evolve over a lot of timesteps

- e.g. neighboring relations
- starting without any connection
- adding edges dependencies over time
- and hopefully converge



Open Research Questions Decomposition

Can we merge similar sub-structures?

- multiple game components can have the same behavior
- e.g. instances of the same character type (butterflies)

Open Research Questions Decomposition

Can we merge similar sub-structures?

- multiple game components can have the same behavior
- e.g. instances of the same character type (butterflies)

What happens if the (1st order) Markov Property is not fulfilled?

- The number of variables to consider grows exponentially with the number of time-steps to consider.
- Theoretically the system would work, but. . .
- . . . the amount of data for model building grows with increasing the order

Open Research Questions Decomposition

Can we merge similar sub-structures?

- multiple game components can have the same behavior
- e.g. instances of the same character type (butterflies)

What happens if the (1st order) Markov Property is not fulfilled?

- The number of variables to consider grows exponentially with the number of time-steps to consider.
- Theoretically the system would work, but...
- ...the amount of data for model building grows with increasing the order

How good is Forward Model Approximation if the approximation does not completly fit the distribution?

- it depends!
- some games can be played even without "understanding" all

Imperfect Information Games

Scenario 2) Partially Unknown Game State

Simulation-based search can be applied to full information games.

- What to do in partial information games?

Scenario 2) Partially Unknown Game State

Simulation-based search can be applied to full information games.

- What to do in partial information games?

Each sensor value provides information on the current state of the game.

- the actual gamestate is unknown

Scenario 2) Partially Unknown Game State

Simulation-based search can be applied to full information games.

- What to do in partial information games?

Each sensor value provides information on the current state of the game.

- the actual gamestate is unknown

Popular example: simple card games like Uno, MauMau

- our own hand cards are known
- the opponent's cards are hidden
- the card draw is random

Proposal 2) State Induction

Use information from replays to guess a probable state during run-time.

- apply simulation-based search with the estimated state

Proposal 2) State Induction

Use information from replays to guess a probable state during run-time.

▶ apply simulation-based search with the estimated state

Idea:

Can observable sensor variables be used to guess the gamestate?

Dependency analysis:

- find dependencies between observable and unobservable variables

Proposal 2) State Induction

Use information from replays to guess a probable state during run-time.

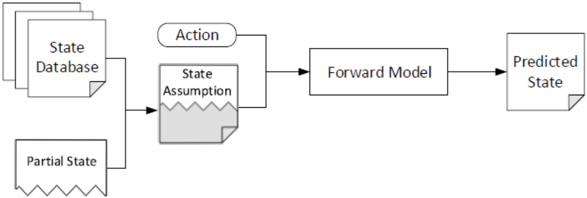
▶ apply simulation-based search with the estimated state

Idea:

Can observable sensor variables be used to guess the gamestate?

Dependency analysis:

- find dependencies between observable and unobservable variables



General Video Game AI (GVGAI) Competition

Competition framework including more than 100 games using Video Game Definition Language providing :

- general rules of a game and its forward model
- multiple levels per game
- visual representation

Example game: Butterflies

General Video Game AI (GVGAI) Competition

Competition framework including more than 100 games using Video Game Definition Language providing :

- general rules of a game and its forward model
- multiple levels per game
- visual representation

Example game: Butterflies

Comparison of GVGAI Tracks

Planning Track

- forward model available,
  10 trials for training

Learning Track

- no forward model,
  5 minutes of training

Comparison of GVGAI Tracks

Planning Track

- forward model available, 10 trials for training
- simulation-based search is widely applied

Learning Track

- no forward model, 5 minutes of training
- reinforcement learning and heuristics are widely applied

Comparison of GVGAI Tracks

Planning Track

- forward model available, 10 trials for training
- simulation-based search is widely applied
- algorithm performance close to or better than human

Learning Track

- no forward model, 5 minutes of training
- reinforcement learning and heuristics are widely applied

Comparison of GVGAI Tracks

Planning Track

- forward model available, 10 trials for training
- simulation-based search is widely applied
- algorithm performance close to or better than human

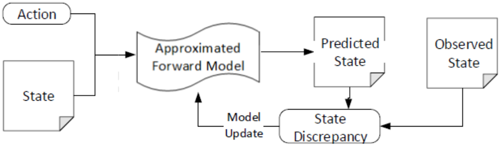
Learning Track

- no forward model, 5 minutes of training
- reinforcement learning and heuristics are widely applied
- algorithm performance on par with random decision-making

Generating a Data Set

Forward Model Approximation is an iterative framework

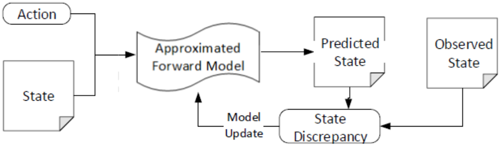
- our data set of observations grows over time
- model building needs to be repeated in case of errors



Generating a Data Set

Forward Model Approximation is an iterative framework

- our data set of observations grows over time
- model building needs to be repeated in case of errors



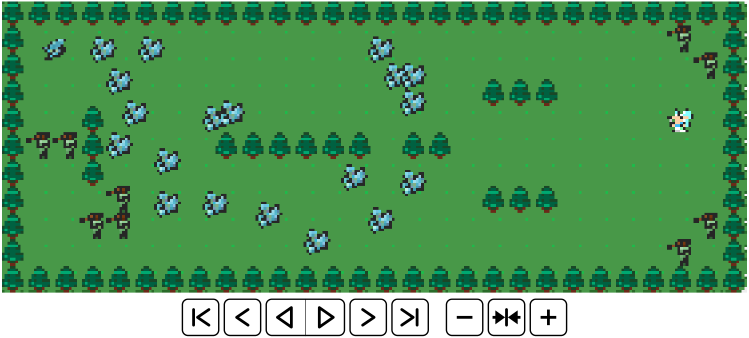
For testing purposes we collect a dataset using a random agent

- the player's actions will be independent from the game state

Each time frame we store all observable variables.

- contains an object's position, neighbors, movement and death

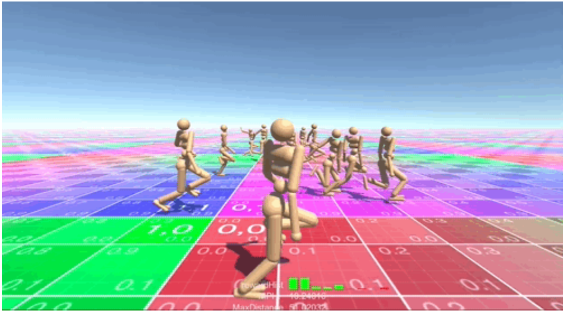
Demo Gameplaying FMA



Benchmarks

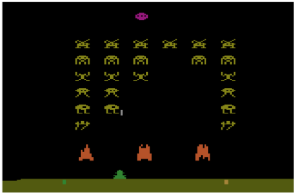
Marathon Environment

- high-dimensional continuous control environments
- $\longrightarrow$ complex game model



Arcade Learning Environment

- Atari 2600 games,
- screen capture: 160 pixel wide and 210 pixels high, 128-colours
- 18 actions: joystick direction and buttons



[1] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013); *The Arcade Learning Environment: An Evaluation Platform for General Agents*, Journal of Artificial Intelligence Research, 47, pp. 253–279

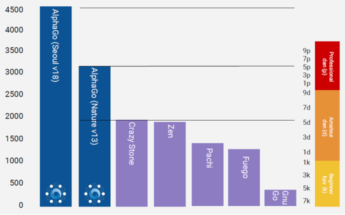
AlphaGO and AlphaZero

Playstrength of Computer Go Agents

Decreasing performance gain of computer agents in previous years

- Methods based on MCTS struggle with the extreme size of the search tree
- Playing Go on an expert level was thought to be unsolvable using known methods

Playstrength of agents in the last years:

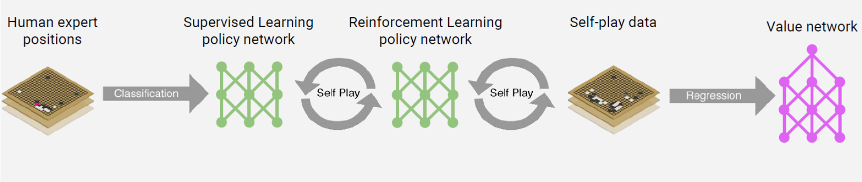


Alpha Go drastically changed the way in which we approach playing Go

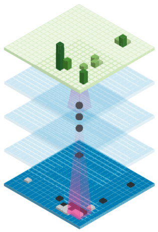
Alpha Go Learning Pipeline

Alpha Go combines the following ideas:

- Learning from Human expert data (gameplay recordings)
- Analysing board states on multiple levels of abstraction
- Predicting the probability of winning on any mid-gamestate
- Predicting valuable moves for better simulations



Policy Network



Try to predict the next best move depending on the current board state

- Classification problem: which move is likely to be played by an expert

Solution:

- Train two Convolutional Neural Networks

A small network was learned for fast rollouts, while a larger more accurate network was used during expansion.

The network receives an encoded input of 25 features per board position.

- Those features were first constructed by humans and automatically adapted in the learning phase.

Convolutional Neural Networks (CNN)

CNNs are a common network type in image processing:

- Feed-Forward Neural Networks
- Efficient processing using feature kernels
- Shift and rotation invariant (depending on the kernels)

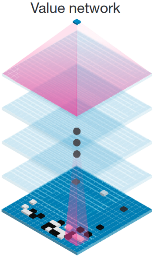
Each layer of a CNN has increasing complexity:

First Layer Representation Second Layer Representation Third Layer Representation

Value Network

Try to predict the probable outcome of the match

- Regression problem: what is the value of the current state

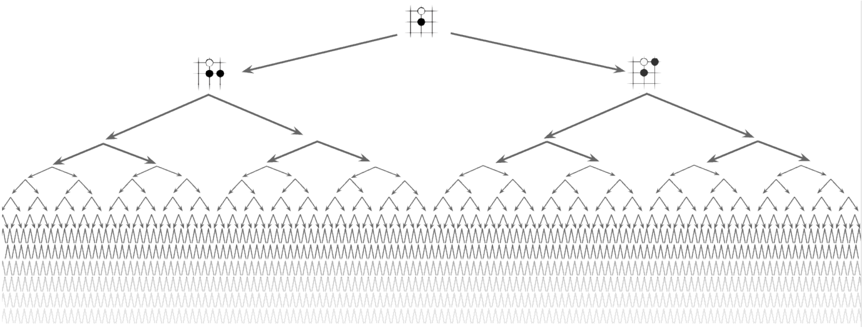


Solution:

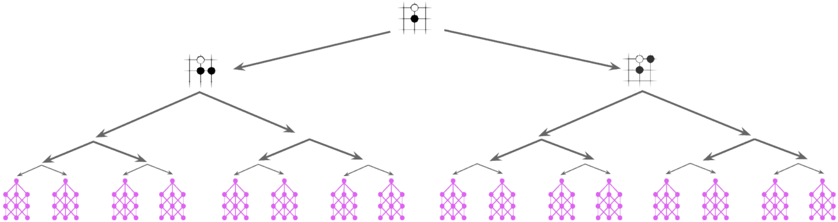
- Train another Convolutional Neural Network using Regression

The value network will be used to differ between good and bad states during the simulation.

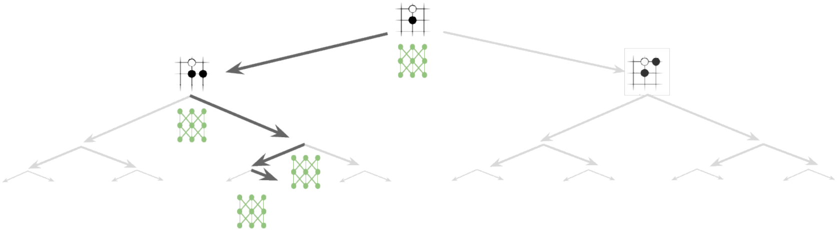
Exhaustive Search



Reducing Depth with the Value Network



Reducing Breadth with the Policy Network



Continuous Self-Improvement

With this approach machines can reach human expert performance

- How can we improve even further?

All networks were further trained by self-play:

- The networks in their current state were used to simulate 2 players in multiple games of Go.
- Those games produce a new database for further learning.

Based on the huge success of training based on Reinforcement Learning Alpha, Alpha Zero was developed

- no expert play training, just reinforcement learning
- faster adaptation, due to multiple improvements and various parallel processing adaptations
- top-performing agent in the games Go, Chess, and Shogi

Limiting Factors

Hardware (Training): the total Hardware costs to develop AlphaZero are about $25 million.

- it involves specialized Tensor Processing Units (TPU)
- the distributed version was using 1,202 CPUs and 176 GPUs

Hardware (End User): for each single position that AlphaGo analyzes, its neural network needs to do almost 20 billion operations

- many states need to be analyzed during the search
- either the user or the server needs to do these computations

Time: replicating the training done on a single machine would take about 1,700 years!

Paper Summaries

Model Decomposition for Forward Model Approximation

Model Building Requirements:

- **Learning Speed:** GVGAI-framework limits learning time to a total of 5 minutes
- **Processing Speed:** Monte Carlo Tree Search benefits from many simulations, applying the model needs to be fast
- **Generalization:** use the model for the prediction of unseen situations or levels
- **Accuracy:** high prediction accuracy for repeated application of the model
- **Interpretability:** beneficial to have an interpretable model

Model Decomposition Accuracy I/II

	Mean Accuracy (unpruned / pruned)				
	G1	G2	G3	G4	G5
1x training level 2	**0.970 / 0.970**	**0.998 / 0.998**	**0.980** / 0.962	0.996 / **0.997**	0.466 / **0.992**
1x training level 3	0.991 / **0.998**	**0.999** / 0.998	0.933 / **0.947**	**0.997** / 0.995	**0.975 / 0.975**
1x validation level 1	**0.806** / 0.753	**0.636** / 0.616	**0.664** / 0.663	0.667 / **0.704**	0.700 / **0.803**
3x validation level 1	**0.773** / 0.663	0.616 / **0.621**	0.666 / **0.718**	**0.850** / 0.776	0.736 / **0.804**
5x validation level 1	**0.717** / 0.677	**0.593** / 0.583	**0.665 / 0.665**	**0.811** / 0.801	0.670 / **0.804**
1x validation level 2	0.707 / **0.708**	0.595 / **0.561**	**0.666 / 0.666**	0.645 / **0.667**	**0.667 / 0.667**
3x validation level 2	0.730 / **0.742**	**0.591** / 0.508	**0.666** / 0.665	0.671 / **0.692**	**0.667 / 0.667**
5x validation level 2	0.647 / **0.780**	**0.614** / 0.581	**0.744** / 0.723	0.667 / **0.671**	**0.667 / 0.667**

Model Decomposition Accuracy II/II

	Mean Accuracy (unpruned / pruned)				
	G6	G7	G8	G9	G10
1x training level 2	0.990 / **0.991**	**0.998 / 0.998**	0.994 / **1.000**	**0.996** / 0.995	**0.987 / 0.987**
1x training level 3	**0.996 / 0.996**	**0.998** / 0.997	**1.000 / 1.000**	**0.995 / 0.995**	**0.998** / 0.997
1x validation level 1	0.635 / **0.643**	**0.980** / 0.936	**0.667 / 0.667**	**0.670** / 0.604	**0.688** / 0.674
3x validation level 1	**0.654** / 0.647	**0.960** / 0.786	0.667 / **0.727**	0.648 / **0.687**	**0.783** / 0.721
5x validation level 1	0.640 / **0.667**	0.667 / **0.950**	**0.941** / 0.733	**0.714** / 0.679	**0.752** / 0.726
1x validation level 2	0.420 / **0.539**	0.773 / **0.798**	**0.667** / 0.633	0.692 / **0.707**	0.683 / **0.724**
3x validation level 2	**0.903** / 0.658	**0.837** / 0.769	**0.667** / 0.542	**0.749** / 0.721	**0.773** / 0.733
5x validation level 2	0.664 / **0.659**	**0.667 / 0.667**	0.667 / **0.733**	**0.699** / 0.646	0.735 / **0.751**

Association Rule Mining for Unknown Video Games

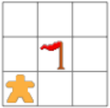
Procedure:

- Repeatedly play a level
  - store two separate interaction datasets: termination interaction and continuouing interactions
- Separately learn rules on both datasets
- Filter rules of the termination set
  - Remove all rules that conflict with rules from the continuouing interaction set

Example Game

(a) Level 1

(b) Level 2



(c) Level 3

Example Rules I/II

1. Moving right wins the game.
2. Moving left wins the game.
3. Reaching the flag wins the game.
4. Moving right and reaching the flag wins the game.
5. Moving left and reaching the flag wins the game.

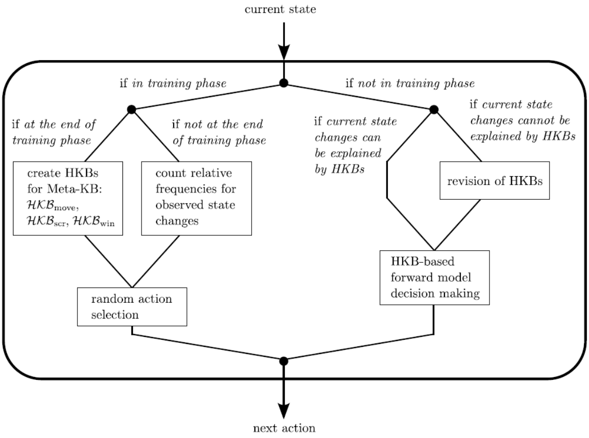
1. Reaching the flag wins the game.
2. Reaching the flag loses the game
3. Moving right and reaching the flag wins the game.
4. Moving left and reaching the flag wins the game.
5. Moving up and reaching the flag loses the game.
6. Moving down and reaching the flag loses the game.

Association Rules Results

Table 2: Association rules extracted from play-trace analysis.

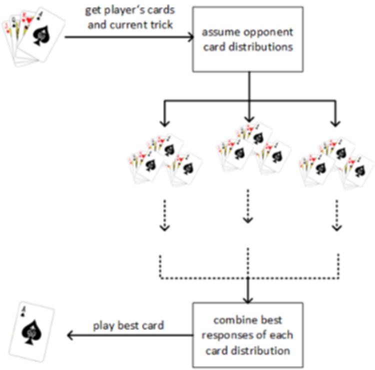
Rule			Support	Confidence
ActionLeft	→	MoveLeft	0.32	1.00
ActionRight	→	MoveRight	0.31	0.98
ActionUse	→	Stay	0.36	1.00
ObjectAbove, ActionUse	→	Stay	0.05	1.00
ObjectAbove, ActionLeft	→	MoveLeft	0.05	1.00
ObjectAbove, ActionRight	→	MoveRight	0.06	0.92
SpriteCollision	↔	HigherScore	0.06	1.00
PlayerCollision, ScoreDecrease	→	GameLost	1.00	1.00

Forward Model Approximation for General Video Game Learning



Predicting Opponent Moves for Improving Hearthstone AI

In case the current state is not known start multiple searches and combine their result.

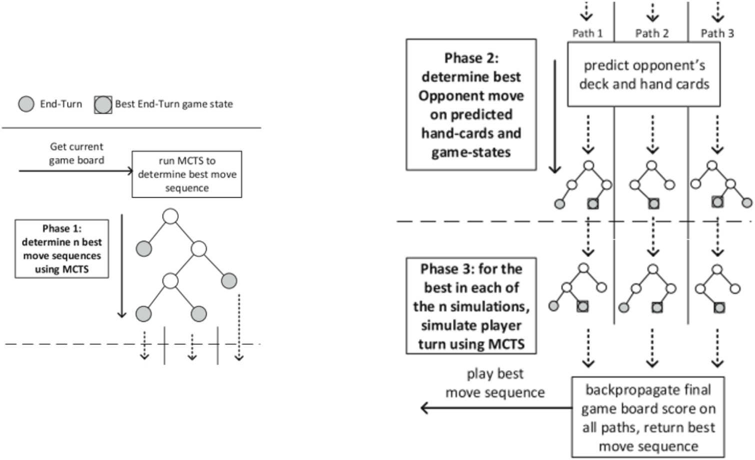


Monte Carlo Tree Search Ensemble I/II

Procdure

- A set of hand-cards is determined using the bi-gram database and all previously seen cards
  - determine the number of co-occurences of each card
  - filter possible cards by the rules of the deckbuilding process
  - sample the opponent's hand cards based on the normalized co-occurence values
- Generate multiple set of opponent's hand cards and run MCTS on each of them
- Use (weighted) majority vote to determine the best card to be played

Monte Carlo Tree Search Ensemble II/II



Monte Carlo Tree Search Ensemble Results

- We tested our agent against multiple other agents playing multiple decks
  - Random = randomly choose the next action
  - flatMC = flat Monte Carlo algorithm, simulate n times for each action
  - plainMCTS = MCTS using a randomly guessed game states
  - foMCTS = MCTS using the true game state (cheating)
  - Exh.s. = exhaustive search for best action, does not consider the opponent

Wins in %	Aggro	Mid	Control
Random	95	100	100
flatMC	81	73	94
plainMCTS	59	47	58
foMCTS	46	36	60
exh.s.	65	47	70

(a) predMCTS Aggro Deck

Wins in %	Aggro	Mid	Control
Random	99	98	100
flatMC	88	85	99
plainMCTS	71	55	76
foMCTS	59	50	76
exh.s.	62	70	85

(b) predMCTS Mid-Range Deck

Wins in %	Aggro	Mid	Control
Random	97	97	100
flatMC	73	54	89
plainMCTS	36	31	68
foMCTS	41	16	51
exh.s.	45	20	61

(c) predMCTS Control Deck